COST-EFFECTIVE AND PRIVACY-CONSCIOUS CLOUD SERVICE PROVISIONING: ARCHITECTURES AND ALGORITHMS

A Thesis
Presented to
The Academic Faculty

by

Balaji Palanisamy

In Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy in the
School of Computer Science, College of Computing

Georgia Institute of Technology
August 2013

COST-EFFECTIVE AND PRIVACY-CONSCIOUS CLOUD SERVICE PROVISIONING: ARCHITECTURES AND ALGORITHMS

Approved by:

Professor Ling Liu, Advisor
School of Computer Science, College of Computing
Georgia Institute of Technology

Professor Edward Omiecinski
School of Computer Science, College of Computing
Georgia Institute of Technology

Professor Mustaque Ahamad
School of Computer Science, College of Computing
Georgia Institute of Technology

Professor Calton Pu
School of Computer Science, College of Computing
Georgia Institute of Technology

Dr. Greg Eisenhauer
School of Computer Science, College of Computing
Georgia Institute of Technology

Dr. Aameek Singh
IBM Research, Almaden

Professor Sham Navathe
School of Computer Science, College of Computing
Georgia Institute of Technology

Date Approved: 6 May 2013

To Mom, Dad and Chandru

ACKNOWLEDGEMENTS

This thesis would not have been possible without the generous advice, support and guidance of many individuals. I would like to express my sincere thanks to everyone who has contributed to the process leading to my dissertation.

First and foremost, I would like to thank my advisor, Prof. Ling Liu for her constant support and guidance throughout my Ph.D. study. Ling's incredible energy and tireless work ethic has been my biggest source of inspiration through all these years. The flexibility and freedom she has provided in deciding and evolving my research has not only helped me pursue my research interests but also helped me make the right career choices. I am deeply indebted for the countless hours she has spent on perfecting my work and for the many life lessons she taught me along the way. I feel extremely fortunate to have crossed paths with her and I hope to be able pass much of what I learned from her to my students in the future.

Next, I would like to express my thanks to my Ph.D. committee members, Prof. Mustaque Ahamad, Dr. Greg Eisenhauer, Prof. Sham Navathe, Prof. Edward Omiecinski, Prof. Calton Pu and Dr. Aameek Singh for their timely feedback and insightful comments that have greatly contributed to my thesis. I convey special thanks to Prof. Pu, Prof. Ahamad and Prof. Navathe for offering welcomed advice and encouraging me along the way. Their frequent discussions and generous advice have helped me avoid many pitfalls during this dissertation.

My three internships with IBM Research have been some of the most memorable experiences during my Ph.D. I express sincere thanks to my IBM managers, Dr. Aameek Singh and Mr. Sandeep Gopisetty, for their strong support and guidance. Aameek has been one of my finest mentors and played a significant role in shaping

my research skills. I owe Sandeep for his strong support and his excellent guidance during these years. My close collaboration with them has been not only a rewarding experience professionally but was truly a great pleasure. I extend thanks to my other IBM collaborators including Nagapramod Mandagere, Gabriel Alatorre, Bryan Langston and various other members of the Storage Optimization Services group including Ramani Routray, Divyesh Jadav and Eric Butler. I have truly enjoyed both my technical interaction with them as well as our interaction outside work including the several cricket games we played for the IBM cricket championship. My thanks to IBM will not be complete without mentioning their Fellowship support. I sincerely thank IBM Research for generously funding my Ph.D. with the IBM Ph.D. Fellowship for the year 2012-2013.

I owe a great debt to every member of the DiSL Research group for providing such a dynamic and friendly environment in the lab. My sincere thanks goes to Sangeetha, Bhuvan, Ting, Peter, Shicong, Sankaran, Myungcheol, Matt, Yuzhe, Kisung, Binh, Yang, Qi and many other past and current members of the group. I convey special thanks to Sangeetha Seshadri for her valuable advice and guidance during my Ph.D. I also thank the Database lab members Rocky, Kunal, De, Danesh and Qinyi for their great company in the lab during these years.

Outside of the lab, I am fortunate to have had the company of many wonderful friends in Atlanta. I thank Arun, Bharani, Sankaran, Dilip, Senthil, Ritvik, Sushmita, George, Deepthi, Mrinalini, Nagesh and Vishal for all the quality time we spent together.

I convey my thanks to my undergraduate advisor, Prof. N. Sreenath for his strong support and encouragement throughout this dissertation. Prof. Sreenath set high standards for me while constantly encouraging me to perform. I really appreciate his confidence in me and for supporting me all along.

Finally, I thank my parents, Vasantha and Palanisamy for their constant love and

blessings. My brother, Chandrasekar has been the greatest source of encouragement for me during my Ph.D. and I express my heartfelt thanks to him and my *anni* Uma Maheshwari for their uplifting encouragement and unconditional support. I thank my family from the bottom of my heart and in return, I dedicate this thesis to them. My success is also theirs.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Cloud Computing represents a recent paradigm shift that enables users to share and remotely access high-powered computing resources (both infrastructure and software/services) contained in off-site data centers thereby allowing a more efficient use of hardware and software infrastructures. This growing trend in cloud computing, combined with the demands for Big Data and Big Data analytics, is driving the rapid evolution of datacenter technologies towards more cost-effective, consumer-driven, more privacy conscious and technology agnostic solutions.

This dissertation is dedicated to taking a systematic approach to develop system-level techniques and algorithms to tackle the challenges of large-scale data processing in the Cloud and scaling and delivering privacy-aware services with anytime-anywhere availability. We analyze the key challenges in effective provisioning of Cloud services in the context of MapReduce-based parallel data processing considering the concerns of cost-effectiveness, performance guarantees and user-privacy and we develop a suite of solution techniques, architectures and models to support cost-optimized and privacy-preserving service provisioning in the Cloud.

At the Cloud resource provisioning layer, we develop a utility-driven MapReduce Cloud resource planning and management system called Cura for cost-optimally allocating resources to jobs. While existing services require users to select a number of complex cluster and job parameters and use those potentially sub-optimal per-job configurations, the Cura resource management achieves global resource optimization in the cloud by minimizing cost and maximizing resource utilization. We also address the challenges of resource management and job scheduling for large-scale parallel data

processing in the Cloud in the presence of networking and storage bottlenecks commonly experienced in Cloud data centers. We develop Purlieus, a self-configurable locality-based data and virtual machine management framework that enables MapReduce jobs to access their data either locally or from close-by nodes including all input, output and intermediate data achieving significant improvements in job response time.

We then extend our cloud resource management framework to support privacy-preserving data access and efficient privacy-conscious query processing. Concretely, we propose and implement VNCache: an efficient solution for MapReduce analysis of cloud-archived log data for privacy-conscious enterprises. Through a seamless data streaming and prefetching model in VNCache, Hadoop jobs begin execution as soon as they are launched without requiring any apriori downloading. At the cloud service delivery tier, we develop mix-zone based techniques for delivering anonymous cloud services to mobile users on the move through Mobimix, a novel road-network mix-zone based framework that enables real time, location based service delivery without disclosing content or location privacy of the consumers.

CHAPTER I

INTRODUCTION

Cloud computing and its pay-as-you-go cost structure have enabled hardware infrastructure service providers, platform service providers as well as software and application service providers to offer computing services on demand and pay per use just like how we use utility today. This growing trend in cloud computing, combined with the demands for Big Data and Big Data analytics, is driving the rapid evolution of datacenter technologies towards more cost-effective, more consumer-driven, more privacy conscious and technology agnostic solutions. Cost effective resource optimization techniques that are highly effective and yet greatly agile are critical for both cloud providers and cloud consumers. We identify three key challenges in effective provisioning and delivery of cloud services to a wide range of audiences. First, cloud service providers are faced with the challenges of offering cost-effective solutions to a much broader range of consumers than traditional data centers that are serving in-house brands. Yield management, once only popular in airline and hotel industry, becomes a critical factor in the equilibrium of the supply-demand and performance-cost trade-off decision making process. Second, the cloud service providers are facing the overwhelming challenge of meeting the service level agreements of diverse cloud consumer jobs in terms of performance demand, resource need and infrastructure and computing rental cost while minimizing the expenses of operating, maintenance and upgrade of their cloud services. This calls for highly effective and yet greatly agile resource management and capacity planning at the cloud provider end to handle workload variations, resource demand variations, as well as energy and maintenance cost variations, while minimizing the overall service provisioning and delivery cost.

Third but not the least, computing in the Cloud also demands for protection of data privacy, access privacy and execution privacy of the consumer jobs against any unauthorized data access and program execution, which can lead to unwanted privacy breaches.

This dissertation research is dedicated to addressing the above mentioned research challenges with the focus on parallel processing of large scale data using Map-Reduce and delivering privacy-conscious continuous data services with anytime-anywhere availability.

1.1 Technical Challenges

A cloud service allows enterprises to cost-effectively analyze large amounts of data without creating large infrastructures and parallel computing platforms of their own. Using virtual machines (VMs) and storage hosted by the cloud, enterprises can simply create virtual clusters to process and analyze their data.

One of the technologies that made large scale data processing and data analytics popular and accessible to enterprises of all sizes is MapReduce [57] (and its open-source Hadoop [4] implementation). With the ability to automatically parallelize the application on a cluster of commodity hardware, MapReduce allows enterprises to analyze terabytes and petabytes of data more conveniently than ever. A MapReduce job is comprised of two main components – a *map* function (Figure 1) that processes key/value pairs from input data to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key. A number of web applications lend themselves well to this model, for example, web search indexing [5], behavior analysis for ad recommendation systems and analyzing clickstream data [3]. A unique capability of this model is its ability to execute map and reduce tasks of a job on a distributed cluster of machines, transparently to the application programmer. The input data is split into blocks that

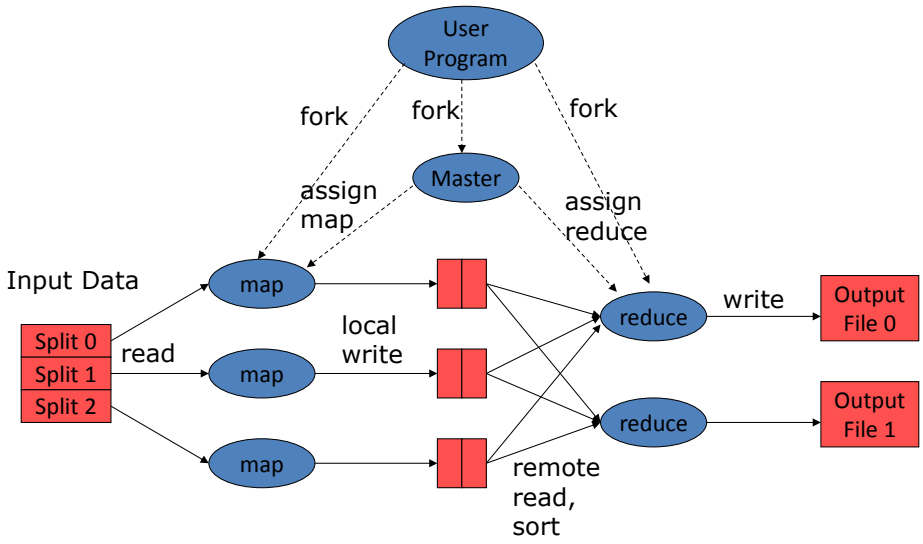


Figure 1: MapReduce Execution Overview

are stored in a distributed filesystem throughout the cluster. A cluster *master* then automatically schedules the *map* tasks at various worker nodes which process those blocks to create intermediate key/value pairs. Some of these blocks may be present locally on the worker node while others may require a *remote-read* to obtain from another node. The intermediate values, generated as the output of the map tasks are then scheduled to be processed by worker nodes to execute the *reduce* tasks which write the output into files stored within the filesystem. The data transfer from the map to the reduce tasks includes a *shuffle* phase in which reducers read data from all mappers.

A cloud service for such large scale data processing introduces several technical challenges in terms of efficient resource management, performance optimization and privacy. We discuss them as follows.

1.1.1 Cost-effectiveness

First, we argue that the existing cloud models for MapReduce are not cost-effective leading to poor resource utilization in the datacenters and results in higher costs for the customers. In general, there are two approaches in use today for MapReduce-based large scale data processing in a cloud. In the first approach, customers use a dedicated MapReduce cloud service (e.g. Amazon Elastic MapReduce [9]) and buy on-demand clusters of VMs for each job or a workflow. Once the MapReduce job (or workflow) is submitted, the cloud provider creates VMs that execute that job and after job completion the VMs are deprovisioned. In the second approach customers lease dedicated clusters from a generic cloud service like Amazon Elastic Compute Cloud [10] and operate MapReduce on them as if they were using a private MapReduce infrastructure. In this case, based on the type of analytics workload they may use different number of VMs for each submitted job, however the entire cluster needs to be maintained by the client enterprise.

Unfortunately, both of these models require users to figure out the complex job configuration parameters (e.g. type of VMs, number of VMs and MapReduce configuration like number of mappers per VM etc.) while simultaneously forcing the cloud provider to use those potentially sub-optimal configurations resulting in poor resource utilization and higher cost. We note that the per-job optimized configurations created from a user perspective are often suboptimal from a cloud provider perspective and hence lead to requiring more cloud resources than that required by a globally optimized schema with resource management performed at the cloud provider-end.

1.1.2 Datacenter Network Load

Another important challenge for the cloud provider is to manage the network resources for multiple virtual clusters executing concurrently, a diverse set of jobs on shared physical machines. Concretely, each MapReduce job generates different loads on the

shared physical infrastructure – (a) computation load: number and size of each VM (CPU, memory), (b) storage load: amount of input, output and intermediate data, and (c) network load: traffic generated during the map, shuffle and reduce phases. The network load is of special concern with MapReduce as large amounts of traffic can be generated in the shuffle phase when the output of map tasks is transferred to reduce tasks. As each reduce task needs to read the output of *all* map tasks [57], a sudden explosion of network traffic can significantly deteriorate cloud performance. This is especially true when data has to traverse greater number of network hops while going across *racks* of servers in the data center [24]. Further, the problem sometimes is exacerbated by TCP *incast* [109] with a recent study finding goodput of the network reduced by an order of magnitude for a MapReduce workload [42].

To reduce network traffic for MapReduce workloads, we argue for improved **data locality** for both Map and Reduce phases of the job. The goal is to reduce the network distance between storage and compute nodes for both map and reduce processing – for map phase, the VM executing the map task should be *close* to the node that stores the input data (preferably local to that node) and for reduce phase, the VMs executing reduce tasks should be close to the map-task VMs which generate the intermediate data used as reduce input.

1.1.3 Privacy-conscious Data access

Additionally, computing in the cloud is fundamentally challenged by the issues of data privacy, access privacy and execution privacy of consumer jobs against any unauthorized data access and program execution, which can lead to serious privacy breaches. As quite many data analysis operate on datasets that can be potentially sensitive and private, privacy-conscious enterprises may choose to keep only encrypted data in public clouds and decrypt to process them at their enterprise sites. In a cloud both data access needs to be private and service execution needs to be private.

Therefore it is important that the privacy-conscious data processing techniques are scalable and cost-effective.

Additionally, when services are requested from client's mobile devices, the location information of the mobile clients need to be private to avoid any location privacy breaches [49].

1.2 Dissertation focus and Contributions

In this dissertation, we focus on addressing the above mentioned challenges through global resource optimization, locality-based job scheduling, effective caching techniques and privacy aware service execution and delivery. This dissertation makes the following contributions.

1.2.1 Cost-optimized Cloud Model

The first contribution of this dissertation research is the development of a utility-driven cloud resource management system for MapReduce called Cura [106]. Cura determines how to best provision virtual MapReduce clusters for jobs submitted to the cloud by taking into consideration two sometimes conflicting requirements namely meeting the service quality requirements in terms of throughputs and response times on one hand while minimizing the overall consumption cost of the MapReduce cloud data center on the other hand. Cura enables cloud consumers to get the desired services at minimal cost while enabling cloud providers to maximize the utilization rate of their service infrastructure. In contrast to existing MapReduce cloud services, the resource management techniques in Cura aim at minimizing the overall resource utilization in the cloud as opposed to per-job or per-customer resource optimization in the existing services. Cura benefits from a number of novel performance enhancements including pre-created virtual machine pools for scheduling, cost-aware resource provisioning, VM-aware scheduling, intelligent capacity planning and online virtual machine reconfiguration. Concretely, the first approach in Cura performs a VM-aware

job scheduling using static virtual machine pools by utilizing pre-created virtual machines. The second approach of Cura performs VM-aware scheduling using dynamic planning of virtual machines through online virtual machine reconfiguration, taking into account dynamic characteristics of workload changes and the cost of the reconfiguration process. The utility-driven scheduling and capacity planning approach of Cura achieves significant improvements on the overall cost of operating the cloud data center from the cloud service providers' point of view without compromising the service level agreement (SLA) of cloud consumers while making services available at a lower cost to cloud users.

1.2.2 Locality-aware Cloud Resource Allocation

The second contribution of this dissertation is the development of a MapReduce Cloud resource allocation system - Purlieus. Purlieus is a self-configurable locality-based data and virtual machine management framework that enables MapReduce jobs to access most of their data either locally or from close-by nodes including all input, output and intermediate data generated during map and reduce phases of the jobs. The first feature of Purlieus is to identify and categorize jobs using a data-size sensitive classifier into three categories namely map-input heavy, reduce-input heavy and map-and-reduce input heavy. The second feature of Purlieus is to provision virtual MapReduce clusters in a locality-aware manner, enabling efficient pairing and allocation of MapReduce virtual machines (VMs) such that the map and reduce processing can access all the input and intermediate data from local or close-by physical machines. We demonstrate how this locality-awareness during both map and reduce phases of the job not only improves runtime performance of individual jobs but also has an additional advantage of reducing network traffic generated in the cloud data center. This is accomplished using a novel coupling of, otherwise independent, data and VM placement steps.

1.2.3 Privacy-conscious Cloud Resource Management

To address the concerns of privacy-conscious data access and query processing, we first investigate on efficient MapReduce analysis for private data archived in a cloud. We then address the challenges of provisioning location privacy-preserving cloud services to mobile cloud users.

1.2.3.1 MapReduce Analysis for Archived Private Data

We note that public storage clouds have become a popular choice for archiving certain classes of enterprise data such as application and infrastructure logs. As these logs can contain sensitive information like IP addresses or user logins, regulatory and security requirements often require data to be encrypted before moved to the cloud. In order to leverage such data for any business value, we note that the current solutions are highly inefficient, requiring to first download the data from the public clouds, decrypt it and then process it at the secure enterprise site.

We propose VNCache: an efficient solution for MapReduce analysis of such cloud-archived log data without requiring an apriori data transfer and loading into the local Hadoop cluster. VNcache dynamically integrates cloud-archived data into a virtual namespace at the enterprise Hadoop cluster. Through a seamless data streaming and prefetching model, Hadoop jobs can begin execution as soon as they are launched without requiring any apriori downloading. With VNcache's accurate pre-fetching and caching, jobs often run on a local cached copy of the data block significantly improving performance. When no longer needed, data is safely evicted from the enterprise cluster reducing the total storage footprint. Uniquely, VNcache is implemented with NO changes to the Hadoop application stack. While our first two contributions of the thesis namely the Cura cost-optimized Cloud system and Purlieus locality-aware resource management are at the cloud provider tier, our VNCache system is implemented at the application service provider tier.

1.2.3.2 Location Privacy-aware Mobile Cloud Services

The next contribution of this dissertation extends our cloud service framework at the cloud consumer tier to support privacy-preserving ubiquitous data access and query processing. Concretely, the goal in this context is to offer anonymous cloud services to mobile users on the move. Cloud users request location dependent data services having their own location privacy constraints and thus the service requests must be honored with respect to personalized location privacy requirements. Spatial cloaking techniques are extensively studied to anonymize GPS or WiFi localization traces such that the exact location of a mobile user is cloaked with a k-anonymized spatial region before forwarding and storing them at an untrusted cloud provider. However spatial cloaking suffers from a number of serious limitations, such as inability to support pseudo-identity based services, lower quality of services due to lower spatial resolution of anonymized locations and vulnerability of privacy leakages for continuous services. As part of this dissertation, we have developed a novel road-network mix-zone based framework for anonymizing location data and serving location based service requests without disclosing privacy in real-time. In contrast to spatial cloaking based location privacy protection, the MobiMix approach aims at breaking the continuity of location exposure through the development of attack resilient road network mix-zones. We have devised non-rectangular road network mix-zones [104] to protect snapshot query users against location privacy risks by offering both timing attack resilience and transition attack resilience even when the mobility patterns and road-network topology are exposed to the attackers. To protect continuous query users, we have developed three types of delay-tolerant road network mix-zones (i.e., temporal, spatial and spatio-temporal) that achieve higher anonymity and attack resilience to CQ-timing attacks and CQ-transition attacks [105]. Overall, the MobiMix approach can provide higher level of anonymity while offering better quality of service for location based

services with hard continuity constraints. The cloud services powered by the MobiMix system offers access privacy protection for cloud consumers enabling them to use cloud service anytime and anywhere worry free of location privacy risks.

The overall goal of this dissertation lies in scalable and cost-effective provisioning and privacy-conscious delivery of Cloud services. We firmly believe that an important enabling technology for cloud services is the ability to offer cost-aware elasticity as well as privacy-conscious scalability and high performance in supporting Big Data applications. This dissertation research scope focuses on taking a systematic approach to developing system-level techniques and algorithms to addressing these challenges and is dedicated to relevant research solutions to the next generation of cloud computing services.

1.3 Organization of the Dissertation

This dissertation is organized into a series of chapters. Each chapter presents the background of the problem being addressed and introduces the technical concepts and discusses the solution techniques followed by experimental evaluation. We discuss related work along every chapter. Below we present a brief overview of every chapter.

Chapter 2: In this chapter, we discuss the cost-inefficiencies in existing cloud models for MapReduce and present the Cura cost-optimized model for MapReduce Clouds. We also discuss the performance enhancement schemes in Cura namely VM-aware scheduling and online virtual machine reconfiguration and present the experimental evaluation.

Chapter 3: We provide an overview of the Purlieus locality-aware resource allocation system. We discuss the challenges related to datacenter networking bottlenecks and argue for improved data locality in the cloud. We present the Purlieus coupled storage-compute architecture for MapReduce and describe its resource allocation techniques.

Chapter 4: In this chapter, we present our caching techniques for privacy-conscious enterprises to efficiently process archived data stored in Public clouds.

Chapter 5: In this chapter, we introduce the location privacy risks involved in accessing cloud services from mobile devices. We present the MobiMix road network based mix-zone framework for protecting location privacy of mobile users. We discuss the unique features of the MobiMix approach and presents its experimental evaluation.

Chapter 6: We introduce the privacy risks involved in accessing continuous query services and present the delay-tolerant mix-zone approach as an effective countermeasure to deal with the privacy attacks related to accessing continuous data services.

Chapter 7: We conclude with a summary of our thesis contribution and discuss open issues and future research directions to the work presented in this thesis.

CHAPTER II

CURA: A COST-OPTIMIZED MODEL FOR MAPREDUCE IN A CLOUD

2.1 Introduction

One of the major IT trends impacting modern enterprises is *big data* and *big data analytics*. As enterprises generate more and more data, deriving business value from it using analytics becomes a differentiating capability – whether it is understanding customer buying behavior or detecting fraud in online transactions. The most popular approach towards such big data analytics is using MapReduce [57] and its open-source implementation called Hadoop [4]. With the ability to automatically parallelize the application on a scale-out cluster of machines, MapReduce can allow analysis of terabytes and petabytes of data in a single analytics job. Today MapReduce forms the core of technologies powering enterprises like Google, Yahoo and Facebook. This MapReduce analytics capability, when paired with another major IT trend – cloud computing, offers a unique opportunity for enterprises interested in big data analytics. A recent Gartner survey shows increasing cloud computing spending with 39% of enterprises having allotted IT budgets for it [77].

In general, there are two approaches in use today for MapReduce in a cloud. In the first approach, customers use a dedicated MapReduce cloud service (e.g. Amazon Elastic MapReduce [9]) and buy on-demand clusters of VMs for each job or a workflow. Once the MapReduce job (or workflow) is submitted, the cloud provider creates VMs that execute that job and after job completion the VMs are deprovisioned. In the second approach customers lease dedicated clusters from a generic cloud service like Amazon Elastic Compute Cloud [10] and operate MapReduce on them as if they

were using a private MapReduce infrastructure. In this case, based on the type of analytics workload they may use different number of VMs for each submitted job, however the entire cluster needs to be maintained by the client enterprise.

In this chapter we argue that these MapReduce cloud models suffer from the following drawbacks:

Interactive workloads: Many modern MapReduce workloads constitute a large fraction of interactive short jobs [37, 94, 43] that require short response times. A recent study on the Facebook and Yahoo production workload traces [44, 43] show that more than 95% of their production MapReduce jobs are short running jobs with an average running time of 30 sec. These jobs typically process a smaller amount of data (less than 200 MB in the Facebook trace [44] that are part of bigger data sets, for example, a friend recommendation query that is issued interactively when a Facebook user browses his/her profile. These jobs process a small amount of data corresponding to a small subset of the social network graph to recommend the most likely known friends to the user. As these jobs are highly interactive, providing high quality of service in terms of job response time is infeasible in a dedicated MapReduce cloud model (the first approach) since it requires virtual clusters to be created afresh for each submitted job. On the other hand an owned cluster in a generic compute cloud (the second approach) has high costs due to low utilization since the cluster needs to be continuously up waiting for jobs and serving them when submitted.

Lack of global optimization: Secondly, both of these models require users to figure out the complex job configuration parameters (e.g. type of VMs, number of VMs and MapReduce configuration like number of mappers per VM etc.) that have an impact on the performance and thus cost of the job. With growing popularity of MapReduce and associated eco-system like Pig [8], Hive [127], many MapReduce jobs nowadays are actually fired by non-engineer data analysts and putting such a burden on those users is impractical. Additionally, even if MapReduce performance

prediction tools like [74] are used in the current cloud models, the per-job optimized configurations created by them from a user perspective are often suboptimal from a cloud provider perspective and hence lead to requiring more cloud resources than that required by a globally optimized schema with resource management performed at the cloud provider-end. A good example of such a cloud managed system is the recent Google BigQuery system [7] which allows to run SQL-like queries against very large datasets with potentially billions of rows. In BigQuery service, customers only submit the queries to be processed on the large datasets and the Cloud service provider intelligently manages the resources for the SQL-like queries.

Low service differentiation: Thirdly, both existing models fail to incorporate significant other optimization opportunities available for the cloud provider to improve its resource utilization. MapReduce workloads often have a large number of jobs that do not require immediate execution, rather feed into a scheduled flow - e.g. MapReduce job analyzing system logs for a daily/weekly status report. By delaying the execution of such jobs, cloud provider can multiplex its resources better for significant cost savings. For instance, the batch query model in Google BigQuery service [7] has 43% lower cost than the interactive query model in which case the queries are instantaneously executed.

To alleviate these drawbacks, we propose a MapReduce cloud service model called Cura. Cura uses a secure instant VM allocation scheme that helps reduce the response time for short jobs by up to 65%. To reduce user complexity, Cura automatically creates the best cluster configuration for the customers jobs with the goal of optimizing the overall resource usage of the cloud. Finally, Cura includes a suite of resource management techniques which leverage deadline awareness for cost-aware resource provisioning. Overall, the use of these techniques including intelligent VM-aware scheduling and online VM reconfiguration techniques lead to more than 80% savings in the cloud infrastructure cost. While Cura focuses on cloud provider's resource

costs, we believe that any cost savings of the cloud provider in terms of infrastructure cost and energy cost based on resource usage would in turn reflect positively in the price of the services for the customers.

2.2 Cura: Model and Architecture

In this section, we present the cloud service model and system architecture for Cura.

2.2.1 Cloud Operational Model

Table 1 shows possible cloud service models for providing MapReduce as a cloud service. The first operational model (immediate execution) is a completely customer managed model where each job and its resources are specified by the customer on a per-job basis and the cloud provider only ensures that the requested resources are provisioned upon job arrival. Many existing cloud services such as Amazon Elastic Compute Cloud [10], Amazon Elastic MapReduce [9] use this model. This model has the lowest rewards since there is lack of global optimization across jobs as well as other drawbacks discussed earlier. The second possible model (delayed start) [123] is partly customer-managed and partly cloud-managed model where customers specify which resources to use for their jobs and the cloud provider has the flexibility to schedule the jobs as long as they **begin** execution within a specified deadline. Here, the cloud provider takes slightly greater risk to make sure that all jobs begin execution within their deadlines and as a reward can potentially do better multiplexing of its resources. However, specifically with MapReduce, this model still provides low cost benefits since jobs are being optimized on a per-job basis by disparate users. In fact customers in this model always tend to greedily choose low-cost small cluster configurations involving fewer VMs that would require the job to begin execution almost immediately. For example, consider a job that takes 180 minutes to complete in a cluster of 2 small instances but takes 20 minutes to complete using a cluster of

Table 1: Cloud Operational Models

Model	**Optimization**	**Provider risk**	**Potential benefits**
Immediate execution	Per-job	Limited	Low
Delayed start	Per-job	Moderate	Low – Moderate
Cloud managed	Global	High	High

6 large instances[1]. Here if the job needs to be completed in more than 180 minutes, the per-job optimization by the customer will tend to choose the cluster of 2 small instances as it has lower resource usage cost compared to the 6 large instance cluster. This cluster configuration, however, expects the job to be started immediately and does not provide opportunity for delayed start. This observation leads us to the next model. The third model – which is the subject of this chapter – is a completely cloud managed model where the customers only submit jobs and specify job completion deadlines. Here, the cloud provider takes greater risk and performs a globally optimized resource management to meet the job SLAs for the customers. Similar high-risk high-reward model is the database-as-a-service model [50, 31, 141] where the cloud provider estimates the execution time of the customer queries and performs resource provisioning and scheduling to ensure that the queries meet their response time requirements. As MapReduce also lends itself well to prediction of execution time [73, 101, 79, 110, 80], we have designed Cura on a similar model. Another recent example of this model is the Batch query model in Google's Big Query cloud service [7] where the Cloud provider manages the resources required for the SQL-like queries so as to provide a service level agreement of executing the query within 3 hours.

2.2.2 System Model: User Interaction

Cura's system model significantly simplifies the way users deal with the cloud service. With Cura, users simply submit their jobs (or composite job workflows) and specify the required service quality in terms of response time requirements. After that, the

[1]Example adapted from the measurements in Herodotou et. al. paper[75]

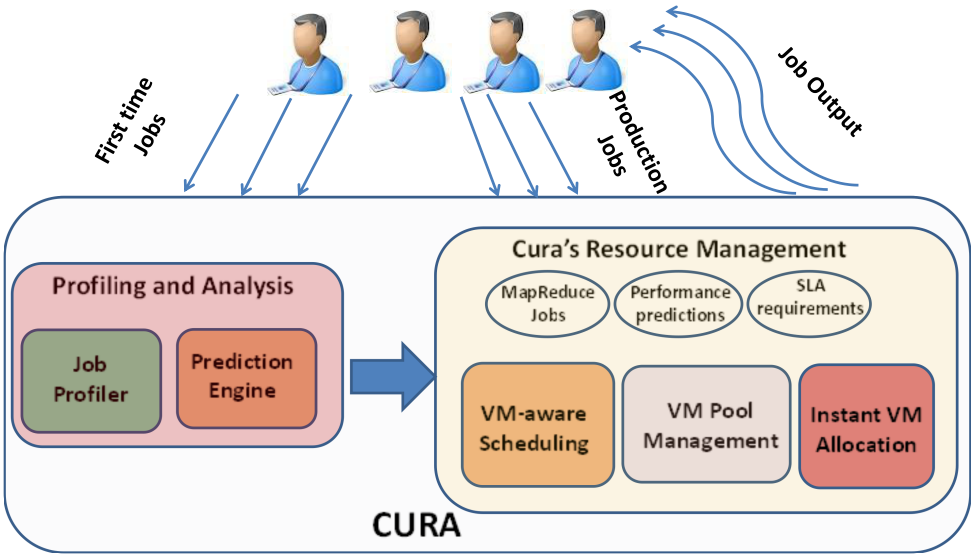


Figure 2: Cura: System Architecture

cloud provider has complete control on the type and schedule of resources to be devoted to that job. From the user perspective, the deadline will typically be driven by their quality of service requirements for the job. As MapReduce jobs perform repeated analytics tasks, deadlines could simply be chosen based on those tasks (e.g. 8 AM for a daily log analysis job). For ad-hoc jobs that are not run per a set schedule, the cloud provider can try to incentivize longer deadlines by offering to lower costs if users are willing to wait for their jobs to be executed[2]. However, this model does not preclude an *immediate execution* mode in which case the job is scheduled to be executed at the time of submission, similar to existing MapReduce cloud service models.

2.2.3 System Architecture

Once a job is submitted to Cura, it may take one of the two paths (Figure 2). If a job is submitted for the very first time, Cura processes it to be *profiled* prior to execution as part of its *profile and analyze* service. This develops a performance

[2]Design of a complete costing mechanism is beyond the scope of this work

model for the job in order to be able to generate predictions for its performance for different VM types, cluster sizes and job parameters. This model is used by Cura in optimizing the global resource allocation. MapReduce profiling has been an active area of research [73, 101, 80] and open-source tools such as Starfish [73] are available to create such profiles. Recent work had leveraged MapReduce profiling for Cloud resource management and showed that such profiles can be obtained with very high accuracy with less than 12% error rate for the predicted running time [79].

The *profile and analyze* service is used only once when a customer's job first goes from development-and-testing into production in its software life cycle. For subsequent instances of the production job, Cura directly sends the job for scheduling. Since typically production jobs including interactive or long running jobs do not change frequently (only their input data may differ for each instance of their execution), profiling will most often be a one-time cost. Further, from an architectural standpoint, Cura users may even choose to skip profiling and instead provide VM type, cluster size and job parameters to the cloud service similar to existing dedicated MapReduce cloud service models like [9]. Jobs that skip the one-time *profile and analyze* step will still benefit from the response time optimizations in Cura described below, however, they will fail to leverage the benefits provided by Cura's global resource optimization strategies. Jobs that are already profiled are directly submitted to the Cura resource management system.

Cura's resource management system is composed of the following components:

2.2.3.1 Secure instant VM allocation

In contrast to existing MapReduce services that create VMs on demand, Cura employs a secure instant VM allocation scheme that reduces response times for jobs, especially significant for short running jobs. Upon completion of a job's execution, Cura only destroys the Hadoop instance used by the job (including all local data) but retains the

VM to be used for other jobs that need the same VM configuration. For the new job, only a quick Hadoop initialization phase is required which prevents having to recreate and boot up VMs[3]. Operationally, Cura creates pools of VMs of different instance types as shown in Figure 3 and dynamically creates Hadoop clusters on them.

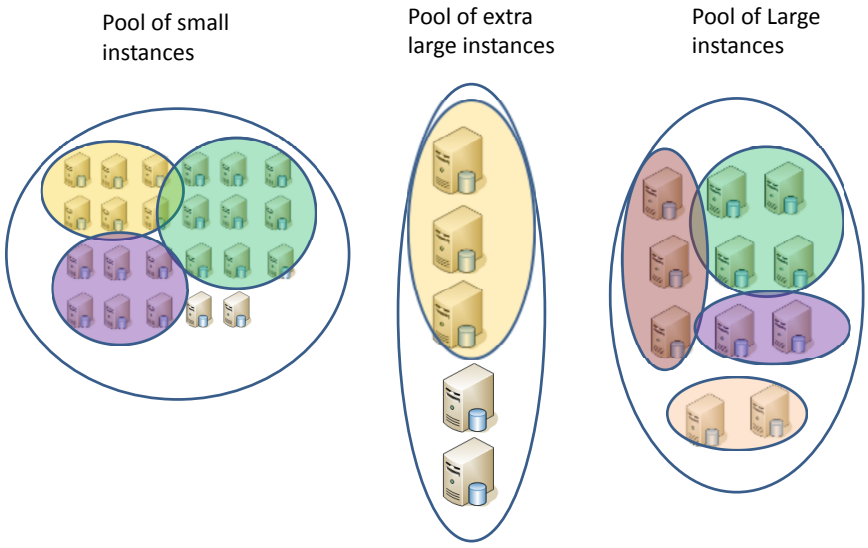


Figure 3: VM Pool

When time sharing a VM across jobs it is important to ensure that an untrusted MapReduce program is not able to gain control over the data or applications of other customers. Cura's security management is based on SELinux [12] and is similar to that of the Airavat system proposed in [114] that showed that enforcing SELinux access policies in a MapReduce cloud does not lead to performance overheads. While Airavat shares multiple customer jobs across the same HDFS, Cura runs only one Hadoop instance at a time and the HDFS and MapReduce framework is used by only one customer before it is destroyed. Therefore, enforcing Cura's SELinux policies does not require modifications to the Hadoop framework and requires creation of only two SELinux domains, one trusted and the other untrusted. The Hadoop framework

[3]Even with our secure instant VM allocation technique data still needs to be loaded for each job into its HDFS, but it is very fast for small jobs as they each process small amount of data, typically less than 200 MB in the Facebook and Yahoo workloads [44].

including the HDFS runs in the trusted domain and the untrusted customer programs run in the untrusted domain. While the trusted domain has regular access privileges including access to the network for network communication, the untrusted domain has very limited permissions and has no access to any trusted files and other system resources. An alternate solution for Cura's secure instant VM allocation is to take VM snapshots upon VM creation and once a customer job finishes, the VM can revert to the old snapshot. This approach is also significantly faster than destroying and recreating VMs, but it can however incur noticeable delays in starting a new job before the VM gets reverted to a secure earlier snapshot.

Overall this ability of Cura to serve short jobs better is a key distinguishing feature. However as discussed next, Cura has many other optimizations that benefit any type of job including long running batch jobs.

2.2.3.2 Job Scheduler

The job scheduler at the cloud provider forms an integral component of the Cura system. Where existing MapReduce services simply provision customer-specified VMs to execute the job, Cura's VM-aware scheduler (Section 2.3.1) is faced with the challenge of scheduling jobs among available VM pools while minimizing global cloud resource usage. Therefore, carefully executing jobs in the best VM type and cluster size among the available VM pools becomes a crucial factor for performance. The scheduler has knowledge of the relative performance of the jobs across different cluster configurations from the predictions obtained from the *profile and analyze* service and uses it to obtain global resource optimization.

2.2.3.3 VM Pool Manager

The third main component in Cura is the VM Pool Manager that deals with the challenge of dynamically managing the VM pools to help the job scheduler effectively obtain efficient resource allocations. For instance, if more number of jobs in the

current workload require small VM instances and the cloud infrastructure has fewer small instances, the scheduler will be forced to schedule them in other instance types leading to higher resource usage cost. The VM pool manager understands the current workload characteristics of the jobs and is responsible for online reconfiguration of VMs for adapting to changes in workload patterns (Section 2.3.2). In addition, this component may perform further optimization such as power management by suitably shutting down VMs at low load conditions.

2.3 Cura: Resource Management

In this section, we describe Cura's core resource management techniques. We first present Cura's VM-aware job scheduler that intelligently schedules jobs within the available set of VM pools. We then present our reconfiguration-based VM pool manager that dynamically manages the VM instance pools by adaptively reconfiguring VMs based on current workload requirements.

2.3.1 VM-aware Scheduling

The goal of the cloud provider is to minimize the infrastructure cost by minimizing the number of servers required to handle the data center workload. Typically the peak workload decides the infrastructure cost for the data center. The goal of Cura VM-aware scheduling is to schedule all jobs within available VM pools to meet their deadlines while minimizing the overall resource usage in the data center reducing this total infrastructure cost. As jobs are incrementally submitted to the cloud, scheduling requires an online algorithm that can place one job at a time on an infrastructure already executing some jobs. To better understand the complexity of the problem, we first analyze an *offline* version which leads us to the design of an online scheduler.

2.3.1.1 *Offline VM-aware Scheduling*

In the offline VM-aware scheduling problem, we assume that information about the jobs, their arrival time and deadlines are known *apriori* and the goal of the algorithm is to schedule all jobs to meet their deadline by appropriately provisioning VM clusters and to minimize the overall resource usage in the cloud. We assume each job, J_i is profiled when it first goes to production and based on the profile it has a number of predictions across various cluster configurations, $C^{k,n}$ in terms of instance types denoted by k and number of VMs denoted by n. Let $t_{arrival}(J_i)$ and $t_{deadline}(J_i)$ denote the arrival time and deadline of job, J_i respectively. The running time of the job, J_i using the cluster configuration, $C^{k,n}$ is given by $t_{run}(J_i, C^{k,n})$ and it includes both execution time and the time for loading data into the HDFS[4]. $Cost(J_i, C^{k,n})$ represents the resource usage cost of scheduling job, J_i using the cluster configuration, $C^{k,n}$. Precisely, the cost, $Cost(J_i, C^{k,n})$ represents the product of the number of physical servers required to host the virtual cluster, $C^{k,n}$ and the running time of the job, $t_{run}(J_i, C^{k,n})$. If R_k represents number of units of physical resources in VM type, k and if each physical server has M units of physical resources[5], the resource usage cost can be computed as:

$$Cost(J_i, C^{k,n}) = t_{run}(J_i, C^{k,n}) \times \frac{n \times R_k}{M}$$

Let $t_{start}(J_i)$ denote the actual starting time of the job, J_i and therefore the end time of job, J_i is given by

$$t_{end}(J_i) = t_{start}(J_i) + \sum_{k,n} X_i^{k,n} \times t_{run}(J_i, C^{k,n})$$

[4]Additionally, $t_{run}(J_i, C^{k,n})$ can also include an error bound in the prediction to ensure that the job will complete within its deadline even when there is prediction error.

[5]Though we present a scalar capacity value, VM resources may have multiple dimensions like CPU, memory and disk. To handle this, our model can be extended to include a vector of resources or compute dimensions can be captured in a scalar value, e.g. the volume metric [139].

where $X_i^{k,n}$ is a Boolean variable indicating if job, J_i is scheduled using the cluster configuration, $C^{k,n}$ and

$$\forall i, \sum_{k,n} X_i^{k,n} = 1$$

In order to ensure that all jobs get completed within their deadlines, we have

$$\forall i, t_{end}(J_i) \leq t_{deadline}(J_i)$$

The sum of concurrent usage of VMs among the running jobs is also constrained by the number of VMs, V_k in the VM pools where k represents the VM type. If S_i^t is a Boolean variable indicating if job, J_i is executing at time, t, we have

$$S_i^t = \begin{cases} 1 & \text{if } t_{start}(J_i) \leq t \leq t_{end}(J_i) \\ 0 & \text{otherwise} \end{cases}$$

$$\forall t, \forall k, \sum_i (S_i^t \times \sum_n (X_i^{k,n} \times n)) \leq V_k$$

With the above constraints ensuring that the jobs get scheduled to meet deadlines, now the key optimization is to minimize the overall resource usage cost of all the jobs in the system.

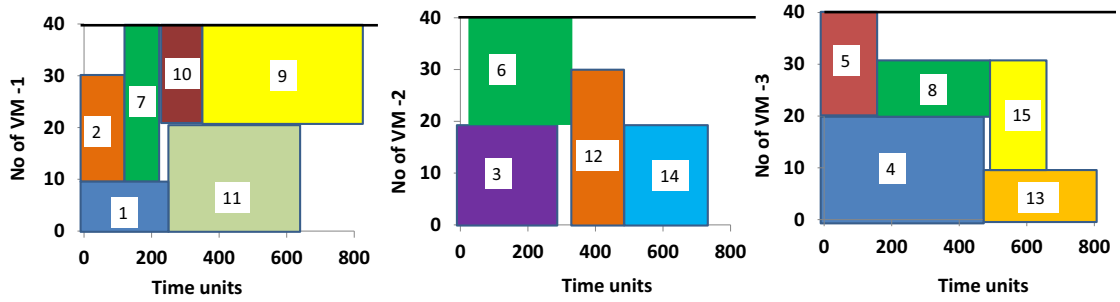
$$Overallcost = \min \sum_{i,k,n} Cost(J_i, C^{k,n}) \times X_i^{k,n}$$

An optimal solution for this problem is NP-Hard with a reduction from the known NP-Hard multi bin-packing problem [62] with additional job moldability constraints. Therefore, we use a heuristics based VM-aware scheduler which is designed to work in an online fashion.

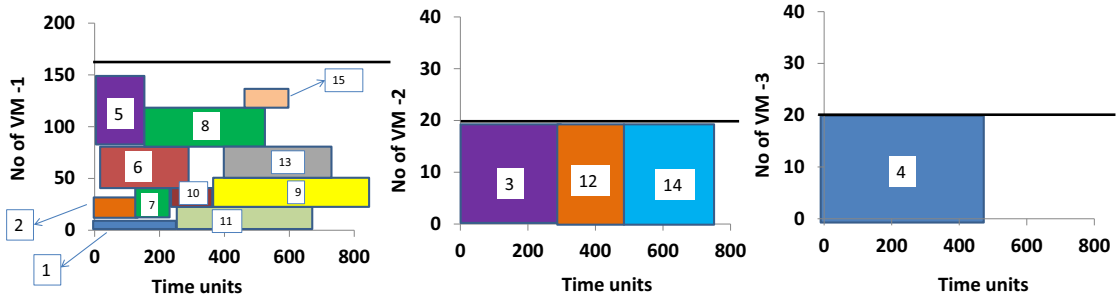
2.3.1.2 Online VM-aware Scheduler

Given VM pools for each VM instance type and continually incoming jobs, the online VM-aware scheduler decides (a) when to schedule each job in the job queue, (b) which

VM instance pool to use and (c) how many VMs to use for the jobs. The scheduler also decides best Hadoop configuration settings to be used for the job by consulting the *profile and analyze* service.



(a) VM-aware schedule



(b) VM-aware schedule with Reconf-based VM pool management

Figure 4: Scheduling in Cura

Depending upon deadlines for the submitted jobs, the VM-aware scheduler typically needs to make future reservations on VM pool resources (e.g. reserving 100 small instances from time instance 100 to 150). In order to maintain the most agility in dealing with incrementally incoming jobs and minimizing the number of reservation cancellations, Cura uses a strategy of trying to create minimum number of future reservations without under-utilizing any resources. For implementing this strategy, the scheduler operates by identifying the *highest priority* job to schedule at any given time and creates a tentative reservation for resources for that job. It then uses the end time of that job's reservation as the bound for limiting the number of reservations i.e. jobs in the job queue that are not schedulable (in terms of start time) within that *reservation time window* are not considered for reservation. This ensures that we are

not unnecessarily creating a large number of reservations which may need cancellation and rescheduling after another job with more stringent deadline enters the queue.

A job J_i is said to have higher priority over job J_j if the schedule obtained by reserving job J_i after reserving job J_j would incur higher resource cost compared to the schedule obtained by reserving job J_j after reserving J_i. The *highest priority* job is picked by performing pairwise cost comparisons. It is chosen such that it will incur higher overall resource usage cost if the highest priority job is deferred as compared to deferring any other job.

For each VM pool, the algorithm picks the highest priority job, J_{prior} in the job queue and makes a reservation for it using the cluster configuration with the lowest possible resource cost at the earliest possible time based on the performance predictions obtained from the *profile and analyze service*. Note that the lowest resource cost cluster configuration need not be the job's optimal cluster configuration (that has lowest per-job cost). For instance, if using the job's optimal cluster configuration at the current time cannot meet the deadline, the lowest resource cost cluster will represent the one that has the minimal resource usage cost among all the cluster configurations that can meet the job's deadline.

Once the highest priority job, J_{prior} is reserved for all VM pools, the reservation time windows for the corresponding VM pools are fixed. Subsequently, the scheduler picks the next highest priority job in the job queue by considering priority only with respect to the reservations that are possible within the current reservation time windows of the VM pools. The scheduler keeps on picking the highest priority job one by one in this manner and tries to make reservations to them on the VM pools within the reservation time window. Either when all jobs are considered in the queue and no more jobs are schedulable within the reservation time window or when the reservations have filled all the resources until the reservation time windows, the scheduler stops reserving.

Then at each time instance, the scheduler picks the reservations for the current time and schedules them on the VM pools by creating Hadoop clusters of the required sizes in the reservation. After scheduling the set of jobs that have reservation starting at the current time, the scheduler waits for one unit of time and considers scheduling for the next time unit. If no new jobs arrived within this one unit of time, the scheduler can simply look at the reservations made earlier and schedule the jobs that are reserved for the current time, however, if some new jobs arrived within the last one unit of time, then the scheduler needs to check if some of the newly arrived jobs have higher priority over the reserved jobs and in that case, the scheduler may require to cancel some existing reservations to reserve some newly arrived jobs that have higher priority over the ones in the reserved list.

Algorithm 1 VM-aware Scheduling

W_{list}: jobs that are waiting to be reserved or scheduled
N_{list}: jobs that arrived since the last time tick
R_{list}: jobs that have a tentative reservation
$_{window}(V)$: reservation time window of VM type V
t_{window} : is the set of time windows of all the VM types
$Cost_{VM}(J_i, J_j, V)$: lowest possible resource usage cost of scheduling jobs J_i and J_j by reserving J_i before job J_j in VM type V
$Cost(J_i, J_j)$: lowest possible cost of scheduling jobs J_i and J_j on any VM type
$Cost_{twindow}(J_i, J_j)$: lowest possible cost of scheduling J_i and J_j by reserving J_i before J_j such that they both start within the time window of the VM pools
$Sched(J_i, t_{window})$: determines if the Job J_i is schedulable within the current time window of the VM pools
All cost calculations consider only cluster configurations that can meet the job's deadline
procedure VMAWARESCHEDULE(W_{list}, N_{list}, R_{list})
    Assign $redo\_reserve = true$ if $\exists J_n \in N_{list}$, $\exists J_r \in R_{list}$ such that $Cost(J_n, J_r) \geq Cost(J_r, J_n)$
    Assign $redo\_timewindow = true$ if $\exists J_n \in N_{list}$, $\exists J_r \in R_{list}$ such that $Cost(J_n, J_r) > Cost(J_r, J_n)$ and J_r is a time window deciding job
    if ($redo\_reserve == false$) **then**
        return
    end if
    if ($redo\_timewindow == true$) **then**
        $CJ_{list} = R_{list} \cup N_{list} \cup W_{list}$
        Cancel all reservations
        for all $V \in VMtypes$ **do**
            Pick and reserve job J_i that maximizes
            $\sum_{J_j \in CJ_{list}} Cost(J_j, J_i) - Cost_{VM}(J_i, J_j, V))$
            $t_{window}(V) = min(t_{end}(J_i), t_{bound})$
        end for
    else
        $CJ_{list} = R_{list} \cup N_{list}$
        Cancel all reservations except t_{window} deciding ones
    end if
    while ($\exists J_i \in CJ_{list} | sched(J_i, t_{window}) == true$) **do**
        Pick and reserve job J_i that maximizes
        $\sum_{J_j \in CJ_{list}} Cost_{twindow}(J_j, J_i) - Cost_{t_{window}}(J_i, J_j)$
    end while
    Run jobs having reservations start at the current time
end procedure

If the scheduler finds that some newly arrived jobs take priority over some jobs in the current reservation list, it first tries to check if the reservation time window of the VM pools need to be changed. It needs to be changed only when some newly arrived jobs take priority over the current highest priority job of the VM pools that decides the reservation time window. If there exists such newly arrived jobs, the algorithm cancels all reserved jobs and moves them back to the job queue and adds all the newly arrived jobs to the job queue. It then picks the highest priority job, J_{prior} for each VM pool from the job queue that decides the reservation time window for each VM pool. Once the new reservation time window of the VM pools are updated, the scheduler considers the other jobs in the queue for reservation within the reservation time window of the VM pools until when either all jobs are considered or when no more resources are left for reservation. In case, the newly arrived jobs do not have higher priority over the time window deciding jobs but have higher priority over some other reserved jobs, the scheduler will not cancel the time window deciding reservations. However, it will cancel the other reservations and move the jobs back to the job queue along with the new jobs and repeat the process of reserving jobs within the reservation time windows from the job queue in the decreasing order of priority. For a data center of a given size, assuming constant number of *profile and analyze* predictions for each job, it can be shown that the algorithm runs in polynomial time with $O(n^2)$ complexity. We present a complete pseudo-code for this VM-aware scheduler in Algorithm 1.

While even a centralized VM-aware scheduler scales well for several thousands of servers with tens of thousands of jobs, it is also straight forward to obtain a distributed implementation to scale further. As seen from the pseudocode, the main operation of the VM-aware scheduler is finding the highest priority job among the n jobs in the queue based on pairwise cost comparisons. In a distributed implementation, this operation can be distributed and parallelized so that if there are n jobs in the queue,

the algorithm would achieve a speed of x with x parallel machines, each of them performing $\frac{n}{x}$ pairwise cost comparisons.

Figure 4(a) shows an example VM-aware schedule obtained for 15 jobs using 40 VMs in each VM type, VM-1, VM-2 and VM-3. Here we assume that jobs 1, 2, 5, 6, 7, 8, 9, 10, 11, 13, 15 have their optimal cluster configuration using VM-1 and jobs 3, 12, and 14 are optimal with VM-2 and job 4 is optimal with VM-3. Here, the VM-aware scheduler tries its best effort to minimize the overall resource usage cost by provisioning the right jobs in the right VM types and using the minimal cluster size required to meet the deadline requirements. However, when the optimal choice of the resource is not available for some jobs, the scheduler considers the next best cluster configuration and schedules them in a cost-aware manner. Below, we discuss a detailed illustrative example.

