2015

An Enhanced MapReduce Workload Allocation Tool for Spot Market Resources

John Stephen Hudzina
Nova Southeastern University, jshudzina@gmail.com

An Enhanced MapReduce Workload Allocation Tool for Spot Market Resources

by

John Stephen Hudzina

A dissertation submitted in partial fulfillment of the requirements
for the degree of Doctor of Philosophy
in
Computer Science

Graduate School of Computer and Information Sciences
Nova Southeastern University

2015

We hereby certify that this dissertation, submitted by John Hudzina, conforms to acceptable standards and is fully adequate in scope and quality to fulfill the dissertation requirements for the degree of Doctor of Philosophy.

Gregory E. Simco, Ph.D. Chairperson of Dissertation Committee	Date
Sumitra Mukherjee, Ph.D. Dissertation Committee Member	Date
Francisco J. Mitropoulos, Ph.D. Dissertation Committee Member	Date

Approved:

Eric S. Ackerman, Ph.D. Dean, Graduate School of Computer and Information Sciences	Date

Graduate School of Computer and Information Sciences
Nova Southeastern University

2015

An Abstract of a Dissertation Submitted to Nova Southeastern University in Partial Fulfillment of the Requirements for the Degree of Doctor of Philosophy

An Enhanced MapReduce Workload Allocation Tool for Spot Market Resources

by
John Stephen Hudzina
2015

When a cloud user allocates a cluster to execute a map-reduce workload, the user must determine the number and type of virtual machine instances to minimize the workload's financial cost. The cloud user may rent on-demand instances at a fixed price or spot instances at a variable price to execute the workload. Although the cloud user may bid on spot virtual machine instances at a reduced rate, the spot market auction may delay the workload's start or terminate the spot instances before the workload completes. The cloud user requires a forecast for the workload's financial cost and completion time to analyze the trade-offs between on-demand and spot instances.

While existing estimation tools predict map-reduce workloads' completion times and costs, these tools do not provide spot instance estimates because a spot market auction determines the instance's start time and duration. The ephemeral spot instances impact execution time estimates because the spot market auction forces the map-reduce workloads to use different storage strategies to persist data after the spot instances terminate. The spot market also reduces the existing tools' completion time and cost estimate accuracy because the tool must factor in spot instance wait times and early terminations.

This dissertation updated an existing tool to forecast map-reduce workload's monetary cost and completion time based on spot market historical traces. The enhanced estimation tool includes three new enhancements over existing tools. First, the estimation tool models the impact to the execution from new storage strategies. Second, the enhanced tool calculates additional execution time from early spot instance termination. Finally, the enhance tool predicts the workloads wait time and early termination probabilities from historic traces. Based on two historical Amazon EC2 spot market traces, the enhancements reduce the average completion time prediction error by 96% and the average monetary cost prediction error by 99% over existing tools.

Acknowledgements

I would like to thank my wife Toni for her care, encouragement, and infinite patience during the dissertation process. Without here love and support, I would not have been able to complete this task. She has helped me remember the important things in life. I would also like to dedicate the dissertation to my mother-in-law, Bern, who passed away while attending the Doctoral classes.

Thank you, Dr. Simco for serving as my dissertation advisor and for the time spent reviewing the many drafts. The copious amount of feedback has been most helpful and encouraging. I would also like to thank Dr. Mukherjee and Dr. Mitropoulos for serving on the committee.

Finally, I would like to thank several friends and colleagues for the support during the classwork and the dissertation. Thank you, Dr. Chris Powell for setting me on this path. Thanks for the support, advice, and friendship Jeremy Ulstad, David Sayer, Dr. Ray Halper, and Dr. Ron Krawitz.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Background

When a cloud user executes a map-reduce workload, the cloud user first specifies a virtualized cluster to meet completion time and budget constraints (Herodotou, Dong, & Babu, 2011). The cloud user must estimate resource requirements and financial costs for the workload because the completion time and monetary cost varies based on the virtual cluster specification. As a result, the cloud user evaluates alternative virtual cluster configurations to minimize the monetary cost of the map-reduce workload (Herodotou et al., 2011). Short term resources, like spot virtual machine instances, provide one option for reducing monetary cost of the virtual cluster to the cloud user. This work examined the impact of short-term resource provisioning on the estimation of resource requirements and financial costs of map-reduce workloads.

While workload estimation tools predict the monetary costs and completion time for on-demand instances, the estimation tools can't predict monetary costs and completion time for spot instances because a spot market auction dictates the workload’s instance’s start time and duration (Chohan et al., 2010). The spot market auction reduces the current estimation tools’ prediction accuracy because the tools don’t forecast the spot instance's availability to process the workload’s data. However, this dissertation’s improved estimation tool allowed cloud users to analyze the tradeoffs between the on-demand and spot instances because an updated tool forecasted each alternative’s completion time and monetary costs based on the spot instance’s availability.

This section introduces map-reduce workloads and spot instance virtual machines to establish context on provisioning map-reduce clusters. First, an overview of map-reduce framework provides background information for later discussions on map-reduce performance. Next, this section examines the resource allocation for map-reduce analytics to highlight the need for providing monetary cost estimation tools and techniques. Next, on-demand and spot instances will be introduced to describe the potential monetary cost impact on the cloud user. Finally, the potential uses of spot instance with map-reduce workloads will be discussed.

Before this paper discusses spot-instance's impact on map-reduce workloads, an overview of map-reduce is required. Map-reduce enables cloud users to perform processing over data stored as key-value pairs in a distributed file system, like the Google File System (GFS) (Dean & Ghemawat, 2008) or the Hadoop Distributed File System (HDFS) (Apache Software Foundation, 2012). Map-reduce processes data expressed as key-value pairs in two stages map and reduce. First, the map stage splits input data into map tasks to be distributed to map workers that are co-located with the distributed file system. When each map task processes the input data, the map task outputs intermediate data to a local file system to be collected and used in the reduce phase. Then, the reduce stage partitions and transfers the intermediate data to a series of reduce tasks. The reduce tasks are distributed to reduce workers that write the output data back to the distributed file system. Both map-reduce and file system processes reside on the same virtual machines in the cloud environment.

The cloud user performs a four step process to execute map-reduce analytics in a cloud environment (Herodotou & Babu, 2011). Step 1, the user must specify the number and type of virtual machines to use for the cluster. Table 1 shows the features and costs of various types of virtual machines from Amazon EC2 (Amazon Web Services LLC, 2011; Chohan et al., 2010). As part of step 1, the user has a choice of pricing options for the given instance type between on-demand, spot, or a combination of the two. Step 2, the user then must upload or use data resident in a file system hosted on the cloud environment. The user chooses the file system to perform the map-reduce processing against. The options for file systems include an external file system, like Amazon S3, or hosted locally, HDFS, on a virtual cluster provisioned in step 1 (Chohan et al., 2010). Step 3, the user selects a program or workload to run. Step 4, the user supplies a configuration and submits the job for execution.

Table 1: EC2 Instance Types and Cost

Type	CPU (EC2 Units)	Memory (GB)	Storage (GB)	I/O Performance	On Demand Cost ($/hour)	Ave. Spot Cost ($/Hour)
m1.small	1	1.7	160	Moderate	0.085	0.0399
m1.large	4	7.5	850	High	0.34	0.1673
m1.xlarge	8	15	1,690	High	0.68	0.3197
c1.medium	5	1.7	350	Moderate	0.17	0.0798
c1.xlarge	20	7	1,690	High	0.68	0.3233

The map-reduce execution process differs from a traditional environment because the responsibility to specify cluster resources has shifted from a system administrator to a cloud user. In a traditional non-cloud environment, an expert system administrator specifies the map-reduce cluster based on resource and monetary requirements of a set of map-reduce workloads for multiple users. The system administrator uses multi-workload traces with node allocation and CPU utilization metrics to determine how well a production cluster is utilized (Kavulya, Tan, Gandhi, & Narasimhan, 2010). In contrast, a cloud user specifies the map-reduce cluster because virtualized map-reduce clusters are provisioned on-demand for a specific workload. For example, Amazon Elastic MapReduce installs and provisions a map-reduce cluster only for the duration of a single map-reduce workload using virtual machines (Herodotou et al., 2011). As part of the cluster specification that occurs in steps 1 and 2, the cloud user decides on the virtual machine type, number of virtual machines, pricing model, and file system.

When the cluster specification moves to the cloud user, the cloud user must also manage the virtual clusters monetary costs. The cloud user must estimate the monetary costs for the specified cluster because virtual resources specified are charged by a usage rate. The pricing for resources include VM usage rates in CPU hours and data transfer rates GB/month (Wieder, Bhatotia, Post, & Rodrigues, 2010).
Thus, the cloud user must estimate the resource usage of the workload to estimate the financial impact.

Although the cloud user specifies the virtual machine type, number of virtual machines, and file system, this dissertation focused on the pricing model's impact on map-reduce workloads financial costs. Cloud providers offer different pricing models to the cloud user, on-demand and spot, because the cloud provider attempts to reduce financial costs by improving server utilization. On-demand instances improve server utilization because on-demand instances enable virtual machine statistical multiplexing. On-demand instances are a form of statistical multiplexing because some cloud users will have monthly or seasonal demand peaks and minimal to no usage the remainder of the year (Armbrust et al., 2010). When a cloud user allocates on-demand virtual machines to a workload, the cloud user pays for the on-demand virtual machines only during use.

Spot instances improve average server utilization because discounted prices incentivize the use of resources during non-peak hours (Liu, 2011). The spot instance virtual machine types contain the same amount of resources as the on-demand virtual machine types. However, cloud users benefit from spot instances because the cloud provider discounts up to 29% of the on-demand price when demand is low (Chohan et al., 2010). When demand is high, spot instance helps shift demand on server resources because the cloud provider terminate the spot instances during peak loads by the cloud provider.

The cloud provide allocates spot instances with an auction (Chohan et al., 2010). Before the auction starts, the user bids a maximum price per hour for a spot instance. The user will only be allocated the virtual machine instance if the bid is at or above the market price. The market price varies based on demand and overall server utilization, however, the provider may set a reserve or minimum price for low demand. If the user bid drops below the market price, the spot instance is terminated early. If provider terminates the instance early, the user does not pay for the partial hour.

Not all workloads can use spot instances because some workloads, like web applications, must service real-time user requests (Liu, 2011). Spot instances are not appropriate for servicing real-time requests because the spot instance allocation maybe delayed longer then the required response time by the auction process. The spot instance allocation occurs during an auction on a fixed interval (Stokely, Winget, Keyes, Grimes, & Yolken, 2009). The auction can delay spot instance allocation because the spot instance allocation might need to wait several intervals for a successful bid.

Previous studies (Chohan et al., 2010; Liu, 2011) have examined map-reduce workloads with spot instances because the cloud user can schedule the map-reduce workload at non-peak hours when prices are low. Map-reduce workloads can be scheduled at non-peak hours because the map-reduce workload has slack in the expected completion time (Herodotou et al., 2011). In essence, the cloud user expects to wait overnight for the workload to complete. The slack in completion time means the map-reduce workload can wait the several auction intervals required for a successful bid.

In summary, spot instances provide a means to reduce monetary cost associated with executing a map-reduce workload on a cloud environment. However, the cloud user must estimate the map-reduce workload's financial impact of using spot instance prior to allocating them in a cluster. Cloud users require a means of estimating resource requirements and financial costs of map-reduce workloads.

Problem Statement

As noted in the background, the cloud user allocates virtual cluster resources for a map-reduce workload to meet processing time and budget constraints (Herodotou et al., 2011). The cloud user may decide to use spot instances in the virtual cluster as a potential means to reduce costs. However, when the cloud user chooses a spot instance based cluster, the cloud user must compare the spot instance cluster against other cluster alternatives to determine the financial savings (Herodotou et al., 2011). Cloud users are unable to compare cluster alternatives because the users cannot estimate the workload's execution time and monetary cost for each alternative (Wieder et al., 2010).

Although this paper focuses on spot instances, the virtual cluster specification impacts the map-reduce workload's execution time and monetary cost. The cloud user specifies a virtual cluster with the following four decisions: virtual machine type, number of virtual machines, pricing model, and storage options. The problem statement examines each decision to analyze the decision's impact on the map-reduce workload's performance, financial cost, and predictability.

Virtual Machine Type

Cloud users need to evaluate different virtual machine types. Cloud environments contain different virtual machine types because cloud providers attempt to improve server utilization by statistical multiplexing of different workloads. For example, Amazon EC2 currently supplies 36 different instance types for different web application, high-performance computing, and image processing workloads (Amazon Web Services Inc, 2014). Cloud providers must provide a variety of virtual machine instance types to enable a diverse set of workloads with different demand peaks for statistical multiplexing. Based on a simulation of a production data center 159K VM traces, provisioning complimentary workloads with different demand peaks reduces the number of physical servers by 45% in a data center (Meng et al., 2010).

Table 2: Comparison of VM types for a 6-node map-reduce job

Type	CPU Units	I/O Performance	M/R slots	Memory pre slot (MB)	Cost ($/hour)	Run Time (min)	Total Cost ($)
m1.small	1	Moderate	3	300	0.085	1,000	8.50
m1.large	4	High	5	1024	0.34	200	6.80

The virtual machine type impacts the map-reduce workload's execution time and monetary cost because of differences in memory, CPU units, and I/O throughput between virtual machine types. When the instance type changed from m1.small to m1.large for a 6 node map-reduce job, the execution time decreased by five times (Herodotou et al., 2011). The change in execution time was a result of increase in computational resources. As seen in Table 2, the m1.large instance type increased the computational resources (memory, CPU units, and I/O throughput) available to the job over the m1.small instance type. The instance type also impacts the workload's total monetary cost because the instance type can improve the workload's execution time. Although the m1.large instance cost four times the hourly price, the m1.large instance reduce the total workload cost by 20% (Herodotou et al., 2011). The cost reduction occurred because the m1.large instance reduced the total number of CPU hours charged to the cloud user by 5 times.

The virtual machine type impacts the cloud user's ability to estimate execution time and monetary cost because the virtual machine type changes the map-reduce task's compute rate. The workload execution time depends on the time to complete individual map and reduce tasks or the compute rate (Lee, Chun, & Katz, 2011). Virtual machine types have different compute rate depending on the resources available and the workload resource utilization. For instance, a Graphics Processor Unit (GPU) capable map-reduce job speeds up the compute rate by 2x using GPU enabled virtual machine types versus standard virtual machine types (Lee et al., 2011).

Number of Virtual Machines

The cloud user must decide on the number of virtual machines to allocate for the map-reduce workload. Cloud environments allow workloads to scale the number of virtual machines because cloud user's resource requirements fluctuates based on demand (Armbrust et al., 2010). Production traces demonstrate the fluctuation. Although a non-virtualized production cluster allocated ~40% of 400 nodes on average, the cluster only peaked beyond 80% only for 5 days in a one-year period (Kavulya et al., 2010). Additionally, the trace contains several days of 0% node allocation. In contrast, a virtualized map-reduce cluster is able to take cluster virtual machines offline and allocate the physical resources to other types of workloads when not in use.

The number of virtual machines impacts the map-reduce workload's execution time and monetary cost because map-reduce executes tasks in parallel. The execution time is impacted by the number of virtual machines because map-reduce sub-divides the input data into tasks (n) to be distributed to workers over slots (k) on each virtual machine instance (Dean & Ghemawat, 2008). If $n \geq k$, then the workload can reduce the execution time by allocating more slots. Herodotou et al. (2011) demonstrated a 4x execution time improvement for 6 different map-reduce workloads when number of m1.large instances increase from 5 to 30 nodes.

As with virtual machine types, the workload's monetary cost depends on the workload's relative execution time because parallel workloads contain cost associative properties (Armbrust et al., 2010). Reduced execution time offsets the monetary cost for more instances because the monetary costs are computed from the total machine hours. For instance, Herodotou et al. (2011) found adding 3 times the nodes to a 10-node m1.xlarge cluster speeded up 2 times at a 50% increase in monetary cost.

Table 3: Example Workload Costs with Local Minima

Number of VMs	6	8	10
Cost ($)	2.90	3.80	2.20

The number of virtual machines impacts the cloud user's ability to estimate execution time and monetary cost because the workload's cluster size contains local maxima and minima effects. When a cloud user increments the number of virtual machines for a map reduce workload to reduce cost, the cloud user might experience local cost minima and decide to stop. Table 3 shows an example of local minima in monetary cost for a workload on an m1.xlarge instance (Herodotou et al., 2011). The cloud user might stop testing the workload after 8 nodes because the cloud user found the local minima at 6 nodes.

Pricing Model

After the cloud user selects the virtual machine type, the cloud user then decides on the virtual machine's pricing model. Cloud environments offer spot pricing as an alternative to on-demand pricing because the cloud provider is attempting to shift demand to improve server utilization. Spot instance's hourly price are based on a market price and data center utilization (Orna Agmon Ben-Yehuda, Ben-Yehuda, Schuster, & Tsafrir, 2011). The spot instance virtual machine types have the same resource specifications as the on-demand virtual machine types. However, spot instances may cost 29% of the on-demand instances' hourly price (Chohan et al., 2010). Spot instance helps shift demand on server resources because the spot instance allocation can be terminated during peak loads.

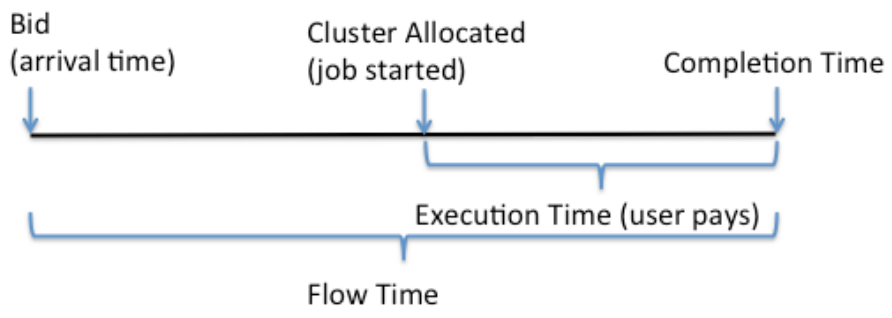

Figure 1. Execution versus Completion Time

Before the problem statement discusses the spot instance's execution time and monetary cost impact, a distinction must be made between completion time and execution time, in respect to monetary cost. Figure 1 shows the difference between completion and execution time. When a cloud user bids on spot instances, a user is able to wait for the cluster to be allocated because the workload contains slack in the completion time (Herodotou et al., 2011). However, the cloud user pays for the instances once the auction allocates the cluster and the workload starts executing. Although the cloud user has slack in the completion time, the cloud user seeks to minimize execution time because the total execution time impacts on monetary costs (Chohan et al., 2010).

Spot instances impact the map-reduce workload's execution time because spot instances provide temporary resources that might terminate before completing the workload. When spot instances are able to complete, additional spot instances reduces the workload's execution time. For a word-count map-reduce workload, adding five spot instances to an existing five instance HDFS cluster improved the execution time by a factor of 2 (Chohan et al., 2010). Similar to increasing the number of on-demand instances, the workload execution time was halved because the number of processing nodes doubled. However, spot instances may terminate before the workload completes.

When spot instances terminate prior to the workload's completion, the additional spot instances increases the execution time. The word-count workload's execution time increased 27% compared to a cluster without the additional five spot instances because an auction terminated the spot instance prior to the workload's completion (Chohan et al., 2010). Early termination increases execution time because the map-reduce scheduler requires time to detect the terminated spot instance and reschedule spot instance's tasks on new nodes. The detection time is caused by a status heartbeat timeout, δ (Chohan et al., 2010). Additionally, the map-reduce scheduler restarts completed tasks from the spot instances on the remaining cluster nodes because the completed tasks' intermediate data is lost from the spot instance's local file system (Wang, Butt, Pandey, & Gupta, 2009). Equation 1 shows the failure cost in additional execution time, where *s* is the total number of instances, *f* is the number of failed instances, and *M* is the total time to complete the map phases (Chohan et al., 2010).

$$failure\ cost = \delta + (\frac{fM}{s})/(s-f) \tag{1}$$

Spot instances reduce the map-reduce workload's monetary cost because spot instances reduce the workload's execution time at a reduced rate compared to on-demand instances. The extra spot instances improved the word-count workload execution time by 2x because the additional five spot instances off-loaded tasks from the on-demand instances (Chohan et al., 2010). The extra spot instances provides extra processing at a reduced rate because the spot instances bid price reduced the rate charged for half of the word count's instances, Equation 2. When the cloud user bid the spot instance \$0.040 per hour, the word-count workload added five spot instances bid for \$0.20 compared to \$0.475 for five additional on-demand instances at \$0.950 per hour (Chohan et al., 2010).

$$totalCost = instances_{si} \times rate_{si} \times time + instances_{od} \times rate_{od} \times time \tag{2}$$

Conversely, spot instances can also increase monetary costs because the intermediate data's recovery increases billed time and rate. The cloud user pays for the map-reduce virtual cluster while the workload is executing. If spot instances terminate early, the extended execution time, Equation 1, increases the total monetary cost. When a spot instance failure causes a 27% slow-down, the workload's price increases by 27% compare to a workload without spot instances because the remaining on-demand instances are being billed for the extra time to recover from the failure (Chohan et al., 2010). When the spot instance terminate before an hour, the price increase correlates to the extra time billed to the core on-demand nodes because the cloud provider charges spot instances in hour increments (Chohan et al., 2010).

The spot instances impact the cloud user's ability to estimate map-reduce workload's execution time and monetary cost because spot instances are subject to early termination causing monetary cost for individual jobs to vary. The spot market's auction varies the market price to remove instances because the spot market contains high demand or inexpensive instances allocated too long (Orna Agmon Ben-Yehuda et al., 2011). In either case, spot instances are intended to be only active for a short amount of time. Based on historical pricing data, the probability a spot instance remains active decreases over time. When the initial market price is \$0.035 and the bid price \$0.041, then the spot instance has an 85% chance of being active after an hour declining to a 40% chance after 6 hours (Chohan et al., 2010).

The cloud user needs to estimate a bid price to avoid early termination and at the same time minimize cost because the early termination probability depends on the bid to market price ratio (Chohan et al., 2010). When the initial market price is \$0.035 and the bid price \$0.036, then the spot instance has a 59% chance of being active after an hour. Compared to an 85% chance of being active after an hour, when the bid price is raised to \$0.041 (Chohan et al., 2010).

Some cloud users tolerate the variability in the map-reduce workload's monetary costs because the cloud user is concerned with the average financial impact of multiple jobs. The cloud user needs to estimate a bid price to reduce the average monetary cost for multiple jobs. If the bid price is too high, then the workload completes without failure but the cloud user's average monetary costs increase. When the initial market price is \$0.035 and the bid price \$0.045, then the workload will complete without failure but the discount is not as great compare to the \$0.085 on-demand price (Chohan et al., 2010). If the bid price is too aggressive or too near the reserve, then the number of failures increases and the cloud user's average monetary costs increase. When the initial market price is at the reserve, \$0.035, and the bid price \$0.036, then the majority of jobs longer then an hour will have spot instance failures increasing the average monetary cost (Chohan et al., 2010).

Low bids increase the workload's failure probability because frequent auctions may increase the market price before the workload completes. Although an auction may allocate the workload's instances near the provider's reserved price, new auctions occur every few minutes to allow new bidders a chance to allocate instances (Chohan et al., 2010). When the new bidder's demand increases beyond the available number of instances, new auctions increase the market price to preempt existing instances (Stokely et al., 2009). The preemptive auction increases the chances a low bid instance terminates before the workload completes because new bidders can preempt existing instances with a higher bid.

Storage Strategy

The cloud user decides on the storage strategy because spot instances require separate storage for the map-reduce workload's output. The temporary spot instances cannot participate in the map-reduce cluster underlying distributed file system because the instance loses any persistent data after de-allocation (Chohan et al., 2010). When the map-reduce cluster contains spot instances, two possible remote storage strategies prevent data loss: HDFS hosted on on-demand nodes or Amazon S3. Each remote storage strategy impacts the map-reduce workload's execution time.

When the cluster uses HDFS with spot instances, the execution time is impacted by reduced data locality compared to a pure on-demand cluster (Chohan et al., 2010). Spot instances reduce data locality because the spot instance accesses persistent data from a remote instance in the map-reduce cluster versus a local disk (Dean & Ghemawat, 2008). The locality percentage impacts the map-reduce workload's execution time because reduced locality increases time to read data from HDFS (Zaharia et al., 2010). In essence, the time to read a byte of persistent data increases with decreased locality because non-local map tasks must wait for data requests to be processed by remote data nodes which could be servicing several requests, Figure 2 (Wang et al., 2009). Specifically, map-reduce workloads with 5% locality increases execution time by as much as a factor of 5 (Zaharia et al., 2010).

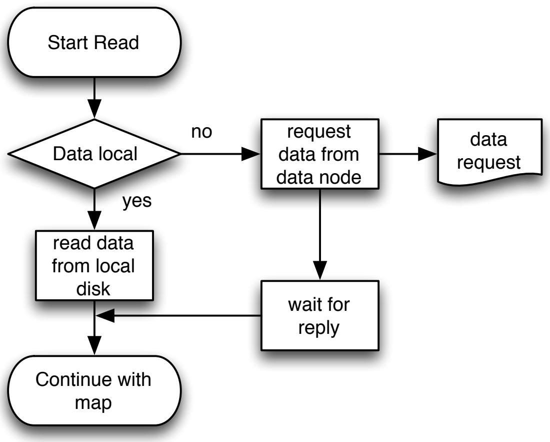

Figure 2. Data Locality Impact

When the cluster uses Amazon S3, the storage strategy impacts execution time because Amazon S3 stores data remotely. Amazon S3 stores files independently of the virtual machines instances in the map-reduce cluster. Although persistent data is accessible on Amazon S3 after the virtual machines are terminated, Amazon S3's file system throughput (bytes/sec) is not a great as HDFS's throughput. Past HDFS and S3 studies demonstrate the difference in throughput between the two file systems. When a map-reduce job reads from HDFS, the retrieval time accounted for less then 1% of the total map processing time (Herodotou & Babu, 2011). When a map-reduce job reads from Amazon S3, then the retrieval time accounted for 60% of the total map processing time (Bicer, Chiu, & Agrawal, 2011). Amazon S3 impacts the retrieval time because the distributed file system's throughput (DfsReadCost in bytes/second) determines the map-reduce workload's read phase time, Equation 3 (Herodotou, 2012).

$$ReadPhaseTime = splitSize \times DfsReadCost + splitSize \times InUncomprCPUCost \qquad (3)$$

The storage strategy impacts the cloud user's ability to estimate map-reduce workload's execution time and monetary cost because the workload interacts with each storage strategy differently (Wieder et al., 2010). When the cloud user evaluates spot instance based workloads, each storage strategy requires independent execution time and monetary cost estimates. The storage strategy impacts execution time and costs because the storage strategy may reduce the task's time to read and write data (Herodotou, 2011).

In summary, a cloud user specifies a cluster comprising a set of virtual resources to execute a map-reduce workload. The cloud user must decide on factors, such as, the virtual machine type, number of instances, pricing model, and data storage approach to specify the virtual cluster. These decision factors exist to help the cloud provider improve server utilization by multi-tenancy, elasticity, and spot markets. However, each factor also impacts the map-reduce workloads completion time, execution time, and financial costs. The cloud user requires a means to estimate the impacts on completion time, execution time, and financial costs to specify a virtual cluster that reduces monetary cost.

Map-Reduce Performance & Cost Estimation Tools

The last section discussed four decision factors for virtual map-reduce cluster allocation: the virtual machine type, number of instances, pricing model, and data storage approach. This section focuses on prior research related to two decision factors, the virtual machine type and number of instances, to better understand the deficiencies with current map-reduce cluster allocation solutions.

Prior research (Herodotou et al., 2011) examined a tool, the Elastisizer, that evaluates virtual cluster alternatives for a cloud user's map-reduce workload. Each virtual cluster alternative contains different virtual machine types and number of instances. The Elastisizer predicts each alternative's performance and financial impact because the tool uses a job profile to model and simulate the workload (Herodotou et al., 2011).

Before this section examines the Elastisizer's profile, Verma, Cherkasova, & Campbell (2011) provides a simple model to estimate a map-reduce workload's execution time. The simple model assumes the map-reduce workload executes a set of tasks in two independent sequential stages (map and reduce). The map-reduce scheduler greedily assigns n tasks to k processing slots for each stage. If μ and λ are the stage's mean and maximum task processing time, then each stage's execution time ranges from $n \times \mu/k$ to $(n-1)\times\frac{\mu}{k}+\lambda$ (Verma et al., 2011).

The simple model provides a starting point to understand the Elastisizer's job profile approach because the simple model decomposes the map-reduce workload into tasks to determine the impact of virtual machine type and number of instances on execution time. The virtual machine type governs the mean and maximum task processing time because the virtual machine type dictates the computational resources (memory, CPU units, and I/O throughput) available to the task (Lee et al., 2011). The virtual machine type and number of instances determine the number of task processing slots per stage, Table 1 (Herodotou, 2012).

Although the tasks' processing time changes with new virtual machine types, the Elastisizer characterizes the workload's task with a baseline profile. The baseline profile serves as a starting point because the profile captures the workload's characteristics for a given virtual machine type. To capture the workload's characteristics, the profile decomposes the tasks to determine the task's execution and data dependencies (Herodotou, 2012).

The profile characterizes the task's execution because the profile decomposes the task into processing phases. The phases model the functions or operations performed by the map-reduce tasks, Table 4 (Herodotou, 2012). For instance, the map task's read phase reads the map task's input data from the distributed file system (Herodotou, 2012). The profile then quantifies the task's execution by capturing each phase's execution time as cost fields, Table 5 (Herodotou & Babu, 2011).

Although the simple model assumes the map and reduce stages are independent, the reduce tasks receive intermediate data from the map tasks (Verma et al., 2011). To capture the task's data dependencies, the profile quantifies the data processed by the task's phases in dataflow fields, Table 6 (Herodotou et al., 2011). The dataflow fields also characterize each stage's size because the dataflow fields capture the total number of map & reduce tasks (Herodotou & Babu, 2011).

Although the initial profile characterizes the task's execution and data, the profile further decomposes the tasks to account for potential changes in virtual machine type and/or input data sizes. The profile characterizes each task's cost statistics (Table 7) and dataflow statistics (Table 8) (Herodotou, 2012). The cost and dataflow statistics fields decompose the individual tasks by capturing the primitive operations' execution times and phase data distribution ratios. The cost and dataflow statistics remain constant for a given virtual machine type because the statistic fields are insensitive to the amount of data being processed per task (Herodotou, 2012). However, the cost and dataflow fields depend on the statistics fields because the task's execution time and data depends on the primitive operations' timing and sequence (Herodotou, 2012).

Table 4: Map Reduce Phases

Task Type	Phase	Description
Map	Setup	Instantiates the map task
Map	Read	Reads the input from the distributed file system as a block of data
Map	Map	Processes the data as an input record
Map	Collect	Partitions the map output and writes the map output to memory
Map	Spill	Writes the map output from memory to local disk when the buffer is full
Map	Merge	Merges the spill files
Map	Clean up	Cleans up the map task
Reduce	Setup	Instantiates the reduce task
Reduce	Shuffle	Reads the intermediate data from the map tasks to the reduce
Reduce	Merge	Merges the intermediate data to minimize space in memory or the local file system
Reduce	Reduce	Processes the intermediate data
Reduce	Write	Writes the reduce output to the distributed file system
Reduce	Clean up	Cleans up the reduce task

Table 5: Cost Fields in a Job Profile

Cost Field	Description
cSetupPhaseTime	Time (ns) to setup a task
cCleanupPhaseTime	Time (ns) to clean up a task
cReadPhaseTime	Time (ns) to preform the read phase in a map task
cMapPhaseTime	Time (ns) to preform the map phase in a map task
cCollectPhaseTime	Time (ns) to preform the collect phase in a map task
cSpillPhaseTime	Time (ns) to preform the spill phase in a map task
cMergePhaseTime	Time (ns) to preform the merge phase in a map or reduce task
cShufflePhaseTime	Time (ns) to preform the shuffle phase in a reduce task
cReducePhaseTime	Time (ns) to preform the reduce phase in a reduce task
cWritePhaseTime	Time (ns) to preform the write phase in a reduce task

Table 6: Dataflow Fields in a Job Profile

Dataflow Field	Description
dNumMappers	Number of map tasks in the job
dNumReducers	Number of reduce tasks in the job
dMapInRecs	Map input records
dMapOutRecs	Map output records
dMapInBytes	Map input bytes
dNumSpills	Number of spills
dSpillBufferRecs	Number of records in spill buffer
dSpillBufferSize	Total size in bytes of records in spill buffer
dSpillFileRecs	Number of records in spill file
dSpillFileSize	Size in bytes of spill
dNumRecSpilled	Total records spilled
dNumMergePasses	Number of merge passes
dShuffleSize	Total shuffle size in bytes
dReduceInGroups	Number of reduce input groups
dReduceInRecs	Number of reduce input record
dReduceInBytes	Number of reduce input bytes
dReduceOutRecs	Number of reduce output records
dReduceOutBytes	Number of reduce output bytes
dCombineInRecs	Number of combine input records
dCombineOutRecs	Number of combine output records
dLocalBytesRead	Bytes read from local file system
dLocalBytesWritten	Bytes written to local file system
dDfsBytesRead	Bytes read from distributed file system
dDfsBytesWritten	Bytes written to distributed file system

Table 7: Cost Statistics from Primitive Operation

Profile Field	Cost Type	Description (time in ns)
csDfsReadCost	I/O	Time to read from HDFS (or Amazon S3) per byte
csDfsWriteCost	I/O	Time to write to HDFS (or Amazon S3) per byte
csLocalIOReadCost	I/O	Time to read from local disk per byte
csLocalIOWriteCost	I/O	Time to write to local disk per byte
csNetworkCost	Network	Time to transfer per byte
csMapCPUCost	CPU	Time to execute mapper per record
csReduceCPUCost	CPU	Time to execute reducer per record
csCombineCPUCost	CPU	Time to execute combiner per record
csPartitionCPUCost	CPU	Time to partition per record
csSerdeCPUCost	CPU	Time to serialize/deserialize per record
csSortCPUCost	CPU	Time to sort per record
csMergeCPUCost	CPU	Time to merge per record
csInUncomprCPUCost	CPU	Time uncompressing input per byte
csIntermUncomCPUCost	CPU	Time uncompressing map output per byte
csIntermComCPUCost	CPU	Time compressing map output per byte
csOutComprCPUCost	CPU	Time compressing job output per byte
csSetupCPUCost	CPU	Time setting up task
csCleanupCPUCost	CPU	Time cleaning up task

Table 8: Dataflow statistics

Profile field	Description
dsInputPairWidth	Average number of bytes of input key-value pairs
dsRecsPerRedGroup	Average number of records per reducers group
dsMapSizeSel	Map selectivity (ratio of input to output) in bytes
dsMapRecSel	Map selectivity (ratio of input to output) in number of records
dsReduceSizeSel	Reducer selectivity (ratio of input to output) in bytes
dsReduceRecSel	Reducer selectivity (ratio of input to output) in number of records
dsCombineSizeSel	Combiner selectivity (ratio of input to output) in bytes
dsCombineRecsSel	Combiner selectivity (ratio of input to output) in number of records
dsInputCompressRatio	Map input compression ratio
dsOutCompressRatio	Map output compression ratio
dsStartupMem	Startup memory per task in bytes
dsSetupMem	Setup memory per task in bytes
dsMemPerMapRec	Memory per map's record in bytes
dsMemPerRedRec	Memory per reducer's record in bytes
dsCleanupMem	Cleanup memory per task in bytes

Based on the initial job profile, the Elastisizer generates a virtual profile to characterize a new target cluster. The Elastisizer can predict the workload's execution time on the target cluster because the virtual profile adapts the workload's characteristics to the target cluster's resources. When the target cluster changes the virtual machine type, the new virtual machine type impacts the primitive operation's execution time. As a result, the task's execution time changes because the task executes a sequence of primitive operations. The virtual profile models the VM type change because the virtual profile contains the execution time for both the primitive operations (i.e. cost statistics) and the tasks (i.e. costs) (Herodotou, 2012).

The virtual profile also enables cost based optimization by adjusting the workload's configuration to fit the new target cluster (Herodotou & Babu, 2011). When the target cluster includes additional virtual machines, the cloud user may adjust the workload's data distribution to improve the workload's execution time (Herodotou, 2012). In essence, the cloud user reduces the map task split size because a smaller split size increases the number of map tasks and reduces the data size processed by each map task. The reduced task data size improves the execution time for individual tasks because each task performs fewer operations. The virtual profile models the split size adjustment because the virtual profile models data proportions (i.e. dataflow statistics) and the number of operations (i.e. the dataflow fields) (Herodotou, 2012).

The Elastisizer combines three different models, Figure 3, to generate the virtual profile for the new virtual cluster that includes: relative fitness model, white box model, and task simulator. The relative fitness model generates the virtual profile's new cost statistics based on the virtual machine type. The white box model generates the virtual profile's cost and dataflow fields based on the statistics and configuration. The task simulator predicts the execution time and data flow for the entire job based on the virtual profile's cost and dataflow fields.

The relative fitness model estimates the virtual profile's cost statistics fields for a new virtual machine type.   As noted, the cost statistics measure the primitive operation's average response time.   A new virtual machine type changes the primitive operation's response time because the new virtual machine provides different resources in terms of CPU and I/O throughput (Seltzer, Krinsky, Smith, & Zhang, 1999). The relative fitness model predicts response time changes because the relative fitness model compares the response times to similar workloads migrating to the new virtual machine type (Herodotou, 2012). Although the relative fitness model predicts primitive operations' average response time caused by a new virtual machine type, the virtual profile requires a white box model and task simulation to estimate the workload's total execution time.

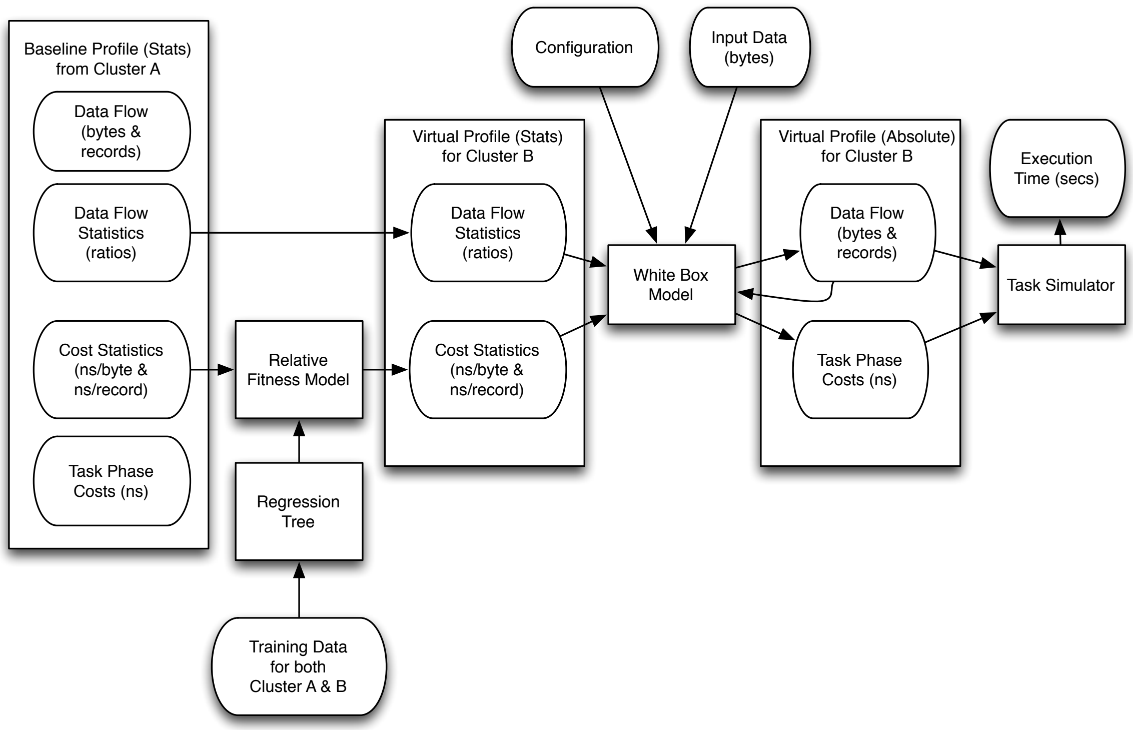
Baseline Profile (Stats) from Cluster A
Data Flow (bytes & records)
Data Flow Statistics (ratios)
Cost Statistics (ns/byte & ns/record)
Task Phase Costs (ns)
Relative Fitness Model
Regression Tree
Training Data for both Cluster A & B
Configuration
Input Data (bytes)
Virtual Profile (Stats) for Cluster B
Data Flow Statistics (ratios)
Cost Statistics (ns/byte & ns/record)
White Box Model
Virtual Profile (Absolute) for Cluster B
Data Flow (bytes & records)
Task Phase Costs (ns)
Execution Time (secs)
Task Simulator

Figure 3. Elastisizer's Workload Simulation

The white box model estimates the virtual profile's cost and dataflow fields. The white box model estimates cost fields calculating the number of operations, operation type, and time per operation for each map-reduce phase based on the cost statistics and job configuration (Herodotou, 2012). Additionally, the white box model quantifies each phase's dataflow by applying the data statistic's ratios to the input sizes from each successive phase (Herodotou, 2012). Although the white box model produces the cost and dataflow field for the individual tasks, the cloud user requires overall execution time to compare alternative clusters.

The task simulator estimates the workload's total execution time by simulating the virtual profile's tasks. The task simulator schedules simulated tasks created from the virtual profile's cost and dataflow on a simulated cluster (Herodotou, 2012). Additionally, the simulator tracks the map-reduce task dependencies to predict the execution order. Based on the simulated schedule, the simulator computes the total execution time (Herodotou, 2012).

The virtual profile's simulation estimates the relative changes between virtual clusters because the simulation's virtual profile captures the relative changes in the primitive operations. However, the simulator over-estimates the absolute execution time by 20.1% on average because the initial job profile contains instrumentation overhead (Herodotou et al., 2011). When the job profiler records the cost statistics from Table 4, the profiler requires extra CPU cycles to measure each operation. The extra CPU cycles skew the initial job profile's measurements because the extra time to record the cost statistics is added to the CPU oriented cost statistic fields. The initial job profile distorts the simulation because the relative fitness model's input contains the extra overhead from the initial profile. However, the profiler overhead causes a uniform prediction error between virtual clusters per job because the simulations all use the same initial profile (Herodotou et al., 2011).

Although the simulator skews absolute execution times, the Elastisizer enables the cloud user to compare alternative cluster configurations for a given workload. Based on the simulator results, the cloud user compares relative workload execution times for each cluster alternative with different virtual machine types and number of instances (Herodotou et al., 2011). The cloud user can also evaluate the workload's monetary costs versus the relative execution time speed-up to select the alternative that provides the best processing time given a budget constraint (Herodotou et al., 2011).

Map-Reduce Performance & Cost Estimation Tools Limitations

The Elastisizer guides the cloud user to allocate the virtual machine type and number of instances for a particular map-reduce workload. However, the Elastisizer does not address the following issues related to the workload's pricing model or storage strategy: determining the spot instance's bid price and storage strategy's impact on the workload's cost.

As previously discussed, the user's bid price impacts the likelihood the workload will complete without the spot instance terminating (Chohan et al., 2010). In essence, spot instance auctions preempt existing users because the new users outbid the existing users. When the auction terminates the spot instance before the workload completes, the failure increases the workload's execution time because the map-reduce framework reschedules tasks to recreate the failed spot instances' missing intermediate data (Chohan et al., 2010).

The Elastisizer does not suggest a bid price for the cloud user because the Elastisizer's simulator lacks the fidelity to estimate the preempted instances impact on the workload's average execution time. Granted, the Elastisizer's simulator estimates the workload's execution time without spot instance failures (Herodotou et al., 2011). However, the Elastisizer's simulator cannot estimate failure costs because the failure's cost impact depends on when the failure occurs. A workload's execution time depends on when the spot instance failure occurs because the failure's time determines the amount of intermediate data to be reprocessed (Wang et al., 2009). The spot instances contain intermediate data produced by several waves of tasks. When the auction terminates a spot instance, the map-reduce scheduler reassigns the spot instance's tasks on the on-demand instances to recreate the intermediate data (Wang et al., 2009). The Elastisizer's simulator must mimic the map-reduce scheduler's behavior to account for the extra task processing time.

Additionally, the Elastisizer's simulator does not calculate the preempted instances' impact on the workload's monetary costs. The workload's monetary cost depends on the number of hours the instances are used. If a spot instance fails after an hour of processing, then the user pays for the spot instance's processing time even though the other instances re-executed the spot instance's tasks (Chohan et al., 2010). The Elastisizer's simulator must account for each instance's duration to calculate the total costs.

The Elastisizer cannot evaluate different storage strategies because the new storage strategies reduce the virtual machine type's cost statistics accuracy. The Elastisizer assumes the cost statistics remain constant for a given virtual machine type because each virtual machine type contains the same CPU and I/O resources. However, the storage strategy changes the workload's I/O throughput. For instance, a local disk workload provides increased I/O throughput compared to a external file system workload due to network bandwidth constraints (Ghemawat, Gobioff, & Leung, 2003).

If the workload accesses an alternative storage system, then the new storage system changes the virtual machine's I/O cost statistics. When comparing Amazon S3 and HDFS performance studies (Bicer et al., 2011; Herodotou et al., 2011) with the same virtual machine type, the storage approach change impacts the cost statistic, "time to read from file system per byte", by 60% relative to the total task processing time. Amazon

S3 increased the map task's read time because the I/O resources dedicated to each virtual machine changed from local disks to an external file system (Bicer et al., 2011).

Additionally, the spot instance storage strategies reduce the cost statistics' accuracy because the workload's locality differs for both Amazon S3 and core HDFS storage strategies. The workload's locality measures the ratio of local versus remote data requests. When the workload uses Amazon S3, all data request are non-local because the workload's map task accesses an external file system. As a result, Amazon S3 workloads require a new virtual profile for each VM type because the workload's cost statistics differs from a workload using the same VM type with local disk (Herodotou et al., 2011).

Although the workload's locality remains constant with the Amazon S3 file system, the individual map task's locality varies with a HDFS workload because the map-reduce framework schedules some map tasks to remote instances. The map-reduce framework attempts to schedule map tasks to instances that contain the task's input data. If the instance local to the task's data is not available, the map-reduce framework schedules the task to the next closest available instance (Zaharia et al., 2010). As a result, the virtual profile uses averages in the cost and cost statistics fields to account for a local and non-local map task mixture (Herodotou, 2012).

When the workload contains tasks assigned to both core HDFS nodes and spot instances, the spot instance to core node ratio varies the workload's cost statistics. The virtual profile assumes the cost statistics remains constant as instances are added because the average task locality remains constant across the instances (Herodotou, 2012). Specifically, the average map read time remains constant because the average map read time includes costs from the same local and non-local map task mixture. However, the average locality depends on the spot instance to core node ratio because all tasks assigned to spot instances access data from non-local instances (Chohan et al., 2010).

In summary, the map-reduce cost estimation tool has not addressed two decision factors the cloud user must make when allocating a virtual cluster: the pricing model and storage approach. The cloud user cannot compare pricing models because the Elastisizer inaccurately simulates the spot market's impact on execution time and monetary costs. The cloud user cannot compare storage approaches because the Elastisizer inaccurately models each new storage approach's impact on the workload's virtual profile.

Dissertation Goal

The goal of this research was to improve map-reduce workloads' completion time and monetary cost prediction accuracy by enhancing previous cost estimation tools to handle spot instances. The enhanced cost estimation tool improved prediction accuracy because the tool forecasted the workload's completion time and monetary cost based on spot instance availability and storage strategy.

The enhanced tool's evaluation compared the enhanced and existing tool's prediction error for the workload's mean completion time and monetary cost. While past studies evaluated on-demand workloads, this work evaluated each of the following scenarios: spot instance workloads without failures, workloads with spot instance failures at fixed times, and workloads with spot instance failures based on a spot market trace. If the enhanced tool reduced each scenario's prediction error, then the enhanced tool successfully improved the map-reduce workload's prediction accuracy for spot instances.

Although the spot instance scenarios evaluated the prediction accuracy based on the spot instance’s availability, this work also evaluated the prediction accuracy based on the storage strategy. The non-failure spot instance scenario also evaluated the prediction error for various cluster sizes and spot instance ratios to measure the enhanced tool’s accuracy improvements. If the enhanced tool reduced the non-failure scenario’s prediction error, then the enhanced tool successfully improved the map-reduce workload’s prediction accuracy for the spot instance’s storage strategy.

Relevance and Significance

As discussed in the problem statement, cloud users are unable to compare cluster alternatives with spot instances because cloud users cannot estimate the map-reduce workload’s execution time and monetary cost for each alternative. In addition, inaccurate workload estimates impact the cloud provider. This section qualifies accurate workload estimates’ relevance and significance on both the cloud user and provider within the context of spot markets. Additionally, this section elaborates on enhanced workload estimation tools that enable cloud users to allocate spot instances.

When cloud users lack accurate workload execution time and monetary cost estimates, the inaccurate estimates affect both cloud users and cloud providers. Inaccurate estimates affect cloud users’ budgets because cloud users cannot allocate the most financially cost effective virtual cluster to execute their workloads. When a cloud user selects the wrong instance type for a 6-node map-reduce job, the wrong choice increases monetary costs over 50% and execution time by 1000% (Herodotou et al., 2011).

Furthermore, inaccurate estimates complicate the cloud user's spot instance cost-benefit analysis because cloud users cannot compare each alternative's monetary cost. For instance, the cloud user saves only 4.8% on a workload's costs using 8 m1.xlarge spot instances compared to 6 c1.xlarge on-demand instances (Herodotou et al., 2011). When spot instances provide minimal monetary cost savings, the cloud user might avoid using spot instances because the cost savings may not warrant the cloud user's effort to migrate the workload (Stokely et al., 2009).

As previously noted, cloud providers benefit from spot markets because spot markets improve average server utilization by shifting demand to non-peak hours (Liu, 2011). In essence, cloud users wait to allocate at low demand periods by bidding a low price. In addition to spot instances, cloud providers offer on-demand and reserve pricing for virtual machines (Wieder et al., 2010). Although spot instances reduce the virtual machine's price per hour, cloud users may select on-demand instances instead.

A purely on-demand market reduces the cloud providers average server utilization compared to the spot market because the cloud provider must rely solely on multi-tenant on-demand instances to improve server utilization. On-demand instances passively improve server utilization because each tenant's individual demand patterns are statistically multiplexed (Armbrust et al., 2010). Multi-tenant map-reduce environments provide a 20% to 40% resource utilization rate (Kavulya et al., 2010) compared to single tenant data centers with server utilization between 5% and 20% (Armbrust et al., 2010). Although multi-tenancy improves server utilization, the cloud provider must equip the data center for peak demand because the cloud provider services all on-demand requests at the time requested.

Inaccurate estimates reduce the spot market's benefit to the provider because cloud users avoid the spot market in favor of the on-demand market without a clear financial incentive. Cloud providers have experienced low spot market demand based on market price history. For example, Amazon EC2 simulated spot market demand until September 2011 because cloud users did not allocate spot instances enough to change the market price on a regular basis (Orna Agmon Ben-Yehuda et al., 2011). After September 2011, spot market demand remains low because the reserve price availability exceeds 90% (Orna Agmon Ben-Yehuda, Ben-Yehuda, Schuster, & Tsafrir, 2013).

Accurate map-reduce estimates benefit both the cloud user and provider because the estimates guide cloud users to provision their workload. If a cloud user accurately predicts the workload's execution time and monetary costs from alternative cluster configurations, then the user will select the most cost effective set of virtual machines (Herodotou et al., 2011). Specifically, accurate estimates allow cloud users to analyze the tradeoffs between workloads with spot and on-demand instances because cloud users are able to quantify each alternative's actual monetary costs (Chohan et al., 2010).

Additionally, accurate workload estimates benefit the cloud provider because accurate estimates incent cloud users to improve resource utilization. Cloud users require the accurate estimates to determine the value of delayed workloads (Chen et al., 2011). The cloud provider benefits from cloud user bidding on spot instances because spot instances are never over-provisioned. Instead, the cloud user waits until the spot instance becomes available at the bid price (Stokely et al., 2009). Delayed requests improve average sever utilization because the spot instance auction delays workloads that would otherwise execute during peak loads.

Accurate workload estimates benefit private cloud providers because private cloud providers prioritize workloads with auctions. Private cloud providers use auctions to prioritize the workloads because the user's bid price expresses the job's value or relative importance to the user (Chun & Culler, 2002). Additionally, the auctions prioritize the workload's immediacy because the user's bid price decreases after each unsuccessful bid. The bid price approaches 0, as the expected completion time approaches the user's required slack time (Chun & Culler, 2002). The auction requires workload estimates to determine when the workload expects to be completed given the allocation delay.

As previously noted, the research goal improves the map-reduce workload's monetary cost prediction accuracy by enhancing previous cost estimation tools to include spot instances. The cost estimation tool addresses the cloud user's allocation problem because the estimation tool predicts monetary costs and execution time for alternative cluster configurations, which includes spot instances. Accurate cost and time estimates enable the cloud user to select the optimal configuration to meet budget and deadline goals (Herodotou, 2012).

The cost estimation tool aids users to allocate spot instances because the estimate includes impacts from bid price (Chohan et al., 2010) and changes in storage strategies (Wieder, Bhatotia, Post, & Rodrigues, 2012). When the cloud user determines the bid price, the cost estimation tool predicts the workload's average monetary cost at the given bid price. When the cloud user determines the most appropriate storage option, the cost estimation tool predicts the monetary costs and execution time for each option. Both spot instance estimates benefit the cloud user because the cost estimation tool provides a complete picture of the costs associated with spot instances.

In summary, cloud users cannot decide when to allocate spot instances because cloud users lack accurate monetary cost and execution time estimates for map-reduce workloads that include spot instances. Inaccurate workload estimates burden both cloud users and cloud providers because cloud users cannot discern the cost of allocating workloads with spot instances. Conversely, accurate workload estimates benefit both the cloud user and cloud provider. The cloud user purchases the most cost effective resources with minimal effort. The cloud provider will improve average server utilization because cloud users are able to shift processing to low demand periods. The research goal addressed the cloud user's allocation problems because the enhanced estimation tool predicts monetary costs for both spot instance and on-demand alternatives.

Barriers and Issue

The previous estimation tools can't compare cluster alternatives with spot instances because the spot market auction impacts the workload's completion time and monetary costs. Although past estimation tools assumed the provider allocates the instances immediately, the auction may delay the instances' start and or terminate the instances early. When a workload allocates spot instances, the auction does not guarantee the workload's completion time because the spot market auction may delay the start of the workload (Stokely et al., 2009). Furthermore, the auction does not guarantee the workload's monetary cost because the terminated spot instances increases the workload's on-demand instances processing time at a higher priced (Chohan et al., 2010).

Although spot markets don't guarantee an instance’s availability, a spot-market aware tool must forecast the workload's completion time and monetary costs. If a cost estimation tool accurately forecasted the spot instance’s duration, then the tool would also accurately estimated the completion time and monetary costs (Chohan et al., 2010). While the enhanced tool needed to forecast the instances duration, the tool also needed to account for the spot instance’s fault tolerance strategy (Schwarzkopf, Murry, & Hand, 2012). The spot instance’s tasks access a remote file system to preserve data after the instance terminates. The enhanced tool needed to estimate the remote tasks’ costs because the remote tasks increased disk and network costs from the concurrent requests (Mesnier, Wachs, Sambasivan, Zheng, & Ganger, 2007).

The solution needed to improve the completion time and monetary cost forecast accuracy. The cost estimation tool needed to improve the completion time forecast accuracy because the tool needed to estimate the wait time to allocate the spot instance and execution time given spot instance failures. Additionally, the cost estimation tool needed to improve the completion time forecast accuracy because the tool needed estimate the slowdown associated with using a fault tolerant file system (Schwarzkopf et al., 2012). The forecast tool needed to improve the monetary cost accuracy because the tool need to forecast the impact of spot instance termination on the monetary costs (Chohan et al., 2010).

Assumptions, Limitations, and Delimitations

This dissertation assumed spot-market traces accurately represented the workload’s lifecycle during an actual spot market. Chohan et al. (2010) and Wieder et al. (2012) assumed spot market traces provide enough information to determine the workload's wait and spot instance termination times. When the spot instances terminated early, both studies assumed a simulator or tool could derive the failure's impact on the workload's execution time (Chohan et al., 2010; Wieder et al., 2012). Wang et al. (2009) assumed a simulator could determine the workload's execution time given node failures.

This dissertation delimited the workloads, spot market traces, and virtual machine types used to evaluate the prediction tools. While the prediction tool's accuracy depends on the specific workload, spot market conditions, and virtual machine type, the results may indicate the tool's performance in other conditions. The previous studies have applied similar delimitations to scope the workloads, spot market traces, and virtual machine types to a tractable level (Chohan et al., 2010; Herodotou, 2012; Wieder et al., 2012).

Definition of Terms

Baseline Profile – A measurement of a workload's average processing time and data flow. The baseline profile is composed of the workload's task phase costs, cost statistics, dataflow, and dataflow statistics.

Completion Time - The time between the job's submission and completion.

Cost Statistics – A decomposition of the task's phase costs into primitive operations.

Dataflow – Quantifies the amount of data processed by the map-reduce tasks and phases.

Dataflow Statistics - Quantifies the data ratios between the map-reduce tasks and phases.

Execution Time – The time between the job's allocation and completion.

Intermediate Data – Temporary data transferred from the workload's map tasks to the reduce tasks.

Map Task – A task, which reads a partitioned input data and writes out intermediate data to be consumed by reduce tasks.

Map-Reduce – A workload that processes data expressed as key-value pairs in two stages: map and reduce.

On-Demand Instance – A virtual machine instance allocated at a fix price when needed by a workload.

Reduce Task – A task, which reads partitioned intermediate data from the map tasks and writes out data to a persistent file system.

Relative Fitness Model – A model that predicts performance changes when a workload is moved from one virtual machine type to another.

Selectivity – A ratio that measures a process' input bytes to the output bytes.

Spot Market – A market where the virtual machine instance's current (or market) price depends on the available supply.

Spot Market Auction – A process where the spot instance's market price is assign by cloud users bids on a number of virtual machine instances.

Spot Instance – A virtual machine instance allocated by a spot market auction.

Spot Instance Accelerator – A map-reduce virtual machine instance that only contains map-reduce processes. In essence, the virtual machine instance contains no underlying distributed file system processed.

Task Phase Costs – The processing time for each phase or part in a task.

Virtualized Cluster – A map-reduce cluster that is composed of virtual machine instances.

Virtual Profile - A model of a workload's average processing time and data flow on a different virtual machine instance type.

Wait Time - The time between the job's submission and allocation.

Summary

When a cloud user executes a map-reduce workload, the user must specify the virtualized resources to execute the workload within a desired budget and or timeframe. To execute a map-reduce workload, the cloud user must decide upon the virtual machine type, number of virtual machines, pricing model, and storage options. The average user requires estimation tools to help evaluate the options because the virtualized cluster's allocation requires specific expertise to estimate the workload's completion time and monetary costs (Herodotou & Babu, 2011).

The cloud user may consider the spot market pricing model as a means to reduce the workload's monetary cost. Spot instances can reduce a workload's monetary cost because the spot instances are allocated by a preemptive auction (Chohan et al., 2010). When the cloud user bids on spot instances during low demand periods, the auction offers the spot instances at a fraction of the on-demand price. However, the cloud user requires an estimate for the workload's completion time and monetary cost because the auction determines the instances start time and duration (Chohan et al., 2010).

Although the estimation tools predict on-demand virtual instance's completion time and costs, the on-demand estimates are unsuitable for spot market based workloads because the spot market auction determines the virtualized instances' availability to process data. The spot market may extend the workload's completion time because the spot market auction may delay the workload's start (Stokely et al., 2009). The spot market may also increase the workload's execution time and, by extension, monetary cost because the auction may terminate the spot instances before the workload completes (Chohan et al., 2010).

The goal of this research was to improve map-reduce workloads' completion time and monetary cost prediction accuracy by enhancing previous cost estimation tools to handle spot instances. This dissertation provided an approach that improved prediction accuracy because the enhanced tool forecasts the workload's completion time and monetary cost based on the spot instance's availability and storage strategy.

Chapter 2

Review of Literature

In cloud environments, cloud users allocate virtualized clusters to execute map-reduce workloads. As noted in the problem statement, cloud providers present cloud users several options to compose the optimal virtual cluster to meet the workload's target monetary cost and processing time requirements. In order for the cloud user to compare the potential virtual clusters, the cloud user requires monetary cost and execution time estimates that includes each of the following options: virtual machine type, number of virtual machines, pricing model, and storage options.

The literature review examines each option to determine each option's execution and financial impact on the workload. Furthermore, this section examines the tools and techniques that predict workload execution time and cost to determine each options impact on the prediction accuracy.

Virtual Machine Type & Number

Both the type and number of virtual machines govern the map reduce workload's execution time because both parameters dictate the amount of resources applied to workload's tasks. The virtual machine type dictates the memory, CPU units, and I/O throughput per task (Herodotou et al., 2011). The number of tasks depends on both the type and number of virtual machines because both parameters govern the number of parallel tasks executed by the cluster. The virtual machine type governs the number of available processing task slots per virtual machine instance (Herodotou et al., 2011). The number of virtual machines determines the total number processing slots in the cluster (Herodotou et al., 2011).

If the task processing time is known, the virtual machine type and number provides the cloud user sufficient information to estimate the execution time. If the cloud user assumes the map and reduce phases to be independent, then makespan estimates each phases task's completion time because map-reduce workloads greedily assigns tasks to the available slots (Verma et al., 2011). Let μ and λ be the mean and maximum task processing time. Let k equal the total number of slots, which is the product of the number of virtual machines and the number of slots per node (see Table 1). When a map reduce phase requires *n* task, the total processing time ranges from $n \times \mu/k$ to $(n-1)\times\frac{\mu}{k}+\lambda$ (Verma et al., 2011).

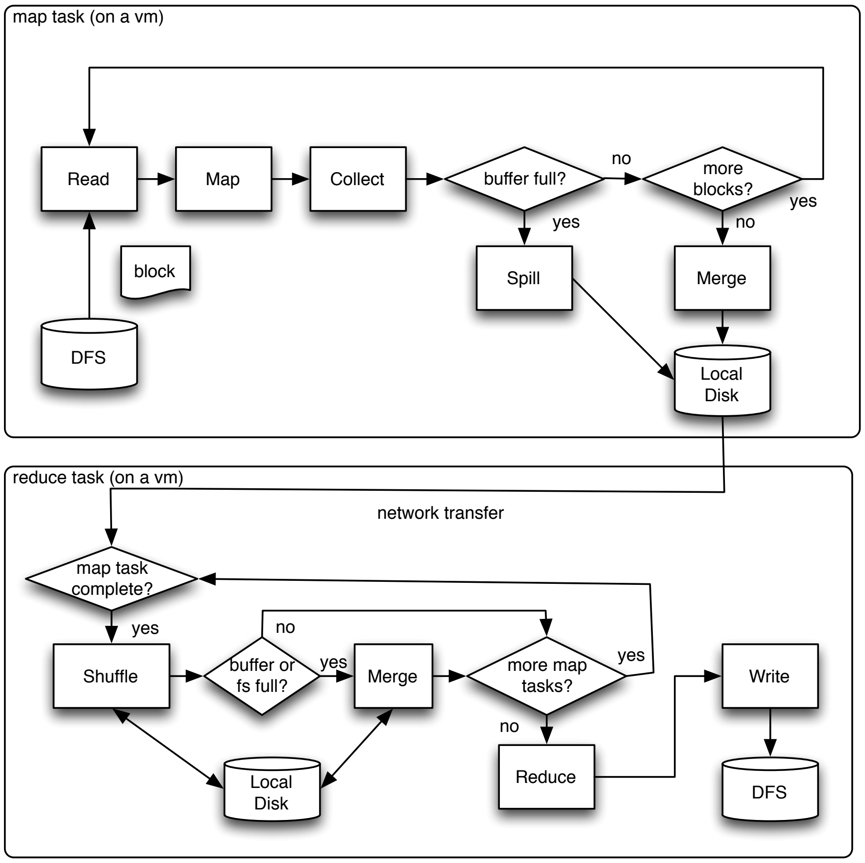

Figure 4. Map Reduce Task Flow

However, the reduce phase depends on the map phase because the map tasks transfer intermediate data to the reduce tasks. The data transfer occurs during the reduce task's shuffle phase because the reduce task sorts the intermediate data in iterative steps. As seen in Figure 4, the shuffle phase transfers intermediate data as each map task completes. When all map tasks complete and the reduce task sorts all the intermediate data, then the reduce task processes the intermediate data in the correct order.

Given the reduce phase dependency, the cloud user still could apply makespan to estimate the execution time because the reduce tasks can be decomposed to account for the overlap in execution time (Verma et al., 2011). Let $T_{Shuffle}$ be the non-overlapping shuffle time where intermediate data is transferred from the map tasks to the reduce tasks.

If $\mathrm{T}_{\mathrm{Map}}$ and T_{Reduce} are the execution times for the map and reduce phases, then the workload's total execution time, T, is Equation 4.

$$\mathrm{T} = \mathrm{T}_{\mathrm{Map}} + T_{Shuffle} + \mathrm{T}_{Reduce} \tag{4}$$

Although makespan estimates the workload's total execution time, the theorem is insufficient for cloud users to estimate workload execution time on new virtual machine types. Makespan requires the cloud user to estimate the average and maximum task execution time for each different virtual machine type. The average and maximum task duration changes for each virtual machine type because the task's available resources (CPU, disk I/O, network I/O) change (Herodotou, 2012). When a workload uses Graphics Processor Unit (GPU) virtual machines, the GPU enabled virtual machine types may decrease the map task execution time up to 2x (Lee et al., 2011). However, the cloud user must execute the tasks on a target virtual machine type to quantify the VM type's impact on task duration because the GPU speed up is workload specific (Lee et al., 2011).

The cloud user can't estimate average and maximum task duration because the map-reduce workload's characteristics change with each virtual machine type. The virtual machine type impacts both the sequence of operations and each operations execution time (Seltzer et al., 1999). Figure 4 illustrates how the spill and merge phase's frequency depends on the memory size available to the virtual machine type (Herodotou, 2012). Furthermore, spill frequency affects the sort operations execution time. The map's spill phase performs both CPU and I/O operations because the spill phase sorts output records and writes the records to local disk. When the spill's memory buffer increases to improve the I/O operations execution time, the buffer size change increases

CPU operation time because the sort operation has more records to sort (Herodotou & Babu, 2011). In conclusion, the cloud user can't simply scale the tasks execution time because the virtual machine type changes the operations performed by the workload.

The Elastisizer

Prior research proposes a tool, the Elastisizer, to address the workload characteristics change issue (Herodotou et al., 2011). The tool quantifies the workload's characteristics via a job profile to capture the dependencies between the workload's operations. Based on the job profile, the tool models the workload's changes in a virtual profile. The virtual profile characterizes the tasks in a potential alternative cluster (Herodotou et al., 2011).

To create the initial profile, Elastisizer model decomposes the map-reduce tasks into phases because each phase's timing and data processed depends on the previous phases (Herodotou, 2012). Table 4 enumerates the map-reduce tasks' phases. Figure 4 illustrates the dependency between each phase. The reduce shuffle and merge phases demonstrate the data dependencies between phases. The shuffle phase times depends on the map tasks because the shuffle phase cannot retrieve data for the map tasks until each map task completes (Wang et al., 2009). Additionally, the merge phase time depends on the data flow from the map tasks because the merge's frequency is governed by the amount of data processed by the shuffle phase (Wang et al., 2009).

The Elastisizer's profile models the individual map and reduce tasks because the profile characterizes each task's execution and data. The profile characterizes the task's execution by capturing each phase's execution time as cost fields, Table 5 (Herodotou & Babu, 2011). The cost fields characterize the task's execution because the cost fields decompose the task's execution time into phases. Additionally, the profile characterizes the task's data because the profile quantifies the data processed by the task's phases in dataflow fields, Table 6 (Herodotou et al., 2011). The dataflow fields also characterizes the job's data dependencies because the dataflow fields captures the number of map & reduce tasks (Herodotou & Babu, 2011).

Although the profile's cost and dataflow fields characterize the map and reduce task's phases, configuration tweaks invalidate the workload's characterization. The workload's configuration governs the amount of data and type of operations performed by the tasks. For instance, the split size configuration governs the amount of data a map task processes. A split size increase invalidates the cost fields because the larger task data size increases the map task's execution time (Herodotou, 2012). A larger split size also invalidates the dataflow fields because the scheduler divides the input data among fewer tasks (Herodotou, 2012).

However, the profile adjusts to configuration changes because the profile estimates costs with normalized cost and dataflow statistics fields. The cost statistics normalize the profile's costs because the cost statistics further decomposes the phases to operations, Table 7. Each phase performs a set of primitive operations to process the input data. Although the workload's configuration controls the number of operations per phase, the configuration does not impact the discrete operation's execution time (Herodotou & Babu, 2011). The cost statistics normalizes the task's execution time because the cost statistics measures the operation's execution time per byte or record (Herodotou, 2012).

The dataflow statistics normalizes the profile's dataflow because the dataflow statistics further decomposes the data distribution, Table 8. The workload's configuration impacts the amount of data per task or phase. Although each phase's data size depends upon the preceding phases, the data distribution ratio remains constant between phases for a given workload (Herodotou & Babu, 2011). The dataflow statistics decomposes the data distribution between phases because the statistics captures input to output ratios for each operation, phase, and task (Herodotou, 2012).

Although the cost and dataflow statistics remain constant for various configurations, the cost statistics change with a new virtual machine type (Herodotou, 2012). The cost statistics represent primitive operation execution times. The primitive operations consume the virtual machine's available CPU, I/O, and network resources (Seltzer et al., 1999). When the workload executes against a new virtual machine type, the virtual machine's resources impact the primitive operation's execution times. For instance, a GPU virtual machine reduces map execution time for a cryptographic workload because additional processors reduce the CPU time to perform hash operations (Lee et al., 2011).

In summation, the workload's profile applies to only one cluster because the workload's cost statistics, cost, and dataflow fields are impacted by changes to the virtual machine type, configuration, and input data. Although the profile cannot directly characterize the workload on a different cluster, the profile provides enough information to model a virtual profile (Herodotou, 2012). The virtual profile captures the changes to the baseline profile when the workload executes with a new virtual machine type, configuration, and or input data (Herodotou, 2012).

The Elastisizer combines three different models to generate the virtual profile for the new cluster (see Figure 3): relative fitness model, white box model, and a task simulator. The relative fitness model predicts the execution time for the task's primitive operations because the model estimates the changes to the virtual profile's cost statistics caused by the new virtual machine type. The white box model predicts the individual task's execution time and data flow for individual tasks because the model calculates the new costs and dataflow fields based on the configuration, cost statistics and dataflow statistics. The task simulator predicts the execution time and data flow for the entire job based on the cost and dataflow fields. The next three subsections describe the details behind each model.

Elastisizer - Relative Fitness Model

The relative fitness model predicts the changes to the workload's cost statistics cause by a new virtual machine type, Figure 5. When the workload executes against a different virtual machine type, the new virtual machine type provides different resources, in terms of CPU units, memory size, and disk throughput (as seen in Table 1). The new resources change the cost statistics' response times because the primitive operations consume a different percentage of the virtual machine's resources (Seltzer et al., 1999).

If the cloud provider quantifies the virtual machine's resource sizes and throughputs, then response times could be estimated by a vector-based methodology (Seltzer et al., 1999). The vector-based methodology estimates primitive operation's response times by calculating the dot product between the available resources and the operation's demands for the resources. However, the cloud user cannot estimate the cost statistics based on the available resources because the cloud provider obfuscates the exact resource details, like disk I/O throughput, from the cloud user (Herodotou et al., 2011).

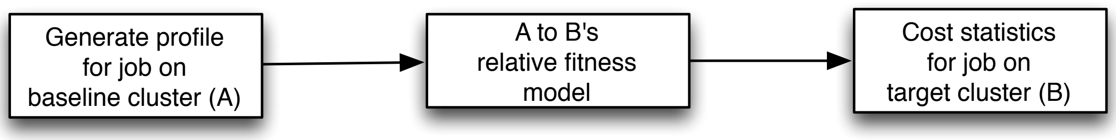
Generate profile for job on baseline cluster (A)
A to B's relative fitness model
Cost statistics for job on target cluster (B)

Figure 5. Relative Fitness Model

Instead, the relative fitness model estimates the new response times by comparing the target workload to benchmarking workloads because benchmarks capture workload characteristics exhibited in a variety of map-reduce jobs for a given VM type. For instance, the benchmarks might include I/O intensive, I/O light, CPU intensive, and CPU light jobs (Herodotou et al., 2011). Each workload type exercises different cost statistics because the primitive operation types and frequencies depend on the workload. For example, a CPU intensive job might be caused by highly selective map tasks. Highly selective map tasks produce few output records relative to the input records read by the map task. As a result, the highly selective map tasks performs fewer spill and write operations then an I/O intensive job (Herodotou, 2012).

The relative fitness model learns the workload characteristics that correspond to the cost statistics' scaling factors or relative fitness by executing a regression tree machine-learning algorithm (Herodotou, 2012). The model generates the regression tree's training set, Table 9, from the source (A) and target (B) VM type benchmark results. The source VM type's cost statistics represent the predictor variables in the training set because the source's VM type characterizes the baseline profile. The relative fitness, $RF_{A\to B}$, represents the predicted variable in the training set because $RF_{A\to B}$

quantifies the percentage the cost statistic changes on the target virtual machine type (Herodotou, 2012)

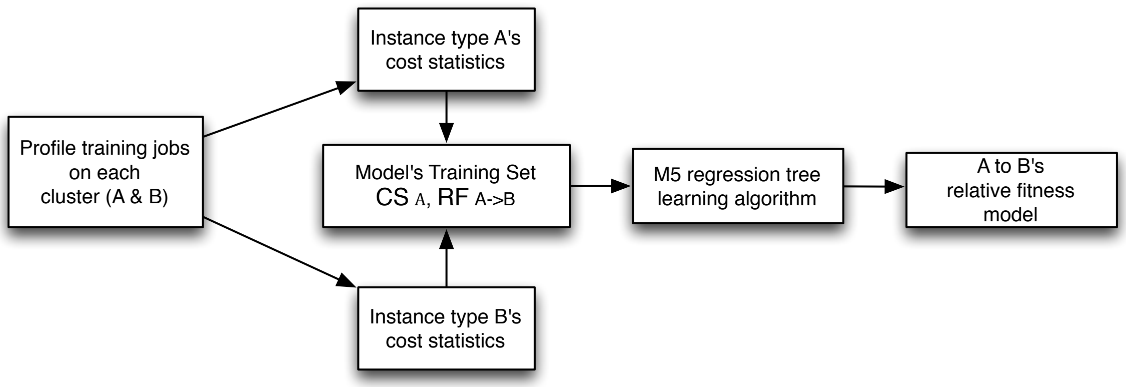
Instance type A's cost statistics
Profile training jobs on each cluster (A & B)
Model's Training Set CS A, RF A->B
M5 regression tree learning algorithm
A to B's relative fitness model
Instance type B's cost statistics

Figure 6. Relative Fitness Model Creation

Table 9: Relative Fitness Training Set for I/O Read Cost from HDFS per Byte

Field_A	Avg_A	*Read HDFS Cost* $RF_{A\rightarrow B}$
Read Local I/O cost / byte	17	.51
Map CPU cost / key	6	.51
Sort CPU cost / key	8	.51
…	…	…

The regression tree predicts the relative fitness because the training set models the relationship between the baseline and virtual profile's cost statistics. If the benchmarks cover all the possible workload characteristics, then the baseline profile should match some of the benchmarks' cost statistics from the source VM type. The regression tree compares the baseline profile's cost statistics against the benchmarks' predictor variables to determine which combination of benchmarks best fits the workload's characteristics (Mesnier et al., 2007). The regression tree then can predict the virtual profile's cost statistics because each predictor variable corresponds to the relative fitness of a virtual profile's cost statistic (Herodotou, 2012).

Although the Elastisizer generates the training set only once, the relative fitness model's accuracy depends on the training set's completeness because the relative fitness model predicts the cost statistic changes based on the similarity to the benchmarks. Prior research demonstrates the prediction accuracy impact caused by an incomplete training set (Herodotou, 2012). An experiment analyzed the prediction error on training sets with and without CPU intensive workloads. When the training set excluded CPU intense workload and the model was applied to a CPU intensive workload, the relative prediction error was 40% greater then the training set that included the CPU intensive characteristics (Herodotou, 2012).

In summary, the relative fitness model predicts the virtual profiles cost statistics because the model quantifies the cost statistics changes for similar workloads. The Elastisizer captures the changes in cost statistics between two VM types for a variety of benchmark workloads. Then a machine-learning algorithm creates a regression tree to match the baseline profile against the benchmark to predict the new virtual profile's cost statistics. However, the new virtual profile is not yet complete because the virtual profile's new dataflow and cost are yet to be determined.

Elastisizer - White Box Model

As discussed, the virtual profile models each individual map-reduce task's execution time and dataflow on a new virtual cluster. Although the virtual machine type impacts the task's execution time, the workload's configuration also influences the task's execution time and dataflow, also known as the application trace. The workload's configuration dictates the type and frequency of operations performed by the map-reduce tasks (Herodotou, 2012). For example, the map task decompresses the input data only if the workload configures the input data to be compressed. When the workload reads compressed input records, the number of decompress operations depend on the configuration's split size. The split size dictates the number of input records to be decompressed because the split size controls the amount of data the task reads (Herodotou, 2012).

Although the configuration impacts the workload's trace, Herodotou & Babu (2011) modeled the trace's changes by using a white box model to predict the operations performed by a task. The white box model decomposes the map-reduce tasks into operations for each phase, Table 10 and Table 11. The white box model predicts the dataflow (amount of data) and costs (amount of time) for the individual tasks because the model uses the cost statistics, data flow statistics, and configuration to calculate each phase's time, type, and frequency of operations (Herodotou, 2012).

The white box model calculates the operation's execution time by using the cost statistics. The white box model requires the cost statistics predicted by the relative fitness model because the virtual profile's cost statistics estimated the primitive operation time changes caused by a new virtual machine type (Herodotou, 2012). Although the cost statistics quantify the operation's timing, the cost statistics do not predict primitive operation's type or frequency (Seltzer et al., 1999).

The white box model calculates each phase's operation types by decomposing the tasks into operations (Herodotou, 2012). The white box model decomposes the map-reduce tasks into a series of phases, Table 5. Each phase performs a number of primitive operations as it processes the input data. Each phase's operation types are controlled by the map-reduce job's configuration. For instance, the map's read phase only executes input decompression when the configuration's input compression field is set to true (Herodotou, 2012).

The white box model calculates the primitive operations' frequency by modeling the dataflow between phases. The white box model requires the phase's dataflow to calculate the phase's execution time because the dataflow quantifies the amount of work performed by each operation (Herodotou, 2012). For instance, the map phase time depends on the map operation's CPU time and number of input records: see the map phase row in Table 10. The cost statistics quantify the map operations' CPU time in nanoseconds per record, Table 7. The white box model calculates the total map phase time using the number of map input records from the virtual profile's dataflow field (Herodotou, 2012):

$$cMapPhaseTime = dMapInRecs \times csMapCPUCost.$$

Table 10: White Box Model Map Parameters

Phase	Cost Statistics	Dataflow Statistics	Configuration	Dataflow (Output)	Costs (Output)
Read	csDfsReadCost csInUncompress	dsInputCompressRatio dsInputPairWidth	pSplitSize pIsInCompressd	dMapInBytes dMapInRecs	cReadPhaseTime
Map	csDfsWriteCost csMapCPUCost csOutComprCPUCost*	dsMapSizeSel* dsOutCompressCPUCost*	pNumReducers pIsOutCompress	dMapOutBytes* dMapOutRecs*	cMapPhaseTime cWritePhaseTime
Collect & Spill	csLocalReadIOCost csLocalWriteIOCost csPartionalCPUCost csSerdeCPUCost csSortCPUCost csCombineCost csIntermComprCPUCost	dsMapSizeSel dsMapPairsSel dsCombineSizeSel dsCombinePairsSel dsIntermCompressRatio	pNumReducers pSortMB pSortRecPerc pSpillPrec pIsIntermCompress	dSpillBufferRecs dSpillBufferSize dNumSpills dSpillFileSize dSpillFileRecs	cCollectPhaseTime cSpillPhaseTime
Merge	csLocalReadIOCost csLocalWriteIOCost csMergeCPUCost csCombineCost csIntermComprCPUCost csIntermUncomprCPUCost	dsMapSizeSel dsMapPairsSel dsCombineSizeSel dsCombinePairsSel dsIntermCompressRatio	pSortFactor pUseCombine pNumSpillsForCombine pIsIntermCompress	dNumRecsSpilled tIntermDataSize tIntermDataRecs	cMergePhaseTime

Table 11: White Box Model Reduce Parameters

Phase	Cost Statistics	Dataflow Statistics	Configuration	Dataflow (Output)	Costs (Output)
Shuffle	csLocalReadIOCost csLocalWriteIOCost csNetworkCost csMergeCPUCost csCominbineCPUCost csIntermComprCPUCost csIntermUncomprCPUCost	dsIntermCompressRatio dsCombineSizeSel dsCombinePairsSel	pNumberReducers pNumMappers pTaskMem pShuffleMergePrec pInMemMergeThr pSortFactor pIsIntermCompress	dShuffleSize	cShufflePhaseTime
Merge	csLocalReadIOCost csLocalWriteIOCost csMergeCPUCost csIntermComprCPUCost csIntermUncomprCPUCost	dsIntermCompressRatio	pReducerInBufPrec pTaskMem pSortFactor pIsIntermCompress		cMergePhaseTime
Reduce	csLocalReadIOCost csIntermUncomprCPUCost csReduceCost	dsIntermCompressRatio dsReduceSizeSel dsReducePairsSel	pIsIntermCompress	dReduceInByte dReduceInRecs	cReducePhaseTime
Write	csDfsWriteCost csOutComprCPUCost	dsOutCompressRatio	pIsOutCompress	dReduceOutBytes dReduceOutRecs	cWritePhaseTime

Elastisizer – Task Simulator

Although the virtual profile characterizes the individual task's dataflow and execution time, the virtual profile does not predict the workload's overall execution time. The Elastisizer simulates the workload's tasks because the map-reduce framework schedules the individual tasks across several virtual machine instances. The workloads total execution time depends on the percentage of tasks executed in parallel or the number of task waves (Verma et al., 2011).