VMs	t_{run} VM-1	Cost VM-1	t_{run} VM-2	Cost VM-2	t_{run} VM-3	Cost VM-3
10	900	1500	562.5	1875	321.42	2142.85
20	473.68	1578.94	296.05	1973.68	169.17	2255.63
30	333.33	1666.66	208.33	2083.33	119.04	2380.95
40	264.70	1764.70	165.44	2205.88	94.53	2521.00

Table 2: Job type -1: Optimal with virtual machine type -1 (VM-1)

VMs	t_{run} VM-1	Cost VM-1	t_{run} VM-2	Cost VM-2	t_{run} VM-3	Cost VM-3
10	1250	2083.33	500	1666.66	357.14	2380.95
20	657.89	2192.98	263.15	1754.38	187.96	2506.26
30	462.96	2314.81	185.18	1851.85	132.27	2645.50
40	367.64	2450.98	147.05	1960.78	105.04	2801.12

Table 3: Job type -2: Optimal with virtual machine type -2 (VM-2)

VMs	t_{run} VM-1	Cost VM-1	t_{run} VM-2	Cost VM-2	t_{run} VM-3	Cost VM-3
10	5000	8333.33	2187.5	7291.66	875	5833.33
20	2631.57	8771.92	1151.31	7675.43	460.52	6140.35
30	1851.85	9259.25	810.18	8101.85	324.07	6481.48
40	1470.58	9803.92	643.38	8578.43	257.35	6862.74

Table 4: Job type -3: Optimal with virtual machine type -3

VMs	t_{run} VM-1	Cost VM-1	t_{run} VM-2	Cost VM-2	t_{run} VM-3	Cost VM-3
10	250	416.66	156.25	520.83	89.28	595.23
20	131.57	438.59	82.23	548.24	46.99	626.56
30	92.59	462.96	57.87	578.70	33.06	661.37
40	73.52	490.19	45.95	612.74	26.26	700.28

Table 5: Job type -4: Optimal with virtual machine type -1

Table 6 shows a simple workload of 15 jobs scheduled using the VM-aware scheduler. The workload consists of 4 types of jobs. Tables 2, 3, 4 and 5 show the performances predictions of these 4 job types made across 3 VM types. VM-1 is assumed to have 2 GB memory and 2 VCPUs and VM-2 and VM-3 are assumed to have 4 GB memory and 4 VCPUs and 8 GB memory and 8 VCPUs respectively. The tables compare 4 different cluster configurations for each VM type by varying the number of VMs from 10 to 40. The running time of the job in each cluster configuration is shown as t_{run} and the resource utilization cost is shown as $Cost$. We find that job type 1 is optimal with the VM-1 and incurs 20% additional cost with VM-2 and 30% additional cost with VM-3. Similarly, job type 2 is optimal with VM-2 and incurs 20% additional cost with VM-1 and 30% additional cost with VM-3. Job type 3 is optimal for VM-3 and incurs 30% additional cost with VM-1 and 20% additional cost with VM-2. Job type 4 is similar to job type-1 which is optimal for VM-1, but it has shorter running time.

In Table 6, the arrival time and the deadline of the jobs are shown. Now, the scheduler's goal is to choose the number of virtual machines and the virtual machine type to use for each job. At time $t = 0$, we find jobs, 1, 2, 3, 4 and 5 in the system. Based on the type of the jobs and by comparing the cost shown in Tables 2 - 5, jobs 1, 2 and 5 are optimal with VM-1 whereas job 3 is optimal with VM-2 and job 4 is optimal with VM-3. The VM-aware scheduler chooses job 1 as the time window deciding job for VM-1 based on the cost-based priority and chooses jobs 3 and 4 as the time window deciding jobs for VM-2 and VM-3 respectively. Once the time windows are decided, it reserves and schedules job 2 in VM-1 based on the cost-based priorities

by referring to the performance comparison tables. Similarly it reserves and schedules job 5 in VM-3, however job 5 is optimal only with VM-1. As there is not enough resources available in the VM pool of VM-1, the scheduler is forced to schedule it in VM-3 although it knows that it is less efficient.

At time $t = 5$, job 6 arrives and it is scheduled in VM-2 within the reservation time window as the other permissible cluster configurations using the VM types can not meet its deadline. When job 7 arrives at time, $t = 105$ it is reserved and scheduled in VM-1 within its reservation time window. At time $t = 160$ When job 8 arrives, the scheduler identifies that it is optimal with VM-1, however as there is not enough VMs in VM-1, it schedules it in VM-3 as the reservation of job 8 starts within the current reservation time window of VM-3. When job 9 arrives, it gets reserved on VM-1 to start at $t = 225$ as it is optimal with VM-1. However, when job 10 arrives at $t = 220$ it overrides job 9 by possessing higher priority and hence job 9's reservation is cancelled and job 10 is reserved and scheduled at $t = 225$.

After job 11 arrives at time $t = 230$ and gets scheduled at $t = 250$, the reservation time window needs to be updated for VM-1. The scheduler compares the priority based on the cost and identifies job 11 as the time window deciding job and schedules it at time $t = 250$. Subsequently, job 9's reservation is also made at the earliest possible, $t = 357$ within the new reservation time window. When job 12 arrives, the scheduler identifies that it is optimal with VM-2 and it is reserved at the earliest possible time $t = 302$ and at that time the reservation time window for VM-2 is also updated with job 12. We note that job 13 is optimal with VM-1, however it gets reserved and scheduled only with VM-3 as it has stronger deadline requirements that only VM-3 can satisfy given the available resources in the other pools. Job 14 arrives at $t = 430$ and gets reserved and scheduled at $t = 450$ which also updates the reservation time window of VM-2. However, Job 15 which is optimal with VM-1 needs to be scheduled with VM-3 due to lack of available resources in VM-1 pool.

Thus the VM-aware scheduler minimizes the overall resource usage cost even though some jobs violate their per-job optimality.

Job id	type	arrival time	deadline	VM	No VMs	start	end
1	4	0	270	1	10	0	250
2	4	0	150	1	20	0	132
3	2	0	275	2	20	0	264
4	3	0	475	3	20	0	461
5	1	0	185	3	20	0	170
6	1	5	310	2	20	0	302
7	1	105	250	1	30	132	225
8	1	160	500	3	10	170	492
9	1	215	850	1	20	357	831
10	1	220	400	1	20	225	357
11	1	230	650	1	20	250	624
12	2	240	460	2	40	302	450
13	1	400	800	3	10	461	783
14	2	430	730	2	20	450	714
15	4	460	700	3	20	492	662

Table 6: VM-aware schedule

2.3.2 Reconfiguration-based VM Management

Although the VM-aware scheduler tries to effectively minimize the global resource usage by scheduling jobs based on resource usage cost, it may not be efficient if the underlying VM pools are not optimal for the current workload characteristics. Cura's reconfiguration-based VM manager understands the workload characteristics of the jobs as an online process and performs online reconfiguration of the underlying VM pools to better suit the current workload. For example, the VM pool allocation shown in Figure 3 can be reconfigured as shown in Figure 5 to have more small instances by shutting down some large and extra large instances if the current workload pattern requires more small instances.

The reconfiguration-based VM manager considers the recent history of job executions by observing the jobs that arrived within a period of time referred to as the reconfiguration time window. For each job, J_i arriving within the reconfiguration time window, the reconfiguration algorithm understands the optimal cluster configuration, $C_{opt}(J_i)$ that incurs the lowest resource usage cost among all cluster configurations that can meet the job's deadline requirements. At the end of the reconfiguration

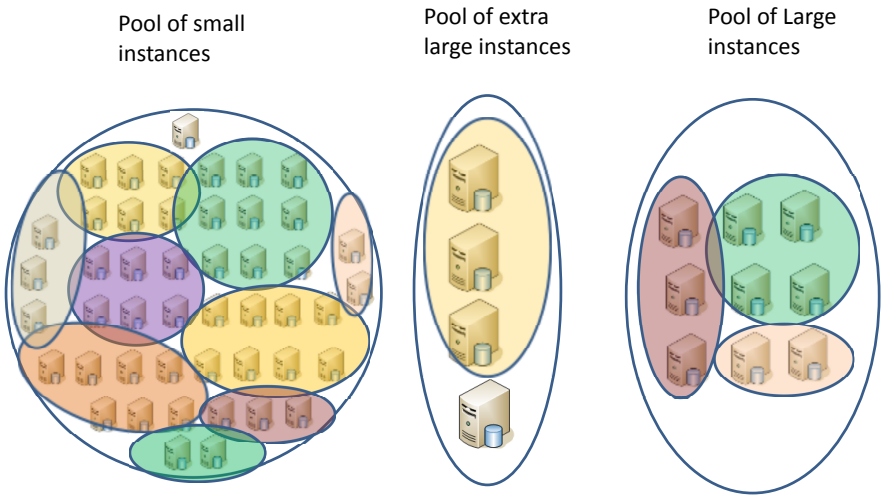


Figure 5: Reconfiguration-based VM Management

time window period, the algorithm decides on the reconfiguration plan by making a suitable tradeoff between the performance enhancement obtained after reconfiguration and the cost of the reconfiguration process. If $Y_i^{k,n}$ is a Boolean variable indicating if $C^{k,n}$ is the optimal cluster configuration for job, J_i, then the proportion of physical resources, P_k to be allocated to each VM type k can be estimated based on the cumulative resource usage in each VM pool computed as the product of total running time of the jobs and the size of the cluster used:

$$P_k = \frac{\sum_{i,n}(t_{run}(J_i, C_{opt}(J_i)) \times n \times Y_i^{k,n})}{\sum_{i,k,n}(t_{run}(J_i, C_{opt}(J_i)) \times n \times Y_i^{k,n})}$$

The total physical resources, R^{total} in the cloud infrastructure can be obtained as

$$R^{total} = \sum_k V_k \times R_k$$

where R_k represents the physical resource in VM type, k, and V_k is the number of VMs in the existing VM pool of type k. Therefore, the number of VMs, V_k' in the new reconfigured VM pools is given by

$$V'_k = P_k \times \frac{R^{total}}{R_k}$$

Such reconfiguration has to be balanced against the cost of reconfiguration operations (shutting down some instances and starting others). For this, we compute the benefit of doing such reconfiguration. The overall observed cost represents the actual cumulative resource cost of the jobs executed during the reconfiguration time window using existing VM pools. Here, $Z_i^{k,n}$ is a Boolean variable indicating if the job J_i used the cluster configuration, $C_i^{k,n}$.

$$Overallcost_{observed} = \sum_{i,k,n} Cost(J_i, C^{k,n}) \times Z_i^{k,n}$$

Next, we compute the estimated overall cost with new VM pools assuming that the jobs were scheduled using their optimal cluster configurations, $C_{opt}(J_i)$. Reconfiguration benefit, $Reconf_{benefit}$ is then computed as the difference between the two.

$$Overallcost_{estimate} = \sum_{i} Cost(J_i, C_{opt}(J_i))$$

$$Reconf_{benefit} = Overallcost_{estimate} - Overallcost_{actual}$$

Assuming the reconfiguration process incurs an average reconfiguration overhead, $Reconf_{overhead}$ that represents the resource usage spent on the reconfiguration process for each VM that undergoes reconfiguration, the total cost of reconfiguration is obtained as

$$Reconf_{cost} = \sum_{k} |(V'_k - V_k)| \times Reconf_{overhead}$$

The algorithm then triggers the reconfiguration process only if it finds that the estimated benefit exceeds the reconfiguration cost by a factor of β, i.e., if $Reconf_{benefit} \geq \beta \times Reconf_{cost}$ where $\beta > 1$. As $Reconf_{benefit}$ only represents an estimate of the benefit, β is often chosen as a value greater than 1. When the reconfiguration process

starts to execute, it shuts down some VMs whose instance types needs to be decreased in number and creates new VMs of the instance types that needs to created. The rest of the process is similar to any VM reconfiguration process that focuses on the *bin-packing* aspect of placing VMs within the set of physical servers during reconfiguration [139, 121].

Continuing the example of Figure 4, we find that the basic VM-aware scheduler in Figure 4(a) without reconfiguration support schedules jobs 5, 6, 8, 13, 15 using VM-2 and VM-3 types even though they are optimal with VM-1, The reconfiguration based VM-aware schedule in Figure 4(b) provisions more VM-1 instances (notice changed Y-axis scale) by understanding the workload characteristics and hence in addition to the other jobs, jobs 5, 6, 8 13 and 15 also get scheduled with their optimal choice of VM-type namely VM-1, thereby minimizing the overall resource usage cost in the cloud data center.

For the same workload shown in Table 6, with the reconfiguration-based VM pool management, the allocation of the VMs in each pool is based on the current workload characteristics. For the example simplicity, we do not show the reconfiguration process in detail, instead we assume that the reconfiguration is performed and illustrate the example with the efficient schedule obtained by the VM-aware scheduler with the reconfigured VM pools. In Table 7, we note that all the jobs of job type 1 and job type 4 are scheduled using their optimal VM type VM-1. Similarly type 2 and type 3 jobs also obtain their optimal VM types VM-2 and VM-3 respectively.

2.4 Experimental Evaluation

We divide the experimental evaluation of Cura into two – first, we provide detailed analysis on the effectiveness of Cura compared to conventional MapReduce services and then we present an extensive micro analysis on the different set of techniques in Cura that contribute to the overall performance. We first start with our experimental

Job id	type	arrival time	deadline	VM	No VMs	start	end
1	4	0	270	1	10	0	250
2	4	0	150	1	20	0	132
3	2	0	275	2	20	0	264
4	3	0	475	3	20	0	461
5	1	0	185	1	70	0	172
6	1	5	310	1	40	0	270
7	1	105	250	1	30	132	225
8	1	160	510	1	30	172	502
9	1	215	850	1	20	357	831
10	1	220	400	1	20	225	357
11	1	230	650	1	20	250	624
12	2	240	460	2	20	264	529
13	1	400	800	1	30	400	734
14	2	480	730	2	20	529	773
15	4	460	700	1	20	460	592

Table 7: Schedule with Reconfiguration-based VM Management

setup.

2.4.1 Experimental setup

Metrics: We evaluate our techniques on four key metrics with the goal of measuring their cost effectiveness and performance– (1) *number of servers*: techniques that require more number of physical servers to successfully meet the service quality requirements are less cost-effective; this metric measures the capital expense on the provisioning of physical infrastructure in the data center, (2) *response time*: techniques that have higher response time provide poor service quality; this metric captures the service quality of the jobs, (3) *per-job infrastructure cost* - this metric represents the average per-job fraction of the infrastructure cost; techniques that require fewer servers will have lower per-job cost and (4) *effective utilization*: techniques that result in poor utilization lead to higher cost; this metric captures both the cost-effectiveness and the performance of the techniques. It should be noted that the effective utilization captures only the useful utilization that represents job execution and does not include the time taken for creating and destroying VMs.

Cluster Setup: Our cluster consists of 20 CentOS 5.5 physical machines (KVM as the hypervisor) with 16 core 2.53GHz Intel processors and 16 GB RAM. The machines are organized in two racks, each rack containing 10 physical machines. The network

is 1 Gbps and the nodes within a rack are connected through a single switch. We considered 6 VM instance types with the lowest configuration starting from 2 2 GHz VCPUs and 2 GB RAM to the highest configuration having 12 2GHz VCPUs and 12 GB RAM with each VM configuration differing by 2 2 GHz VCPUs and 2 GB RAM with the next higher configuration.

Workload: We created 50 jobs using the Swim MapReduce workload generator [44] that richly represent the characteristics of the production MapReduce workload in the Facebook MapReduce cluster. The workload generator uses a real MapReduce trace from the Facebook production cluster and generates jobs with similar characteristics as observed in the Facebook cluster. Using the *Starfish* profiling tool [73], each job is profiled on our cluster setup using clusters of VMs of all 6 VM types. Each profile is then analyzed using *Starfish* to develop predictions across various hypothetical cluster configurations and input data sizes.

Before discussing the experimental results, we briefly discuss the set of techniques compared in the evaluation.

Per-job cluster services: Per job services are similar to dedicated MapReduce services such as Amazon Elastic MapReduce [9] that create clusters per job or per workflow. While this model does not automatically pick VM and Hadoop parameters, for a fair comparison we use *Starfish* to create the optimal VM and Hadoop configuration even in this model.

Dedicated cluster services: Dedicated clusters are similar to private cloud infrastructures where all VMs are managed by the customer enterprises and Hadoop clusters are formed on demand when jobs arrive. Here again the VM and job parameters are chosen via Starfish.

Cura: Cura incorporates both the VM-aware scheduler and reconfiguration-based VM pool management. For the micro-analysis, we also compare the following sub-techniques to better evaluate Cura: *1) Per-job Optimization* technique that uses

Cura's secure instant VM allocation but always uses the per-job optimal number of VMs and the optimal VM type, *2) VM-aware scheduler* described in Section 2.3.1 and *3) Reconfiguration based VM Management* (Section 2.3.2).

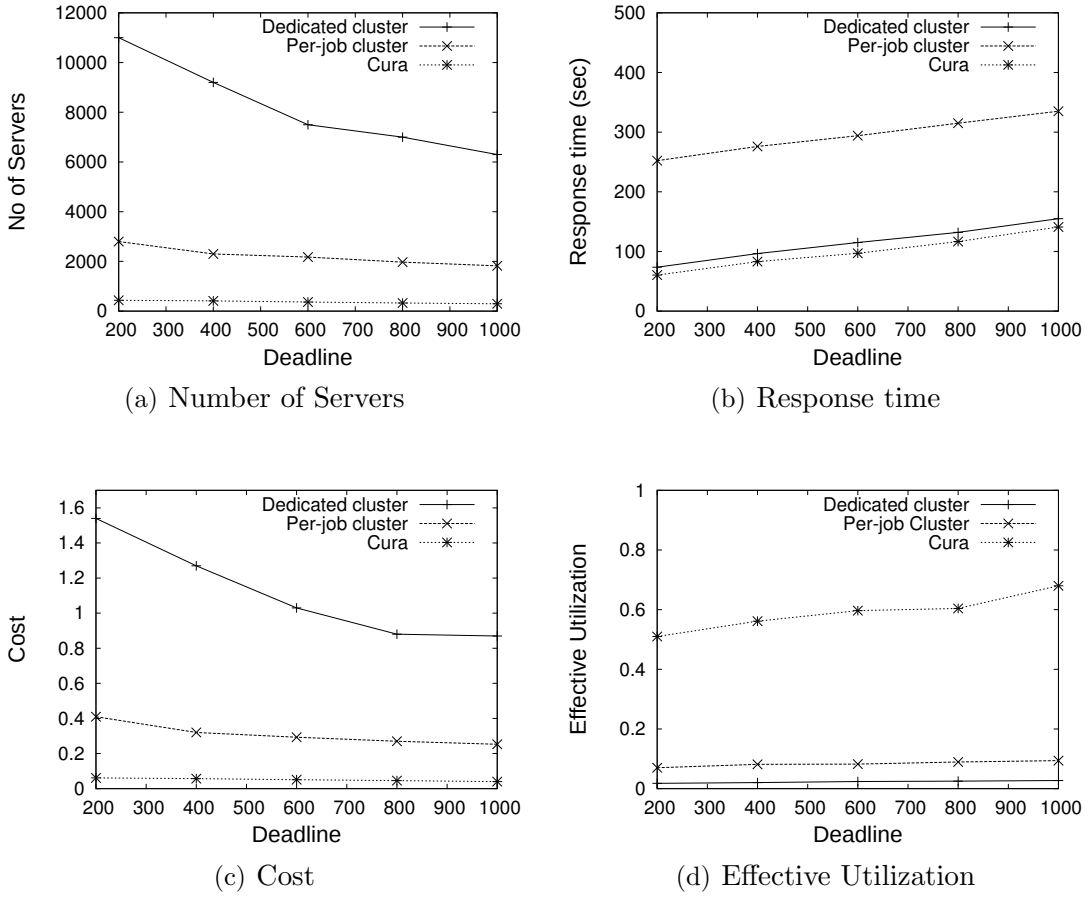
2.4.2 Experimental Results

We first present the experimental evaluation of Cura by comparing with the existing techniques for various experimental conditions determined by distribution of the job deadlines, size of the MapReduce jobs, number of servers in the system and the amount of prediction error in the *profile and analyze* process. By default, we use a composite workload consisting of equal proportion of jobs of three different categories: small jobs, medium jobs and large jobs. Small jobs read 100 MB of data, whereas medium jobs and large jobs read 1 GB and 10 GB of input data respectively. We model Poisson job arrivals with rate parameter, $\lambda = 0.5$ and the jobs are uniformly distributed among 50 customers. The evaluation uses 11,500 jobs arriving within a period of 100 minutes. Each of the arrived job represents one of the 50 profiled jobs with input data size ranging from 100 MB to 10 GB based on the job size category. By default, we assume that jobs run for the same amount of time predicted in the *profile and analyze* process, however, we dedicate a separate set of experiments to study the performance of the techniques when such predictions are erroneous. Note that a job's complete execution includes both the data loading time from the storage infrastructure to the compute infrastructure and the Hadoop startup time for setting up the Hadoop cluster in the cluster of VMs. The data loading time is computed by assuming a network throughput of 50 MBps per VM [6] from the storage server and the Hadoop startup time is taken as 10 sec.

[6]Here, the 50 MBps throughput is a conservative estimate of the throughput between the storage and compute infrastructures based on measurement studies on real cloud infrastructures [63].

2.4.2.1 *Effect of job deadlines*

In this set of experiments, we first study the effect of job deadlines on the performance of Cura with other techniques (Figure 6) and then we analyze the performance of Cura in terms of the contributions of each of its sub-techniques (Figure 7). Figure 6(a) shows the performance of the techniques for different maximum deadlines with respect to number of servers required for the cloud provider to satisfy the workload. Here, the deadlines are uniformly distributed within the maximum deadline value shown on the X-axis.



(a) Number of Servers (b) Response time

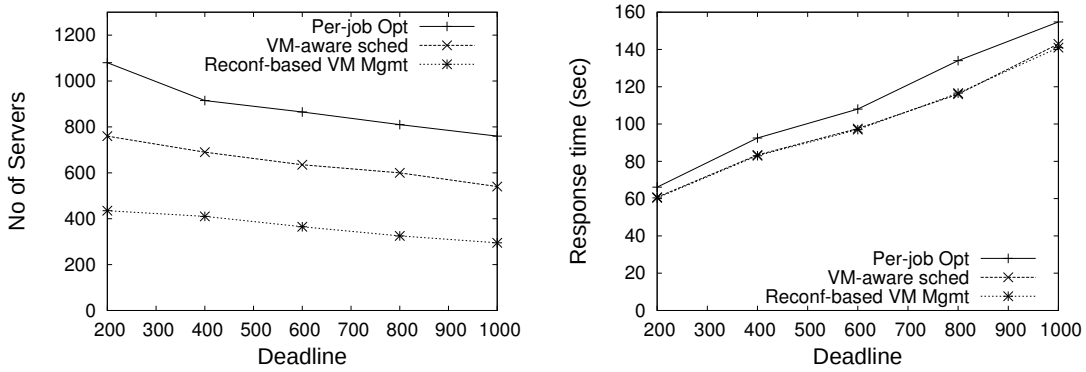
(c) Cost (d) Effective Utilization

Figure 6: Effect of Job-deadlines

We find that provisioning dedicated clusters for each customer results in a lot of resources as dedicated clusters are based on the peak requirements of each customer

and therefore the resources are under-utilized. On the other hand, per-job cluster services require lower number of servers (Figure 6(a)) as these resources are shared among the customers. However, the Cura approach in Figure 6(a) has a much lower resource requirement having up to 80% reduction in terms of the number of servers. This is due to the designed *global optimization* capability of Cura. Where per-job and dedicated cluster services always attempt to place jobs based on per-job optimal configuration obtained from *Starfish*, resources for which may not be available in the cloud, Cura on the other hand can schedule jobs using other than their individual optimal configurations to better adapt to available resources in the cloud.

We also compare the approaches in terms of the mean response time in Figure 6(b). To allow each compared technique to successfully schedule all jobs (and not cause failures), we use the number of servers obtained in Figure 6(a) for each individual technique. As a result, in this response time comparison, Cura is using much fewer servers than the other techniques. We find that the Cura approach and the dedicated cluster approach have lower response time (up to 65%).



(a) Number of Servers (b) Response time

Figure 7: Effect of Job-deadlines

In the per-job cluster approach, the VM clusters are created for each job and it takes additional time for the VM creation and booting process before the jobs can begin execution leading to the increased response time of the jobs. Similar to the

comparison on the number of servers, we see the same trend with respect to the per-job cost in Figure 6(c) that shows that the Cura approach can significantly reduce the per-job infrastructure cost of the jobs (up to 80%). The effective utilization in Figure 6(d) shows that the per-job cluster services and dedicated cluster approach have much lower effective utilization compared to the Cura approach. The per-job services spend a lot of resources in creating VMs for every job arrival. Especially with short response time jobs, the VM creation becomes a bigger overhead and reduces the effective utilization. The dedicated cluster approach does not create VMs for every job instance, however it has poor utilization because dedicated clusters are sized based on peak utilization. But the Cura approach has a high effective utilization having up to 7x improvement compared to the other techniques as Cura effectively leverages global optimization and deadline-awareness to achieve better resource management.

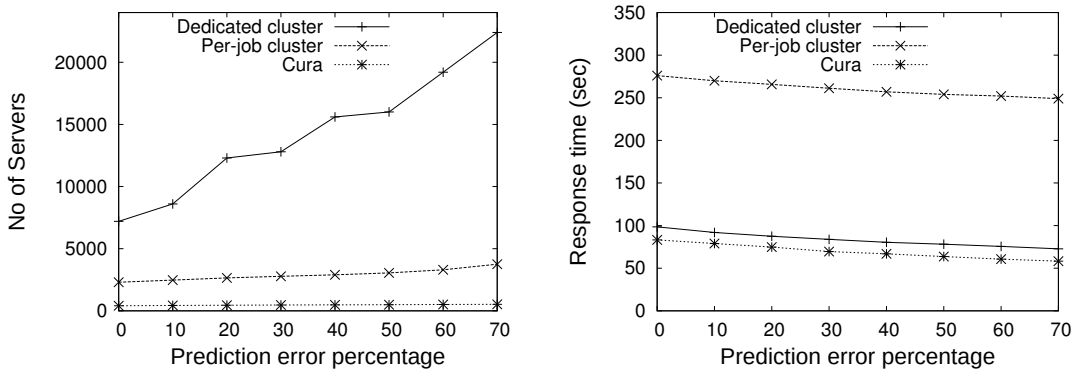
Micro Analysis: Next, we discuss the performance of the sub-techniques of Cura and illustrate how much each sub-technique contributes to the overall performance under different deadlines. Figure 7(a) shows that with only per-job optimization (which only leverages instant VM allocation), it requires up to 2.6x higher number of servers compared to using reconfiguration-based VM pool management scheme with the VM-aware scheduler. The per-job optimization scheduler always chooses the optimal VM type and the optimal number of VMs for each job and in case the optimal resources are not available when the job arrives, the scheduler keeps on queuing the job until the required optimal resource becomes available when some other jobs complete. It drops the request when it finds out that the job cannot meet its deadline if the optimal resources are provisioned. However, with the VM-aware approach, the scheduler will be able to still schedule the job by provisioning higher resources in order to meet the deadline. Second, with the per-job optimization scheduler, even when some sub-optimal resources are available when the job is waiting, they remain unused as the job is expecting to be scheduled only using the optimal resources. Therefore

the per-job optimization results in poor performance. The number of servers required by the VM-aware approach is significantly reduced by up to 45% servers by efficient reconfiguration-based VM management that dynamically manages the VMs in each VM pool. Figure 7(b) shows the mean response time of the jobs for various sub-techniques. We find that the sub-techniques have similar response times except for the per-job optimization case that has up to 11% higher mean response time. As per-job optimization scheduler keeps the jobs waiting until it finds their optimal resources, it leads to higher queuing time that causes this increase.

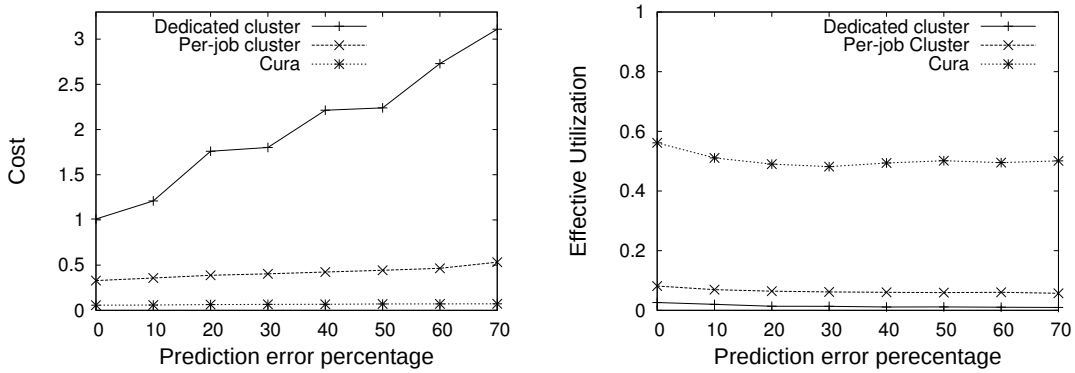
2.4.2.2 Effect of Prediction Error

This set of experiments evaluates the techniques by studying the effect of inaccuracies in the performance prediction. As accurate performance predictions may not always be available, it is important that the techniques can tolerate inaccuracies in performance prediction and yet perform efficiently. Figure 8 shows the comparison of the techniques while varying the error rate from 0 to 70%. Here, the mean deadline of the jobs is taken as 200 second. The error rate means that accurate running time of the jobs can be anywhere within the error range on both sides of the predicted value. The comparison of number of servers in Figure 8(a) shows that all the techniques require more number of servers when the prediction error increases. The Cura approach on an average requires 4% additional number of servers for every 10% increase in prediction error. Note that even the per-job cluster and dedicated cluster schemes require increased number of servers as they also decide the resource requirements based on the performance predictions of the jobs across different cluster configurations.

Figure 8(b) shows that the response time of the techniques decreases with increase in the error rate. While the Cura and dedicated cluster approaches have a decrease of 4.2% and 3.7% respectively, the per-job cluster approach has a decrease of only 1.4% for every 10% increase in error rate as the major fraction of the response time



(a) Number of Servers (b) Response time



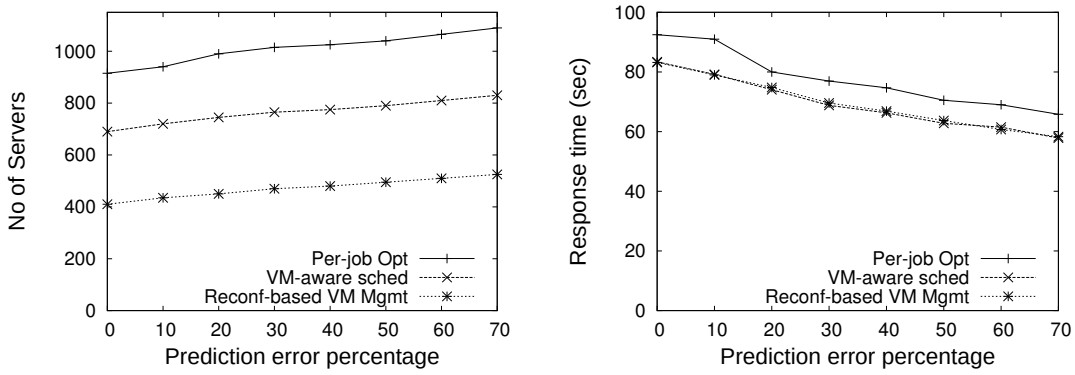
(c) Cost (d) Utilization

Figure 8: Effect of Prediction Error

in these services is due to the VM creation process. As error rate increases, the techniques provision more resources to ensure that even in the worst case, when the jobs run for the maximum possible time within the error range, the jobs complete within the deadline. Therefore, in cases where the job completes within the maximum possible running time, these additional resources make the job complete earlier than its deadline and therefore it speeds up the execution resulting in lower response time. The cost trend shown in Figure 8(c) also shows that the techniques that require fewer servers result in lower per-job cost. Similarly the effective utilization comparison in Figure 8(d) shows similar relative performance as in Figure 6(d)

We compare the performance of the sub-techniques of Cura under different error

rates in Figure 9. We find that the number of servers in Figure 9(a) shows a similar relative performance among the sub-techniques as in 8(a). Here again, the response time as shown in Figure 9(b) shows that the per-job optimization scheduler leads to higher response time due to queue wait times and the response time of the sub-techniques increases with increase in error rate.



(a) Number of Servers (b) Response time

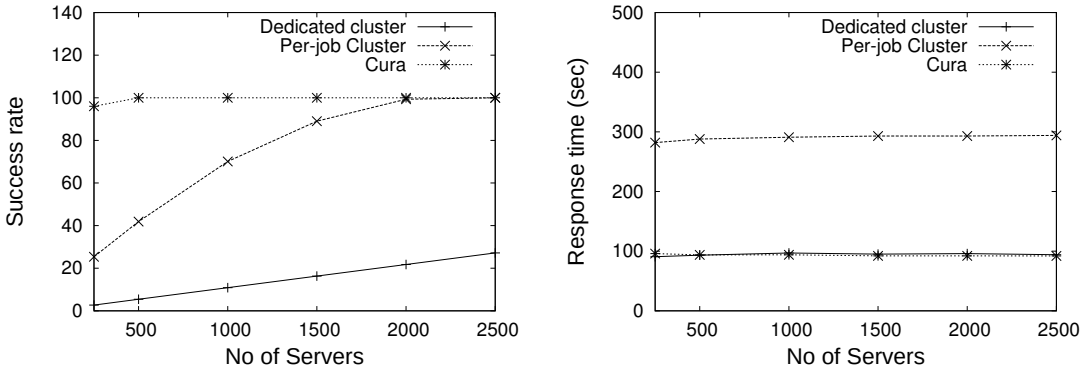
Figure 9: Effect of Prediction Error

2.4.2.3 Varying number of Servers

We next study the performance of the techniques by varying the number of servers provisioned to handle the workload. Figure 10(a) shows the success rate of the approaches for various number of servers. Here, the success rate represents the fraction of jobs that successfully meet their deadlines. We find that the Cura approach has a high success rate even with 250 servers, whereas the per-job cluster approach obtains close to 100% rate only with 2000 servers. Figure 10(b) shows that the response time of successful jobs in the compared approaches show a similar trend as in Figure 6(b) where the Cura approach performs better than the per-job cluster services.

2.4.2.4 Varying job sizes

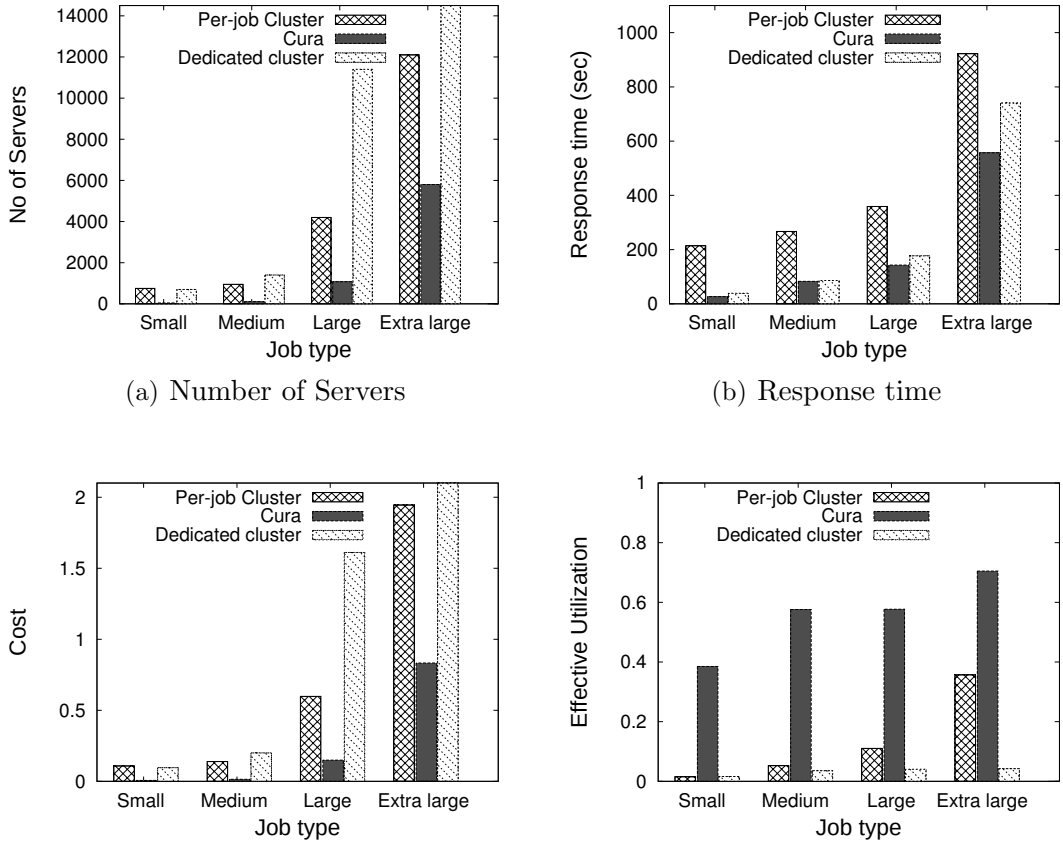
This set of experiments evaluates the performance of the techniques for various job sizes based on the size of input data read. Note that small jobs process 100 MB of



(a) Success Rate (b) Response time

Figure 10: Effect of servers

data, medium jobs process 1 GB of data and large and extra large jobs process 10 GB and 100 GB of data respectively. Also small, medium and large jobs have a mean deadline of 100 second and the extra large jobs have a mean deadline of 1000 second as they are long running. We find that the performance in terms of number of servers in Figure 11(a) has up to 9x improvement for the short and medium jobs with Cura approach compared to the per-job cluster approach. It is because in addition to the VM-aware scheduling and reconfiguration-based VM management, these jobs benefit the most from the secure instant VM allocation as these are short jobs. For large and extra large jobs, the Cura approach still performs significantly better having up to 4x and 2x improvement for large and extra large jobs compared to the per-job cluster services. The dedicated cluster service requires significantly higher resources for large jobs as the peak workload utilization becomes high (its numbers significantly cross the max Y-axis value). This set of experiments show that the global optimization techniques in Cura are not only efficient for short jobs but also for long running batch workloads. The response time improvements of Cura and dedicated cluster approach in Figure 11(b) also show that the improvement is very significant for short jobs having up to 87% reduced response time and up to 69% for medium jobs. It is reasonably significant for large jobs with up to 60% lower response time and extra



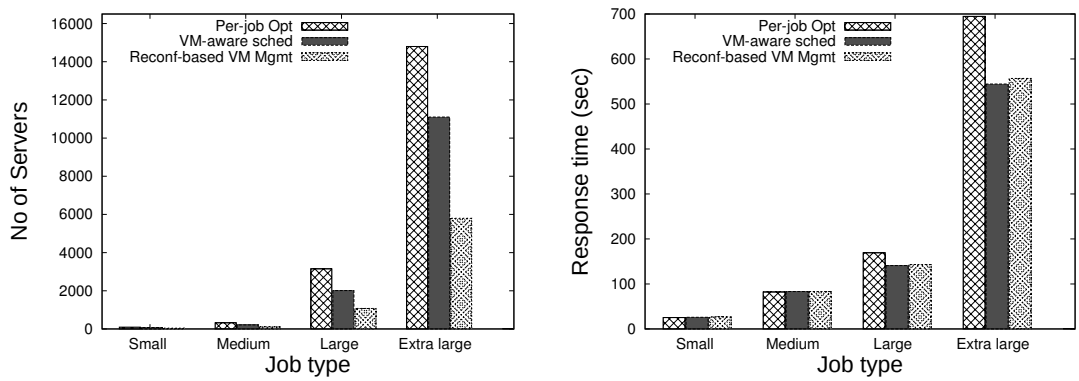
(a) Number of Servers (b) Response time

(c) Cost (d) Effective Utilization

Figure 11: Effect of Job type

large jobs with up to 30% reduced response time. The cost comparison in Figure 11(c) also shows a similar trend that the Cura approach, although is significantly effective for both large and extra large jobs, the cost reduction is much more significant for small and medium jobs.

The sub-technique comparison of Cura for various job types in terms of number of servers is shown in Figure 12(a). We find that the sub-techniques have impact on all kind of jobs irrespective of the job size. While secure instant VM allocation contributes more to the performance of the small jobs compared to large jobs, the sub-techniques in Cura have equal impact on the overall performance for all job categories. The response time comparison of the sub-techniques in Figure 12(b) shows that the



(a) Number of Servers (b) Response time

Figure 12: Effect of Job type

sub-techniques have similar response time, however, for large and extra large jobs, the per-job optimization leads to increased response time by up to 24.8% as large jobs in the per-job optimization require incur longer waiting time in the queue as they often request more resources that may not be immediately available.

2.5 Related Work

Resource Allocation and Job Scheduling: There is a large body of work on resource allocation and job scheduling in grid and parallel computing. Some representative examples of generic schedulers include [102, 122]. The techniques proposed in [26, 113] consider the class of malleable jobs where the number processors provisioned can be varied at runtime. Similarly, the scheduling techniques presented in [124, 46] consider moldable jobs that can be run on different number of processors. These techniques do not consider a virtualized setting and hence do not deal with the challenges of dynamically managing and reconfiguring the VM pools to adapt for workload changes. Therefore, unlike Cura they do not make scheduling decisions over *dynamically* managed VM pools. Chard et. al present a resource allocation framework for grid and cloud computing frameworks by employing economic principles in job scheduling [41]. Hacker et. al propose techniques for allocating virtual clusters by

queuing job requests to minimize the spare resources in the cloud [70]. Recently, there has been work on cloud auto scaling with the goal of minimizing customer cost while provisioning the resources required to provide the needed service quality [90]. The authors in [123] propose techniques for combining on demand provisioning of virtual resources with batch processing to increase system utilization. Although the above mentioned systems have considered cost reduction as a primary objective of resource management, these systems are based on either per-job or per-customer optimization and hence unlike Cura, they do not lead to a globally optimal resource management.

MapReduce task placement: There have been several efforts that investigate task placement techniques for MapReduce while considering fairness constraints [115, 142]. *Mantri* tries to improve job performance by minimizing outliers by making network-aware task placement [24]. Similar to Yahoo's capacity scheduler and Facebook's fairness scheduler, the goal of these techniques is to appropriately place tasks for the jobs running in a given Hadoop cluster to optimize for locality, fairness and performance. Cura, on the other hand deals with the challenges of appropriately provisioning the right Hadoop clusters for the jobs in terms of VM instance type and cluster size to globally optimize for resource cost while dynamically reconfiguring the VM pools to adapt for workload changes.

MapReduce in a cloud: Recently, motivated by MapReduce, there has been work on resource allocation for data intensive applications in the cloud context [78, 103]. Quincy [78] is a resource allocation system for scheduling concurrent jobs on clusters and Purlieus [103] is a MapReduce cloud system that improves job performance through locality optimizations achieved by optimizing data and compute placements in an integrated fashion. However, unlike Cura these systems are not aimed at improving the usage model for MapReduce in a Cloud to better serve modern workloads with lower cost.

MapReduce Profile and Analyze tools: A number of MapReduce profiling tools

have been developed in the recent past with an objective of minimizing customer's cost in the cloud [80, 129, 101, 110, 133]. Herodotou et al. developed an automated performance prediction tool based on their profile and analyze tool *Starfish* [73] to guide customers to choose the best cluster size for meeting their job requirements [74]. Similar performance prediction tool is developed by Verma. et. al [133] based on a linear regression model with the goal of guiding customers to minimize cost. Popescu. et. al developed a technique for predicting runtime performance for jobs running over varying input data set [110]. Recently, a new tool called Bazaar [79] has been developed to guide MapReduce customers in a cloud by predicting job performance using a gray-box approach that has very high prediction accuracy with less than 12% prediction error. However, as discussed earlier, these job optimizations initiated from the customer-end may lead to requiring higher resources at the cloud. Cura while leveraging existing profiling research, addresses the challenge of optimizing the global resource allocation at the cloud provider-end with the goal of minimizing customer costs. As seen in evaluation, Cura benefits from both its cost-optimized usage model and its intelligent scheduling and online reconfiguration-based VM pool management.

2.6 Summary

In this chapter, we presented a new MapReduce cloud service model, Cura, for data analytics in the cloud. We argued that existing cloud services for MapReduce are inadequate and inefficient for production workloads. In contrast to existing services, Cura automatically creates the best cluster configuration for the jobs using MapReduce profiling and leverages deadline-awareness which, by delaying execution of certain jobs, allows the cloud provider to optimize its global resource allocation efficiently and reduce its costs. Cura's resource management techniques include cost-aware resource provisioning, VM-aware scheduling and online virtual machine reconfiguration.

While in this chapter we addressed the cost-inefficiencies of the current cloud

models, in the next chapter, we discuss on the job performance issues related to datacenter networking and storage bottlenecks and present our architecture and solution techniques to alleviate them.

CHAPTER III

LOCALITY-AWARE RESOURCE ALLOCATION FOR MAPREDUCE IN A CLOUD

3.1 Introduction

As discussed in Chapter 2, MapReduce offered as a service in the cloud provides an attractive usage model for enterprises. Using virtual machines (VMs) and storage hosted by the cloud, enterprises can simply create virtual MapReduce clusters to analyze their data. However, an important challenge for the cloud provider is to manage multiple virtual MapReduce clusters executing concurrently, a diverse set of jobs on shared physical machines. Concretely, each MapReduce job generates different loads on the shared physical infrastructure – (a) computation load: number and size of each VM (CPU, memory), (b) storage load: amount of input, output and intermediate data, and (c) network load: traffic generated during the map, shuffle and reduce phases. The network load is of special concern with MapReduce as large amounts of traffic can be generated in the shuffle phase when the output of map tasks is transferred to reduce tasks. As each reduce task needs to read the output of *all* map tasks [57], a sudden explosion of network traffic can significantly deteriorate cloud performance. This is especially true when data has to traverse greater number of network hops while going across *racks* of servers in the data center [24]. Further, the problem sometimes is exacerbated by TCP *incast* [109] with a recent study finding goodput of the network reduced by an order of magnitude for a MapReduce workload [42].

To reduce network traffic for MapReduce workloads, in this chapter, we argue for improved **data locality** for both Map and Reduce phases of the job. The goal is

to reduce the network distance between storage and compute nodes for both map and reduce processing – for map phase, the VM executing the map task should be *close* to the node that stores the input data (preferably local to that node) and for reduce phase, the VMs executing reduce tasks should be close to the map-task VMs which generate the intermediate data used as reduce input. Improved data locality in this manner is beneficial in two ways – (1) it reduces job execution times as network transfer times are big components of total execution time and (2) it reduces cumulative data center network traffic. While map locality is well understood and implemented in MapReduce systems, reduce locality has surprisingly received little attention in spite of its significant potential impact. As an example, Figure 13 shows the impact of improved reduce locality for a Sort workload. It shows the Hadoop task execution timelines for a 10 GB dataset in a 2-rack 20-node physical cluster, where 20 Hadoop VMs were placed without and with reduce locality (top and bottom figures respectively). As seen from the graph, reduce locality resulted in a significantly shorter shuffle phase helping reduce total job runtime by 4x.

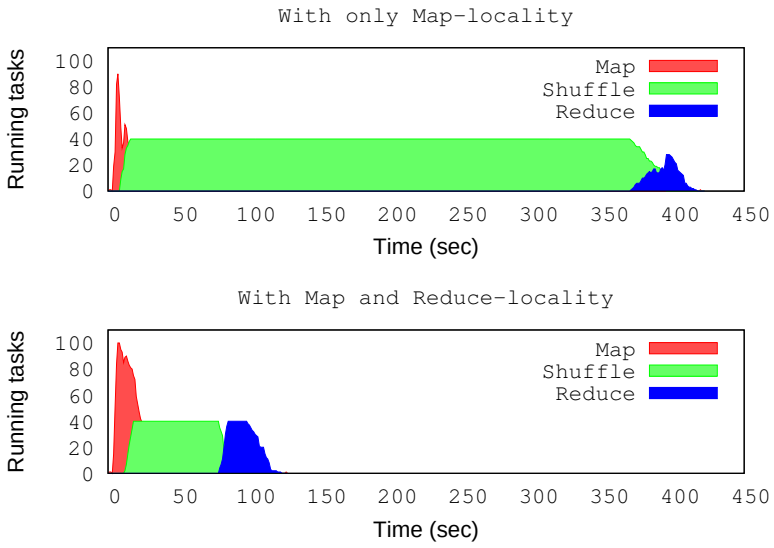


Figure 13: Impact of Reduce-locality. Timeline plotted using Hadoop's *job_history_summary*. *Merge* and *Waste* series are omitted since they were negligible

In this chapter, we present Purlieus – an intelligent MapReduce cloud resource allocation system. Purlieus improves data locality during both map and reduce phases of the MapReduce job by carefully coupling data and computation (VM) placement in the cloud. Purlieus categorizes MapReduce jobs based on how much data they access during the map and reduce phases and analyzes the network flows between sets of machines that store the input/intermediate data and those that process the data. It places data on those machines that can either be used to process the data themself or are close to the machines that can do the processing. This is in contrast to conventional MapReduce systems which place data independent of map and reduce computational placement – data is placed on any node in the cluster which has sufficient storage capacity [57, 4] and only map tasks are attempted to be scheduled local to the node storing the data block.

Additionally, Purlieus is different from conventional MapReduce clouds (e.g., Amazon Elastic MapReduce [9]) that use a separate compute cloud for performing MapReduce computation and a separate storage cloud for storing the data persistently. Such an architecture delays job execution and duplicates data in the cloud. In contrast, Purlieus stores the data in a dedicated MapReduce cloud and jobs execute on the same machines that store the data without waiting to load data from a remote storage cloud.

To the best of our knowledge, Purlieus is the first effort that attempts to improve data locality for MapReduce in a cloud. Secondly, Purlieus tackles the locality problem in a fundamental manner by coupling data placement with VM placement to provide both map and reduce locality. This leads to significant savings and can reduce job execution times by close to 50% while reducing up to 70% of cross-rack network traffic in some scenarios.

3.2 System Model

In this section, we provide a brief background on MapReduce and introduce our system model.

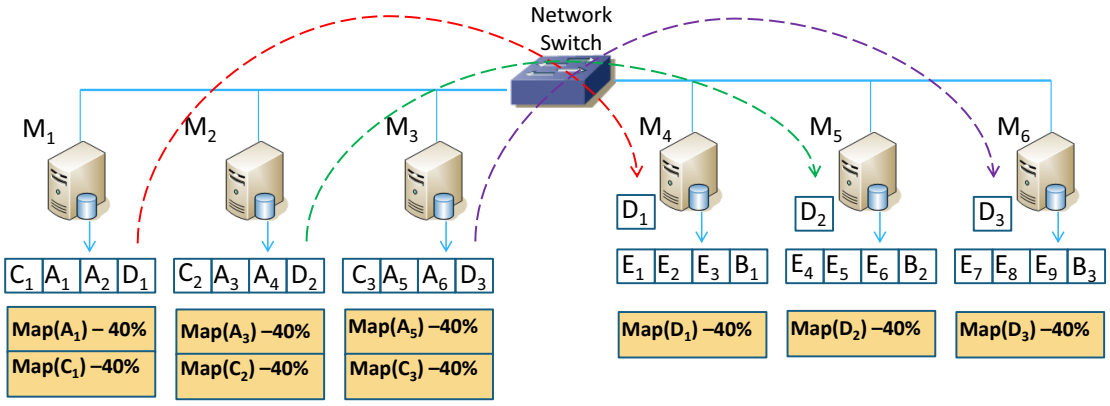
3.2.1 MapReduce: Background

A MapReduce job is comprised of two main components – a *map* function that processes key/value pairs from input data to generate a set of intermediate key/value pairs, and a *reduce* function that merges all intermediate values associated with the same intermediate key to generate the output [57]. A unique capability of this model is the execution of map and reduce tasks on a distributed cluster of machines, transparently to the application programmer. The input data is split into blocks that are stored in a distributed filesystem throughout the cluster. A cluster master then automatically schedules the map tasks at various worker nodes which process those blocks to create intermediate key/value pairs. Some of these input blocks may be present locally on the worker node while others may require a *remote-read* from another node. The map outputs are then scheduled to be processed by worker nodes for reduce tasks which write the output into files stored within the filesystem. The data transfer from the map to the reduce tasks includes a *shuffle* phase in which reducers read data from all mappers. This phase, in particular, can cause significant network traffic and perform poorly [24].