When the white box model completes the virtual profile, the task simulator creates a schedule based on the virtual profile's data flow and execution time to calculate the job's overall execution time. The task simulator performs a discrete event simulation to determine the virtual task's timing and order (Herodotou, 2012). The simulator emulates the map-reduce framework's scheduler to assign tasks to the available slots, Table 1, in a FIFO fashion (Herodotou, 2012). The virtual profile's cost fields, Table 5, quantify the execution time for each simulated task.

The task simulator tracks the map-reduce phases' data dependencies. As previously noted, the reduce shuffle occurs before the map phase completes. Before the map phase completes, the scheduled reduce tasks start to transfer and shuffle the intermediate data from the completed map tasks (Wang et al., 2009). In other words, the initial shuffle's completion time overlaps with the map tasks execution time. The discrete event simulation must estimate the initial shuffle's overlapping time to calculate the overall execution time (Wang et al., 2009).

In summary, cloud users can compare workloads on virtualized map-reduce clusters with different virtual machine types and numbers because the Elastisizer predicts the impact on execution time caused by new resources. To model the workload with a new VM type and or number of instances, the Elastisizer creates a profile that characterizes the map-reduce task's execution time and dataflow on a baseline cluster. The Elastisizer then models a virtual profile to predict the changes to the task's execution time and dataflow caused by the target cluster. Finally, the Elastisizer simulates the workload using the virtual profile to calculate the overall execution time.

Spot Instances

In addition to the virtual machine type and number, the cloud user also decides on the pricing model. The cloud user's pricing options include on-demand and spot instances. Although both options provide the exact same resources for a given virtual machine type (Table 1), the spot instance option impacts both the workload's execution time and monetary costs.

Spot instances affect the workload's execution time and monetary costs because the spot instances are allocated by a preemptive auction. Preemptive auctions allocate a finite number of resources and allow new bidders to preempt existing bidders. These auctions preempt users because the number of successful bids, N, cannot exceed the available number of virtual machines (Orna Agmon Ben-Yehuda et al., 2011). When an existing user's bid is higher then the (N + 1)th bid, the auction terminates the user's existing instances.

```
Given: U users, V virtual machines, $v_u$ allocated VMs, starting price p', increment price
function $g: (v, p)$, and bid selection function $\mathcal{G}(p)$
Set $t = 0, p(0) = p'$
Loop
    Collect winning bids: $v_u(t) = \mathcal{G}(p(t))\ \forall u$
    Calculate excess demand: $z(t) = \sum_u v_u(t) - V$
    if $z(t) \leq 0$ then
        Break
    Else
        Update prices: $p(t+1) = p(t) + g(v(t), p(t))$
        $t = t + 1$
    end if
end loop
```
Figure 7. Ascending Clock Auction

Spot instances impact the workload's monetary cost because supply and demand regulates the spot instance price. For instance, the spot instance price could be regulated by an ascending clock auction algorithm, Figure 7 (Stokely et al., 2009). The algorithm starts with an initial reserve price for the spot instances, *p'*. The auction collects the virtual machine allocation's winning bids. If the number of winning bids exceed the available virtual machines, then the price is incremented using an increment function, $g(v(t), p(t))$. The process repeats until the number of allocated virtual machines is less then the number of available virtual machines (Stokely et al., 2009).

If the cloud user is willing to wait to start the workload, preemptive auctions provide access to instances at reduced prices when demand is low. However, the cloud user cannot predict when a price will be available (Orna Agmon Ben-Yehuda et al., 2011) or the duration the price lasts (Chohan et al., 2010). Previous work has focused on three models to quantify the impact of preemptive auctions on the workload's monetary costs and execution time: initial bid success probability, expected lifetime, and execution time impacts from early termination.

Initial bid success probability

The cloud user may be able to wait for a successful bid because the cloud user's expected completion time contains slack time prior to the workload's start (Herodotou et al., 2011). However, the cloud user needs to know the bid's acceptance chances because the cloud user's slack time is constrained. In other words, the cloud user can only delay the workload's start for a finite amount of time in order to complete the workload on time.

To estimate the bid's acceptance probability, Agmon Ben-Yehuda et al. (2011) quantified the bid's availability based on a spot market's price history or trace. The cloud user's bid price must be equal or greater then the market price for a spot instance to be active. The bid's availability, Equation 5, is the fraction of time in the market price history when a spot instance is active (Orna Agmon Ben-Yehuda et al., 2011). When the bid's availability is low, the cloud user waits longer on average for the desired bid price to be accepted.

$$P(Available|History) = \frac{time_{Active}|bid}{time_{Trace}} \tag{5}$$

Although the availability estimates the bid acceptance probability, the current market price also provides the cloud user a starting point for the bid. Given a current market price and desired slack time, the cloud user can also examine the price duration's cumulative distribution function (CDF) to determine the bid price (Orna Agmon Ben-Yehuda et al., 2011). The CDF measures the length of time between price changes based on the market price history. The price duration CDF aids the cloud user setting a bid price because the CDF indicates the time expected before the next price change. If the cloud user's maximum start time is 60 minutes and the current market price lasts on average 120 minutes before the next change, then the cloud user should set the bid at the current market price to allocate the instance in time (Orna Agmon Ben-Yehuda et al., 2011).

Although the availability probability and price duration CDF help the cloud user select a bid price, neither method estimates the spot instances' early termination probability. Both approaches focus on the bid's initial acceptance by an auction. However, the bid's acceptance does not guarantee the auction will not raise the market price before the workload completes because multiple auctions occur after the initial auction allocates the spot instance (Chohan et al., 2010).

Expected Lifetime

As discussed, preemptive auctions may terminate spot instances before a workload's processing completes because the cloud user can be outbid while the instance is active. Cloud users require an estimate on the spot instance's lifetime because the instance expected lifetime helps the user select a bid price (Chohan et al., 2010). If the instance's expected lifetime is greater than the job's expected execution time, then the workload probably will complete before the spot instance terminates. Otherwise, the workload requires additional processing time and monetary cost because the spot instance likely will terminate before the workload completes.

Cloud providers limit the cloud user's ability to estimate the expected lifetime because the bids cannot be observed and the exact auction algorithm is unknown. In Amazon EC2, a cloud user is unable to observe the bidding history of the competitor (Chohan et al., 2010). Additionally, Amazon does not publish the auction algorithm (Orna Agmon Ben-Yehuda et al., 2011). Instead, Amazon EC2 provides a price history for each instance type, which provides the market price over the last 30 days.

Although the cloud user cannot directly observe the auction, statistical methods exist to model the expected lifetime from a market price history because the market price history provides a means to observe the auction's output. The price history provides market price transitions based on the auction results (Chohan et al., 2010). Chohan et al. (2010) created a price transition probability matrix, $M_{i,j,}$ from the market price transition history. The matrix, $M_{i,j}$, models the likelihood the market price changes from a starting price, i, to a finish price, j, during a fixed time interval. The matrix uses a one-hour time interval because cloud users are billed by the hour (Chohan et al., 2010).

Although $M_{i,j}$ models price transition probabilities in one-hour intervals, Chohan et al. (2010) used the price transition matrix to model the price transition over several hours. Let B represent the set of prices or states resulting in the cloud user being out bid. Let i equal the current market price and b equal the cloud users desired bid price. If the model predicts over a one-hour interval, then the probability the spot instance remains active is the sum of state transition probabilities not in set B (Chohan et al., 2010). Equation 6 provides the probability the market price remains at or below the bid price over n hours because Equation 6 calculates the price transitions over several steps.

$$P(i,b,n) = \sum_{j \notin B} M_{ij} P(j,b,n-1) \tag{6}$$

While Equation 6 calculates the probability for *n* hours, the cloud user requires an estimate for the mean spot instance duration for a bid. If the workload's duration is less than the mean spot instance duration, the workload completes without interruption greater then 50% of the time. Chohan et. al (2010) calculated the mean spot instance duration or expected lifetime based on the market price history length, τ, in hours, Equation 7.

$$mean\ lifetime = \sum_{n=1}^{\tau} n\, P(i,b,n) \tag{7}$$

Although expected lifetime provides insight to the cloud user's bid strategy, the cloud user also requires execution time and cost estimates to determine a bid price. The cloud user requires execution time estimates because the cloud user must compare the expected lifetime against the workloads execution time (Herodotou et al., 2011). Furthermore, the cloud user must estimate the early termination failure cost because the expected lifetime only provides the average spot instance duration (Chohan et al., 2010). While cloud users can select bids for their workloads to complete before the expected lifetime, some workload runs will contend with spot instance failures (Chohan et al., 2010).

Early Termination Costs

Terminated spot instances increase the workload's execution time because the workload requires rework of tasks from the failed instances. The rework not only includes partially completed tasks but also completed map tasks. The map tasks store the intermediate data on the local file system prior to being transported to the reduce tasks (Wang et al., 2009). When an instance fails, the failed node's scheduled tasks require re-execution because the reduce tasks lose intermediate data created by the map tasks (Wang et al., 2009).

Chohan, et al. (2010) approximates the early termination execution costs by estimating the time to detect the failure, δ, and the proportion of map tasks requiring rework. Equation 8 shows the failure cost in additional execution time, where s is the total number of instances, f is the number of failed instances, and M is the total time to complete the map phases (Chohan et al., 2010).

$$failure\ cost = \delta + (\frac{fM}{s})/(s - f) \tag{8}$$

However, the failure cost approximation only focuses on the impact for the failed map tasks because the impact on the reducers depends on the number of reduce waves (Chohan et al., 2010). Terminated spot instances impact the reduce phase's execution time because the surviving reduce tasks need to reshuffle the lost intermediate data from the failed map tasks (Wang et al., 2009). As seen in Figure 4, the reduce tasks transfer the intermediate data as the map tasks complete. When a node fails with completed map tasks, all the intermediate data must be recreated because reduce tasks in subsequent waves may not have transferred the intermediate data yet, Figure 8. However, the initial reduce tasks may not require the intermediate data because the initial reduce tasks transferred the intermediate data prior to the node failure (Chohan et al., 2010).

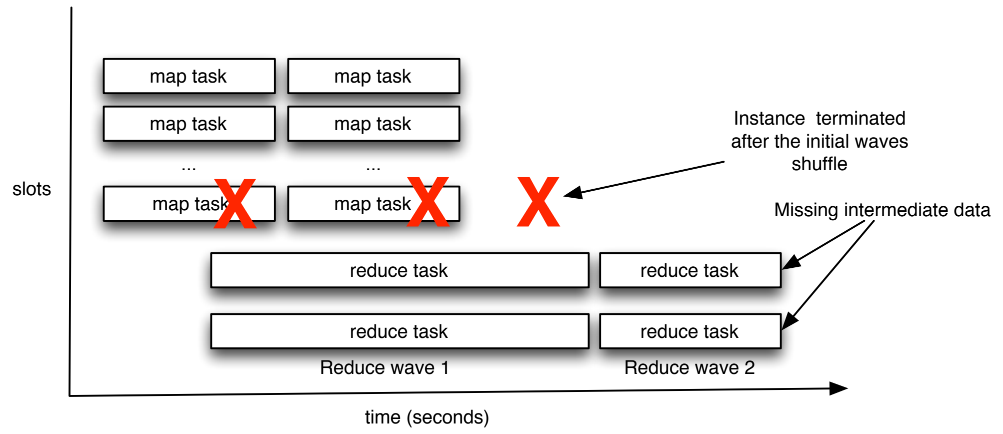

Figure 8. Intermediate Data Loss

While terminated spot instances impact the cloud user's completion deadline, the terminated spot instance increase the workloads monetary costs. Cloud users pay for spot instances in hour increments. When a spot instance terminates after an hour, the cloud user pays for the spot instances processing time and the additional execution time to recreate the lost work (Chohan et al., 2010). Therefore, the cloud user must assess the early termination execution time as well as the monetary costs before selecting a bid price.

In summary, the following statistical methods quantified the impact of preemptive auctions on a map-reduce workload's monetary costs and execution time: initial bid success probability (Orna Agmon Ben-Yehuda et al., 2011), expected lifetime, and execution time impacts from early termination (Chohan et al., 2010). The statistical methods aid cloud users because statistical methods predict the bid price's availability, spot instance duration for a given bid, and the workload's preemption costs. However, each statistical method requires the workload execution time estimate to determine the bid's impact on the workload's execution time and monetary cost (Herodotou et al., 2011).

Storage Approach

If the cloud user decides to use spot instances, then the provider presents the cloud user with an additional choice. The cloud user must select an alternative storage option because spot instances cannot host the map-reduce framework's corresponding distributed file system. Spot instances are unsuitable for hosting the distributed file system because the spot instance may terminate before providing access to the workloads results (Chohan et al., 2010). For instance, Amazon EC2 changed spot instance prices every 1.25 hours on average during a period from July 2010 to Feb 2011 (Orna Agmon Ben-Yehuda et al., 2011). If a workload executes in 1 hour, then the cloud user may only have 15 minutes or less to review the workloads results.

To avoid loss of the workload's results, prior research proposed the following storage alternatives for spot instances: Spot Instance Accelerators and Amazon S3.

Spot Instance Accelerators

Spot instance accelerators provide a method to process data with spot instances and persist the results after the instance terminates (Chohan et al., 2010). With the spot instance accelerator approach, the map-reduce cluster contains a combination of on-demand and spot instance virtual machines. The on-demand virtual machines are core nodes. The core nodes persist the workload's results because the core nodes host both the distributed file system processes and map-reduce tasks like a traditional map-reduce cluster node. The spot instance virtual machines are accelerator nodes. The accelerator nodes enable the spot instances to process data because the accelerator nodes host map-reduce tasks by providing additional map-reduce task slots.

The accelerator nodes impact the cloud user's ability to estimate the map-reduce execution time because the accelerator nodes increase the workload's average number of remote data requests. Although accelerators increase the number of task processing slots, the accelerators isolate the map tasks from the distributed file system (Chohan et al., 2010). When the map-reduce scheduler divides the input data to be processed, the scheduler assigns a map task to a corresponding data block stored in the distributed file system. The scheduler attempts to place the map task on the same file system node that contains the task's data block (Dean & Ghemawat, 2008). However, the accelerator's map tasks access their data blocks remotely because the accelerator node does not store the corresponding input data blocks (Chohan et al., 2010).

When the map-reduce scheduler assigns a map task to a remote node, the map task's read throughput decreases because the remote read requests increase network and disk contention. Although the map-reduce nodes provide network bandwidth greater then 1 Gb/sec, remote tasks contend with other remote tasks to access a given data node (Dean & Ghemawat, 2008). When the number of remote tasks increases, the task's effective bandwidth decreases because the remote tasks share the data node's network and disk bandwidth (Dean & Ghemawat, 2008).

The map phase's read throughput depends on the workload's average data locality. Data locality expresses the distance between a map task and the task's file system block. The map's read throughput depends on the locality type: data local, virtual machine local, rack local, or rack remote. Each locality type quantifies the distance the map read request travels and potential resource contention.

Data local task requests travel the shortest distance because the map task reads data from the same node. Node locality reduces network contention because the map task's data is read directly from the local disk. The map task's data request does not cause network interference (Zaharia et al., 2010). However, data local tasks create local disk contention because the local tasks share disk bandwidth (Zaharia, Konwinski, Joseph, Katz, & Stoica, 2008).

Virtual machine local requests travel the next shortest distance because the map tasks reads data from the same physical machine but different virtual machine instances (Li, Subhraveti, Butt, Khasymski, & Sarkar, 2012). Virtual machine local tasks connect to a remote data node via a virtualized network connection instead of using the physical network. Virtual machine local tasks reduce the physical network contention caused by the task because the data requests are channeled over a virtual network. However, co-located virtual machines compete for disk bandwidth. Although the impact varies depending on the virtual machine type, small instance types experience up to a 60% reduction in disk throughput (Zaharia et al., 2008).

Rack local requests travel one network hop because the map tasks access remote data nodes that are located on the same physical rack (Wang et al., 2009). Rack local tasks may cause localized network congestion and contention on remote data nodes because the remote data node on the same switch must service several remote data requests. The workload's execution time increased by 40% with rack local tasks because data request latency caused the map task's read phase to increase 5 times (Wang et al., 2009).

Rack remote requests travel the furthest because the map tasks access data across several network switches (Wang et al., 2009). Rack remote requests cause network congestion between the network switches because the map phase's data requests are concentrated on the connections between the racks. In the remote rack case, execution time increased by 284% with a double rack topology because latent data requests caused the map tasks' read phase to increase 175 times (Wang et al., 2009).

Although the individual task data locality impacts the map phase read throughput, map-reduce workloads contains a mixture of locality types. The map-reduce scheduler attempts to place the map tasks on the nearest tasks slot to the file system block. However, the map-reduce scheduler assigns tasks in a FIFO order and does not wait for a local slot to become available (Zaharia et al., 2010). When the workload's input data concentrates on a few data nodes or several jobs request data from the same node concurrently, the map-reduce workload contains a mixture of data local, rack local, and rack remote tasks (Zaharia et al., 2010).

When prediction tools estimate the workload's execution time, the tools must account for the mixture of locality types. To account for the mixture of local and remote map tasks, the Elastisizer measures the average map read time in the workload's profile. The average map read time characterizes the impact from workload's locality because the profiler measures the throughput for the distributed file system, which contains both local and remote reads (Herodotou, 2012). However, the profile does not include an explicit locality measurement because the cloud provider obfuscates the physical network topology (Herodotou, 2012). Therefore the profile does not quantify the percentage of data local, virtual machine local, rack local, and rack remote tasks.

Although the Elastisizer quantifies data locality impacts on the map execution time, the average read time measurement is flawed because the accelerator nodes change the workload's locality. Although the Elastisizer assumes all cost statistics fields remain constant for a given virtual machine type, the workload's average data locality impacts the distributed file system's average read time per byte (csDFSReadCost). The csDFSReadCost measures the average read time for both the data local and rack local tasks. When a map task executes on a rack local node, the task's read phase increases 5x compared with a node local task from a sort workload (Wang et al., 2009). When the cloud user adds accelerator nodes to a workload's configuration, the data to rack local task mixture changes because the map tasks assigned to the accelerator are not data local tasks (Chohan et al., 2010).

Given that accelerator nodes impact the map-reduce workload's locality and data locality impacts the workload's execution time, cloud users require an estimate of the impact on overall execution time caused by additional accelerator nodes. If additional accelerator nodes fail to proportionally speed up the execution time, then the cloud user receives diminishing marginal utility from the additional accelerators (Chohan et al., 2010).

Amazon S3

Amazon S3 provides another approach to access the workloads results after spot instances terminate (Chohan et al., 2010). Amazon S3 provides access to the map-reduce results after spot instances are terminated because Amazon S3 stores data on an external file system independent from the virtual machine instances. The cloud user accesses the workload's results via an HTTP interface after the job completes (Wieder et al., 2012).

Map-reduce workloads process the Amazon S3 data because S3 organizes the data in a block-oriented fashion, similar to HDFS. Amazon S3 stores the workload's input data using key-value pairs. The key is the filename and bucket or location of the file and the value is the content of the data (Bicer et al., 2011). Although standard file system clients interact with the file system via a HTTPS REST interface, map-reduce tasks access Amazon S3 as a block storage file system, similar to HDFS. Each map task accesses a block using HTTPS requests to the block. As with typical HTTP connections, multiple threads retrieve the blocks content in parallel chunks (Bicer et al., 2011).

Although Amazon S3 provides a similar block oriented access pattern, Amazon S3 increases the map-reduce workload's network overhead compared to HDFS. When a map-reduce workload uses HDFS, the map tasks are scheduled on the same node as the data being processed. When a map-reduce workload uses Amazon S3, map tasks must access data remotely. The remote access increases network overhead because the workload's tasks share network bandwidth to the remote storage system (Dean & Ghemawat, 2008).

Although Amazon obfuscates S3 proprietary details, past studies compared S3's throughput to HDFS's remote throughput. When a large EC2 instance uploads a 64 MB file, HDFS throughput was 20 MB/s compared to S3's 16 MB/s (Wieder et al., 2012). The test used m1.large instances because the large instance has greater bandwidth available to Amazon S3 compared to other instances, see Table 1 (Wieder et al., 2012).

Amazon S3 increases the map-reduce workload's network overhead compared to HDFS because Amazon S3's reads include additional HTTP and SSL overhead per data block request (Bresnahan, Keahey, LaBissoniere, & Freeman, 2011). The overhead's throughput impact depends on the requested data block's size. An S3 clone provided read throughput ranging from 10 MB/s for 2MB blocks to 310 MB/s for 512 MB blocks (Bresnahan et al., 2011). In a separate experiment, Amazon S3 provided a read throughput average 1MB/s for 1MB blocks and 17 MB/s for 100 MB blocks (Palankar, Iamnitchi, Ripeanu, & Garfinkel, 2008).

Although Amazon S3 impacts the workload's overhead, cloud users might consider Amazon S3 as an alternative to reduce monetary costs. Cost reductions are possible because Amazon S3 enables a cloud user to schedule the workload during non-peak hours when spot instance prices are low (Wieder et al., 2010). In essence, Amazon S3 enables the cloud user to analyze the workload's results after the workload completes and the virtual machine instances are terminated. However, the cloud user requires execution time and monetary cost estimates to evaluate Amazon S3 as an alternative (Wieder et al., 2010).

When the Elastisizer compares Amazon S3 workloads, the model requires new costs statistics for Amazon S3 backed virtual machines because the external file system changes the I/O resources available to the virtual machines (Herodotou et al., 2011). Although Amazon S3 reduces the distributed file system throughput, the external file system supplements the virtual machines' local I/O resources. As seen in Figure 4, the map-reduce tasks write to both the distributed file system and local disk. With HDFS based workloads, the local disks service requests for both intermediate data and HDFS file system blocks (Wang et al., 2009). When the workload uses Amazon S3, the local disks only processes intermediate data requests because Amazon S3 persists the distributed file system's data (Wieder et al., 2012).

Summary

The cloud user decides on the virtual machine type, number of virtual machines, pricing model, and storage to configure the workload's virtualized cluster. Each option impacts the workload's execution time and monetary costs. The literature review examined each option's impact on the workload. In addition, existing tools and techniques were examined to quantify each option's impact for a given workload. Although map-reduce cost estimation tools exist to evaluate the workload's execution time and monetary costs for a user, the past tools have not addressed two options: the pricing model and storage approach.

Chapter 3

Methodology

This dissertation achieved the dissertation goal by enhancing previous cost estimation tools to include virtualized map-reduce clusters using spot instances. The methodology includes the following enhancements: storage strategy specific virtual profiles, a failure-aware task simulator, and spot market forecasts. These enhancements improved the completion time and monetary cost prediction accuracy for map-reduce workloads using spot instances.

The approach first enhanced the estimation tool to include storage strategy specific virtual profiles. The enhanced tool requires specific virtual profiles because the storage strategy creates different cost statistics for both the on-demand and spot instance tasks (Lee et al., 2011). To account for the differences between on-demand and spot instance tasks, the tool will create new virtual profiles, which include spot instance nodes.

Additionally, the spot instance tasks impact the cost statistics because the remote tasks can reduce the workload's aggregate file system throughput. When a workload exceeds 72 nodes or 50% accelerator nodes, the workload generates enough remote data calls to cause file system congestion (Wang et al., 2009; Zaharia et al., 2010; Zaharia et al., 2008). To account for file system congestion, the virtual profile updates also model the cluster size and accelerator ratio's impact on the cost statistics.

When the storage strategy's profile enhancements were completed, the dissertation evaluated the enhanced tool's prediction accuracy over a series of workloads. The test workloads allocated various cluster sizes and spot instance ratios to measure the enhanced virtual profiles' prediction error as the workload adds spot instances. For each workload, the evaluation compared the original and enhanced tool's execution time prediction error to determine the accuracy improvements.

Although the profile updates improved the execution time estimate's accuracy in the absence of failures, the tool required task simulator updates to estimate the execution time with spot instance failures. When a node fails during the workload's execution, the workload requires extra processing time because the map-reduce framework must reschedule the workload's tasks to recreate the failed nodes' intermediate data (Wang et al., 2009). The updated task simulator calculated the extra processing time required given a specific failure time.

When the task simulator updates were completed, the dissertation evaluated the updated simulator's prediction error during spot instance failures. The evaluation tested spot instance failures during different times within a workload. For each failure time, the enhanced tool predicted the workload's execution time. The evaluation compared the simulator's predicted execution time against the workload's actual time to determine the prediction error. If the updated simulator reduced the prediction error compare to the original tool's estimates, then the enhanced tool successfully improved the prediction accuracy for terminated spot instance workloads.

When the tool completed the virtual profile and task simulator updates, the approach enhanced the tool to forecast the completion time and monetary costs based on the spot market history. The enhanced tool requires a forecast because a spot market auction controls the spot instances' availability to the workload (Stokely et al., 2009). The forecast contained the expected completion time and cost based on spot instance availability for a given bid price.

Once the forecast enhancements were completed, the dissertation evaluated the forecast's accuracy by comparing two traces from the same spot market. The first trace provides the input to compute the predicted values; and the second trace computes the actual values via simulation. The evaluation compared the predicted and actual values to measure the forecast's prediction error. If the forecast reduced the prediction error compared to the original tool's on-demand estimates, then the enhanced tool successfully improved the map-reduce workload's prediction accuracy for spot instances.

Storage Strategy Updates

The enhanced tool updated the virtual profile to handle both Amazon S3 and accelerator storage strategies. While the enhancements include a separate virtual profile for each storage strategy, the virtual profile enhancements also include relative fitness updates to model file system contention. Once the enhancements were complete, this study evaluated each enhancement to determine the prediction accuracy improvements over the original tool.

The methodological approach first updated the cost estimation tool to address the storage strategy because the workload's storage affects the virtual profile's cost fields. The updated estimation tool must predict the virtual profile's cost fields to determine the spot instance failure costs because the cost fields quantify the time to re-execute a failed task (Chohan et al., 2010). As previously noted, the Elastisizer's virtual profile models the individual map-reduce task's execution time and dataflow on a new virtual cluster. When the Elastisizer evaluates the workload on a new virtual machine type, a relative fitness model captures the changes to the workload's primitive operation response times as the virtual profile's cost statistics (Herodotou, 2012). A white-box model then converts the cost statistics to phase costs (i.e. execution times) based on the workload's dataflow (Herodotou, 2012).

Although the spot instances contain the same local resources as the on-demand instances, the storage approach created new spot instance specific relative fitness models. Both the spot instance storage strategies require new relative fitness models because the storage strategy impacts characteristics like the distributed file system read time (Herodotou & Babu, 2011). When the virtual profile models the Amazon S3 storage strategy, the Amazon S3 approach created one new relative fitness model per virtual machine type because the workload only allocates one node type per cluster.

Accelerator Virtual Profile

While the Amazon S3 workload allocates one virtual machine type per cluster, the accelerated workload allocates two different node types: core and accelerator nodes. The accelerated cluster requires an updated virtual profile because the underlying distributed file system contains fewer resources relative to the non-accelerate cluster. When Lee et al. (2011) benchmarked a cluster with 5 core and 4 accelerator nodes, the accelerator nodes increased execution time by 5% compared to a 9-core node cluster. The accelerated cluster increased the execution time because the accelerated nodes increased the map task's file system read costs compared to the pure core workload.

Although this study considered separate virtual profiles for the core and accelerator nodes, the task simulator used a single virtual profile for both node types because the each type used the same virtual resources. When this study executed TeraSort on an accelerated cluster with a 1:1 core to accelerator ratio, the average task completion time for accelerated and core tasks only varied by 4%. If the cluster workload generates HDFS contention, then the contention impacts both node types similarly because tasks on both node types wait for HDFS requests to be serviced form the core nodes (Wang et al., 2009).

Enhanced Relative Fitness Models

While the updated virtual profile quantifies the accelerated file system read and write costs, remote accelerator tasks increase file system contention. As seen in Figure 4, the map-reduce tasks generate network traffic during the map read, reduce shuffle, and reduce write operations. When the workload contains remote tasks, the remote operations cause congestion because the aggregated data transfers exceed the cluster's available bandwidth (Wang et al., 2009; Zaharia et al., 2008).

The remote tasks increase the workload's execution time because the tasks create both network and disk contention. Remote tasks increases execution time between 70% and 284% for 72+ node clusters because the tasks generate enough remote data calls to cause network congestion (Wang et al., 2009). However, remote tasks also impact execution time on smaller clusters because the remote tasks can cause disk contention. Remote task create disk contention because HDFS data nodes service both local and remote task requests (Wang et al., 2009). When the data node services five times the tasks, disk throughput reduces by 44% due to disk contention (Zaharia et al., 2008).

The virtual profile update modified the relative fitness model to predict virtual profile's cost statistics given the file system load. The relative fitness model includes the cluster size and spot instance percentage as predictor variables because the new features quantify the underlying file system's load created by the remote tasks, Figure 9.

When the relative fitness model covers utilization features, the utilization features improve the relative fitness models accuracy. Network utilization features improved a file system model's relative prediction error from 100% to 40% because the model's training set captured network utilization impact on the file system's available bandwidth (Mesnier et al., 2007).

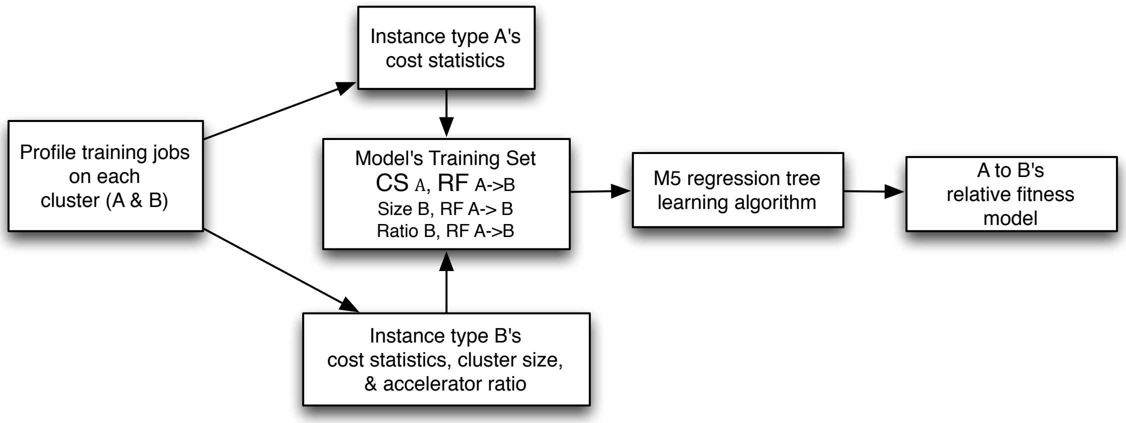
Instance type A's cost statistics
Profile training jobs on each cluster (A & B)
Model's Training Set
CS A, RF A->B
Size B, RF A-> B
Ratio B, RF A->B
M5 regression tree learning algorithm
A to B's relative fitness model
Instance type B's cost statistics, cluster size, & accelerator ratio

Figure 9. Enhanced Fitness Model Training Set

To capture the utilization features, the enhanced tool extended Herodotou's (2012) training set algorithm because the algorithm captures the cluster resource to relative fitness relationships. The training set algorithm generates the fitness model's training set from the source (A) and target (B) clusters' benchmark profiles. The original algorithm predicts the cost statistic changes between the source and target virtual machine types given the workload cost statistics on the source cluster. While the original algorithm only captures the source's cost statistics as predictor variables, the enhanced algorithm will add the spot instance percentage as a predictor variable, Figure 10.

```
Algorithm for Enhanced Training Set
Input: $Prof_A$ = Profile A, $Prof_B$ = Profile B, $Prof_B$.spot = Cluster B's Percentage of Spot Instances
Output: $RF_{A->B}$ Training Set
For each cost statistic, i, in table x
    For each cost statistic, j, in table x
        If $i \neq j$ then
          $WC_{A,i}$ = cost statistic i from $Prof_A$
          $P_{A,j}$ = cost statistic j from $Prof_A$
          $P_{B,j}$ = cost statistic j from $Prof_B$
          Write $WC_{A,i}$  $Prof_B$.spot => $P_{B,j}$/ $P_{A,j}$ to training set for $RF_{A->B,j}$
        End if
    End for
End For
```

Figure 10. Enhanced Training Set Algorithm

Storage Strategy Evaluation

Methodology Overview

The storage approach evaluated the virtual profile enhancements for both the Amazon S3 and Accelerator storage strategies. For each virtual profile enhancement, this study compared the enhanced and original tool's prediction accuracy. Both tools predicted the execution time against a representative set of workloads, Table 12. The evaluation calculated both tools' prediction error compared to the average actual execution. If the enhanced tool reduced the prediction error compared to the original tool for the given storage strategy, then the enhancement successfully met the dissertation goal.

Both storage strategy evaluations executed workloads in Table 12 because the workloads cover typical map-reduce workload characteristics. The grep (Dean & Ghemawat, 2008) workload represents an I/O intensive workloads because the workload avoids sorting data by running map-only jobs. The sort, join, TF-IDF and word count workloads contain a mixture of I/O and CPU tasks because the workloads index and/or sort data (Wang et al., 2009). The pi-estimator and co-occurrence contain mostly CPU-bound tasks because the workloads contain CPU intensive map tasks or sort intensive operations (Chohan et al., 2010; Herodotou, 2012).

While the workloads covered different characteristics, the evaluation required a sufficiently sized dataset for the target cluster. The workload's data sizes enable experiments up to 30 nodes because the data sets are large enough to occupy all the available map slots for the target systems (Herodotou, 2012). When the experiments evaluated disk and network utilization, the dataset size was tripled to fully occupy the map slots in a 90-node target system.

Table 12: Evaluation Workloads

Workload	Data Set	Properties	Study
Pi Estimator	>60KB of control data	CPU bound workload	(Chohan et al., 2010)
Grep	60GB synthetic	I/O bound, map only	(Dean & Ghemawat, 2008), (Wang et al., 2009)
Join	60GB data from the TPC-H Benchmark	CPU and I/O Mixture	(Herodotou, 2012), (Zaharia et al., 2010)
TF-IDF	60GB of documents from Wikipedia	CPU and I/O Mixture	(Verma et al., 2011), (Herodotou, 2012)
TeraSort	60GB synthetic workload	CPU and I/O Mixture	(Dean & Ghemawat, 2008), (Zaharia et al., 2008), (Wang, Butt, Monti, & Gupta, 2011), (Herodotou, 2012)
Word Count	60GB from Wikipedia	CPU and I/O Mixture	(Verma et al., 2011), (Herodotou, 2012)
Word Co-occurrence	10GB from Wikipedia	CPU bound workload	(Herodotou, 2012)

Although each experiment evaluated specific virtual profile enhancements, all experiments extended Herodotou et al. (2011) methodology for measuring the virtual profile's prediction accuracy. Each experiment computed the prediction error for a given workload, source cluster resources, and target cluster resources because the virtual profile predicts the workload's relative changes between a source and target cluster pair (Herodotou, 2012). Figure 11 enumerates the general procedure for the prediction accuracy comparison on a given cluster pair.

The procedure evaluated two different virtual profiles because the evaluation must demonstrate the enhancement's prediction accuracy improvement over the existing tool. Each experiment created a control and enhanced virtual profile. The original tool created the control virtual profile to quantify the baseline prediction error. The enhanced tool created the enhanced virtual profile to quantify the individual enhancement's prediction error.

```
**Procedure for comparing task simulator**
**Input:** W workloads, $C_s$ source cluster resources, $C_t$ target cluster resources
**Output:** $E_{control}$ control profile prediction error, $E_{enhance}$ enhance profile prediction error
Generate relative fitness models: $RF_{control}, RF_{enhanced}$
For each (job in W)
  Create baseline profile on $C_s$: $P_{baseline}$
  Predict execution time with $RF_{control}$ on $C_t$: $T_{control} = \mathcal{S}(P_{baseline}, RF_{control})$
  Predict execution time with $RF_{enhanced}$ on $C_t$: $T_{enhanced} = \mathcal{S}(P_{baseline}, RF_{enhanced})$
  Capture job mean execution time on $C_t$: $T_{actual}$
  Calculate the control profile prediction error: $E_{control} = \frac{T_{control} - T_{actual}}{T_{actual}}$
  Calculate the enhanced profile prediction error: $E_{enhanced} = \frac{T_{enhanced} - T_{actual}}{T_{actual}}$
End for
```

Figure 11. Virtual Profile Comparison Procedure

To calculate each profile's prediction error, the procedure executes the workload on EC2 virtual machines to determine the workload's actual execution time. While Herodotou (2012) executed the workloads on EC2 virtual machines, the Amazon EC2 experiments can experience execution time variance because the experiments are not performed in isolation (Schwarzkopf et al., 2012). The variance indicates contention caused by other multi-tenant workloads executed on the same cloud environment (Schad, Dittrich, & Quiané-Ruiz, 2010). When the procedure executed the workloads on EC2 nodes, the experiment conducted three trials to quantify the workloads' mean execution time.

Before each experiment evaluates the control and enhanced virtual profile, each virtual profile requires a corresponding relative fitness model. The original and enhanced tools generated relative fitness models by executing benchmark workloads on the source and target cluster resources. Each tool captures each benchmark's cost statistics to generate a training set to create the relative fitness model.

The benchmark workloads contain jobs that create tasks' different CPU, network, and disk characteristics. Although the benchmarks follows the Herodotou (2012) methodology to cover the test workloads' prediction space, the benchmarks executed the same tasks across the entire job to generate a necessary aggregate amount of file system contention. If each task requested different amounts of data, then the job's file system contention could not be quantified.

Amazon S3 Evaluation

Based on the general evaluation procedure, this study evaluated the Amazon S3 virtual profile's prediction accuracy to determine the enhanced profile's achievement of the dissertation goal. This dissertation created both HDFS and S3 virtual profiles to compare the original and enhanced tools' prediction accuracy. This dissertation evaluated cluster sizes ranging from 10 to 90 nodes because the workload may experience file system congestion at larger cluster sizes. If the S3 virtual profile reduced the average prediction error over the HDFS virtual profile on the different cluster configurations, then the virtual profile updates are successful.

The first experiment evaluated the Amazon S3 virtual profile updates on a 10-node cluster to determine the prediction accuracy improvements over the original virtual profile. The 10-node experiment provides a comparison point to Herodotou's (2012) experiments because the small fixed cluster size reduces the file system congestion's impact on the execution time (Zaharia et al., 2008). The 10-node experiment will follow the procedure in Figure 11 to calculate the control and enhanced virtual profiles' prediction error.

While the 10-node experiment required a fixed cluster size, the experiment evaluated several virtual machine types, Table 14. For each target virtual machine type, the experiment generated a relative fitness model, Table 13, because the tool predicts relative execution time changes between virtual machines. One relative fitness model covered a virtual machine type pair because the model determines the cost statistics' changes between a source and target virtual machine type (Herodotou, 2012). The experiment created the workload's baseline profile on the source type because the relative fitness model predicts the workload's changes on the target virtual machine (Herodotou, 2012). The experiment created the baseline profile on a 10-node m1.large cluster using HDFS on-demand nodes.

Table 13: Amazon S3 Evaluation Relative Fitness Models

RF Model	Source Type	Target Type	File System
1	m1.large	m1.large	HDFS
2	m1.large	c1.medium	HDFS
3	m1.large	m1.large	Amazon S3
4	m1.large	c1.medium	Amazon S3

The 10-node experiment created HDFS and Amazon S3 relative fitness models because the experiment followed the virtual profile comparison procedure, Figure 11. The HDFS and S3 RF models served as the control and enhanced fitness models, respectively. Once the experiment created the relative fitness model, the Amazon S3 experiment predicted each workload's execution time in Table 12 against the target cluster configurations in Table 14. The experiment also executed each workload against the target cluster configurations to determine the control and enhance models prediction error.

Table 14: Amazon S3 Accuracy Evaluation (Single Benchmark Run)

Experiment	HDFS RF Model	S3 RF Model	VM Type	Number of Nodes	File System
10 m1.large S3 single	1	3	m1.large	10	S3
10 c1.medium S3 single	2	4	c1.medium	10	S3

Although the 10-node experiment evaluated the relative fitness model's execution time accuracy, the S3 evaluation needed to evaluate the 10-node fitness model's scalability. The scale experiment reproduced Herodotou's (2012) scale experiment to determine the single 10-node benchmark's accuracy up to a 30 node target cluster. While the scale experiment evaluated a larger target cluster size, the scale experiment continued to follow the virtual profile comparison procedure, Figure 11. For each workload, the tool's simulator estimated the execution time for 10, 20, and 30 instances. The experiment compared the simulation results with the actual execution time to evaluate the simulation's accuracy as nodes are added to the workload.