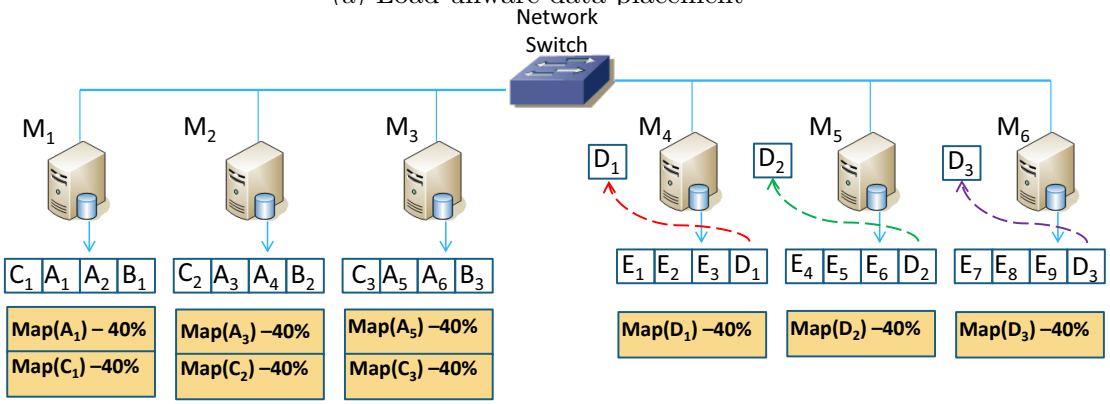
3.2.2 Model

In our system model, customers using the MapReduce cloud service load their input datasets and MapReduce jobs into the service. This step is similar to any typical cloud service which requires setting up of the application stack and data. There is one key distinction, however. Typically cloud service providers use two distinct infrastructures for storage and compute (e.g. Amazon S3 [11] for storage and Amazon EC2 [10] for compute). Executing a MapReduce job in such infrastructures requires

an additional *loading* step, in which data is loaded from the storage cloud into the distributed filesystem (e.g. Hadoop's HDFS) of the MapReduce VMs running in the compute cloud before even the job begins execution. Such additional loading has two drawbacks –(1) depending upon the amount of data required to be loaded and connectivity between the compute and storage infrastructures, this step adversely impacts performance, and (2) while the job is running (often for long durations) the dataset is duplicated in the cloud – along with the storage cloud original, there is a copy in the compute cloud for MapReduce processing, leading to higher costs for the provider. In contrast, we propose a dedicated MapReduce service, in which data



(a) Load-unware data placement



(b) Load-aware data placement

Figure 14: Load Awareness in Data placement

is directly stored on the same physical machines that run MapReduce VMs. This

prevents the need for a wasteful data loading step before executing a MapReduce job. We contend that since MapReduce input data is often predominantly used for MapReduce analysis, storing it into a dedicated cloud service provides the greatest opportunity for optimization.

In our proposed service, when customers upload their data into the service, the data is broken up into chunks corresponding to MapReduce blocks and stored on a distributed filesystem of the *physical* machines. The placement of data – deciding which machines to use for each dataset – is done intelligently based on techniques described later. When the job begins executing (i.e. MapReduce VMs are initialized), the key challenge is to make data stored in the physical machines seamlessly available to the MapReduce VMs running on those machines. We tackle this challenge using two specific techniques in our design – (1) *loopback mounts*: For a job, when its data is loaded into the cloud, the chunks being placed on each machine are stored via a loopback mount [14] into a single data file (we refer to it as a vdisk-file)[1] and (2) *VM disk-attach*: The vdisk-file is then attached to the VM as a block device using server virtualization tools (e.g. Xen's `xm block-attach` command[2]). The VM can then mount the vdisk file like it would any typical filesystem. The mount point of this vdisk-file inside the VM serves as the MapReduce DFS directory (e.g. Hadoop's `data.dir` configuration variable). Note that if a MapReduce VM is required to be placed on a physical machine other than the one containing that job's data chunks, the vdisk file is copied over to the appropriate physical machine and then attached to the VM – analogous to traditional MapReduce's *remote-read*. Also, similar to Amazon Elastic MapReduce [9] the same VMs are used for both map and reduce task execution. Note that when the job finishes execution, its VMs are de-provisioned

[1]To do this, a sparse empty file is created which is then loopback mounted and formatted using ext2 providing access similar to any local filesystem, even though all data is being stored in a single file on the physical filesystem.

[2]Similar commands exist for KVM and VMware

while the input data continues to reside on the physical machines.

3.3 Purlieus: Principles and Problem Analysis

Under our proposed system model, the cloud provider faces two key questions – (1) *Data Placement*: Which physical machines in the cluster should be used for each dataset? and (2) *VM Placement*: Where should the VMs be provisioned to process these data blocks? Poor placement of data or VMs may result in poor performance. Purlieus tackles this challenge with a unique coupled placement strategy, where data placement is aware of likely VM placement and attempts to improve locality. In this section, we describe the principles of our design and provide a formal analysis of the problem.

3.3.1 Principles

We argue that unlike traditional MapReduce, where data is placed independently of the type of job processing it or loads on the servers, in a multi-tenant virtualized cloud these attributes need to be accounted during data placement.

1. Job Specific Locality-awareness: *Placing data in the MapReduce cloud service should incorporate job characteristics* - specifically the amount of data accessed in the map and reduce phases. For example, a job that processes a lot of reduce data (referred to as a *reduce-input heavy* job) is best served by provisioning the VMs of MapReduce cluster close to each other on the cluster network, as each reducer reads the outputs of all mappers. If the VMs are far from each other, each reducer would have to read map outputs over longer network paths, which will increase job execution time and also increase cross-rack traffic in the data center. On the other hand, *map-input heavy* jobs that generate little intermediate data do not benefit by placing its data blocks close to each other on the cluster. An efficient data placement scheme could distribute data blocks for such a *map-input heavy* job across the network to

preserve resources for placing reduce-input heavy jobs on a set of closely connected machines.

Specifically, we use three distinct classes of jobs – (1) Map-input heavy (e.g. a large *grep* workload which generates small intermediate data simply indicating if a word occurs in input data), (2) Map-and-Reduce-input heavy (e.g. a *sort* workload: intermediate data is equal to input data) and (3) Reduce-input-heavy (e.g. a *permutation generator* workload which generates all permutations of given input strings). Purlieus uses different strategies of data placement for different job types with the goal of improving data locality[3].

2. Load Awareness: *Placing data in a MapReduce cloud should also account for computational load (CPU, memory) on the physical machines.* A good technique should place data only on machines that are likely to have available capacity to execute that job, else remote-reads will be required to pull data from busy machines to be processed at less-utilized machines. For example, in Figure 14(a), consider datasets A, B, C, D and E placed on six physical machines, M_1 to M_6. A load unaware placement may collocate the blocks of datasets A, C and D together as shown in Figure 14(a), even if jobs execute on A, C, D more frequently and generate higher load than B and E. Here, when the job on the dataset D arrives and requests for a virtual cluster of 3 VMs, say each with 40% CPU resources of the physical machine, even though it would be best to place the VMs on the physical machines, M_1, M_2 and M_3 as they contain the data blocks of the dataset D, the system may be forced to place the VMs on M_4, M_5 and M_6, resulting in remote reads for the job executing on dataset D. In contrast, the data placement shown in Figure 14(b) is able to achieve local execution for all the map tasks. Here, while placing the data blocks, it is made sure that the expected load on the servers does not exceed a particular

[3]For completeness a Map-and-Reduce-input light class can also be considered, however data locality has little impact on its overall performance

threshold. This incorporates the frequency and load generated by jobs executing on datasets stored on these servers.

It is important to note that information about expected loads is available to a cloud provider by monitoring the cloud environment. Typically with MapReduce, a set of jobs are repeatedly executed on a similar input data set – e.g. periodic execution of indexing on web crawled data. This allows the cloud provider to understand the load characteristics of such jobs and use this knowledge. Additionally, there are many proposals that *profile* MapReduce jobs via trial executions on a small subset of data [101, 32, 80, 40]. These show that understanding MapReduce job characteristics can be quick and accurate. For the scope of this work, we assume that the expected load on each dataset is known. Also that the cloud provider has enough data to estimate job arrival rate and the mean execution time. Later in Section 3.5, we will demonstrate that our proposed techniques perform well even when such estimates are partly erroneous.

3. Job-specific Data Replication: Traditionally, data blocks in MapReduce are replicated within the cluster for resiliency (by default, each block of the dataset is replicated 3 times). While the job is executing any replica of the block can be used for processing. Purlieus handles replicas in a different manner. Depending upon the type and frequency of jobs, we place each replica of the entire dataset based on a particular strategy. As an example, if an input dataset is used by three sets of MapReduce jobs, two of which are reduce-input heavy and one map-input heavy, we place two replicas of data blocks in a reduce-input heavy fashion and the third using map-input heavy strategy. This allows maintaining greater data locality, especially during the reduce phase, since otherwise by processing data block replicas *far* from other input data blocks during the map phase, the reducers may be forced to read more data over the network.

3.3.2 The Data Placement Problem

Next, we formally analyze the data placement problem. We start with notation for representing datasets, physical cloud infrastructure and their relationship.

Datasets and Jobs: Let $\mathcal{D} = \{D_i : 1 \leq i \leq |\mathcal{D}|\}$ be the set of datasets that need to be stored in the MapReduce cloud. For the sake of presentation simplicity, assume that each dataset is associated with only one MapReduce job-type and that the replication factor is 1[4] Each dataset D_i is divided into uniform sized blocks $B_{i,j} : 1 \leq i \leq |\mathcal{D}|, 1 \leq j \leq Q_i\}$ where $\mathcal{Q} = \{Q_i : 1 \leq i \leq |\mathcal{Q}|\}$ represent the number of blocks for D_i.

We assume that the job arrivals on the datasets follow a Poisson process and let $\lambda = \{\lambda_i : 1 \leq i \leq |\mathcal{D}|\}$ denote the arrival rate of the jobs on the datasets. After a job starts, it first executes map tasks. We denote the mean size of the expected map output of each block of dataset, D_i by $mapoutput(D_i)$.

Cloud Infrastructure: Let $\mathcal{M} = \{M_k : 1 \leq k \leq |\mathcal{M}|\}$ denote the set of physical machines. Each physical machine, M_k has some compute resources (CPU, memory) with capacity $Pcap(M_k)$[5] and some storage resources (disk) with capacity expressed in number of data blocks and denoted by $Scap(M_k)$. In the data center, the physical machines are connected to each other by a local area network. Let $dist(M_l, M_m)$ denote the *distance* between the physical machines M_l and M_m – we use the number of network hops as the *dist* measure.

Relationship Notation: Let $P_i \in \mathcal{M}$ be the set of servers used to store the dataset D_i and X_i^k be a Boolean variable indicating if the physical machine M_k is used to store the dataset Di. Therefore, $M_k \in P_i$ if X_i^k =1. Let $\mathcal{N} = \{N_i : 1 \leq i \leq |\mathcal{N}|\}$

[4]If the job is associated with multiple jobs of different job types, as mentioned earlier different replicas are used to support each type.

[5]Though we present a scalar capacity value, compute resources may have multiple dimensions like CPU and memory. To handle this, either our model can be extended to include a vector of resources or compute dimensions can be captured in a scalar value, e.g. the volume metric presented in [139].

denote the number of machines used to store D_i. Thus,

$$\sum_k X_i^k = N_i, \forall i$$

Within P_i, let $Y_{i,j}^k$ be the boolean variable indicating if the specific block $B_{i,j}$ is present in the physical machine $M_k \in \mathcal{M}$. Thus, in order to ensure that the blocks are evenly distributed among the nodes in P_i, we have

$$\forall i, k \sum_{1 \leq j \leq Q_i} Y_{i,j}^k = \frac{Q_i}{N_i}$$

Locality based Cost: To capture locality, we define a cost function that measures the amount of data transfer during job execution. Consider a job, A on the dataset, D_i. Let $V(A)$ be the set of physical machines that host the VMs for job A. The total cost of a MapReduce application is the sum of map and reduce costs that represent the overhead involved in the data transfers during the map and reduce phases.

$$Cost(A, D_i) = Mcost(A, D_i) + Rcost(A, D_i)$$

If $Snode(B_{i,j}) \in P_i$ is the physical machine storing the data block, $B_{i,j}$, and its map task gets scheduled on the physical machine, $Cnode(B_{i,j}) \in V(A)$, we consider

$$Mcost(A, D_i) = \sum_{1 \leq j \leq Q_i} size(B_{i,j}) \times dist(Snode(B_{i,j}), Cnode(B_{i,j}))$$

This cost definition captures the amount of data and the distance it travels over the network. Similarly, the reduce cost can be computed as the overhead involved in transferring the map outputs to the servers where the reducers are executed. As each reducer needs to see the output of all the map tasks, assuming the number of reducers is greater than the number of VMs used (as recommended by MapReduce) and each VM runs atleast one reducer, the map outputs need to be transferred to every physical machine used for the MapReduce job (i.e. $V(A)$). Therefore the reduce cost is given

by:

$$Rcost(A, D_i) = \sum_{1 \leq j \leq Q_i, m \in V(A)} dist(Cnode(B_{i,j}), M_m) \times m_{out}(A, B_{i,j})$$

where $m_{out}(A, B_{i,j})$ is the amount of output data generated by the map task on the data block, $B_{i,j}$. To improve locality, the goal is to minimize $Mcost$ and $Rcost$, subject to not violating the storage capacity constraint on physical machines

$$\forall k \ \sum_{i,j} Y_{i,j}^k \leq Scap(M_k)$$

Minimizing Map Cost: To minimize map cost, the computations should get placed on the same physical machines storing the map-input blocks (*dist* is zero). The *data placement* technique, in turn, should try to maximize the probability of such co-location. This is achieved by upper-bounding the expected resource load on the servers for hosting the VMs at any given time. By placing data blocks such that every server has a low expected utilization, there is higher probability that the server will be available to host a VM when a request for a job on the datasets arrives. Concretely, we model each physical machine, M_k as a $M/M/1$ single server queue. Let a dataset, D_i have a service time distribution with mean, μ_i, where μ_i is the mean time to process the blocks by each VM and $\rho_i = \frac{\lambda_i}{\mu_i}$. Therefore the expected number of jobs on the dataset D_i running on the physical machine M_k is given by

$$W_i^k = \frac{\rho_i}{\rho_i - \mu_i} . X_i^k$$

Now, the expected load on physical machine M_k is given by

$$E^k = \sum_i W_i^k \times CRes(D_i)$$

where $CRes(D_i)$ denotes the computational resource required by each VM of the job on D_i, given by the type of VM chosen by the user (e.g. Amazon EC2's *small* VM

instance that uses 1.7 GB memory and 1 vCPU). We upper-bound the expected load on any physical machine based on the load parameter, α.

$$\forall k, E^k \leq \alpha \times Pcap(M_k)$$

Here, a low value of α would indicate a conservative data placement where the expected load on the physical machines is less and therefore there is a high probability for a job on a data chunk on a physical machine to get executed locally.

Minimizing Reduce Cost: With the above method for minimizing map cost, now the key optimization is to improve reduce locality. At the time of data placement, the node used to host the VM that processes the data, $Cnode(B_{i,j})$ is not fixed. Hence *Rcost* can not be obtained precisely during data placement. Instead, we compute an estimated reduce cost during data placement – we assume that at the time of job execution, the VMs get placed on the physical machines storing the data block, which based on the previous map cost optimization should be likely. Now the optimization is

$$min \sum_i Rcost_{est}(A, D_i)$$

$$Rcost_{est}(A, D_i) = \sum_{1 \leq j \leq Q_i, m \in P_i} dist(Snode(B_{i,j}), M_m) \times mapoutput(D_i)$$

$mapoutput(D_i)$ is the mean size of the expected map output of each block of dataset, D_i. While being an estimate, this definition serves as a useful guideline for placement decisions, which as our evaluations show provides significant benefits.

It is easy to see that an optimal solution for this problem is NP-Hard – both data and VM placement involve bin-packing, which is known to be NP-Hard [62]. Therefore, we use a heuristics based approach, which is described next.

3.4 Purlieus: Placement Techniques

Next, we describe Purlieus's data and VM placement techniques for various classes of MapReduce jobs. The goal of these placements is to minimize the total *Cost* by reducing the *dist* function for map (when input data, Q_i is large) and/or reduce (when intermediate data, m_{out} is large).

3.4.1 Map-input heavy jobs

Map-input heavy jobs read large amounts of input data for map but generate only small map-outputs that is input to the reducers. For placement, mappers of these jobs should be placed close to input data blocks so that they can read data locally, while reducers can be scheduled farther since amount of map-output data is small.

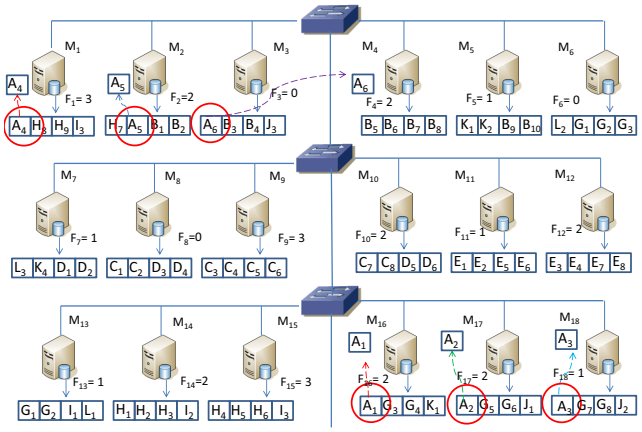


Figure 15: Placing Map-input heavy jobs

3.4.1.1 Placing Map-input heavy data

As map-input heavy jobs do not require reducers to be executed close to each other, the VMs of the MapReduce cluster can be placed anywhere in the data center. Thus, physical machines to place the data are chosen only based on the storage utilization and the expected load, E_k on the machines. As discussed in the cost model, E^k

denotes the expected load on machine, M_k.

$$E^k = \sum_i W_i^k \times CRes(D_i)$$

To store map-input heavy data chunks, Purlieus chooses machines that have the least expected load. This ensures that when MapReduce VMs are placed, there is likely to be capacity available on machines storing the input data.

3.4.1.2 VM placement for Map-input heavy jobs

The VM placement algorithm attempts to place VMs on the physical machines that contain the input data chunks for the map phase. This results in lower $MCost$ – the dominant component for map-input heavy jobs. Since data placement had placed blocks on machines that have lower expected computational load, it is less likely, though possible that at the time of job execution, some machine containing the data chunks does not have the available capacity. For such a case, the VM may be placed close to the node that stores the actual data chunk. Specifically, the VM placement algorithm iteratively searches for a physical machine having enough resources in increasing order of network distance from the physical machine storing the input data chunk. Among the physical machines at a given network distance, the one having the least load is chosen.

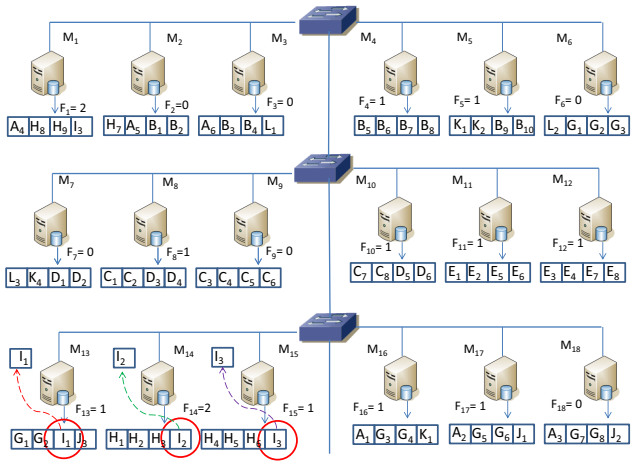
3.4.2 Map-and-Reduce-input heavy jobs

Map-and-reduce-input heavy jobs process large amounts of input data and also generate large intermediate data. Optimizing cost for such jobs requires reducing the *dist* function during both their map and reduce phases.

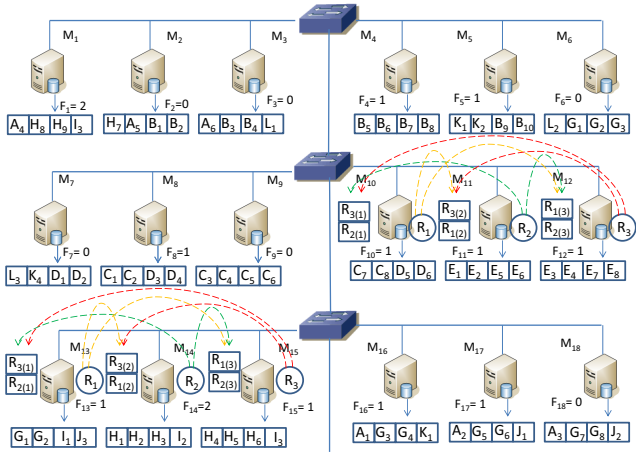
3.4.2.1 Placing Map-and-Reduce-input heavy data

To achieve high map-locality, data should be placed on physical machines that can host VMs locally. Additionally, this data placement should support reduce-locality – for which the VMs should be hosted on machines close to each other (preferably

within the rack) so that reduce traffic does not significantly load the data center network.



(a) Map-phase



(b) Reduce-phase

Figure 16: Data and VM placement for Map and Reduce-input heavy jobs.

Ideally, a subgraph structure that is densely connected, similar to a *clique*, where every node is connected to every other node in 1-hop would be a good candidate for placing the VMs. However, it may not always be possible to find cliques of a given size as the physical network may not have a *clique* or even if it does, some of the machines may not have enough resources to hold the data or their expected computational load may be high to not allow VM placement later. An alternate approach would be to

find subgraph structures similar to cliques. A number of clique relaxations have been proposed, one of which is *k-club* [99]. A *k-club* of a graph G is defined as a maximal subgraph of G of diameter k. While finding *k-club* is NP-Complete for a general graph, data center networks are typically hierarchical (e.g. *fat-tree* topologies) and this allows finding a *k-club* in polynomial time. In a data center tree topology, the leaf nodes represent the physical machines and the non-leaf nodes represent the network switches. To find a *k-club* containing n leaf nodes, the algorithm simply finds the sub-tree of height $\frac{k}{2}$ containing n or more leaf nodes.

For map-and-reduce-input heavy jobs, data blocks get placed in a set of closely connected physical machines that form a *k-club* of least possible k (least possible height of the subtree) given the available storage resources in them. If several subtrees exists with the same height, then the one having the maximum available resource is chosen. As an illustration, in Figure 16(a), the input data blocks, I_1, I_2, and I_3 are stored in a closely connected set of nodes M_{13}, M_{14} and M_{15} that form a *k-club* of least possible k in the cluster.

3.4.2.2 VM placement for Map and Reduce-input heavy jobs

As data placement had done an optimized placement by placing data blocks in a set of closely connected nodes, VM placement algorithm only needs to ensure that VMs get placed on either the physical machines storing the input data or the close-by ones. This reduces the distance on the network that the reduce traffic needs to go over, speeding up job execution while simultaneously reducing cumulative data center network traffic. In the example shown in Figure 16, VMs for job on dataset I get placed on the physical machines storing input data. As a result, map tasks use local reads (Figure 16(a)) and reduce tasks also read within the same rack, thereby maximizing reduce locality (Figure 16(b)). In case node M_{15} did not have available resources to host the VM, then the next candidates to host the VM would be M_{16},

M_{17} and M_{18}, all of which can access the input data block I_3 by traversing one network switch and are close to the other reducers executing in M_{13} and M_{14}. If any of M_{16}, M_{17} and M_{18} did not have available resources to host a new VM, then the algorithm would iteratively proceed to the next rack (M_7, M_8, M_9, M_{10}, M_{11} and M_{12}) and look for a physical machine to host the VM. Thus the algorithm tries to maximize locality even if the physical machines containing input data blocks are unavailable to host the VMs.

3.4.3 Reduce-input heavy Applications

Jobs that are reduce-input heavy read small sized map-inputs and generate large map-outputs that serve as the input to the reduce phase. For these type of jobs, reduce locality is more important than map-locality.

3.4.3.1 Placing Reduce-input heavy data

As map-input to these jobs is light, the map-locality of the data is not as important. Therefore, the map-input data can be placed anywhere within the cluster as it can be easily transferred to the corresponding VMs during map execution. The data placement algorithm chooses the physical machine with maximum free storage. The example in Figure 16(a) shows the placement of input data blocks for dataset L consisting of L_1, L_2 and L_3 on M_3, M_6 and M_7 which are chosen only based on storage availability, even though they are not closely connected.

3.4.3.2 VM placement for Reduce-input heavy jobs

Network traffic for transferring intermediate data among MapReduce VMs is intense in reduce-input heavy jobs and hence the set of VMs for the job should be placed close to each other. For an example job using the dataset, L, containing L_1, L_2, and L_3 in Figure 16(a), the VMs can be hosted on any set of closely connected physical machines, for instance, M_{10}, M_{11} and M_{12}. These machines are within a single rack

and form a *2-club* (diameter of 2 with a single network switch). Although the map phase requires remote reads from the nodes storing the input data, M_3, M_6 and M_7, it does not impact job performance much as the major chunk of data transfer happens only during the reduce phase. In the reduce phase, as VMs are placed in a set of densely connected nodes, the locality of the reads is maximized, leading to faster job execution.

3.4.4 Complexity of Techniques

There are two key operations used in our algorithms – (1) finding a *k-club* of a given size with available resources and (2) finding a node close to another node in the physical cluster. As noted before, with typical data center hierarchical topologies, both of these operations are very efficient to compute. As a result our techniques scale well with increasing sizes of datasets or the cloud data center.

3.5 Experimental Evaluation

We divide the experimental evaluation of Purlieus into two – first, we provide detailed micro-benchmarking on effectiveness of our data and VM placement techniques for each MapReduce job class on a real cluster testbed of 20 physical machines. Then, we present an extensive macro analysis with mix of job types and evaluate scalability of our approach on a large cloud scale data center topology through a simulator which is validated based with experiments on the real cluster. We first start with our experimental setup.

3.5.1 Experimental setup

Metrics: We evaluate our techniques on two key metrics with the goal of measuring the impact of data locality on the MapReduce cloud service – (1) *job execution time*: techniques that allow jobs to read data locally result in faster execution; thus this metric measures the per-job benefit of data locality, and (2) *Cross-rack traffic*:

techniques that read a lot of data across racks result in poorer throughput [24]; this metric captures such characteristics of the network traffic.

Data Placement Techniques: We compare two data placement schemes – our proposed *locality and load-aware data placement (LLADP)* accounts for MapReduce specific job characteristics and estimated loads on servers while placing data as described in Section 3.4. In contrast, the *random data placement* (RDP) scheme does not differentiate between job categories and places data blocks in a set of randomly chosen physical machines that have available storage capacity. It also has no knowledge of the server loads (analogous to conventional MapReduce data placement). Note that both the locality-aware and random data placement schemes are rack-aware [15]; no two replicas of a given data block are placed on the same cluster rack for reliability purposes.

VM Placement Techniques: We compare five techniques:

- *Locality-unaware VM Placement(LUAVP)*: LUAVP places VMs on the physical machines without taking into consideration the locations of the input data blocks for the job. The LUAVP scheme does try to pick a set of least loaded physical machines for placing the VMs.

- *Map-locality aware VM placement (MLVP)*: MLVP considers locality of only the input-data blocks for the map phase and considers the current load and resource utilization levels of the machines while placing the VMs (load-aware).

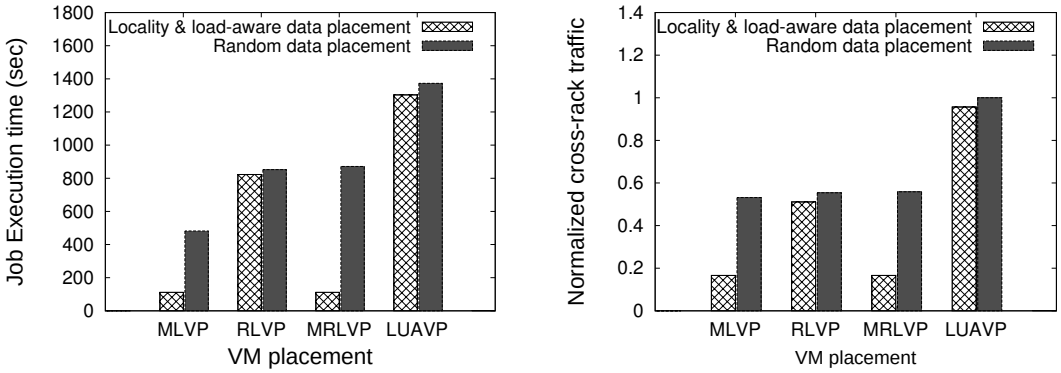
- *Reduce-locality aware VM placement (RLVP)*: RLVP does not consider map locality, but it tries to improve reduce locality by packing VMs in a set of closely connected machines. It is also load aware.

- *Map and Reduce-locality aware VM placement (MRLVP)*: MRLVP is aware of both map and reduce locality and is also load aware.

- *Hybrid locality-aware VM placement (HLVP)*: Our proposed HLVP technique adaptively picks the placement strategy based on type of the input job. It uses MLVP for map-input heavy, RLVP for reduce-input heavy jobs and MRLVP for map and reduce-input heavy jobs.

Key Comparison: The important comparison is between the combination of LLADP + HLVP (Purlieus proposal) with RDP + MLVP – analogous to traditional MapReduce. The other techniques help us understand the benefits of individual map or reduce locality as well as benefits gained from data vs. VM placement.

Cluster Setup: Our cluster consists of 20 CentOS 5.5 physical machines (KVM as the hypervisor) with 16 core 2.53GHz Intel processors. The machines are organized in two racks, each rack containing 10 physical machines. The network is 1 Gbps and the nodes within a rack are connected through a single switch. Each job uses a cluster of 20 VMs with each VM configured with 4 GB memory and 4 2GHz vCPUs. A description of the various job types and the dataset sizes is shown in Table 8. Each workload uses 320 map tasks. The *Grep* workload uses only one reducer since it requires little reduce computation while the *Sort* and *Permutation Generator* workloads use 80 reducers. The Hadoop parameter, *mapred.tasktracker.map.tasks.maximum* that controls the maximum number of map tasks run simultaneously by a task tracker is set as 5. Similarly, the *mapred.tasktracker.reduce.tasks.maximum* parameter is set as 5. Similar to typical data center topologies, the inter-rack link between the two switches becomes the most contentious resource as all the VMs hosted on a rack transfer data across this link to the VMs hosted on the other rack. For example, with 10 physical machines on each rack, and each physical machine hosting a nominal 8 VMs, 80 VMs (and thus, Hadoop nodes) on each rack will contend for the inter-rack link bandwidth of 1 Gbps. To simulate this contention in a more controlled environment that lets us accurately measure per-job improvements, we set the bandwidth of the inter-rack link to 100 Mbps while running one job at a time. The other alternative would be



(a) Job Execution Time (b) Network Utilization

Figure 17: Map and Reduce-input heavy workload

to run multiple jobs at the same time on the cluster, however, that would have made micro analysis on a per-job type basis tougher to evaluate.

Workload Type	**Job**	**Input data**	**Output data**
Map-input heavy	Grep: word Search	20 GB	2.43 MB
Reduce-input heavy	Permutation Generator	2 GB	20 GB
Map and Reduce-input heavy	Sort	10 GB	10 GB

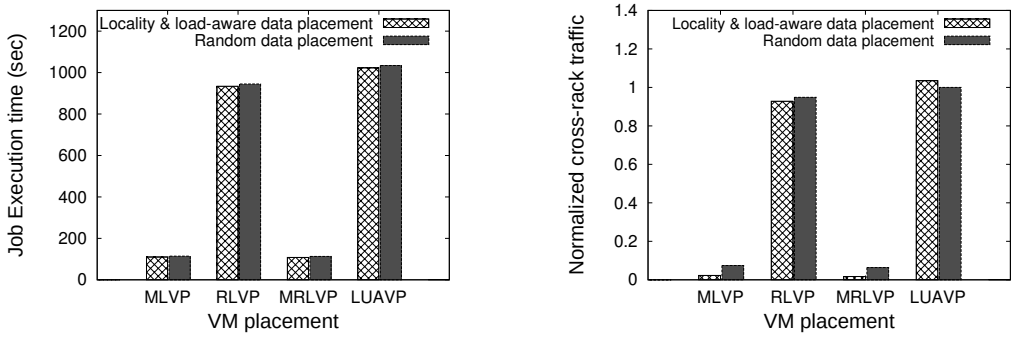
Table 8: Workload types

3.5.2 Micro-benchmarking Results

We first present evaluation of our proposed techniques for various MapReduce job types.

3.5.2.1 Map and Reduce-input heavy workload

In Figure 17, we study the performance for jobs that are both Map and Reduce-input heavy using the *Sort* workload on a dataset generated using Hadoop's RandomWriter. The job execution time in Figure 17(a) for map-and-reduce VM placement with locality and load-aware data placement (LLADP + MRLVP) shows the least value among all schemes with more than 76% reduction compared to RDP + MLVP. For data



(a) Job Execution Time

(b) Cross-rack traffic

Figure 18: Map-input heavy workload

placement, MRLVP with RDP performs poorly indicating that *without a locality-aware data placement, it is hard to achieve high locality during VM placement* and therefore leads to higher job execution time. This justifies our coupled data placement and VM scheduling technique.

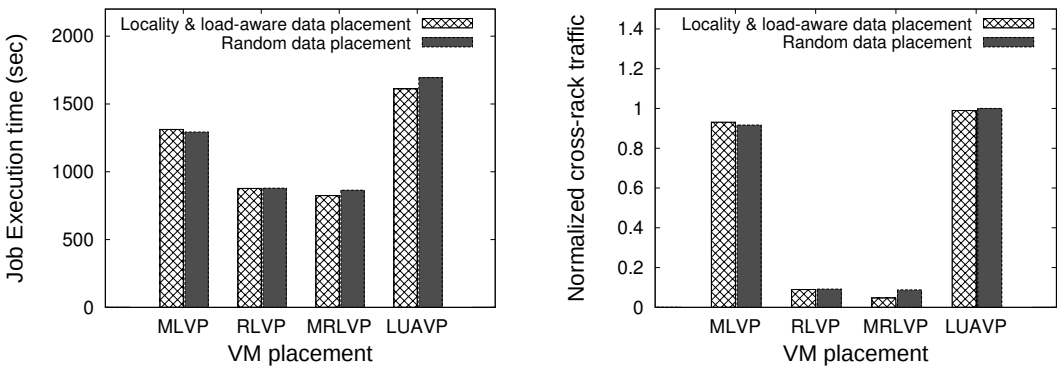
Also, RLVP does not perform well as it tries to consider only reduce locality. The LUAVP scheme places the VMs randomly without considering locality and therefore does not perform well either. An interesting trend here is that MLVP performs well with LLADP as the locality-awareness in data placement tried to place the data in a set of closely connected physical machines and hence, when the map-locality aware VM placement tries to place the VMs close to the input data, the reduce-locality is implicitly accounted for. These benefits can be explained by the trend in cross-rack traffic (normalized with respect to RDP + LUAVP) in Figure 17(b), showing 68% lesser cross rack reads when using LLADP + MRLVP compared to RDP + MLVP.

3.5.2.2 Map-input heavy workload

Next we evaluate data and VM placement for map-input heavy jobs using the *Grep* workload. Figure 18 compares our metrics with various schemes. In Figure 18(a), first notice that the job execution time for the locality-unaware VM placement (LUAVP) and reduce-locality aware VM placement (RLVP) schemes is much higher than that

of map-locality aware (MLVP) and map-and-reduce locality aware (MRLVP) VM placements for both the random (RDP) and locality and load-aware data placement (LLADP) schemes. As map-input heavy jobs generate only small map-outputs and have little reduce traffic, the *techniques that optimize for map locality – MLVP and MRLVP perform much better than the reduce-locality only technique (RLVP) (up to 88% reduction in job execution time).* The job execution time difference can be explained by cross-rack network traffic (Figure 18(b)), normalized with respect to RDP + LUAVP, shows that map-locality awareness has a big impact. Lower cross-rack network traffic suggests that the data reads are more local to the rack, avoiding more than 95% of cross-rack traffic.

3.5.2.3 Reduce-input heavy workload



(a) Job Execution Time (b) Cross-rack traffic

Figure 19: Reduce-input heavy workload

Figure 19 shows the performance for reduce-input heavy workload using a permutation generator job that generates and sorts the first 10 permutations of each record of a dataset generated by Hadoop's RandomWriter. We find in Figure 19(a) that RLVP and MRLVP have lower execution time for both the random (RDP) and locality-aware data placement (LLADP), having up to 32% faster execution time when compared to RDP + MLVP. Reduce-locality awareness in VM placement ensures that

the reducers are packed close to each other and reduce traffic does not traverse a long distance on the network. Here, the underlying data placement scheme makes little impact as these jobs do not have large input data, so violating map locality does not cost much.

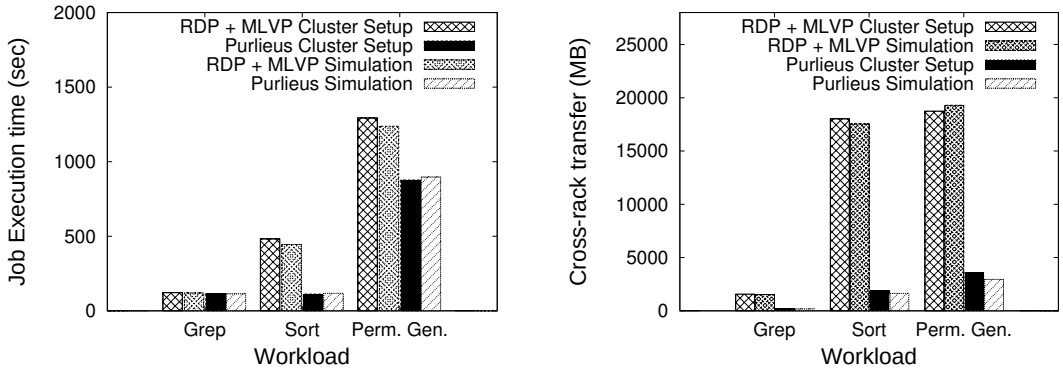
The LUAVP and MLVP schemes perform poorly since they do not capture reduce locality which is key for this reduce-intensive workload. A similar trend is seen for the ratio of cross-rack reads in Figure 19(b), where the (LLADP + MRVLP) technique has 10x+ higher number of reads within racks as compared to RDP + MLVP.

Summary: *This micro-analysis demonstrates that data and VM placement techniques when applied judiciously to MapReduce jobs can have a significant impact on the job execution time as well as total datacenter traffic. To realize these benefits, the right technique needs to be applied for each MapReduce job type. Our Purlieus technique (LLADP + HLVP) identifies and uses the right strategy for each type of workload.*

3.5.3 Macro Analysis: Mix of workloads, Scalability and Efficiency

Following the per-job-type analysis, next we consider a mix of workloads and evaluate the scalability of the techniques with respect to the size of data center network and number of VMs in virtual MapReduce clusters using a mix of workload types.

For a thorough analysis at scales of 100s and 1000s of machines and with varying job, workload and physical cloud characteristics, we implemented a MapReduce simulator, called PurSim, similar to the existing NS-2 based MRPerf simulator [134]. However, unlike MRPerf, PurSim does not perform a packet-level simulation of the underlying network. Per-packet approach simulates every single packet over the network which makes it difficult to scale for even reasonably large workloads and cluster sizes. For instance, a per-packet simulator for a cluster size of 1000 hosts sending traffic at 1Gbps would generate 3×10^{10} packets for a 60 second simulation and



(a) Job Execution Time (b) Cross-rack traffic

Figure 20: Simulator Validation

simulating a million packets per second would take 71 hours to simulate just that one case [23]. Instead we use a *network flow* level simulation. Our discrete event simulator simulates the MapReduce execution semantics similar to the Hadoop implementation. The inter-node traffic is simulated in terms of network flows between the source-destination pairs. The simulation framework uses a data center of 1000 compute nodes with 1 Gbps network configured in the typical tree topology for the default setting. The performance metrics were averaged over the jobs executed during a 2 hour simulation period. By default, we use a mixed workload of jobs consisting of equal proportions of all MapReduce job types in Table 8. We use a 30 GB dataset for both the *Grep* and *Sort* workloads and a 2 GB dataset for *Permutation* workload. For the default setting, a total of 150 datasets were used, 50 for each of the job types and 3 replicas were created by default. The arrival rate of the jobs on the datasets is uniformly distributed from 200 to 2000 seconds.

3.5.3.1 Simulator Validation

Before presenting our simulation experiments, we provide a validation of the simulator based on the experiments on our real 20 node cluster. To bootstrap the simulator, we used measurements obtained from the cluster experiments to configure simulator

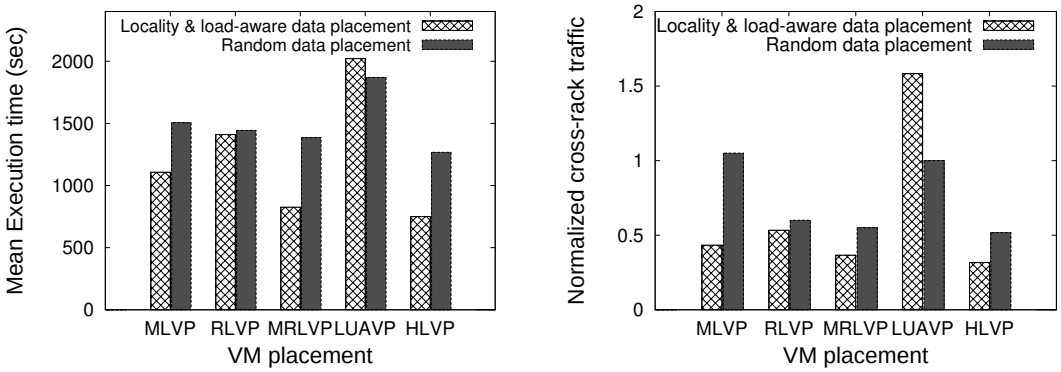
parameters, e.g. map and reduce compute times. We used the same settings from our cluster setup including the cluster network topology and workload characteristics in Table 8. As the key comparison is between the (RDP + MLVP) and Purlieus (LLADP + HLVP) schemes, we compare these two techniques for various job types.

In figure 20(a), we compare the job execution time of the two schemes for the three workloads. We find that for most cases, the execution time produced by the simulator is within 10% of the execution time obtained in our cluster experiments. The cross-rack transfer in Figure 20(b) shows that the simulator estimated cross-rack transfer matches closely with that of our cluster experiments, having less than 5% error in the cross-rack transfer estimated by the simulator. While not validated against large scale clusters, these low error rates when compared to our 20-node cluster experiments, provide good confidence in the quality of the simulator.

3.5.3.2 Mixed workload

For our first macro analysis, we study the performance with a composite workload that consists of an equal mix of all MapReduce job categories with the default setting of 150 datasets and jobs using 20 VMs per job. Recall that Purlieus's HLVP decides on the placement policy based on the type of MapReduce job. For example, it uses RLVP for reduce-input heavy jobs and MRLVP for jobs that are both map and reduce heavy. The execution time in Figure 21(a) shows that HLVP works best for a mixed workload compared to all other VM placement policies. As discussed earlier, a reduce-locality aware VM placement would lose map locality for map-input heavy jobs and a map-locality VM placement might lose reduce-locality while trying to achieve map-locality. While the map and reduce locality-aware VM placement could be a conservative policy for all types of jobs, it may not be needed in all cases and in fact may use valuable dense-collection of machines for jobs that do not need it. This explains the difference between HLVP and MRLVP. – HLVP uses the right kind of

resources for each job type. Overall, *HLVP with LLADP shows 2x faster execution time when compared to RDP + MLVP schemes* and a 9.1% improvement with most conservative policy of LLADP + MRVLP. Figure 21(b) shows the same trend with the normalized cross-rack traffic – LLADP + HLVP shows a lower cross-rack traffic (only 30.1%) compared to the RDP + MLVP. Overall, it is vivid that *with random data placement, it is hard to achieve a higher ratio of rack-local reads no matter what VM placement algorithm is used, thus validating our claim made in the Purlieus design.*

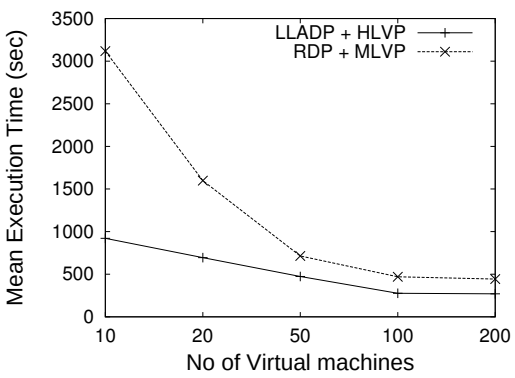


(a) Job Execution Time (b) Cross-rack traffic

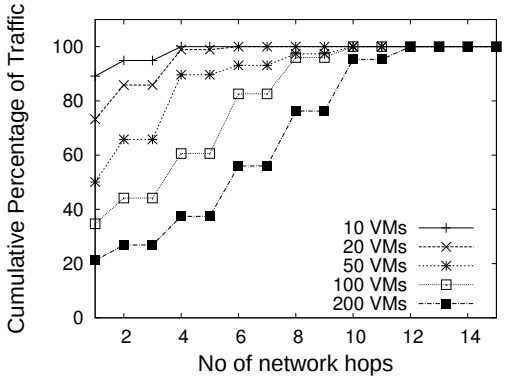
Figure 21: Mixed workload

3.5.3.3 Impact of number of VMs

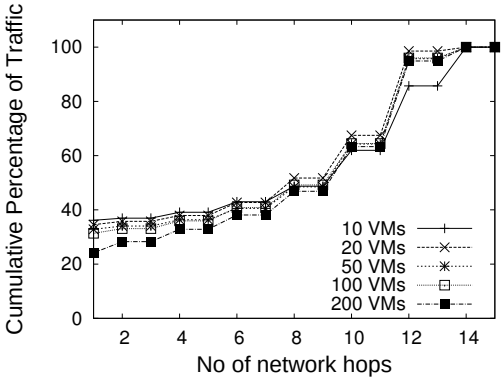
We study the impact of varying the number of VMs used for a given job in Figure 22 using the default PurSim setting. In figure 22(a), the number of VMs is varied from 10 to 200 and the average job execution time is compared. The job execution time decreases with increasing number of VMs but that decrease almost stops beyond a certain number of VMs (100 VMs in this case). The initial increase in number of VMs increases the computational parallelism and improves execution time. But as the number of VMs exceed a certain value, the reduce tasks get distributed across the network since not all of them can be placed on a set of closely connected machines



(a) Job Execution time



(b) CDF of Traffic with LLADP and HLVP



(c) CDF of Traffic with RDP and MLVP

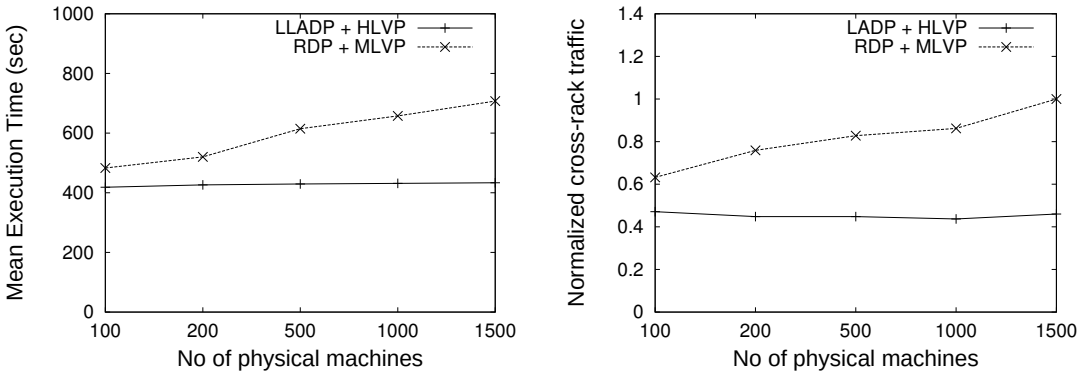
Figure 22: Varying number of Virtual Machines

(racks get exhausted). This reduction in data locality and increased network transmission time counters the improved parallelism. This also shrinks the advantage of Purlieus approach over RDP + MLVP. For instance, there is a performance gain of 2.3x in execution time while using 20 VMs and it drops down to a gain of 1.7x when 100 VMs are used. This is expected since when a large virtual cluster is provisioned, it is tough to provide both map and reduce locality. This impact can be further analyzed by visualizing the CDFs of number of network hops in both schemes with varying number of VMs in Figures 22(b) and 22(c). When number of VMs increase, there are more reads over longer network paths. However, we always find higher percentage of closer reads with (LLADP + HLVP) compared to (RDP + MLVP).

Overall, there are two key take-aways. First, Purlieus approach outperforms other approaches for varying sizes of virtual MapReduce clusters per job. Secondly, we notice that for a given job and cloud topology, there is a sweet-spot in the size of the virtual MapReduce cluster which gives the most bang for the buck. A tool that helps customers identify this would be very valuable.

3.5.3.4 Varying Network Size

In this experiment, we measure the job execution time and cross-rack traffic for various sizes of the cloud topology using 50 VMs for each job. The other parameters are based on PurSim's default setting. The job execution time in figure 23(a) is fairly constant for various network sizes with LLADP + HLVP. However, with RDP + MLVP, the data blocks gets distributed all over the network and with bigger clusters, the VMs are spread across the network and hence the reduce phase obtains poor locality leading to longer execution times. The normalized cross-rack traffic in Figure 23(b) is also indicative of the same trend. Thus, *Purlieus techniques work well with varying size of the cloud datacenter topology while conventional technique perform worse for larger network topologies.*

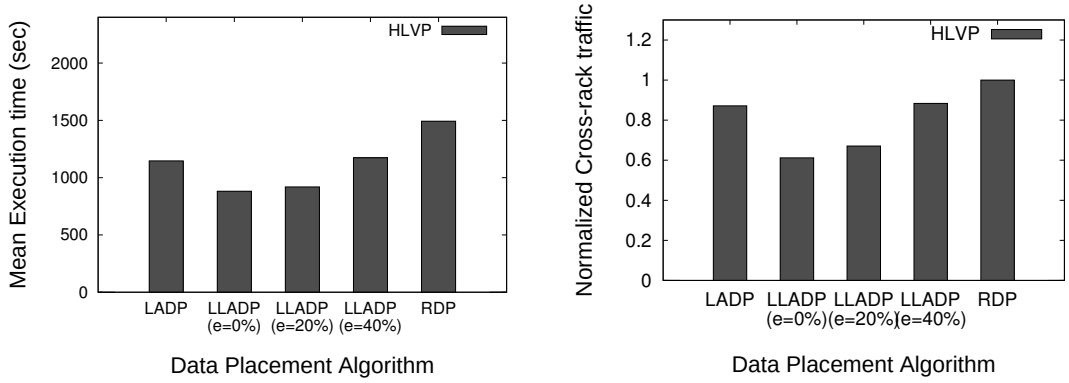


(a) Job Execution time (b) Cross-rack traffic

Figure 23: Varying Network Size

3.5.3.5 Impact of Load-aware Data Placement

Our next experiment evaluates the effectiveness of load-awareness in data placement. The experiments use the workload in the default PurSim setting using 20 VMs for each job. A good load-aware technique should make good decisions even with reasonably accurate estimates. We study the locality and load aware data placement (LLADP) with only locality-aware data placement (LADP) and random data placement (RDP). Figure 24 compares the LADP scheme with RDP and LLADP scheme for several load estimation error values, e. The estimation error, e directly corresponds to the percentage error in the estimation of the job arrival rates on the datasets. In figure



(a) Job Execution time (b) Cross-rack traffic

Figure 24: Load Aware Data placement

24(a), we find that without any load estimation error, LLADP ($e = 0\%$) performs better than the load-unaware (LADP) and random placement (RDP) schemes. Also, we find that even with an estimation error of 20 % or 40%, the LLADP scheme performs better than the random and load-unaware (LADP) schemes. A similar trend is seen in Figure 24(b) for the cross-rack traffic normalized with respect to (RDP + HLVP). It suggests that *even an approximate estimate of the arrival rate of the jobs on the datasets helps balance the expected load among physical machines and increases data locality.*

3.6 Related Work

To the best of our knowledge, Purlieus, with its coupled data and VM placement, is unique in exploiting both map and reduce locality for MapReduce in a cloud. We briefly review some of the related work in this area. There have been several efforts that investigate efficient resource sharing while considering fairness constraints [13]. For example, Yahoo's capacity scheduler uses different job queues, so each job queue gets a fair share of the cluster resources. Facebook's fairness scheduler aims at improving the response times of small jobs in a shared Hadoop cluster. Sandholm et al [115] presented a resource allocation system using regulated and user-assigned priorities to offer different service levels to jobs over time. Zaharia et al. [142] developed a scheduling algorithm called LATE that attempts to improve the response time of short jobs by executing duplicates of some tasks in a heterogenous system. Herodotou et al. propose *Starfish* that improves MapReduce performance by automatically tuning Hadoop configuration parameters [73]. The techniques in Purlieus are complementary to these above mentioned optimizations. Recent work, *Mantri*, tries to minimize outliers by making network-aware task placement, task restarting and protecting the output of valuable tasks [24]. It also identifies that cross-rack traffic during the reduce phase is a crucial factor for MapReduce performance. However, without a locality-aware data placement scheme in *Mantri*, there are only limited opportunities for optimizations during task placement. Purlieus solves the fundamental problem of optimizing data placement so as to obtain a highly local execution of the jobs during scheduling, minimizing the cross-rack traffic during both map and reduce phases. As seen in evaluations, Purlieus benefits from its locality-aware data as well as computation placement.