Although the 10-node experiment evaluated two different virtual machine types, the scale experiments focused on file system throughput. The scale experiments evaluated two different instance types because the instance types contain different network bandwidth to Amazon S3 (Bicer et al., 2011). The experiments allocated target clusters with the c1.medium and m1.large instance types because the instances provide medium and high I/O throughput respectively.

Table 15: Amazon S3 Small Scale Accuracy Evaluation (Single Benchmark Run)

Experiment	HDFS RF Model	S3 RF Model	VM Type	Number of Nodes	File System
30 m1.large S3 single	1	3	m1.large	10,20,30	S3
30 c1.medium S3 single	2	4	c1.medium	10,20,30	S3

Once the 30-node experiment completed, the S3 evaluation extended the test cluster size to determine the enhanced fitness model's accuracy up to 90 nodes. The enhanced fitness model experiment evaluated cluster sizes up to 90 nodes because past map-reduce experiments demonstrated reduced network throughput after 72 nodes for non-local maps (Wang et al., 2009). This large-scale experiment tested the enhanced fitness model's accuracy improvements because the experiment compared the 10-node relative fitness models to a fitness model created with multiple cluster sizes, Table 16. The enhanced fitness models conducted benchmark runs at 30, 60, and 90 nodes because the enhanced model included the cluster size in the training set.

Table 16: Amazon S3 Large Scale Evaluation Relative Fitness Models

RF Model	Source Type	Target Type	Configurations
5	m1.large	m1.large	30 S3 nodes 60 S3 nodes 90 S3 nodes
6	m1.large	c1.medium	30 S3 nodes 60 S3 nodes 90 S3 nodes

As with the prior experiments, the large-scale experiment followed the procedure in Figure 11 to determine the control and enhance model's prediction accuracy. While the test workloads remained the same as the 10-node experiment, the large-scale experiment tripled the workloads' dataset size to enable sufficient coverage of the available map-reduce slots. For each workload, the control and enhanced fitness models estimated the execution time for configurations in Table 17. To determine the actual execution time, the experiment then executed the workload against the corresponding target configuration. When the experiment completed the workload's run, the experiment compared the simulation results with the actual execution time to evaluate the simulation's accuracy as nodes are added to the workload.

Table 17: Amazon S3 Large Scale Accuracy Evaluation (Enhanced Relative Fitness)

Experiment	HDFS RF Model	S3 RF Model	VM Type	Number of Nodes	File System
90 m1.large S3 enhanced	1	5	m1.large	30,60,90	S3
90 c1.medium S3 enhanced	2	6	c1.medium	30,60,90	S3

Accelerator Node Evaluation

While the Amazon S3 experiments evaluated the Amazon S3 enhancements' prediction accuracy, this study focused primarily on the accelerator enhancements' prediction accuracy. The dissertation evaluated the prediction accuracy improvement over the original tool for the accelerator's virtual profile, task simulator, and relative fitness model enhancements. The initial experiments evaluated the enhanced virtual profile and task simulator prediction accuracy at small scale. The evaluation then conducted accelerator ratio experiments to test the enhanced relative fitness model's prediction accuracy with file system contention.

The first experiment evaluated the accelerator virtual profile updates at small scale to determine the prediction accuracy improvements over the original virtual profile. Like the Amazon S3 experiment, the 10-node experiment provided a comparison point to Herodotou's (2012) experiments because the small fixed cluster size limited the file system congestion's impact on the execution time (Zaharia et al., 2008). The 10-node experiment followed the procedure in Figure 11 to calculate the control and enhance virtual profiles' prediction error.

Table 18: Accelerator Relative Fitness Models

RF Model(s)	Source Type	Target Type	Configuration
1	m1.large	m1.large	10 HDFS
2	m1.large	c1.medium	10 HDFS
7	m1.large	m1.large	5 HDFS, 5 accelerator
8	m1.large	c1.medium	5 HDFS, 5 accelerator

While the 10-node experiment required a fixed cluster size, the experiment evaluated accelerated clusters with two different virtual machine types, Table 19. For each target virtual machine type, the experiment generated a relative fitness model, Table 18, because the tool predicts relative execution time changes between virtualized resources. Although the core and accelerator nodes reside on the same virtual machine type, the experiment created a new relative fitness model because the updated virtual profile captured the cost statistics for the reduced number of data nodes. As with the

Amazon S3 experiments, the 10-node experiment created the baseline profile on a 10-node m1.large cluster using HDFS on-demand nodes.

Table 19: Accelerated HDFS Accuracy Evaluation (Single Benchmark Run)

Experiment	HDFS RF Model	Acc. RF Model	Configuration(s)
10-node m1.large	1	7	5 core, 5 accelerator
10-node c1.medium	2	8	5 core, 5 accelerator

The 10-node experiment created HDFS and accelerator relative fitness models because the experiment followed the virtual profile comparison procedure Figure 11. The HDFS and accelerator RF models served as the control and enhanced fitness models, respectively. Once the experiment created the relative fitness model, the accelerator experiment predicted each workload's execution time in Table 12 against the accelerated cluster configurations in Table 19. The experiment also executed each workload against the target cluster's configuration to determine the control and enhanced models' prediction error.

Similar to the Amazon S3 evaluation, the remaining experiments evaluated the various target cluster configurations. The next experiment reproduced Herodotou's (2012) scale experiment to determine the accelerated virtual profile's accuracy as the target clusters scales up to 30 nodes. The scale experiments executed each workload in Table 12 against the accelerated configurations in Table 20 to determine the accelerated virtual profiles accuracy as nodes are added. The scale experiment followed the procedure in Figure 11 to calculate each target cluster's control and enhanced profile's prediction error.

The scale experiment examined the c1.medium and the m1.large instance types because each instance type provides different I/O throughput to the core data nodes.

When the cluster contains accelerator nodes, the map tasks' processing time depends on the core node's I/O throughput because the map tasks read input from the shared core nodes. The c1.medium instance provides relatively poor file system throughput because c1.medium type allocates 5 virtual CPU units and provides "medium" I/O performance (Herodotou, 2012). In contrast, the m1.large instance provides relatively better file system throughput per CPU unit because the m1.large instance allocates 4 virtual CPU units and provides "high" I/O performance.

Table 20: Accelerated HDFS Accuracy Evaluation (Single Benchmark Run)

Experiment	HDFS RF Model	Accl. RF Model	Configuration(s)
30-node m1.large	1	7	5 core, 5 accelerator 10 core, 10 accelerator 15 core, 15 accelerator
30-node c1.medium	2	8	5 core, 5 accelerator 10 core, 10 accelerator 15 core, 15 accelerator

Although the scale experiment provided a comparison point to Herodotou et al. (2011) results, the scale experiment did not evaluate accelerator ratio's impact on the tools' prediction accuracy. While additional nodes can impact prediction accuracy, the accelerator ratio also impacts prediction accuracy because the accelerator tasks can reduce file system throughput. Accelerated tasks reduce file system throughput because the non-local tasks can cause disk and or network contention (Wang et al., 2009; Zaharia et al., 2010).

The accelerator ratio experiment examined the accelerator ratio's impact on the control and enhanced profile's prediction accuracy. The accelerator ratio experiment generated the relative fitness models in Table 21 to cover the various ratios. The selected ratios extend Chohan et al. (2010) spot instance study to cover a core to accelerator ratio greater then 1 to 1. Once the relative fitness models are generated, the accelerator ratio experiment executed each workload in Table 12 against the accelerated configurations in Table 22. The ratio experiment followed the procedure in Figure 11 to calculate each ratio's control and enhanced profile's prediction error.

Table 21: Enhanced Relative Fitness Models

RF Model	Source Type	Target Type	Benchmark Runs
1	m1.large	m1.large	10 HDFS
2	m1.large	c1.medium	10 HDFS
9	m1.large	c1.large	20 core, 10 accelerators 15 core, 15 accelerators 10 core, 20 accelerators 5 core, 25 accelerators
10	m1.large	c1.medium	20 core, 10 accelerators 15 core, 15 accelerators 10 core, 20 accelerators 5 core, 25 accelerators

While map-reduce locality studies focused on cluster sizes ranging from 72 to 100 nodes (Wang et al., 2009; Zaharia et al., 2010), the variable ratio experiments were conducted with 30 nodes because the small cluster produced data node or disk contention. Wang et al. (2009) experiments required 72 nodes because the study focused on network congestion cause by the rack configuration. The 30-node experiments were sufficiently sized to validate data node contention because the data node must service remote requests for multiple tasks (Zaharia et al., 2008).

Table 22: 30-Node Accelerated HDFS Accuracy Evaluation (Enhanced Relative Fitness)

Experiment	HDFS RF Model	Accl. RF Model	Configuration(s)
30-node m1.large	1	9	20 core, 10 accelerators 15 core, 15 accelerator 10 core, 20 accelerator 5 core, 25 accelerator
30-node c1.medium	2	10	20 core, 10 accelerators 15 core, 15 accelerator 10 core, 20 accelerator 5 core, 25 accelerator

Task Simulator Failure Updates

Although the virtual profile updates addressed issues related to the storage strategy, the task simulator required updates to address spot market early termination. The Elastisizer's task simulator assigns tasks to map-reduce slots based on the time the slot is ready (Herodotou, 2012). When the spot market terminates an instance early, the task scheduler can't assign tasks to the spot instance's slots. Additionally, the task simulator reschedules the completed map tasks because the reduce tasks can't retrieve the spot instance's intermediate data (Wang et al., 2009).

To simulate intermediate data loss, this dissertation updated the task simulator to failover spot instance slots, Figure 12. While the original task simulator does not check for failed slots, the updated task simulator checks for failures because the actual task scheduler doesn't assign new tasks to the failed slots (Wang et al., 2009). If the accelerator instance fails before the new slot's ready time, then the simulator will not add the slot back to the slot queue. The updated simulator also tracks the tasks assigned to accelerator slots because the simulator needs to reassign the failed slots' tasks.

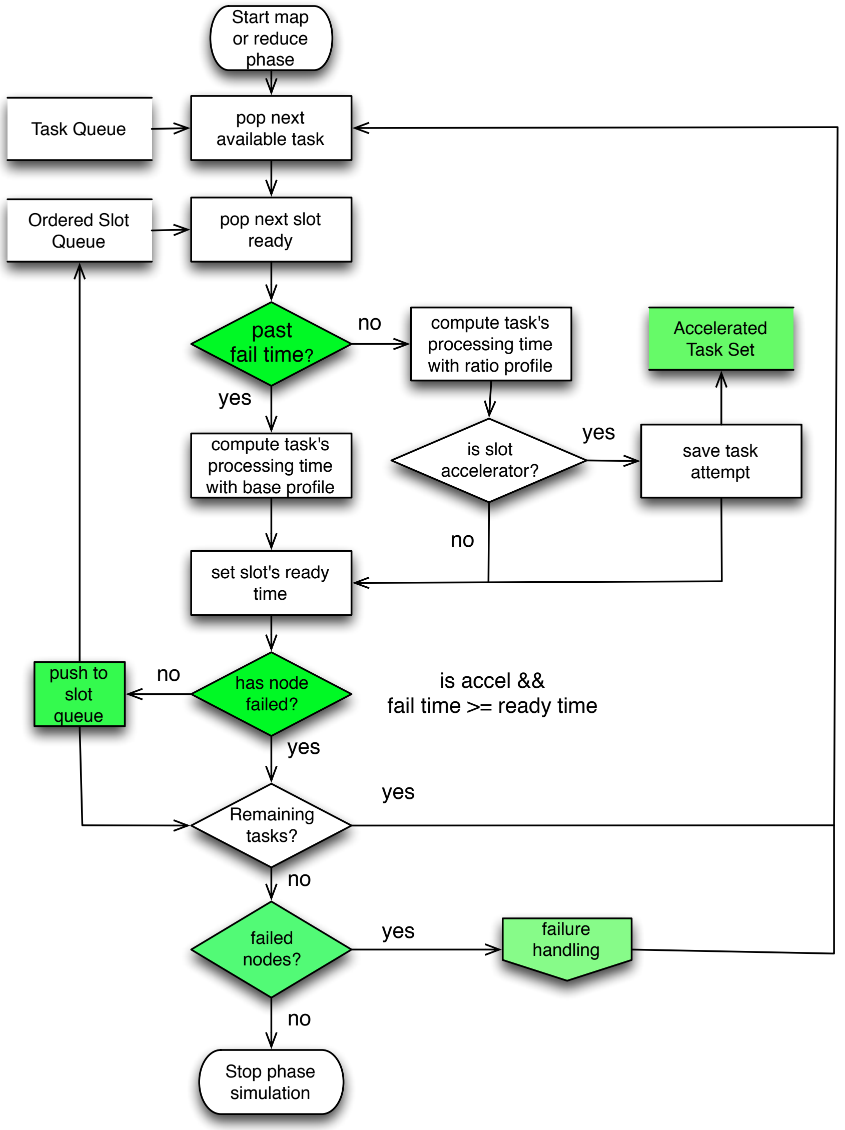

Figure 12. Task Simulator Failure Updates

When the simulator assigns all the initial tasks, the updated simulator reassigns the failed tasks, Figure 12. The updated simulator mimics the actual task scheduler's error handling, Figure 13. Before the simulator reschedules the failed instances' tasks, the simulator removes accelerator slots in the slot queue because the actual scheduler doesn't reassign tasks to failed slots. The simulator also updates the slot ready times to a heartbeat after the failure time because the real task scheduler doesn't detect the node failures immediately (Chohan et al., 2010).

Once the updated task simulator prepares the slots' ready times, the simulator reassigns the failed spot instances' tasks. If the failed slot is a map slot, then the simulator reschedules all tasks assigned to the slot because the slot lost the map task's intermediate data (Wang et al., 2009). If the failed slot is a reduce slot, then the simulator only reschedules incomplete tasks because the reduce tasks don't store intermediate data (Wang et al., 2009).

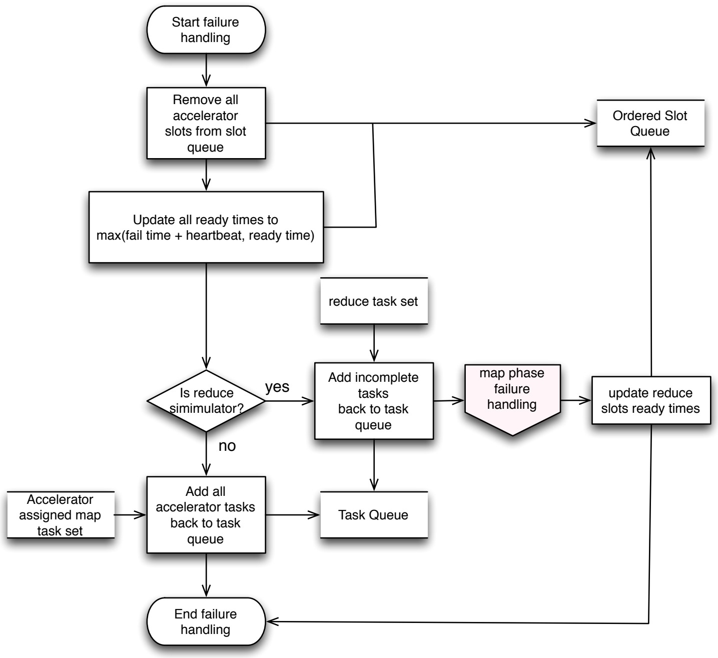
Start failure handling
Remove all accelerator slots from slot queue
Ordered Slot Queue
Update all ready times to max(fail time + heartbeat, ready time)
reduce task set
Is reduce simimulator?
yes
Add incomplete tasks back to task queue
map phase failure handling
update reduce slots ready times
no
Accelerator assigned map task set
Add all accelerator tasks back to task queue
Task Queue
End failure handling

Figure 13. Task Simulator Failure Handling

When the spot market terminates instances during the reduce phase, the failure also impacts the core node's reduce tasks. If a node fails during the reduce phase, then the map-reduce scheduler must reschedule the spot instance map tasks because the failed node's intermediate data is lost (Wang et al., 2009). If an active reduce task has not completed the shuffle, then the reduce task must wait for the rescheduled map tasks' to access the missing intermediate data (Wang et al., 2009).

The updated task simulator reschedules incomplete reduce tasks. The task simulator calculates the non-overlapping shuffle start time by tracking the time the last map task completes (Herodotou, 2012). When a failure occurs during the reduce phase, the updated task simulator resets the last map complete time for reduce tasks that have not completed the shuffle prior to the failure.

While the task simulator handles failed tasks, the task simulator must also adjust to different cluster resources because the accelerator ratio changes when the failure occurs. The task simulator uses two virtual profiles to account for the clusters configuration before and after the failure. Before the auction terminates the accelerator node, the simulator calculates the task completion time with the accelerated virtual profile. Once the auction terminates the accelerator node, the simulator calculates the remaining tasks completion time with the base HDFS profile.

In summary, the updated task simulator mimics the actual task scheduler's failure handling. The updated task simulator removes the accelerator slots from the slot queue when a failure is detected. When the workload completes the initial tasks, the simulator reschedules the failed tasks. If the workload fails during the map phase, then the simulator reschedules the failed node's map tasks. If the workload fails during the reduce phase, the task simulator reschedules the incomplete reduce tasks and the failed node's map tasks.

Task Simulator Evaluation

Methodology

While the accelerator profile experiments focus on the updated virtual profile's prediction accuracy, the task simulator experiments evaluates the enhanced tool's prediction accuracy during terminated workloads. The simulator experiments calculate the enhanced and original tools' prediction error given different failure times. If the enhanced tool predicts the execution time with a smaller average prediction error, then the updated task simulator satisfies the dissertation goal.

To calculate each task simulator's prediction accuracy, the simulator experiments followed the terminated workload procedure, Figure 14. While the simulator experiment followed the same process as the virtual profile experiment, Figure 11, the simulator experiments evaluated the prediction accuracy for a given failure time, t_{fail}. The experiment evaluated failures during four different times within the workload, Table 23. Each time corresponded to a different map-reduce wave because each wave exhibits different map-reduce phase data dependencies (Wang et al., 2009).

```
Procedure for comparing task simulators
Input: W workloads, $C_s$ source cluster resources, $C_t$ target cluster resources, $S_{control}$ control simulator, $S_{enhanced}$ enhance simulator, $t_{fail}$ failure time
Output: $E_{control}$ control profile prediction error, $E_{enhanced}$ enhance profile prediction error
Generate relative fitness models: $RF_{control}$, $RF_{enhanced}$
For each (job in W)
  Create baseline profile: $P_{baseline}$
  Predict execution time with $S_{control}$ with on $C_t$: $T_{control} = \mathcal{S}(P_{baseline}, RF_{control})$
  Predict execution time with $S_{enhanced}$ on $C_t$: $T_{enhanced} = \mathcal{S}(P_{baseline}, RF_{enhanced}, t_{fail})$
  Capture job execution time on $C_t$ with failure at $t_{fail}$: $T_{actual}$
  Calculate the control simulator prediction error: $E_{control} = \frac{T_{control} - T_{actual}}{T_{actual}}$
  Calculate the enhanced simulator prediction error: $E_{enhanced} = \frac{T_{enhanced} - T_{actual}}{T_{actual}}$
End for
```

Figure 14. Task Simulator Comparison Procedure for Terminated Workloads

The simulator experiments calculated the enhanced and original tools' prediction error for each failure time in Table 23. The HDFS and enhanced RF models served as the control and enhanced fitness models, respectively. Once the experiment created the relative fitness model, the simulator experiment predicted each workload's execution time in Table 12 against the target cluster configurations in Table 23. The experiment also executed each workload with a simulated failure to determine the control and enhance simulators prediction error.

While the simulator experiment evaluated spot instance failures during the workload, the simulator experiment examined the accelerator ratio's impact on prediction accuracy. The simulator experiments covered the same accelerator configuration as the non-failure accelerator ratio experiments, Table 22. The accelerator experiment configurations created different failure costs because each accelerator ratio corresponded to a different number of failed tasks (Chohan et al., 2010). In essence, the failure costs are proportional to the number of failed accelerator nodes.

Table 23: Failure Simulations

	Relative Fitness Model			
Experiment	HDFS	Accelerated	Configuration(s)	Failure time
m1.large	1	9	20 core, 10 accelerators	1st map wave
			15 core, 15 accelerators	2nd map wave
			10 core, 20 accelerators	1st reduce wave*
				2nd reduce wave**
c1.medium	2	10	20 core, 10 accelerators	1st map wave
			15 core, 15 accelerators	2nd map wave
			10 core, 20 accelerators	1st reduce wave*
				2nd reduce wave**

*Note: Some workloads do not include reduce phases. * excludes grep, ** excludes grep and pi*

Task Failure Simulation

While the simulator experiments configured a target cluster with different accelerator ratios, the simulator experiments also required the accelerators to fail at given times. When the experiment conducted the actual runs, the experiment simulated the spot instance termination by programmatically killing the task tracker daemon on the accelerator nodes. The killed task tracker simulated a node failure because the map-reduce scheduler lost contact with the node (Wang et al., 2009).

Table 24: Failure Timing Measures

Measure	**Description**
μ_{map}	Average map task completion time
μ_{reduce}	Average reduce task completion time excluding the shuffle phase
μ_{suffle}	Average shuffle time excluding the first shuffle wave
s_{job}	Start of the job
e_{maps}	End of last map task

For each workload, the experiment conducted non-failure runs to calculate the task trackers' kill time. The non-failure runs provided the mean task and phase times in Table 24. Although the task completion times vary, the mean task times indicated each map reduce wave's approximate time (Verma et al., 2011). If the experiment killed the daemons in the wave's center, then the kill times, then Equation 9 and Equation 10 represent t_{map} and t_{reduce}.

$$t_{map} = (w_{map} - 1)\times \mu_{map} + \mu_{map}/2 + s_{job} \tag{9}$$

$$t_{reduce} = (w_{red} - 1)\times \mu_{red} + \frac{\mu_{red}}{2} + (w_{red} - 1)\times\mu_{shuffle} + e_{map} \tag{10}$$

Once the experiment estimated failure run kill times, the experiment caused the task trackers to fail. Before the experiment submitted the test workload, the experiment started the kill daemons on the accelerator nodes. If the experiment targets the failure during the map wave, the kill daemons watch the map-reduce log directory to determine the jobs start. When a new job is added to the log directory, the kill daemon starts a timer to kill the task tracker processes at the given kill time.

While the map kill daemons determined the job's start from the log directory, the reduce kill daemons required the last map's completion time to start the timer. The kill daemon monitored the map-reduce job tracker. Once the last map task completed, the kill daemon started a timer to kill the task tracker.

When the failure job completed, the experiment validated the daemons killed the task tracker during the appropriate wave. If the task tracker died at the appropriate time, then several task attempts should have failed. When the task tracker fails during a given map wave, the number of failures should be $k_{accel} \times w_{map}$ where k_{accel} is the number of accelerator slots (Chohan et al., 2010). When the task tracker fails during a given reduce wave, then the previous waves reduce tasks should have completed without failure. Although the workload completes some reduce tasks prior to the failure, the kill daemons should cause all accelerator map tasks and one reduce wave's tasks to fail (Wang et al., 2009).

Spot Instance Forecast

Once the simulator updates are completed, the dissertation created a spot instance forecast tool. Although the Elastisizer only predicted the workload's execution time, the enhanced tool predicted the spot instance workload's average completion time and average monetary costs. The enhanced tool created the forecast with a spot market history trace to determine the spot market's impact on the workload's completion time and monetary costs.

While cloud providers allocate on-demand instances immediately, cloud users wait for a spot market auction to allocate the spot instances. Unlike on-demand workloads, the spot instance workload's completion time includes the auction wait time and the workload's execution time (Herodotou, 2012). Once the auction allocates the workload, the auction may increase the workload's execution time because the auction may terminate the spot instances before the workload completes (Chohan et al., 2010). When a cloud user allocates a series of spot instance workloads, the workload's expected completion time includes the expected wait time and the expected execution time, Equation 11.

$$E[comp] = E[wait] + E[exec] \tag{11}$$

While the execution time depends on the workload, the forecast tool predicts spot instance availability based on the bid price and the spot market history (Orna Agmon Ben-Yehuda et al., 2011). For a given bid price, *b*, the forecast tool will compute the average wait time. The forecast tool divides the price history into *N* intervals based on transitions where the market price rises or falls below the bid price. If $time_t$ equals the trace's total duration and $time_i$ equals an intervals duration, then the probability a job is submitted during a given interval, *p(submitted$_i$),* equals *$time_i/time_t$*. Based on the submission probability, the tool calculates expected wait time at the bid price, *b*, using Equation 12 and Equation 13.

$$E[wait] = \sum_{i=1}^{N} p(submitted_i) \times time_i/2 \times \mathrm{F}(i) \tag{12}$$

$$F(i) = \begin{cases} 0 \; if \; price_i \leq b \\ 1 \; if \; price_i > b \end{cases} \tag{13}$$

Once the forecast tool calculates the average wait time, the forecast tool must predict the workload's expected execution time because the spot market may terminate the spot instance's before the workload completes. The forecast tool computes the probability the spot instance terminates before the workload completes. For a given bid price, *p(terminates|b*) is the probability the spot instance terminates during a time interval. The forecast tool divides the price history into time intervals. If the bid price equals *b* and the interval's initial market price is less then *b*, then the auction allocates the spot instance at the interval's start (Chohan et al., 2010). If the interval's market price rises above the bid price before the interval's completion, then the auction terminates the spot instance. The interval's termination probability is Equation 14.

$$p(terminates|b) = \frac{\sum intervals\ terminated}{\sum intervals\ intially\ allocated} \tag{14}$$

The forecast tool also calculates the probability the spot instances remain active for the workload's duration. Let τ time intervals equal the workload's non-preemptive execution time. If the market price is independent of the bid price and *p(active|b) = 1 –*

p(terminates|b), then the probability the workload completes after τ time intervals without termination is Equation 15 (Chohan et al., 2010).

$$p(completes|b) = p(active|b)^{\tau} \tag{15}$$

Once the forecast tool calculates spot instance termination probabilities, the updated tool predicts the workload's expected execution time, Equation 16. Given *x* on-demand and *y* spot-instance nodes, the tool predicts the workload's non-preemptive execution time, *time(x,y)*. When the auction terminates spot instances at time interval *n*, the tool also predicts the workload's execution time, *time(n,x,y).* Given the set of possible execution times, the workload's expected execution time is the weighted average of execution times.

$$E[exec] = \frac{p(comp|b) \times time(x,y) + \sum_{n=1}^{\tau} p(active|b)^{n-1} \times p(term|b) \times time(n,x,y)}{p(comp|b) + \sum_{n=1}^{\tau} p(active|b)^{n-1} \times p(term|b)} \tag{16}$$

While the enhanced tool forecasts the workload completion time, cloud users require workload monetary cost estimates. When the cloud user allocates a map-reduce cluster with spot instance accelerators, the cloud user pays for the time the on-demand and spot instances are active. If the spot instance fails, the cloud user pays for the extra processing time to re-create the intermediate data (Chohan et al., 2010). Let *x* equal the number of on-demand nodes and *y* equal the number of spot instance nodes. If the auction terminates the spot instance at time *n*, then the workload's charged time equals the on-demand node's total processing time plus the time the spot instances are active, Equation 17.

$$cost(n,x,y) = time(n,x,y) \times x \times core_{price} + \lfloor n \rfloor \times y \times accel_{price} \tag{17}$$

If the workload completes without early termination, then the cloud user pays for both instance types over the workload's duration, τ, Equation 18.

$$cost_{np} = \tau \times x \times core_{price} + \tau \times y \times accel_{price} \tag{18}$$

Based on Equation 17 and Equation 18, the forecast tool can computes the expected cost for the workload at bid price *b*. The expected cost is a weighted average of the workload's costs with and without failures, Equation 19.

$$E(cost) = \frac{p(comp|b) \times cost_{np} + \sum_{n=1}^{\tau} p(active|b)^{n-1} \times p(term|b) \times cost(n,x,y)}{p(comp|b) + \sum_{n=1}^{\tau} p(active|b)^{n-1} \times p(term|b)} \tag{19}$$

In summary, the forecast tool updates include the expected completion time and expected cost. The workload's forecast enables the cloud user to select the configuration with the best average cost or completion time. Additionally, the cloud user can select the cluster based on multiple objectives (Herodotou, 2012). For instance, the cloud user can indicate their delay tolerance by specifying a desired expected completion time (Chun & Culler, 2002). Although the user wants to minimize the workload's average cost with spot instances, the cloud user may constrain the spot instance and bid price selection based on minimizing the completion time.

Forecast Tool Evaluation

When this study completed the forecast tool update, the forecast experiments evaluated the enhanced tool's prediction accuracy for completion time and monetary cost.

The forecast experiment calculated the enhanced and original tool's prediction error given a spot market trace. If the enhanced tool predicted the completion time and monetary costs with a smaller average prediction error, then the updated task simulator satisfied the dissertation goal.

The evaluation followed Chohan et al. (2010) trace-based methodology to assess prediction accuracy. The trace-based methodology compares two traces from the same spot market to determine the prediction algorithm's accuracy. The first trace provides the input to compute the predicted values. The second trace provides the market prices to apply the workload against. The evaluation assesses the prediction accuracy by comparing the predicted values with the actual values (Chohan et al., 2010).

Although the split trace assesses prediction accuracy, the spot market mechanics must remain constant for the trace method to be valid (Orna Agmon Ben-Yehuda et al., 2011). Cloud providers can change spot market mechanics by refining the auction algorithm (Orna Agmon Ben-Yehuda et al., 2013). Therefore, the evaluation selected traces with distinguishable market epochs where the same pricing patterns are observed over the entire trace (Orna Agmon Ben-Yehuda et al., 2013). Wieder et al. (2012) study combined historical and synthetic traces to validate the approach works given different market conditions.

This study considered two recent market history traces to predict and validate the tool's forecasts. Although the traces covered roughly the same time period, each trace represents different demand patterns exhibited in the Agmon Ben-Yehuda et al. (2011) and Wieder et al. (2012) studies. The first trace represented, Figure 15, frequent price transitions near the reserve price found in the Agmon Ben-Yehuda et al. (2011) studies.

The first trace captured market conditions for the c1.medium us-east-1c spot market from January 26, 2014 to February 16, 2014. The c1.medium trace creates frequent failures at low bids because the low bids are constantly outbid once allocated (Chohan et al., 2010).

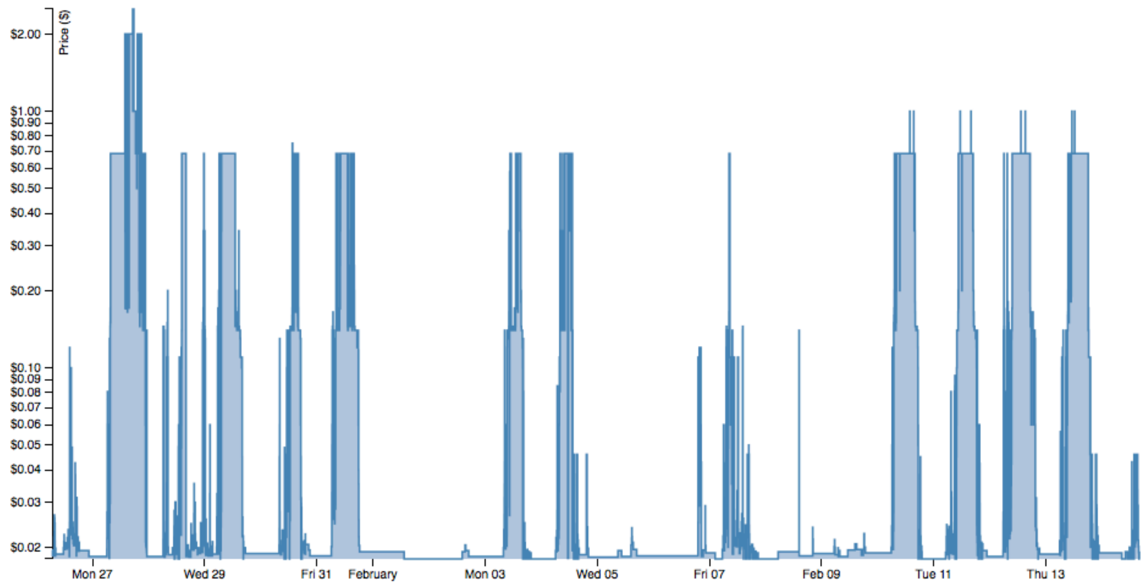

Figure 15. C1.Medium US-East-1C Market Price History

The second trace, Figure 16, provided similar behaviors to Wieder et al. (2012) synthetic trace because the trace exhibited regular demand patterns. The second trace captured market conditions for the m1.large us-east-1a spot market from January 05, 2014 to January 26, 2014. Although the m1.large trace is an actual trace, the m1.large trace displays regular low activity periods during the weekends and holidays. The m1.large trace should be easier to predict the spot instance's availability because the demand pattern is less bursty than the c1.medium trace.

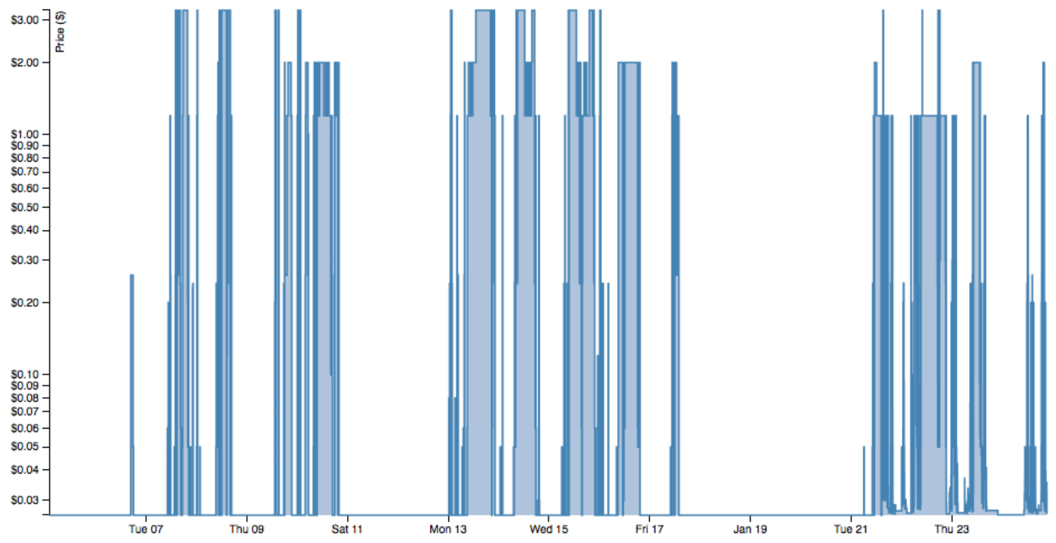

Figure 16. M1.Large US-East-1A Market Price History

While both traces display some similar characteristics to past studies, both traces contain a key difference to the (Chohan et al., 2010), Agmon Ben-Yehuda et al. (2011), and Wieder et al. (2012) studies. The current traces are different because both traces contain market prices up to 10 times greater than the on-demand price. The previous studies assumed the market price was capped at the demand price. Contrary to (Chohan et al., 2010) assumptions, the cloud user can't assume a bid at the on-demand price will guarantee a spot instance.

The experiment evaluated the forecast tool's completion time and financial cost prediction accuracy. The forecast tool experiments followed the procedure in Figure 17 to apply the trace. For each spot market trace, the experiment calculated each tool's prediction errors. The control tool predicted the workload's completion time and cost without the trace because the Elastisizer didn't factor in spot market conditions (Herodotou, 2012). The enhanced tool forecasted the workload's completion time and cost using the forecast trace. Once each tool predicts the workload's completion time and costs, the experiment simulated the actual completion time and monetary costs with the actual trace to determine each tool's prediction error.

```
Forecast Evaluation Procedure
Input: $R$ Trace Replays, $B$ bid prices, $W$ workloads, $C_t$ target cluster resources, $S_{control}$ control tool, $S_{enhanced}$ enhanced tool
Output: $E_{control}$ control tool's prediction errors, $E_{enhanced}$ enhanced tool's prediction errors
For each (r in R)
  Let $r_{forecast}$ = the forecast trace, $r_{actual}$ = the actual trace
  For each (job in W)
      For each($b$ in B)
        Predict the ave. cost and ave. completion time (CT) with $S_{control}$ on $C_t$
        Forecast the ave. cost, ave. CT, and max CT with $S_{enhanced}(b,r_{forecast})$ on $C_t$
        Simulate the actual ave. cost, ave. CT, and max CT at $b$ with $r_{actual}$
        Calculate $E_{control}$ for ave. cost, ave. CT, and max CT
        Calculate $E_{enhanced}$ for ave. cost, ave. CT, and max CT
      End for
  End for
End For
```

Figure 17. Forecast Tool Evaluation Procedure

Similar to prior experiments, the forecast experiment evaluated different target cluster configurations because each configuration provided different failure probabilities, wait times, and failure costs (Chohan et al., 2010). The forecast experiment evaluated c1.medium and m1.large instance types because the forecast experiment reused the virtual profiles from the accelerator experiments, Table 21. The original and enhanced tools predicted the average completion time, maximum completion time, and average cost for each configuration in Table 25 using the first part of each trace.

Table 25: Bid price experiments

	Relative Fitness Model			
Experiment	HDFS	Accelerated	Data Size	Configuration(s)
m1.large forecast	1	9	1.2 TB	20 core, 10 accelerators 15 core, 15 accelerators 10 core, 20 accelerators
c1.medium forecast	2	10	1.2 TB	20 core, 10 accelerators 15 core, 15 accelerators 10 core, 20 accelerators

To validate the forecast's accuracy for a given bid, the forecast replayed the second trace to calculate the workload's wait time and termination. Liu (2011) executed market price trace replays to determine the spot instance's allocation and termination points. While Liu's (2011) study selected a submission time for his evaluation, the forecast experiment selected random times to submit the workload in a trace replay, Figure 18. Given the bid price and the random submit time, the second trace provided the spot instances' wait time, *w*, and the termination point, *n*.

```
Workload Submission Monte Carlo Simulation
Input: r actual trace, b bid price, time(x,y) non-preemptive execution time
Output: costs = trail costs, times = trial completion times
For n from 0 to m trials
  cost = $cost_{np}$; exec = time(x,y)
  $t_{submit}$ = Random time between r.start and r.end
  w = 0;   $t_{start} = t_{submit}$;
  if b < r.marketPrice($t_{submit}$)
      w = time to the next price transition where $b \geq r.marketPrice(t)$
  end if
  $t_{start} = t_{submit} + w$
  if b < r.minMarketPrice($t_{start}$, $t_{start}$ + exec)
     n = time to the fatal price transition
     cost = cost(n,x,y); exec = time(n,x,y)
  end if
  costs.add(cost); times.add(w + exec)
End for
```

Figure 18. Workload Submission Monte Carlo Simulation

Although the trace replay provided the workload run's wait time and termination points, the experiment calculated the run's completion time and monetary cost. The experiment calculated the run's completion time using Equation 20 where $time(n, x, y)$ equals the simulated execution time. The experiment also calculated the run's monetary cost using Equation 17.

$$comp = w + time(n, x, y) \tag{20}$$

When the experiment completed the individual workload runs, the experiment calculated the workload's average completion time, maximum completion time, and average cost from the trace replays. The evaluation then compared the forecast estimates with the trace replay results to compute the forecast's average and maximum completion time prediction error, Equation 21.

$$error_{comp} = \frac{comp_{pred} - comp_{sim}}{comp_{sim}} \tag{21}$$

The evaluation also calculated the monetary cost prediction error using the simulated cost from the spot instance trace, Equation 22.

$$error_{cost} = \frac{cost_{pred} - cost_{sim}}{cost_{sim}} \tag{22}$$

While the prediction error measures the forecast tool's accuracy, the evaluation must compare the enhanced tools' forecast against the existing tool's predictions to determine the forecast's accuracy improvements. The evaluation reused the HDFS control predicted execution times from the accelerator experiments, Table 22, for the bid price experiment. The experiment then computed the control tool's predicted monetary cost with the non-preemptive cost equation, Equation 18, because the original tool's cost predictions assume no spot instance failures (Herodotou, 2012). The evaluation computed the original tool's prediction error based on the simulated results, Equation 21 and Equation 22.

Summary

When a cloud user executes a map-reduce workload, the user must specify the virtualized resources to execute the workload within a desired budget and or timeframe. To execute a map-reduce workload, the cloud user must decide upon the virtual machine type, number of virtual machines, pricing model, and storage options. The average user requires estimation tools to help evaluate the options because the virtualized cluster's allocation requires specific expertise to estimate the workload's completion time and monetary costs (Herodotou & Babu, 2011).

The cloud user may consider the spot market pricing model as a means to reduce the workload's monetary cost. Spot instances can reduce a workload's monetary cost because a preemptive auction allocates the spot instances (Chohan et al., 2010). When the cloud user bids on spot instances during low demand periods, the auction offers the spot instances at a fraction of the on-demand price. However, the cloud user requires an estimate for the workload's completion time and monetary cost because the auction determines the instances start time and duration (Chohan et al., 2010).

Although existing estimation tools predict on-demand virtual instance's completion time and costs, the on-demand estimates are unsuitable for spot market based workloads because the spot market auction determines the virtualized instances' availability to process data. The spot market may extend the workload's completion time because the spot market auction may delay the workload's start (Stokely et al., 2009). The spot market may also increase the workload's execution time and, by extension, monetary cost because the auction may terminate the spot instances before the workload completes (Chohan et al., 2010).

This study created an enhanced cost estimation tool to forecast the spot market map-reduce workloads' completion time and monetary cost. The enhanced tool improves the prediction accuracy over current tools because the enhanced tool estimates the unreliable spot instances' impact on the workload. The estimates include the workload's increased execution time caused by external storage and early spot instance termination. Given the spot market's history and cloud user's bid price, the enhanced tool forecasts the workload's average completion time and monetary cost based on the spot instance's availability.

When this study completed the external storage and termination enhancements, the study compared the original and enhanced tool's prediction accuracy. The study followed Herodotou's (2012) methodology to determine each tool's prediction error. Given a target workload and cluster configuration, each tool predicts the workload's execution time. The evaluation then executed the workload on the target cluster to determine the workload's actual execution time. Based on the actual and predicted execution times, the evaluation calculated each tool's prediction error. If the enhanced tool reduced the prediction error over the original tool, then the enhanced tool successfully improved the workload's prediction accuracy.

While Herodotou's (2012) methodology evaluated execution time, this dissertation also evaluated the enhanced tool's completion time and monetary cost forecast accuracy. This dissertation followed Chohan et al. (2010) trace-based methodology to determine the forecast tool's prediction accuracy. The evaluation split the spot market price history into forecast and actual traces. The forecast trace provided the market price history to predict the workload's completion time and monetary costs.

The evaluation then executed a Monte Carlo simulation over the actual trace to calculate the workload's average completion time and monetary costs.

When the evaluation completed Monte Carlo simulation, the evaluation compared the original and enhanced tools' prediction accuracy based on the simulation's results. For each workload, the evaluation computed each tool's prediction error. If the enhanced tool reduced the prediction error over the original tool, then the enhanced tool successfully improved the workload's prediction accuracy.

Chapter 4

Results

Cloud users require a new means to evaluate map-reduce workloads on spot instances because the existing tools don't predict completion time and monetary costs accurately. As noted in the problem statement, past tools do not address two decision factors when a cloud user allocates a virtual cluster: the storage strategy and the pricing model. Cloud users cannot compare storage strategies because past tools, like the Elastisizer, inaccurately models each new storage approach's impact on the workload's virtual profile. Cloud users cannot compare pricing models because the Elastisizer inaccurately simulates the spot market's impact on completion time and monetary costs.

This dissertation created an enhanced cost estimation tool, MapReduce Workload Allocation Tool for Spot instances (MRWATS) to address the storage strategy and price model decision factors. MRWATS improves upon the Elastisizer because the new tool models spot instance storage strategies and price changes. Specifically, MRWATS includes three enhancements to improve completion time and monetary cost prediction accuracy over the Elastisizer: storage strategy specific virtual profiles, a failure-aware task simulator, and spot market forecasts.

This chapter will first analyze MRWATS' storage strategy enhancements. MRWATS models each storage strategy (Amazon S3 and Accelerated HDFS) with a storage specific relative fitness model to improve execution time prediction accuracy over the Elastisizer. The new models contain two improvements over the Elastisizer: enhanced training benchmarks and contention predictor variables. The new training benchmarks improve prediction accuracy because the benchmarks characterize each specific file system. The additional predictor variables improve prediction accuracy because the new features model the workload's file system contention. The new features include cluster size and spot instance ratio to quantify file system contention.

The storage strategy data analysis will demonstrate that each storage strategy enhancement reduces the average prediction error over the Elastisizer. While the Amazon S3 and Accelerated HDFS storage strategies will be evaluated independently, both storage strategies follow the same structure. The data analysis section will first demonstrate the improvement from the storage specific training benchmarks by comparing a fixed size/ratio model to the Elastisizer's relative fitness models. Next, the data analysis section will demonstrate the new predictor variables' improvement by comparing variable size/ratio models to the Elastisizer's relative fitness models. If the enhanced models reduce the average execution prediction error compared to the Elastisizer's model, then the individual enhancement met the dissertation goal.

The results chapter will next analyze the failure-aware task simulator enhancements. While MRWATS's enhanced profiles improved the workload's non-failure prediction accuracy, MRWATS also improved upon the Elastisizer by enhancing the task scheduler to handle terminated spot instances. When the spot market terminates the instances early, the enhanced task simulator estimates the time to recover missing intermediate data from the terminated spot instances. In contrast, the Elastisizer under-predicts the terminated workload's execution time because it does not account for the recovery time.

The failure-aware task simulator data analysis will demonstrate that the task simulator enhancements reduce the prediction error compared to the Elastisizer. To validate the enhance task simulator improvements; the evaluation terminates workloads during four different map-reduce waves. For each termination point, the enhanced (MRWATS) and original (Elastisizer) task schedulers predict the workloads' execution times. The data analysis then compares the prediction error for both task schedulers. If the enhanced task scheduler reduces the average execution prediction error compared to the Elastisizer's task scheduler, then the task scheduler enhancements met the dissertation goal.

The result section will finally analyze MRWATS's forecast tool enhancements. Although the enhanced task simulator improves a terminated workload's prediction accuracy, the cloud user requires completion time and cost estimates without foresight into the exact termination time. MRWATS further improves upon the Elastisizer because MRWATS estimates the average completion time and monetary cost based on the spot market price history. While the Elastisizer simply assumes completion times remain constant and costs increase linearly with the bid price, MRWATS factors in wait times, early terminations, and past market prices into the completion time and cost estimates.

The forecast tool data analysis will demonstrate MRWATS's market-based enhancements improve upon the Elastisizer using two actual price histories to cover different spot market behaviors. For each price history, the evaluation split the history into training and test traces. The training trace provided MRWATS a history to predict the test workloads' average completion time and monetary costs. After both tools generated the predictions, the test trace provided a history to calculate the test workloads' actual average completion time and monetary costs. The trace analysis then compared the predicted and actual values for each workload to quantify MRWATS's prediction accuracy improvements over the Elastisizer. The analysis then breaks down the test results into low and high bid regions to highlight each specific MRWATS enhancement.

While the results will demonstrate accuracy improvements for each enhancement, the enhancements have a cumulative effect on MRWATS's prediction accuracy. Each enhancement provides an additional improvement over the last. The findings section will summarize each enhancement's improvements over the Elastisizer.

Storage Strategy Data Analysis

The storage strategy was the first enhancement evaluated because the other enhancements depend on accurate spot instance storage models. The Elastisizer primarily assumed the distributed file system scaled as compute nodes are added to the clusters because the new compute nodes contained additional file system resources (Herodotou, 2012). While a map-reduce cluster scales with new on-demand nodes, the spot instances won't scale the same as on-demand nodes because the spot instances don't contain additional file system resources to avoid data loss (Chohan et al., 2010).

MRWATS improved upon the Elastisizer for spot instance workloads because MRWATS created enhanced relative fitness models for both the Amazon S3 and HDFS accelerated file systems. The new relative fitness models served two purposes: to capture file system specific cost statistics and to quantify file system contention impacts on the cost statistics. The new models capture more accurate cost statistics by running file system specific training benchmarks. The models also quantify file system contention by measuring the workload's cluster size and spot instance ratio impacts on the I/O cost statistics.

Although the evaluation analyzed each file system separately, each evaluation followed the same general approach. The evaluation first compared MRWATS's and the Elastisizer's models built from benchmarks with a fixed cluster size and accelerator ratio to determine the accuracy improvements from storage specific benchmarks. MRWATS should at least provide the same average prediction error as the Elastisizer with the fixed models. The evaluation then compared MRWATS and the Elastisizer with different cluster configurations to determine the accuracy improvements from contention adaptive benchmarks. To meet the dissertation goal, MRWATS should reduce the average prediction error compared to the Elastisizer given the different cluster configurations.

Amazon S3 Virtual Profile Enhancements

MRWATS created two different enhancements to improve the Amazon S3 workloads' execution time prediction accuracy over the Elastisizer's original HDFS model. The first enhancement simply created the relative fitness models from Amazon S3 specific benchmarks because Amazon S3 contains different IO characteristics than HDFS (Bicer et al., 2011). The second enhancement added the cluster size as a predictor variable to the relative fitness model because remote tasks increase disk and network contention at large clusters sizes (Wang et al., 2009; Zaharia et al., 2008).

Amazon S3 Fixed-Size Relative Fitness Model

The first enhancement offers an improvement over the Elastisizer's original HDFS model because the Amazon S3 specific model characterizes the Amazon S3 file system on a fixed-size cluster. The fixed-size Amazon S3 model reduces workload prediction error because MRWATS builds the relative fitness model from training benchmarks against the actual file system resources. The S3 benchmarks primarily improve Amazon S3 bound workloads because the new benchmarks provide more accurate read and write cost statistics.

To validate the fixed-size Amazon S3 model's accuracy improvements over the HDFS model, the data analysis compares the simple Amazon S3 and HDFS models' accuracy against Amazon S3 workloads. The fixed-size analysis will demonstrate the accuracy improvements on clusters between 10 and 30 nodes. The analysis will compare relative fitness models generated from benchmarks on a 10-node virtual cluster, Table 13. The analysis will then scale up the comparisons to 30 nodes to demonstrate the fixed-size models scalability, Table 15.

The fixed-size S3 model demonstrated an improvement over the Elastisizer's HDFS model at 10-nodes because the new relative fitness model captured Amazon S3 specific I/O characteristics. While the fixed-size model reduced the average prediction error by 22.62% (from 19.54 to 15.12 percent error) across all workloads, the fixed size model perform better with I/O bound workloads, Table 26. For instance, the fixed model reduced grep's prediction error by 36.97% because grep spends a majority of its processing time on Amazon S3 reads.

Although the S3 relative fitness model improved the prediction accuracy for most workloads, the Amazon S3 model performed worst for the word co-occurrence workload at small-scale, Table 26. The Amazon S3 model over-predicted the co-occurrence's execution time because the workload spends the majority of the execution time on sort operations. In essence, the external file system does not impact the co-occurrence workload's performance because the workload is CPU bound.

The fixed-size model also offered an improvement over the Elastisizer because the new model captures the instance type impact on file system utilization. The m1.large S3 model reduced the prediction error by 31.4% compared to the c1.medium model's 7.35% reduction. The m1.large model performed relatively better than the c1.medium model because the m1.large workloads generated a larger load on Amazon S3. The m1.large instances executed more map tasks per node. Where as the c1.medium cluster allocated only 20 map slots, the m1.large cluster allocated 30 map slots.

Table 26: Fixed-Size Amazon S3 Model's Prediction Error

	Execution Time (sec.)			% Prediction Error	
Experiment	S3 Actual	HDFS Pred.	S3 Fixed Pred.	HDFS Model	S3 Fixed Model
S3 c1.medium small					
Co-Occurrence	1057.61	1275.67	1396.28	20.62	32.02
Grep	193.48	121.58	139.93	37.16	27.68
Pi Est.	486.09	459.51	529.72	5.47	8.98
TDF-IDF	2718.73	2223.11	2369.60	18.23	12.84
TPCH Q12	401.62	447.10	429.09	11.33	6.84
TPCH Q14	357.30	351.90	362.43	1.51	1.43
TeraSort	1888.77	1576.65	1577.02	16.53	16.51
Word Count	2183.66	1970.10	2097.76	9.78	3.93
S3 m1.large small					
Co-Occurrence	1216.29	1241.53	1298.92	2.07	6.79
Grep	215.40	101.63	156.72	52.82	27.24
Pi Est.	597.40	709.81	703.67	18.82	17.79
TF-IDF	2853.93	2306.95	2589.76	19.17	9.26
TPCH Q12	384.87	327.19	336.98	14.99	12.44
TPCH Q14	339.36	231.03	243.61	31.92	28.21
TeraSort	1577.09	962.44	1205.18	38.97	23.58
Word Count	2444.27	2122.42	2289.16	13.17	6.35
			Average Error	19.54	15.12

Note. The Amazon S3 model (MRWATS) was generated with 10-node training benchmarks against an Amazon S3 file system. The HDFS model (Elastisizer) was generated with 10-node training benchmarks against HDFS.

While the 10-node cluster analysis demonstrated the fixed-size model improved prediction accuracy with a light-load, the fixed-size model still offers an improvement over the HDFS model for slightly larger cluster sizes. If the additional nodes don't start to saturate Amazon S3's resources, then the fixed-size model should generate relatively more accurate cost statistics than the Elastisizer's HDFS model. The fixed-size model improves upon the Elastisizer's HDFS model because the fixed-sized model benchmarks Amazon S3's file system resources.

To validate the fixed-size model scales up to 30 nodes, the evaluation conducted 20 and 30-node experiments. The 20 and 30-node experiments compared the same virtual profiles created for the 10-node experiment on slightly larger clusters. If the fixed-size S3 model scales similarly to the HDFS models, then the fixed-size S3 model should still reduce the prediction error compared to the Elastisizer's HDFS model. The experiments tested the fixed-size models scalability for both the c1.medium and m1.large instance types.

For the c1.medium clusters, the fixed-size model demonstrated improved prediction accuracy compared to the Elastisizer's HDFS model because the fixed-size model captures the load created by the additional map tasks against Amazon S3. The c1.medium fixed-size model reduced the average prediction error by 9.58% (from 14.51 to 13.12 percent error) across the 10, 20, and 30 node clusters, Table 27. Although the cluster's size increased to 20 and 30-nodes, the fixed-size model remained more accurate than the Elastisizer's HDFS model. The fixed-size model reduced the average prediction error by 14.34% and 7.34% compared to the HDFS model, for the 20 and 30 node clusters respectively.

Table 27: C1.Medium Fixed-Size Amazon S3 Model's Accuracy

	Execution Time (Sec.)			% Prediction Error	
Node Count	S3 Actual	HDFS Pred.	S3 Fixed Pred.	HDFS Model	S3 Fixed Model
Co-Occurrence					
10	1057.61	1275.67	1396.28	20.62	32.02
20	621.34	646.45	706.41	4.04	13.69
30	388.03	482.89	530.07	24.45	36.60
Grep					
10	193.48	121.58	139.93	37.16	27.68
20	98.66	66.79	75.97	32.30	23.00
30	76.08	48.53	54.64	36.21	28.17
Pi Estimate					
10	486.09	459.51	529.72	5.47	8.98
20	292.19	281.22	323.32	3.75	10.65
30	206.87	192.08	220.12	7.15	6.40
TeraSort					
10	1888.77	1576.65	1577.02	16.53	16.51
20	932.21	825.19	844.14	11.48	9.45
30	623.85	515.44	539.89	17.38	13.46
TF-IDF					
10	2718.73	2223.11	2369.60	18.23	12.84
20	1380.39	1173.24	1247.65	15.01	9.62
30	990.78	783.64	833.15	20.91	15.91
TPCH Q12					
10	401.62	447.10	429.09	11.33	6.84
20	229.77	246.32	236.80	7.21	3.06
30	174.52	167.20	160.88	4.19	7.81
TPCH Q14					
10	357.30	351.90	362.43	1.51	1.43
20	229.83	214.51	220.73	6.67	3.96
30	176.15	145.81	149.88	17.23	14.91
Word Count					
10	2183.66	1970.10	2097.76	9.78	3.93
20	1093.55	991.86	1055.68	9.30	3.46
30	739.18	661.99	705.37	10.44	4.57
			Average Error	14.51	13.12

Note. The Amazon S3 model (MRWATS) was generated with 10-node training benchmarks against an Amazon S3 file system. The HDFS model (Elastisizer) was generated with 10-node training benchmarks against HDFS.

Similar to the 10-node results, the fixed-size S3 model improvements also depended on the instance type. The m1.large model performed better than the c1.medium model because the m1.large workloads generated a larger load on Amazon S3. The m1.large workloads create more concurrent file system requests than the c1.medium workloads because the m1.large instances allocate 3 map slots per node versus the c1.medium node's 2 map slots.

For the m1.large clusters, the fixed-size model demonstrated improved prediction accuracy compared to the Elastisizer's HDFS model because the fixed-size model captures the load created by the additional map tasks against S3. The m1.large fixed-size model reduced the average prediction error by 29% (from 25.9 to 18.39 percent error) across the 10, 20, and 30 node clusters, Table 28. The m1.large fixed-size model remained more accurate than the Elastisizer's HDFS model as the cluster sizes increase. The fixed-size model reduced the average prediction error by 28.35% and 27.48% compared to the HDFS model, for the 20 and 30 node clusters respectively.

Table 28: M1.Large Fixed-Size Amazon S3 Model's Accuracy

		Execution Time (Sec.)			% Prediction Error	
Node Count		S3 Actual	HDFS Pred.	S3 Fixed Pred.	HDFS Model	S3 Fixed Model
Co-Occurrence						
	10	1216.29	1241.53	1298.92	2.07	6.79
	20	652.83	629.84	658.61	3.52	0.88
	30	485.36	422.24	441.45	13.00	9.05
Grep						
	10	215.40	101.63	156.72	52.82	27.24
	20	169.35	56.82	84.36	66.45	50.19
	30	102.19	45.61	66.27	55.37	35.15
Pi Est.						
	10	597.40	709.81	703.67	18.82	17.79
	20	360.65	361.81	358.82	0.32	0.51
	30	282.02	361.81	358.82	28.29	27.23
TeraSort						
	10	1577.09	962.44	1205.18	38.97	23.58
	20	738.17	506.67	644.78	31.36	12.65
	30	748.16	339.08	432.40	54.68	42.21
TF-IDF						
	10	2853.93	2306.95	2589.76	19.17	9.26
	20	1553.72	1195.07	1334.78	23.08	14.09
	30	1058.22	872.74	978.37	17.53	7.55
TPCH Q12						
	10	384.87	327.19	336.98	14.99	12.44
	20	226.17	185.29	190.42	18.08	15.81
	30	162.9	149.72	155.77	8.09	4.38
TPCH Q14						
	10	339.36	231.03	243.61	31.92	28.21
	20	189.49	138.83	146.61	26.73	22.63
	30	154.07	100.56	106.48	34.73	30.89
Word Count						
	10	2444.27	2122.42	2289.16	13.17	6.35
	20	1635.09	1068.53	1151.92	34.65	29.55
	30	868.96	749.91	808.25	13.70	6.99
				Average Error	25.90	18.39

Note. The Amazon S3 model (MRWATS) was generated with 10-node training benchmarks against an Amazon S3 file system. The HDFS model (Elastisizer) was generated with 10-node training benchmarks against HDFS.

Amazon S3 Variable-Size Relative Fitness Model

While the fixed-size model enhancements improved Amazon S3 based-workloads' prediction accuracy below 30-nodes, the fixed-size model does not quantify the file system utilization's impact on the workloads execution. File system utilization impacts workload execution times at cluster sizes above 72 nodes because the remote task cause both network and disk contention (Wang et al., 2009). Although the fixed-size model captures the Amazon S3 cost statistics at 10-nodes, the model requires benchmarks at larger sizes to model potential network and disk contention caused by a given workload.

The second enhancement improves upon the Elastisizer's original HDFS model because the variable cluster-size model captures file contention on clusters above 30 nodes. The variable-size model quantifies file system contention because the model adds the cluster size as a training feature, Figure 10. If the workload's operations take longer at larger cluster sizes, then the cluster-size feature should capture the change in the relevant cost statistics.

To validate the variable-size model improves upon the Elastisizer, large-scale experiments compared the variable-size and HDFS relative fitness model's prediction accuracy up to 90 nodes. While the experiments create the Elastisizer's HDFS model from a 10-node benchmark, the variable-size models were generated from benchmarks on multiple clusters, Table 16. The experiments assessed the prediction accuracy at 30, 60, and 90 nodes because the larger size clusters tended to generate more file system contention (Wang et al., 2009).

The variable-size model demonstrated an improvement over the Elastisizer's HDFS model with cluster sizes between 30 and 90 nodes. The variable-size model reduced the average prediction error by 34.56% across all workloads and instance types. The variable-size model improved the prediction accuracy over the Elastisizer because the cluster-size dependent model was build from a more representative training set. For a deeper analysis, the large-scale results are broken down by instance type: c1.medium and m1.large.

For the c1.medium instance type, the variable size model improved the prediction accuracy compared to the Elastisizer's HDFS model because the training set covers the prediction space better above 30 nodes. Table 29 displays the prediction error by workload and node count for the c1.medium instance type. If the variable-size model produced a smaller prediction error than the Elastisizer's HDFS model, then MRWATS successfully improved the prediction accuracy for workload's using Amazon S3 on the c1.medium instance type.

Although the variable-size model did not reduce the prediction error on every c1.medium workload tested, the variable-size model improved the prediction accuracy compared to the Elastisizer for the majority of the workloads. The variable-size model reduced the prediction error in 19 out of 24 test configurations and the average prediction error for 6 out of 8 workloads, Table 29. The variable-size model reduced the c1.medium average prediction error by 35.39% (from 25.23% to 16.30% error) across all workloads, Table 29.

Table 29: C1.Medium Variable-Size Amazon S3 Model's Accuracy

	Execution Time (Sec.)			% Prediction Error	
Node Count	Actual	HDFS Pred.	S3 VS Pred.	HDFS Model	S3 VS Model
Co-Occurrence					
30	1052.01	1419.78	1487.94	34.96	41.44
60	581.13	719.37	766.34	23.79	31.87
90	416.54	518.07	546.41	24.38	31.18
Grep					
30	199.65	121.66	148.52	39.07	25.61
60	123.09	66.83	79.85	45.71	35.13
90	110.75	48.55	56.96	56.16	48.57
Pi Estimate					
30	520.93	395.98	545.45	23.99	4.71
60	307.02	243.32	322.16	20.75	4.93
90	218.35	166.99	212.28	23.52	2.78
TeraSort					
30	1850.60	1640.22	1732.91	11.37	6.36
60	1130.83	859.45	980.60	24.00	13.28
90	843.91	538.73	647.77	36.16	23.24
TF-IDF					
30	1372.09	1228.25	1287.39	10.48	6.17
60	766.40	679.27	702.57	11.37	8.33
90	573.84	460.40	467.29	19.77	18.57
TPCH Q12					
30	518.22	359.04	535.20	30.72	3.28
60	245.30	226.79	324.91	7.54	32.45
90	193.00	149.59	207.23	22.49	7.38
TPCH Q14					
30	386.80	258.08	379.05	33.28	2.00
60	261.92	158.04	229.57	39.66	12.35
90	202.36	108.02	154.67	46.62	23.57
Word Count					
30	1046.67	1006.10	1091.98	3.88	4.33
60	565.88	509.94	552.49	9.88	2.37
90	383.61	360.48	388.93	6.03	1.39
			Average Error	25.23	16.30

Note. The Amazon S3 variable size (VS) model (MRWATS) was generated with 30, 60, and 90-node training benchmarks against an Amazon S3 file system. The HDFS model (Elastisizer) was generated with 10-node training benchmarks against HDFS.

While the c1.medium analysis demonstrated one instance type's improvement, the analysis also covered the m1.large instance type because the m1.large instances allocate additional map slots. Table 30 displays the prediction error by workload and node count for the m1.large instance type. If the variable-size S3 model produced a smaller prediction error than the Elastisizer's HDFS model, then MRWATS successfully improved the prediction accuracy for workload's using Amazon S3 on the m1.large instance type.

The m1.large instance type results shows similar improvements to prediction accuracy compared to the c1.medium instance type. The variable-size model improved the prediction accuracy compare to the Elastisizer for the majority of the m1.large workloads. The variable-size S3 model reduce the prediction error in 20 out of 24 test configurations and the average prediction error for 6 out of 8 workloads. In aggregate, the variable-size S3 model reduced the m1.large average prediction error over the HDFS model by 33.88% across all workloads.

Table 30: M1.Large Variable-Size Amazon S3 Model's Accuracy

		Execution Time (Sec.)			% Prediction Error	
Node Count		Actual	HDFS Pred.	S3 VS Pred.	HDFS Model	S3 VS Model
Co-Occurrence						
	30	1329.18	1412.14	1425.41	6.24	7.24
	60	690.04	715.26	967.48	3.65	40.20
	90	631.37	458.33	509.99	27.41	19.22
Grep						
	30	254.98	101.15	113.61	60.33	55.44
	60	201.93	56.58	80.68	71.98	60.05
	90	91.55	45.43	63.51	50.37	30.63
Pi Est.						
	30	671.54	710.32	750.82	5.78	11.81
	60	388.16	362.32	389.70	6.66	0.40
	90	299.71	362.32	389.71	20.89	30.03
TeraSort						
	30	1477.92	957.66	1224.91	35.20	17.12
	60	863.46	504.19	733.09	41.61	15.10
	90	677.21	337.17	479.95	50.21	29.13
TF-IDF						
	30	1503.88	1199.13	1237.64	20.26	17.70
	60	879.51	701.39	763.30	20.25	13.21
	90	563.42	445.91	470.05	20.86	16.57
TPCH Q12						
	30	563.25	299.69	680.57	46.79	20.83
	60	309.34	170.49	327.65	44.88	5.92
	90	233.03	137.45	255.31	41.02	9.56
TPCH Q14						
	30	455.11	349.31	505.07	23.25	10.98
	60	263.48	204.95	251.56	22.22	4.52
	90	219.65	142.99	174.14	34.90	20.72
Word Count						
	30	1246.01	1060.12	1097.83	14.92	11.89
	60	668.71	594.80	651.64	11.05	2.55
	90	589.43	367.87	446.52	37.59	24.24
				Average Error	29.93	19.79

Note. The Amazon S3 variable size (VS) model (MRWATS) was generated with 30, 60, and 90-node training benchmarks against an Amazon S3 file system. The HDFS model (Elastisizer) was generated with 10-node training benchmarks against HDFS.

In summary, MRWATS includes two enhancements to the Elastisizer's relative fitness models that improve Amazon S3 workloads' prediction accuracy: Amazon S3 fixed-size and variable-size relative fitness models. The first enhancement improves upon the Elastisizer's relative fitness models because the fixed-size model explicitly captures the Amazon S3 file system cost statistics via Amazon S3 specific benchmarks. Although the fixed-size model primarily improves the I/O bound workloads' prediction accuracy, the fixed-size model reduced the average prediction error by 28.23% across multiple workloads executed on two instance types.

The second enhancement improves upon the Elastisizer's relative fitness models because it models Amazon S3 file system contention as the cluster size increases. The new variable-size model generates the workload's virtual profile based on training benchmarks across multiple cluster sizes. While the Elastisizer assumes the file system cost statistics remain constant as the cluster size increases, the variable-size model predicts the workload's contention as more nodes concurrently access the Amazon S3 file system.

The variable-size model improved upon the Elastisizer's fixed-size model because workload contention increased the I/O bound workloads' execution time. The variable-size model reduced the average prediction error by 34.56% across multiple workloads executed on clusters up to 90 nodes. For heavily I/O bound workloads, the variable-size model reduced the prediction error compared to the Elastisizer's HDFS model by as much as 76%.

Accelerator Virtual Profile Enhancements

Like the Amazon S3 storage strategy, MRWATS created two different enhancements to the Elastisizer's original relative fitness model. The first enhancement simply created a fix-ratio relative fitness model built from an accelerated cluster because an accelerated cluster allocates different resources to the distributed file system than a traditional HDFS cluster (Chohan et al., 2010). The second enhancement added the accelerator ratio as a predictor variable to the relative fitness model because remote tasks increase disk and network contention (Wang et al., 2009; Zaharia et al., 2008).

Accelerator Fixed-Ratio Relative Fitness Model

The first enhancement improves upon the Elastisizer because the fix-ratio model quantifies the cost statistics for an accelerated file system. MRWATS builds the fixed-ratio model from training benchmarks against the actual file system resources. Unlike the Elastisizer's HDFS model, the fix-ratio model captures the cost statistics for a cluster allocated with both core and accelerator nodes.

To validate that the fix-ratio model improved prediction accuracy over the Elastisizer, the fix-ratio experiments evaluated both models against clusters with 50% accelerators. Both the fixed-ratio and HDFS models are built with benchmarks run on a single cluster, Table 18. This analysis section will demonstrate the fixed-model improvements over the Elastisizer's HDFS model on small-scale, Table 19, and large-scale, Table 20, clusters.

The fixed-ratio model demonstrated improvements over the Elastisizer on a 10-node cluster because the fixed-ratio model better captured the accelerated clusters I/O characteristics. The fixed-ratio accelerated model reduced the average prediction error 11% compared to the HDFS model, Table 31. Although the original model performed similarly for CPU bound workloads, the fixed-ratio model performed better for I/O bound workloads, like TeraSort and Grep. For instance, the accelerated model reduced grep's prediction error by almost 50% on the c1.medium instance type. The accelerated model performed better for the I/O bound workloads because the accelerators generated some HDFS file contention compared to an un-accelerated cluster.

Table 31: Fixed-Ratio Model's Prediction Error with 50% Accelerators

	Execution Time (sec.)			% Prediction Error	
Workload	Actual	HDFS Pred.	Fixed Pred.	HDFS Model	Fixed Model
C1 Medium					
Co-Occurrence	1033.58	1275.67	1403.82	23.42	35.82
Grep	266.34	121.58	193.3	54.35	27.43
Pi Est.	490.55	459.51	560.43	6.33	14.24
TF-IDF	2318.76	2223.11	2260.69	4.13	2.50
TPCH Q12	547.1	447.1	576.86	18.28	5.44
TPCH Q14	413.38	351.9	342.82	14.87	17.07
TeraSort	1688.82	1576.65	1583.05	6.64	6.26
Word Count	1976.66	1970.1	2004.65	0.33	1.42
M1 Large					
Co-Occurrence	1227.93	1241.53	1356.98	1.11	10.51
Grep	223.85	101.15	149.96	54.81	33.01
Pi Est.	598.57	709.81	751.19	18.59	25.5
TF-IDF	2665.1	2306.95	2402.44	13.44	9.86
TPCH Q12	496.65	689.9	720.74	38.91	45.12
TPCH Q14	466.24	389.29	402.79	16.5	13.61
TeraSort	1045.74	964.82	1060.53	7.74	1.41
Word Count	2321.87	2122.42	2184.76	8.59	5.91
			Average Error	18.00	15.94

Note. The fixed accelerator ratio model (MRWATS) was generated with training benchmarks against a cluster with 5 core-nodes and 5 accelerator-nodes. The HDFS model (Elastisizer) was generated with training benchmarks against a 10-node non-accelerated cluster.

The fixed-ratio model also improves upon the Elastisizer's HDFS model at larger scales because the ratio captures the relative amount of contention the accelerator nodes generate. Unlike the Amazon S3 model, the map-reduce cluster allocates additional file system resources with additional core nodes. If the accelerator ratio remains constant, then a larger cluster should generate relatively the same contention on each core file system node. When the map-reduce cluster size increases with the same accelerator ratio, the fixed-ratio model should scale well because more core nodes are added to handle the increased file system load.

To validate the new model scaled well, this section will compare the fixed-ratio and original HDFS models to determine the execution time prediction accuracy up to 30 nodes. The analysis evaluated larger cluster sizes to determine if sized factored into file system contention. Like the 10-node experiment, the large-scale experiment evaluated the execution time prediction accuracy for the workloads in Table 12. For each workload, the evaluation captured the predicted and actual execution times to calculate the prediction error. If the fixed-ratio model produced a smaller average prediction error than the Elastisizer's original HDFS model, then MRWATS accomplished the dissertation goal.

The fixed-ratio model demonstrated improvements over the Elastisizer for cluster sizes up to 30-nodes because the ratio provided a scalable file system contention indicator. The fixed-ratio model reduced the average prediction error for both the c1.medium (Table 32) and m1.large (Table 33) instance types up to 30 nodes. Across both instance types, the fixed-ratio accelerator model reduced the average prediction error by 11.54% compared to the HDFS model because the fixed-ratio model reduced the prediction error for I/O bound workloads. For instance, the fixed-ratio model reduced the grep workload's error by 28.95% compared to the Elastisizer. To better analyze the prediction improvements the c1.medium and m1.large instance types will be analyzed separately.

For the c1.medium clusters up to 30 nodes, the fixed-ratio model demonstrated improved prediction accuracy compared to the Elastisizer's base HDFS model. The fixed-ratio model reduced the average prediction error by 11.18% across all cluster sizes, Table 32. Although the fixed-ratio model performed similarly at 10 nodes, the model improved relative to the HDFS model as nodes were added. The c1.medium fixed-ratio model reduced the average error over the HDFS model by 14.59% and 12.41%, for 20 and 30 nodes respectively.

Like the small-scale results, the fixed-ratio model reduced the average prediction error because the fixed-ratio model quantified the remote file system cost statistics better than the Elastisizer's HDFS model. I/O bound workloads generate file system contention with 50% accelerators because the core data nodes must respond to local and remote file system request simultaneously (Wang et al., 2009). The fix-ratio benchmarks capture file system contention because the benchmarks' remote tasks generate the same contention. The c1.medium results demonstrate that the benchmarks modeled the file system contention because the fixed-ratio model reduced the prediction error for the I/O bound workloads (Grep, TeraSort, TF-IDF, TPC-H Q12) by 38.75%.

Table 32: C1.Medium Fixed-Ratio Model's Prediction Error with 50% Accelerators

	Execution Time (Sec.)			% Prediction Error	
Nodes	Acc. Actual	HDFS Pred.	Fixed Pred.	HDFS Model	Fixed Model
Co-Occurrence					
10	1033.58	1275.67	1403.82	23.42	35.82
20	544.18	646.45	711.5	18.79	30.75
30	405.34	482.89	533.16	19.13	31.53
Grep					
10	266.34	121.58	193.3	54.35	27.43
20	166.27	66.79	102.65	59.83	38.27
30	108.79	48.53	72.43	55.39	33.42
Pi Est.					
10	490.55	459.51	560.43	6.33	14.24
20	306.62	281.22	341.77	8.28	11.46
30	199.45	192.08	232.44	3.7	16.54
TeraSort					
10	1688.82	1576.65	1583.05	6.64	6.26
20	885.28	825.19	827.39	6.79	6.54
30	690.66	515.44	519.78	25.37	24.74
TF-IDF					
10	2318.76	2223.11	2260.69	4.13	2.50
20	1182.48	1173.24	1191.53	0.78	0.77
30	850.56	783.64	795.76	7.87	6.44
TPCH Q12					
10	496.03	447.1	576.86	9.86	16.29
20	345.65	246.32	314.9	28.74	8.9
30	215.99	167.2	213.01	22.59	1.38
TPCH Q14					
10	413.38	351.9	342.82	14.87	17.07
20	319.05	214.51	209.08	32.77	34.47
30	185.45	145.81	142.21	21.38	23.31
Word Count					
10	1976.66	1970.1	2004.65	0.33	1.42
20	972.55	991.86	1009.13	1.99	3.76
30	692.92	661.99	674.27	4.46	2.69
			Average Error	18.24	16.20

Note. The fixed accelerator ratio model (MRWATS) was generated with training benchmarks against a cluster with 5 core-nodes and 5 accelerator-nodes. The HDFS model (Elastisizer) was generated with training benchmarks against a 10-node non-accelerated cluster.

The fixed-ratio model also scaled well for the m1.large instance types up to 30 nodes. The fixed-ratio model reduced the average prediction error compared to the Elastisizer's HDFS model by 13.10% (from 18.24 to 16.20 percent error) across all m1.large cluster sizes, Table 33. The m1.large fixed-ratio model scaled consistently as the number of nodes and tasks increased. The fixed-ratio model reduced the prediction error by 9.24%, 18.59%, and 11.61% for the 10, 20, and 30 node clusters respectively.

Again, the fixed-ratio model reduced the average prediction error because the fixed-ratio model quantified the remote file system cost statistics better than the pure HDFS model on the m1.large instance type. The fix-ratio benchmarks capture file system contention because the benchmarks' remote tasks generate the same type of contention. For example, word count actually became I/O bound with the accelerator nodes. Although the HDFS model should have over-predicted the word-count workload's execution time (Herodotou et al., 2011), the HDFS model actually under-predicted by ~9% because the workload shifted to an I/O bound workload. The fixed-model actually captured the accelerator's impact on execution time because the fixed-model reduced the prediction error by 29% compared to the Elastisizer.

Table 33: M1.Large Fixed-Ratio Model's Prediction Error with 50% Accelerators

	Execution Time (Sec.)			% Prediction Error	
Nodes	Acc. Actual	HDFS Pred.	Fixed Pred.	HDFS Model	Fixed Model
Co-Occurrence					
10	1227.93	1241.53	1356.98	1.11	10.51
20	736.49	629.84	689.05	14.48	6.44
30	516.58	422.24	461.53	18.26	10.66
Grep					
10	223.85	101.15	149.96	54.81	33.01
20	265.89	56.58	80.98	78.72	69.54
30	206.28	45.43	63.74	77.98	69.10
Pi Est.					
10	598.57	709.81	751.19	18.59	25.50
20	378.21	361.81	382.54	4.34	1.14
30	269.10	361.81	382.54	34.45	42.15
TeraSort					
10	1045.74	964.82	1060.53	7.74	1.41
20	637.27	507.86	558.54	20.31	12.35
30	522.98	339.97	371.37	34.99	28.99
TF-IDF					
10	2665.10	2306.95	2402.44	13.44	9.86
20	1404.78	1195.07	1243.08	14.93	11.51
30	956.44	872.74	908.32	8.75	5.03
TPCH Q12					
10	496.65	689.90	720.74	38.91	45.12
20	315.65	387.89	404.66	22.89	28.20
30	366.71	308.55	322.35	15.86	12.10
TPCH Q14					
10	466.24	389.29	402.79	16.50	13.61
20	251.37	231.39	239.22	7.95	4.84
30	245.95	162.64	167.86	33.88	31.75
Word Count					
10	2321.87	2122.42	2184.76	8.59	5.91
20	1183.84	1068.53	1099.72	9.74	7.11
30	820.86	749.91	771.60	8.64	6.00
			Average Error	23.58	20.49

Note. The fixed accelerator ratio model (MRWATS) was generated with training benchmarks against a cluster with 5 core-nodes and 5 accelerator-nodes. The HDFS model (Elastisizer) was generated with training benchmarks against a 10-node non-accelerated cluster.

Accelerator Variable-Ratio Relative Fitness Model

While the first enhancement focused on execution time estimates for fixed accelerator ratio clusters, cloud users need to evaluate workloads against different cluster configurations to find the lowest monetary cost solution. MRWATS further improves upon the Elastisizer because MRWATS enhanced the Elastisizer's relative fitness models to predict a workload's execution time given different accelerator ratios. The second enhancement added the accelerator ratio as a predictor variable to the relative fitness model because remote tasks increase disk and network contention (Wang et al., 2009; Zaharia et al., 2008).

The second enhancement improves upon the Elastisizer because the variable-ratio relative fitness model adapts its predictions as accelerator nodes are added to an existing cluster.   Where as the Elastisizer assumes a workload's execution remains constant regardless of the accelerator ratio. The variable-ratio model captures the cost statistics for a cluster allocated with different accelerator ratios because MRWATS builds the model from training benchmarks against different accelerator ratios. The variable-size enhancement enables the cloud user to compare the different accelerator ratios to find the configuration with the lowest monetary cost and or execution time.

To validate the variable-ratio model improved the prediction accuracy over the Elastisizer, the variable-ratio experiments evaluated the variable-ratio and HDFS models against clusters allocated with different accelerator ratios. While the HDFS benchmarks were built from an un-accelerated cluster, the variable-ratio relative fitness models were built with benchmarks run on clusters with different accelerator ratios, Table 21.   Once the models were built, the variable-ratio experiment evaluated the test workloads in Table 12 against 30-node clusters allocated with different accelerator ratios, Table 22. The experiments evaluated the prediction accuracy for both the c1.medium and m1.large instance types.

The variable-ratio model improved the prediction accuracy over the Elastisizer on the c1.medium instance type because the variable-ratio benchmarks captured representative workload cost statistics at different accelerator ratios.   While CPU bound workloads' execution time remained constant as the accelerator ratio changes, some I/O bound workloads' execution time increased over three times. For example, grep's execution time increased from 128 to 422 seconds when the accelerator ratio changed from 33% to 83%, Table 34. The variable-ratio relative-fitness model distinguishes between the CPU and I/O bound workloads based on the workload's baseline cost statistics. If the training benchmarks correctly model the workloads I/O characteristics, then the variable-ratio model not only should improve the average execution time prediction accuracy but also determine which workload's execution time will degrade with the higher accelerator ratios (Mesnier et al., 2007).

The variable-ratio model correctly captured the c1.medium workload's I/O characteristics because the variable-ratio model reduced the average prediction error and predicted the workloads accelerator ratio degradation better than the Elastisizer. The variable-ratio model reduced the average prediction error compared to the Elastisizer's HDFS model, Table 34. The variable-ratio model's average prediction error was 16.26% compared to the HDFS model's 29.39% prediction error.

The accelerated model captured each ratio's file system contention because the accelerated model further improved upon the HDFS model as the accelerator ratio increased. When the cluster allocated only 10 accelerators, the variable-ratio model reduced the prediction error by 18.54% over the HDFS model. When the cluster allocated 25 accelerators, the variable-ratio model reduced the prediction error by 57.95% over the HDFS model.