A large body of work has explored the placement of applications in a virtualized data center to minimize energy consumption [132], perform load balancing [121]

or perform server consolidation [83]. These approaches primarily focus on the *bin-packing* aspect and place applications (VMs) independent of the underlying data placement. Purlieus differs from these in terms of its consideration of both input and intermediate data locality for MapReduce. Recently, motivated by MapReduce, there has been work on resource allocation for data intensive application, especially in the cloud context [85, 68]. Gunarathne et al.[68] present a new MapReduce runtime for scientific applications built using Microsoft Azure cloud infrastructure services. Tashi [85] identifies the importance of location awareness but does not propose a complete solution. Tara [88] presents an architecture for optimized resource allocation using a genetic algorithm. Quincy [78] is a resource allocation system for scheduling concurrent jobs on clusters, but it considers only input data locality and does not optimize for locality of any intermediate data generated during job execution which is a key factor to scaling MapReduce in large data centers. Purlieus differentiates from these through its locality optimizations achieved for both input and intermediate data. Also, as discussed in Section 3.4, without an efficient underlying data placement, even a sophisticated locality-aware compute placement may not be able to achieve high data locality.

3.7 Summary

In this chapter, we illustrated how locality-awareness in data and virtual machine placement impact on the performance of the jobs and on the total network traffic in the data center. We presented a coupled compute-storage cloud architecture for MapReduce and developed a suite of data and VM placement techniques that achieves high data locality. Our detailed evaluation showed significant performance gains with some scenarios showing close to 50% reduction in execution time and upto 70% reduction in the cross-rack network traffic.

In the next chapter, we investigate locality optimizations for MapReduce when

data and compute can not be collocated in a cloud datacenter such as when the cloud consumer has privacy concerns in processing the stored data within the cloud.

CHAPTER IV

VNCACHE: MAPREDUCE ANALYSIS FOR CLOUD-ARCHIVED DATA

4.1 Introduction

Data Storage requirements have seen an unprecedented growth owing to more stringent retention requirements and longer term mining potential of data. Enterprises have adopted various techniques such as tiering, archiving [112], Deduplication [38] and remote cloud storage to combat data growth with varying degrees of success. For certain classes of enterprise data - application and infrastructure logs, enterprises are increasingly resorting to cloud storage solutions mainly due to lower cost and expandable on demand nature of solutions.

Logs often contain sensitive information like IP addresses, login credentials, etc. which necessitate encrypting the data before it leaves the enterprise premises. Temporal mining of archived data is gaining increased importance in a variety of domains for multitude of use cases such as fault isolation, performance trending, pattern identification,etc. After securely archiving data in the storage cloud, extracting any business value for the above mentioned use cases from this data using any analytics platforms such as MapReduce[57] or Hadoop[4] is non trivial. Exploiting compute resources in the public cloud for such analytics is often not an option due to security concerns. Most state-of-the-art cloud solutions today are highly sub-optimal for such use-cases. All (encrypted) data sets need to be first transferred to the enterprise cluster from remote storage clouds, decrypted, and then loaded into the Hadoop Distributed File System (HDFS)[16]. It is only after these steps complete that the job will start executing. Secondly, this results in extremely inefficient storage utilization. While the

job is executing, the same dataset will reside in both the public storage cloud and the enterprise cluster and is in fact replicated multiple times at both of these places for resiliency purposes, resulting in higher costs. For example, Hadoop by default will replicate the data 3 times within the enterprise Hadoop cluster. This is on top of the storage replication cost incurred at the public storage cloud.

In this chapter, we propose a unique hybrid cloud platform called VNcache that alleviates the above mentioned concerns. Our solution is based on developing a virtual HDFS namespace for the encrypted data stored in the public storage cloud that becomes immediately addressable in the enterprise compute cluster. Then using a seamless streaming and decryption model, we are able to interleave compute with network transfer and decryption resulting in efficient resource utilization. Further by exploiting the data processing order of Hadoop, we are able to accurately prefetch and decrypt data blocks from the storage clouds and use the enterprise site storage only as a cache. This results in predominantly local reads for data processing without the need for replicating the whole dataset in the enterprise cluster.

Uniquely we accomplish this without modifying any component of Hadoop. By integrating VNCache into the filesystem under HDFS, we are able to create a new control point which allows greater flexibility for integrating security capabilities like encryption and storage capabilities like use of SSDs. Our experimental evaluation shows that VNCache achieves up to 55% reduction in job execution time while enabling private data to be archived and managed in public clouds.

The rest of the chapter is organized as follows. Section 4.2 provides the background and the use-case scenario for supporting MapReduce analysis for Cloud-archived. In Section 4.3, we present the design of VNCache and its optimization techniques. We discuss our experimental results in Section 4.4 and we present a discussion of alternate solutions and design choices for VNCache in Section 4.5. In Section 4.6, we discuss related work and we summarize in Section 4.7.

4.2 Background

We consider enterprise applications that perform MapReduce analysis over log data that get generated at the enterprise site and archived in a public cloud infrastructure. For example, an application that monitors the status of other application software and hardware typically generates enormous amounts of log data. Such log data is often associated with a timestamp and data analysis needs to performed on them periodically.

With current cloud storage solutions, the logical method to perform analysis of archived data would be as follows. Log data generated at the enterprises would be encrypted and archived at a possibly nearest public storage cloud. Upon a need to execute a Hadoop analytics job, the enterprise cluster would download all relevant input data from the public clouds (time for which depends on WAN latencies). It will then create a virtual Hadoop cluster by starting a number of VMs. Data is then decrypted locally (time for which depends on CPU/Memory availability on local nodes and denoted by Decryption Time) and then ingested into HDFS of the Hadoop cluster (HDFS Ingestion Time) and then the job can start executing (Hadoop Job Execution Time). Upon finishing the job, local copy of the data and the virtual Hadoop cluster can be destroyed.

Figure 25 shows the breakdown of execution time for running a grep hadoop job on a 5GB dataset using the conventional execution model mentioned above. The network latencies 45, 90 and 150 milliseconds represent various degrees of geographic separation such as co-located datacenters, same coast data centers, and geographically well-separated data centers. Results show that WAN copy time and HDFS load time can have significant impact on overall execution time, thus making this model inefficient.

Further, depending upon the amount of data required to be loaded and connectivity between the enterprise cluster and the remote storage cloud infrastructure, this

step adversely impacts performance, and while the job is running (often for long durations) the dataset is duplicated in both the public storage cloud as well as the local enterprise cluster– along with the storage cloud original, there is a copy in the enterprise cluster, leading to higher costs for the enterprise.

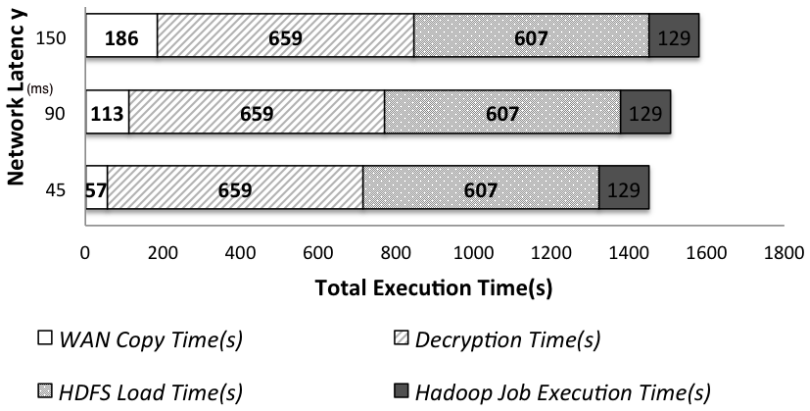


Figure 25: Breakdown of Fullcopy Runtime: 5 GB dataset with varying network latency

In contrast to this conventional model, VNCache aims at minimizing the impact of *apriori* data ingestion and decryption steps by intelligent pipelining of compute with those steps; specifically, by creating a *virtual* HDFS namespace which lays out the HDFS data blocks across the compute cluster. Whenever the job needs to access any data block, VNCache streams it on-demand from the appropriate storage clouds, decrypting it on-the-fly, and making it available to the job. As an additional performance optimization, VNCache prefetches data ahead of processing so that the map tasks read the data from local storage.

We next present the design overview of VNCache and describe its various components.

4.3 Design Overview

We start with a brief overview of HDFS and its interaction with the underlying filesystem.

4.3.1 HDFS and underlying filesystem

Hadoop Distributed Filesystem (HDFS) is a distributed user-space filesystem used as the primary storage by Hadoop applications. A HDFS cluster consists of a Namenode that manages filesystem metadata and several Datanodes that store the actual data as HDFS blocks. HDFS is designed to be platform independent and can be placed on top of any existing underlying filesystem (like Linux ext3) on each node of the cluster. It follows a master/slave architecture. HDFS exposes a file system namespace and allows user data to be stored in files. The HDFS Namenode manages the file system namespace and regulates access to files by clients. The individual Datanodes manage storage attached to the nodes that they run on. When a client writes a file into HDFS, the file is split into several smaller sized data blocks (default size is 64 MB) and stored on the storage attached to the Datanodes.

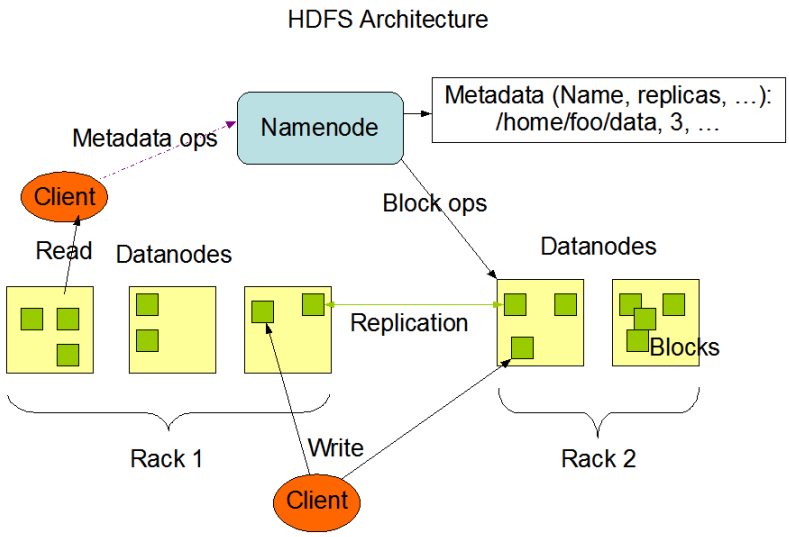


Figure 26: HDFS Architecture

Within the cluster, the Namenode stores the HDFS filesystem image as a file called *fsimage* in its underlying filesystem. The entire HDFS filesystem namespace, including the mapping of HDFS files to their constituent blocks, is contained in this

file. Each Datanode in the cluster stores a set of HDFS blocks as separate files in their respective underlying filesystem[1]. As the Namenode maintains all the filesystem namespace information, the Datanodes have no knowledge about the files and the namespace. As a HDFS cluster starts up, each Datanode scans its underlying filesystem and sends a Block report to the Namenode. The Block report contains the list of all HDFS blocks that correspond to each of these local files.

When an application reads a file in HDFS, the HDFS client contacts the Namenode for the list of Datanodes that host replicas of the blocks of the file and then contacts the individual Datanodes directly and reads the blocks from them. We refer the interested readers to [16] for a detailed documentation on the design and architecture of the Hadoop Distributed Filesystem.

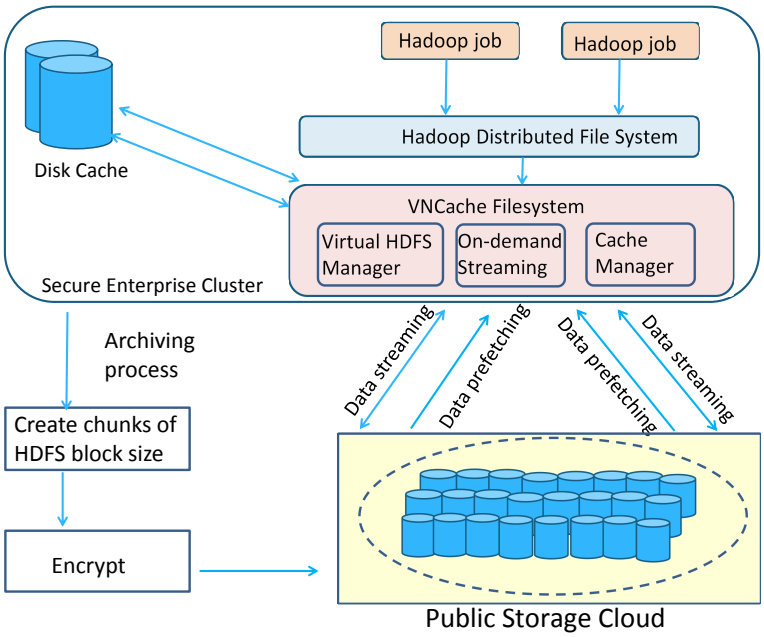


Figure 27: System Model

4.3.2 VNCache overview

VNCache is a FUSE based filesystem [19] used as the *underlying* filesystem on the

[1]Location in the underlying filesystem is determined by the *dfs.data.dir* configuration setting

Namenode and Datanodes of the HDFS cluster. It is a virtual filesystem (similar to */proc* [18] on Linux) and simulates various files and directories to the HDFS layer placed on it. For the Namenode, VNCache exposes a virtual HDFS namespace with an artificially constructed *fsimage* file and for Datanodes, it exposes a list of data files corresponding to the HDFS blocks placed on that datanode. Figure-27 shows the overall framework which is composed of the following key components.

4.3.2.1 Data Archiving Process

In our approach, we pre-process the log data created at the enterprise cluster to encrypt and make it HDFS friendly before archiving them in a public cloud. Specifically, large log data gets chunked into several small files of HDFS block size (64 MB default), gets encrypt it, and we label them with the timestamp information (e.g. 1-1-2013.to.2-1-2013.data) before uploading to the public storage cloud. The enterprise site can use a symmetric key encryption scheme to encrypt the dataset before archiving in the cloud.

When log data belonging to a given time window needs to be analyzed later on, VNCache can identify all blocks stored in the storage cloud that contain data relevant to that analysis. Next, we describe how these data blocks are presented to the HDFS layer at the enterprise cluster so that jobs can begin execution right away. We note that archiving the data in this manner does not preclude the data being accessed in a non-HDFS filesystem when needed. In such cases when there is a need to download the data in a non-HDFS filesystem, VNCache can download it through a normal Hadoop *dfs -get* command.

4.3.2.2 Virtual HDFS Creation

When a Hadoop job at the enterprise cluster needs to process an archived dataset, a virtual cluster is created by starting a number of VMs including one designated to be the primary Namenode. Before starting Hadoop in the VMs, a virtual HDFS

Table 9: HDFS fsimage

Image Element	Datatype
Image version	Integer
NAMESPACE_ID	Integer
NumInodes	Integer
GENERATION_STAMP	Long

namespace is created on the Namenode. It starts by generating a list of relevant HDFS blocks B_{job} for the job based on the input dataset. For example, for an analysis of 1 month of log data archived in the cloud, all blocks stored in the storage cloud that contains any data for the chosen time window would become part of the virtual filesystem[2]. A virtual file F_{job} is then created to contain $|B_{job}|$ HDFS blocks, where each block is given a unique HDFS identifier while maintaining its mapping to the filename in the remote cloud. Similar to HDFS Namenode filesystem formatting, a *fsimage* (filesystem image) file is generated and the virtual file is inserted into this *fsimage* filesystem image file (using our HDFS virtualization technique described next).

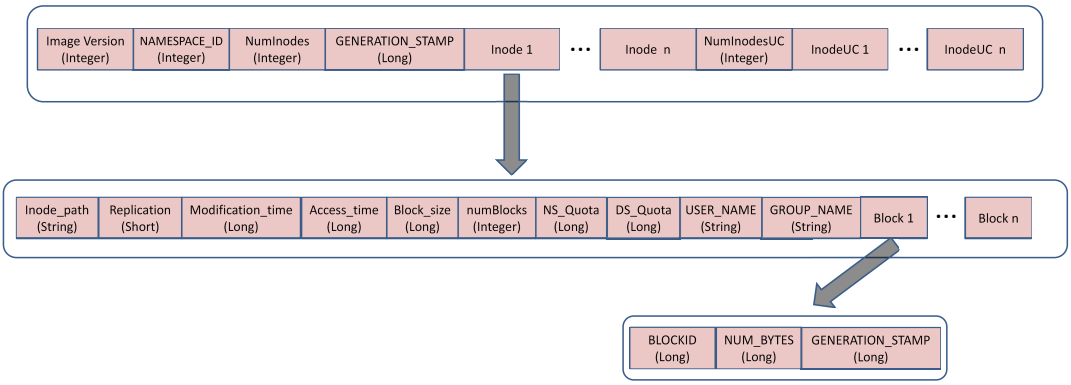


Figure 28: HDFS: FileSystem Image

The HDFS virtualization in VNCache initially creates a HDFS filesystem image and inserts an inode corresponding to the new file to be added into the virtual HDFS. The *fsimage* file is a binary file and its organization in shown in Tables 9 - 12. The

[2]Any unaligned time boundaries are handled in a special manner, details of which are omitted due to space constraints.

Table 10: HDFS INode

Image Element	Datatype
INODE_PATH	String
REPLICATION	Short
MODIFICATION_TIME	Long
ACCESS_TIME	Long
BLOCK_SIZE	Long
numBlocks	Integer
NS_QUOTA	Long
DS_QUOTA	Long
USER_NAME	String
GROUP_NAME	String
PERMISSION	Short

Table 11: HDFS Block

Image Element	Datatype
BLOCKID	Long
NUM_BYTES	Long
GENERATION_STAMP	Long

spatial layout of the HDFS filesystem is shown in Figure 28. The *fsimage* begins with the image version, Namespace identifier and number of Inodes stored as Integers and Generation stamp stored as Long. The Generation stamp is generated by the Namenode to identify different versions of the Filesystem image. Here the INode represents the HDFS datastructure used to represent the metadata of each HDFS file. For inserting a Virtual file into the virtualized HDFS, VNCache creates a new INode entry corresponding to the INode organization described in Table 10. The first field in the INode structure is the INode path stored as a String, followed by replication factor, modification and access times for the file. It also contains other fields such as the HDFS block size used by the file, number of HDFS blocks, namespace and disk space quotas, user name and group names and permission. The INode structure is followed by the information of each of the individual blocks of the file. As shown in Table 11, each block representation consists of a block identifier, number of bytes and generation stamp. The block generation stamp is a monotonically increasing number assigned by the Namenode to keep track of the consistency of the block replicas. Since these are assigned by the Namenode, no two HDFS blocks can ever have the same Generation Timestamp. The HDFS filesystem image also has a list of INodes under

Table 12: HDFS INode

Image Element	Datatype
INODE_PATH	String
REPLICATION	Short
MODIFICATION_TIME	Long
PREFERRED_BLOCK_SIZE	Long
numBlocks	Integer
USER_NAME	String
GROUP_NAME	String
PERMISSION	Short
CLIENT_NAME	String
CLIENT_MACHINE	String

construction (INodesUC) whose description is shown in Table 12.

An example HDFS Filesystem image is shown in Figure 29 where we find that there is an Inode created for the file /input/perf_svc_volumes.csv with a replication factor of 3 and it has the listing of the block ids of the 2 HDFS blocks present in this file. While the *fsimage* file is a binary file, for simplicity, it is shown in an XML format converted using the Hadoop Offline Image viewer Utility [17]. We find that the INode corresponding to the archived file */input/perf_svc_volumes.csv* is added with a replication factor of 3 and similarly, the individual blocks of the file with randomly generated block ID is added with the generation stamp information.

At the enterprise cluster, the namenode is started using the virtual HDFS filesystem image which enables Hadoop to understand that the required file and its individual blocks are present in the HDFS.

Next, we determine the virtual data layout of these HDFS blocks on the Datanodes. It is done similar to Hadoop's default data placement policy with its default replication factor of 3. Once Hadoop is started on the cluster, Datanodes report these blocks in the Block report to the Namenode, which assumes all HDFS blocks in the HDFS filesystem namespace are present even though initially the data still resides in the public storage cloud. Thus, from a Hadoop application stack the job execution can begin immediately.

```
<FS_IMAGE>
<IMAGE_VERSION>-18</IMAGE_VERSION>
<NAMESPACE_ID>838262466</NAMESPACE_ID>
<GENERATION_STAMP>1003</GENERATION_STAMP>
<INODES NUM_INODES="11">
<INODE>
<INODE_PATH>/input/perf_svc_volumes.csv
</INODE_PATH>
<REPLICATION>3</REPLICATION>
<MODIFICATION_TIME>2012-07-24 13:31
</MODIFICATION_TIME>
<ACCESS_TIME>2012-07-24 13:31</ACCESS_TIME>
<BLOCK_SIZE>67108864</BLOCK_SIZE>
<BLOCKS NUM_BLOCKS="2">
<BLOCK>
<BLOCK_ID>-3240566806188966147</BLOCK_ID>
<NUM_BYTES>67108864</NUM_BYTES>
<GENERATION_STAMP>1001</GENERATION_STAMP>
</BLOCK>
<BLOCK>
<BLOCK_ID>-3809894240632508629</BLOCK_ID>
<NUM_BYTES>67108864</NUM_BYTES>
<GENERATION_STAMP>1001</GENERATION_STAMP>
</BLOCK>
<NS_QUOTA>-1</NS_QUOTA>
<DS_QUOTA>-1</DS_QUOTA>
<PERMISSIONS>
<USER_NAME>balaji</USER_NAME>
<GROUP_NAME>supergroup</GROUP_NAME>
<PERMISSION_STRING>rw-r--r--</PERMISSION_STRING>
</PERMISSIONS>
</INODE>
</INODES>
</FS_IMAGE>
```

Figure 29: *fsimage* **file converted to XML format**

4.3.2.3 On-demand Data Streaming

VNCache enables on-demand streaming and on-the-fly decryption of HDFS data blocks. Once the read request for a HDFS block reaches the Datanode that (virtually) stores the block, VNCache on the Datanode looks up the mapping to its public cloud storage location and begins fetching the data from the public storage cloud. Once the block has been downloaded, it is decrypted before returning the data to the call. The enterprise site uses a symmetric key encryption scheme to encrypt the dataset before archiving in the cloud and therefore for decryption, we use the same key to decrypt the blocks prior to passing them to HDFS. Please note that the read requests received by the underlying VNCache may be for a portion of a data

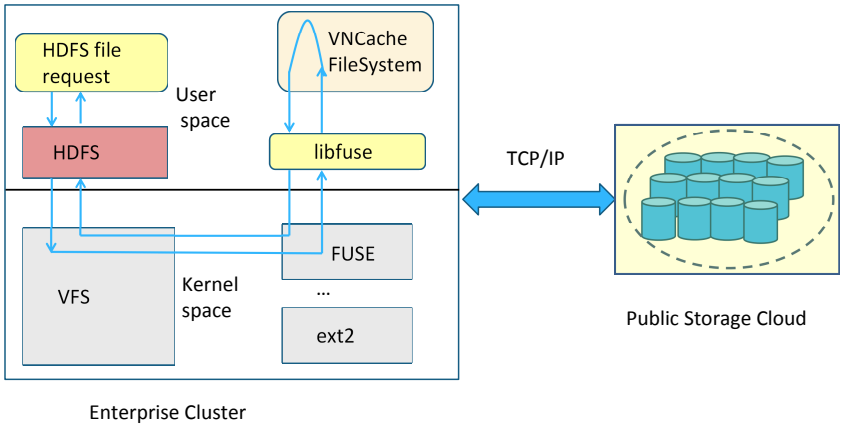


Figure 30: VNCache: Data Flow

block (e.g. Hadoop often does multiple 128k byte reads while reading a complete 64 MB block). For our implementation, we have chosen to start downloading the block when an *open* call is received and corresponding read requests are served from that downloaded and decrypted block.

Overall, from the HDFS standpoint, the HDFS data blocks - stored as files on the VNCache filesystem - are seamlessly accessible so Hadoop works transparently without the interference of streaming and decryption happening along this process.

4.3.2.4 Caching and Pre-fetching

The performance of the VNCache approach can be significantly improved if HDFS block read requests can be served from the disks of the enterprise cluster as opposed to streaming for each access. The goal of the caching algorithm is to maximize the reads from the local disks on the VMs and minimize streaming requests from the storage server in order to minimize the read latency for the jobs. Additionally, a good caching algorithm is expected to yield high cache hit ratios even for reasonable size of the cache on the disks of the VMs and should aim at minimizing the cache space used. VNCache incorporates a distributed cache prefetching algorithm that understands

the processing order of the blocks by the MapReduce workflows and prefetches the blocks prior to processing. For subsequent open and HDFS block read operations, the data from the disk cache in the enterprise cluster is used for reading. In case of a cache miss, VNCache still streams the data block from the remote storage clouds as explained above.

The cache manager follows a master/slave architecture where a dynamic workflow-aware prefetch controller monitors the job progress of the individual jobs in the workflow and determines which blocks need to be prefetched next and sends instructions to the slave prefetchers running on individual Hadoop nodes. Each slave prefetcher is multi-threaded and follows a worker model where each worker thread processes from a queue of prefetch requests. Each worker thread prefetches one HDFS data block file from the storage cloud and replicates the block within the Hadoop cluster based on the replication factor.

As mentioned earlier, the cache prefetcher logic needs to be capable of predicting the processing orders of the individual HDFS blocks by the MapReduce jobs so that the order of accesses corresponds to the prefetch order. Secondly, the caching algorithm needs to be dynamically adaptive to the progress of the jobs in terms of the map tasks that have been already launched and the ones that are to be launched next, thereby it does not attempt to prefetch data for tasks that have already completed. In addition, the prefetcher should also be aware of the rate of processing the job in terms of the average task execution time and as well as on the current network throughput available between the storage Clouds and the enterprise site.

Prefetching order: The cache prefetcher logic in VNCache is capable of predicting the processing order of the individual HDFS blocks. From the Hadoop design, we note that the default processing order of the blocks in a Hadoop job is based on the decreasing order of the size of the files in the input dataset and within each individual file, the order of data processed is based on the order of the blocks in the HDFS

filesystem image file - *fsimage*. While this ordering is followed in the default FIFO scheduler in Hadoop, some other sophisticated task placement algorithms ([143], [24]) may violate this ordering to achieve other goals such as higher fairness and locality. One direction of our ongoing work is focused on developing cache-aware task placement algorithms that achieve the goals of these sophisticated scheduling algorithms in addition to being aware of the blocks that are already cached.

Dynamic rate adaptive prefetching: We have designed the pre-fetching algorithm to be adaptive to the progress of the jobs so that it does not attempt to prefetch data for tasks that have already completed or likely to start before prefetching is complete. The algorithm constantly monitors the job progress information from log files generated in the logs/history directory of the master Hadoop node. It parses the execution log file to obtain the Job SUBMIT_TIME and Job LAUNCH_TIME and looks for task updates related to map task launching and completion. Based on the differences in the speed of job progress (primarily dictated by the type of job) and the time being taken to prefetch a block (dependent on connectivity between the enterprise site and the public storage cloud), the algorithm aims to pick the right *offset* for fetching a block. For example, if a job is progressing quickly and is currently processing block-4, the prefetcher may choose to prefetch blocks from an offset 20; in contrast, it may start from an offset 5 for a slow job.

To further react dynamically to the prefetching requests, the prefetch controller obtains the list of all tasks that are launched since the beginning of the job and the set of tasks that have already completed. Thus, based on the task start time and completion time, the caching algorithm understands the distribution of the task execution times of the current job.

In a similar manner, the slave prefetchers periodically report the average time to prefetch and replicate an HDFS block and the bandwidth observed by them from the Storage cloud to the enterprise site. Based on these reports, the cache controller

understands the average time for a block prefetch operation and accordingly makes the prefetching decision. If map task is launched for an input split whose block is not prefetched, the prefetch controller understands that the prefetchers are unable to prefetch at a rate similar to the rate of processing the blocks and hence makes an intelligent decision to skip prefetching the next few blocks and start prefetching blocks that are n blocks after the currently processing block in the prefetch ordering. Concretely, if $mtime_{avg}$ represents the average map execution time of a job running on a cluster with M map slots on each task tracker and if $ptime_{avg}$ represents the average time to prefetch a block, then upon encountering a task launch for a map task t whose data block B_i is not prefetched, the cache controller skips the next few blocks and starts prefetching blocks after block B_{i+n} where

$$n = \frac{ptime_{avg}}{mtime_{avg}} \times M$$

Cache eviction: Additionally, VNCache implements a cache eviction logic that closely monitors the job log and evicts the blocks corresponding to tasks that have already completed execution. It thus minimizes the total storage footprint resulting in a fraction of local storage used as compared to the conventional model in which the entire data set has to be stored in the enterprise cluster. Similar to the cache prefetching, the cache manager sends direction to the slave daemons for evicting a data block upon encountering a task completion status in the job execution files. The daemons on the VMs evict the replicas of the block from the cache creating space in the cache for prefetching the next data block.

Workflow-awareness: When dealing with workflows (multiple back-to-back jobs processing a set of data), the cache manager understands the input and output data of the individual jobs and makes prefetch and eviction decisions based on the flow of data within the workflows. If a workflow has multiple jobs each processing the same input dataset, the cache prefetch logic recognizes it and prefetches the data blocks only once from the storage cloud and subsequent accesses to the data is served

from the disk cache. Thus, the workflow-aware cache eviction policy makes its best effort to retain a data block in the cache if the workflow is processing that data block through another job in the future.

4.4 Experimental Evaluation

We present the experimental evaluation of VNCache based on three key metrics: (1) *job execution time*: this metric captures the response time of the jobs. It includes data transfer time, data loading and decryption time, and job processing time. (2) *cache hit-ratio*: this metric captures the effectiveness of the VNCache's caching algorithm. It measures the amount of data read from the local disks of the enterprise site as compared to streaming from the public storage cloud. (3) *Cache size*: this metric captures the total storage footprint required at the enterprise site for processing a remotely archived dataset. It thus indirectly captures the storage equipment cost at the enterprise cluster.

We compare three techniques primarily:

- *Full copy + Decrypt Model*: This technique downloads the entire dataset prior to processing and decrypts it and loads it onto the HDFS of the enterprise cluster. Therefore it incurs higher delay in starting the job.

- *VNCache: Streaming*: This technique incorporates the HDFS virtualization feature of VNCache and enables Hadoop jobs to begin execution immediately. It streams all data from the public storage cloud as blocks need to be accessed.

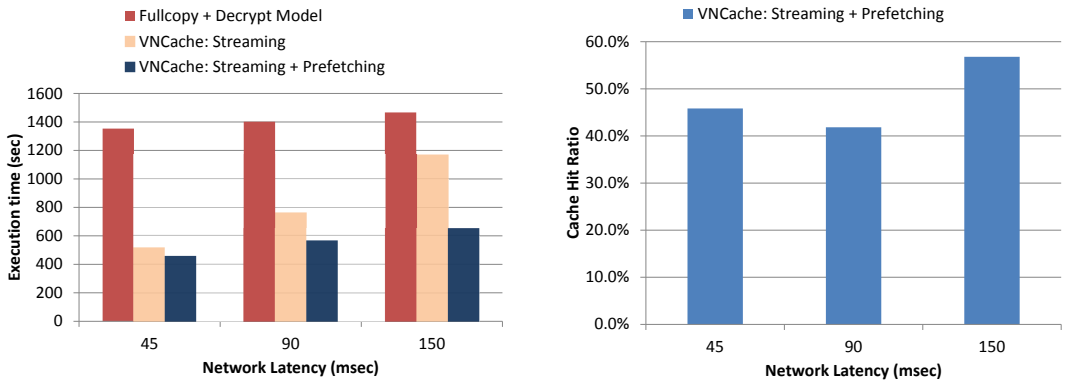
- *VNCache: Streaming + Prefetching*: It incorporates both the HDFS virtualization and streaming feature of VNCache and in addition, incorporates the VNCache prefetching and workflow-aware persistent caching mechanisms to improve job performance.

We begin our discussion with our experimental setup.

4.4.1 Experimental setup

Our cluster setup consists of 20 CentOS 5.5 physical machines (KVM as the hypervisor) with 16 core 2.53GHz Intel processors and 16 GB RAM. Out of these 20 servers, we considered 10 of them as the secure enterprise cluster nodes and used 5 other servers for functioning as public storage cloud servers. Our enterprise cluster had VMs having 2 2 GHz VCPUs and 4 GB RAM and by default we artificially injected a network latency of 90 msec (using the *tc* Linux command) between the public storage cloud and the enterprise cluster nodes to mimic the geographically separated scenario. Based on our cross-datacenter measurement experiments on Amazon EC2 and S3 (details explained in Section 4.4.2.1), this latency setting mimics the scenario where the public storage cloud and the enterprise cluster are present within the same coast (Oregon and Northern California datacenters) but physically separated.

The FUSE-based VNCache filesystem is implemented in C using FUSE 2.7.3. Our Virtual HDFS and VNCache cache manager are implemented in Java. We use DES symmetric key encryption scheme for encrypting the blocks. We use four kinds of workloads in our study including a data-intensive workload using grep and the Facebook workload generated using the Swim MapReduce workload generator [44] that richly represent the characteristics of the production MapReduce traces in the Facebook cluster. The workload generator uses a real MapReduce trace from the Facebook production cluster and generates jobs with similar characteristics as observed in the Facebook cluster. The trace consists of thousands of jobs depending upon the trace duration. Out of these, we randomly pick up 5 jobs and use that as a representative sample. Each job processes 5 GB of data by default and uses 5 VMS, each having 2 2 GHz VCPUs and 4 GB RAM. For a compute-intensive workload, we use the sort workload. In addition we consider two workflow-based workloads namely (i) *tf-idf* workflow and (ii) a workflow created as a combination of the jobs in the facebook workload trace. While the *tf-idf* workflow is reasonably compute-intensive,



(a) Execution time

(b) Cache hit ratio

Figure 31: Performance of Grep Workload

the facebook workflow is more data-intensive.

4.4.2 Experimental Results

Our experimental results are organized in the following way. We first present the comparison of VNCache Streaming + Prefetching model with the basic full copy + decrypt model and the VNCache streaming model for the single job workloads. We analyze the job performance enhancements of VNCache under a number of experimental setting by varying the network latency between the public storage cloud and enterprise site, the size of the archived dataset, the size of the disk cache present in the enterprise site. We show the impact of both the HDFS virtualization and streaming techniques in VNCache as well as its caching and prefetching mechanisms on the overall job performance. We then present a performance study of our techniques by considering workflow-based workloads and show that VNCache performs better than the full copy + decrypt model even in such cases.

4.4.2.1 Impact of Network Latency

We study the performance of the VNCache approach for several network latencies representing various geographical distance of separation between the public storage cloud and the enterprise site. In order to simulate the scenarios of various degrees of

geographic separation, we did cross-datacenter measurement experiments on Amazon EC2 and S3. As Amazon blocks ICMP packets and does not allow Ping based network measurements, we measured the average transfer time for transferring a file of HDFS block size (64 MB) between the datacenters and used that measurement to set our network latencies to obtain the similar block transfer times. For example, with S3 server in Oregon and EC2 in Northern California, a 64 MB HDFS block file takes 11 seconds to get transferred. Here, the 90 msec network latency scenario represents the public storage cloud and enterprise site located within the same coast (Northern California and Oregon datacenters corresponding to 11 second transfer time in our measurements) and a 250 msec scenario would represent another extreme scenario where the public storage cloud at the west coast (Oregon site) and the compute site at the east coast (Virginia datacenter). Therefore, we use the 150 msec setting to represent a geographic separation that is in between these two extremes. In Figure 31(a), we present the execution time of the Grep workload at various latencies. We find that with increase in network latency, the execution time of the jobs increase for both the Fullcopy + decrypt model and the VNCache approaches. Here, VNCache: Streaming consistently performs better than the Fullcopy + decrypt model at various latencies showing an average reduction of 42% in execution time. Further, the execution time of the streaming approach is reduced by more than 30% by the prefetch optimization. As evident from the figure, this improvement comes from both the Virtual HDFS based streaming model as well as through VNCache's intelligent prefetching. The cache hit ratios shown in Figure 31(b) illustrate that a significant amount of the input data (more than 45 %) were prefetched and read locally from the enterprise cluster.

While the Grep workload represents an I/O intensive workload, we next consider a more compute-intensive workload namely Sort. Figure 32 shows the execution time of the sort workload for the three approaches. Here we notice that VNCache achieves

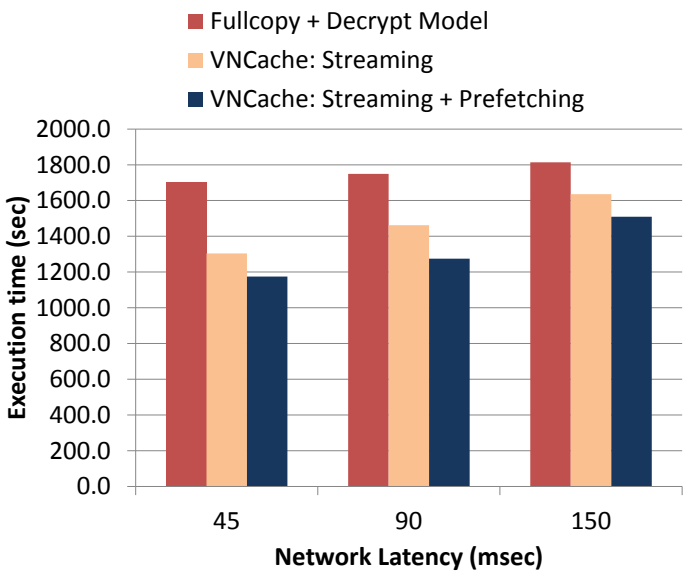


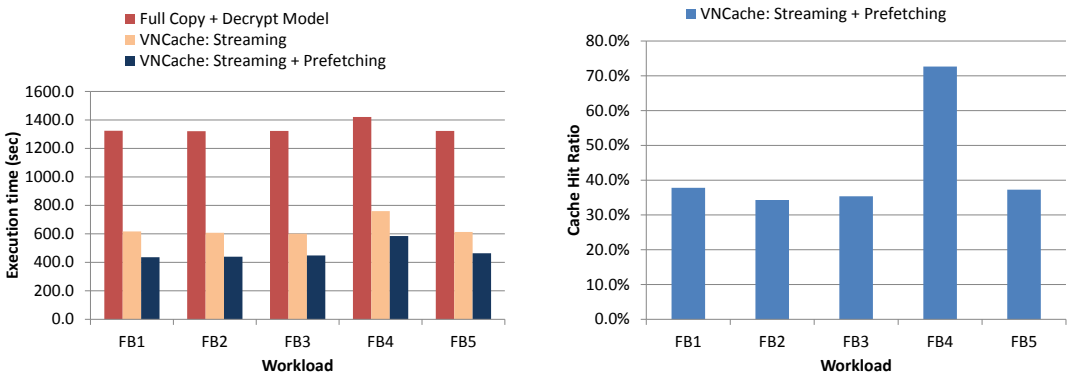
Figure 32: Performance of Sort workload - Execution time

a reasonable improvement of 25% for even a compute-intense workload such as Sort.

4.4.2.2 Performance of Facebook workload

Our next set of experiments analyze the performance of VNCache for the Facebook workload. Figure 33(a) shows the comparison of the execution time of 5 randomly picked jobs from the Facebook workload for streaming and streaming + prefetching techniques in VNCache and the basic Full copy + decrypt model. Here each job processes 5 GB of data and the network latency between the public storage cloud and enterprise cluster is 90 msec. We note that since the basic Full copy + decrypt model copies the entire (encrypted) dataset from the public storage cloud, decrypts and loads it into HDFS of the enterprise cluster, the jobs take longer time to execute.

VNCache streaming technique on the other hand uses its Virtual HDFS to start the job immediately while streaming the required data on demand. We find that the streaming approach consistently achieves higher performance than the Fullcopy + decrypt model showing an average reduction of 52.3 % in job execution time for the jobs.



(a) Execution time

(b) Cache hit ratio

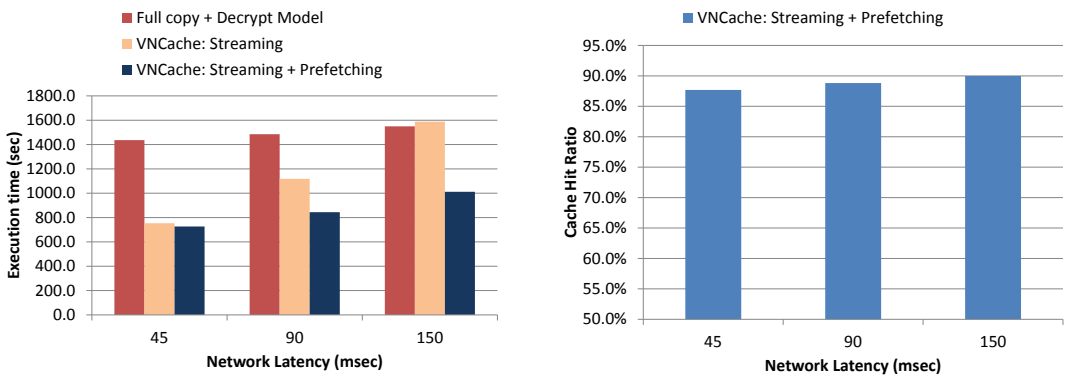
Figure 33: Performance of Facebook workload

Additionally, the VNCache prefetching techniques give further performance benefits to the jobs achieving an average reduction of 25.8% in execution time compared the VNCache streaming approach.

We present the obtained cache hit ratios for the VNCache: streaming + prefetching technique in Figure 33(b). We find that the prefetching optimization achieves an average cache hit ratio of 43.5% and thus serves 43.5% of the data from the local disks at the enterprise site as compared to streaming from the public storage clouds. These local reads contribute to the reduction in job execution times shown in Figure 33(a). We also notice that FB4 job has a higher cache hit ratio compared to the other jobs as its running time (excluding the data loading and loading time) is longer which gives more opportunity to interleave its compute and data prefetching resulting in higher local reads from prefetched data.

4.4.2.3 Performance of Job Workflows

Next, we study the performance of VNCache for job workflows that constitutes several individual MapReduce jobs. We first study the performance for a I/O intensive workflow composed of three randomly picked facebook jobs that process the same input dataset. As the three jobs in this workflow process the same dataset as input,



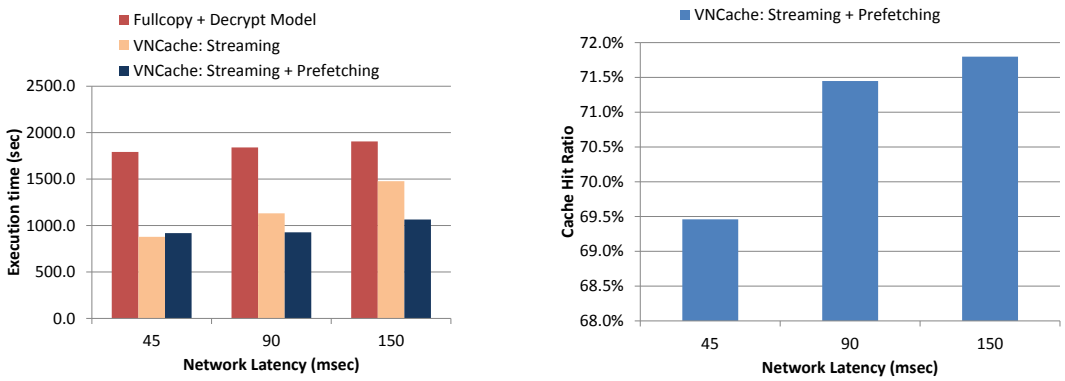
(a) Execution time

(b) Cache hit ratio

Figure 34: Performance of workflow (facebook jobs)

we notice in Figure 34(a) that the VNCache:Streaming model is not too significantly better than the full copy model especially at some higher latency such as 150 msec. Here, since three individual jobs of the workflow use the same input dataset, just merely streaming the data blocks for each of the three jobs becomes less efficient. Instead, the workflow-aware persistent caching approach in VNCache: Streaming + Prefetching caches the prefetched data at the enterprise site for the future jobs in the workflow and thereby achieves more than 42.2% reduction in execution time compared to the Full copy model. The cache hit ratios shown in Figure 34(b) shows that VNCache enables more than 88.8% of data to be read locally from the enterprise cluster for this workflow. Thus,the workflow-aware persistent caching avoids multiple streaming of the same block and helps the individual jobs read data within the enterprise cluster.

For a compute-intensive workflow, we use the *tfidf* workflow which computes the term frequency - inverse document frequency (*tf-idf*) for the various words in the given dataset. It consists of three jobs, the first two of which read the input dataset while the third job reads the output of the first two jobs. In Figure 35(a), we find that the job execution time for this workflow is again significantly reduced (by more than 47%) by VNCache. Also, the cache hit ratio in this case (Figure 35(b)) suggests that



(a) Execution time

(b) Cache hit ratio

Figure 35: Performance of Tfidf workflow

VNCache is able to prefetch a significant fraction (more than 70%) of the data.

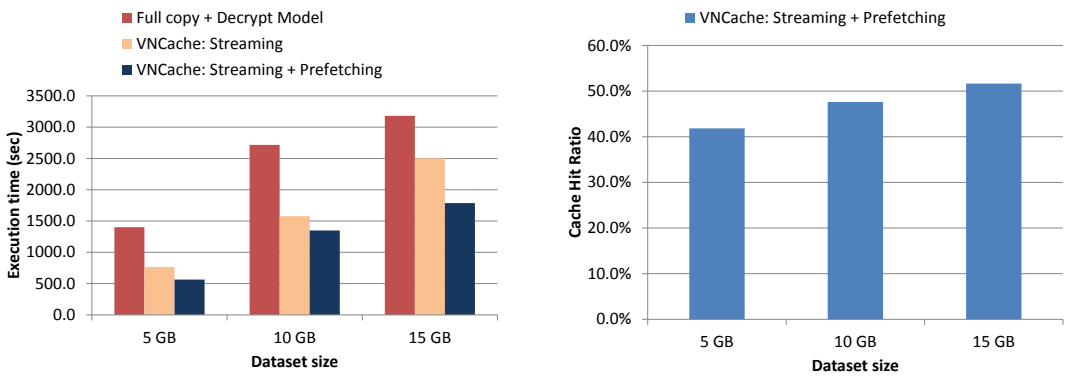
4.4.2.4 Impact of Data size

Our next set of experiments vary the input dataset size for the jobs and study the performance of the individual jobs as well as the workflows. We present the execution time of Grep workload in Figure 36(a) for different input dataset size. We find that the techniques perform effectively for various datasets achieving an average reduction of 50.6% in execution time. We also find a good average cache hit ratio of 47% in Figure 36(b).

Similarly, for a compute-intensive workflow, we present the *tfidf* workflow performance for different dataset size. We find in Figure 37(a) that the VNCache techniques continue to perform well at even bigger dataset size with an average reduction of 35.9% in execution time. The performance improvement is further explained by the high average cache hit ratio (61.2 %) in Figure 37(b).

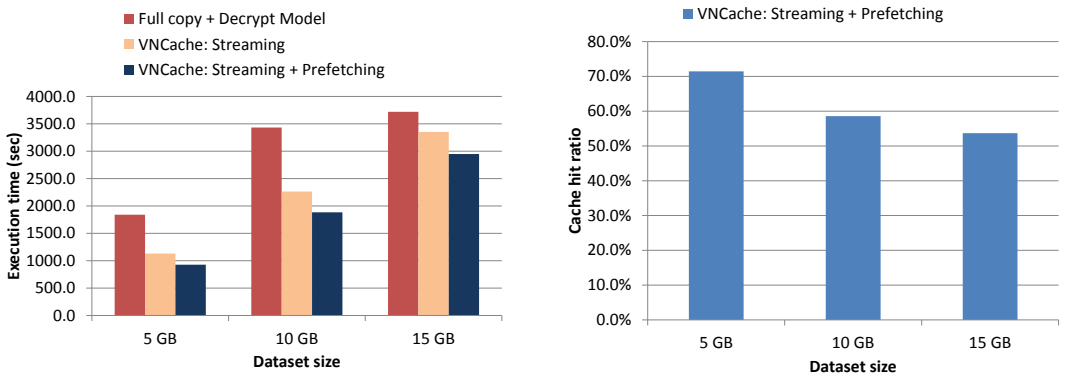
4.4.2.5 Impact of Cache Size

Next we study the impact of cache size at the enterprise cluster on the running time of the jobs. We vary the disk cache size on each VM in terms of the number of HDFS blocks that they can hold. Each HDFS block in our setting is 64 MB and we vary



(a) Execution time (b) Cache hit ratio

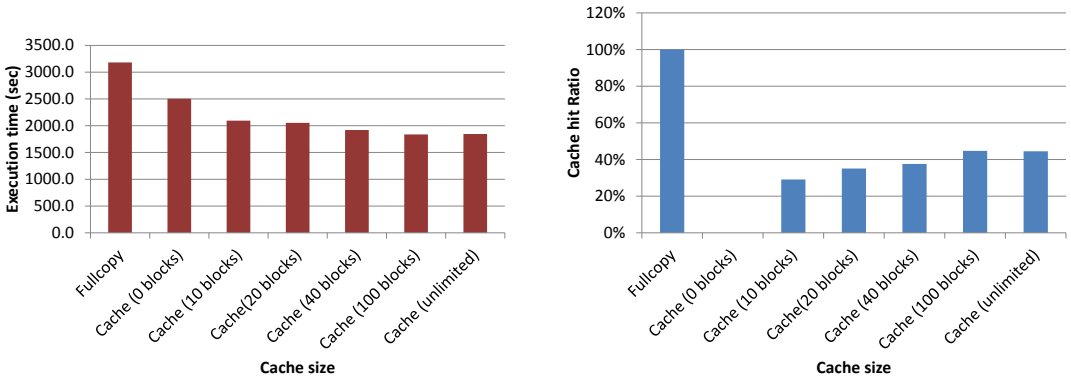
Figure 36: Performance of Grep with different data size



(a) Execution time (b) Cache hit ratio

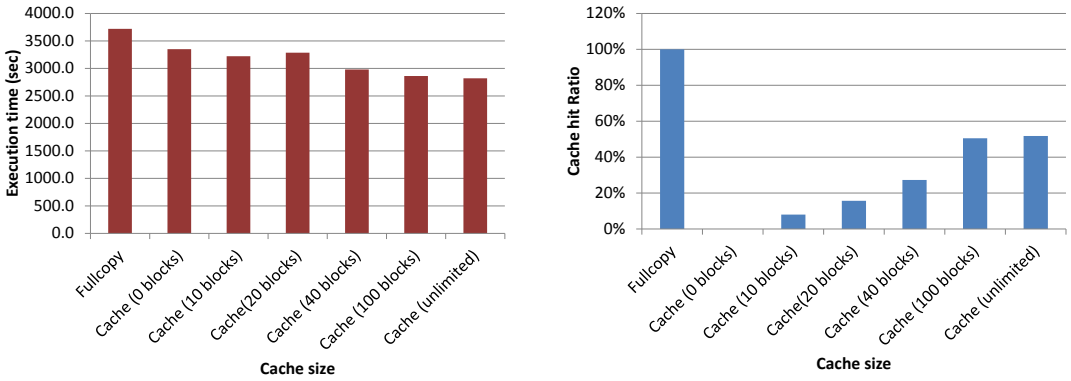
Figure 37: Performance of Tf-idf workflow with different data size

the cache size on the VMs from 10 to 100 blocks representing a per-VM cache of 640 MB to 6400 MB. We first study the performance of the Grep workload with cache sizes 10, 40, 100 blocks in Figure 38(a) and we find that the execution time of the VNCache:Streaming + Prefetching approach decreases with increase in cache size as a larger cache gives enough opportunity to hold the prefetched data blocks. Here the cache size of 0 blocks refers to the VNCache pure streaming approach. We find that even with a cache size of 10 blocks, VNCache achieves significantly lower execution time (Figure 38(a)) compared to the Fullcopy + decrypt model with a reasonable cache hit ratio (more than 35%) as shown in Figure 38(b).



(a) Execution time (b) Cache hit ratio

Figure 38: Impact of Cache size - Grep workload



(a) Execution time (b) Cache hit ratio

Figure 39: Impact of Cache size - Tfidf workflow

The performance tradeoffs with cache size for the *tfidf* workflow shown in Figure 39(a) also shows that with a reasonable cache, the privacy-conscious enterprise can tradeoff job performance to save storage cost at the local cluster.

4.4.2.6 Effect of number of VMs

Our next set of experiments studies the performance of VNCache under different number of VMs in the Hadoop cluster. In Figure 40 We vary the Hadoop cluster size from 5 VMs to 10 VMs and compare the performance of VNCache (streaming + prefetching model) with the full copy + decrypt model. We find that VNCache continues to perform well at different cluster sizes achieving an average reduction of

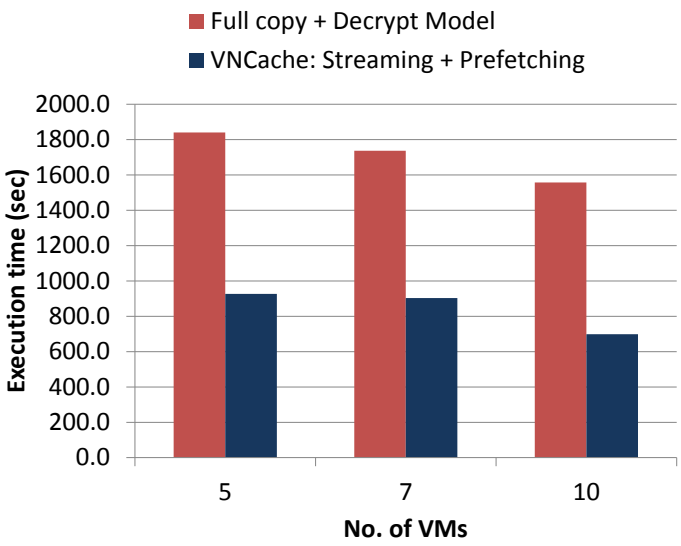


Figure 40: Effect of number of VMs

51%.