Table 34: C1.Medium Variable-Ratio Model's Prediction Error

	Execution Time (Sec.)			% Prediction Error	
Ratio	Acc. Actual	HDFS Pred.	VAR Pred.	HDFS Model	VAR Model
Co-Occurrence					
0.33	407.59	482.89	525.36	18.48	28.89
0.5	405.34	482.89	527.01	19.13	30.01
0.67	409.97	482.89	539.35	17.79	31.56
0.83	346.46	482.89	540.99	39.38	56.15
Grep					
0.33	128.17	48.53	66.78	62.14	47.9
0.5	108.79	48.53	72.15	55.39	33.68
0.67	213.54	48.53	132.27	77.27	38.06
0.83	422.45	48.53	371.67	88.51	12.02
Pi Est.					
0.33	205.94	192.08	221.85	6.73	7.72
0.5	199.45	192.08	221.85	3.7	11.23
0.67	205.36	192.08	217.84	6.47	6.08
0.83	201.02	192.08	217.84	4.45	8.37
TeraSort					
0.33	674.72	515.44	588.76	23.61	12.74
0.5	690.66	515.44	639.01	25.37	7.48
0.67	837.06	515.44	717.49	38.42	14.29
0.83	1505.23	515.44	1229.81	65.76	18.3
TF-IDF					
0.33	849.25	783.64	794.97	7.73	6.39
0.5	850.56	783.64	796.61	7.87	6.34
0.67	956.66	783.64	815.3	18.09	14.78
0.83	1209.87	783.64	930.26	35.23	23.11
TPCH Q12					
0.33	175.7	167.2	185.96	4.84	5.84
0.5	215.99	167.2	202.24	22.59	6.37
0.67	290.76	167.2	259.09	42.49	10.89
0.83	541.98	167.2	402.72	69.15	25.69
TPCH Q14					
0.33	179.23	145.81	164.91	18.65	7.99
0.5	185.45	145.81	168.96	21.38	8.89
0.67	265.16	145.81	227.56	45.01	14.18
0.83	537.47	145.81	472.03	72.87	12.18
Word Count					
0.33	692.85	661.99	679.15	4.45	1.98
0.5	692.92	661.99	678.16	4.46	2.13
0.67	705.63	661.99	677.17	6.18	4.03
0.83	711.47	661.99	676.32	6.95	4.94
			Average	29.39	16.26

Note. The variable accelerator ratio model (MRWATS) was generated with training benchmarks against four 30-node clusters with 33%, 50%, 67%, and 83% accelerator ratios. The HDFS model (Elastisizer) was generated with training benchmarks against a 30-node non-accelerated cluster.

The variable-ratio model offers an additional improvement over the Elastisizer's original relative-fitness models because the variable-ratio model captures the workload's relative cost statistic changes between accelerator ratios. While the execution time prediction error measures each model's accuracy, the cloud user also requires the accelerator ratio's relative impact for a given workload. The relative changes might be more valuable than absolute accuracy because the relative changes indicate the workload's appropriateness for acceleration (Chohan et al., 2010). If a workload's execution time triples as the accelerator ratio increases, then the workload is not a good candidate for acceleration because the accelerated workload will cost the user more money to execute than an un-accelerated workload.

The variable-ratio model predicts workload degradation better than the Elastisizer because the variable-ratio benchmarks capture the workloads' change in I/O and CPU characteristics. If the I/O bound workloads degrades as the accelerator ratio increases, then the variable-ratio benchmarks represent the change in the cost statistics. As a result, the variable-ratio model predicted the workload degradation better than the HDFS model for the c1.medium virtual machine type, Figure 19. While the HDFS model assumes all workloads remain constant, the variable-ratio model accurately predicted execution time degradation in five out of six HDFS bound workloads.

The variable-ratio model also predicted execution time remains relatively constant in both CPU bound workloads. Although the variable-ratio model over-predicted the Co-Occurrence workload's execution time compared to the HDFS model, the variable-ratio model did at least predict the workload would remain relatively flat at a only 2% execution time degradation, Figure 19. The slight degradation indicates the workload will be monetarily cost effective at high accelerator ratios on the c1.medium instance type.

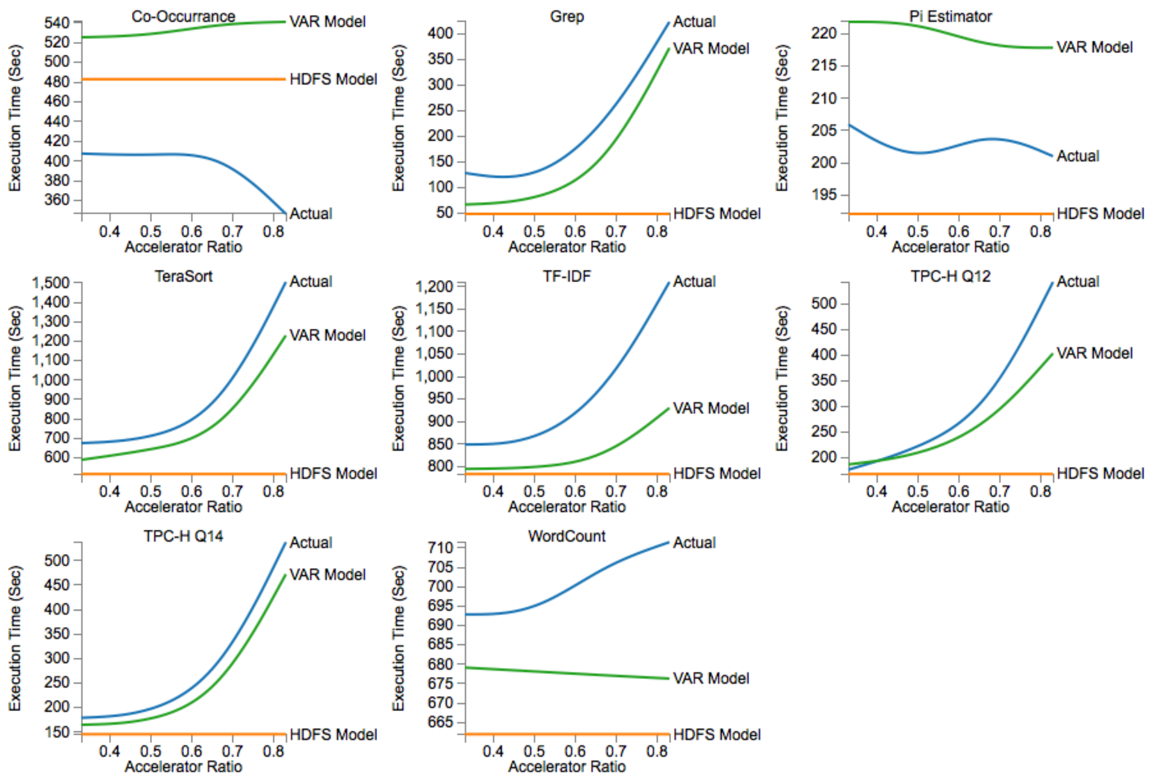

Figure 19: Actual, VAR model predicted (MRWATS) and HDFS model predicted (Elastisizer) execution time versus accelerator ratio on a c1.medium cluster.

Like the c1.medium variable-ratio model, the m1.large variable-ratio model improved the prediction accuracy over the Elastisizer because the variable-ratio benchmarks model representative workload cost statistics at different accelerator ratios. The accelerator ratio does impact some workloads' I/O cost statistics. Although the m1.large's workloads don't degrade as much as the c1.medium workloads with additional accelerators, TeraSort's actual execution time increased over two times, from 472 to 1057 seconds, with 20 additional accelerators, Table 35. If the training benchmarks correctly model the workloads I/O characteristics, then the variable-ratio model not only should improve average execution time prediction accuracy but also determine which workload's execution time will degrade with the higher accelerator ratios (Mesnier et al., 2007).

The variable-ratio model also correctly captured the m1.large's workload's I/O characteristics because the variable-ratio model reduced the average prediction error compared to the Elastisizer. Although the workloads perform differently on the m1.large instances, the variable-ratio model reduced the average prediction error by 24.18% (from 30.89 to 23.49 percent error) compared to the HDFS model, Table 35.

The m1.large variable-ratio model captured each ratio's file system contention because it performed better compared to the HDFS as the accelerator ratio increased, Figure 20. When the m1.large cluster allocated 10 accelerators, the accelerated model reduced the average prediction error by 12.98% over the HDFS model. When the m1.large cluster allocated 25 accelerators, the accelerated model reduced the average prediction error by 35.27% over the HDFS model.

Table 35: Accelerated File System M1.Large Accuracy for Various Accelerator Ratios

	Execution Time (Sec.)			% Prediction Error	
Ratio	Actual	HDFS Model	VAR Model	HDFS Model	VAR Model
Co-Occurrence					
0.33	475.87	422.24	411.82	11.27	13.46
0.5	516.58	422.24	417.83	18.26	19.12
0.67	466.81	422.24	423.84	9.55	9.2
0.83	464.34	422.24	427.19	9.07	8
Grep					
0.33	168.7	45.43	78.68	73.07	53.36
0.5	206.28	45.43	95.69	77.98	53.61
0.67	322.3	45.43	112.69	85.9	65.04
0.83	281.18	45.43	129.69	83.84	53.87
Pi Est.					
0.33	283.43	361.81	367.6	27.65	29.7
0.5	269.1	361.81	367.63	34.45	36.61
0.67	278.4	361.81	367.66	29.96	32.06
0.83	289.01	361.81	367.66	25.19	27.21
TeraSort					
0.33	471.46	339.97	379.28	27.89	19.55
0.5	522.98	339.97	402.89	34.99	22.96
0.67	756.6	339.97	426.49	55.07	43.63
0.83	1057.24	339.97	609.83	67.84	42.32
TF-IDF					
0.33	1015.67	872.74	899.21	14.07	11.47
0.5	956.44	872.74	913.85	8.75	4.45
0.67	1018.53	872.74	929.47	14.31	8.74
0.83	1247.82	872.74	1080.2	30.06	13.43
TPCH Q12					
0.33	241.53	308.55	281.4	27.75	16.51
0.5	366.71	308.55	316.1	15.86	13.8
0.67	352.04	308.55	350.79	12.35	0.35
0.83	464.71	308.55	385.45	33.6	17.05
TPCH Q14					
0.33	220.73	162.64	139.16	26.32	36.96
0.5	245.95	162.64	162.53	33.88	33.92
0.67	227.42	162.64	185.9	28.49	18.26
0.83	286.46	162.64	207.22	43.23	27.66
Word Count					
0.33	812.77	749.91	775.25	7.73	4.62
0.5	820.86	749.91	773.76	8.64	5.74
0.67	816.81	749.91	772.29	8.19	5.45
0.83	799.25	749.91	771.13	6.17	3.52
			Average	30.98	23.49

Note. The variable accelerator ratio (VAR) model (MRWATS) was generated with training benchmarks against four 30-node clusters with 33%, 50%, 67%, and 83% accelerator ratios. The HDFS model (Elastisizer) was generated with training benchmarks against a 30-node non-accelerated cluster.

The m1.large variable-ratio model improves on Elastisizer's HDFS model because the variable-ratio model captures the workload's relative cost statistic changes between accelerator ratios. While absolute execution time prediction accuracy improves the cloud user's ability to predict workload monetary costs, the cloud users require the accelerator's relative impact on execution time for a given instance type. The relative execution time changes allow the cloud user to select the accelerator configuration with the lowest monetary cost and or execution time. For instance, an HDFS bound workload might make better sense on the m1.large instance type than the c1.medium because the workload can provision relatively more accelerators.

While the m1.large workloads did not degrade as much as the c1.medium workloads, the variable-ratio model predicted the accelerator ratio's execution time impact better than the HDFS model for the m1.large virtual machine type, Table 20. While the HDFS model assumes all workloads remain constant, the variable-ratio model predicted execution time increases in five out of five HDFS bound workloads. The variable-ratio model also correctly predicted 2/3 of the CPU bound workload's execution time would not degrade.

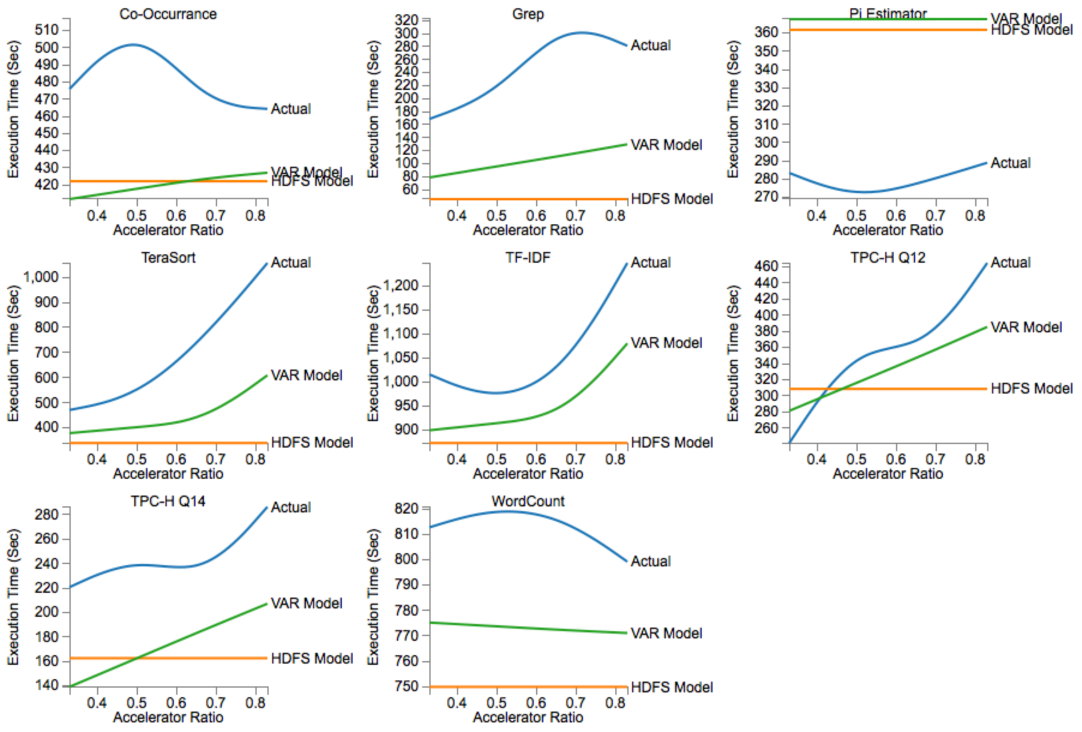

Figure 20. Actual, VAR model predicted (MRWATS) and HDFS model predicted (Elastisizer) execution times versus accelerator ratio on an m1.large cluster.

In summary, MRWATS includes two enhancements to the Elastisizer's relative fitness models that improve accelerated workloads' prediction accuracy. The first enhancement improves upon the Elastisizer's relative fitness model because the enhancement modeled the accelerated file system cost statistics for a fixed accelerator ratio. Although the model mainly improved the I/O bound workloads' prediction accuracy, the fixed-ratio model reduced the average prediction error by 11% for a mixture of CPU and I/O bound workloads across two instance types.

The second enhancement improves upon the Elastisizer's relative fitness model because the enhancement models the file system contention as the accelerator ratio changes. While the Elastisizer's model assumes all workloads have the same execution time regardless of the accelerator ratio, MRWATS's variable-ratio model predicts the workload's performance degradation as the accelerator ratio increases. The variable-ratio model improved both the execution time prediction accuracy and workload classification accuracy over the Elastisizer's static relative fitness model. The variable-ratio model improved the execution time prediction accuracy because it reduced the average prediction error by 34% over the Elastisizer's model. The variable-ratio also classified workload degradation better than the Elastisizer because the variable-ratio model's false positive rate was only 12% compared to the HDFS model's 68% false positive rate.

Failure Aware Task Simulator Data Analysis

While the relative fitness model enhancements addressed issues related to the storage strategy, the Elastisizer does not model the execution time impacts from early spot instance termination. When the spot market auction terminates spot instances, the map reduce scheduler can't allocate tasks to the spot instances' slots. Furthermore, the map reduce scheduler must reschedule the tasks from the accelerated slots because the early termination creates missing intermediate data (Chohan et al., 2010).

MRWATS improves upon the Elastisizer during spot instance early termination because MRWATS enhanced the task scheduler and virtual profiles to handle failures. The enhanced task simulator mimics the actual task scheduler's failure handling. Once the scheduler detects the failure, the enhanced tasks simulator removes the failed slot from the available slot queue. The simulator then reschedules the tasks with missing intermediate data.

While the task simulator handles failed tasks, MRWATS has to adjust to different cluster resources because the accelerator ratio changes when the failure occurs. The task simulator uses two virtual profiles to account for the clusters configuration before and after the failure. Before the spot market auction terminates the accelerator node, the simulator calculates the task completion time with the accelerated virtual profile. Once the auction terminates the accelerator node, the simulator calculates the remaining tasks completion time with the base HDFS profile.

To validate that the enhanced scheduler improved the prediction accuracy over the Elastisizer, the evaluation executed several failure experiments, Table 23. The experiments compared MRWATS's and the Elastisizer's prediction error given early termination in the first map wave, second map wave, first reduce wave, and second reduce wave. If MRWATS predicted the execution time with a smaller prediction error than the Elastisizer, then the enhanced scheduler successfully meets the dissertation goal.

MRWATS scheduler reduced the prediction error when failures occurred because the scheduler correctly estimated the rework from failed tasks. When the spot market terminated the workload early, the updated scheduler reduced the execution time prediction error by 76.12% (from 53.68 to 12.82 percent error) compared to the Elastisizer, Table 36. While MRWATS reduce the prediction errors across all the accelerator configurations, the original scheduler performed worst at the 67% accelerator ratio with a 66% error because the high accelerator ratio contained the most failed tasks. In contrast, MRWATS reduce the prediction error to ~12% at the high accelerator ratio. To explain the improvement, this section will analyze both the c1.medium and m1.large instances in detail.

Table 36: Average Execution Time Prediction Error by Accelerator Ratio

	Average ET % Error	
Accelerator Ratio	Elastisizer	MRWATS
C1 Medium		
33%	41.86	11.84
50%	51.47	12.36
67%	65.84	13.95
M1 Large		
33%	40.86	13.87
50%	55.36	15.42
67%	66.72	9.98
All Tests	53.68	12.82

C1.Medium Failure Simulations

When the experiment evaluated the c1.medium cluster with 67% accelerators, MRWATS scheduler improved the prediction accuracy over the Elastisizer because the enhanced scheduler simulated the rework from the failed task, Table 37. This cluster configuration demonstrated a larger improvement compared to the other cluster configurations because the cluster lost 67% of the worker nodes during the tests. As a result the Elastisizer (TS) under-predicted the workload execution time by ~66% because the scheduler did not account for the failures. Conversely, MRWATS scheduler (ETS) reduced the prediction error to ~14% because the enhanced scheduler estimated the rework from the failed tasks.

Table 37: Early Termination Accuracy by Wave for C1.Medium with 67% Accelerators

	Failed Maps		Failed Reduces		Execution Time (Seconds)			ET % Error	
Terminated Wave	Act.	Pred.	Act.	Pred.	Actual	TS	ETS	TS	ETS
Co-Occurrence									
Map 1	40	40	0	0	1004.01	482.89	1292.16	51.90	28.70
Map 2	80	80	40	40	996.37	482.89	1307.33	51.53	31.21
Reduce 1	111	105	40	40	1130.38	482.89	1521.06	57.28	34.56
Reduce 2	107	105	46	23	1129.18	482.89	1511.32	57.24	33.84
Grep									
Map 1	40	40	0	0	204.68	48.53	126.15	76.29	38.37
Map 2	45	40	0	0	210.45	48.53	117.26	76.94	44.28
Pi Est.									
Map 1	40	40	0	0	512.31	192.08	474.38	62.51	7.40
Map 2	68	63	1	1	594.78	192.08	559.94	67.71	5.86
Reduce 1	71	65	1	1	663.43	192.08	638.94	71.05	3.69
TeraSort									
Map 1	49	40	3	0	1537.46	515.44	1595.01	66.47	3.74
Map 2	88	80	40	40	1504.48	515.44	1636.89	65.74	8.80
Reduce 1	172	160	42	40	1611.36	515.44	1797.62	68.01	11.56
Reduce 2	172	160	40	23	1603.56	515.44	1442.43	67.86	10.05
TF-IDF Job1									
Map 1	40	40	0	0	1713.75	563.17	1670.62	67.14	2.52
Map 2	80	80	40	40	1724.73	563.17	1671.87	67.35	3.07
Reduce 1	411	400	40	40	1803.87	563.17	1736.26	68.78	3.75
Reduce 2	404	400	48	23	1789.63	563.17	1728.41	68.53	3.42
Word Count									
Map 1	40	40	0	0	1925.44	661.99	1971.73	65.62	2.40
Map 2	80	80	40	40	1919.98	661.99	1973.18	65.52	2.77
Reduce 1	409	400	41	40	2205.82	661.99	2035.70	69.99	7.71
Reduce 2	403	400	25	23	2142.80	661.99	2030.72	69.11	5.23
							Average Error	65.84	13.95

Note. MRWATS' Enhanced Task Scheduler (ETS) used virtual profiles generated from the variable ratio-model and the Elastisizer's 10-node model. The Elastisizer's Task Scheduler (TS) used only the virtual profiles from the Elastisizer's 10-node model.

MRWATS outperformed the Elastisizer because the enhanced task scheduler estimated the failed tasks corresponding to the map-reduce wave the failure occur. Table 37's failed tasks compare the enhanced task scheduler's predicted failures to the actual failures. The Elastisizer failed tasks are not shown because the Elastisizer doesn't predict failures. In essence, the Elastisizer always produced a 100% prediction error on the number of failed tasks, regardless of the map-reduce wave.

While the Elastisizer does not predict failed tasks, the enhanced task scheduler predicted the map waves' task failures within 3.8% on average. When the task tracker failed during a given map wave, the number of failures are approximately $k_{accel} \times w_{map}$ where k_{accel} is the number of accelerator slots (Chohan et al., 2010). When the c1.medium workload contained 20 accelerators (or 40 accelerator map slots), the first map wave test contained about 40-failed map tasks. If the workload failed during the second wave, then the second map wave test contained about 80 failed map tasks, except for grep. The grep workload created approximately k_{accel} or 40 failures because the grep workload contained only map tasks and did not lose intermediate data.

The enhanced task scheduler also added reducer wave failure predictions. The enhanced task scheduler predicts the reducer wave's task failures with an average 6.7% prediction error, Table 37. Again, the prediction error represents an improvement because the Elastisizer does not predict the number of failed tasks for a given reduce wave. The failed tasks extend the workloads execution time because the accelerated tasks need to be rescheduled to supply the intermediate data to the reducers.

For the reduce wave failures, the experiments contain both map and reduce failures. If the spot market terminates the accelerator during a reduce wave, the terminated accelerators causes all accelerated map tasks and one reduce wave's tasks, k_{accel}, to fail (Wang et al., 2009). When the c1.medium workload contained 20 accelerators (or 40 accelerator reduce slots), the first reduce wave tests contained about 40 failed reduce tasks and approximately 67% failed map tasks. The only exception was the Pi Estimator workload because the Pi Estimator only contained one reduce task.

While the enhanced task scheduler accurately predicted the reduce failure counts in most experiments, the enhance simulator under-predicted the reduce failure counts in the second reduce wave. For instance, TeraSort contained 40 failed reduces versus 23 failed reduces, Table 37. The simulator under-predicted the second wave failures because the workloads do not contain enough tasks to entirely fill the second wave's accelerator slots. The simulator implementation biased task assignment to the core slots for the partial wave.

Although the enhanced task scheduler underestimated the reduce failure counts in the second wave, MRWATS still improved the execution time prediction accuracy over the Elastisizer because the enhanced task scheduler at least estimated the second waves recovery time. MRWATS reduced the second wave execution time prediction error by 73% over the Elastisizer because the enhanced task simulator found at least one failure in the second wave. The enhanced task simulator only requires one reduce task failure to produce an approximate recovery time estimate because the execution time for one reduce wave equals the longest task in the given wave (Verma et al., 2011). More importantly, one reduce task failure forces the enhanced task scheduler to reprocess the accelerator's map tasks to recreate the missing intermediate data.

While the enhanced task simulator predicted the post failure tasks to reschedule, the task execution times don't remain constant. When the spot instances terminate early, the spot instances no longer contend for file system resources. For the 67% accelerator clusters, the spot instance failure reverts the cluster configuration to a 10-node on-demand cluster. Therefore, the remaining tasks should execute at the same compute-rate as the 10-node on-demand cluster.

The enhanced task scheduler improved it's own workload recovery prediction by using two virtual profiles for the tasks. Before the spot market terminates the spot instances, the enhanced task scheduler estimates the task cost with an accelerated virtual profile because the spot instance accelerators generate file system contention. After the spot market terminates the spot instances, the enhanced task scheduler estimates the task cost with an un-accelerated virtual profile because the spot instance tasks no longer generate file system contention.

The dual virtual profile approach improved the prediction accuracy for I/O bound workloads like TeraSort, Figure 21. With a single virtual profile (ETS 1VP), the enhanced task schedule over-predicted the execution time because the failed task completed at a much faster rate than predicted. The recovery attempts completed in 64.483 seconds on average verses the accelerated virtual profiles predicted time, 81.229 seconds. When the enhanced task scheduler used two virtual profiles (ETS 2VP), the dual profile improve the enhanced task scheduler's own prediction accuracy by 73% on average because the enhanced task scheduler better predicted the recovery rate for the failed task. The predicted recovery attempt time matched the actual recovery attempt time of 64.483 seconds.

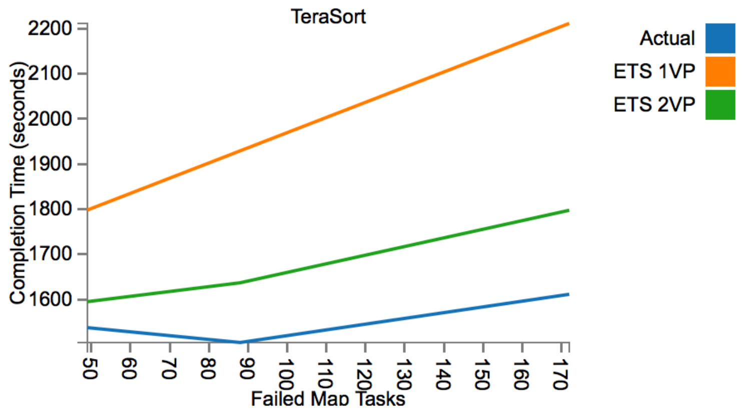

Figure 21. Execution time predictions for the enhanced task simulator using single and dual virtual profiles.

M1.large Failure Simulations

The enhanced task scheduler also improved the execution time prediction accuracy on the m1.large instance type because the enhanced task scheduler estimates the failed tasks' recovery time based on the cluster resources. The enhanced task scheduler mimics the actual task scheduler's failure handling. Once the scheduler detects the failure, the enhanced task simulator removes the failed slots from the available slot queue. The simulator then rescheduled the tasks missing intermediate data with the appropriate m1.large virtual profile.

When the experiments evaluated the m1.large cluster with 67% accelerators, MRWATS's scheduler improved the prediction accuracy over the Elastisizer because the enhanced scheduler simulated the rework from the failed task, Table 38. As with the c1.medium cluster, the 67% accelerator configuration demonstrated a larger improvement compared to the other cluster configurations because the cluster lost 67% of the worker nodes during the tests. As a result the Elastisizer (TS) under-predicted the workload execution time by 66.72% because the scheduler did not account for the failures. Conversely, MRWATS's scheduler (ETS) reduced the prediction error to 9.98% because the enhanced scheduler estimated the rework from the failed tasks.

MRWATS improved upon the Elastisizer because the enhanced task scheduler estimated the failed tasks corresponding to the map-reduce wave the failure occurred. Table 38 compares the enhanced task scheduler's predicted failed tasks to the actual workload failed tasks. As with the c1.medium instance type, the Elastisizer failed tasks are not shown because the Elastisizer doesn't predict failures. When a failure occurs, the

Elastisizer always produced a 100% prediction error on the number of failed tasks, regardless of the map-reduce wave.

While the Elastisizer does not predict failed tasks, the enhanced task scheduler predicted the map waves' task failures within 5.8% on average. The m1.large instances generate more failed tasks than the c1.medium instances because the m1.large instance type contains three map slots versus the c1.medium's two slots. When the m1.large workload contained 20 accelerators (or 60 accelerator map slots), the first map wave test contained about 60-failed map tasks. If the workload failed during the second wave, then the second map wave test contained up to 120 failed map tasks. In contrast, the Elastisizer under-predicts the failed tasks because it doesn't handle failures.

MRWATS also accurately predicted the recovery time for reduce failures because the enhanced task scheduler predicted the number of task failures with a 4.7% prediction error, Table 38. Like the c1.medium configuration, the m1.large cluster contains 40 accelerated reduce slots. When the accelerators terminate on a reduce wave, the termination caused about 40 reduce tasks and 67% map tasks to fail. Although the enhanced scheduler under-predicted the second wave reduce task failures, MRWATS still reduced the execution time prediction error by 86.64% compared to the Elastisizer because the enhance task scheduler estimated the recovery time for one reduce wave and the depended map tasks.

Table 38: Early Termination by Wave for m1.large with 67% Accelerators

	Failed Maps		Failed Reduces		Execution Time (Seconds)			ET % Error	
Terminated Wave	Act.	Pred.	Act.	Pred.	Actual	TS	ETS	TS	ETS
Co-Occurrence									
Map 1	60	60	0	0	1275.73	422.24	1242.46	66.90	2.61
Map 2	108	107	40	40	1290.25	422.24	1240.04	67.27	3.89
Reduce 1	105	107	40	40	1362.25	422.24	1295.93	69.00	4.87
Reduce 2	108	110	54	23	1402.36	422.24	1280.16	69.89	8.71
Grep									
Map 1	60	60	0	0	197.67	45.43	123.57	77.02	37.49
Map 2	67	41	0	0	221.70	45.43	171.29	79.51	22.74
Pi Est.									
Map 1	60	60	0	0	648.97	361.81	712.74	44.25	9.83
Map 2	67	68	0	1	708.05	361.81	799.52	48.90	12.92
Reduce 1	69	66	1	1	800.29	361.81	958.52	54.79	19.77
TeraSort									
Map 1	67	60	0	0	1031.50	339.97	993.47	67.04	3.69
Map 2	147	120	40	40	999.00	339.97	1008.73	65.97	0.97
Reduce 1	157	161	40	40	1107.00	339.97	1113.02	69.29	0.54
Reduce 2	160	163	42	23	1194.91	339.97	932.15	71.55	21.99
TF-IDF Job1									
Map 1	60	60	0	0	1965.97	631.00	1787.58	67.90	9.07
Map 2	120	120	40	40	1963.94	631.00	1792.89	67.87	8.71
Reduce 1	403	403	40	40	2069.65	631.00	1962.63	69.51	5.17
Reduce 2	405	403	40	23	2093.23	631.00	1955.32	69.86	6.59
Word Count									
Map 1	60	60	0	0	2352.76	749.91	2125.76	68.13	9.65
Map 2	120	120	40	40	2341.91	749.91	2129.05	67.98	9.09
Reduce 1	404	403	40	40	2445.23	749.91	2307.34	69.33	5.64
Reduce 2	403	403	42	23	2438.28	749.91	2300.53	69.24	5.65
							Average Error	66.72	9.98

Note. MRWATS' Enhanced Task Scheduler (ETS) used virtual profiles generated from the variable ratio-model and the Elastisizer's 10-node model. The Elastisizer's Task Scheduler (TS) used only the virtual profiles from the Elastisizer's 10-node model.

In summary, MRWATS performs better than the Elastisizer during early termination because MRWATS enhanced the task scheduler and virtual profile to handle failures. Once the scheduler detects the failure, the enhanced tasks simulator removes the failed slots from the available slot queue. The simulator then rescheduled the failed tasks with the appropriate virtual profile to match the new cluster resources. The enhancements allow MRWATS to predict the early termination recovery time, where as the Elastisizer has no such mechanism.

The enhanced task scheduler reduced the early termination prediction error by 76.12% on average over the Elastisizer's scheduler because the updated scheduler accurately determined the increased execution time due to the failures. For the 67%

accelerator ratio, the enhanced scheduler estimated the failed tasks with 5.1% and 5.8% average prediction error for the c1.medium and m1.large instance types, respectively. While MRWATS accounted for the failed tasks, the Elastisizer did not account for the failed tasks. As a result, the Elastisizer consistently under-predicted the execution time by 50 to 80 percent.

Forecast Tool Data Analysis

While the early termination scheduler determines a given failure's impact on execution time, it does not fully predict the spot market's impact on the workload's completion time and or financial costs. The spot market auction might cause a workload to wait because the bid price might be below the market price (Chohan et al., 2010). The cloud user also doesn't know the exact time an auction will terminate the spot instances because the market price is partly based on demand (Orna Agmon Ben-Yehuda et al., 2013).

This study created a forecast tool, MRWATS, to determine the completion time and monetary cost for spot market workloads. MRWATS improves upon the Elastisizer because MRWATS estimates the completion time and monetary costs based on a spot market trace. Unlike the Elastisizer's execution time predictions, completion time predictions include spot auction wait and recovery time estimates. MRWATS also improves the monetary cost estimates because the forecast tool incorporates the average market price and potential recovery costs.

To validate the MRWATS enhancements, this section assessed MRWATS' and the Elastisizer's prediction accuracy against two different actual market traces: c1.medium (Figure 15) and m1.large (Figure 16). Each trace represents different historical market conditions for the tool to predict the workloads' completion time and monetary costs. MRWATS improved upon the Elastisizer's predictions for both traces because each trace contains market conditions, which the Elastisizer does not account for. The c1.medium trace contains frequent early terminations at low bids. The m1.large trace contains regular demand patterns where some workloads need to wait.

For each price history, the experiments split the price history into training and test traces to compare MRWATS against the Elastisizer.   MRWATS predicted the completion time and monetary cost based on the training trace. The Elastisizer predicted the completion time based on the workload's virtual profile. The Elastisizer computed the monetary cost by naively assuming the bid price equaled the market price.   The experiment then computed the actual workload average completion time and monetary cost via a Monte Carlo simulation using the test trace.

When the experiment completed the Monte Carlo simulation, the experiment compared MRWATS and the Elastisizer based on the simulation results. For each workload, the evaluation computed both tools' prediction error. If MRWATS reduced the prediction error over the Elastisizer, then the MRWATS' forecast enhancements met the dissertation goal.

C1 Medium Trace Data Analysis

MRWATS improves upon the Elastisizer for market conditions like the c1.medium trace because MRWATS estimates the completion time and monetary cost based on the market history. Where as the Elastisizer assumes the execution time remains constant and costs increase linearly with bid price changes, MRWATS factors in wait times, early terminations, and past market prices into the completion time and cost predictions. Furthermore, MRWATS also improves upon the Elastisizer because the new tool adapts the prediction based upon the spot instance accelerator ratio.

For the c1.medium trace, MRWATS outperformed the Elastisizer for both average completion time and costs across all accelerator ratios, Table 39. MRWATS improved the average completion time prediction accuracy because it includes the average wait and recovery time in the prediction. MRWATS also improved the completion time prediction accuracy by using accelerator specific virtual profiles. In combination, the completion time improvements reduced the average completion time prediction error by 91.5% (from 31.12 to 2.62 % error) compared to the Elastisizer, Table 39. MRWATS improved the average monetary cost prediction accuracy because MRWATS includes recovery costs and the average market price in the prediction. Both cost adjustments reduced the average cost prediction error by 99.1% (from 347.40 to 3.00 % error), Table 39.

MRWATS created better completion time predictions than the Elastisizer for the c1.medium trace because MRWATS estimates the wait times and early termination costs. The c1.medium trace's market prices ranged from \$0.018 to \$2.50. Workloads bid at \$0.018 were more likely to wait and terminated early. For workloads allocated at the minimum bid, the Elastisizer misestimated the average completion time with as much as a 99% error for the bids at \$0.018. Comparatively, MRWATS misestimated the average completion time with only a 3.1% error for bids at \$0.018.

MRWATS also created better completion time predictions than the Elastisizer for the c1.medium trace because MRWATS predicted the completion time with accelerator specific virtual profiles. As a result, MRWATS reduced the completion time prediction error over the Elastisizer as the accelerator ratio increased. The Elastisizer approach performed worst for the 67% ratio because the original HDFS relative fitness model does not factor in the performance degradation for I/O bound workloads. The Elastisizer model's error ranged from 37% to 73% for the I/O bound workloads (TeraSort, TPC-H, and grep). In comparison, MRWATS's error ranged from 1% to 6% for the same workloads because the accelerated model accounts for HDFS contention.

While MRWATS's completion time accuracy performed well compared to the Elastisizer, MRWATS also improved the monetary cost prediction accuracy because MRWATS factors in the average market price into the prediction. The Elastisizer's average error ranged from 177% to 710% because it's cost model naïvely assumed the bid and market price were equal. While the market price did peak at $2.50, the average market price for the test trace was only $0.1236. In comparison, MRWATS perform much better than the Elastisizer's naïve cost model with an average error between 0.83% and 7.56%.

Table 39: Completion Time and Financial Cost Error by Accelerator Ratio

Ratio	Ave. Completion Time % Error		Ave. Cost % Error	
	Elastisizer	MRWATS	Elastisizer	MRWATS
Co-Occurrence				
0.33	18.74	1.63	246.35	2.02
0.50	20.86	2.69	396.64	3.39
0.67	25.69	4.61	555.11	5.27
Grep				
0.33	51.58	2.04	159.87	0.74
0.50	55.30	1.84	237.17	1.29
0.67	75.22	1.39	148.68	1.50
Pi Est.				
0.33	29.53	0.88	222.37	0.82
0.50	30.34	0.78	360.61	1.44
0.67	30.81	1.10	529.92	2.40
TF-IDF Job1				
0.33	12.89	2.17	266.90	2.54
0.50	15.39	3.80	427.06	4.35
0.67	19.53	6.51	610.88	6.75
TPCH Q12				
0.33	27.51	0.79	232.92	0.99
0.50	33.67	0.77	337.55	1.43
0.67	47.96	1.15	362.88	2.13
TPCH Q14				
0.33	30.44	1.06	232.92	1.15
0.50	33.64	0.82	356.41	1.98
0.67	50.84	0.73	349.14	1.91
TeraSort				
0.33	21.97	2.06	228.95	2.40
0.50	29.57	3.59	337.62	4.13
0.67	38.87	6.46	430.40	6.54
Word Count				
0.33	12.37	2.73	266.58	3.12
0.50	14.95	4.88	427.59	5.33
0.67	19.25	8.39	613.08	8.28
Average	31.12	2.62	347.40	3.00

C1 Medium Trace - 50% Accelerator Ratio Data Analysis

Although the overall results demonstrated that MRWATS improved the completion time and cost prediction accuracy, the 50% accelerator results further explain the differences between MRWATS and the Elastisizer. MRWATS improved both completion time and cost predictions because it estimated the spot market impacts on both measures. When the simulation tested the 50% accelerator ratio configuration, MRWATS reduced the completion time prediction error by 91.8% and the cost prediction error by 99.19% over the Elastisizer's predictions. Appendix B details similar results for the 33% and 67% accelerator ratios.

To better explain the spot market impact on completion time and cost, this section analyzes the completion time and cost prediction accuracy for low and high bids. MRWATS improves upon the Elastisizer at low bids because MRWATS predicts the average wait and recovery time when the workload is most likely to encounter both. MRWATS also improves upon the Elastisizer at high bids because MRWATS factors the average market price into the prediction.

At low bids, MRWATS outperformed the Elastisizer's completion time prediction because MRWATS factored in wait times and early terminations. For instance, the c1.medium trace contains frequent spot instance terminations at bid prices near the minimum bid. When the user bids the minimum bid, \$0.018, the Elastisizer is off by an order of magnitude, Table 40. The Elastisizer underestimates the completion by a large measure because most low bid jobs require an extra 15 to 16 hours of spot market overhead with the c1.medium trace, Figure 22. MRWATS performed much better at the minimum bid with a 2.4% maximum error because MRWATS includes the wait and recovery times.