4.5 Discussions

VNCache is developed with the goals of providing on-demand streaming and prefetching of encrypted data stored in public clouds. Here we discuss some of the merits of the design choice of implementing VNCache streaming and prefetching techniques at the Filesystem layer. We note that as an alternate solution, the Hadoop Distributed Filesystem (HDFS) can be modified to add the caching and prefetching techniques of VNCache. In a similar manner, HDFS can be also modified to implement additional functionalities such as encryption support for handling privacy-sensitive data. However, we argue that such an implementation suffers from two drawbacks. First, it can not seamlessly operate with Hadoop as it requires changes to the Hadoop stack. Also, it makes it difficult to implement any changes to caching and prefetching policies as it requires modifying the Hadoop source each time. Additionally, implementing the HDFS virtualization and caching techniques at the Filesystem layer provides a seamless control point to introduce further optimizations such as dealing with storage

hierarchies. For instance, VNCache can be easily extended to deal with in-memory processing of blocks by caching the blocks in a memory location and using a memory cache in addition to the disk cache. In a similar way, VNCache can also provide support to optimize for introducing SSDs into the solution where the data blocks can be moved between memory, SSDs and disks based on a prefetch/evict plan. One direction of our future research is focused on extending VNCache to optimize job latency through in-memory computations.

4.6 Related Work

Hybrid Cloud solutions for MapReduce: There is some recent work on hybrid cloud architectures for security-conscious MapReduce applications [144, 84] that use public clouds for storing and processing non-private data while using a secure enterprise site for storing and processing private data. VNCache on the other hand addresses the challenge of processing archived (encrypted) data stored in public clouds in a privacy-conscious manner by providing a seamless interface for Hadoop to process the data within the enterprise site. Heintz et. al. [71] propose a solution to process geographically distributed data by scheduling map tasks close to their data. We note that such solutions are not suitable for security-conscious applications that prohibit the use of public clouds for data processing.

Caching Solutions: Recently, caching techniques have been shown to improve the performance of MapReduce jobs for various workloads [25, 97]. The PACMan framework [25] provides support for in-memory caching and the MixApart system [97] provides support for disk based caching when the data is stored in an enterprise storage server within the same site. VNCache differentiates from these systems through its ability to seamlessly integrate data archived in a public cloud into the enterprise cluster in a security-conscious manner and through its seamless integration with Hadoop requiring no modifications to the Hadoop stack. Furthermore, VNCache

provides a flexible control point to seamlessly introduce additional security-related functionality and other performance optimizations for storage hierarchies.

Locality Optimizations: In the past, there have been several efforts that investigate locality optimizations for MapReduce. Zaharia et al. [143] developed delay scheduler that attempts to improve job performance through increased task locality. Mantri [24] identifies that cross-rack traffic during the reduce phase of MapReduce jobs is a crucial factor for MapReduce performance and optimizes task placement. Quincy [78] is a resource allocation system for scheduling concurrent jobs on clusters considering input data locality. Purlieus [103] solves the problem of optimizing data placement so as to obtain a highly local execution of the jobs during both map and reduce phases. These above mentioned systems assume that the data is collocated with compute within the same Hadoop cluster and thus do not provide solutions for decoupled storage and compute clouds.

Resource Allocation and Scheduling: There have been several efforts that investigate efficient resource sharing while considering fairness constraints [13]. For example, Yahoo's capacity scheduler uses different job queues, so each job queue gets a fair share of the cluster resources. Facebook's fairness scheduler aims at improving the response times of small jobs in a shared Hadoop cluster. Sandholm et al [115] presented a resource allocation system using regulated and user-assigned priorities to offer different service levels to jobs over time. Zaharia et al. [142] developed a scheduling algorithm called LATE that attempts to improve the response time of short jobs by executing duplicates of some tasks in a heterogeneous system. Herodotou et al. propose *Starfish* that improves MapReduce performance by automatically tuning Hadoop configuration parameters [73]. The techniques in VNCache are complementary to these above mentioned optimizations.

4.7 Summary

In this chapter, we presented an efficient solution for privacy-conscious enterprises that deal with cloud-archived log data. We showed that current solutions are highly inefficient as they require large encrypted datasets to be first transferred to the secure enterprise site, decrypted, and loaded into a local Hadoop cluster before they can be processed. We presented our filesystem layer called VNcache that dynamically integrates data stored at public storage clouds into a virtual namespace at the enterprise site. VNCache provides a seamless data streaming and decryption model and optimizes Hadoop jobs to start without requiring apriori data transfer, decryption, and loading. Our experimental evaluation shows that VNCache achieves up to 55% reduction in job execution time while enabling private data to be archived and managed in public clouds

In the next chapter, we extend our cloud service provisioning framework to support ubiquitous access to data and compute services in the cloud without compromising user privacy when accessing services over a mobile device. Concretely, we will discuss the privacy risks involved in accessing location-based cloud services over mobile devices and present a suite of mix-zone based location anonymization schemes to support anonymous cloud services to mobile users on the move.

CHAPTER V

PROTECTING LOCATION PRIVACY OF MOBILE USERS IN A CLOUD

5.1 Introduction

We are entering a world where people and vehicles are being connected and tracked automatically on an ongoing basis. Such location tracking, on one hand, can offer useful and continuous services to mobile users, and on the other hand, generates enormous amount of potentially sensitive information. Location-based services (LBS) are becoming increasingly popular due to the advancement in wireless communication and the availability of low-cost mobile positioning devices. Such services require the mobile clients to report their location information to the Location-based Service running in an untrusted infrastructure such as a Cloud datacenter. Examples of Location-based services include searching nearest points of interest (*"Where is the nearest gas station to my current location?*"), spatial alerts (*"Remind me when I drive close to the ATM, I need to deposit a check*"), location-based social networking (" *Where is my friend, Tom?*"). Although LBS applications provide a lot of interesting and convenient services to users, it opens up new security risks that can endanger the location privacy of the mobile clients [81, 21]. Location privacy is a system-level capability of location systems, which controls the access to this information at different spatial granularity and different temporal and continuity scale, rather than stopping all access to location information. Provisioning efficient location-based cloud services to mobile users while strictly guarding and meeting their location privacy requirements is the focus of this chapter.

Several strategies have been suggested to protect personal location information.

The first strategy is to restrict access. Users who do not want location based services should be provided an option to refuse being tracked [72]. The second category of strategies is using spatial cloaking of locations, often referred to as location *k-anonymization*. This approach degrades the spatial resolution of location information in a controlled fashion before releasing it through location *k-anonymity* guarantee. A subject is considered *k-anonymous* if its location is indistinguishable from that of $k-1$ other users [33, 64, 67, 98, 135]. Location $k-anonymization$ approaches are targeted at applications that can operate completely anonymously and thus do not require true identity of users, such as finding nearby gas-stations or restaurants, and notifying the sale price of items of interest when we pass a shopping mall. However, the use of spatially cloaked resolution instead of exact position of users does not prevent continuous exposure of location information and thus may lead to breaches of location privacy due to statistics-based inference attacks [86]. The third category of strategies is the use of mix-zone model that anonymizes user identity by restricting the positions where users can be located [35] and by introducing methods to break the continuity of location exposure.

Mix-zones are regions in space where no applications can trace user movements. This is guaranteed by enforcing that a set of users enter, change pseudonyms and exit a mix-zone in a way such that the mapping between their old and new pseudonym is not revealed [35, 60, 61, 39]. However, most of the existing mix-zone proposals fail to provide attack resilient mix-zone construction algorithms that are effective for mobile users traveling on road networks and yet robust against timing and transition attacks. Concretely, theoretic mix-zones [35] are constructed independently of the spatially constrained road networks and thus limit their applicability to real world situations that people travel in spatially constrained networks or walk-paths. For instance, an adversary can utilize the timing information of users' entry into and exit from a mix-zone and the non-uniformity in the transitions taken at the road

intersections to guess the mapping between the old and new pseudonyms [60].

In this chapter, we present MobiMix, a road network based Mix-Zone framework to protect location privacy of mobile users. Compared to the existing approaches, the MobiMix mix-zones have a number of unique features.

First, the MobiMix mix-zones are developed based on a formal study of the assumptions of the theoretic mix-zone model and the detrimental impact on its obtained anonymity when certain assumptions are violated when taking into consideration of road network characteristics and motion behavior of mobile users (Section 5.2).

Second, we present the adversary model, including goals and types of attacks, and formally describe the MobiMix road network mix-zone model and the evaluation metrics in the presence of this adversary model (Section 5.3). Third but not the least, we develop a suite of attack resilient mix-zone construction techniques in terms of unlinkability between the old and new pseudonyms (Section 5.5.1). Our algorithms take into account of the constraints and limitations imposed by the road networks, the timing of users entering and exiting a mix-zone, and the transitioning probability of users in terms of their movement trajectory. We also present a detailed analysis of our mix-zone construction algorithms against the timing and transition attacks. We discuss mix-zone placement in Section 5.6 and evaluate the MobiMix approach and algorithms through extensive experiments conducted on traces produced by GTMobiSim [108] using different scales of geographic maps (Section 5.7). Our experiments show that MobiMix provides significantly higher level of resilience to timing and transition attacks compared to existing mix-zone approaches and yet efficient and scalable with respect to different types of road networks (maps) and different number of mobile users with different mobility patterns. We discuss related work in Section 5.8 and summarize in Section 5.9.

5.2 Analysis of Theoretical Mix-zones

In this section, we review the concept of theoretical mix-zone and the implications of its assumptions on the level of anonymity it provides.

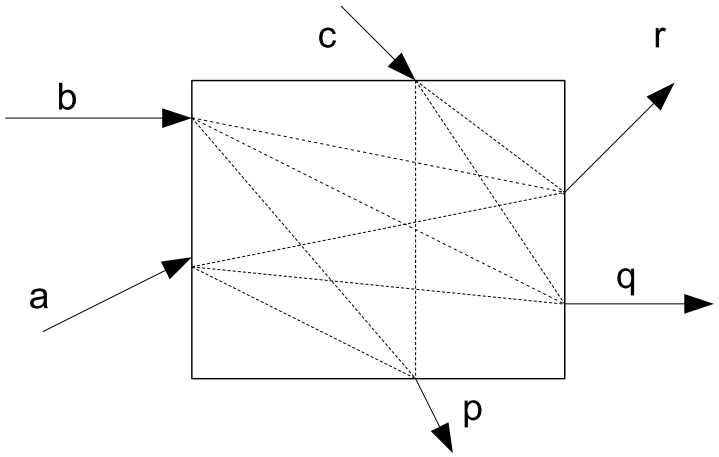


Figure 41: Mix Zone Model

A mix-zone of k participants refers to a k-anonymization region in which users can change their pseudonyms such that the mapping between their old and new pseudonyms is not revealed. In a mix-zone, a set of k users enter in some order and change pseudonyms but none leave before all users enter the mix-zone. These k users exit the mix-zone in an order different from their order of arrival, providing unlinkability between their entering and exiting events. Figure 41 shows a mix-zone of three participants, a, b and c exiting with new pseudonyms p, q and r. We formally present the theoretic model of a mix-zone and illustrate the strong assumptions used by the model to ensure high privacy guarantee.

Definition 1 *A mix-zone Z is said to offer k-anonymity for a set A of users iff*

1. *The set A has k or more members, i.e., $|A| \geq k$.*

2. *All users in A must enter the mix-zone Z before any user $i \in A$ exits. Thus, there exists a point in time where all k users of A are inside the zone.*

3. *Each user $i \in A$, entering the mix-zone Z through an entry point $e_i \in E$ and leaving at an exit point $o_i \in O$, spends a completely random duration of time inside.*

4. *The probability of transition between any point of entry to any point of exit follows a uniform distribution. i.e., an user entering through an entry point, $e \in E$, is equally likely to exit in any of the exit points, $o \in O$.*

Inside the mix-zone, the location of users cannot be tracked.

In the theoretical mix-zone model, the anonymity is measured in terms of the unlinkability between the old and new pseudonyms. For user i, exiting with a new pseudonym, i', let $p_{i' \to j}$ denote the probability of mapping i' to j, where $j \in A$. According to Definition 1, the theoretical mix-zone ensures an equi-probable distribution of mapping i' to $j \in A$. In other words, for every outgoing user, i', it is equally probable for i' to be any of the k users in the anonymity set A, having $p_{i' \to j} = \frac{1}{|A|}$. Therefore, the entropy, $H(i')$ of each outgoing user i' is computed according to the information theoretic measure of anonymity [59, 117]

$$H(i') = -\sum_{j \in A} p_{i' \to j} \times log_2(p_{i' \to j})$$

The Entropy is a measure of the amount of information required to break the anonymity provided by the system. In other words, the new pseudonym of user, i is indistinguishable from that of $|A_i|$ other users. We refer to this as pseudonym anonymity. Additionally, if each outgoing user exits in a uniquely different direction, then as an effect, the users would also obtain transition anonymity that ensures that the directions taken by an user, i during the exit is indistinguishable from that of $|A|$ users. In the theoretical model, although users take uniform transitions while exiting, there is no guarantee that all users exit in a unique direction and hence it can not make guarantees on the transition anonymity.

Next, we discuss the significance of the two important assumptions in the mix-zone model namely (1) users stay random time inside. (2) users follow uniform transition probability when entering and exiting a mix-zone.

When the users inside the mix-zone spend random time, it ensures a random reordering between the entry and exit orders providing a strong unlinkability between their old and new pseudonyms. However, a mix-zone that does not ensure random duration of time inside for its users usually leaks information [35, 60]. Such leakage may aid attackers to infer the mapping between the old and new pseudonyms of users. For example, when all users spend a constant time inside, the system would simply function in a FIFO (first-in-first-out) style, with the first exit event corresponding to the first entry event and so on. In that case, even though the users might have changed pseudonyms inside, their mapping from the old and new pseudonyms can still be inferred. A good mix-zone should therefore ensure sufficient randomness in the time spent inside it in order to obtain a high anonymity in terms of unlinkability after the pseudonym change process.

Similarly in a theoretical mix-zone, the probability of transition between an entry point and an exit point follows a uniform distribution. By relaxing this assumption, some transitions between entry and exit points may be more probable than the others. The attacker can use such knowledge to infer the mapping between the old and new pseudonyms. For example, if some transitions are less probable, the attacker may eliminate the pseudonym mappings corresponding to those transitions and thereby improve the success rate of his inference.

5.3 MobiMix: Overview

In this section, we present an overview of the MobiMix framework. We begin by introducing the challenges imposed by road networks for the construction of mix-zones and then present the associated attack models and anonymity metrics.

Theoretical mix-zones assume mobile users move in an Euclidian space without any spatial constraints. In real world, mobile users always move on a spatially constrained space, such as road networks or walk paths. Each road network mix-zone corresponds to a road intersection on a road network. Mix-zones constructed at road intersections have a limited number of ingress and egress points corresponding to the incoming and outgoing road segments of the intersection. Furthermore, users in a road network mix-zone are also constrained by the limited trajectory paths and speed of travel that are limited by the underlying road segments and the travel speed designated by their road class category [20]. Thus, users are not able to stay random time inside a road network mix-zone and no longer follow uniform transition probability when entering and exiting the mix-zone.

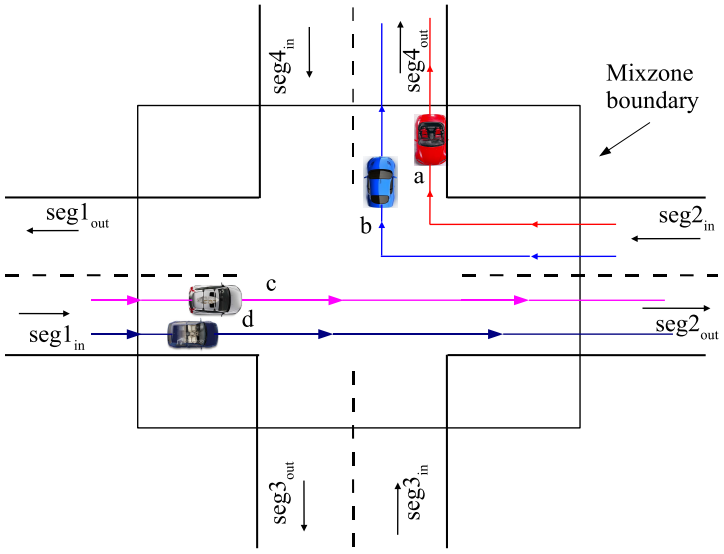


Figure 42: Road Network Mix Zone

For example, in figure 42, users a and b enter the road intersection from segment 2 and turn on to segment 4. Users c and d enter from segment 1 and leave on segment 2. When user a and b exit the mix-zone on segment 1 with their new pseudonyms, say α and β, the attacker tries to map their new pseudonyms α and β to some of the old pseudonyms a,b, c, and d of the same users. The new pseudonym α is more

likely to be mapped to two of the old pseudonyms, a or b, than the other pseudonyms because users a and b entered the mix-zone well ahead of users c and d and it is thus less probable for c and d to leave the mix-zone before users a and b given the speed and trajectory of travel. Here, the limited randomness on the time spent inside a road network mix-zone introduces more challenges to construct efficient mix-zones. Similarly, in figure 42, in order for the attacker to map α and β to c and d, the old pseudonyms, users c and d should have taken a left turn from segment 1 to segment 4 and users a and b should have taken an U-turn on segment 2. Based on common knowledge of inference, the attacker knows that the transition probability of an $U - turn$ is small and the mapping of α and β to c and d is very less probable. Hence, an efficient road network mix-zone should be resilient to such transition and timing attacks. Next, we introduce the attack models and the anonymity measures for road network mix-zones.

5.3.1 Adversary Model

The MobiMix development requires only the MobiMix engine that performs the pseudonym change for users inside a mix-zone to be trusted. Thus location based service providers are untrusted. For an adversary associated with an untrusted location based service provider, he may obtain a time series of locations and can partition such location data into subsequences by pseudonyms. By sorting different subsequences of locations, each corresponding to a different pseudonym, in terms of timing and road network location, the adversary is knowledgeable about mix-zones. Even though no true user identity is available in the location service requests received, the adversary is considered successful if he can utilize timing and transition based inference to infer the correct linkage between a pseudonym observed from the service requests received before a mix-zone and a pseudonym observed after a mix-zone. Similarly, an attacker can also be external observers of mobile users and traffics on the road networks and can intercept messages in the selected spatial and temporal space. The overall goal

of an adversary (curious or malicious) is to track the where about of certain users by linking a sequence of pseudonyms and by associating a user's pseudonym to the actual user's identity through establishing one to one mapping between the set of observed pseudonyms, combined with the association of a sequence of locations to a sensitive location such as home address or office building of a specific user or a location of special interest to a given user at a given time window. Also we assume that attackers in the adversary model can be semi-honest in the sense that they can be curious rather than completely malicious when launching an inference attack. Upon the success of an attack, an adversary is able to infer the where about of a user based on the correct mapping between a pseudonym and the true identity of a user and the linkage among a time series of pseudonyms. By intruding location privacy of a user, the adversary associated with a location service provider may further track location queries of the user associated with the set of her pseudonyms, leading to intrusion of user's content privacy.

The design of MobiMix aims at protecting location privacy of users by preventing timing and transition attacks to the road network mix-zones. We below describe three types of attacks based on the characteristics of road networks: (1) Timing Attack, (2) Transition Attack and (3) Combined Timing and Transition Attack.

Timing Attack: In timing attack, the attacker observes the time of entry, $t_{in}(i)$ and time of exit $t_{out}(i)$ for each user entering and exiting the mix-zone. When the attacker sees an user i' exiting, he tries to map i' to one of the users of the anonymity set, A_i. The attacker assigns a probability, $p_{i' \to j}$ that corresponds to the probability of mapping i' to j, where $j \in A$. The mapping probabilities are computed through inference based on the likelihoods of the rest of the users to exit at the exit time of i', denoted by $t_{out}(i')$. Once the mapping probabilities are computed, the attacker can utilize the skewness in the distribution of the mapping probabilities to eliminate some low probable mappings from consideration and narrow down his inference to

only the high probable mappings. Such timing attack can be detrimental if not handled appropriately in the mix-zone construction and usage model.

Transition Attack: In transition attack, the attacker estimates the transition probability for each possible turn in the intersection based on previous observations. On seeing an exiting user, i', the attacker assigns the mapping probability $p_{i' \to j}$ for each $j \in A$ based on the conditional transitional probabilities $T((ingress(j), egress(i'))$. Recall, $T((ingress(j), egress(i'))$ denotes the conditional probability of an user i' entering through the entry point, $ingress(j)$ given that the user exited at the exit point, $egress(i')$. Transition attack can equally affect the effectiveness of road network mix-zones as timing attack if not handled with care.

Combined Timing and Transition Attack: In the combined timing and transition attack model, the attacker is aware of both the entry and exit timing of the users and as well the transition probabilities at the road intersection for a given road network mix-zone. The attacker can estimate the mapping probabilities $p_{i' \to j}$ for each $j \in A$ based on both the likelihoods of every user j exiting at time $t_{out}(i')$ and the conditional transition probabilities $T(ingress(j), egress(i'))$. This combined attack is often more powerful than the timing and transition attacks in isolation.

5.4 MobiMix System Architecture

The system architecture of MobiMix consists of following components (1) MobiMix Anonymizer, (2) Road Network Monitor, (3) Mix-zone construction modules, (4) Mix-zone placement and (5) Computing Infrastructure. We describe each of them below:

5.4.0.1 Mix-zone Anonymizer

The Mix-zone anonymizer is responsible for anonymizing the raw location updates received from the mobile clients before releasing it to the Location Based Service

provider for processing. The anonymizer stores two important information: (1) Mix-zone-junctions Map that stores which junctions are presently functioning as mix-zones and (2) User-pseudonyms Map that stores the mapping between the user's real identity and their current pseudonyms. Upon arrival of a location update from a client, the anonymizer checks to see if the present location of the client corresponds to a mix-zone region. If so, the anonymizer drops the location update from being sent to the Location-based service (LBS) provider and denies service to the mobile client. Also, the mobile user is assigned a new pseudonym and the corresponding entry is updated in the User-pseudonym Map. If the mobile user is not currently inside a mix-zone, then the anonymizer passes the location update to the LBS server by replacing the real identity of the user its the current pseudonym.

5.4.0.2 Road Network Monitor

The road network monitor works closely with the mix-zone anonymizer. It examines each location update of the mobile client and monitors the current behaviour of the road network in terms of the user speeds and their arrival patterns. It consists of the following sub-components:

Arrival Rate Monitor: The arrival rate monitor observes the user arrivals in each road junction along each road segment and identifies the user arrival process and the associated parameters. It provides the arrival rate parameter to the mix-zone construction module.

Transition Monitor The transition monitor observes the transitions taken by the users in each road junction and computes the transition probabilities for all possible transitions in the road intersections. This information is used to compute the conditional transition probability in the attack-resilient mix-zone construction phase.

Road Speed Monitor Based on the location updates received from the clients, the road speed monitor computes the current speed of the road segments in terms of

the mean speed and standard deviation. Also, it is aware of the speed limits of the road segments based on the road category they belong to.

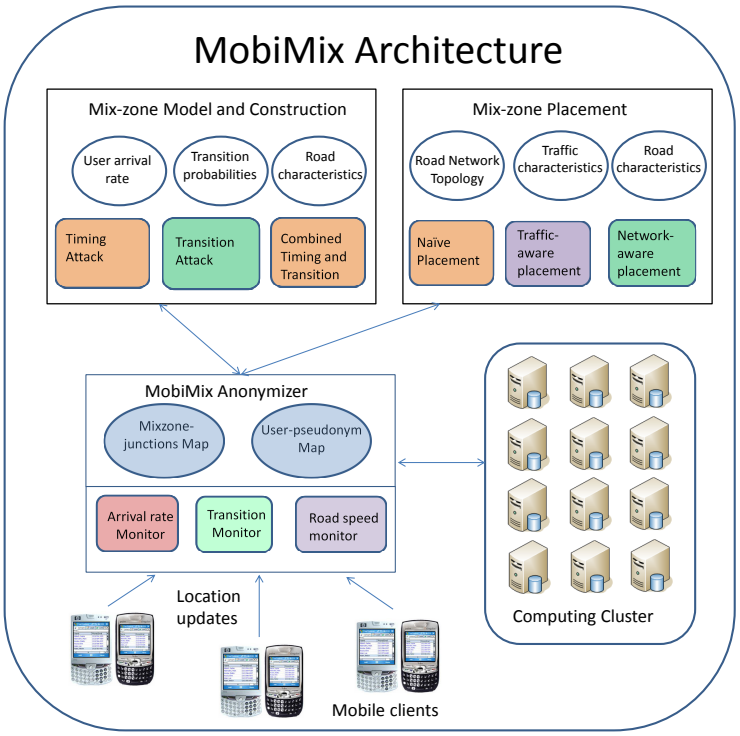


Figure 43: MobiMix System Architecture

5.4.0.3 Mix-zone Construction

The mix-zone construction module consists of the implementation of the MobiMix attack-resilient mix-zone techniques. It has information about the user arrival rate, transition probability in the junctions and speed distribution in the road segments through the road network monitor. The mix-zone construction takes into account the effect of both timing and transition attacks and ensures an expected number of users in the mix-zone that directly corresponds to the level of anonymity obtained. The construction module outputs the mix-zone size and shape for each mix-zone and also assists the mix-zone placement module to determine the best set of road intersections to function as mix-zones based on the user arrival rate, the transition probabilities at the junctions and the speed characteristics of the road segments.

5.4.0.4 Mix-zone Placement

The mix-zone placement component is responsible for deploying the mix-zones in the road network. In a huge road network of several tens of thousands of road junctions, the critical decision of which road junctions function as mix-zones can significantly impact the anonymity of the users. Improper selection of road junctions may result in unacceptably large size of mix-zones due to low user arrival rate or skewed transition probability distribution in the junctions. The placement module has knowledge of the road network topology, road characteristics in terms of road segment speed and arrival rate and also the mobility profiles of the users in terms of the transitioning probabilities at the road junction. MobiMix implements three mix-zone placement techniques namely (i) Naive Placement, (ii) Road characteristics aware (top-n) placement and (iii) Quadtree based (Grid) Network-aware placement.

5.4.0.5 Computing Infrastructure

The anonymizer with its monitoring sub-components run in a computing infrastructure. This computing infrastructure can be a dedicated infrastructure within the anonymizer's organization. Here, a set of servers would be responsible for anonymizing users in one geographical area and each server gets to receive only the location updates corresponding to its geographic area thereby balancing the overall load in the system.

5.4.1 Evaluation Metrics

In this subsection, we discuss the evaluation and anonymity metrics for measuring the level of anonymity provided by road network mix-zones.

The goal of mix-zone construction in MobiMix is to provide a guarantee by means of a lower bound on the anonymity obtained in them. We first review various existing metrics and discuss their inapplicability for the design of mobimix mix-zones and then present the metrics used in MobiMix.

Anonymity set size: The size of anonymity set is the most straight forward measure of anonymity. However, this metric alone is insufficient given the mapping probabilities may not be uniform in a road network mix-zone. Unlike an ideal mix-zone, in a road network mix-zone the attacker can identify which members are low-probable. Here, the low probable mappings do not effectively count for the anonymity. When the mapping probability distribution is not uniform, there can be attacks based on probability analysis [59, 117, 130]. In other words, we can not say that a road intersection performs as a good mix-zone just by the mere fact that the anonymity set is greater than k. A number of users in the anonymity set can become low probable under timing and transition attacks and will not effective count towards anonymity.

Entropy: An alternate measure of anonymity would be based on Entropy that captures the attacker's uncertainty in guessing the mapping between a new and old pseudonym [131, 55, 59, 117, 130]. However, entropy of a user is a measure over all members of the anonymity set. Therefore it may not effectively capture the cases where there is a few skewed mapping probabilities and a large number of non-skewed mapping probabilities. In such cases, a few high probable mappings can significantly increase the attacker's success of guessing the correct pseudonym mapping even though the entropy value may be high. In such cases, a significant part of the entropy could be contributed by a large number of non-skewed mapping probabilities leading to a high value of entropy. Hence, we cannot consider that a mix-zone provides good anonymity for a user if its entropy is greater than a certain value. To illustrate this, two systems can be shown to have the same entropy but however provide different levels of anonymity [130]. Let us consider two mix-zone based anonymity systems, one with uniform probability distribution among m users, i.e., $D_1 : p_u = \frac{1}{m}$, the other with a non-uniform distribution D_2 among n users, which is given by:

$$D_2 : p_u = \begin{cases} 0.5 & \text{for the actual subject} \\ \frac{0.5}{n-1} & \text{otherwise} \end{cases}$$

m	n	Entropy S
10	26	3.3219
20	101	4.3219
50	626	5.6439

Table 13: Two example systems yielding the same Entropy

We set different values of m and n as shown in Table 13 in order to obtain the same value of entropy. When $m = 10$, the first system gives only 10% chance for the attacker to guess the actual subject (according to D1). However, the attacker has 50% chance (according to D2) to guess the actual subject in the second system although both systems have the same entropy of 3.3219. This example suggests that entropy is not a sufficient measure to guarantee anonymity when the mapping probability distribution is non-uniform. As shown in the example, even one bad mapping probability can significantly break the anonymity of the user although there may be other users in the anonymity set with a probability similar to that of an uniform distribution. In summary, the entropy measure may not be used as an accurate estimation of the privacy when the mapping probabilities are non-uniform [130] as in our road network mix-zone case.

Normalized Entropy: Normalized entropy, also called Degree of Entropy, is defined as the ratio of the entropy obtained from the road network mix-zone to the entropy obtained from a theoretical mix-zone with the same anonymity set. In other words, it is a measure of how close is the entropy of the roadnet mix-zone as compared to a theoretical mix-zone. As entropy itself is a measure over all members of the anonymity set, comparing the entropy of the realistic mix-zone with the theoretical mix-zone also may not accurately capture the non-uniformity in the mapping probability distribution in all cases. It can be shown that there are still cases, such as when the normalized entropy is close to 1 but the mapping probabilities significantly deviate from the others [130].

We next present our proposed anonymity metric, Pairwise Entropy and explain our anonymity model based on Pairwise Entropy and the anonymity set size.

Pairwise Entropy: In order to ensure that the distribution of the mapping probabilities does not deviate much from the uniform distribution, we argue that it is important to measure the deviation of the mapping probabilities in a pairwise fashion. Pairwise entropy between two users i and j is the entropy obtained by considering i and j to be the only members of the anonymity set. In that case, we have two events: the event of i exiting as i' and the event of j exiting as j'. For the first event, we have only two mapping probabilities: $p_{i' \to i}$ and $p_{i' \to j}$. If the probabilities $p_{i' \to i}$ and $p_{i' \to j}$ are equal, then i' is equally likely to be i or j. The attacker has the lowest certainty of linking the outgoing user i' to i or j (50%). However, if one of the probabilities is much larger than the other, then the new pseudonym i' is more likely to be associated with one of the two old pseudonyms with high certainty ($> 50\%$) by eliminating the low probable one. In comparison, by Definition 1, a theoretical mix-zone ensures a uniform distribution for all possible mappings between old and new pseudonyms and a high pairwise entropy of 1.0 for all pairs of users in the anonymity set. If the pairwise entropy, $H(i, j)$ between users i and j when i exits as i' is close to 1, it means that the attacker will have a high uncertainty similar to that of an ideal mix-zone in guessing the old pseudonym of i'. However, the attacker also has another event namely the exit of j as j'. If this event leaks information, with a low pairwise entropy, $H(j, i)$, for instance if one of the mapping probabilities, $p_{j' \to i}$ and $p_{j' \to j}$ is significantly different from the other, the attacker will be able to identify the old pseudonym of j'. Consequently the attacker can also guess the old pseudonym of i' as i' and j' are mutually exclusive events. Therefore, both the pairwise entropies, $H(i, j)$ and $H(j, i)$ need to be close to 1. Hence, the effective pairwise entropy between users i and j can be assumed as the minimum of the two pairwise entropies $H(i, j)$ and $H(j, i)$.

We argue that an effective mix-zone should provide a pairwise entropy close to 1.0 for all possible pairs of the anonymity set. In general if there are k members in the anonymity set, then it requires that the pairwise entropy for all k^2 possible

pairs of users in the anonymity set is close to 1.0. For example, with three users, i, j, and k, in order for the anonymity set of i to contain $\{i, j, k\}$, we require that all pairwise entropies, $H(i, j)$, $H(j, i)$, $H(i, k)$, $H(k, i)$, $H(j, k)$ and $H(k, j)$ are high and close to 1. Only when the pairwise entropy of all possible pairs in $\{i, j, k\}$ is high and greater than the predefined threshold, α, they can belong to the effective anonymity set of i, A_i. Here, the lower bound pairwise entropy α decides the level of allowable variations in the levels of anonymity being offered. Higher value of α will restrict the variation of the likelihood probabilities and make them more closely resemble a uniform distribution. Here we would also like to note that the constraint of high pairwise entropy between all pairs of users in the anonymity set ensures that any low pairwise entropy between one pair of users gets propagated to affect the anonymity of the other users. For instance, in the above example let us assume $H(i, j) = 0$ and other pairwise entropies, $H(j, i)$, $H(i, k)$, $H(k, j)$, $H(j, k)$ and $H(k, j)$ are greater than α. Here, we can find that for user k, both $H(j, k)$ and $H(k, j)$ are greater than α and hence the anonymity set of k contains $\{j, k\}$. We also find that both $H(i, k)$ and $H(k, i)$ are greater than α, however it does not mean that user i belongs to the anonymity set of k. This is because, if the anonymity set of k contains $\{i, j, k\}$ then we require all pairs of users to have high pairwise entropy. This contradicts with the given fact that $H(i, j) = 0$. Thus, the low pairwise entropy $H(i, j)$ gets propagated to impact the anonymity of k even though user k has high pairwise entropy individually with users i and j.

As the pairwise entropy only measures how much uncertainty each member provides to the other, we also need to measure for a given user i, how many members belong to the effective anonymity set, A_i such that all pairs of users in A_i have high pairwise entropy with each other. Those members would form the effective anonymity set of i. In MobiMix, we use the pairwise entropy metric in combination with the effective anonymity set size to measure the anonymity.

Next, we describe the relative anonymity and success rate metrics used for evaluating MobiMix mix-zones.

Relative Anonymity: The relative anonymity level is a measure of the level of anonymity provided by the mix-zones, normalized by the level of anonymity required by the users. Higher relative anonymity levels mean that, on the average, users get anonymized with larger k values than the system-specified minimum k-*anonymity* levels.

Success Rate: The success rate measures the ratio of the number of times users obtain anonymity equal or greater than the system-specified minimum k-*anonymity* levels. A good mix-zone should provide anonymization with a success rate close to 100%.

5.4.2 Road Network Mix-zone Model

In this section, we present the MobiMix model for road network mix-zones and discuss the level of anonymity offered in terms of pairwise entropy and the anonymity set size, k. We model the road network as a directed graph $G = (V_G, E_G)$ where the node set V_G represent the road junctions and the edge set E_G represent the road segments connecting the junctions. In this work, we consider only the road junctions that connect three or more road segments as candidate junctions for mix-zones. Consider a mix-zone constructed at a road intersection v as shown in Figure 44. Assume that each user i enters the mix-zone at time $t_{in}(i)$ and exits at time $t_{out}(i)$ with a new pseudonym i'. Let $iseg(i)$ denote the incoming segment of user i through which i enters the mix-zone, $oseg(i)$ denote the outgoing road segment of user i through which i leaves the mix-zone. The speed followed by the users in a road segment is assumed to follow a Gaussian distribution with a mean μ and standard deviation σ, where μ and σ are specific to each road class category. For user i, the set of all other users who had entered the mix-zone during the time window defined by $t_{in}(i) - \tau$ to

$t_{in}(i) + \tau$, forms the anonymity set of i, denoted as A_i where τ is a small value.

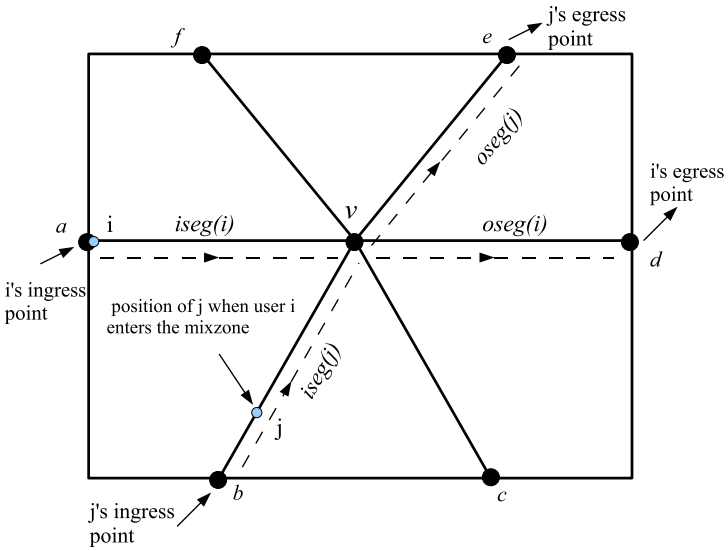


Figure 44: Road Network Model

We first derive the pairwise entropy corresponding to user i and its anonymity set A_i under timing attack. Then, we discuss the anonymity obtained under transition attack. We define $d_i(i)$ as the distance travelled by i inside the mix-zone. It is the sum of the lengths of the mix-zone regions on the incoming and exiting segments ,$iseg(i)$ and $oseg(i)$. $d_i(j)$ is defined as the distance that j needs to travel inside the mix-zone if it were to exit on the outgoing segment of i namely $oseg(i)$ instead of its actual outgoing segment, $oseg(j)$. $d_i(j)$ is the sum of the lengths of the mix-zone regions on the segments, $iseg(j)$ and $oseg(i)$. If $l_{iseg(i)}$ and $l_{oseg(i)}$ represent the lengths of the mix-zone on the incoming and outgoing segments of i, then $d_i(i)$ is given by

$$d_i(i) = l_{iseg(i)} + l_{oseg(i)}$$

Similarly,

$$d_i(j) = l_{iseg(j)} + l_{oseg(i)}$$

Let $speed_i$ and $speed_j$ denote the random variables of the speed of users i and j. As the speed is assumed to follow a Gaussian distribution, the variables $speed_i$ and

$speed_j$ become Normal variables. We also assume that time is slotted and let t be the time of exit of user i, that is $t_{out}(i)$. Let $p_{i' \to j}$ be the probability that the exiting user i' is j and $p_{i' \to i}$ be the probability that the exiting user is i. Users i and j become anonymous from each other if the probability, $p_{i' \to j}$ is exactly equal to the probability, $p_{i' \to i}$ which happens when users i and j enter the mix-zone at the same time and travel the same distance to exit the mix-zone. In short, the more one of these probabilities differs from the other, the higher confidence the attacker will have in linking the old and new pseudonyms.

Let $P(j,t)$ denote the likelihood that user j exits the mix-zone in the time interval, t to $t+1$. $P(j,t)$ numerically equals to the probability that user j takes time in the interval $(t - t_{in}(j))$ to $(t+1-t_{in}(j))$ to travel the distance $d_i(j)$. Accordingly, j needs to travel with an average speed in the range $s_1 = \frac{d_i(j)}{(t-t_{in}(j))}$ to $s_2 = \frac{d_i(j)}{(t+1-t_{in}(j))}$ in order to exit during the time interval between t to $t+1$. Therefore, we have

$$P(j,t) = \int_{s_2}^{s_1} speed_j(s) ds$$

Similarly,

$$P(i,t) = \int_{s_2}^{s_1} speed_i(s) ds$$

where $s_1 = \frac{d_i(i)}{(t-t_{in}(i))}$ to $s_2 = \frac{d_i(i)}{(t+1-t_{in}(i))}$

If $P(i',t)$ represents the likelihood that some user i' exits at time t to $t+1$, where i' can be either of i or j, we have

$$P(i',t) = P(i,t) + P(j,t)$$

Therefore, applying Baye's Theorem, the probability of i' being j when i' exits at time t, denoted as $p_{i' \to j}(t)$ is given by

$$p_{i' \to j}(t) = P((j,t)/(i',t)) = \frac{P((i',t)/(j,t)) \times P((j,t)}{(i',t))}$$

Here $P((i',t)/(j,t)) = 1$, as $P(j,t)$ is contained in $P(i',t)$. Therefore

$$p_{i' \to j}(t) = \frac{P(j,t)}{P(i',t)}$$

Similarly, the probability of i' being i, $p_{i' \to i}(t)$ is given by

$$p_{i' \to i}(t) = P((i,t)/(i',t)) = \frac{P(i,t)}{P(i',t)}$$

The pair-wise entropy between users i and j when i exits as i' is given by

$$H_{pair}(i,j,t) = -(p_{i' \to i}(t) log p_{i' \to i}(t) + p_{i' \to j}(t) log p_{i' \to j}(t))$$

Similarly, the pair-wise entropy between users i and j when j exits as j' is given by

$$H_{pair}(j,i,t) = -(p_{j' \to i}(t) log p_{j' \to i}(t) + p_{j' \to j}(t) log p_{j' \to j}(t))$$

Here, we notice that even though when i' exits, it might resemble both i and j with a closely equal probability and a high pairwise entropy, $H_{pair}(i,j,t)$, when user j' exits, it might reveal that j' is more likely to be one of i and j than the other as these are mutually exclusive events. Therefore, although the pair-wise entropy between i and j, $H_{pair}(i,j,t)$ may be close to 1 when i' exits, it may happen that the pair-wise entropy of j, $H_{pair}(j,i,t_{out(j')})$ when j' exits is well below 1. Hence, it is important that both of the two pair-wise entropies are high enough to make the attacker harder to guess the mapping. Therefore, the effective pairwise entropy of users i and j is given by the minimum of the two pairwise entropies, $H_{pair}(i,j,t_{out(i')})$ and $H_{pair}(j,i,t_{out(j')})$

$$H_{pair}(i,j) = \min\{H_{pair}(i,j,t_{out(i')}), H_{pair}(j,i,t_{out(j')})\}$$

Also, we find that the pairwise entropy is a function of the exit time, t of i'. As the exit time depends on the time spent inside the mix-zone which is inversely proportional to the speed of the user inside the mix-zone, the pairwise entropy becomes a function of the speed of the user inside the mix-zone. A good mix-zone should offer high pairwise entropy for a wide range of user speeds, for example, say 0 to 90 mph on a highway

road and 0 to 40 mph on a residential road. The lowest pairwise entropy offered by the mix-zone within this speed range would define the lowerbound pairwise entropy of the mix-zone. A good mix-zone should therefore offer a high lowerbound, α on the pairwise entropy for a wide range of user speeds.

We now extend our discussion with the pairwise entropy under transition attack. Based on the transition probabilities of the road junction, let $T(seg_l, seg_m)$ be the conditional transition probability computed by the attacker on exit of i'. $T(seg_l, seg_m)$ represents the conditional probability of user i' entering through an incoming segment seg_l given that i' exited on the outgoing segment seg_m. The mapping probabilities, $p_{i' \to i}$ and $p_{i' \to j}$ under the transition attack are therefore given by

$$p_{i' \to i} = \frac{T(iseg(i), oseg(i'))}{T(iseg(i), oseg(i')) + T(iseg(j), oseg(i'))}$$

and

$$p_{i' \to j} = \frac{T(iseg(j), oseg(i'))}{T(iseg(i), oseg(i')) + T(iseg(j), oseg(i'))}$$

Hence, the pairwise entropy under transition attack will be

$$H_{pair}(i, j) = -(p_{i' \to i} log p_{i' \to i} + p_{i' \to j} log p_{i' \to j})$$

In order for the mix-zone to be resilient to transition attacks, the mix-zone should offer a high lowerbound, β on the pairwise entropy after transition attack for all pairs of users in the anonymity set.

Next, we define the criteria for a roadnet mix-zone to function as an effective mix-zone based on the lowerbounds α and β on the pairwise entropies after timing and transition attacks.

Definition 2 *A road network mix-zone offers k-anonymity to a set A of users if and only if the following conditions are met:*

1. *There are k or more users in the anonymity set A.*

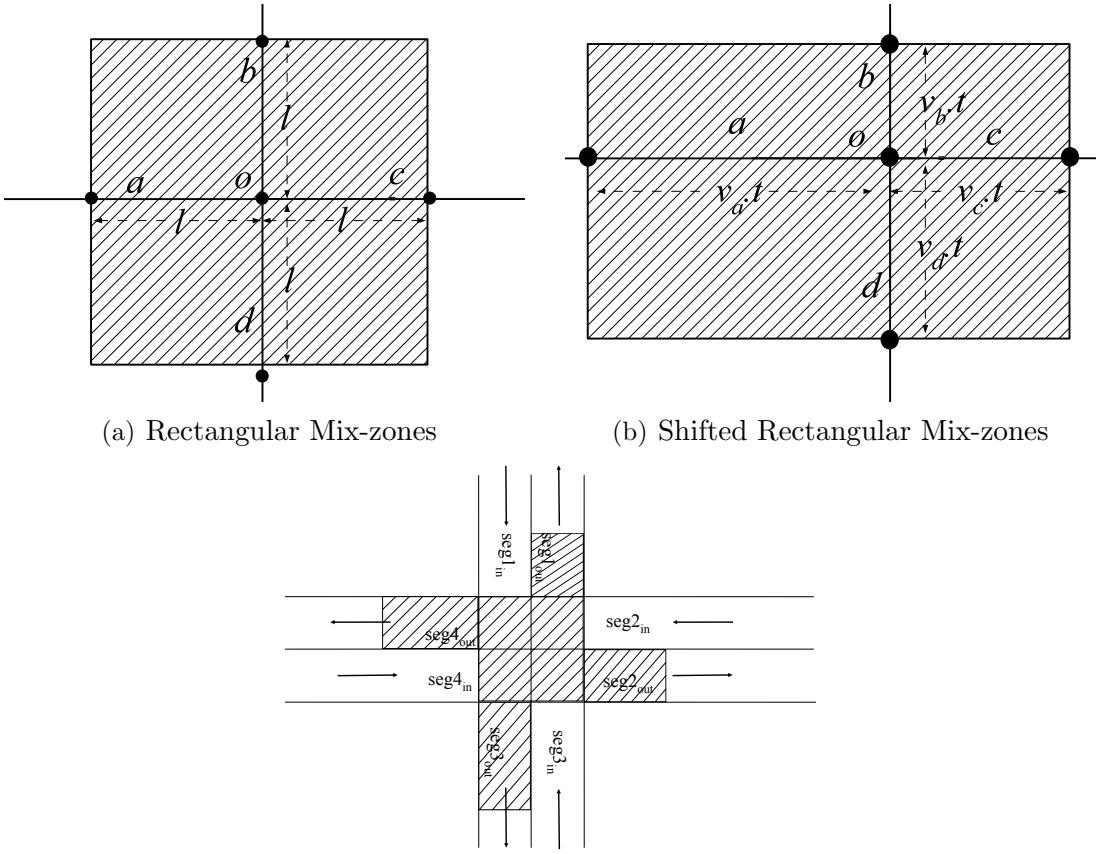
2. *Given any two users $i, j \in A$ and assuming i exiting at time t, the pairwise entropy after timing attack should satisfy the condition: $H_{pair}(i, j, t) \geq \alpha$.*

3. *For any two users $i, j \in A$, the pairwise entropy after transition attack should meet the condition: $H_{pair}(i, j) \geq \beta$.*

In the next section, we present our proposed techniques and approaches to construct road network mix-zones that effectively satisfy the above conditions.

5.5 Mix-zone Construction

In this section, we present the MobiMix techniques to construct road network mix-zones. We compare and analyze their effectiveness against timing and transition attacks.



(a) Rectangular Mix-zones

(b) Shifted Rectangular Mix-zones

(c) Non-rectangular Mix-zones

Figure 45: Mix-zone Shapes

5.5.1 Construction Approaches

We first describe the weaknesses of the naive rectangular mix-zone approach and then propose three MobiMix mix-zone construction techniques taking into consideration the geometry of the zones and their impact on the resilience to timing attack. We propose: (i) Time Window Bounded(TWB) Rectangular, (ii) Time Window Bounded(TWB) Shifted Rectangular and (iii) Time Window Bounded(TWB) Non-rectangular mix-zones. All perform better than the naive Rectangular mix-zones under timing attack.

5.5.1.1 Naive Rectangular Mix-zones

A straight forward approach to construct mix-zones around the road junction is to define a rectangular region centered at the road junction as shown in figure 45(a). The rectangle is defined based on some default size. For each exiting user i', the set of users that were inside the mix-zone at any given time during user i''s presence in the mix-zone forms its anonymity set, A_i. Here, any two users that were present together at any same given time, become members of each other's anonymity sets.

5.5.1.2 TWB Rectangular Mix-zones

In the time window bounded approach, the rectangle is constructed in the same way as in naive rectangular mix-zone, however, the anonymity set for each user, i is assumed to comprise of users who had entered within a time window in the interval, $|t_{in}(i) - \tau_1|$ to $|t_{in}(i) + \tau_2|$. Here, $t_{in}(i)$ is the arrival time of user i and τ_1 and τ_2 are chosen to be small values so that the time window ensures that the anonymity set of i comprises only of the users entering the mix-zone with a closely similar arrival time as that of i. The goal of the mix-zone construction is to ensure high pairwise entropy for every pair of users entering within the time window. We would like to note that the anonymity guarantee made by the mix-zone is by design a lower bound on the anonymity observed by the adversary for two reasons. First, we argue that a

good anonymity system should anonymize users in such a way that there is similar probability of mapping the actual subject to all the other users in the anonymity set. Thus, by discarding the low probable mappings and the corresponding users from the guaranteed anonymity set, we get an estimate of the number of users whose mapping probability distribution closely resembles a uniform distribution. Thus we get a measure of the number of users to belong to the anonymity set in such a way that they get anonymized in a way very similar to that of an ideal system. For road intersections that have segments with the same speed distribution, we can precisely guarantee a lowerbound on the pairwise entropy for the members of the anonymity set by constructing the anonymity set with the right value of time window based on our MobiMix road network model. Although, the notion of mix-zone time window has been adopted in existing mix-zone proposals [60, 39] where a default value of time window is assumed for the junctions, the TWB rectangular approach decides the right size of the time window based on the arrival rate of users so that k or more users enter within the time window. Also as mentioned earlier, for road intersections that have road segments with same speed distribution, we can guarantee a lowerbound pairwise entropy based on the Mobimix model for each pair of users entering with the time bound window.

5.5.1.3 TWB Shifted Rectangular Mix-zones

In the Time window bounded shifted rectangular approach, the rectangle is not centered at the center of the junction, instead it is shifted in such a way that from any point of entry into the mix-zone, it takes the same amount of time to reach the center of the road junction when travelled at the mean speed as shown in figure 45(b). In the same way, from the center of the junction, it takes the same time to reach any exit point when travelling at the mean speed of the road segments. Here, a set of users entering within the short time window, $|t_{in}(i) - \tau_1|$ to $|t_{in}(i) + \tau_2|$ are likely

to exit the mix-zone at the same time. Hence, when user i exits as i' the attacker would find that i' is likely to be any of the members of the anonymity set, A_i. If t represents the average time to reach the center of the road junction from an entry point which is the same as the average time to reach an exit point from the junction center, then the mix-zone lengths on the segments would be given by the product of their mean speed, say v and the average time, t as shown in 45(b). Compared to naive rectangular and time window bounded rectangular mix-zones, shifted rectangular mix-zones provide good pairwise entropy for many cases, however, they do leak information when the speed of the users deviate from the mean speed resulting in a weaker anonymity system [59, 117, 130]. Another limitation of this approach is that it may not be possible to satisfy the shifted rectangle property if the road segments are not orthogonal. Hence, this approach is limited to only road junctions with orthogonal segments.

5.5.1.4 TWB Non-Rectangular mix-zones

A more effective way to construct mix-zones would be to have the mix-zone region start from the center of the junction only on the outgoing road segments as shown in figure 45(c). We refer to this technique as non-rectangular approach. The non-rectangular approach is free from timing attacks caused by the heterogeneity in the speed distribution on the road segments. As in the rectangular approaches, the anonymity set for each user, i comprises of users who had entered the mix-zone within a time window in the interval, $|t_{in}(i) - \tau_1|$ to $|t_{in}(i) + \tau_2|$. The length of the mix-zone along each outgoing segment is chosen based on the mean speed of the road segment, the size of the chosen time window and the minimum pairwise entropy required. We discuss details on computing the mix-zone size and time window in section 5.5.4.

5.5.2 Timing Attack Analysis

In this sub-section, we analyze the privacy strengths of the proposed mix-zone approaches under timing attack and compare their attack-resilience.

5.5.2.1 Naive Rectangular Mix-zones

Timing attack is highly effective in Naive rectangular Mix-zones. In Naive Rectangular mix-zones, although the anonymity set size is typically large, a large number of members of the anonymity set become low probable under the timing attack. For instance, in figure 45(a), consider two users i and j entering from the segments a into the mix-zone. Let user i exit with a new pseudonym i' on segment c and let us assume the four road segments in the mix-zone, a, b, c and d have the same speed distribution. If the arrival times of i and j differ by a large value, then although users i and j might have been present together in the mix-zone for some amount of time, the attacker might infer that the user who entered first is more likely to exit first and that it is unlikely for j to have overtaken i before i exits the mix-zone. Therefore, the pairwise entropy of the naive rectangular mix-zones is low under timing attack, leaking more information to aid the attacker.

5.5.2.2 TWB Rectangular Mix-zones

TWB rectangular mix-zones have high resilience to timing attack in road junctions that have segments with the same speed distribution as the members of its anonymity set have similar time of arrival into the mix-zone. However, when the segments of the road intersection have different mean speeds, for instance if they belong to different road classes, the attacker may be able to eliminate some mappings based on the timing information. For example, in figure 45(a), let us assume a mix-zone of size 0.5 miles $\times$ 0.5 miles with segments a and c of residential road category having a mean speed of 20 mph and segments b and d of highway roads with a mean speed 60 mph. Consider two users i and j entering the mix-zone at the same time. Let user i enter

through the highway segment b and exit through the highway segment d and let user j enter though the residential segment a and exit through the residential segment c. If both i and j travel around the mean speed of their respective road segments, then i and j would exit approximately in 30 seconds and 90 seconds respectively. When user i exits out with a changed pseudonym i' in 30 seconds, the attacker can infer that i' is more likely to be i than j. Thus, even though the anonymity set consists of users entering with closely similar arrival time, the differences in the speed distribution on the roads leaks information to aid the timing attack.

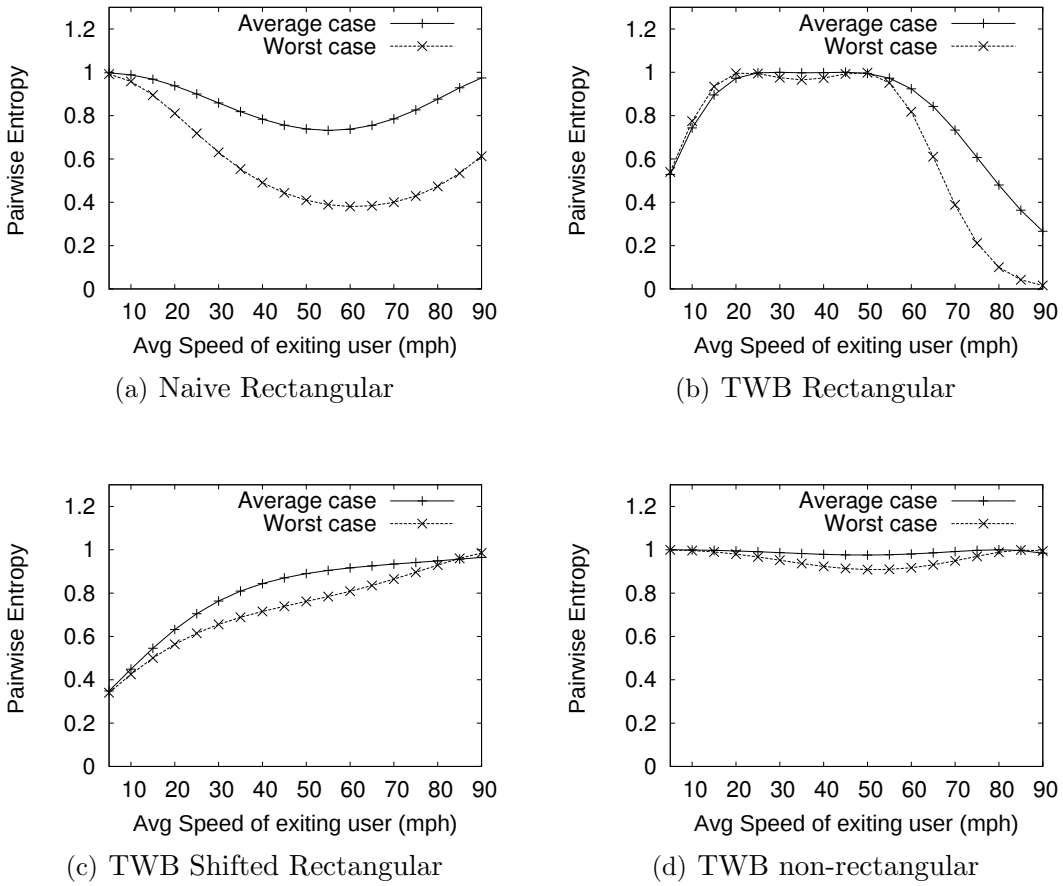
5.5.2.3 TWB Shifted Rectangular Mix-zones

TWB shifted rectangular mix-zone are resilient to timing attacks even on road junctions that have segments with different mean speeds if the users travel at the mean speed of the segments. However, they are also prone to timing attack when the speed of the users deviate from the mean speed of the road segments. For example, in figure 45(b), consider a mix-zone of size 0.5 miles X 0.5 miles in a road intersection with a slow residential road segment, a having mean speed 20 mph and three other highway segments, b, c, and d having mean speed 60 mph. Let all road segments have a standard deviation of 10 mph from their mean speed. The computation would yield $v_a.t = 0.375$ miles and $v_b.t = v_c.t = v_d.t = 0.125$ miles. Let users i and j enter the mix-zone at the same time. Let user i enter through the highway segment, b and exit through the highway segment, d and let j enter through the residential road segment, a and exit through the highway segment, c. Let us assume user j travels with a speed of 10 mph on segment a and travels at 60 mph on segment, c. In this case, the attacker would see j' exiting in 2 minutes, 32.5 seconds. With this timing information, the attacker can find that j' is more likely to be mapped to j than i because if j' is i, then i should have travelled really slow on the highway segments b and c, with an average speed of 5.9 mph in order to exit after 2 minutes, 32.5 seconds. However, if j' is j, then j needs to have travelled only at 10 mph on the residential

road segment, a which is more likely to happen. Thus, the attacker can guess that j' is j with high confidence. In general, the shifted rectangular approach performs badly when the user's speed deviate from the mean speed of the road segments.

5.5.2.4 TWB Non-rectangular Mix-zones

The TWB non-rectangular mix-zone is most resilient to timing attacks as it does not encounter any disparity in the speed distributions. Here, as long as a pair of users enter within each other's time window, the attacker can not infer the correct pseudonym mappings if the length of the mix-zone is sufficiently large for the chosen time window. In the next subsection, we compare the effectiveness of the mix-zone approaches.



(a) Naive Rectangular

(b) TWB Rectangular

(c) TWB Shifted Rectangular

(d) TWB non-rectangular

Figure 46: Effectiveness of Mix-zones against timing attack.

5.5.2.5 Pairwise Analysis

In order to better understand the effect of timing attack on guessing the mapping between the old and new pseudonyms, we perform a pairwise analysis considering only two users in the mix-zones. We compare the effectiveness of the different approaches in figure 46. As an example, we consider a mix-zone of length 400 meter in a road junction that has two highway road segments where the speed is normally distributed with 60 mph mean and 20 mph standard deviation and 2 residential road segments where the speed is distributed with 25 mph mean and 10 mph standard deviation. For a rectangular and shifted rectangular mix-zone, the mix-zone length corresponds to the longer side of the rectangle and for the non-rectangular mix-zone, the mix-zone length refers to the length of the longest mix-zone region on the outgoing road segments. In this pairwise analysis, for the rectangular mix-zones, the breadth is also taken as 400 meter. We consider two users i and j and measure the worst case and average case pairwise entropies. User i travels on the fast highway segments and user j travels on the slow residential segments. The worst case typically represents the arrival times of i and j separated by the maximum possible value defined by the mix-zone time window. Here the mix-zone time window is taken as 4 sec for the example mix-zone considered. The average case represents the case where the arrival times of i and j are separated by half the size of the time window, namely 2 sec. User i changes its pseudonym to i' and the X-axis shows the average speed followed by the exiting user, i' inside the mix-zone and the Y-axis shows the worst case and average case pairwise entropies. We find that both the naive rectangular approach and the time window bounded rectangular approach have low pairwise entropy for both the worst case and average case for speeds even close to 60 mph, the mean speed of the highway segments that i travelled. Interestingly, the TWB rectangular approach shows higher pairwise entropy when user i' travels slow on its highway segments. This is because, if i' travels slow on the highway segments, then it's exit time would resemble that of

j much better as j is travelling on a slow residential segment. Similarly, the shifted rectangular approach shows good pairwise entropy when the speed of i' is close to the mean speed, 60 mph. However, its pairwise entropy drops when the speed of i' deviates from its mean speed. Outperforming all these approaches, the TWB non-rectangular approach has a very steady high pairwise entropy for a wide range of speeds of i'. This is because, in this mix-zone geometry, users travel only on one segment in the mix-zone and thereby do not encounter any disparity in the speed distributions and therefore it is the most resilient geometry for timing attack.

5.5.3 Transition Attack Analysis

We now analyse the impact of transition attack that can be launched to guess the mapping between the pseudonyms. For each exiting user, i' the attacker observes the exiting segment of i' and tries to maps i' to one of the users, j in the anonymity set based on the conditional transitional probability of exiting in the outgoing segment, $oseg(i')$ given that j entered from the incoming segment, $iseg(j)$. We study the significance of protecting mix-zones against transition attack by measuring the distribution of the pairwise entropy among the road junctions based on the skewness in their transition probabilities. We show the distribution of worst case and average pairwise entropies after transition attack in table 14 for the Northwest Atlanta map of Georgia. The worst case refers to the least possible pairwise entropy obtained in the junction. We notice that most junctions have only reasonably high average pairwise entropy after transition attack, suggesting that the transition probabilities at these junction do not follow an uniform distribution. We find that only less than 12 % of the junctions have a high pairwise entropy in the range 0.9 to 1.0 after the transition attack. Also, the worst case entropy of many junctions (more than 90%) have a low value of 0, corresponding to the mappings that indicate an U-turn. Clearly, in these cases of low pairwise entropy, the attacker would able to eliminate the mappings if transition attack is not handled properly in the mix-zone construction.

(a) Average

$H_{(i,j)}$	% of junctions
0.0-0.1	0
0.1-0.2	0
0.2-0.3	0
0.3-0.4	0
0.4-0.5	0.25
0.5-0.6	1.33
0.6-0.7	7.75
0.7-0.8	37.75
0.8-0.9	41.33
0.9-1.0	11.58

(b) Worst case

$H_{(i,j)}$	% of junctions
0.0-0.1	95.58
0.1-0.2	0.166
0.2-0.3	0.5
0.3-0.4	0.42
0.4-0.5	0.25
0.5-0.6	0.42
0.6-0.7	0.33
0.7-0.8	1.0
0.8-0.9	0.58
0.9-1.0	0.75

Table 14: Pairwise Entropy with Transition attack

In order to protect against transition attack in cases where the transition probability is skewed, the mix-zone time window should be chosen in such a way that for each outgoing segment, l, there are enough number of users (k or more) entering the mix-zone from the road segments that have similar transitioning probability to the outgoing segment, l, and hence have a higher pairwise entropy, say greater than or equal to β. Therefore, the attacker will have at least k users in the anonymity set that he cannot ignore from consideration.

Figure 47 shows a TWB non-rectangular mix-zone with 3 incoming segments, u, v, w and three outgoing segments, r, y, z. Let $T(u, y)$ be the conditional probability of an user entering the junction through segment u given that the user exited on segment y. The attacker assigns probability $p_{i' \to j}$ to each of the users $\{a_1, a_2, a_3, ...a_{k1}, b_1, b_2, b_3, ...b_{k2}, c_1, c_2, c_3, ...c_{k3}\}$ based on the conditional transition probabilities $T(u, y), T(v, y), t(w, y)$. Assume the conditional transition probability $T(u, y)$ is too small compared to $T(v, y)$ and $T(w, y)$ and let the probabilities $T(v, y)$ and $T(w, y)$ be similar. Let us assume an user i enters from segment w and exits in segment y as i'. Here, the attacker may be able to ignore $\{c_1, c_2, c_3, ...c_{k3}\}$ from the anonymity set of i'. However, i' would have a higher pairwise entropy with $\{a_1, a_2, a_3, ...a_{k1}, b_1, b_2, b_3, ...b_{k2}\}$. Thus, for outgoing segment y, if we can ensure that

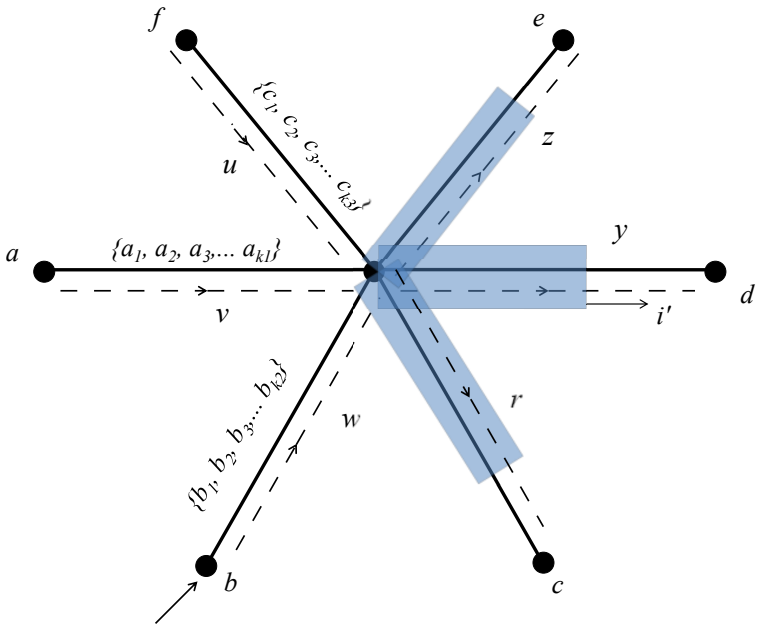


Figure 47: Countering Transition Attack

there are always k or more users entering from the segments v and w, then for any user, i' exiting on segment y, the attacker would be confused to differentiate i' from at least k other users that forms the effective anonymity set, A'_i. Here it should be also noted that even though the exit of i' on segment y does not leak information, the exit of some user, say a_2 along segment z may leak some information if the transition probability, $T(v, z)$ is much smaller than $T(w, z)$. Therefore, the effective anonymity should not contain those members that exit in a segment where user i's probability of exiting is lower as these are mutually exclusive events.

In the next sub-section, we discuss how to determine the time window and size of the mix-zone so as to make it resilient to both timing and transition attacks, yielding a high lowerbound, α and β on the pairwise entropies after timing and transition attacks respectively.

5.5.4 Combination of Timing and Transition attacks

The mix-zone time window directly impacts the number of users arriving from the various segments and therefore decides the mix-zone's resilience to transition attack. Once the right size of the mix-zone time window is determined for a specified level

of resilience to transition attack in terms of a high lowerbound, β on the pairwise entropy after transition attack, we need to determine the length of the mix-zone for the given time window so as to ensure a high lowerbound on the pairwise entropy after timing attack.

We assume that the user arrival on the road segments follows a poisson process. Given the mean arrival rate, λ_l on each incoming segment, l, let $\lambda_L^{x,y}$ represent the cumulative mean arrival rate of the users that effectively count towards the anonymity set of an user, i' exiting along segment y that entered through segment, x. If $M^{x,y}$ is a subset of the road segments in the mix-zone, we have $\lambda_L^{x,y} = \sum_{l \in M^{x,y} | H^y_{pair(l,x)} > \beta} (\lambda_l - \sum_{z | \exists m \in M^{x,y}, H^z_{pair(m,l)} < \beta} T(l,z) \times \lambda_l)$. It is the sum of the arrival rate of the segments such that the members have high pairwise entropy with each other and with i' during the exit of i' in segment y. Note that it excludes among the users who entered from segment, l, those that would exit in some segment, z where the conditional probability of exiting in z is significantly different. Here $M^{x,y}$ is chosen as that subset of the road segments that maximizes $\lambda_L^{x,y}$. If $N(t)$ represents the number of users who had entered the mix-zone at time t since the beginning, then the probability of having n users enter during a short time window, $\tau^{x,y}$ is given by

$$P[N(t+\tau^{x,y}) - N(t) = n] = \frac{e^{-\lambda_L^{x,y}\tau^{x,y}}(\lambda_L^{x,y}\tau^{x,y})^n}{n!}$$

$N(t+\tau^{x,y}) - N(t)$ would represent the number of users arrived within the short time interval, $\tau^{x,y}$. The probability that k or more users enter the mix-zone in the time window, $\tau^{x,y}$ is

$$P[(N(t+\tau^{x,y}) - N(t) \geq k] = 1 - \sum_{1 \leq n \leq k} \frac{e^{-\lambda_L^{x,y}\tau^{x,y}}(\lambda_L^{x,y}\tau^{x,y})^n}{n!}$$

By adjusting the size of the time window, $\tau^{x,y}$, we can lowerbound the number of users arriving from the segments whose conditional probability of exiting in segment y is similar to that of users from segment x. For instance, we may choose the time window, $\tau^{x,y}$ such that there are $k = 5$ or more users entering with a high probability,

say $p = 0.9$. The overall time window, τ of the mix-zone is given by the maximum value of $\tau^{x,y}$ among the various segments, y in the road junction.

$$\tau = \max_{y} \tau^{x,y}$$

Once the value of τ is decided, we determine the length of the mix-zone so that the mix-zone provides a high lowerbound, α on the pairwise entropy after timing attack for a wide range of user speeds. For example, we might want a lowerbound pairwise entropy of $\alpha = 0.9$ for a wide range of users' speed, say 0 mph to 90 mph. Our algorithm iteratively increments the length of the mix-zone till the expected lowerbound on the pairwise entropy is met for the chosen time window, τ. In this context, we note that except for the TWB non-rectangular mix-zones, the other approaches suffer from timing attacks and hence it is not possible to have a time window and mix-zone length for them to ensure a high lowerbound on the pairwise entropy. However, the TWB non-rectangular mix-zones offer high lowerbounds even for small mix-zone lengths. As we have a lower bound on the pair-wise entropy and a lower bound on k, the number of users, the mix-zone can now make probabilistic guarantees on the anonymity provided.

In addition to pseudonym anonymity k, the number of outgoing segments used by the k users differentiates the trajectories of the users. We discuss in Section 5.6 that we can maximize the trajectory anonymity through careful placement of the mix-zones on those road junctions where there is minimal skew in the transition probability distribution. Here, we would like to point out that in some extreme cases, it can happen that even though there is a guaranteed pseudonym anonymity of k in each mix-zone, the trajectories followed by the users may be the same. We believe that it is extremely rare in real life that all of the following three assumptions are true at the same time for such a case to happen: (1) All k users of the anonymity set travel together with the same velocity using the same trajectory right from their starting point such as home or office. (2) The trajectory of these k users is reserved

solely for themselves. No other users can enter these road junctions and mix-zones during the time when these K users are going through. (3) The destination of the trajectory can uniquely identify a privacy sensitive location such as cancer clinic or AIDs Clinic. If any of the above assumptions is not true, the MobiMix model will guard the location privacy of the user. It is obvious that the second assumption above is false for public road networks operational in most of countries today. Thus, we believe that the MobiMix development is original and beneficial for protecting location privacy of moving objects on road networks.

5.6 Mix-zone placement

In this section, we present the mix-zone placement algorithms that find the best set of road intersections to function as mix-zones based on the user arrival rates, statistics of user movements, road network topology and road characteristics in terms of mean user speeds and the temporal and spatial resolution of location exposure. Although individual mix-zones are efficient with respect to providing the required level of anonymity, careful deployment on the road is crucial to ensure good cumulative anonymity for users as they traverse through multiple mix-zones on their trajectories. Mix-zones placed too far from each other may lead to longer distances between adjacent mix-zones in users' trajectories. On the other hand, if there are too closer mix-zones, users may need to often go through mix-zones although they might have already gained the anonymity they wanted. An optimal solution to the mix-zone placement problem is NP-complete for even small road networks [61]. Thus we use a heuristic-based placement approach in MobiMix. A good placement algorithm should (i) provide sufficient anonymity in each of the mix-zones (ii) ensure that users go through sufficient number of mix-zones along their path to the destination and (iii) minimize the total number of mix-zones in the system, thereby minimizing the overall cost of the privacy protection.

An optimal solution to the mix-zone placement problem may be obtained using a formulation similar to that discussed in [14], however such optimal solutions to the placement problem become NP-complete for even small road networks. We present three heuristic-based strategies for mix-zone placement. A naive placement strategy is to randomly select a subset of road junctions with three or more road segments. A better strategy is to place mix-zones at intersections that have high density of traffic and low skewness in the transition probability distribution. While high density of traffic yields higher pseudonym anonymity, low skewness in transition probability helps maximize segment *l-diversity* obtained. We call this approach the road-aware top n placement. An alternative approach is the grid-based quadtree placement strategy, which divides a road-network into grid cells using quadtree index partition and maximizes the average distance between any pair of mix-zones within each quadrant (grid cell). The mix-zone placement algorithms find the best set of road intersections to function as mix-zones based on the user arrival rates, statistics of user movements, road network topology and road characteristics in terms of mean user speeds and the temporal and spatial resolution of location exposure. We know that the anonymity strength of the mix-zone is directly proportional to the anonymity set size and the attack resilience of the mix-zone, however, for a given value of anonymity set size, k, the size of the mix-zone is directly proportional to the arrival rate of the users form various road segments connected to the road junction and the skewness in the transition probability distribution. Therefore, the cost of a mix-zone is directly proportional to the size of the mix-zone as it directly impacts the limits on the usage of the location based service. A good placement algorithm should provide sufficient anonymity in each of the mix-zones, should also ensure that users go through sufficient number of mix-zones along their path to the destination, while minimizing the total number of mix-zones maintained in the system.

5.6.1 Naive Placement

In the naive placement scheme, the mix-zones are chosen based on only the structure of the intersections, considering only those that connect to three or more road segments. This set of road intersections forms the candidate set of mix-zones. Among the candidate set of road intersections, the mix-zones are placed by choosing a random subset of the candidate set of mix-zones. Although this straight-forward approach of mix-zone placement is aware of the road intersection topology, the approach lacks knowledge of the user arrival rate and user travel characteristics and hence it does not make careful decisions to minimize the cost of the constructed mix-zones. For example, even road intersections having low user arrival rates and skewed transition probability distributions may get chosen for placing mix-zones. However, constructing mix-zones at them would lead to huge mix-zone sizes in order for them to be sufficiently resilient to timing and transition attacks. Hence, the overall cost of the mix-zone placement in the naive approach may not be minimal.

5.6.2 Road-aware top -n Placement

In this placement methodology, the mix-zones are placed at intersections that have high density of traffic and low skewness in the transition probability distribution. The mix-zones constructed at such intersections are small in size, incurring minimal cost in terms of limiting the service inside the mix-zones. All the mix-zones are constructed to yield a certain lower-bounded anonymity in terms of the anonymity set size, k, and resilience to timing and transition attacks. This is done by carefully choosing the time window, τ to ensure that sufficient number of vehicles arrive in the anonymizing time window and the size of the mix-zone in such a way that every member of the anonymity set has a high pairwise entropy after transition and timing attacks. In this approach, the top-n mix-zones are selected based on their average estimated anonymity levels of the road intersections, precisely in terms of the cost of

the mix-zones. If $C(v)$ is the cost of the mix-zone constructed at road junction v for the privacy guarantees H_{min}. The selection algorithm sorts the road junctions in the increasing order of the cost of the mix-zones $C(v)$ and chooses the top-n candidates for the placement. Although, this approach minimizes the overall cost of the mix-zones in the road network, the distribution of the mix-zones may not be uniform across the road network. For example, while some parts of the network may be densely populated with mix-zones, some other parts may be very scarce in mix-zones. As a result, users following some trajectories will pass through unnecessarily more mix-zones, while some users may not be able to find sufficient mix-zones in their trajectories.

5.6.3 Quadtree/Grid Network-aware Placement

In the quadtree-based network-aware approach, the placement algorithm considers the topology of the road map in addition to the user and road characteristics. Similar to the top-n placement approach, this approach also considers only road intersections having low skewness in transition probability and high traffic arrival rates. However in order to ensure a uniform distribution of the mix-zones, the placement decision is made by closely considering the underlying road network topology. For instance, the placement of the mix-zones should ensure that the trajectories followed by the users have sufficient number of mix-zones at evenly separated distances. Hence, the mix-zone deployment in the road network has to ensure saptial uniformity while minimizing the overall cost of the mix-zones in terms of their size.

The Quadtree-based network-aware placement is a two phase algorithm. The first phase of the algorithm recursively divides the entire road map to construct a quadtree index. The quadtree construction divides the area based on the number of road junctions in it, the overall geographical area and the total length of the road segments and the number of candidate junctions for mix-zones. The algorithm dynamically decides and partitions if it needs to recursively partition the space further

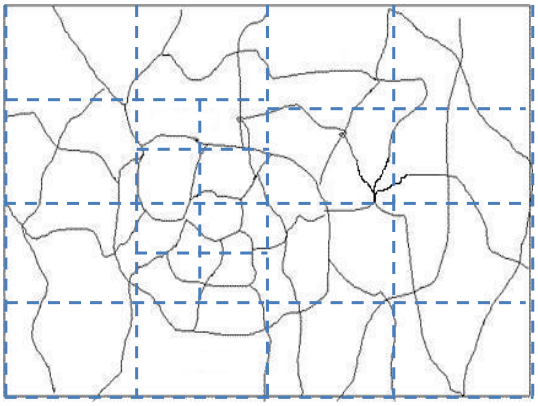


Figure 48: Grid-based Placement

into four quadrants. At the end of the quadtree construction, each quadrant roughly consists of the same number of road junctions, total segment length and number of candidate mix-zones as shown in figure 48.

The second phase of the algorithm deploys the mix-zones on a quadrant by quadrant basis. In each quadrant, the algorithm attempts to deploy the same number of mix-zones, however, the decision of which road junctions function as mix-zones is done to minimize the overall mix-zone cost while maximizing the average distance between any pair of mix-zones in a given quadrant. The objective is to maximize the pairwise distance between the mix-zones while not exceeding a certain specified maximum cost. This ensures that the mix-zones are uniformly distributed within each quadrant achieving higher spatial uniformity. Let Q represent the set of quadrants in the road network and let each quadrant, q have m mix-zones. If V_q and M_q respectively represent the set of all intersections in quadrant q and the set of intersections that functions as mix-zones in quadrant, $q \in Q$, then the objective function is given by

$$\min_{q \in Q} \sum_{v1, v2 \in M_q} dist(v1, v2)$$

subject to the constraints:

$$\sum_{v \in V_q} x_v = m$$

$$\sum_{v \in V_q} C(v) x_v \leq C_{max} \times m$$

where x_v is a boolean variable indicating if vertex $v \in V_q$ is a mix-zone and vertex v belongs to M_q if $x_v = 1$.

5.7 Experimental Evaluation

We divide the experimental evaluation of MobiMix into three components: (i) the effectiveness of our mix-zone construction approaches in terms of their resilience to timing and transition attacks (ii) their performance in terms of success rate and relative anonymity levels and (iii) the effectiveness of the mix-zone placement algorithms in terms of overall cumulative anonymity, mix-zone size and spatial uniformity of placement. Before reporting our experimental results, we first briefly describe the experimental setup.

5.7.1 Experimental setup

We use the GT Mobile simulator [108] to generate a trace of 10000 cars moving on a real-world road network, obtained from maps available at the National Mapping Division of the USGS [20]. By default we use the map of Northwest Atlanta region of Georgia that has 6831 road intersections with 10000 mobile users. The GTMobiSim mobile simulator extracts the road network based on three types of roads − *expressway*, *arterial* and *collector* roads. Our experimentation uses maps from three geographic regions namely that of Chamblee and Northwest Atlanta regions of Georgia and San Jose West region of California to generate traces for a two hour duration. We generate a set of 10,000 cars on the road network that are randomly placed on the road network according to a uniform distribution. The speed of the cars are distributed based on the road class categories as shown in Table 15. We use the Random Router mobility model in GTMobiSim where Cars generate random trips with source and destination chosen randomly and shortest path routing is used to

route the cars for the random trips. This captures more realistic scenarios than the random walk model. For instance, unlike the random walk model, the highway roads and expressways are more populated than the small residential roads as these roads share more parts of the shortest paths used by the users. Also, the random router model gives more realistic transition probabilities at the junctions which is essential to our evaluation.

Road type	**Expressway**	**Arterial**	**Collector**
Mean speed(mph)	60	50	25
Std. dev.(mph)	20	15	10
Speed Distribution	Gaussian	Gaussian	Gaussian

Table 15: Motion Parameters

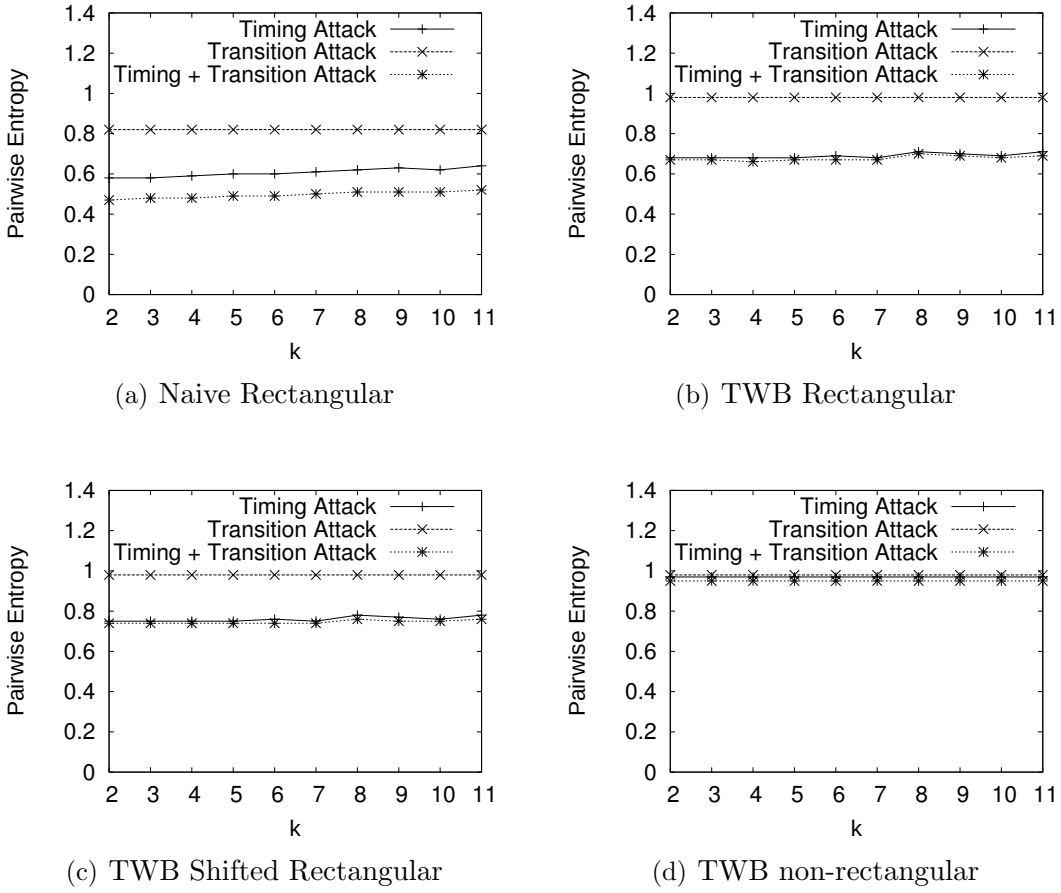
Parameter	**Value**
Map	Northwest Atlanta region
Mobility Model	Random Roadnet Router
Total number of vehicles	10000
Number of Road junctions	6831
Number of Road segments	9187

Table 16: Simulation Parameters and Setting

5.7.2 Experimental results

Our experimental evaluation consists of three parts. First, we evaluate the effectiveness of the mix-zone construction algorithms by measuring their attack resilience to timing and transition attacks. We then evaluate the effectiveness of the mix-zones in terms of the success rate in providing the desired value of k and study the relative anonymity level which is defined as the ratio of the obtained value of k to the expected value of k. We observe how these parameters behave when we vary the settings of a number of parameters, such as the expected value of k, the expected probability of success, p. Our final set of experiments evaluates the performance of the mix-zone placement algorithms in terms of the overall cumulative anonymity of the users, average mix-zone size and spatial uniformity of mix-zone placement. Our results show that the MobiMix construction techniques are effective, fast and scalable and outperform the basic construction methods by a large extent.

5.7.2.1 Resilience to Timing and Transition Attacks



(a) Naive Rectangular

(b) TWB Rectangular

(c) TWB Shifted Rectangular

(d) TWB non-rectangular

Figure 49: Average Pairwise Entropy after Attacks

In our first set of experiments, we analyse the effectiveness of the mix-zones against timing, transition and combined attacks. Out of the 6831 road junctions in the Northwest Atlanta region map, more than 2000 candidate junctions were chosen to build mix-zones based on their user arrival rate and the number of road segments that connect to them. Figure 49 shows the average pairwise entropy of the mix-zones for various values of k, the size of the anonymity set. We observe that the pairwise entropy after transition attack is low in the naive rectangular mix-zone compared to the other MobiMix approaches as the MobiMix mix-zones are protected for transition attack with their anonymity sets consisting of only members that have high pairwise entropy

to each other. The effect of timing attack is different across various approaches: we find that the TWB non-rectangular mix-zones perform the best under timing attack with the average pairwise entropy close to 1.0. Here, the length of the non-rectangular mix-zone is computed so as to ensure a lowerbound pairwise entropy of $\alpha = 0.9$ for the chosen time window size, τ which is computed based on the user arrival rate in the road junction to ensure the expected value of k with a high probability of $p = 0.9$. However, as discussed in section 5.5.2.5, it is not possible to lowerbound the pairwise entropy for the other mix-zone approaches. Hence, in order to compare the effectiveness of these approaches with the TWB non-rectangular approach, we construct the TWB rectangular and TWB shifted rectangular mix-zones with the same length and time window as used by the non-rectangular mix-zone. Similarly, the size of the naive rectangular mix-zone is fixed in such a way that the mean time to cross the mix-zone equals the time window of the TWB non-rectangular mix-zone. In figure 49, we also find that the naive rectangular and time window bounded rectangular mix-zones have low pairwise entropies after timing attack but the pairwise entropy of the TWB shifted rectangular approach is relatively higher, close to 0.8 as it's geometry is more resilient to timing attack. However, a high pairwise entropy of 0.9 or higher may be often required to ensure strong anonymity. In such cases, the time window bounded rectangular approach becomes the most efficient approach. Additionally, in the figure, we find that the effect of combined timing and transition attack is at least as severe as either of these attacks in isolation and it gets worse in naive rectangular mix-zones which is least resilient to both timing and transition attacks.

Similarly, figure 50 shows the comparison of the worst case pairwise entropy after timing attack for various mix-zones. The worst case pairwise entropy represents the lowest possible pairwise entropy obtained by the users after timing attack. Here also, only the TWB non-rectangular approach offers a high value for the worst case pairwise

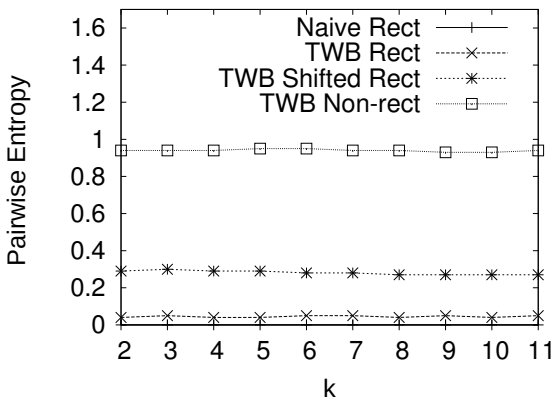
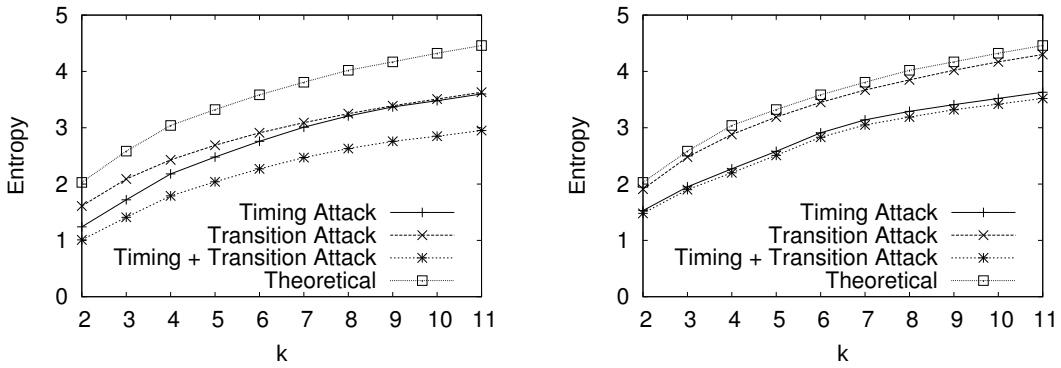


Figure 50: Worst-case Pairwise Entropy

entropy. The other approaches in their bad cases leak a lot information to aid the attacker. We also compare the overall entropy under attacks for various values of k in figure 51 for the same experimental setting. The overall entropy is computed by assigning the probability distribution, $P_{i'->j}$ for each user $j \in A_i$ based on the likelihood of user j to exit at the exit time of i. The line showing the theoretical value of entropy corresponds to the actual entropy obtained from an ideal mix-zone for the same anonymity set as the realistic mix-zones. We find that the TWB non-rectangular approach has the highest overall entropy after timing, transition and combined attacks closely resembling that of a theoretical mix-zone.

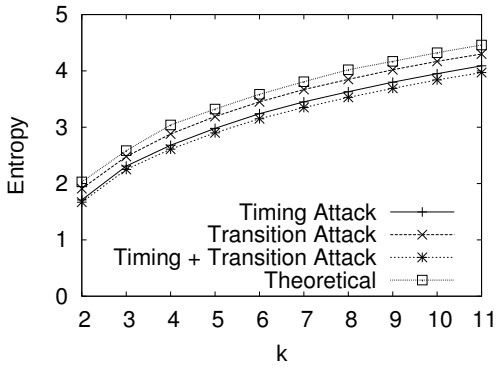
5.7.2.2 Success Rate and Relative Anonymity

In order to measure the effectiveness of the mix-zones, we study the success rate of them in providing the expected value of k. Here, the expected probability of getting k or more users, p is taken to be 0.9 and the value of k is varied from 2 to 11. Figure 52(a) shows the comparison of the success rate among the mix-zone approaches. A mix-zone is considered successful for an user if the user has at least k other users in its anonymity set with pairwise entropies greater than 0.9 under both timing and transition attacks. As evident from the figure, the TWB non-rectangular

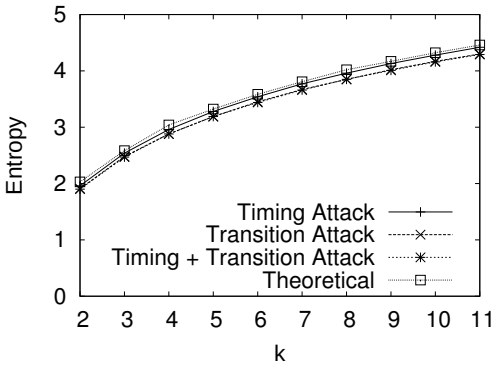


(a) Naive Rectangular

(b) TWB Rectangular



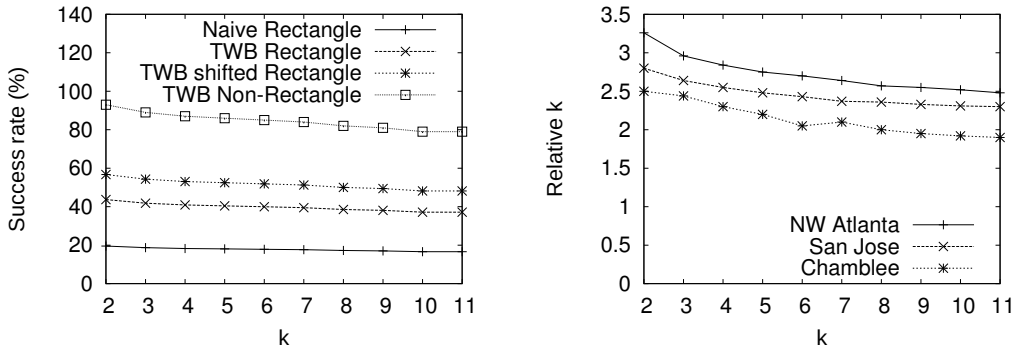
(c) TWB Shifted Rectangular



(d) TWB non-rectangular

Figure 51: Comparison of Entropy after attacks

mix-zones have the highest success rate, the other mix-zones have low success rate due to their lack of resilience to timing attack. In order to compare the level of anonymity offered by the mix-zones with the anonymity expected from them, we measure relative anonymity which is defined as the ratio of the value of obtained k to the value of expected k. Figure 52(b) shows the variation of relative-k of TWB non-rectangular mix-zones with respect to the expected value of k for different geographic maps. The expected success rate is set to 90%. The graphs show that the value of relative k lies within the range of 2 to 3, meaning that the mix-zone on an average offers two to three times the anonymity requested by the users.



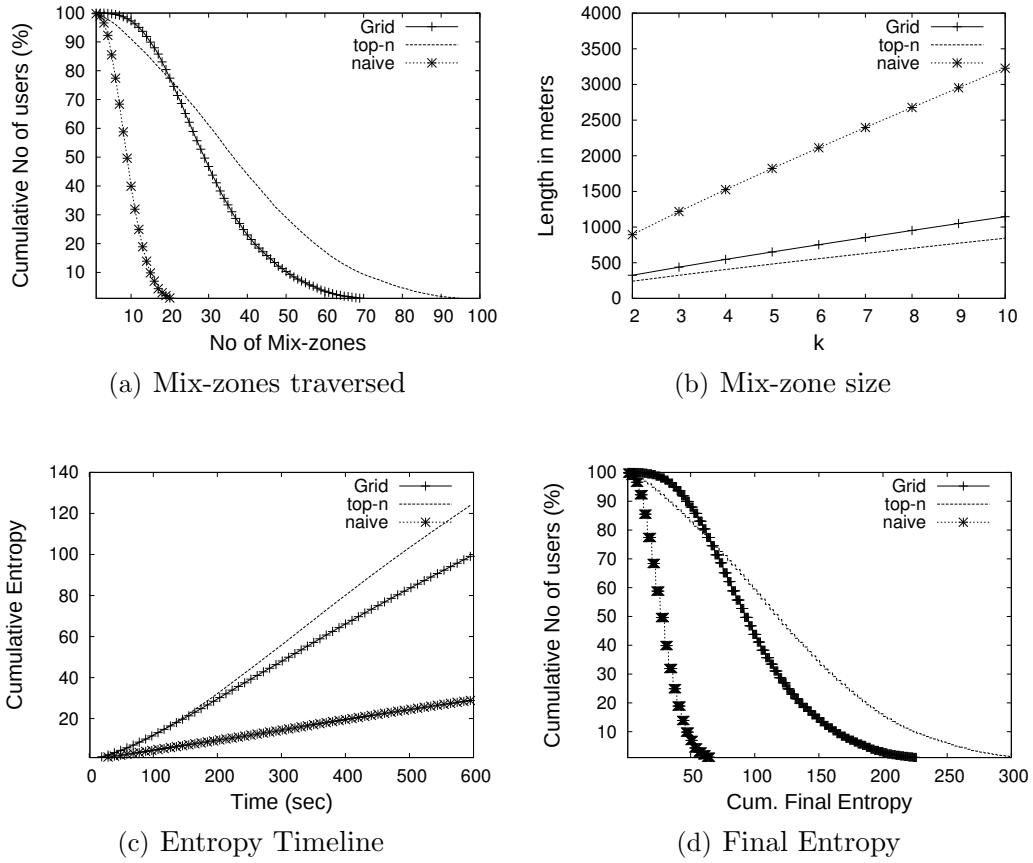
(a) Success Rate (b) Relative Anonymity

Figure 52: Success rate and Relative-k

5.7.2.3 Performance of Placement Techniques

We now study the performance of the various mix-zone placement algorithms in terms of the mix-zone size, spatial uniformity of placement, the average number of mix-zones traversed by the mobile clients and the entropy obtained during user's travel with the three mix-zone placement algorithms namely (i) Naive placement (ii) top-n (user and road characteristics-aware) placement and (iii) Grid (Quadree) based network-aware placement. The experiment uses the NW atlanta region map that contains 6831 road junctions, out of which the placement algorithms chooses 7% of the road intersections for deploying mix-zones that corresponds to 478 road junctions. The experiment uses a 10 minute simulation period. Figure 53(a) shows the cumulative distribution function (CDF) of the users in percentage for various number of mix-zones traversed during their trip. We find that users traverse less number of mix-zones in the naive mix-zone deployment scheme. We find more than 60% of the users traverse less than 10 mix-zones during their entire 10 minute travel. The top-n (user and road characteristics-aware) placement scheme enables users to pass through higher number of mix-zones as it basically finds all the intersections that have dense traffic. Here, users go through more number of mix-zones in short intervals of distance which may not be necessary. Such unnecessary traversal of mix-zones may deteriorate the quality of service for the mobile clients. In figure 53(a), we also find

that there is a significant percentage of users traversing less number of mix-zones. For example, more than 9% of the users traverse only less than 10 mix-zones during the 10 minute trajectory. This is due to the non-uniformity in the spatial distribution of the mix-zones. Hence, users traversing some part of the road networks go through few mix-zones while users travelling in other parts unnecessarily go through many mix-zones. The Grid (Quadree-based network-aware) deployment ensures a higher level of spatial uniformity in the distribution of mix-zones. In the Grid approach, we find that almost all users traverse at least 10 mix-zones during the 10 minute interval. Also, we find that users do not unnecessarily traverse many mix-zones, only few users travel a large number of mix-zones as compared to the top-n placement scheme.



(a) Mix-zones traversed

(b) Mix-zone size

(c) Entropy Timeline

(d) Final Entropy

Figure 53: Mix-zone Placement

Figure 53(b) shows the average size of the mix-zone in meters for values of k. We find that the naive placement approach leads to larger mix-zone sizes for even

small values of k as it lacks knowledge of the user arrival rate and user transition probability. Such large mix-zone size would significantly impact the service quality of the mobile users. The top-n scheme has the lowest mix-zone length among the three approaches as it identifies the most densely populated road junctions where even small mix-zone sizes yield higher k. However, the Grid placement scheme is also able to achieve almost similar mix-zone lengths as the top-n placement as it considers the road characteristics and user population factors in addition to the road network-aware spatial uniformity. Figure 53(c) shows the time line of cumulative entropy. The x-axis shows the time in seconds and the Y-axis represents the average cumulative entropy obtained. The naive placement shows low cumulative entropy, particularly in the beginning of the timeline (0 to 200 sec). Also, we find that both the top-n and Grid placements show similar average cumulative entropy in the beginning of the timeline although the top-n scheme has higher cumulative entropy at the later part of the timeline as users go through a large number of mix-zones with the top-n placement. In order to better understand the impact of the spatial uniformity of the mix-zone deployment on the cumulative entropy, in figure 53(d) we study the cumulative distribution of the users in percentage for various values of average final cumulative entropy at the end of the 10 minute interval. It shows a very similar trend as in figure 53(a). We find the naive placement scheme does not achieve high final cumulative entropy for all users. The top-n scheme has overall higher final cumulative entropy but has a significantly higher percentage of users having low final cumulative entropy. In the Grid approach, almost all users obtain higher final cumulative entropy and therefore the distribution has low skewness. Thus, the Grid placement scheme becomes the most effective choice in deploying mix-zones.

5.8 Related work

Location Anonymization has been proposed in [67] and adopted by several others [64], [98], [65], [33]. Some recent work on location anonymity had focused from the road network perspective [135] and [96]. The *XStar* framework presented in [135] performs location cloaking based on road-network-specific privacy and QoS requirements, striking a balance between the attack resilience of the performed protection and the processing cost of the anonymous query. The *Cachecloak* algorithm proposed in [96] uses cache prefetching to hide the exact location of the user by requesting the location based data along an entire predicted path. While the approaches based on location cloaking do not work for applications that require an exact point location of the mobile user, the approach presented in [96] is not suitable when users ask different queries as they move.

The concept of mix-zones to change pseudonyms for location privacy has been introduced in [35] and the idea of building mix-zones at road intersections has been proposed in [60] and [39]. A formulation for optimal placement of mix-zones in a road map is discussed in [61], however such optimal solutions to the placement problem become NP-complete for even small road networks. Almost all of these mix-zone techniques follow a straight forward approach of using rectangular or circular shaped zones and their construction methodologies do not take into account the effect of timing and transition attacks in the construction process. The approaches presented in MobiMix differ from these in two folds: firstly, the mix-zone construction process of MobiMix tries to minimize the effect of timing and transition attacks based on the characteristics of the underlying road network at the construction time and secondly, the framework attempts to address the issue of guaranteeing an expected value of anonymity by taking into consideration the statistics of user arrival users and other factors in the road network.

5.9 Summary

We have presented MobiMix, a framework for building attack resilient road network mix-zones for protecting the location privacy of mobile clients accessing location-based cloud services. We first provided a formal analysis of the theoretical mix-zone model and the vulnerabilities of applying them to road networks where some of the assumptions may be violated. We presented a suite of road network mix-zone construction and placement techniques that consider number of factors such as the mix-zone geometry, the statistics of the user population, and the spatial and velocity constraints on the movement patterns of the users. We show analytically and experimentally that the MobiMix construction and placement techniques are efficient and more resilient to timing and transition attacks compared to the existing mix-zone approaches.

The next chapter extends the MobiMix approach to support continuous privacy-conscious location-based services in the cloud where the mobile client exposes continuity in the information obtained from the cloud service in addition to exposing location information.

CHAPTER VI

PRIVACY-PRESERVING PROVISIONING OF CONTINUOUS LOCATION-BASED SERVICES

6.1 Introduction

Continuous location-based queries are gaining growing interest and attention in both mobile service industry as well as in mobile cloud computing. Examples of continuous queries (CQs) include "informing me the nearest gas stations coming up along the highway I-85 south every 1 minute in the next 30 minutes" or "show me the restaurants within 2 miles every two minutes during the next hour". Many consider continuous spatial queries as a fundamental building block for continuous provisioning of location services to mobile users traveling on the roads.

Continuous query attacks (CQ-attacks) refer to the query correlation attacks. Concretely, a CQ represents a time series of query evaluations of the same query within a given validity time window. For example, if Alice is traveling on I-85 south and requested a CQ service: show me the restaurants within 2 miles every two minutes during the next hour, then the CQ server will process this CQ as a standing query for 1 hour period upon its installation and it will consist of a sequence of 30 evaluations along the trajectory of Alice. The CQ-attack refers to the risk that an adversary can perform inference attacks by correlating the semantic continuity in the time series of query evaluations of the same CQ and the inherent trajectory of locations. In this chapter we show that such CQ-attacks can intrude location privacy of mobile users (i.e., exposing the identity of Alice), even though the locations of mobile users are anonymized through well-known location anonymization techniques.

A fair amount of research efforts have been dedicated to protecting location privacy of mobile travelers. We can broadly classify the state of art research and development results into two categories. The first category is represented by location cloaking techniques [67, 33, 64, 98, 135]. Spatial location cloaking typically adds uncertainty to the location information exposed to the location query services by increasing the spatial resolution of a mobile user's locations while meeting location k-anonymity and/or location l-diversity [33]. More specifically, the spatially cloaked region is constructed to ensure that at least k users (*location k anonymity*) are located in the same region, which contains l different static sensitive objects (locations). Spatial cloaking is effective for snapshot queries but vulnerable to CQ-attacks. The second class of location anonymization techniques is represented by mix-zone development [35, 60, 61, 39, 104]. Mix-zones are spatial regions where a set of users enter, change pseudonyms in such a way that the mapping between their old and new pseudonyms is not revealed. Also inside a mix-zone, no applications can track user movements. Mix-zones break the continuity of location exposure by introducing uncertainty such that it is very hard to perform correlation attacks to link old pseudonym with new pseudonym of mobile users. However, neither spatial cloaking nor mix-zone techniques are resilient to CQ attacks as they are vulnerable to query correlation. Concretely, with spatial cloaking, an adversary can infer user movement by performing query correlation attacks over the adjacent or overlapped cloaking boxes. Similarly, mobile users requesting CQ services are vulnerable under CQ-attacks even though their movements on the road networks are protected by mix-zone anonymization. With these problems in mind, in this chapter we present a delay-tolerant mix-zone framework for protecting location privacy of mobile users with continuous query services in a mobile environment. First, we describe and analyze the continuous query correlation attacks (CQ-attacks) that perform query correlation

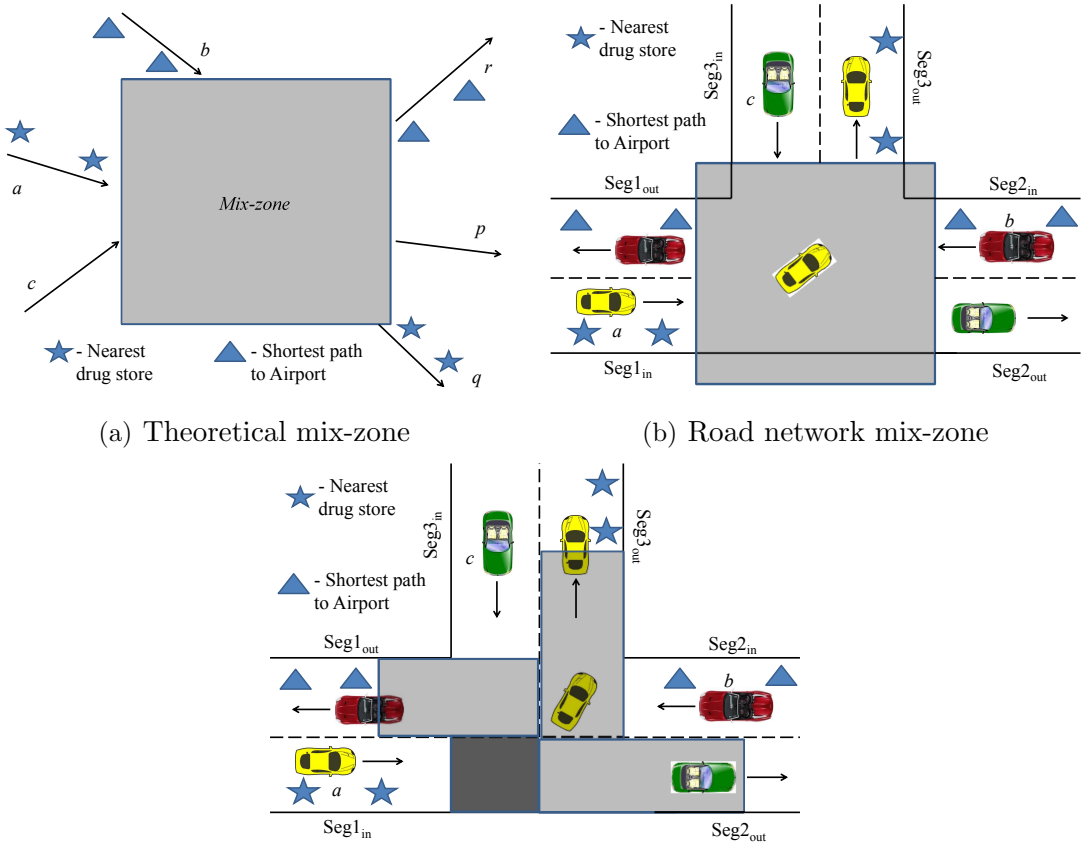
based inference to break the anonymity of road network-aware mix-zones. We formally study the privacy strengths of the mix-zone anonymization under the CQ-attack model and identify that providing high initial anonymity in the mix-zone model is the key to anonymizing continuous queries in a mix-zone framework. We argue that spatial cloaking or temporal cloaking over road network mix-zones is ineffective and susceptible to attacks that carry out inference by combining query correlation with timing correlation (CQ-timing attack) and transition correlation (CQ-transition attack) information.