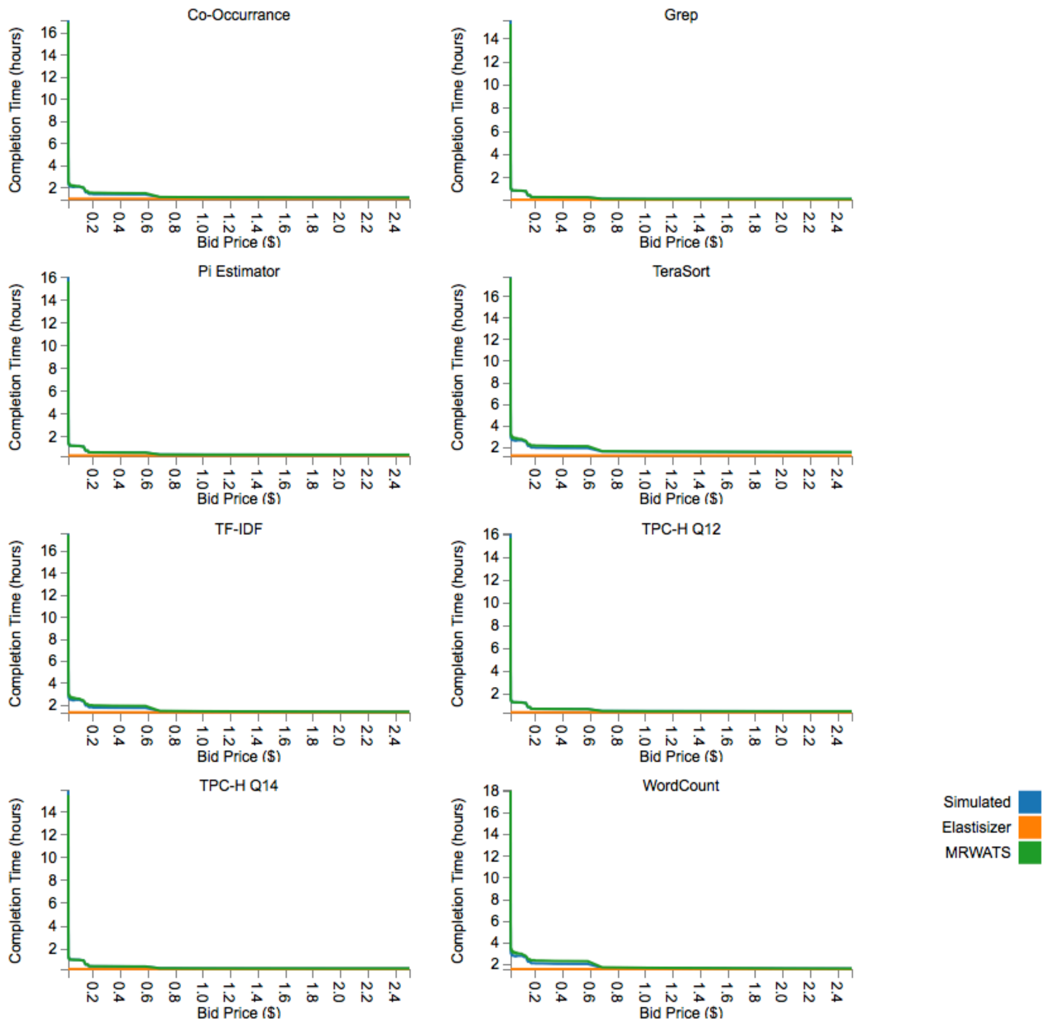

Figure 22: Predicted and actual completion times versus bid price for the c1.medium trace with 50% accelerators.

While the Elastisizer performed best with large bids, the low bid completion times are more relevant to the user because the optimal bid prices are below $0.58. Figure 22 shows the completion time verses bid price. Compared to the Elastisizer, MRWATS performed better at bid prices under $0.58 because the low bids contains greater wait times and failures. While the user waits for the auction to allocate instances at low bids, the low bids also provide the lowest average workload costs.

At high bids, MRWATS outperformed the Elastisizer's completion time prediction for I/O bound workloads because MRWATS models the workloads cost statistics with accelerator specific virtual profiles. The Elastisizer assumes the execution time remains constant regardless of the cluster's accelerator ratio. In contrast, MRWATS predicts the workload degradation with additional accelerators. When the TeraSort workload contained no failures or wait times ($2.50 bid), MRWATS reduced the completion time prediction error from 20.93% to 0.29%.

Table 40: Completion Time Error for C1.Medium with 50% Accelerators by Bid Price

	Average Completion Time (hrs.)			Ave CT % Error	
Bid	Simulated	Elastisizer	MRWATS	Elastisizer	MRWATS
Co-Occurrence					
0.0180	17.13	1.05	16.93	93.90	1.17
0.1300*	2.04	1.05	2.04	48.64	0.42
0.3425**	1.45	1.05	1.56	27.78	7.77
2.5000	1.15	1.05	1.15	8.82	0.09
Grep					
0.0180	15.59	0.09	15.29	99.39	1.94
0.0185	3.72	0.09	3.48	97.46	6.41
2.5000	0.15	0.09	0.15	38.42	0.00
Pi Est.					
0.0180	16.05	0.38	15.68	97.66	2.31
0.0500*	1.23	0.38	1.23	69.40	0.25
0.1300**	1.16	0.38	1.15	67.66	0.91
2.5000	0.44	0.38	0.44	14.23	0.08
TF-IDF Job1					
0.0180	17.60	1.37	17.51	92.21	0.52
0.1300*	2.40	1.37	2.43	42.76	1.58
0.2000**	1.80	1.37	2.00	23.71	11.27
2.5000	1.41	1.37	1.41	2.88	0.39
TPCH Q12					
0.0180	16.08	0.37	15.71	97.70	2.31
0.0500**	1.25	0.37	1.25	70.30	0.36
0.1300*	1.18	0.37	1.17	68.64	0.90
2.5000	0.46	0.37	0.45	18.66	0.08
TPCH Q14					
0.0180	15.87	0.26	15.49	98.39	2.40
0.0500*	1.07	0.26	1.07	76.21	0.38
0.1300**	1.01	0.26	1.00	74.70	0.98
2.5000	0.30	0.26	0.30	14.99	0.07
TeraSort					
0.0180	17.79	1.28	17.71	92.81	0.43
0.1300*	2.58	1.28	2.63	50.38	2.04
0.1715**	2.02	1.28	2.21	36.65	9.55
2.5000	1.60	1.28	1.60	20.13	0.29
Word Count					
0.0180	18.00	1.62	18.04	91.01	0.20
0.1300*	2.72	1.62	2.81	40.49	3.36
0.5800**	2.09	1.62	2.34	22.53	12.16
2.5000	1.67	1.62	1.66	2.82	0.39

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

While completion time aids in the cloud user's decision to use spot instances, spot instance users sacrifice completion time to reduced the workload's average price. Spot instance users require monetary cost estimates to determine if the potential financial savings is worth the extra completion time. MRWATS improves upon the Elastisizer's monetary cost estimates because MRWATS factors in early termination recovery costs and the average market price into the monetary cost estimates. With both factors baked into the monetary cost estimate, MRWATS reduced the Elastisizer's average cost prediction error by 99.1%.

MRWATS improve the monetary cost prediction accuracy at the low and high bids for different reasons. MRWATS outperforms the Elastisizer cost accuracy for low bids because the spot instance auction terminates spot instances more frequently at low bids (Chohan et al., 2010). The Elastisizer under-predicts costs at low bids because it does not factor in the extra run time for early terminated spot instances, Figure 23. When the user bids at \$0.018, the MRWATS's average error ranged from 0.6% to 16%. For the same bid, the Elastisizer's average error ranged from 31% to 41%, Table 41.

MRWATS improved the prediction accuracy on high bids because it calculates the average costs for a given bid instead of the maximum possible market price. While an individual run might allocate the spot instance at the bid price, the auction allocates the workload at a much lower market price on average. When the Elastisizer assumes the bid and market price are the same, the Elastisizer over predicts the average workload cost with a prediction error up to 907.48% at the \$2.50 bid, Table 41. MRWATS outperforms the naïve approach at the \$2.50 bid with a maximum prediction error of only 1.50%.

Table 41: Monetary Cost Error for C1.Medium with 50% Accelerators by Bid Price

	Average Cost ($)			Ave Cost % Error	
Bid	Simulated	Elastisizer	MRWATS	Elastisizer	MRWATS
Co-Occurrence					
0.0180	3.84	2.56	4.02	33.34	4.89
0.1300*	3.26	4.31	3.25	32.27	0.29
0.3425**	3.23	7.65	3.46	136.84	7.30
2.5000	4.39	41.48	4.37	844.33	0.55
Grep					
0.0180	0.38	0.23	0.38	38.81	0.64
0.0185	0.38	0.23	0.38	38.40	0.13
2.5000	0.59	3.76	0.59	535.39	0.83
Pi Est.					
0.0180	1.41	0.92	1.28	34.84	9.50
0.0500*	1.19	1.10	1.17	7.64	1.69
0.1300**	1.21	1.55	1.15	28.63	4.88
2.5000	1.70	14.91	1.67	779.94	1.48
TF-IDF Job1					
0.0180	4.90	3.35	5.29	31.52	8.12
0.1300*	4.10	5.66	4.15	38.06	1.26
0.2000**	4.04	7.10	4.42	75.69	9.50
2.5000	5.41	54.40	5.36	905.83	0.86
TPCH Q12					
0.0180	1.46	0.91	1.33	38.10	9.18
0.0500**	1.24	1.08	1.22	12.26	1.43
0.1300*	1.25	1.53	1.19	22.18	4.67
2.5000	1.76	14.70	1.74	734.38	1.50
TPCH Q14					
0.0180	1.00	0.62	0.84	37.41	15.63
0.0500*	0.82	0.75	0.79	8.91	4.19
0.1300**	0.83	1.05	0.78	26.44	6.84
2.5000	1.16	10.12	1.14	775.78	1.06
TeraSort					
0.0180	5.32	3.13	5.72	41.16	7.69
0.1300*	4.54	5.28	4.61	16.33	1.72
0.1715**	4.50	6.07	4.90	34.91	8.89
2.5000	6.13	50.75	6.09	727.50	0.71
Word Count					
0.0180	5.79	3.96	6.44	31.70	11.17
0.1300*	4.86	6.68	5.02	37.45	3.31
0.5800**	4.80	17.60	5.34	266.92	11.25
2.5000	6.37	64.21	6.33	907.48	0.75

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

While MRWATS reduced the cost prediction error compared to the Elastisizer, the cost estimate tool should indicate the optimal bid price to minimize the workload's monetary costs. MRWATS's predicted optimal bid minimized the average cost better than the Elastisizer's optimal bid because MRWATS doesn't assume the lowest bid results in the lowest cost for the workload. For the c1.medium trace, low bids actually increased the average workload monetary costs due to early termination, Figure 23.

Although MRWATS did not predict the exact optimal bid for every workload, MRWATS optimal bids minimized the workloads average costs better than the Elastisizer's projected optimal bids. MRWATS predicted optimal bid increased the workloads' cost by only 0.6% to 3% over the simulated optimal bid. Comparatively, the Elastisizer's predicted optimal bid increased cost by 20% to 30% because the Elastisizer assumed the lowest possible bid ($0.018) was optimal. The Elastisizer's bid strategy increased the workload average costs because the c1.medium test trace contained a high failure probability at low bids ($0.018), Figure 23.

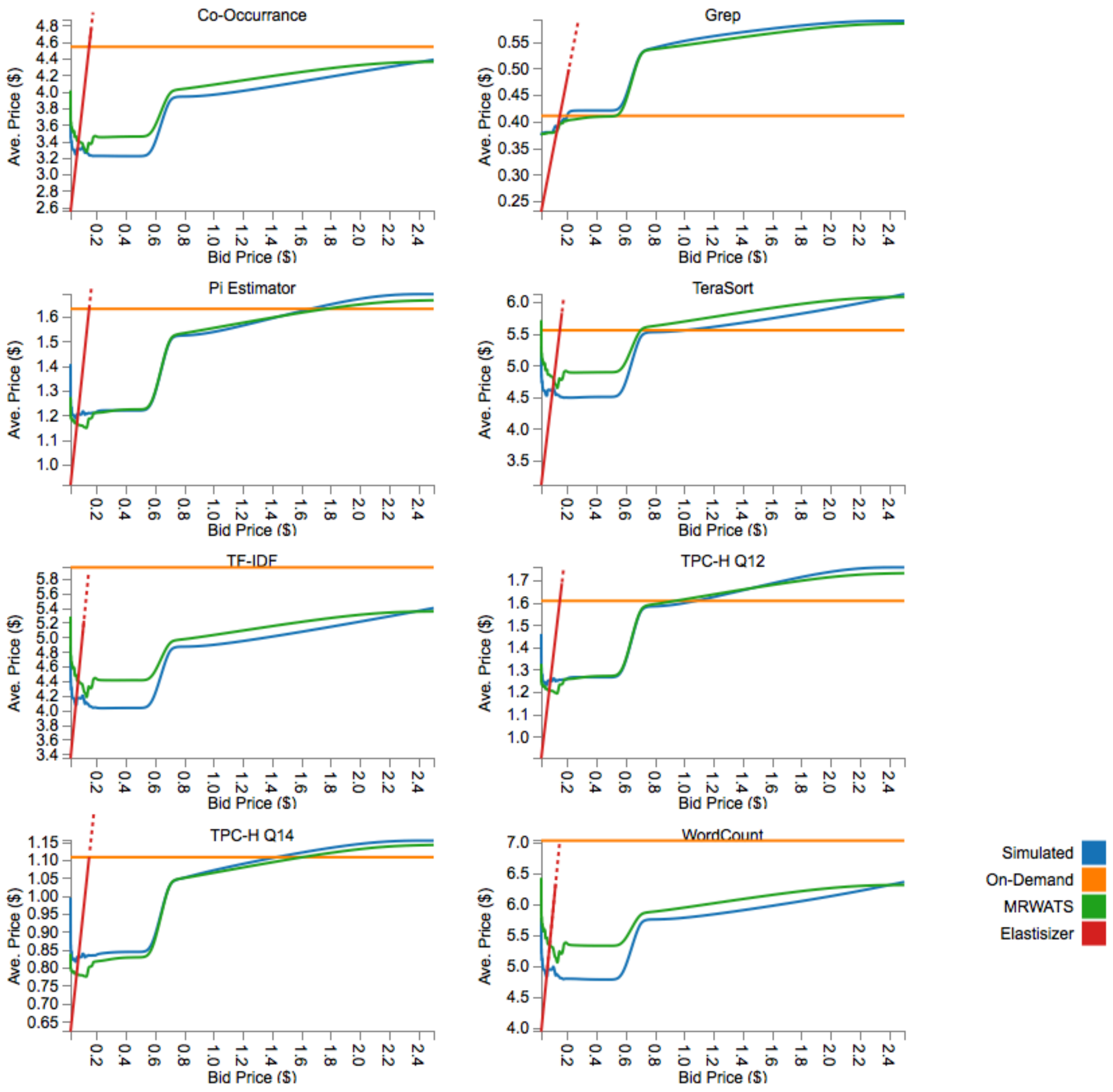

Figure 23. Actual (simulated) average, predicted average, and on-demand costs versus bid price for the c1.medium trace with 50% accelerators.

C1 Medium Trace – Ratio Selection Data Analysis

Although a cost estimation tool should accurately predict the workload's completion time and monetary costs, the cost estimate tool should also select the best cluster configuration (Herodotou et al., 2011). Where as the Elastisizer compares different virtual machine types, MRWATS compares different accelerator ratios. MRWATS improves upon the Elastisizer because MRWATS adapts the monetary cost predictions based on market conditions and workload characteristics. For a given spot market history, the cloud user should be able to compare different accelerator ratio configurations against an on-demand cluster, Figure 24.

When an estimation tool compares accelerator ratio configurations, the tool also needs to consider the bid price because the bid price impacts the workload's cost (Chohan et al., 2010). MRWATS aids the cloud user to decide on both the bid price and accelerator ratio because it factors in recovery costs and the average market price to the average workload cost. Where as the Elastisizer assumes the lowest bids translated to the lowest average workload costs, MRWATS indicates a cost-effective bid range based on the price history.

MRWATS helps user select a configuration better than the Elastisizer because MRWATS enables cloud users to select bids that avoid frequent terminations while minimizing the average market price. For the c1.medium trace, the cloud user could beat the on-demand workload price with bids from \$0.10 to \$0.50 regardless of the accelerator ratio, Figure 24. In contrast, a cloud user would miss potential cost savings with the Elastisizer because the Elastisizer doesn't factor in the market price to the cost estimates. The Elastisizer would bias towards on-demands instances for the same bid range because the \$0.50 bid is much larger than the on-demand price, \$0.145.

MRWATS helps user select an accelerator configuration better than the Elastisizer because MRWATS adapts the monetary cost predictions to the workloads characteristics. The Elastisizer always assumes the highest accelerator ratio provides the best monetary cost saving because the user allocates a greater percentage of nodes at a cheaper price. In contrast, MRWATS adjusts the accelerator ratio recommendation based on the variable-ratio virtual profiles.

Once MRWATS estimates the monetary cost based on the bid, it enables the cloud user to determine the best accelerator ratio for a given workload. The accelerator ratio's impact differs based on the workload's characteristics. If the workload is CPU bound, then the highest accelerator ratio provided the largest cost reduction because the workload's execution time remains constant as accelerators are added. For example, the co-occurrence workload's execution time does not degrade as the accelerator ratio changes, Figure 19. The cloud user should provision a cluster with 20 accelerators for the Co-Occurrence workload because 20 accelerators reduced the average workload cost by 37% compared to a 15% monetary cost reduction from 10 accelerators, Figure 24.

If the workload is I/O bound, then the cloud user should avoid high accelerator ratios because high ratios increase the workloads' monetary cost due to extra runtime. I/O bound workloads execute longer as the accelerator ratio increases, Figure 19. If the workload takes longer to run, then the user must pay for the extra time. For example, the TPC-H Q14 workload's costs increased at the high accelerator ratio, Figure 24. The TPC-H Q14 workload costs an average four cents more on a 67% ratio than a 50% ratio because the workload took 30 minutes longer to execute.

MRWATS provided better ratio recommendations than the Elastisizer because MRWATS predicts the extra runtime from additional accelerators. For the TPC-H Q14 workload, MRWATS recommended the 50% ratio because the 67% ratio's extra runtime increased the workloads costs, Figure 24. In contrast, the Elastisizer selected the wrong ratio, 67%, for the TPC-H workload because the Elastisizer doesn't adjust the execution time estimates based on the accelerator ratio.

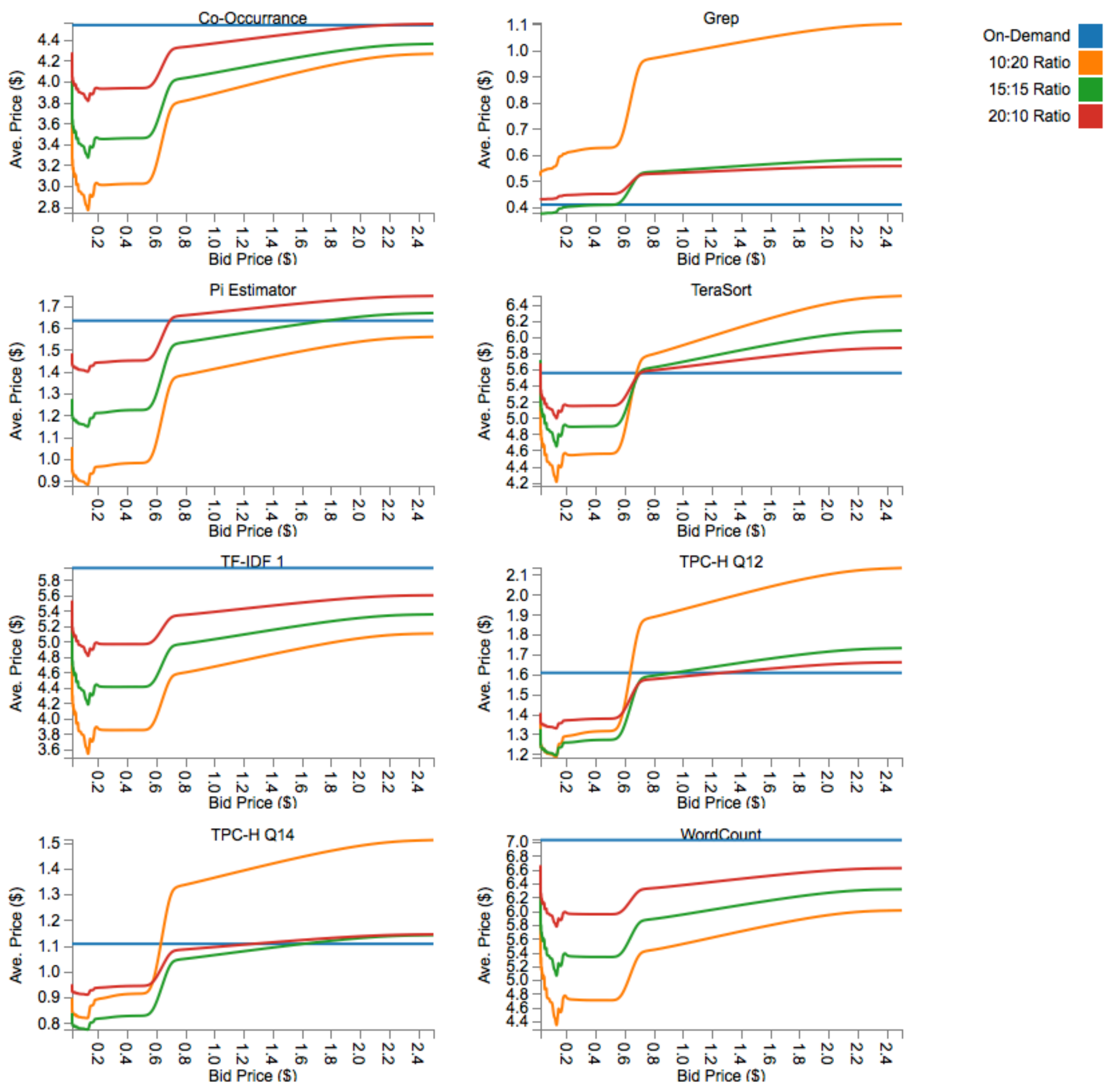

Figure 24: MRWATS predicted average costs by accelerator ratio versus the bid price for c1.medium trace compared to the on-demand cost.

M1 Large Trace Data Analysis

The m1.large experiments evaluated MRWATS's prediction accuracy compared to the Elastisizer against the m1.large spot market history trace. While the experiment changed instance types, the price history trace also exhibited different features compared to the c1.medium evaluation. Although the c1.medium trace contained frequent price transitions, the m1.large trace provided more periodic price transitions. The workloads low bids did not wait as long to allocate the cluster. The m1.trace also provided fewer opportunities for early termination.

Given the new price history trace, MRWATS still improved upon the Elastisizer because MRWATS estimates the completion time and cost based on the market price history. Although the m1.large trace contains shorter wait times and fewer early terminations, both still impacted the workloads completion time and monetary costs. Where as the Elastisizer is insensitive to the spot market impacts on the workload, MRWATS factors in past wait times, early terminations, accelerator ratio, and market prices into its' completion time and cost predictions.

While the m1.large trace represented regular market patterns, MRWATS still demonstrated an improvement over the Elastisizer. MRWATS reduced both the average completion time and cost error compared to the Elastisizer for all workloads, Table 42. MRWATS reduced the average completion time error by 95.6% (from 30.01 to 1.72 percent error) compared to the Elastisizer. MRWATS also reduced the average cost error by 98.2% (from 199.01 to 3.53 percent error) compared to the Elastisizer.

Table 42: M1.Large Completion Time and Cost Error by Workload & Ratio

Ratio	Ave. Completion Time % Error		Ave. Cost % Error	
	Elastisizer	MRWATS	Elastisizer	MRWATS
Co-Occurrence				
0.33	24.39	0.66	162.42	2.24
0.50	27.27	1.79	228.49	3.49
0.67	32.57	3.65	294.83	5.00
Grep				
0.33	73.46	2.47	62.21	1.97
0.50	77.45	2.43	70.20	2.47
0.67	80.30	2.22	76.20	2.95
Pi Est.				
0.33	32.95	0.27	156.02	2.84
0.50	34.45	1.44	225.18	4.63
0.67	37.11	3.94	295.92	7.00
TF-IDF Job1				
0.33	24.17	0.56	145.72	1.37
0.50	28.07	0.85	211.91	2.34
0.67	34.12	1.77	280.33	3.52
TPCH Q12				
0.33	31.64	0.41	180.15	3.35
0.50	34.70	1.79	218.61	4.48
0.67	42.40	3.67	254.33	5.51
TPCH Q14				
0.33	41.44	1.34	207.14	3.21
0.50	39.63	0.80	236.38	4.76
0.67	46.50	2.14	264.07	6.62
TeraSort				
0.33	32.53	0.35	132.52	1.90
0.50	37.92	1.67	181.77	3.46
0.67	43.46	3.08	233.32	4.82
Word Count				
0.33	21.29	0.76	151.24	1.37
0.50	25.76	1.09	218.93	2.20
0.67	32.58	2.10	288.25	3.21
Average	39.01	1.72	199.01	3.53

M1 Large Trace - 50% Accelerator Ratio Data Analysis

Although the overall results demonstrated that MRWATS improved the completion time and cost prediction accuracy, the 50% accelerator results further explain the differences between MRWATS and the Elastisizer. MRWATS improves completion time and cost estimates because it estimates the spot market specific impacts on both measures. When the simulation tested the 50% accelerator ratio with the m1.large trace, MRWATS reduced the completion time prediction error by 96.11% and the cost prediction error by 98.25% over the Elastisizer's predictions. Appendix B details similar results for the 33% and 67% accelerator ratios.

Like the c1.medium trace analysis, this section analyzes the completion time and monetary cost prediction accuracy for low and high bids. MRWATS improved upon the Elastisizer at low bids because MRWATS predicts the completion time and monetary costs based on the wait and recovery times. At high bids, MRWATS improves upon the Elastisizer by factoring in the accelerator ratio and average market price to the completion time and monetary cost predictions.

At low bids, the m1.large trace impacted the workload's completion time slightly differently than the c1.medium trace. Much like the c1.medium trace, the m1.large's market prices impacted the workload's completion time. Low bids increased the workloads' average completion time because the low bid workloads incurred additional wait times and early terminations, Table 43. Although the test workloads still experienced wait times and early terminations, the m1.large trace produced about 10x shorter completion times for the low bids than the c1.medium trace. The m1.large workloads completed faster at low bids because the m1.large contained larger stretches of low demand periods compared to the c1.medium trace.

Table 43: Completion Time Error for M1.Large with 50% Accelerators by Bid Price

	Average Completion Time (hrs.)			Ave CT % Error	
Bid	Simulated	Elastisizer	MRWATS	Elastisizer	MRWATS
Co-Occurrence					
0.0260	1.88	0.92	1.90	51.25	0.86
0.0500**	1.84	0.92	1.81	50.20	1.82
0.2000*	1.80	0.92	1.74	49.16	3.53
3.3000	0.91	0.92	0.91	0.22	0.10
Grep					
0.0260* **	0.90	0.08	0.95	91.42	5.32
3.3000	0.19	0.08	0.19	58.99	0.00
Pi Est.					
0.0260	1.32	0.49	1.33	63.08	0.32
0.0300**	1.32	0.49	1.32	62.95	0.01
0.2000*	1.23	0.49	1.20	60.39	2.69
3.3000	0.50	0.49	0.50	1.74	0.10
TF-IDF Job1					
0.0260	2.83	1.47	2.87	47.86	1.50
0.2000*	2.68	1.47	2.66	45.06	0.90
0.2410**	2.61	1.47	2.59	43.49	0.69
3.3000	1.58	1.47	1.57	6.74	0.37
TPCH Q12					
0.0260	1.40	0.53	1.40	61.92	0.06
0.0500**	1.36	0.53	1.33	60.91	2.55
0.2000*	1.31	0.53	1.27	59.35	3.15
3.3000	0.55	0.53	0.55	3.72	0.10
TPCH Q14					
0.0260	1.07	0.30	1.09	72.00	1.23
0.0800* **	0.98	0.30	0.99	69.17	1.31
3.3000	0.30	0.30	0.30	0.48	0.09
TeraSort					
0.0260	1.99	0.85	1.98	57.44	0.52
0.0500**	1.93	0.85	1.90	56.15	1.86
0.2000*	1.90	0.85	1.83	55.33	3.52
3.3000	1.02	0.85	1.02	16.86	0.09
Word Count					
0.0260	3.23	1.76	3.32	45.39	2.70
0.2000*	3.09	1.76	3.07	42.95	0.54
0.2400**	3.01	1.76	3.00	41.50	0.33
3.3000	1.84	1.76	1.83	4.02	0.57

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

While the m1.large trace produced shorter completion times, MRWATS still improved the completion time prediction accuracy compared to the Elastisizer because MRWATS factors in wait and recovery time to the completion time estimate. The bid price impacted each tools' prediction accuracy, Table 43. To analyze the bid impact on completion time, this section will analyze each tool's completion prediction accuracy at low and high bids.

When the workloads were submitted with low bids, the spot market still caused some workloads to wait for allocations and or terminated the workload early, Figure 25.

MRWATS improved upon the Elastisizer because MRWATS included the average wait and recovery time in the completion time prediction. MRWATS completion time error ranged from 0.06% to 5.32%. In contrast, the Elastisizer performed worse than MRWATS with a 45.39% to 91.42% error because the Elastisizer's predictions didn't include the wait and recovery times, Table 43.

When the simulation submitted the workload with the maximum bid ($3.30), the workloads didn't contain any wait times or failures. However, MRWATS still improves upon the Elastisizer at the high bids because MRWATS predicts the completion time based on the workload's I/O characteristics and accelerator ratio. For instance, MRWATS and Elastisizer predicted almost exactly the same completion time on the co-occurrence workload because the co-occurrence workload creates very little HDFS contention, Figure 25. In contrast, MRWATS perform better than the Elastisizer on the I/O bound workloads because MRWATS models the file system contention. While the Elastisizer underestimated TeraSort's completion time by 10 minutes, MRWATS predicted the TeraSort's completion time almost exactly, Figure 25.

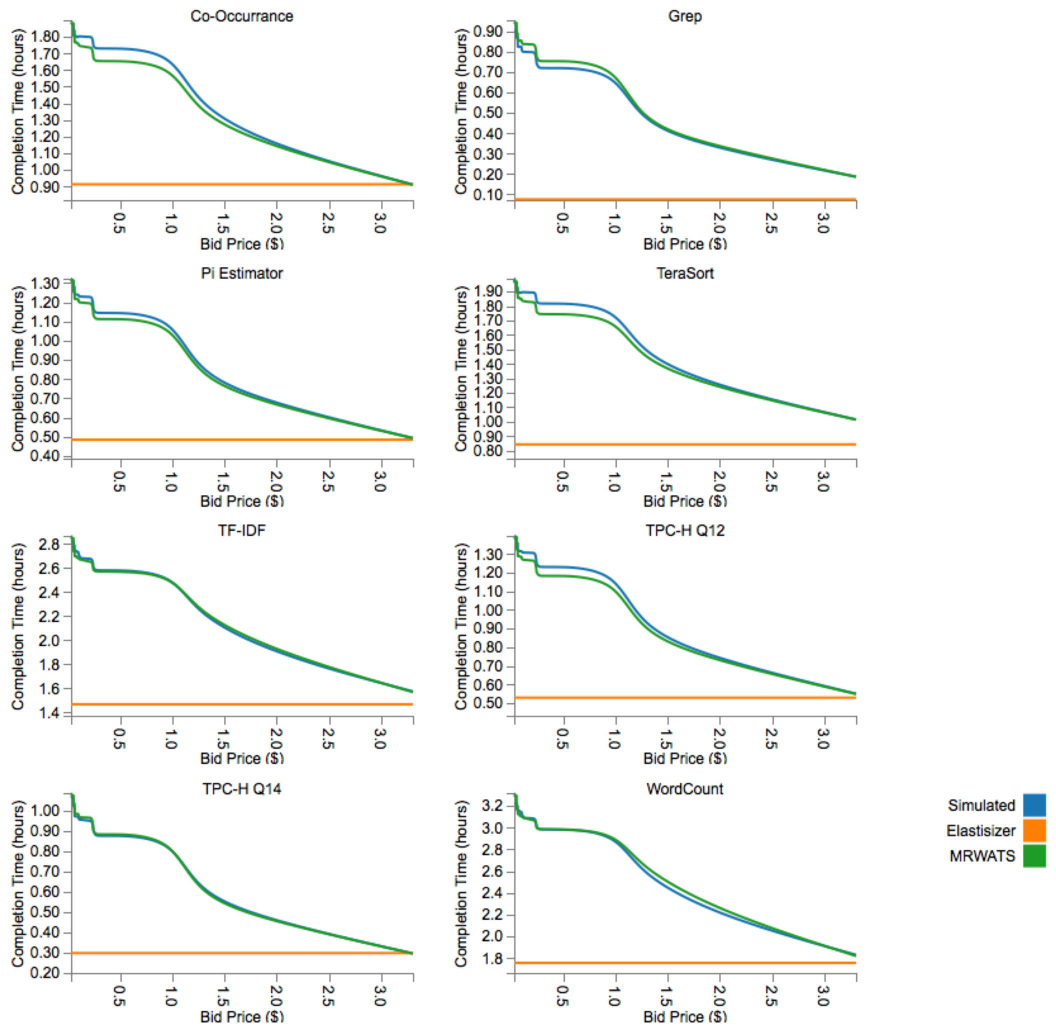

Figure 25. Predicted and actual (simulated) completion times versus bid price for the m1.large trace with 50% accelerators.

While MRWATS predicted the completion time based on the m1.large trace, the spot instance user also requires monetary cost estimates to determine if the extra processing time is worth the wait. MRWATS improves upon the Elastisizer monetary cost estimates because MRWATS factors in early termination recovery costs and the average market price into the estimates. Although the m1.large contains fewer early terminations than the c1.medium trace, the early terminations and the average market price still impacted the workload's monetary costs. Both factors translated into an average 98.16% prediction error reduction by MRWATS. To better examine MRWATS monetary cost improvements, this section will examine monetary cost at low and high bids.

When the simulation bid at the minimum price ($0.026), MRWATS improves upon the Elastisizer's cost predictions because the MRWATS factors in the monetary cost from early termination recovery. While the m1.large trace contains fewer early terminations than the c1.medium trace, the m1.large terminations did impact the workload costs, Figure 26. The Elastisizer under-predicted the workload's costs between 13.24% and 58.25% because it underestimates the workload's execution time, Table 44. Comparatively, the MRWATS's monetary cost error only ranged from 0.10% to 8.06% because the execution time estimates include reprocessing time for failed tasks.

When the simulation bid at the maximum price ($3.30), MRWATS improves upon the Elastisizer's cost predictions because MRWATS factors in the average market price into the cost estimates. When the spot instance auction allocates the instances, users only pay the market price for spot instances and not the bid price (Wieder et al., 2012). The Elastisizer performed worse than MRWATS at high bids because the naïve approach assumes the bid price equals the market price. The Elastisizer over-predicted the workload's cost by 91% to 364% at the maximum bid, Table 44. MRWATS performed much better at the same bid with the prediction error ranging from 0.10% to 3.57%.

While MRWATS reduced the cost prediction error compare to the Elastisizer, the cost estimate tool should indicate the optimal bid price to minimize the workload's monetary costs. MRWATS improves upon the Elastisizer's bid strategy because MRWATS doesn't assume the minimum bid results in the workload's lowest average cost. While MRWATS did not predict the exact optimal bid for every workload, MRWATS's predicted optimal bid only increased cost by 3.41% on average. Comparatively, the Elastisizer's projected optimal bid under-estimated the costs by 24.83% on average.

Table 44: Monetary Cost Error for M1.Large with 50% Accelerators by Bid Price

	Average Cost ($)			Ave Cost % Error	
Bid	Simulated	Elastisizer	MRWATS	Elastisizer	MRWATS
Co-Occurrence					
0.0260	4.44	3.66	4.35	17.69	2.08
0.0500**	4.43	3.99	4.26	10.06	3.89
0.2000*	4.54	6.05	4.23	33.14	6.97
3.3000	10.78	48.67	10.79	351.45	0.10
Grep					
0.0260* **	0.74	0.31	0.75	58.25	0.97
3.3000	2.16	4.12	2.24	90.93	3.57
Pi Est.					
0.0260	2.33	1.95	2.19	16.43	6.01
0.0300**	2.32	1.98	2.19	14.71	5.64
0.2000*	2.38	3.22	2.16	35.34	9.39
3.3000	5.79	25.90	5.86	347.70	1.25
TF-IDF Job1					
0.0260	7.98	5.88	7.97	26.35	0.10
0.2000*	7.87	9.72	7.69	23.62	2.23
0.2410**	7.82	10.63	7.74	35.93	1.08
3.3000	18.68	78.24	18.59	318.81	0.47
TPCH Q12					
0.0260	2.62	2.12	2.47	18.92	5.78
0.0500**	2.61	2.32	2.43	11.33	6.78
0.2000*	2.67	3.51	2.42	31.37	9.36
3.3000	6.46	28.26	6.52	337.76	1.03
TPCH Q14					
0.0260	1.38	1.20	1.27	13.24	8.06
0.0800* **	1.32	1.44	1.26	9.31	4.50
3.3000	3.44	15.97	3.53	364.01	2.64
TeraSort					
0.0260	4.87	3.38	4.67	30.54	3.97
0.0500**	4.79	3.69	4.61	23.07	3.87
0.2000*	4.91	5.59	4.58	13.98	6.68
3.3000	12.01	44.99	12.03	274.59	0.14
Word Count					
0.0260	9.49	7.04	9.63	25.84	1.50
0.2000*	9.39	11.64	9.24	23.91	1.58
0.2400**	9.34	12.69	9.28	35.95	0.57
3.3000	21.70	93.62	21.57	331.39	0.60

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

While prediction accuracy enables a user to cost-effectively bid, MRWATS also improves upon the Elastisizer by allowing users to balance price with completion time. In essence, users will want to minimize wait time while executing the workload at a financial savings compared to the on-demand costs. When a user bids up to $1.00, MRWATS shows the actual cost only varies about 6% for the m1.large trace, Figure 26. Yet, MRWATS also shows a $1.00 bid reduced the average completion time by 15 minutes for most workloads, Figure 25. These subtle tradeoffs aren't expressed with the Elastisizer because the original tool doesn't adjust the completion time and cost predictions based on the spot market history.

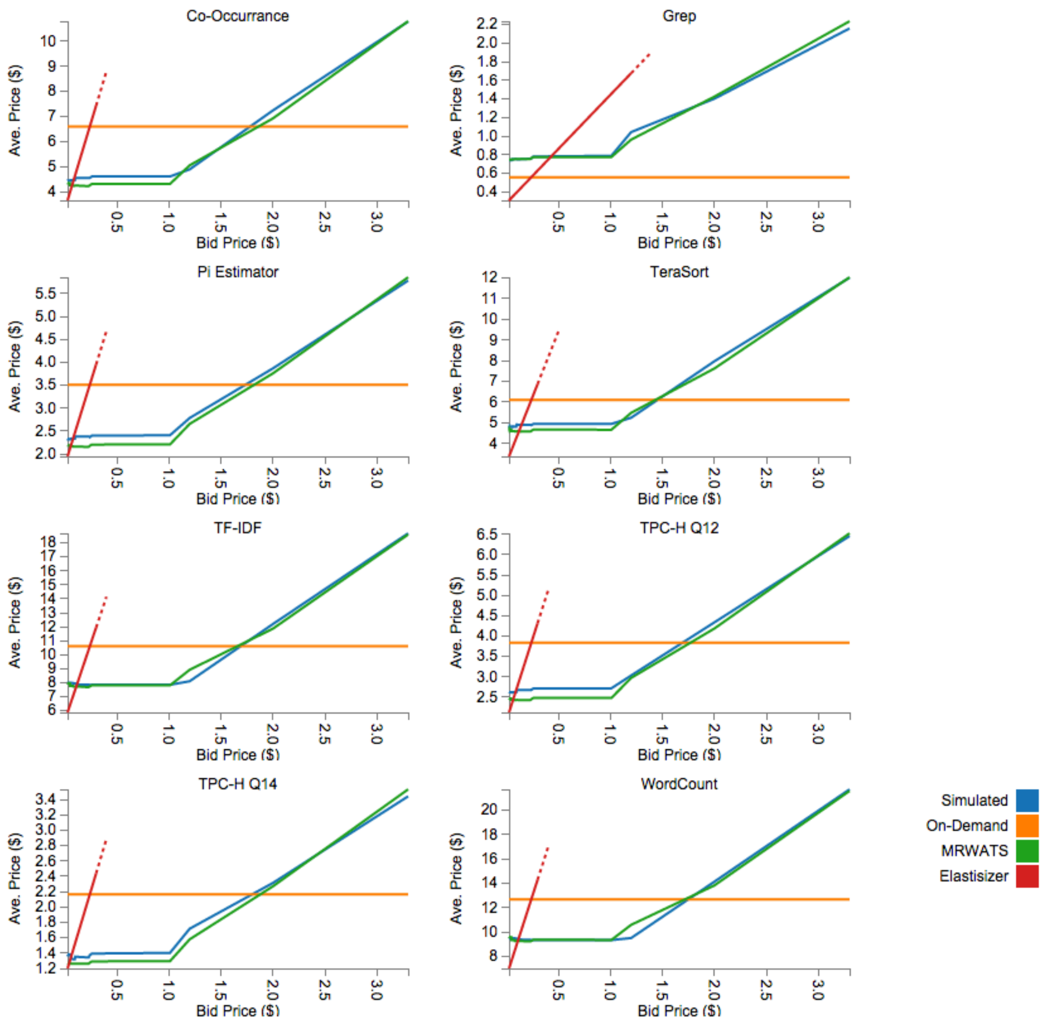

Figure 26: Actual (simulated) average, predicted average, and on-demand costs versus bid price for the m1.large trace with 50% accelerators.

M1 Large Trace – Ratio Selection Data Analysis

As noted with the c1.medium trace, cloud users need to compare different accelerator ratios and a fully on-demand cluster to find the best configuration for their workload. Spot instance users look to minimize the average monetary cost given some minimally acceptable average completion time. The best accelerator configuration depends on both bid price and the workloads own characteristics.