Next, we introduce three types of delay-tolerant road network mix-zones (i.e., temporal, spatial and spatio-temporal) that are free from CQ-timing and CQ-transition attacks and in contrast to conventional mix-zones, perform a combination of both location mixing and identity mixing of spatially and temporally perturbed user locations to achieve stronger anonymity under the CQ-attack model. In the delay-tolerant mix-zone model, users expose spatially or temporally perturbed locations outside the mix-zone area. However, on the exit of each temporal delay tolerant mix-zone, the mix-zone changes their perturbed locations by introducing a random temporal shift to their already perturbed locations. Similarly, the spatial delay tolerant mix-zones introduce a random spatial shift to the spatially perturbed locations when the users exit them. While conventional mix-zones only change pseudonyms inside them, the additional ability of delay-tolerant mix-zones to change and mix user locations brings greater opportunities for creating anonymity. Our third type of delay tolerant mix-zones, namely spatio-temporal delay-tolerant mix-zones effectively combine temporal delay-tolerant and spatial delay tolerant mix-zones to obtain the highest anonymity for continuous queries while making acceptable tradeoff between anonymous query processing cost and temporal delay incurred in anonymous query processing. We evaluate the proposed techniques through extensive experiments conducted using traces

produced by GTMobiSim [108] on different scales of geographic maps. Our experiments show that the delay-tolerant mix-zone techniques are efficient and offer the desired level of anonymity for continuous queries.

6.2 Mix-zones and CQ-attacks

In this section, we introduce the basic mix-zone concepts, illustrate the vulnerabilities of mix-zones to continuous query correlation attacks (*CQ-attacks*) and present a formal analysis of the continuous query anonymization problem.



(a) Theoretical mix-zone

(b) Road network mix-zone

(c) Non-rectangular mix-zone

Figure 54: Mix-zone anonymization and its risks under CQ-attack

6.2.1 Mix-zone concepts

A mix-zone of k participants refers to a k-anonymization region in which users can change their pseudonyms such that the mapping between their old and new

pseudonyms is not revealed. In a mix-zone, a set of k users enter in some order and change pseudonyms but none leave before all users enter the mix-zone. Inside the mix-zone, the users do not report their locations and they exit the mix-zone in an order different from their order of arrival, thus, providing unlinkability between their entering and exiting events. The properties of a mix-zone can be formally stated as follows:

Definition 3 *A mix-zone Z is said to provide k-anonymity to a set of users A iff*

1. *The set A has k or more members, i.e., $|A| \geq k$.*
2. *All users in A must enter the mix-zone Z before any user $i \in A$ exits. Thus, there exists a point in time where all k users of A are inside the zone.*
3. *Each user $i \in A$, entering the mix-zone Z through an entry point $e_i \in E$ and leaving at an exit point $o_i \in O$, spends a completely random duration of time inside.*
4. *The probability of transition between any point of entry to any point of exit follows a uniform distribution. i.e., a user entering through an entry point, $e \in E$, is equally likely to exit in any of the exit points, $o \in O$.*

Figure 54(a) shows a mix-zone with three users entering with pseudonyms a, b and c and exiting with new pseudonyms, p, q and r. Here, given any user exiting with a new pseudonym, the adversary has equal probability of associating it with each of the old pseudonyms a, b and c and thus the mix-zone provides an anonymity of $k = 3$. Therefore, the uncertainty of an adversary to associate a new pseudonym of an outgoing user i' to its old pseudonym is captured by Entropy, $H(i')$ which is the amount of information required to break the anonymity.

$$H(i') = -\sum_{j \in A} p_{i' \to j} \times log_2(p_{i' \to j})$$

where $p_{i' \to j}$ denotes the probability of mapping the new pseudonym, i' to an old pseudonym, j. Here note that when users change pseudonyms inside mix-zones along their trajectories, an adversary observing them loses the ability to track their movements.

Unlike the theoretical mix-zones, mix-zones constructed at road intersections (Figure 54(b)) may violate some conditions. For instance, in a road network mix-zone, users do not stay random time inside while entering and exiting the mix-zone [60, 104] and also violate the assumption of uniform transition probabilities in taking turns. Such violations provide additional information to the adversary in inferring the mapping between the old and new pseudonyms. Precisely, the timing information of users entry and exit in a road network mix-zone leads to timing attack and the non-uniformity in the transition probabilities at the road intersection leads to transition attack respectively. The MobiMix road network mix-zone model (Definition 4) and the construction techniques [104] deal with the challenges of constructing road network mix-zones that are resilient to such timing and transition attacks. Accordingly a road network mix-zone is defined as:

Definition 4 *A road network mix-zone offers k-anonymity to a set A of users if and only if:*

1. *There are k or more users in the anonymity set A.*
2. *Given any two users $i, j \in A$ and assuming i exiting at time t, the pairwise entropy after timing attack should satisfy the condition: $H_{pair}(i, j) \geq \alpha$.*
3. *Given any two users $i, j \in A$, the pairwise entropy after transition attack should satisfy the condition: $H_{pair}(i, j) \geq \beta$.*

Here the pairwise entropy, $H_{pair}(i, j)$ between two users i and j represents the entropy obtained by considering i and j to be the only members of the anonymity

set. In comparison, a theoretical mix-zone offers a high pairwise entropy of 1 for all pairs of users. Therefore, a good road network mix-zone would offer a pairwise entropy close to 1 for all pairs of users in the anonymity set. The non-rectangular road network mix-zone geometry (Figure 54(c)) proposed in [104] enables each user to travel along only one road segment inside the mix-zone and thereby avoids timing attack due to difference in speed distributions. Similarly, the MobiMix road network mix-zone model discusses the necessary criteria to ensure transition attack resilience in the mix-zone construction process. However as we discuss next, when users run continuous queries, any type of mix-zone is prone to CQ-attacks.

6.2.2 CQ-attack

When a user is executing a continuous query, even though her pseudonym is changed whenever she enters a road network mix-zone, an adversary may simply utilize the consecutive snapshots of the query to reveal the correlation between the old and new pseudonyms. Consider the example in Figure 54(a) where three users enter with pseudonyms a, b and c and exit with new pseudonyms p, q and r. The attacker finds that before entering the mix-zone, users a and b run continuous queries on obtaining nearest drug store and shortest path driving directions to the airport respectively. Upon their exits, the attacker again finds more instances of their corresponding continuous queries with different pseudonyms, q and r. Here, although users a and c change their pseudonyms to q and r, the continuous exposure of their CQ information breaks their anonymity. Similar attack can happen in a road network mix-zone as shown in Figure 54(b) where three users with pseudonyms, a, b and c enter and leave the mix-zone. As users a and b are running continuous queries, the attacker finds an instance of $a's$ continuous query before entering the mix-zone and when user a exits with a new pseudonym, say α and receives another instance of the same query, the attacker infers that the new pseudonym α must correspond to the old pseudonym

a. To the best of our knowledge, no existing road network mix-zone technique is effective against CQ-attack. For instance, we find in Figure 54(c) that even the non-rectangular mix-zone [104] that is most effective against road network timing attack is also prone to the CQ-attack.

Anonymity under CQ-attack model: When a user executes a continuous query, it induces a trajectory corresponding to the movement of the user even though the user's pseudonym is changed whenever she crosses a road network mix-zone. When a mobile user starts a continuous query from a personal location like office or home address, then the user's trajectory induced by the continuous query can be easily linked with the user's personal location and hence to user's real identity even though the pseudonyms are changed time to time. Therefore, in the proposed mix-zone anonymization model, any mobile user who wishes to obtain CQ service first moves to the nearest mix-zone and starts to run the CQ service from the mix-zone. Consider the user with pseudonym, a in Figure 55 who wants to start a CQ service for obtaining k-nearest drug stores. User a first goes to the nearest mix-zone and starts the first instance of the query after exiting the mix-zone with a new pseudonym e. Here, the attacker becomes confused to associate this continuous query with the users $\{a, b, c\}$ who entered the mix-zone as each of them have equal likelihood of starting this continuous query. Hence the continuous query obtains an initial anonymity of $k = 3$. However, it should be noted that at the subsequent mix-zones, the query correlation reveals the mapping between the old and new pseudonyms of the continuous query. For example, in the second mix-zone the mapping between new pseudonym j and old pseudonym e is revealed and similarly in the third mix-zone, the mapping between new pseudonym m and old pseudonym j is revealed. Thus, the anonymity strengths of the mix-zones is weakened under the continuous query correlation attack model.

In contrast, if users in the system ask only snapshot queries, then user a's anonymity

keeps on increasing as a traverses through more mix-zones as it does not run continuous queries. For example, when user a changes its pseudonym from e to j in the second mix-zone, its anonymity increases from $k = 3$ (corresponding to $\{a, b, c\}$) to $k = 5$ (corresponding to $\{a, b, c, f, g\}$). Hence, even though the initial anonymity is low ($k = 3$), in the snapshot query model, the anonymity gained in the intermediate mix-zones add to the user's anonymity.

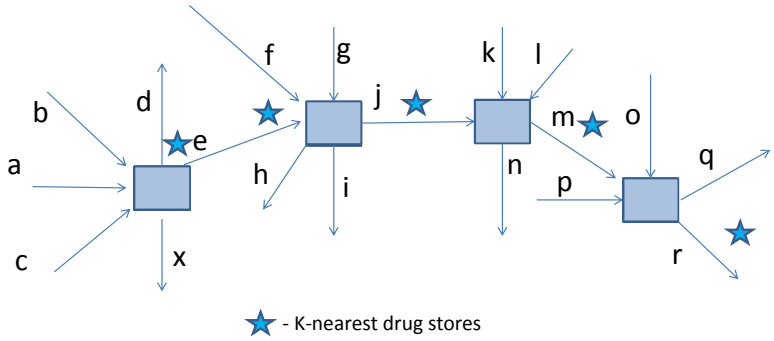


Figure 55: CQ-induced trajectory

However for a continuous query, its initial anonymity forms the major component and intermediate mix-zones add anonymity only when users in the intermediate mix-zones ask the same query. For instance, if m out of the k users traversing an intermediate mix-zone run continuous queries and if there are R number of unique continuous queries run by the m users and if A^r is the set of users running the continuous query, Q_r, $1 \leq r \leq R$, then the entropy of each continuous query user, i executing the query, Q_r and exiting the mix-zone with a new pseudonym, i' is given by

$$H(i) = -\sum_{j \in A^r} p_{i' \to j} \times log_2(p_{i' \to j})$$

where $p_{i' \to j}$ denotes the probability of mapping the user exiting with the new pseudonym, i' to an old pseudonym, j that runs the same continuous query. Therefore, for a user starting to execute a CQ from mix-zone m_1, if f_i users out of the m_i continuous

query users in the i^{th} mix-zone execute the same CQ, then the entropy X of the user executing the continuous query is given by

$$X = log|k_1 - m_1| + \sum_{2 \leq i \leq n} log|f_i|$$

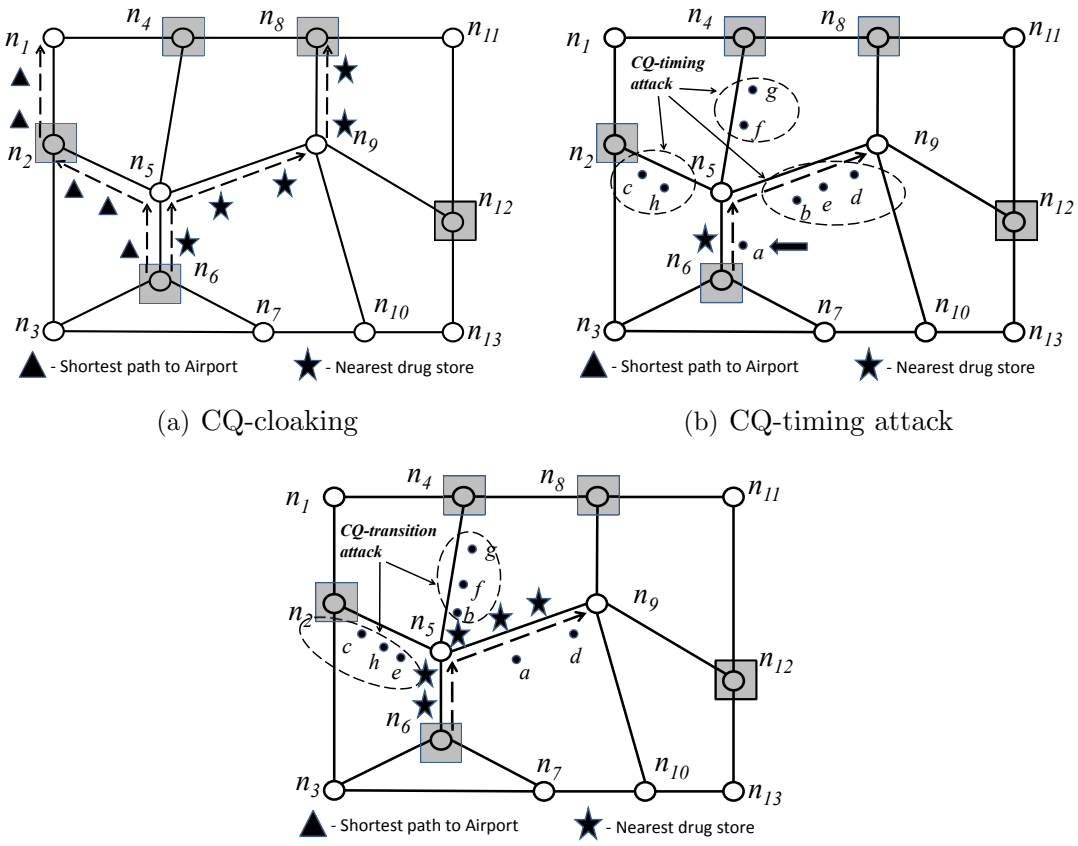
where $log|k_1 - m_1|$ represents the initial anonymity of the user while starting the continuous query in mix-zone, m_1.

The goal for designing CQ-attack resilient solutions is to increase the anonymity strengths of the mix-zones by considering the fact that the attacker has the continuous query correlation information at the intermediate mix-zones to infer and associate the CQ induced trajectory with its user. Note that the initial anonymity forms the major component of the anonymity under the CQ-attack model and therefore it is important that the mix-zones provide high initial anonymity for the continuous queries so that even when the attacker breaks the anonymity in the subsequent mix-zones, the initial anonymity remains sufficient to meet the required privacy level. For instance, if the first mix-zone in the above example provides a higher anonymity, say $k = 100$ instead of $k = 3$, then the initial anonymity may be sufficient to meet the privacy requirements of the continuous query even though the attacker breaks the anonymity obtained in the intermediate mix-zones under the CQ-attack model. In the next subsection, we discuss CQ-cloaking techniques (spatial cloaking or temporal cloaking of CQs) over road network mix-zones as a candidate approach for achieving higher query anonymity and show that it is ineffective and susceptible to attacks that combine CQ information with timing correlation (CQ-timing attack) and transition correlation (CQ-transition attack).

6.2.3 CQ-cloaking approach and its vulnerabilities

In the CQ-cloaking approach, the continuous queries are either temporally or spatially perturbed while the snapshot queries continue to be unperturbed. In the CQ-cloaking approach, the locations used by the CQ is perturbed such that a continuous query

originating from a mix-zone is indistinguishable from at least k users traversing the mix-zone. While this technique does not make changes to the mix-zone model, we show that the location exposure of snapshot queries makes the CQ anonymization susceptible to CQ-timing attack and CQ-transition attack.



(a) CQ-cloaking

(b) CQ-timing attack

(c) CQ-transition attack

Figure 56: Continuous Query: Timing and Transition attacks

Consider the example shown in Figure 56(a) where we have two CQs labeled as star CQ and triangle CQ respectively. The square nodes represent road network mix-zones. We observe two different CQ traces starting from the mix-zone at road junction n_6. The triangle trace crossed the junctions n_6, n_5 and n_2 and the star trace crossed the junctions n_6, n_5 and n_9 respectively and each star or triangle represents one snapshot execution of the corresponding CQ. Intuitively, if we delay the execution of the individual CQ snapshots of CQ users starting at mix-zone n_6 such that at least

k_c users leave the mix-zone within the temporal delay, it will make it harder for an adversary to associate the CQ-induced trajectory with the corresponding CQ user. For instance, in Figure 56(a), if the continuous query on the shortest path to the airport (marked by stars) originating from the mix-zone n_6 is perturbed temporally in such a way that there are k or more users coming out of the mix-zone at road junction n_6 within the continuous query's temporal cloaking window, then from the attacker's perspective, the query could have originated from any of the k users who entered the mix-zone within the time window. CQ-spatial cloaking is similar to CQ-temporal cloaking except that instead of delaying the snapshots of the continuous queries, the CQ exposes a larger spatial region such that there are k or more users within the spatial region.

CQ-timing Attack: As mentioned earlier, CQ-cloaking techniques are vulnerable to CQ-timing attack when users in the anonymity set violate the steady motion assumption, i.e., if all users do not travel at the imposed speed of the road segment. In the example shown in Figure 56(b), we find that users with pseudonyms a, b, c, d, e, f, g and h enter the mix-zone during the continuous query's temporal cloaking window, d_{tmax}. When the steady motion assumption fails, user a travels slowly and stays on segment $\overline{n_5 n_6}$ while other users move ahead of the segment, $\overline{n_5 n_6}$. If user a is the issuer of the continuous query, then the continuous query would stay on segment, $\overline{n_5 n_6}$ even though it is executed with a temporal delay while other users of the anonymity set move ahead. By observing this, the attacker can eliminate the low probable members and identify the issuer of the continuous query with high confidence. Concretely, we assume the attacker has knowledge of the maximum temporal delay, d_{tmax} used by the users. Let $A_{C(i)}$ represents the anonymity set of the continuous query, C_i, for each user, $j \in A_{C(i)}$ and let $M_{C_i,j}$ be the likelihood that the continuous query originates from user j. Since only continuous query's location is temporally perturbed, the attacker observes the movement of the users through their location exposure for

snapshot queries. Let $loc(j,t)$ and $loc(C_i,t)$ represent the location of user j and the location of the temporally cloaked continuous query, C_i observed by the attacker at time t. If $X_j^{t-d_t}(loc(C_i,t))$ is a Boolean variable indicating if the attacker observed user j moving through the location, $loc(C_i,t)$ at time $t-d_t$, the likelihood $M_{C_i,j}$ is given by

$$M_{C_i,j} = \sum_{0 \le d_t \le d_{tmax}} q(C_i,d_t) \times X_j^{t-d_t}(loc(C_i,t))$$

where $q(C_i,d_t)$ denotes the probability that the continuous query, C_i uses a temporal lag d_t. Based on the proportion of the likelihoods of the users, the attacker can assign the probability, $Q_{C_i,j}$ that represents the probability of user j to be source of the continuous query, C_i.

$$Q_{C_i,j} = \frac{M_{C_i,j}}{\sum_{j \in A(C_i)} M_{C_i,j}}$$

Based on the probabilities, the attacker may either ignore the low probable members from consideration or narrow down the search to the high probable members. We note that similar attack is also possible in the case of CQ-spatial cloaking when the steady motion assumption fails.

CQ-transition Attack: Additionally, CQ-cloaking techniques are also prone to CQ-transition attack. When the transitions taken by a subset of the users in the anonymity set differ from that of the user executing the query, then those members can be eliminated based on transition correlation. For example, in Figure 56(c), at road intersection n_5, users c and e and h take a left turn on to the road segment, $\overline{n_2n_5}$ whereas users b, f and g move straight on segment, $\overline{n_4n_5}$ and users a and d turn right on to segment, $\overline{n_5n_9}$. When the continuous query uses CQ-temporal cloaking or CQ-spatial cloaking and follows the querying user after a temporal delay or using a spatial cloaking region, from the transition taken by the continuous query from segment, $\overline{n_6n_5}$ to $\overline{n_4n_5}$, the adversary will be able to eliminate the users, c, e, h, b, f and g from consideration as their transitions differ from that of the continuous query.

Concretely, if Y_j is a Boolean variable indicating if the path taken by user j is the same as that of the continuous query, C_i, then the probability $Q_{C_i,j}$ of user j being the source of the continuous query, C_i is given by

$$Q_{C_i,j} = \begin{cases} \frac{1}{\sum_{j \in A(C_i)} Y_j} & \text{if } Y_j = 1 \\ 0 & \text{otherwise} \end{cases}$$

In the next section, we introduce the concept of delay-tolerant mix-zones that use a modified mix-zone model to perform both location mixing and identity mixing to achieve higher anonymity for continuous queries as compared to only identity mixing in the conventional mix-zone model. Delay -tolerant mix-zones perturb both continuous queries and snapshot queries and are free from CQ-timing and CQ-transition attacks.

6.3 Delay-tolerant Mix-zones

Delay-tolerant mix-zones combine mix-zone based identity privacy protection with location mixing to achieve high anonymity that is otherwise not possible with conventional mix-zones. In the delay-tolerant mix-zone model, users expose spatially or temporally perturbed locations outside the mix-zone area. However, on the exit of each delay tolerant mix-zone, the mix-zone changes their perturbed locations by introducing a random temporal or spatial shift to their already perturbed locations. While conventional mix-zones only change pseudonyms inside them, the additional ability of delay-tolerant mix-zones to change and mix user locations brings greater opportunities for creating anonymity.[1] Therefore, the anonymity strength of delay-tolerant mix-zones comes from a unique combination of both identity mixing and location mixing.

[1]This model is analogous to anonymous delay-tolerant routing in mix networks where network routers have the additional flexibility to create anonymity by delaying and reordering incoming packets before forwarding them [120].

Before presenting the detailed analysis of the privacy strengths of the delay-tolerant mix-zones, we first illustrate the concept of delay-tolerant mix-zones with an example temporal delay-tolerant mix-zone. Table 17 shows the entry and exit time of users in a conventional rectangular road network mix-zone. We find that user a enters the mix-zone as $t = 100$ and exits at time $t = 104$. Similarly the other users enter and exit as shown in Table 17. Here the adversary may know that the average time taken by the users to cross the mix-zone is 4 sec. Therefore when user a exits at a' at time $t = 104$, the attacker can eliminate users e and f from consideration as they have not even entered the mix-zone by the time user a exits[2]. Similarly, the adversary can eliminate users o and n from consideration based on timing inference that users o and n have exited the mix-zone by the time a and b enter the mix-zone. Therefore when a exits as a', the attacker has uncertainty only among the users $\{a, b, c, d, m\}$. Also, among the users $\{a, b, c, d, m\}$, the attacker can eliminate more users through sophisticated reasoning based on timing inference described later.

$User$	t_{in}	t_{inside}	t_{out}
o	94	4	98
n	96	4	100
m	98	4	102
a	100	4	104
b	101	4	105
c	103	4	107
d	103	4	107
e	106	4	110
f	108	4	112

Table 17: Conventional Road Network Mix-zone

However, in the delay-tolerant mix-zone model, each user uses a temporal delay, d_t within some maximum tolerance, d_{tmax}. Inside the mix-zone, the temporally perturbed location of each user is assigned a random temporal shift. In the delay-tolerant mix-zone example shown in Table 18, we find that user a initially uses a temporal delay, d_{told} of 4 sec and inside the mix-zone it is shifted randomly to 16 sec. Here d_{tmax} is assumed as 20 sec. Therefore when user a exits as a', it becomes possible that

[2]For the sake of example simplicity, we assume that the users take the average time of 4 sec to cross the mix-zone, in a real road intersection, it could actually take slightly longer or shorter time to cross based on the speed of travel.

User	Observed t_{in}	t_{inside}	d_{told}	d_{tnew}	Observed t_{out}
w	81	4	4	20	105
v	84	4	7	20	108
u	84	4	7	20	108
s	87	4	10	20	111
r	89	4	12	20	113
q	90	4	13	20	114
p	92	4	15	20	116
o	94	4	18	20	118
n	96	4	19	20	120
m	98	4	20	18	120
a	100	4	4	16	120
b	101	4	4	15	120
c	103	4	7	13	120
d	103	4	7	13	120
e	106	4	10	10	120
f	108	4	12	8	120
g	109	4	13	7	120
h	111	4	15	5	120
i	113	4	18	3	120
j	115	4	19	1	120
k	117	4	20	0	120
l	118	4	20	0	121

Table 18: An example temporal Delay-tolerant mixing

many users can potentially exit in the exit time of user a. The example in Table 18 shows one possible assignment of new temporal delays, d_{tnew} for other users in order for them to exit at the same time as a'. Thus, during the exit of user a as a', the attacker is confused to associate the exiting user a' with the members of the anonymity set, $\{a, b, c, d, e, f, g, h, i, j, k, m, n\}$. In principle, users' new temporal delays, d_{tnew} are randomly shifted inside the mix-zone ensuring the possibility of each of the users to exit at the exit time of each other and thus the delay-tolerant mix-zone model provides significantly higher anonymity compared to conventional mix-zones. Such high anonymity provides the initial anonymity required for the continuous queries under the CQ-attack model.

Timing attack in delay-tolerant mix-zone: Before we proceed to analyze the privacy strengths of the delay-tolerant mix-zone, we formally define the timing attack described above. The attacker observes the time of entry, $t_{in}(i)$ and time of exit $t_{out}(i)$ for each user entering and exiting the mix-zone. When the attacker sees an user i' exiting, he tries to map i' to one of the users of the anonymity set, A_i. The attacker assigns a probability, $p_{i' \to j}$ that corresponds to the probability of mapping i' to j, where $j \in A$ based on the likelihood of user j exiting at the exit time of i', denoted by $t_{out}(i')$. Once the mapping probabilities are computed, the

attacker can utilize the skewness in the distribution of the mapping probabilities to eliminate some low probable mappings from consideration and narrow down his inference to only the high probable mappings. Consider the example shown in Table 17 for the conventional mix-zone case, when user a exits as a', we have the anonymity set, $A = \{a, b, c, d, e, f, o, n, m\}$. Clearly the probability of mapping a' to the users $\{e, f, o, n\}$ is zero as they either enter the mix-zone after a' exits or leave the mix-zone before a even enters. However, between the users $\{a, b, c, d, m\}$, the attacker can assign a non-zero likelihood of resembling a' as they have been inside the mix-zone while a was present. Let these likelihoods be 0.37, 0.35, 0.1, 0.1 and 0.12 respectively. In this case, we show that it is easy to analyze the anonymity based on pairwise entropy. The pairwise entropy, $H_{pair}(a, b)$ between two users a and b during the exit of a' is the entropy obtained by considering a and b to be the only members of the anonymity set. In this case, we have two mapping probabilities, $p_{a' \to a}$ and $p_{a' \to b}$ which are computed as: : $p_{a' \to a} = \frac{0.37}{0.37+0.35} = 0.513$, $p_{a' \to b} = \frac{0.35}{0.37+0.35} = 0.486$. Hence the pairwise entropy $H(a, b)$ is given by

$$H_{pair}(a, b) = -(p_{a' \to a} log p_{a' \to a} + p_{a' \to b} log p_{a' \to b})$$

Therefore, $H_{pair}(a, b) = 0.99$. In general, a mix-zone defines a short mix-zone time window, τ such that a set of users entering within the short time window, τ have high pairwise entropy with each other. In the above example, if we take $\tau = 1$ sec, then we find users a and b enter within 1 sec and hence they have high pairwise entropy close to 1. However, users a and c do not enter within the mix-zone time window of 1 sec and therefore we find that their pairwise entropy is lower($H_{pair}(a, c) = 0.75$). In comparison, a theoretical mix-zone (recall Definition 1) ensures a uniform distribution for all possible mappings between old and new pseudonyms and therefore ensures a high pairwise entropy of 1.0 for all pairs of users in the anonymity set. Thus, the effective anonymity set of a road network is assumed to comprise of only those users that have high pairwise entropy with each other.

Definition 5 *A delay-tolerant road network mix-zone offers k-anonymity to a set A of users if and only if:*

1. *There are k or more users in the anonymity set A.*
2. *Given any two users $i, j \in A$ and assuming i exiting at time t, the pairwise entropy after timing attack should satisfy the condition: $H_{pair}(i, j) \geq \alpha$.*
3. *Given any two users $i, j \in A$, the pairwise entropy after transition attack should satisfy the condition: $H_{pair}(i, j) \geq \beta$.*

In the above example, only users $\{a, b\}$ will belong to the effective anonymity set of a under the conventional mix-zone model as only they have high pairwise Entropy with each other. However as discussed earlier, under the delay-tolerant mix-zone model more number of users ($\{a, b, c, d, e, f, g, h, i, j, k, m, n\}$) will have high pairwise entropy with each other and belong to the anonymity set of user a. Here, in addition to the pairwise entropy with respect to timing attack, the pairwise entropy with respect to transition attack is also considered for cases where the road intersections do not have uniform transition probabilities to different segments (Condition 3). In such cases, the effective anonymity set will contain only those members who enter from road segments such that their transition probability to the exit segment of user a is similar to the transition probability of their exit segments. As the delay-tolerant mix-zones primarily influence the impact of timing attack, we focus our discussion based on timing attack. However, in our experiments using real road networks, we take into account the fact that road intersections do not have uniform transition probability and accordingly construct the anonymity set with only those members which have similar transition probability as discussed in [104]. Next, we present the design and formal analysis of three proposed delay-tolerant mix-zone techniques namely (i)temporal delay-tolerant mix-zones, (ii) spatial delay-tolerant mix-zones and (iii) spatio-temporal delay-tolerant mix-zones.

6.3.1 Temporal Delay-tolerant Mix-zones

As discussed before, in a temporal delay tolerant mix-zone, every mobile user delays the location exposure with a randomly chosen delay, d_t within the maximum temporal tolerance, d_{tmax}. The temporal time window, d_{tmax} is chosen based on the arrival rate of the users in the road junction so as to ensure an expected number of users arriving into the mix-zone within the temporal tolerance, d_{tmax}. Note that the random delay used by a mobile client does not change during its travel between mix-zones. Only when the mobile client enters a new mix-zone, its temporal delay is randomly shifted to a new value within the temporal window, d_{tmax}. Based on the delayed exposure of users' location information, the attacker knows their current temporally cloaked location.

Based on the temporally cloaked location exposed by the users, the adversary observes each user i entering the mix-zone at a temporally cloaked time $tcloak_{in}(i)$ and exiting at a temporally cloaked time $tcloak_{out}(i')$ with a new pseudonym i'. The speed followed by the users in a road segment is assumed to follow a Gaussian distribution with a mean μ and standard deviation σ, where μ and σ are specific to each road class category. For user i, the set of all other users who had entered the mix-zone during the time window defined by $|tcloak_{in}(i) - \tau - d_t|$ to $|tcloak_{in}(i) + \tau + d_t|$, forms the anonymity set of i, denoted as A_i where τ is a small value and represents the mix-zone time window. Let t be the temporally cloaked exit time of user i, which is also $tcloak_{out}(i')$.

For each user, j in the anonymity set, A_i, we compute $p_{i' \to j}$, the probability that the exiting user i' at temporally cloaked time, $tcloak_{out}(i')$ is j and $p_{i' \to i}$ be the probability that the exiting user is i. Let $P(j,t)$ define the probability that user j exits the mix-zone at the cloaked time, t. Here, the observed movement of the temporally cloaked user location in the mix-zone is controlled by two factors: the speed of the user inside the mix-zone and the change in the temporal delay used by

the user. Let $P_v(j,x)$ be numerically equal to the probability that user j takes x units of time to traverse the mix-zone region. It is computed based on the speed distribution on the road segments. Let $P_t(j,t)$ be the probability that the temporal delay of user j is shifted by t seconds after exiting the mix-zone. In order for user j to exit at temporally cloaked time, $tcloack_{out}(i')$, it depends on both the temporal shift introduced to user j in the mix-zone as well as the time j takes to cross the mix-zone. Thus, $P(j,t)$ is given by

$$P(j,t) = \int_0^\infty P_v(j,x) \times P_t(j, t - tcloack_{in}(j) + x)dx$$

Here we note that our temporal location mixing algorithm assigns a new temporal delay, d_{tnew} based on the current temporal delay, d_t and the maximum temporal delay, d_{tmax}.

$$d_{tnew} = |d_{tmax} - d_t|$$

and hence it ensures a uniform distribution of shift values, $P_t(j,t)$ while also ensuring that the temporal lags of the users are uniformly distributed. Similar to $P(j,t)$, we can also obtain $P(i,t)$ based on the temporally cloaked arrival time of i, $tcloack_{in}(i)$. We have

$$P(i',t) = P(i,t) + P(j,t)$$

Here, $P(i',t)$ is the cumulative likelihood of both i or j exiting as i' as i' is either of them. Therefore, the probability of i' being j when i' exits at time t, denoted as $p_{i' \to j}(t)$ is given by the following conditional probability

$$p_{i' \to j}(t) = P((j,t)/(i',t))$$

Similarly, the probability of i' being i, $p_{i' \to i}(t)$ is given by

$$p_{i' \to i}(t) = P((i,t)/(i',t))$$

and the pair-wise entropy after timing attack, $H_{pair}(i,j)$ between users i and j when

i exits as i' can be obtained as

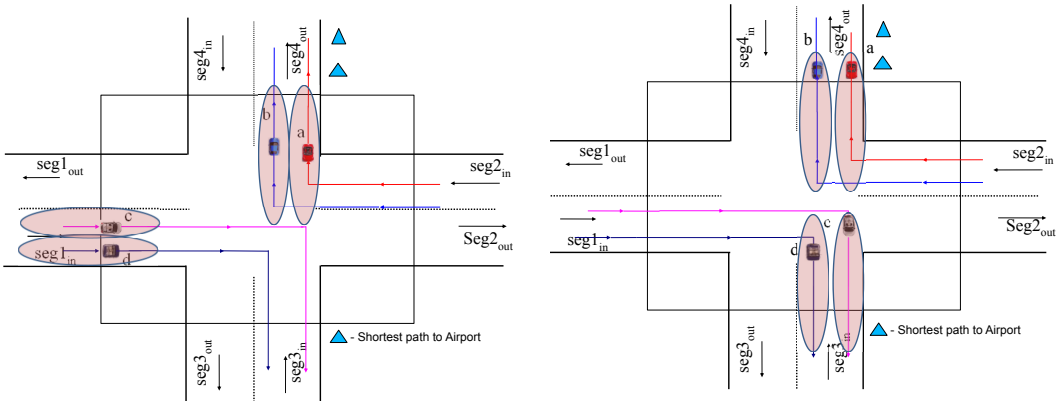
$$H_{pair}(i,j) = -(p_{i'\to i}(t) log p_{i'\to i}(t) + p_{i'\to j}(t) log p_{i'\to j}(t))$$

As it is intuitive, for an exiting user, i', the number of users in the effective anonymity set (i.e., those members that have high pairwise entropy with each other and with i') is directly proportional to the temporal tolerance, d_{tmax}, ie., the greater the temporal tolerance value, d_{tmax}, the more the number of users that could possibly resemble i during the exit of i' with a high pairwise Entropy. Thus by varying the temporal tolerance, d_{tmax} the temporal delay tolerant-mix-zones can offer any desired level of anonymity to the users.

6.3.2 Spatial Delay-tolerant Mix-zones

We now present our second class of delay-tolerant mix-zones namely spatial delay-tolerant mix-zones. Unlike the temporal delay-tolerant mix-zones, in the spatial delay-tolerant mix-zone approach, users' locations are instantaneously sent out using a spatial region instead of the exact point location. Here, the spatial region masks the exact time of traversal of the user inside the mix-zone ensuring the possibility that the user could be located at any point within the spatial region. This ensures that the adversary can not infer the exact time of traversal of the user. The spatial region is constructed by first identifying the temporal window size, d_{tmax} based on the arrival rate of the users in the mix-zone and by translating the user's current location into a spatial region based on the temporal window size, d_{tmax}. A spatial region corresponding to a temporal window size d_{tmax} includes all road segments that can be reached within d_{tmax} units of time (i.e., the corresponding d_l units of length) from the center of the region when travelled at the mean speed of the road segments. The delay-proportional spatial cloaking algorithm described in Algorithm 2 computes the spatial region in such a way that the distance from the center of the spatial region and the location of the mobile user exactly corresponds to the spatial

distance, d_l proportional to the temporal delay, d_t of the user in the temporal delay-tolerant approach. Inside the delay-tolerant mix-zone, the spatial regions of the users are randomly changed by introducing a spatial shift.



(a) Before spatial location-mixing (b) After spatial location-mixing

Figure 57: Illustration of Delay-tolerant mix-zones

We illustrate the principle of spatial delay-tolerant mix-zone through an example in Figure 57(a) and Figure 57(b). Figure 57(a) shows the entry of the spatial regions of users a, b, c and d into the mix-zone and Figure 57(b) illustrates the spatial location mixing process. We find that the location mixing process changes the spatial regions of the users, a, b, c and d in such a way that the distance of the users from the center of their spatial region is randomly shifted (Notice changed distance in the spatial regions in Figure 57(b)). Therefore, after the spatial location mixing process, the spatial regions of users a, b, c and d all have similar probability to exit at a given time. Note that without this spatial mixing, the attacker would still infer that the regions of a and b entered well ahead of c and d and hence a and b will exit before c and d.

Here we assume that each user, i's spatial region enter the mix-zone at time, $t_{in}(region(i))$ and exits at time $t_{out}(region(i))$ with a new pseudonym, i'. For each user, i, the set of other users whose spatial regions entered the mix-zone during the time window defined by $|t_{in}(region(i)) - \tau|$ to $|t_{in}(region(i)) + \tau|$ forms the anonymity

Algorithm 2 Delay-proportional Spatial Cloaking

1: d_{tmax}: continuous query temporal window
2: v: vertex v corresponds to the road junction v
3: $pathtime_v$: mean travel time to vertex v from starting mix-zone, if v is the mix-zone junction, then $pathtime_v = 0$
4: *region*: is a global list of road segments representing the cloaking region. It is empty at beginning and represents the cloaking region when the algorithm terminates
5: **procedure** FINDCLOAKREGION(d_{tmax}, $pathtime_v$, v)
6: **for all** $segments(v,u) \in segs(v)$ **do**
7: $pathtime_u = pathtime_v + \frac{length(v,u)}{speed(v,u)}$
8: **if** $(pathtime_u < d_{tmax})$ **then**
9: **if** $(region.contains(v,u) == false)$ **then**
10: $region.add(v,u)$
11: FindCloakRegion($d_{tmax}, pathtime_u, u$)
12: **end if**
13: **end if**
14: **end for**
15: **end procedure**

set of i, denoted by A_i. The anonymity obtained in the mix-zone is dependent on the pairwise mapping probabilities, $p_{i' \to i}$ and $p_{i' \to j}$ where $j \in A_i$ and $p_{i' \to j}$ denotes the probability that the exiting user i' at time, $t_{out}(region(i'))$ is j and $p_{i' \to i}$ represents the probability that the exiting user is i.

The movement of a user's spatial region is governed by two factors in a spatial delay-tolerant mix-zone namely the randomness of the user movement in terms of its velocity inside the mix-zone and the spatial shift introduced in the spatial region after exiting the mix-zone. Let us assume that user j takes x units of time in the mix-zone and let $P_v(j,x,l)$ be the probability that user j travels with a velocity such that it takes x units of time to cross l units of distance of the mix-zone region. Let l_{mix} be the distance to cross the mix-zone and let us assume that j's spatial region is shifted by y units of length in the location mixing process. Let $P_s(j,y)$ denote the probability that the spatial shift introduced in the mix-zone is y units of length. Therefore, for user j to exit at time, $t_{out}(region(i'))$, it should travel a distance of $l_{mix} + y$ in the mix-zone instead of the usual distance l_{mix}. Therefore, we have

$$P(j,t) = \int_0^{\infty} P_v(j, x, l_{mix} + y) \times P_s(j, y) dx$$

where $x = t - t_{in}(region(j))$. Here, the spatial location mixing process assigns a new spatial region to each user ensuring a uniform distribution of spatial shift values, $P_s(j, y)$. i.e., inside the mix-zone, every user's distance from the center of its spatial region is shifted by a random length, y.

Similar to $P(j,t)$, we can obtain $P(i,t)$ and hence $p_{i' \to j}(t)$, the probability of i' being j when i' exits at time t can be obtained from the conditional probability, $p_{i' \to j}(t) = P((j,t)/(i',t))$. Similarly, as discussed in section 6.3.1, the probability of i' being i, $p_{i' \to i}(t)$ can be computed and thus the pair-wise entropy, $H_{pair}(i,j,t)$ between users i and j when i exits as i' can be obtained.

We note that the spatial delay-tolerant mix-zone approach does not incur temporal delays, however they lead to higher query processing cost that is directly proportional to the size of the spatial regions. In the next subsection, we discuss spatio-temporal delay-tolerant mix-zones that yield suitable tradeoffs between the incurred delay and the cost of query processing.

6.3.3 Spatio-temporal delay-tolerant Mix-zones

In the spatio-temporal delay-tolerant mix-zone approach, user locations are perturbed using both a temporal delay as well as a spatial region instead of the exact point location and the mix-zone introduces both random temporal and spatial shifts to the spatio-temporally perturbed user locations. Therefore, to an adversary observing an user, the user could have been located at any point in the spatial region at any instance of time during the temporal time window.

The anonymity strength of this model is analyzed as follows. We assume that each user uses a temporal time window of d_{tmax} and a spatial cloaking region of length, d_{lmax}. Based on the delayed exposure of the spatial regions, the attacker estimates

the current temporally perturbed position of the spatial region of each user's location. Here we assume that each user, i's spatial region, $region(i)$, enters the mix-zone at a temporally cloaked time, $tcloak_{in}(region(i))$ and exits at a temporally cloaked time, $tcloak_{out}(region(i))$ with a new pseudonym, i'. For user i, the set of all other users whose spatial regions entered the mix-zone during the mix-zone time window defined by $|tcloak_{in}(region(i)) - \tau - d_t|$ to $|tcloak_{in}(region(i)) + \tau + d_t|$, forms the anonymity set of i, namely A_i.

In a spatio-temporal delay-tolerant mix-zone, the movement of a user's spatio-temporally perturbed location is governed by three factors: (i) the randomness of user's movement inside the mix-zone determined by velocity, (ii) the random temporal shift introduced inside the mix-zone and (iii) the random spatial shift introduced inside the mix-zone. For all $j \in A_i$, let us assume that user j's spatial region, $region(j)$ takes x units of time to cross the mix-zone and let $P_v(j, x, l)$ be the probability that user j travels with a velocity such that it takes x units of time to cross l units of distance of the mix-zone region and let j's spatial region be shifted by y units of length and its temporal delay be shifted by z units of time. Let $P_s(j, y)$ denote the probability that the spatial shift introduced in the mix-zone is x units of length and $P_t(j, z)$ be the probability that the temporal delay of user j is shifted by z seconds after exiting the mix-zone. Therefore, for user j's spatial region, $region(j)$ to exit at temporally cloaked time, $t_{out}(region(i'))$, it should travel a distance of $l_{mix} + y$ in the mix-zone instead of the usual l_{mix} where l_{mix} is the length of the mix-zone region. Therefore,

$$P(j,t) = \int_0^\infty P_v(j, x, l_{mix} + y) \times P_s(j, y) \times P_t(j, t - tcloack_{in}(j) + x) dx dy$$

Here we note that the spatio-temporal delay-tolerant mix-zone introduces both

random temporal and spatial shifts inside and thus it ensures a uniform random distribution of $P_s(j, y)$ and $P_t(j, t - tcloack_{in}(j) + x)$ in the above equation.

Thus, the pair-wise entropy between users i and j when i exits as i' can be obtained after knowing the probabilities $p_{i' \to i}(t)$ and $p_{i' \to j}(t)$ which are deduced from $P(i, t)$ and $P(j, t)$ similar to the analysis on temporal delay-tolerant mix-zones in section 6.3.1

6.4 Experimental Evaluation

We divide the experimental evaluation of our techniques into three components: (i) the effectiveness of the proposed techniques under the CQ-attack model, (ii) performance in terms of query processing cost, incurred temporal delays and success rate of anonymization and (iii) evaluation of the spatio-temporal tradeoffs between incurred temporal delays and query processing cost. Before reporting our experimental results, we first describe the experimental setup, including the road-network mobile object simulator used in the experiments.

6.4.1 Experimental setup

We use the GT Mobile simulator [108] to generate a trace of cars moving on a real-world road network, obtained from maps available at the National Mapping Division of the USGS [20]. The simulator extracts the road network based on three types of roads – *expressway*, *arterial* and *collector* roads. Our experimentation uses maps from three geographic regions namely that of Chamblee and Northwest Atlanta regions of Georgia and San Jose West region of California to generate traces for a two hour duration. We generate a set of 10,000 cars on the road network that are randomly placed on the road network according to a uniform distribution. Cars generate random trips with source and destination chosen randomly and shortest path routing is used to route the cars for the random trips. The speed of the cars are distributed based on the road class categories as shown in Table 19.

Road type	Expressway	Arterial	Collector
Mean speed(mph)	60	50	25
Std. dev.(mph)	20	15	10
Speed Distribution	Gaussian	Gaussian	Gaussian

Table 19: Motion Parameters

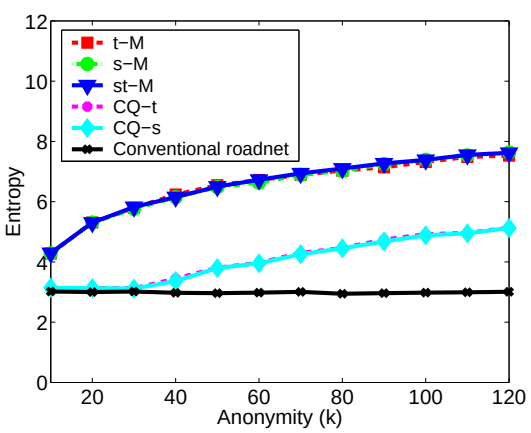
Parameter	Value
Map	Northwest Atlanta region
Mobility Model	Random Roadnet Router
Total number of vehicles	10000
Number of Road junctions	6831
Number of Road segments	9187

Table 20: Simulation Parameters and Setting

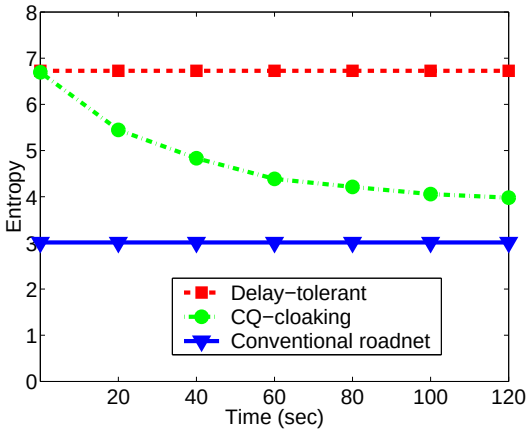
6.4.2 Experimental results

Our experimental evaluation consists of three parts. First, we evaluate the effectiveness of the proposed techniques in terms of their anonymization effectiveness. We compare the delay-tolerant mix-zone techniques with CQ-cloaking techniques and conventional road network mix-zones in terms of the obtained entropy that captures the amount of information required to break the anonymity and then compare the various delay-tolerant mix-zones in terms of the average temporal delays incurred and the cost of query processing in terms of query processing time. Next, we evaluate the spatio-temporal tradeoffs of delay-tolerant mix-zones that helps understand the best tradeoffs in terms of the incurred temporal delay and the query processing time. Our final set of experiments evaluate the effectiveness of the delay-tolerant mix-zones in terms of the average temporal delays, query execution time and success rate in providing the desired value of k. Our default setting uses the map of Northwest Atlanta that has 6831 road junctions and 9187 road segments as shown in table 20. Among the 6831 junctions in the road network, 1025 (15 %) road junctions are chosen as the candidate mix-zones and the experimental results are averaged among these mix-zones. By default, each delay-tolerant mix-zone is constructed over a non-rectangular road

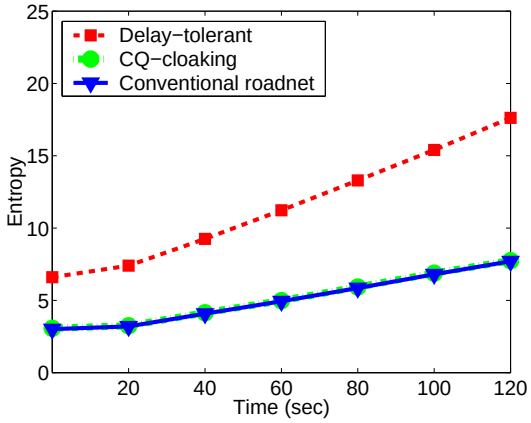
network mix-zone with pairwise Entropy lowerbound after timing attack, α taken as 0.9 so that the effective anonymity set of the mix-zone comprises only of users who have pairwise Entropy greater than 0.9 with each other. Similarly for comparison, each conventional mix-zone is also constructed using the non-rectangular geometry with $\alpha = 0.9$. For both delay-tolerant mix-zones and conventional mix-zones, the pairwise Entropy lowerbound β after transition attack is taken as 0.9. We assume that all continuous queries are unique and by default 10% of users in the system run continuous queries.



(a) Entropy after CQ-attack



(b) Entropy Timeline - CQs



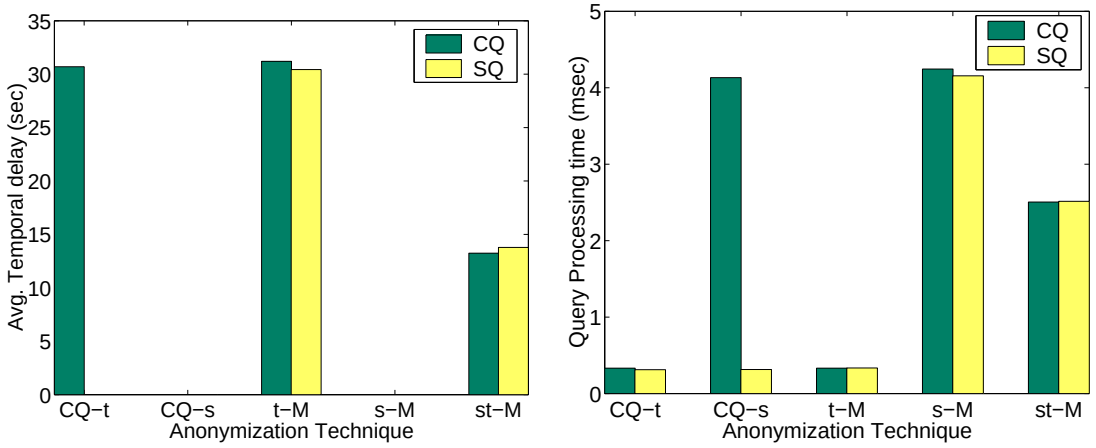
(c) Entropy Timeline - SQs

Figure 58: Comparison with Conventional Mix-zones

6.4.2.1 Comparison with Conventional Mix-zones and CQ-cloaking

This set of experiments compares the delay-tolerant mix-zone approaches with the conventional mix-zones and CQ-cloaking techniques in terms of their anonymity measured by Entropy. Here, the delay-tolerant mix-zones are constructed over a conventional road network mix-zone whose size is chosen to offer an anonymity of 4. In Figure 58(a), we compare the average entropy of the temporal, spatial and spatio-temporal delay-tolerant mix-zone approaches (t-M, s-M, and st-M) with the conventional mix-zone approach and the temporal and spatial CQ-cloaking approaches (CQ-t and CQ-s) for various values of required anonymity, k. Here, the temporal window and spatial region size are chosen based on the arrival rate of the users in the mix-zones to ensure the required number of users, k with a high probability, $p = 0.9$. For the spatio-temporal delay-tolerant mix-zones, the spatial region size is fixed as 800 m and the temporal window is varied according to the required value of k. We find that the average entropy of the conventional mix-zone approach is significantly lower than that of the delay-tolerant mix-zones as they can not adapt to higher levels of anonymity but the delay-tolerant mix-zones always provide the required anonymity level for all values of k as shown by the high Entropy. Here, we also note that the CQ-cloaking approaches (CQ-t and CQ-s) have low level of Entropy due to the effect of CQ-timing and CQ-transition attacks. In Figure 58(b) and Figure 58(c), we plot the timeline of the Entropy obtained by continuous queries (CQ) and snapshot queries (SQ) respectively. Here, we use the spatio-temporal mix-zone as the candidate delay-tolerant mix-zone and temporal CQ-cloaking as the candidate CQ-cloaking technique. We find that with conventional mix-zones, the continuous queries obtain low initial anonymity and it stays constant throughout the timeline. With the CQ-cloaking approach, the queries obtain higher anonymity in the beginning but their anonymity is gradually reduced due to the impact of CQ-timing and CQ-transition attacks. However, the delay-tolerant mix-zones offer very high anonymity to meet

the privacy requirements of the continuous queries under the CQ-attack model. For snapshot queries, we find that the techniques have a different trend as shown in Figure 58(c). The conventional mix-zone model shows an increasing Entropy timeline where users gain more anonymity at the intermediate mix-zones as CQ-attack has no impact on snapshot queries. We also find that the delay-tolerant mix-zone offers greater anonymity to snapshot queries with a much steeper Entropy timeline but the CQ-cloaking technique offers only similar anonymity as the conventional mix-zone.



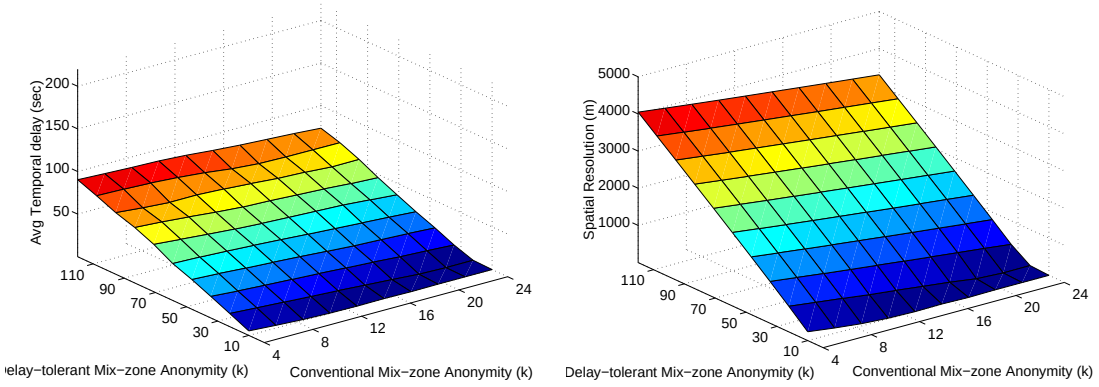
(a) Average temporal delay (b) Query Cost

Figure 59: Performance of Continuous and SnapShot queries

6.4.2.2 Performance of Continuous and Snapshot queries

Next, we study the performance impact of the proposed approaches for continuous and snapshot queries individually. We measure the average temporal delay incurred and the average query execution time of the techniques in figure 59(a) and 59(b). Here, all queries are anonymized corresponding to a $k = 50$. The spatio-temporal delay-tolerant mix-zone uses its default spatial region size of 800 m. The query execution time represents the average time to process a snapshot of a k-NN query for a k-nearest neighbor value of ($k_q = 7$) over 14000 uniformly distributed objects on the road network using the road network based anonymous query processor described in [135]. We find that with the CQ-temporal cloaking (CQ-t), only the continuous queries

incur delay before getting processed and in CQ-spatial cloaking (CQ-s), neither of the queries incur any delay. With the spatial cloaking approach, we obtain the results of the query for all possible locations within the cloaking region, however the continuous queries in the CQ-spatial cloaking approach result in higher query execution time. In temporal delay-tolerant mix-zones (t-M), both continuous and snapshot queries incur temporal delays but have low query execution time. Conversely, the spatial delay-tolerant mix-zones (s-M) do not incur any delays for the queries but have increased query execution time for both snapshot and continuous queries. The spatio-temporal delay-tolerant mix-zones technique (st-M) finds a tradeoff between these approaches and has more than 55% lower average temporal delay compared to the temporal cloaking case as well as a 40% lower query execution time compared to the spatial delay-tolerant mix-zones.

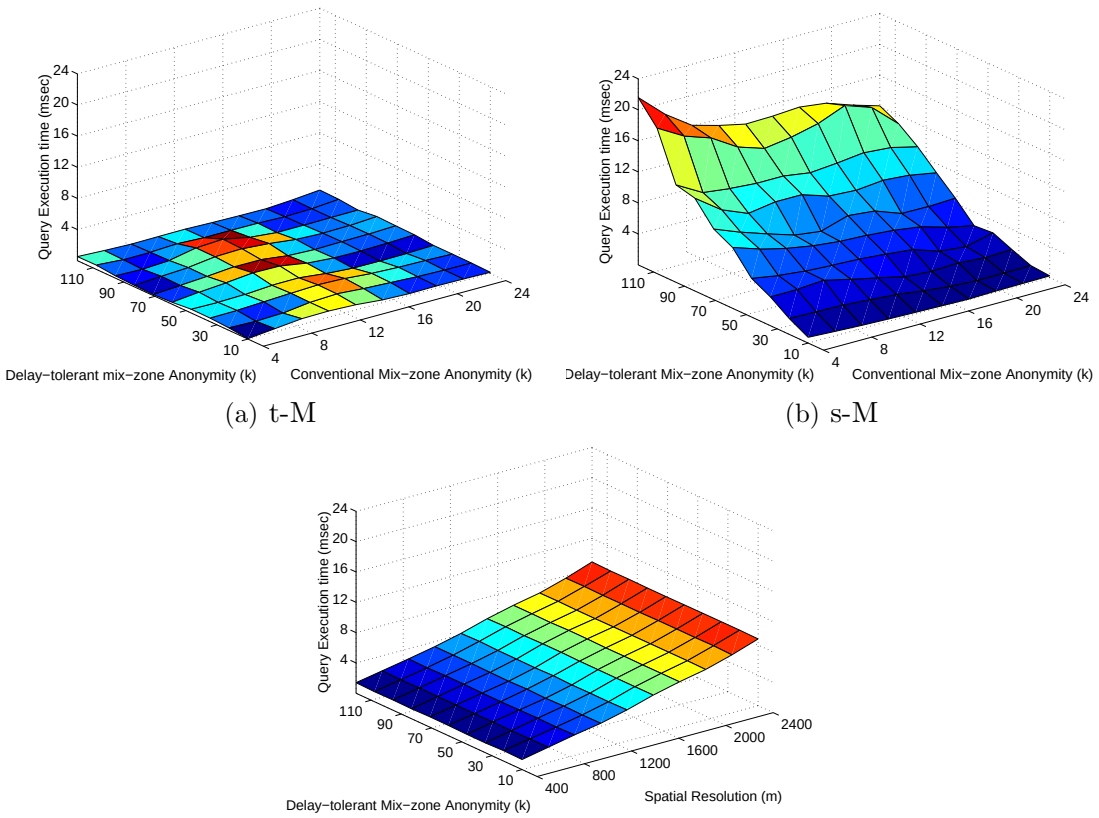


(a) t-M

(b) s-M

(c) st-M

Figure 60: Performance of delay-tolerant mix-zones



(a) t-M

(b) s-M

(c) st-M

Figure 61: Comparison of Query Execution time

6.4.2.3 Evaluating Spatio-temporal tradeoff

Our next set of experiments compares the performance of the delay-tolerant mix-zones in terms of the average temporal delays incurred and the query execution time under various values of required anonymity, k. Figure 60(a) shows the average temporal delay required to anonymize users for various anonymity levels. The X-axis shows the anonymity offered by the conventional mix-zone and the Y-axis shows the anonymity level offered by the delay-tolerant mix-zones constructed over the conventional mix-zones and the Z-axis represents the average temporal delay, d_t used by the delay-tolerant mix-zones. We find that the temporal delay increases with increase in the anonymity level of the delay-tolerant mix-zone. Also, we find that there is only a small decrease in temporal delay with increase in mix-zone anonymity as the temporal

window has a significant impact on the obtained anonymity.

Similarly, we study the performance of spatial delay-tolerant mix-zones in Figure 60(b). The X-axis represents the anonymity offered by the conventional road network mix-zone and the Y-axis represents the anonymity of the delay-tolerant mix-zones. We plot the required spatial resolution along Z-axis. We observe a similar trend with spatial resolution values as with the temporal delay in Figure 60(a). We notice that higher query anonymity levels require larger spatial cloaking regions for location perturbation. In the context of delay-tolerant mix-zones, we measure the data quality in terms of the size of the spatial region (spatial resolution) instead of the number of objects as those in the location k-anonymized cloaking [45]. This is because the spatial perturbation in a delay-tolerant mix-zone is focused on changing the pseudo-identity with a higher anonymity rather than obtaining a perturbed location with a k-anonymized cloaking region. However, we refer the interested readers to [45] for additional object distribution based metrics for measuring data quality of the location perturbation process.

Next, we study the spatio-temporal tradeoff between the spatial resolution that determines query processing cost and the temporal delay incurred before processing the query requests. In Figure 60(c), the spatial resolution value is varied along the X-axis and the Y-axis represents the delay-tolerant mix-zone anonymity level. Here, the conventional mix-zone anonymity is set as 12. We find that the average temporal delay along Z-axis is much smaller with the effect of spatio-temporal perturbation compared to the temporal and spatial delay-tolerant mix-zones. Higher spatial resolution greatly reduces the temporal window size required to provide the required anonymity. However, the right trade-off between the temporal window size and the spatial resolution can be made based on the acceptable delay in processing the user queries and the desired cost of query processing.

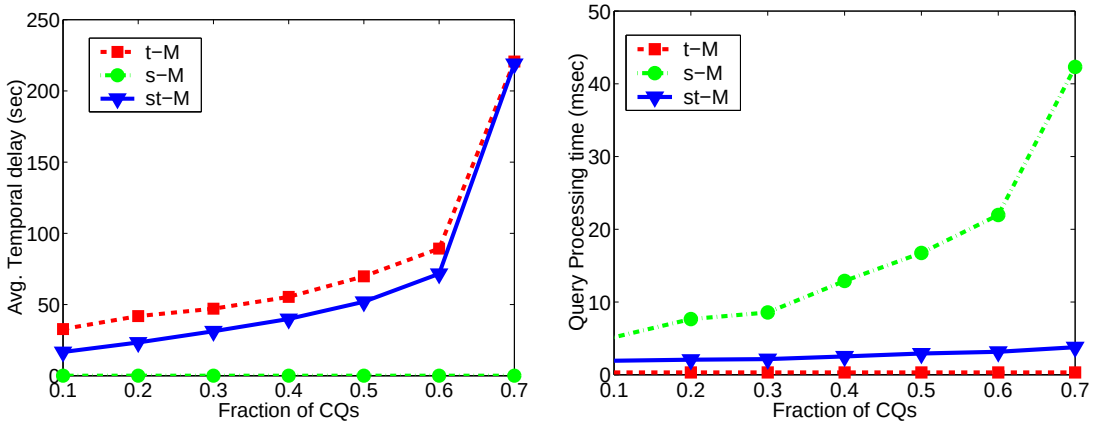
Our next set of experiments compares the delay-tolerant mix-zones in terms of

their average query execution time. In Figure 61, the average query execution time to process a snapshot of a k-NN query with ($k_q = 7$) over 14000 uniformly distributed objects on the road network is shown. In Figure 61(a), observe that the temporal delay-tolerant mix-zones incur very low query execution time (less than 1 msec) as these queries use the point location of the mobile client with a temporal delay. Whereas, the query execution time of spatial delay-tolerant mix-zones in Figure 61(b) increases with increase in the required anonymity as larger anonymity requires larger spatial regions for processing. The query execution time of spatio-temporal delay-tolerant mix-zone approach in 61(c) shows the reduction in the query processing cost at the expense of the incurred temporal delay in processing the queries (Figure 60(c)).

6.4.2.4 Impact of fraction of Continuous Query users

The anonymity of the delay-tolerant mix-zones also depends on the fraction of total users who run continuous queries. For instance, if a number of continuous queries exist in the system, each uniquely identifying the querying user, the anonymization process might result either in long temporal delays or large spatial regions leading to higher query processing cost. The continuous query fraction denotes the fraction of the total users who currently execute a continuous query. Here, each query is anonymized with an anonymity, $k = 50$. In Figure 62(a), we measure the average temporal delay incurred by the approaches for varying values of fraction of continuous queries. We find that the average temporal delay of both (t-M) and (st-M) approaches increases steadily with increase in the proportion of continuous queries till a point, (0.6 in the figure) and then increases steeply indicating that it could be expensive to anonymize continuous queries if more than 60% of the users execute continuous queries. A similar trend is exhibited in Figure 62(b) for query processing cost suggesting that the anonymization cost could be higher when the fraction of continuous

queries approaches 0.7. As a large number of mobile users are passive in general (i.e., execute no queries while traveling), we believe that a 60% continuous query fraction of the entire user population is expected to be rarely crossed in practice.



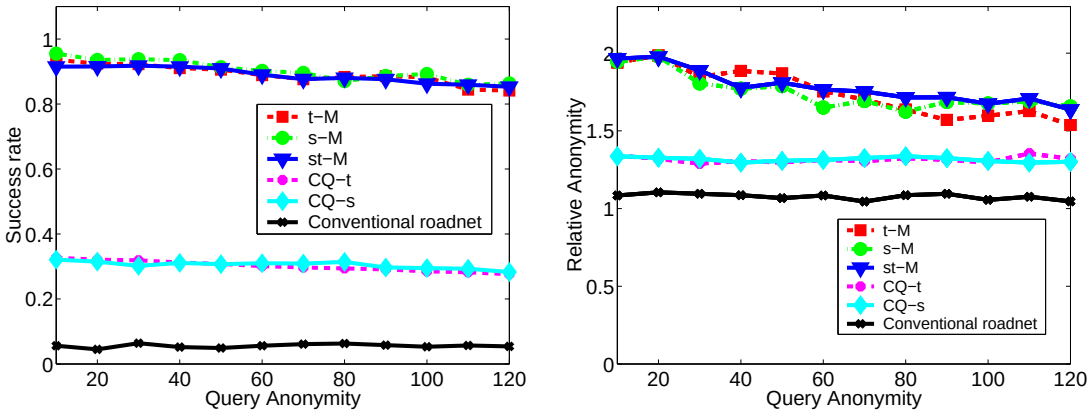
(a) Average Temporal delay (b) Query processing time

Figure 62: Fraction of Continuous Query users

6.4.2.5 Success Rate and Relative-k

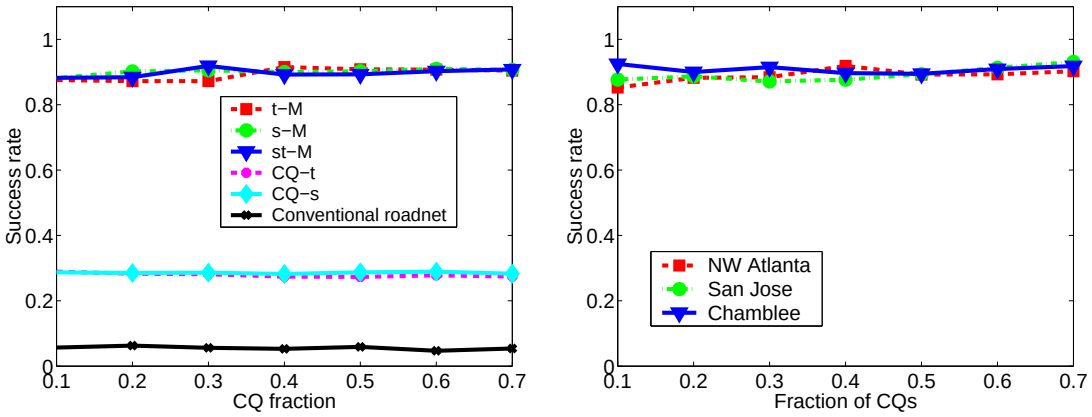
Our final set of experiments evaluates the performance of the delay-tolerant mix-zones in terms of their success rate in providing the desired level of anonymity. The success rate represents the fraction of the cases where the proposed framework is able to provide an anonymity equal or greater than the requested value, k. In Figure 63(a), the query anonymity level is varied along the X-axis and the Y-axis represents the obtained success rate. Based on the arrival rate of the users in the mix-zone, the expected success rate is chosen as 0.9 so that the delay-tolerant mix-zones provide an anonymity of k or higher in more than 90% of the cases. We find that all the delay-tolerant mix-zone techniques obtain a success rate close to the expected success rate of 0.9, however the success rate of the CQ-cloaking approach is much lower (less than 0.3) and the conventional mix-zone approach has a even lower success rate of less than 0.06. Similarly, we study relative-k which is defined as the ratio of the anonymity obtained by the queries to the query anonymity requested. In Figure

63(b), we find that the relative anonymity level of delay-tolerant mix-zones ranges from 1.5 to 2.0 showing that the queries on an average obtain an anonymity which is 1.5 to 2.0 times the requested value. The successful cases of CQ-cloaking and conventional mix-zone approaches have a lower relative anonymity as the mix-zones have lower success rate and provides lower value of k in general. In order to evaluate



(a) Success Rate (b) Relative k

Figure 63: Success rate and Relative k



(a) Varying CQ fraction (b) Varying Maps

Figure 64: Success rate

the success rate under different fractions of continuous queries in the system, we study the approaches by varying the fraction of users executing continuous queries. Here, each query is anonymized with an anonymity of 50. Figure 64(a) shows that the obtained success rate is close to the expected success rate for the delay-tolerant

mix-zones across different fractions of CQs in the system. However, the CQ-cloaking and conventional mix-zone techniques have much lower success rate. Similarly, the success rate of the techniques is compared across different scales of geographic maps described in section 6.4.1. We compare the success rate of spatio- temporal delay-tolerant mix-zones in Figure 64(b) that shows that the technique performs well across different geographic maps.

6.5 Related work

Location privacy has been studied over the past decade along two orthogonal dimensions: spatial cloaking through location k-anonymity represented by [64, 98, 33, 135, 45, 29] and mix-zone based privacy protection and its variations represented by [35, 60, 61, 39, 104, 96]. However, these approaches are suitable only for snapshot queries and are inadequate and ineffective for protecting location privacy of mobile users with continuous query services.

In recent years, there had been research efforts that dealt with location privacy risks of continuous queries. [36] describes various attacks in Location-based systems, including the continuous query attacks and the challenges of supporting continuous query services. [54] proposes spatial cloaking using the memorization property for continuous queries. This is further used in [107] for clustering queries with similar mobility patterns. However, this type of techniques may lead to large cloaking boxes resulting in higher query processing cost as users may not always move together. [58] identifies that location cloaking algorithms with only k-anonymity and l-diversity guarantee are not effective for continuous LBS and therefore propose query m-invariance as a necessary criterion when dealing with continuous location queries. However, m-invariance based approach is ineffective when the mobile users ask uniquely different CQ services as they move on the road. An alternative thread

of research is represented by the *Personal information retrieval* techniques as an alternate to location cloaking for anonymous query processing [66]. PIR techniques guarantee privacy of mobile users regardless of which types of queries (continuous or snapshot) they ask. However PIR based solutions are known to be expensive in both computation and storage overheads, even with the recent new techniques such as hardware-assisted PIR techniques [138], developed to improve the scalability and efficiency of the PIR approach. Another general issue with PIR based solutions is its limitation in terms of what kinds of queries can be protected under PIR [136].

The concept of mix-zones was first presented in the context of location privacy in [35]. The idea of building mix-zones at road intersections is proposed in [60] and [39]. In [61], a formulation for optimal placement of mix-zones in a road map is discussed. Almost all existing mix-zone techniques follow a straight forward approach of using a rectangular or circular shaped zone and their construction methodologies do not take into account the effect of timing and transition attacks in the construction process. The MobiMix framework presented in [104] is the first road-network aware attack-resilient mix-zone that guarantees an expected value of anonymity by leveraging the characteristics of both the underlying road network and motion behaviors of users traveling on spatially constrained road networks. However, all existing road network mix-zone approaches, to the best of our knowledge, fail to protect mobile users from continuous query attacks.

Inspired by the mix-zone concept, the *Cachecloak* algorithm [96] employs an alternate technique for path-mixing by using cache prefetching to hide the exact location of mobile user by requesting the location based data along an entire predicted path. Although these techniques are effective when all users obtain the same service, they are vulnerable to continuous query correlation attacks when the mobile users obtain uniquely different CQ services. Recently, content caching[27] has been proposed as an alternate solution to location privacy. However, caching large amounts of information

on tiny mobile devices may not be effective. In addition, they may limit the usability of the services by restricting mobile clients to ask only services that are cached before-hand.

In this chapter, we introduce delay-tolerant road-network mix-zones as an effective countermeasure against CQ attacks. We show that by performing a combination of both location mixing and identity mixing in the mix-zones, the delay-tolerant mix-zones offer greater level of anonymity that is sufficient to meet the anonymity levels of continuous queries under the CQ-attack model while maintaining acceptable quality of continuous query services. Though both location perturbation based techniques and mix-zone anonymization are well researched topics by themselves, to the best of our knowledge, this is the first work to systematically study the benefits of combining location perturbation based techniques with mix-zone based location anonymization schemes to tackle sophisticated attacks such as the continuous query attacks. In general, we believe that an effective combination of location perturbation with mix-zone based techniques has the potential to provide the strongest defense against the sophisticated CQ attacks.

6.6 summary

We presented a delay-tolerant mix-zone framework for protecting location privacy of mobile users against continuous query correlation attacks. First, we described and formally analyzed the mix-zone anonymization problem under the CQ-attack model and showed that spatial cloaking or temporal cloaking over road network mix-zones is ineffective and susceptible to CQ-timing and CQ -transition attacks. We introduced three types of delay-tolerant road network mix-zones that are free from CQ-timing and CQ-transition attacks and in contrast to conventional mix-zones, perform a combination of both location mixing and identity mixing of spatially and temporally perturbed user locations to achieve stronger anonymity for the continuous queries under

the CQ-attack model. Extensive experiments using traces generated by GTMobiSim on different scales of geographic maps showed that the delay-tolerant mix-zones are effective under the CQ attack model and offer the required level of anonymity to the continuous queries.

CHAPTER VII

CONCLUSIONS AND FUTURE WORK

Cloud computing and its pay-as-you-go cost structure have enabled infrastructure providers, platform providers and application service providers to offer computing services on demand and pay per use just like how we use utility today. This growing trend in cloud computing with the exponential data growth witnessed in recent years calls for resource management techniques that are highly cost-effective, consumer-driven and privacy conscious. This dissertation addresses these challenges with systematic approaches and techniques with the focus on parallel processing of large scale data using Map-Reduce and scaling and delivering privacy-aware services in the Cloud. In this chapter, we first summarize the contributions made by this thesis and then discuss open problems and interesting directions for future work.

7.1 Summary

In summary, this dissertation makes the following contributions. First, we argued that existing cloud services for MapReduce are highly cost-inefficient and inadequate for production workloads. We presented a new MapReduce cloud service model, Cura, for large scale data analytics in the cloud. In contrast to existing services, Cura automatically creates the best cluster configuration for the jobs using MapReduce profiling and leverages deadline-awareness which, by delaying execution of certain jobs, allows the cloud provider to optimize its global resource allocation efficiently and reduce its costs. Cura's resource management techniques include cost-aware resource provisioning, VM-aware scheduling and online virtual machine reconfiguration.

Second, we developed Purlieus, a locality-aware resource allocation system for

MapReduce in a cloud. We discussed the Purlieus system architecture for MapReduce cloud service and described how existing data and virtual machine placement techniques lead to longer job execution times and large amounts of network traffic in the data center. We identified *data locality* as the key principle which if exploited can alleviate these problems and developed a unique coupled data and VM placement technique that achieves high data locality. Uniquely, Purlieus's proposed placement techniques optimize for data locality during both map and reduce phases of the job by considering VM placement, MapReduce job characteristics and load on the physical cloud infrastructure at the time of data placement.

Third, we developed a suit of techniques for enabling privacy-preserving access to data and compute services in the Cloud. In this context, we developed VNCache: an efficient solution for MapReduce analysis of cloud-archived log data without requiring an apriori data transfer and loading into the local Hadoop cluster. We showed that current solutions are highly inefficient as they require large encrypted datasets to be first transferred to the secure enterprise site, decrypted, and loaded into a local Hadoop cluster before they can be processed. We presented our filesystem layer called VNcache that dynamically integrates data stored at multiple public storage clouds into a virtual namespace at the enterprise site. VNCache provides a seamless data streaming and decryption model and optimizes Hadoop jobs to start without requiring apriori data transfer, decryption, and loading.

The next contribution in this context extended our cloud service framework to support privacy-preserving ubiquitous data access and query processing. In a cloud, mobile users request location dependent data services having their own location privacy constraints. We have presented MobiMix, a framework for building attack resilient road network mix-zones that enables handling cloud service requests with respect to system-defined location privacy requirements. The construction techniques proposed in MobiMix are efficient and are more attack-resilient than the existing mix-zone

approaches. We then extended the MobiMix framework to develop a delay-tolerant mix-zone framework for protecting location privacy of mobile cloud users against continuous query correlation attacks (CQ-attacks). We introduced three types of delay-tolerant road network mix-zones (temporal, spatial and spatio-temporal) that achieve stronger anonymity for the continuous query services.

7.2 Open issues and Future directions

Large-scale data processing in the Cloud is an interesting area of research both from performance and cost-optimization perspective as well as from the perspective of system privacy. We are interested in pushing this research in several interesting directions. We highlight some of them below.

7.2.1 Performance and Cost-optimization

As we argued in Chapter 2, most of today's Cloud services for Big Data applications function merely as a Virtual-Machine-as-a-service model leading to poor resource utilization and higher cost for customers. In such models, we notice that there is a significant wastage of I/O and CPU resources due to virtualization overheads. Given the exponential data growth witnessed in the recent years, customers often look for cheaper services given that many Big Data analytic jobs do not require strong performance guarantees. One interesting direction of future research is to devise Cloud solutions that leverage such opportunities and optimize job performance in non-virtualized cloud environments to minimize cost for customers who are willing to trade off strict performance guarantees. As an extension of this research direction, we are interested in exploring to build service models for fine-grained resource sharing using a layered virtualization approach. We note that conventional server virtualization virtualize every aspect of the physical resources leading to higher virtualization overheads and poor cost-effectiveness. We are interested in building fine-grained virtualization techniques that virtualize physical resources at different granularities,

virtualizing precisely what customer jobs need while leaving the rest of the physical resources to perform without the virtualization overhead and cost. We believe that such fine-grained virtualization would make future cloud services more utility-aware and cost-effective.

7.2.2 Locality-optimizations

Our research on locality-aware MapReduce processing continues in two directions. First, for placement techniques, we would like to capture relationships between datasets, e.g. if two datasets are accessed together (MapReduce job doing a *join* of two datasets), their data placement can be more intelligent while placing their blocks in relation to each other. Second, we plan to develop online techniques to handle dynamic scenarios like changing job characteristics on a dataset. While core principles developed in this work will continue to apply, such scenarios may use other virtualization technologies like live data and VM migration.

7.2.3 Dealing with Geographically Distributed Data

We also plan to extend our cost-optimization and locality-aware resource management frameworks to deal with the upcoming challenges in effectively processing globally distributed data where massive amounts of data get generated from disparate sources at possibly different geographical locations of the world. While conventional MapReduce framework and MapReduce algorithms assume that the data is present in one physical location, scalable processing of geographically distributed big data brings an entirely new challenge. Such scenarios require both framework level optimizations to MapReduce as well as middleware-level solutions to support analytics over geographically distributed sites.

7.2.4 Cloud System Privacy

In the cloud system privacy context, an important challenge that cloud computing poses is a way for customers to delegate processing of the data to the cloud without giving away access to it. While completely homomorphic data encryption techniques enable query processing on the encrypted data so that the Cloud provider does not have direct access to the plain text, such techniques consume a lot of time and computational resources making them difficult to scale. One direction of our future research is on designing private query processing techniques that protect data privacy through anonymity while retaining high utility of the data, enabling the process to scale higher than encryption-based query processing techniques. Such techniques would primarily involve private query processing using both homomorphic encryption as well as data anonymization. When computation on an anonymized dataset is delegated to the Cloud, an attacker observing the computed output has the uncertainty in associating the actual result of the computation as the computation in this case, will return several possible values, one out of which represents the actual result. Under such a scheme, the Cloud provider can process the final outputs using homomorphically encrypted secrets so that the encrypted final results are sent back to the clients while the Cloud itself has no access to the actual result. We believe that ensuring execution privacy through anonymity based guarantees would provide a more realistic and practical solution to leveraging public clouds for sensitive data analytics on private data.

7.2.5 Cloud Consumer Privacy

At the cloud consumer tier, we plan to carry out further research on our mix-zone construction frameworks presented in Chapters 5 and 6 along multiple directions. First we would like to develop attack-resilient mix-zone anonymization schemes considering more sophisticated attack models based on background knowledge about the users'

trajectory patterns and travel behavior. In this context, we would like to consider more powerful attack models based on background knowledge from location-based social networking information such as user location check-ins. While many principles developed in the MobiMix framework will apply in such scenarios, we will require more dynamic and adaptive mix-zone construction schemes that effectively consider the additional information leaked through such background knowledge.

Further, we plan to extend our consumer-tier privacy protection schemes to deal with online social networks. With the extensive popularity of social networking applications and the enormous amounts of social network data being generated, supporting privacy preserving analytics and queries on real-time social network data needs support for anonymizing dynamically evolving social graphs. For instance, how can one understand and predict the real-time propagation of information based on social influence of the users without compromising user privacy? Another example will be an anonymous voting system in a social network which exposes the graph properties corresponding to the trust among the users without compromising individual user privacy. We intend to devise anonymization algorithms that provide utility on these dynamically evolving datasets while being attack-resilient on exposing various snapshots of the streaming graphs to an application hosted in the cloud.

REFERENCES

[1] *http://en.wikipedia.org/wiki/Big-data*

[2] *http://en.wikipedia.org/wiki/Clickstream*

[3] Facebook. http://www.facebook.com.

[4] Hadoop. http://hadoop.apache.org.

[5] Yahoo! http://www.yahoo.com.

[6] Cloudera. *http://www.cloudera.com/blog/2010/08/hadoop-for-fraud-detection-and-prevention/*

[7] Google BigQuery . https://developers.google.com/bigquery/.

[8] Pig User Guide. *http://pig.apache.org/.*

[9] Amazon Elastic MapReduce. *http://aws.amazon.com/elasticmapreduce/*

[10] Amazon Elastic Compute Cloud. *http://aws.amazon.com/ec2/*

[11] Amazon Simple Storage Service. *http://aws.amazon.com/s3/*

[12] SELinux user guide. *http://selinuxproject.org/page/Main_Page.*

[13] Scheduling in hadoop. *http://www.cloudera.com/blog/tag/scheduling/.*

[14] *http://en.wikipedia.org/wiki/Loop_device*

[15] Hadoop DFS User Guide. *http://hadoop.apache.org/.*

[16] Hadoop DFS User Guide. *http://hadoop.apache.org/.*

[17] Hadoop Offline Image Viewer Guide. *http://hadoop.apache.org/docs/hdfs/current/hdfs_imageviewer.html*

[18] Proc Filesystem. *http://en.wikipedia.org/wiki/Procfs.*

[19] FUSE: Filesystem in User Space *http://fuse.sourceforge.net/.*

[20] U.S. Geological Survey. http://www.usgs.gov.

[21] USAToday. Authorities: Gps systems used to stalk woman. http://www.usatoday.com/tech/news/2002-12- 30-gps-stalker x.htm.

[22] C. Aggarwal. On k-Anonymity and the Curse of Dimensionality. In *VLDB*, 2005.

[23] M. Al-Fares, S. Radhakrishnan, B. Raghavan, N. Huang, A. Vahdat. Hedera: Dynamic Flow Scheduling for Data Center Networks. In *NSDI*, 2010.

[24] G. Ananthanarayanan, S. Kandula, A. Greenberg, I. Stoica, Y. Lu, B. Saha and E. Harris. Reining in the Outliers inMap-Reduce Clusters using Mantri. In *OSDI*, 2010.

[25] G. Ananthanarayanan, A. Ghodsi, A. Wang, D. Borthakur, S. Kandula, S. Shenker, I. Stoica PACMan: Coordinated Memory Caching for Parallel Jobs. In *NSDI*, 2012.

[26] S. Anastasiadis , K. Sevcik Parallel Application Scheduling on Networks of Workstations. In JPDC, 1997

[27] S. Amini, J. Lindqvist, J. Hong, J. Lin, E. Toch and N. Sadeh Cache': Caching Location-Enhanced Contentto Improve User Privacy In *Mobisys*, 2011.

[28] C. 'Ardagna, S. Jajodia, P. Samarati and A. Stavrou Privacy Preservation over Untrusted Mobile Networks. In Privacy in Location-Based Applications: Introduction, Research Issues and Applications, Lecture Notes of Computer Science 5599, Springer, 2009

[29] C. Ardagna, M. Cremonini, S. Vimercati, P. Samarati. An Obfuscation-Based Approach for Protecting Location Privacy. In *IEEE TDSC*.

[30] O. Arndt, B. Freisleben, T. Kielmann, F. Thilo A comparative study of online scheduling algorithms for networks of workstations In Cluster Computing, 2000.

[31] S. Aulbach, T. Grust, D. Jacobs, A. Kemper, J. Rittinger Multi-Tenant Databases for Software as a Service: Schema-Mapping Techniques In *SIGMOD*, 2008.

[32] S. Babu. Towards Automatic Optimization of MapReduce Programs. In *SOCC*, 2010.

[33] B. Bamba, L. Liu, P. Pesti, and T. Wang. Supporting Anonymous Location Queries in Mobile Environments with PrivacyGrid. In *WWW*, 2008.

[34] R. Bayardo and R. Agrawal. Data Privacy Through Optimal k-Anonymization. In *ICDE*, 2005.

[35] A. Beresford and F. Stajano. Location Privacy in Pervasive Computing. *Pervasive Computing, IEEE*, 2003.

[36] C. Bettini, S. Mascetti, X. Wang, D. Freni, and S. Jajodia. Anonymity and Historical-Anonymity in Location-Based Services In Privacy in Location-Based Applications: Introduction, Research Issues and Applications, Lecture Notes of Computer Science 5599, Springer, 2009.

[37] D. Borthakur et al. Apache Hadoop goes realtime at Facebook In *SIGMOD*, 2011.

[38] A. Z. Broder Identifying and Filtering Near-Duplicate Documents. In *COM*, 2000.

[39] L. Buttyan and T. Holczer and I. Vajda. On the effectiveness of changing pseudonyms to provide location privacy in VANETs In *ESAS 2007*

[40] M. Cardosa, P. Narang, A. Chandra, H. Pucha and A. Singh. STEAMEngine: Optimizing MapReduce provisioning in the cloud. *Dept. of CSE, Univ. of Minnesota*, 2010.

[41] K. Chard, K. Bubendorfer, P. Komisarczuk High Occupancy Resource Allocation for Grid and Cloud systems, a Study with DRIVE. In HPDC, 2010.

[42] Y. Chen, R. Griffith, J. Liu, R. H. Katz and A. D. Joseph. Understanding TCP Incast Throughput Collapse in Datacenter Networks. In *WREN*, 2009.

[43] 4 Y. Chen, S. Alspaugh, D. Borthakur and R. Katz Energy Effciency for Large-Scale MapReduce Workloads with Significant Interactive Analysis In *EUROSYS*, 2012.

[44] Y. Chen, A. Ganapathi, R. Griffith, R. Katz The Case for Evaluating MapReduce Performance Using Workload Suites In *MASCOTS*, 2011.

[45] C. Chow, M. Mokbel, J. Bao and X. Liu. Query-aware location anonymization for road networks. In *Geoinformatica*, July 2011.

[46] W. Cirne Using Moldability to Improve the Performance of Supercomputer Jobs. Ph.D. Thesis. Computer Science and Engineering, University of California San Diego, 2001

[47] W. Cirne When the Herd is Smart: The Emergent Behavious of SA. In TPDS, 2002.

[48] W. Cirne and F. Berman Adaptive Selection of Partiion Size for Supercomputer Requests. In Workshop on Job Scheduling Strategies for Parallel Processing, 2000

[49] J.R. Cuellar, J.B. Morris, D.K. Mulligan, J. Peterson and J. Polk. Geopriv requirements. *IETF Internet Draft*, 2003.

[50] C. Curino, E. P. C. Jones, R. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, N. Zeldovich Relational Cloud: A Database-as-a-Service for the Cloud In *CIDR*, 2011.

[51] Y. Chi, H. Moon, H. Hacigumus iCBS: Incremental Cost-based Scheduling under Piecewise Linear SLAs In *VLDB*, 2011.

[52] S.H. Chiang and M. K. Vernon Production job scheduling for parallel shared memory systems. In *IPDPS*, 2001

[53] S. Chiang , R. Mansharamani , M. Vernon Use of Application Characteristics and Limited Preemption for Run-To-Completion Parallel Processor Scheduling Policies. In SIGMETRICS, 1994.

[54] C. Chow and M. Mokbel Enabling Private Continuous Queries For Revealed User Locations. In *SSTD*, 2007.

[55] G. Danezis and C. Troncoso. Vida: How to use Bayesian inference to de-anonymize persistent communications Privacy Enhancing Technologies Symposium (PETS 2009)

[56] A. Dave, W. Lu, J. Jackson, R. S. Barga CloudClustering: Toward an Iterative Data Processing Pattern on the Cloud. In IPDPS Workshops, 2011

[57] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. In *OSDI*, 2004.

[58] R. Dewri, I Ray, I. Ray and D. Whitley. Query m-Invariance: Preventing Query Disclosures in Continuous Location-Based Services. In *MDM*, 2010.

[59] C. Daz, S. Seys, J. Claessens, B. Preneel. Towards Measuring Anonymity. *PETS*, 2002.

[60] J. Freudiger, M. Raya, M. Flegyhazi, P. Papadimitratos, and J.-P. Hubaux. Mix-Zones for Location Privacy in Vehicular Networks. In *WiN-ITS*, 2007.

[61] J. Freudiger, R. Shokri and J.-P. Hubaux. On the Optimal Placement of Mix Zones. In *PETS*, 2009.

[62] M. R. Garey, D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman. ISBN 0-7167-1045-5.

[63] S. L. Garfinkel An Evaluation of Amazon's Grid Computing Services: EC2, S3 and SQS . *Technical Report, TR-08-07, Harvard University, available at: ftp://ftp.deas.harvard.edu/techreports/tr-08-07.pdf*

[64] B. Gedik and L. Liu. Location Privacy in Mobile Systems: A Personalized Anonymization Model. In *ICDCS*, 2005.

[65] G. Ghinita, P. Kalnis, and S. Skiadopoulos. PRIVE: Anonymous Location-Based Queries in Distributed Mobile Systems. In *WWW*, 2007.

[66] G. Ghinita, P. Kalnis, A. Khoshgozaran, C. Shahabi, K. Tan Private Queries in Location Based Services: Anonymizers are not Necessary. In *SIGMOD*, 2008.

[67] M. Gruteser and D. Grunwald. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking. In *MobiSys*, 2003.

[68] T. Gunarathne, T. Wu, J. Qiu, G. Fox MapReduce in the Clouds for Science. In *CloudCom*, 2010.

[69] Z. Guo, G. Fox, M. Zhou Investigation of Data Locality in MapReduce. In *CCGrid*, 2012.

[70] T. Hacker, K. Mahadik Flexible Resource Allocation for Reliable Virtual Cluster Computing Systems. In *IEEE/ACM Supercomputing*, 2011

[71] B. Heintz, A. Chandra, R. Sitaraman Optimizing MapReduce for Highly Distributed Environments *University of Minnesota, Technical Report TR12-003*, 2012.

[72] U. Hengartner and P. Steenkiste. Protecting access to people location information. In *Security in Pervasive Computing*, 2003.

[73] H. Herodotou and S. Babu On Optimizing MapReduce Programs / Hadoop Jobs. In *VLDB*, 2011.

[74] H. Herodotou, F. Dong and S. Babu No One (Cluster) Size Fits All: Automatic Cluster Sizing for Data-intensive Analytics. In *SOCC*, 2011.

[75] H. Herodotou et. al Starfish: A Selftuning System for Big Data Analytics. In *CIDR*, 2011.

[76] J. Hong and J. Landay. An Architecture for Privacy-Sensitive Ubiquitous Computing. In *Mobisys*, pages 177–189, 2004.

[77] B. Igou "User Survey Analysis: Cloud-Computing Budgets Are Growing and Shifting; Traditional IT Services Providers Must Prepare or Perish". Gartner Report, 2010

[78] M. Isard, V. Prabhakaran, J. Currey, U. Wieder, K. Talwar, and A. Goldberg. Quincy: fair scheduling for distributed computing clusters. In *SOSP*, 2009.

[79] V. Jalaparti, H. Ballani, P. Costa, T. Karagiannis, A. Rowstron Bazaar: Enabling Predictable Performance in Datacenters *MSR Cambridge, UK, Technical Report MSR-TR-2012-38*.

[80] K. Kambatla, A. Pathak and H. Pucha. Towards Optimizing Hadoop Provisioning in the Cloud. In *HotCloud*, 2009.

[81] P. Karger and Y. Frankel. Security and privacy threats to its. In *World Congress on Intelligent Transport Systems*, 1995.

[82] S. Kavulya, J. Tan, R. Gandhi, P. Narasimhan An Analysis of Traces from a Production MapReduce Cluster In *CCGrid*, 2010.

[83] G. Khanna, K. Beaty, G. Kar, and A. Kochut. Application performance management in virtualized server environments. In *NOMS*, 2006.

[84] S. Ko, K. Jeon, R. Morales The HybrEx model for confidentiality and privacy in cloud computing In *HotCloud*, 2011.

[85] M. A. Kozuch, M. P. Ryan, R. Gass et al. Tashi: Location-aware Cluster Management. In *ACDC*, 2009.

[86] J. Krumm. Inference Attacks on Location Tracks In *PERVASIVE*, 2007.

[87] K. Lee, L. Liu, S. Meng, B. Palanisamy Road Network Aware Spatial Alarm Processing. *ICWS* 2012.

[88] G. Lee, N. Tolia, P. Ranganathan, R. Katz. Topology-Aware Resource Allocation for Data-intensive workloads. In *APSys*, 2010.

[89] A. Machanavajjhala, J. Gehrke, D. Kifer, and M. Venkitasubramaniam. l-Diversity: Privacy Beyond k-Anonymity. In *ICDE*, 2006.

[90] M. Mao, M. Humphrey Auto-Scaling to Minimize Cost and Meet Application Deadlines in Cloud Workflows. In SC, 2011.

[91] M. Mao, J. Li and M. Humphreyd Cloud Auto-Scaling with Deadline and Budget Constraints. In ACM Grid, 2010.

[92] P. Marshall, K. Keahey, T. Freeman Improving Utilization of Infrastructure Clouds In CCGrid, 2011.

[93] M. Maurer, I. Brandic, R. Sakellariou Enacting SLAs in Clouds Using Rules In Euro-Par, 2011

[94] S. Melnik et al. Dremel: interactive analysis of web-scale datasets In *VLDB*, 2010.

[95] S. Meng, A. Iyengar, I. Rouvellou, L. Liu, K. Lee, B. Palanisamy, Y. Tang Reliable State Monitoring with Messaging Quality Awareness. *IEEE Cloud*, 2012.

[96] J. Meyerowitz and R. Choudhury. Hiding Stars with Fireworks: Location Privacy through Camouflage In *MOBICOM 2009*

[97] M. Mihailescu, G. Soundararajan, C. Amza MixApart: Decoupled Analytics for Shared Storage Systems In *FAST*, 2013.

[98] M. Mokbel, C. Chow, and W. Aref. The New Casper: Query Processing for Location Services without Compromising Privacy. In *VLDB*, 2006.

[99] R. J. Mokken. Cliques, clubs and clans. In *Quality and Quantity*, 1973.

[100] K. Mouratidis and M. Yiu Anonymous Query Processing in Road Networks In *TKDE*, 2010.

[101] K. Morton, A. Friesen, M. Balazinska, D. Grossman. Estimating the Progress of MapReduce Pipelines. In *ICDE*, 2010.

[102] A. Mu'alem , D. Feitelson, Utilization, Predictability, Workloads, and User Runtime Estimates in Scheduling the IBM SP2 with Backfilling. In *IEEE TPDS*, 2001

[103] B. Palanisamy, A. Singh, L. Liu and B. Jain Purlieus: locality-aware resource allocation for MapReduce in a cloud. In *IEEE/ACM Supercomputing*, 2011.

[104] B. Palanisamy and L. Liu Protecting Location Privacy with Mixzones over Road Networks In *ICDE* 2011.

[105] B. Palanisamy, L. Liu, K. Lee, A. Singh and Y. Tang Location Privacy with Road Network Mix-zones In *IEEE MSN* 2012

[106] B. Palanisamy, A. Singh, L. Liu and B. Langston Cura: A Cost-optimized Model for MapReduce in a Cloud. In *IPDPS*, 2013

[107] X. Pan, X. Meng and J. Xu Distortion based Anonymity for Continuous Queries in Location Based Mobile Services. In *GIS*, 2009.

[108] P. Pesti, B. Bamba, M. Doo, L. Liu, B. Palanisamy, M. Weber. GTMobiSIM: A Mobile Trace Generator for Road Networks. College of Computing, Georgia Institute of Technology, 2009, *http://code.google.com/p/gt-mobisim/*.

[109] A. Phanishayee, H. Shah, E. Krevat, D. Andersen, G. Ganger, G. Gibson, B. Mueller, V. Vasudevan. Safe and Effective Fine-grained TCP Retransmissions for Datacenter Communication. In *SIGCOMM* 2009.

[110] A. Popescu, V. Ercegovac, A. Balmin, M. Branco, A. Ailamaki Same Queries, Different Data: Can we Predict Runtime Performance? *SMDB*, 2012.

[111] P. Prabhu, S. B. Navathe, S. Tyler, V. Dasigi, N. Narkhede, B. Palanisamy LITSEEK: Public health literature search by metadata enhancement with External knowledge bases *DTMBIO* 2009.

[112] S. Quinlan and S. Dorward Venti: a new approach to archival storage. In *FAST*, 2001.

[113] E. Rosti , E. Smirni , L. Dowdy , G. Serazzi , B. Carlson Robust Partitioning Policies of Multiprocessor Systems. In Performance Evaluation, 1993.

[114] I. Roy, Srinath. Setty, A. Kilzer, V. Shmatikov, E. Witchel Airavat: Security and Privacy for MapReduce *NSDI*, 2010.

[115] T. Sandholm and K. Lai. Mapreduce optimization using dynamic regulated prioritization. In *ACM SIGMETRICS/Performance*, 2009.

[116] S. Sehgal, M. Erdlyi, A. Merzky, S. Jha Understanding application-level interoperability: Scaling-out MapReduce over high-performance grids and clouds. In Future Generation Computer Systems, 2011

[117] A. Serjantov and G. Danezis. Towards an Information Theoretic Metric for Anonymity. *PETS*, 2002.

[118] S. Setia, S. Tripathi A Comparative Analysis of Static Processor Partitioning Policies for Parallel Computers. In MASCOTS, 1993.

[119] K. C. Sevcik Application Scheduling and Processor Allocation in Multiprogrammed Parallel Processing Systems. In Performance Evaluation, 1994.

[120] V. Shmatikov and M. Wang. Timing analysis in low-latency mix networks: attacks and defenses. In *ESORICS*, 2006.

[121] A. Singh, M. Korupolu, and D. Mohapatra. Server-storage virtualization: Integration and load balancing in data centers. In *IEEE/ACM Supercomputing*, 2008.

[122] J. Skovira, W. Chan, H. Zhou, D. Lifka The EASY - LoadLeveler API Project In *IPPS*, 1996.

[123] B. Sotomayor, K. Keahey, I. Foster Combining Batch Execution and Leasing Using Virtual Machines In *HPDC*, 2007.

[124] S. Srinivasan, V. Subramani, R. Kettimuthu, P. Holenarsipur, P. Sadayappan Effective Selection of Partition Sizes for Moldable Scheduling of Parallel Jobs. In HIPC, 2002.

[125] A. Streit On Job Scheduling for HPC-Clusters and the dynP Scheduler. In *HiPC*, 2001

[126] Y. Tang, T. Wang, L. Liu, S. Meng, B. Palanisamy Privacy-Preserving Indexing for e-Health information networks *CIKM* 2012.

[127] A. Thusoo, J. Sarma, N. Jain, Z. Shao, P. Chakka, S. Anthony, H. Liu, P. Wyckoff and R. Murthy Hive - A Warehousing Solution Over a MapReduce Framework In *VLDB*, 2009.

[128] A. Thusoo et al. Data warehousing and analytics infrastructure at Facebook In *SIGMOD*, 2010.

[129] F. Tian and K. Chen Towards Optimal Resource Provisioning for Running MapReduce Programs in Public Clouds In *CLOUD*, 2011

[130] G. Toth, Z. Hornak and F. Vajda. Measuring Anonymity Revisited. In Norsec, 2004.

[131] C. Troncoso, B. Gierlichs, B. Preneel and I. Verbauwhede. Perfect Matching Disclosure Attacks *PETS*,2008.

[132] A. Verma, P. Ahuja, and A. Neogi. pMapper: Power and Migration Cost Aware Placement of Applications in Virtualized Systems. In *ACM Middleware*, 2008.

[133] A. Verma, L. Cherkasova, and R. H. Campbell Resource Provisioning Framework for MapReduce Jobs with Performance Goals In *Middleware*, 2011.

[134] G. Wang, A. Butt, P. Pandey, K. Gupta. A Simulation Approach to Evaluating Design Decisions in MapReduce Setups. *MASCOTS*, 2009.

[135] T. Wang and L. Liu. Privacy-Aware Mobile Services over Road Networks In *VLDB 2009*

[136] T. Wang and L. Liu. Execution Assurance for Massive Computing Tasks In *IEICE Transactions on Information and Systems*, Vol. E93-D, No. 6 (June 2010), Special session on Info-Plosion.

[137] J. B. Weissman, A. S. Grimshaw A framework for partitioning parallel computations in heterogeneous environments. In Concurrency - Practice and Experience 7(5): 455-478 (1995)

[138] P. Williams, R. Sion Usable PIR. In *NDSS*, 2008.

[139] T. Wood, P. Shenoy, A. Venkataramani and M. Yousif Black-box and Gray-box Strategies for Virtual Machine Migration. In *NSDI*, 2007.

[140] P. Xiong, Y. Chi, S. Zhu, H. Moon, C. Pu, H. Hacigumus Intelligent Management of Virtualized Resources for Database Systems in Cloud Environment In *ICDE*, 2011.

[141] P. Xiong, Y. Chi, S. Zhu, J. Tatemura, C. Pu, H. Hacigumus ActiveSLA: A Profit-Oriented Admission Control Framework for Database-as-a-Service Providers In *SOCC*, 2011.

[142] M. Zaharia, A. Konwinski, A. D. Joseph, R. Katz, I. Stoica. Improving MapReduce Performance in Heterogeneous Environments. In *OSDI*, 2008.

[143] M. Zaharia, D. Borthakur, J. Sen Sarma, K. Elmeleegy, S. Shenker, I. Stoica Delay Scheduling: A Simple Technique for Achieving Locality and Fairness in Cluster Scheduling *EuroSys*, 2010.

[144] K. Zhang, X. Zhou, Y. Chen, X. Wang, Y. Ruan Sedic: Privacy-Aware Data Intensive Computing on Hybrid Clouds In *CCS*, 2011.

VITA

Balaji Palanisamy was born and raised in Pondicherry, India. He received the B.Tech degree in Computer Science from Pondicherry Engineering College in 2006. Balaji came to Georgia Institute of Technology in August 2007 for his graduate studies in Computer Science and since August 2008, he has been working toward his Ph.D. degree at the College of Computing, Georgia Institute of Technology, advised by Prof. Ling Liu. At Georgia Tech, Balaji was affiliated with the Distributed Data-intensive Systems Laboratory (DiSL) and the Center for Experimental Research in Computer Science (CERCS). His research interests include large scale distributed systems with a focus on performance optimization, cost-effectiveness and privacy. Since May 2010, he has collaborated with the Storage Optimization Services group in IBM Almaden Research Center. His dissertation proposes new techniques for optimizing MapReduce services in a Cloud infrastructure considering the aspects of performance, cost-optimization and privacy.

Outside academics, Balaji has a deep passion for music. He has studied piano for several years in his childhood and undergraduate days and has taken Grade exams from the Trinity College of Music, London. He enjoys playing the piano during his spare time.