Although the m1.trace terminated fewer workloads than the c1.medium trace, the best configuration depended on the bid price. MRWATS improves upon the Elastisizer because MRWATS allows users to compare accelerator ratios based on the bid price. If the user bids below $1.00, then MRWATS indicated the 10:20 ratio provided the most cost-effective accelerated configuration in most cases, Figure 27. If the user bid greater than $1.50, then MRWATS indicated the user should have allocated all on-demand instances because the average market price was greater than the on-demand price at high bids, Figure 27. In comparison, the Elastisizer might unnecessarily dissuade users from bidding higher because the Elastisizer switches to all on-demand instances with a $0.24 bid, Figure 26.

While additional accelerators reduced the financial cost for most workloads executed against the m1.trace, the cost saving depended on the workload's I/O characteristics. MRWATS provides a better cost comparison then the Elastisizer because MRWATS models the workloads I/O degradation with additional accelerators. MRWATS shows that CPU bound workloads provided a greater cost savings then I/O bound workloads. For instance, the co-occurrence workload reduced cost by 43% compared to TeraSort's 33% reduction, Figure 27. If the workload was highly HDFS bound (i.e. grep), then the on-demand cluster provided the most cost effective configuration because the accelerators increased the workload's execution time, Figure 27.

Once MRWATS finds the most cost effective accelerator configuration for a virtual machine instance type, the cloud user still needs to compare virtual machine types. When the user compares virtual machine types, MRWATS improves upon the Elastisizer for the instance type selection because MRWATS incorporates the market price into the decision. If the m1.large forecasts, Figure 27, are compared against the c1.medium forecasts, Figure 24, then the c1.medium instances provided the lowest cost configuration. The c1.medium trace allocated spot instances at a lower average market price ($0.1236) compared to the m1.large trace ($0.2676).

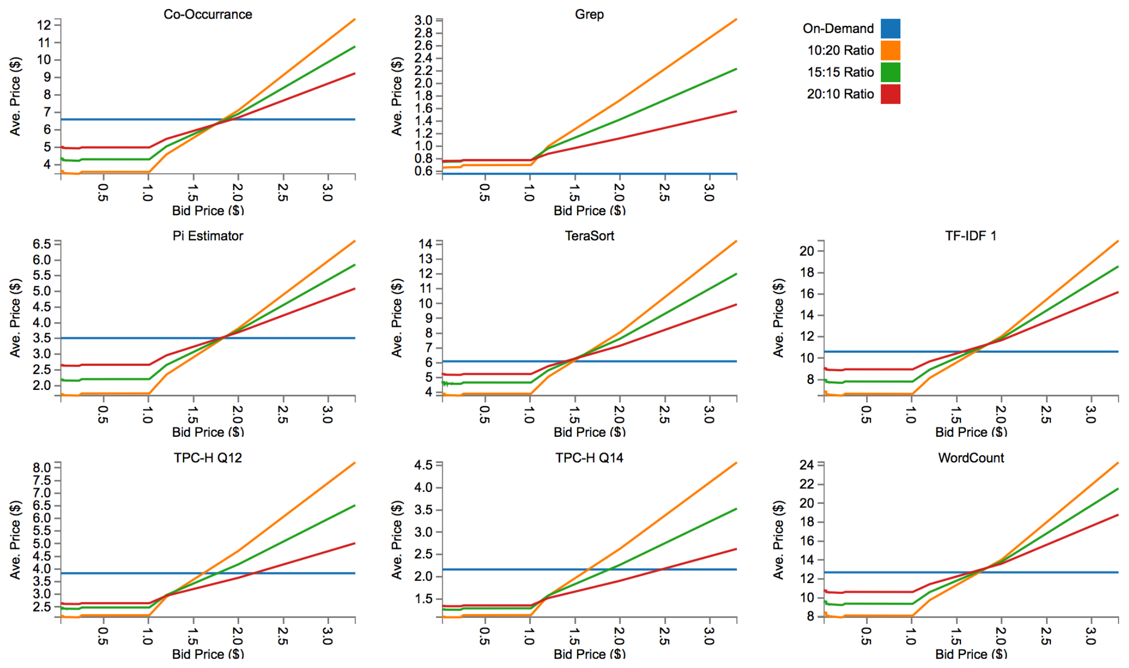

Figure 27. MRWATS predicted average cost by accelerator ratio versus the bid price for the m1.large trace compared to the on-demand cost.

Findings

Cloud users may consider spot market resources as a means to reduce the workload's monetary costs. Spot instances provide a means to reduce a workload's monetary cost because a preemptive auction allocates the instances (Chohan et al., 2010).

The spot instance's price varies based on demand. When the auction allocates the instances during low-demand periods, the auction offers the spot instances at a reduced price compared to the on-demand price. When the auction allocates the instances during high-demand periods, the auction offers the spot instances at a price greater than the on-demand price.

This dissertation created a tool, MRWATS, to determine the spot market's impact on a map-reduce workload. MRWATS improves upon the Elastisizer because MRWATS enables the user to compare the workload's average completion time and monetary costs for different bid prices and accelerator configurations. In order to model the spot markets' impact on completion time and monetary costs, MRWATS created three enhancements to improve completion time and monetary cost prediction accuracy over the Elastisizer: storage strategy specific virtual profiles, a failure-aware task simulator, and spot market forecasts.

MRWATS improved upon the Elastisizer because MRWATS includes storage specific virtual profiles that model each workload's file system contention. Specifically, MRWATS enhanced the Elastisizer virtual profiles by generating the profiles from file system specific relative fitness models and including additional features to model file system contention. MRWATS enhanced the virtual profiles for two storage strategies: Amazon S3 and spot instance accelerators.

Although both storage strategies generated file system contention differently, MRWATS demonstrated improved prediction accuracy for both strategies. MRWATS improved the execution time prediction accuracy for the Amazon S3 file system because it modeled file system contention generated by concurrent tasks accessing the remote file system. While MRWATS reduced the prediction error compare to the Elastisizer at small-scale, MRWATS's relative accuracy improved with larger cluster sizes. When the evaluation tested workloads at 10-nodes, MRWATS reduced the average prediction error by 23%. When the evaluation scaled up to 90 nodes, MRWATS reduced the average prediction error by 35%.

In a similar fashion, MRWATS improved the execution time prediction accuracy for an HDFS accelerated cluster because it models the file system contention generated by different accelerator ratios. MRWATS demonstrated improved prediction accuracy relative to the Elastisizer as the accelerator ratio increased. When the cluster contained only 33% accelerators, MRWATS reduced the prediction error compared to the Elastisizer by 18.54% and 13.96% for the c1.medium and m1.large instance types, respectively. When the cluster contained 83% accelerators, MRWATS further reduced the prediction error compared to the Elastisizer by 57.95% and 35.43% for the c1.medium and m1.large instance types, respectively.

While MRWATS aided the cloud user to estimate the storage strategy's impact on a workload's execution time, the prediction tool should also provide accurate estimates when spot market early termination occurs. MRWATS improves upon the Elastisizer during spot instance early termination because MRWATS enhanced the task scheduler and virtual profiler to handle failures. The enhanced scheduler demonstrated improved prediction accuracy over the Elastisizer scheduler during different failure points in a series of workloads. MRWATS reduced the prediction error compared to the Elastisizer by 76.12% on average across the different failure scenarios.

While MRWATS improved the early termination prediction accuracy compare to the Elastisizer, cloud users won't know when the spot market auction will allocate or terminate the spot instances (Chohan et al., 2010). The cloud user requires bid-based estimates to minimize costs by avoiding overly frequent early terminations. The Elastisizer lacks accurate bid-based estimates because it assumes the execution time remains constant and costs increase linearly with bid price changes. However, MRWATS improves upon the Elastisizer because MRWATS estimates the completion time and monetary costs based on bid and the spot market's prior history.

Specifically, MRWATS' forecasts improve upon the Elastisizer's execution time predictions because MRWATS completion time estimates include wait and recovery times based on the spot market's history. This dissertation demonstrated the enhancements improved the completion time prediction accuracy by comparing both tools' accuracy against two different spot market traces (c1.medium & m1.large). While both traces contain different wait times and termination frequencies, MRWATS reduced the completion time prediction error compared to the Elastisizer by 91.58% and 95.59% for the c1.medium and m1.large traces, respectively.

MRWATS also improved upon the Elastisizer's monetary cost estimates because MRWATS incorporates the average market price and potential recovery costs in the monetary estimates. This dissertation demonstrated that the enhancements improved the cost prediction accuracy by comparing both tools against two different spot market traces (c1.medium & m1.large). Although the two traces contained different early termination frequencies, MRWATS reduced the monetary cost prediction error compared to the Elastisizer by 99.14% and 98.23% for the c1.medium and m1.large traces, respectively.

Chapter 5

Conclusions

This dissertation demonstrated a cost estimation tool for virtualized map-reduce clusters using spot instances, MapReduce Workload Allocation Tool for Spot instances (MRWATS). The tool predicts a spot instance workload's cost and completion time. The tool enables a spot instance user to select a virtual cluster configuration and bid price to reduce the workload's costs while minimizing the impacts on completion time.

This work validated the MRWATS's prediction accuracy by comparing the completion time and monetary cost prediction error to a non-spot instance tool, the Elastisizer. If MRWATS reduced the prediction error compared to the Elastisizer, then the work achieved the dissertation goal. This work achieved the dissertation goal because MRWATS reduced the cost and completion time prediction error compared to the Elastisizer. Specifically, MRWATS reduced the prediction error for spot instance storage strategies, auction terminated workloads, and spot market forecasts.

MRWATS improved the execution time prediction accuracy for spot instance storage strategies over the Elastisizer. Chapter 4 evaluated both tools' prediction accuracy for the Amazon S3 and accelerated storage strategies. MRWATS performed better than the Elastisizer because MRWATS models file system contention for both Amazon S3 and accelerated storage strategies. When MRWATS predicted large Amazon S3 workloads, the updated model reduced the prediction error up to 35% compared to the Elastisizer. When MRWATS predicted workloads with high accelerator ratios, MRWATS reduced the prediction error up to 58% compared to the Elastisizer.

When the spot market auction terminates the spot instances early, MRWATS also improved upon the Elastisizer because MRWATS factors in the early termination's recovery time. Chapter 4 compared the prediction error between MRWATS and the Elastisizer during several workload failure points. MRWATS reduce the prediction error by 76.12% compared to the Elastisizer during early terminations.

Finally, MRWATS improved the completion time and financial cost prediction accuracy compared to the Elastisizer because MRWATS incorporates the spot market history into its predictions. While the tool's accuracy depends on the spot market trace (Wieder et al., 2012), MRWATS reduced the completion time and cost error compared to the Elastisizer for two traces with different market characteristics. When the evaluation tested a termination heavy market trace, MRWATS reduced the average completion time prediction error by 91.58% compared to the Elastisizer. MRWATS also reduced the average cost prediction error by 99.14%, for the same trace. When the evaluation tested a periodic market trace, MRWATS reduced the average completion time prediction error by 95.59% compared to the Elastisizer. MRWATS also reduce the average cost prediction error by 98.23% for the periodic trace.

Implications

While this dissertation purpose was to improve prediction accuracy over existing tools, this work expanded on past work's findings on spot markets. The study uncovered new issues because the study simulated workload execution times based on more recent spot market traces. The new traces change some previous assumptions because the recent traces did not behave the same way as Chohan et al. (2010) and Orna Agmon Ben-Yehuda et al. (2013) traces.

Chohan et al. (2010) suggested that map reduce user's should avoid low bids because the early Amazon's spot market histories terminated low bids frequently. However, cloud users should avoid general "rule's of thumb" because the workload's cost highly depends on the spot market's price dynamics (Wieder et al., 2012). Contrary to Chohan et al. (2010) traces, low bids don't necessarily cause a dramatic increase in the workload's total monetary costs. Although early termination impacted the workload's cost with low bids on the c1.medium trace, the m1.large trace only varied costs by 6% with low bids.

Conversely, high bids don't necessarily eliminate the potential cost savings over on-demand instances because the workloads' costs depend on the duration the workloads are repeatedly executed in the cloud environment. While Chohan et al. (2010) and Orna Agmon Ben-Yehuda et al. (2013) used price histories capped at the on-demand price, this study's price histories contained bids 10 times greater than the on-demand prices. The high bids increased the average workload cost by 2 times for the m1.large virtual machine type. While the short-term demand resulted in higher workload costs, cloud users benefit from workload's repeatedly executed over several months. Although the m1.large market price average 11% more than the on-demand price over the tested three week trace, the market price averaged 31% less than the on-demand price over a 3-month period.

While MRWATS improves the prediction accuracy compared to past tools, the forecast does not handle structural changes to the spot market. A spot market's behavior changes from auction algorithmic changes (Orna Agmon Ben-Yehuda et al., 2013) and or provider price wars (Orna Agmon Ben-Yehuda, Ben-Yehuda, Schuster, & Tsafrir, 2014;

Barr, 2014). This study evaluated MRWATS with a shortened one week prediction window compared to the Chohan et al. (2010) study because the price history radically changed over a 3-month period due to general Amazon price reductions (Barr, 2014). While the shortened window improved the tool's prediction accuracy, the shortened window also reduced the tool's value for long-term cost estimation.

Recommendations

Although MRWATS improved spot instance prediction accuracy compare to past tools, the tools contain several areas where prediction accuracy could be improved. While the evaluation assumed a uniform distribution for submission time, users may choose to submit workloads during business hours or other diurnal patterns (Armbrust et al., 2010). The user's bid strategy might change given different submission profiles. If the forecast includes the cloud user's desire submission patterns, then the forecast provides a more precise cost estimate than the average costs.

Although MRWATS derives the completion time and cost predictions from spot market histories, like Chohan et al. (2010) and Orna Agmon Ben-Yehuda et al. (2013), structural spot market changes reduce the cost estimation tool's accuracy (Orna Agmon Ben-Yehuda et al., 2013). MRWATS reduced the prediction window from one month (Chohan et al., 2010) to one week because the shortened window improved the tools accuracy. Additional commodity market research might lengthen the prediction window for virtual machine spot markets forecast (Sharma & Srinivasan, 2007).

Finally, further research should extend this work to other spot instance workloads beyond MapReduce. Researchers will create new workloads that function better in cloud and spot market environments than MapReduce workloads (Orna Agmon Ben-Yehuda et al., 2014). For instance, new frameworks, like Spark, improves the recovery time over MapReduce from failed nodes because Spark tracks the intermediate data lineage to avoid unnecessary re-computation (Zaharia et al., 2012). While Spark provides better fault tolerance, cloud users still need to compare alternative spot instance cluster configurations to find the most cost effective configuration for the Spark workload.

Summary

Users execute map-reduce workloads on cloud environments because users want to reduce the workload's monetary costs while meeting a deadline. Before the user executes the workload, the user must specify virtual cluster resources to meet both the completion time and budget objectives (Herodotou et al., 2011). Short-term resources, like spot instances, provide a potentially cost effective option for virtual cluster resources because the users can request the virtual machines at a reduced price (Chohan et al., 2010). However, the user must evaluate the spot instances' impact on the map-reduce workloads monetary costs and completion time to compare against other virtualized cluster alternatives.

When a cloud user allocates a spot instance virtual cluster, the user must decide on four factors: the virtual machine type, number of instances, pricing model, and data storage strategy. Each factor impacts the map-reduce workload's completion time and monetary cost.

The virtual machine type impacts the workload's execution time and monetary cost because the type governs the map-reduce tasks' compute rate. Map reduce tasks process data at different rates because the memory, CPU units, and I/O throughput changes between virtual machine types (Lee et al., 2011). While the virtual machine impacts individual task execution times, the number of virtual machines impacts the workload's total execution time. Additional virtual machines can reduce execution time without a price increase because map-reduce workloads are cost associative (Armbrust et al., 2010). In essence, the reduced execution time offsets the per-unit cost of the extra instances.

The pricing model impacts the workload's completion time and monetary cost because cloud providers incentivize users to execute their workloads during low demand periods (Chohan et al., 2010). While the cloud provider guarantees access to on-demand instances, on-demand instances are 29% more expensive than spot instances (Chohan et al., 2010). Although cloud users can allocate the less expensive spot instances, the workload might take longer to complete because the user may need to wait for the instances to be allocated by a spot market auction. Once an auction allocates the instances, subsequent auctions may terminate the instances pre-maturely because the cloud provider's demand has peaked (Orna Agmon Ben-Yehuda et al., 2013). While the user does not pay for the terminated instances, the early termination increases the workload's costs because the workload requires additional time to reprocess missing intermediate data on the more expensive on-demand instances (Chohan et al., 2010).

The storage strategy impacts the workload's execution time and monetary cost because the data's location impacts the workload's reads and writes. When a map-reduce workload uses spot instances, the workload cannot store persistent data on the spot instances because the spot instances are ephemeral. Users must either store data on a set of on-demand instances (Chohan et al., 2010) or an external file system, such as Amazon S3 (Bicer et al., 2011).

To help cloud users decide on virtual resources, Herodotou et al. (2011) proposed a workload cost estimation tool, the Elastisizer, that evaluates alternative clusters with different virtual machine types and sizes. The Elastisizer profiles the workload on a baseline cluster configuration. For the baseline profile, the Elastisizer uses a relative fitness model to create a virtual profile for different virtual machine types (Mesnier et al., 2007). Once the Elastisizer creates the virtual profile, the tool predicts the workload's execution time and cost from a white box model (Herodotou, 2012).

While the Elastisizer aids the cloud user to determine the type and number of virtual machines, the Elastisizer does not help the user determine a bid price or storage strategy. The cloud user can't determine a bid price because the Elastisizer inaccurately simulates the spot market's impact on execution time and cost. The spot market's auction reduces the current estimation tools' prediction accuracy because the tools don't forecast the spot instance's availability to process the workload's data (Chohan et al., 2010). The cloud user can't compare storage strategies because the tool inaccurately models the storage approach's impact on the workload's virtual profile. The storage strategy reduces the current estimation tools' prediction accuracy because the alternative storage strategies can create file system contention from non-local tasks (Zaharia et al., 2012).

This dissertation created a tool, MapReduce Workload Allocation Tool for Spot instances (MRWATS), which handles spot-instance specific storage strategies, spot market early termination, and auction wait times. The dissertation's goal was to improve the spot instance workload's mean completion time and monetary cost prediction accuracy compared to existing tools. The enhanced tool should reduce the average completion time prediction error for various cluster sizes and spot instance ratios.

MRWATS extends the Elastisizer (Herodotou et al., 2011) to incorporate spot instance storage strategies, spot instance early termination, and spot market forecasts. MRWATS improves the prediction accuracy for both the Amazon S3 and accelerated storage strategies because MRWATS creates a new relative fitness model for each new storage strategy. MRWATS also incorporated new features (size and accelerator ratio) into the relative fitness model to predict the file system contention's impact on the workload.

The dissertation evaluated the storage strategy updates by comparing MRWATS and the Elastisizer's storage specific prediction error. The evaluation tested a variety of I/O and CPU workloads against various storage configurations. When the workloads did not produce file system contention, MRWATS provided comparable prediction accuracy to the Elastisizer. When the workload produced heavy file system contention, MRWATS reduce the average prediction error 58% compared to the Elastisizer.

MRWATS improved the prediction accuracy over the Elastisizer for terminated spot instances because the MRWATS scheduler simulates failed tasks. If the spot market auction terminates the spot instances at a given time, the simulator determines the number of failed tasks and then reschedules them. The simulator also selects a new virtual profile once the failure occurs because the terminated spot instances no longer generate file contention.

This dissertation evaluated MRWATS's enhanced scheduler by comparing MRWATS and the Elastisizer's early termination prediction error. Although the evaluation tested the same workloads as the storage strategy, the tests examined prediction errors during failures at different map-reduce waves. The tests varied the termination point by wave because each wave creates a different number of failed tasks. When the workload terminates early, MRWATS reduced the average prediction error by 76% compared to the Elastisizer.

MRWATS improves the prediction accuracy for spot instance workloads compared to the Elastisizer because MRWATS estimates the workload's completion time and cost based on the spot instance's availability. MRWATS incorporates Chohan et al. (2010) trace-based approach to determine the spot instances availability. Based on the spot market traces, MRWATS estimates the expected wait time and early termination probabilities for a given bid price. MRWATS then calculates the workloads' average completion time and cost from the wait time and early termination probabilities.

This dissertation evaluated historical spot market traces to quantify MRWATS's completion time and cost accuracy improvements over the Elastisizer. To evaluate each tool against different market conditions, the evaluation analyzed two recent spot market price histories. Each price history experiment splits the spot market price history into forecast and actual traces. The forecast trace provided MRWATS a market price history to predict the workload's completion time and monetary costs. The evaluation then executed a Monte Carlo simulation over the actual trace to calculate the workload's average completion time and average monetary costs.

Based on the Monte Carlo simulation results, the evaluation compared MRWATS's and the Elastisizer's average spot market prediction error. While the results varied based on workload and bid price, MRWATS reduced the average prediction error for each trace evaluated. MRWATS reduced the average completion time prediction error up to 96% and the average cost prediction error up to 99%.

Appendix A

Additional Early Termination Results

The 30 node early termination results for additional accelerator ratios and virtual instance types.

Early Termination Accuracy by Wave for C1.Medium with 33% Accelerators

	Failed Maps		Failed Reduces		Completion Time (Seconds)			CT % Error	
Terminated Wave	Act.	Pred.	Act.	Pred.	Actual	HDFS	ET	HDFS	ET
Co-Occurrence									
Map 1	20	20	0	0	622.60	482.89	662.95	22.44	6.48
Map 2	40	40	20	20	628.31	482.89	677.02	23.14	7.75
Reduce 1	51	49	20	20	724.65	482.89	892.84	33.36	23.21
Reduce 2	49	49	20	8	747.64	482.89	891.61	35.41	19.26
Grep									
Map 1	20	20	0	0	162.05	48.53	88.13	70.05	45.62
Map 2	20	20	0	0	165.72	48.53	113.13	70.72	31.73
Pi Est.									
Map 1	20	20	0	0	289.74	192.08	296.09	33.71	2.19
Map 2	33	28	0	1	324.81	192.08	313.59	40.86	3.46
Reduce 1	35	30	0	1	372.10	192.08	375.51	48.38	0.92
TeraSort									
Map 1	30	20	17	0	924.46	515.44	843.55	44.24	8.75
Map 2	43	40	20	20	895.60	515.44	860.30	42.45	3.94
Reduce 1	78	80	20	20	962.50	515.44	776.47	46.45	19.33
Reduce 2	78	80	19	8	886.09	515.44	777.75	41.83	12.23
TF-IDF Job1									
Map 1	20	20	0	0	927.18	563.17	842.88	39.26	9.09
Map 2	43	40	22	20	971.32	563.17	844.06	42.02	13.10
Reduce 1	201	200	20	20	996.23	563.17	907.51	43.47	8.91
Reduce 2	198	200	28	8	1006.91	563.17	907.54	44.07	9.87
Word Count									
Map 1	20	20	0	0	1060.39	661.99	993.49	37.57	6.31
Map 2	40	40	20	20	1048.37	661.99	995.15	36.85	5.08
Reduce 1	198	200	20	20	1123.48	661.99	1069.85	41.08	4.77
Reduce 2	199	200	26	20	1134.36	661.99	1059.47	41.64	6.60

Early Termination Accuracy by Wave for C1.Medium with 50% Accelerators

	Failed Maps		Failed Reduces		Completion Time (Seconds)			CT % Error	
Terminated Wave	Act.	Pred.	Act.	Pred.	Actual	HDFS	ET	HDFS	ET
Co-Occurrence									
Map 1	30	30	0	0	711.50	482.89	978.10	32.13	37.47
Map 2	30	30	0	0	704.33	482.89	978.10	31.44	38.87
Reduce 1	79	81	30	30	856.18	482.89	1068.33	43.60	24.78
Reduce 2	84	79	31	19	852.91	482.89	1049.66	43.38	23.07
Grep									
Map 1	30	30	0	0	150.61	48.53	90.96	67.78	39.60
Map 2	36	30	0	0	141.71	48.53	108.13	65.76	23.70
Pi Est.									
Map 1	30	30	0	0	388.91	192.08	385.24	50.61	0.94
Map 2	53	49	1	1	420.40	192.08	402.73	54.31	4.20
Reduce 1	46	50	1	1	477.69	192.08	465.66	59.79	2.52
TeraSort									
Map 1	41	30	57	0	1138.00	515.44	1175.64	54.71	3.31
Map 2	72	60	30	30	1034.64	515.44	1195.17	50.18	15.52
Reduce 1	128	120	30	30	1165.38	515.44	1296.16	55.77	11.22
Reduce 2	128	120	30	19	1079.64	515.44	1113.60	52.26	3.15
TF-IDF Job1									
Map 1	30	30	0	0	1144.93	563.17	1121.45	50.81	2.05
Map 2	60	60	30	30	1168.82	563.17	1122.66	51.82	3.95
Reduce 1	306	300	30	30	1257.09	563.17	1153.90	55.20	8.21
Reduce 2	304	300	30	19	1248.07	563.17	1185.15	54.88	5.04
Word Count									
Map 1	30	30	0	0	1294.71	661.99	1323.27	48.87	2.21
Map 2	60	60	30	30	1343.23	661.99	1324.82	50.72	1.37
Reduce 1	307	300	30	30	1423.77	661.99	1337.72	53.50	6.04
Reduce 2	306	300	38	19	1417.19	661.99	1383.96	53.29	2.35

Early Termination Accuracy by Wave for M1.Large Instances with 33% Accelerators

	Failed Maps		Failed Reduces		Completion Time (Seconds)			CT % Error	
Terminated Wave	Act.	Pred.	Act.	Pred.	Actual	HDFS	ET	HDFS	ET
Co-Occurrence									
Map 1	30	30	0	0	713.11	422.24	624.77	40.79	12.39
Map 2	52	59	20	20	692.63	422.24	616.33	39.04	11.02
Reduce 1	52	59	20	20	784.73	422.24	669.76	46.19	14.65
Reduce 2	51	59	20	8	778.94	422.24	667.60	45.79	14.29
Grep									
Map 1	30	30	0	0	164.39	45.43	90.14	72.36	45.16
Map 2	2	30	0	0	173.50	45.43	120.14	73.81	30.75
Pi Est.									
Map 1	30	30	0	0	394.04	361.81	364.69	8.18	7.45
Map 2	30	37	0	1	411.66	361.81	451.51	12.11	9.68
Reduce 1	36	35	1	1	484.54	361.81	610.51	25.33	26.00
TeraSort									
Map 1	31	30	0	0	633.78	339.97	514.07	46.36	18.89
Map 2	79	60	20	20	594.09	339.97	506.55	42.77	14.74
Reduce 1	74	89	20	20	642.13	339.97	535.37	47.06	16.62
Reduce 2	77	89	7	8	570.11	339.97	515.48	40.37	9.58
TF-IDF Job1									
Map 1	30	30	0	0	1011.52	631.00	904.25	37.62	10.60
Map 2	60	60	20	20	1007.73	631.00	909.71	37.38	9.73
Reduce 1	198	206	20	20	1094.11	631.00	1079.13	42.33	1.37
Reduce 2	197	206	22	8	1103.20	631.00	1078.79	42.80	2.21
Word Count									
Map 1	30	30	0	0	1188.52	749.91	1072.29	36.90	9.78
Map 2	60	60	20	20	1180.64	749.91	1076.01	36.48	8.86
Reduce 1	199	206	20	20	1290.72	749.91	1259.58	41.90	2.41
Reduce 2	200	206	20	8	1293.89	749.91	1254.77	42.04	3.02

Early Termination Accuracy by Wave for M1.Large Instances with 67% Accelerators

	Failed Maps		Failed Reduces		Completion Time (Seconds)			CT % Error	
Terminated Wave	Act.	Pred.	Act.	Pred.	Actual	HDFS	ET	HDFS	ET
Co-Occurrence									
Map 1	45	45	0	0	854.54	422.24	835.37	50.59	2.24
Map 2	58	79	30	30	764.80	422.24	829.94	44.79	8.52
Reduce 1	80	80	30	30	1045.34	422.24	877.36	59.61	16.07
Reduce 2	83	80	43	19	1043.73	422.24	879.60	59.54	15.73
Grep									
Map 1	45	45	0	0	232.52	45.43	95.62	80.46	58.88
Map 2	44	30	0	0	290.36	45.43	160.14	84.35	44.85
Pi Est.									
Map 1	45	45	0	0	468.19	361.81	538.72	22.72	15.06
Map 2	52	52	1	1	543.83	361.81	625.52	33.47	15.02
Reduce 1	51	50	1	1	632.05	361.81	784.52	42.76	24.12
TeraSort									
Map 1	55	45	2	0	1055.17	339.97	753.77	67.78	28.56
Map 2	95	120	30	30	868.68	339.97	799.99	60.86	7.91
Reduce 1	123	120	32	30	942.69	339.97	703.26	63.94	25.40
Reduce 2	116	120	22	30	732.98	339.97	767.70	53.62	4.74
TF-IDF Job1									
Map 1	45	45	0	0	1336.77	631.00	1258.96	52.80	5.82
Map 2	89	90	30	30	1388.77	631.00	1264.34	54.56	8.96
Reduce 1	297	302	30	30	1433.23	631.00	1346.97	55.97	6.02
Reduce 2	298	302	30	19	1427.96	631.00	1342.94	55.81	5.95
Word Count									
Map 1	45	45	0	0	1596.84	749.91	1494.86	53.04	6.39
Map 2	90	90	30	30	1638.95	749.91	1498.36	54.24	8.58
Reduce 1	301	302	30	30	1701.27	749.91	1572.03	55.92	7.60
Reduce 2	301	302	30	19	1693.93	749.91	1567.94	55.73	7.44

Appendix B

Additional MRWATS Results

Completion Time Error for C1.Medium Instances with 33% Accelerators by Bid Price

	Average Completion Time (hrs.)			Ave CT % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0180	16.83	1.05	16.58	93.79	1.44
0.1300	1.91	1.05	1.93	45.32	0.87
0.1705	1.40	1.05	1.46	25.51	4.07
2.5000	1.14	1.05	1.14	8.48	0.04
Grep					
0.0180	15.57	0.09	15.27	99.39	1.91
2.5000	0.14	0.09	0.14	32.52	0.00
Pi Est.					
0.0180	15.96	0.38	15.63	97.64	2.07
0.0500	1.20	0.38	1.21	68.59	0.83
0.1300	1.13	0.38	1.14	66.78	0.47
2.5000	0.44	0.38	0.44	14.19	0.04
TF-IDF					
0.0180	17.21	1.37	17.02	92.03	1.12
0.1300	2.23	1.37	2.26	38.51	1.28
0.2000	1.68	1.37	1.79	18.41	6.35
2.5000	1.41	1.37	1.41	2.63	0.19
TPCH Q12					
0.0180	15.94	0.37	15.60	97.68	2.13
0.0500	1.18	0.37	1.18	68.54	0.51
0.1300	1.11	0.37	1.11	66.70	0.20
2.5000	0.42	0.37	0.42	11.18	0.04
TPCH Q14					
0.0180	15.78	0.26	15.45	98.38	2.10
0.0490	1.04	0.26	1.04	75.40	0.53
0.1300	0.97	0.26	0.98	73.79	0.51
2.5000	0.29	0.26	0.29	11.22	0.04
TeraSort					
0.0180	17.26	1.28	17.07	92.59	1.11
0.1300	2.28	1.28	2.32	44.00	1.50
0.1705	1.78	1.28	1.86	27.94	4.94
2.5000	1.47	1.28	1.47	13.22	0.18
Word Count					
0.0180	17.54	1.62	17.41	90.77	0.74
0.1300	2.52	1.62	2.58	35.81	2.17
0.3425	1.96	1.62	2.09	17.34	6.87
2.5000	1.66	1.62	1.66	2.77	0.19

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

Monetary Cost Error for C1.Medium Instances with 33% Accelerators by Bid Price

	Average Cost ($)			Ave Cost % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0180	4.14	3.22	4.29	22.24	3.50
0.1300	3.80	4.39	3.80	15.54	0.09
0.1705	3.79	4.81	3.92	27.20	3.58
2.5000	4.58	29.17	4.56	537.60	0.35
Grep					
0.0180	0.43	0.29	0.43	31.59	0.98
2.5000	0.56	2.64	0.56	369.04	0.54
Pi Est.					
0.0180	1.56	1.16	1.49	25.92	4.97
0.0500	1.42	1.28	1.42	10.26	0.64
0.1300	1.43	1.58	1.40	10.13	2.38
2.5000	1.77	10.49	1.75	494.25	0.95
TF-IDF Job1					
0.0180	5.28	4.22	5.54	19.94	5.09
0.1300	4.76	5.76	4.80	21.02	0.77
0.2000	4.72	6.72	4.98	42.25	5.36
2.5000	5.64	38.26	5.61	578.09	0.55
TPCH Q12					
0.0180	1.52	1.14	1.41	24.69	6.92
0.0500	1.37	1.26	1.34	7.73	1.49
0.1300	1.37	1.56	1.33	13.23	3.16
2.5000	1.68	10.33	1.66	515.16	0.94
TPCH Q14					
0.0180	1.04	0.79	0.95	24.19	8.08
0.0490	0.94	0.87	0.92	7.83	2.23
0.1300	0.95	1.07	0.91	13.20	3.71
2.5000	1.15	7.12	1.15	516.91	0.62
TeraSort					
0.0180	5.41	3.94	5.68	27.23	4.98
0.1300	4.93	5.37	4.98	9.09	1.05
0.1705	4.91	5.89	5.14	19.95	4.58
2.5000	5.91	35.69	5.87	504.36	0.54
Word Count					
0.0180	6.24	4.98	6.67	20.12	6.96
0.1300	5.64	6.80	5.75	20.51	1.91
0.3425	5.60	10.24	5.96	82.70	6.43
2.5000	6.66	45.15	6.63	577.57	0.48

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

Completion Time Error for C1.Medium Instances with 67% Accelerators by Bid Price

	Average Completion Time (hrs.)			Ave CT % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0180	17.73	1.05	17.63	94.10	0.57
0.1300	2.30	1.05	2.30	54.48	0.01
0.1705	1.71	1.05	1.90	39.02	11.07
2.5000	1.18	1.05	1.18	11.24	0.17
Grep					
0.0180	15.73	0.09	15.43	99.40	1.86
2.5000	0.30	0.09	0.30	68.86	0.00
Pi Est.					
0.0180	16.26	0.38	15.79	97.69	2.88
0.0500	1.29	0.38	1.27	70.85	1.25
0.1300	1.22	0.38	1.17	69.22	3.93
2.5000	0.43	0.38	0.43	12.64	0.17
TF-IDF					
0.0180	18.39	1.37	18.50	92.54	0.60
0.1300	2.72	1.37	2.78	49.64	2.05
0.2000	2.03	1.37	2.42	32.37	19.37
2.5000	1.42	1.37	1.41	3.31	0.77
TPCH Q12					
0.0180	16.35	0.37	15.98	97.73	2.24
0.0500	1.43	0.37	1.45	74.19	1.26
0.1300	1.37	0.37	1.35	72.88	1.48
2.5000	0.59	0.37	0.59	37.07	0.11
TPCH Q14					
0.0180	16.07	0.26	15.68	98.41	2.44
0.0500	1.22	0.26	1.22	79.08	0.07
0.1300	1.15	0.26	1.14	77.85	1.44
2.5000	0.42	0.26	0.42	38.80	0.11
TeraSort					
0.0180	18.55	1.28	18.76	93.10	1.17
0.1300	3.04	1.28	3.17	57.91	4.45
0.1650	2.47	1.28	2.80	48.23	13.17
2.5000	1.80	1.28	1.79	29.05	0.55
Word Count					
0.0180	18.93	1.62	19.29	91.45	1.91
0.1300	3.11	1.62	3.28	48.04	5.21
0.3425	2.38	1.62	2.89	32.12	21.22
2.5000	1.67	1.62	1.66	3.05	0.78

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

Monetary Cost Error for C1.Medium Instances with 67% Accelerators by Bid Price

	Average Cost ($)			Ave Cost % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0180	3.52	1.89	3.77	46.30	7.03
0.1300	2.75	4.23	2.74	54.02	0.42
0.1705	2.71	5.08	2.98	87.72	10.12
2.5000	4.31	53.79	4.27	1149.41	0.75
Grep					
0.0180 * **	0.48	0.17	0.52	64.30	8.68
2.5000	1.12	4.87	1.11	334.40	1.45
Pi Est.					
0.0180	1.27	0.68	1.06	46.37	16.62
0.0500	0.95	0.92	0.91	3.08	4.02
0.1300	0.97	1.52	0.88	56.91	9.38
2.5000	1.59	19.34	1.56	1113.47	2.05
TF-IDF Job1					
0.0180	4.51	2.48	5.03	44.91	11.64
0.0500 * **	3.41	3.36	3.94	1.58	15.43
0.0600	3.54	3.63	3.84	2.55	8.24
TPCH Q12					
0.0180	1.43	0.67	1.36	53.00	4.85
0.0500	1.21	0.91	1.21	24.81	0.56
0.1300	1.23	1.50	1.18	21.94	3.82
2.5000	2.18	19.06	2.14	775.82	1.80
TPCH Q14					
0.0180	1.00	0.46	0.90	53.80	10.00
0.0500	0.84	0.63	0.83	25.97	1.87
0.1300	0.87	1.03	0.82	19.45	5.28
2.5000	1.55	13.13	1.51	749.92	2.01
TeraSort					
0.0180	4.80	2.32	5.42	51.76	13.00
0.1300	4.00	5.18	4.17	29.58	4.17
0.1650	3.97	6.08	4.43	53.10	11.69
2.5000	6.59	65.81	6.52	898.74	1.11
Word Count					
0.0180	5.33	2.93	6.18	45.04	16.02
0.1300	4.07	6.55	4.28	60.99	5.15
0.3425	3.99	13.43	4.71	236.79	18.18
2.5000	6.08	83.26	6.02	1268.77	1.05
Co-Occurrence					

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

Completion Time Error for M1.Large Instances with 33% Accelerators by Bid Price

	Average Completion Time (hrs.)			Ave CT % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0260	1.74	0.92	1.77	47.43	1.66
0.0540	1.71	0.92	1.70	46.29	0.51
0.2000	1.65	0.92	1.64	44.56	0.95
3.3000	0.90	0.92	0.90	1.75	0.05
Grep					
0.0260	0.87	0.08	0.91	91.06	5.38
0.0290	0.87	0.08	0.91	91.06	5.37
3.3000	0.15	0.08	0.15	48.77	0.00
Pi Est.					
0.0260	1.27	0.49	1.29	61.51	2.02
0.0300	1.26	0.49	1.28	61.44	1.58
0.2000	1.17	0.49	1.17	58.35	0.14
3.3000	0.50	0.49	0.50	1.68	0.05
TF-IDF Job1					
0.0260	2.55	1.47	2.59	42.12	1.88
0.2000	2.42	1.47	2.43	39.18	0.46
0.2410	2.36	1.47	2.37	37.51	0.47
3.3000	1.58	1.47	1.58	6.65	0.18
TPCH Q12					
0.0260	1.27	0.53	1.29	58.15	1.56
0.0800	1.19	0.53	1.19	55.36	0.18
0.2000	1.17	0.53	1.17	54.67	0.33
3.3000	0.49	0.53	0.49	8.83	0.06
TPCH Q14					
0.0260	1.00	0.30	1.03	69.88	3.32
0.0800	0.91	0.30	0.93	67.03	2.48
3.3000	0.26	0.30	0.26	17.49	0.07
TeraSort					
0.0260	1.80	0.85	1.81	52.93	0.78
0.0600	1.71	0.85	1.71	50.31	0.15
0.2000	1.69	0.85	1.69	49.79	0.08
3.3000	0.97	0.85	0.97	12.55	0.04
Word Count					
0.0260	2.88	1.76	2.95	38.77	2.40
0.2000	2.76	1.76	2.77	36.09	0.49
0.2400	2.69	1.76	2.71	34.54	0.50
3.3000	1.84	1.76	1.83	3.92	0.29

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

Monetary Cost Error for M1.Large Instances with 33% Accelerators by Bid Price

	Average Cost ($)			Ave Cost % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0260	5.10	4.64	5.02	9.14	1.62
0.0540	5.09	4.90	4.96	3.81	2.48
0.2000	5.16	6.23	4.94	20.69	4.36
3.3000	9.24	34.65	9.25	274.91	0.10
Grep					
0.0260	0.77	0.39	0.76	48.71	0.41
0.0290	0.77	0.40	0.76	48.40	0.47
3.3000	1.52	2.93	1.56	93.72	2.78
Pi Est.					
0.0260	2.75	2.47	2.65	10.16	3.46
0.0300	2.74	2.49	2.65	9.16	3.24
0.2000	2.78	3.32	2.63	19.25	5.48
3.3000	5.05	18.44	5.09	265.36	0.96
TF-IDF Job1					
0.0260	9.04	7.46	9.05	17.54	0.09
0.2000	8.97	10.02	8.87	11.74	1.07
0.2410	8.94	10.62	8.90	18.87	0.40
3.3000	16.25	55.69	16.19	242.83	0.36
TPCH Q12					
0.0260	2.77	2.69	2.64	2.69	4.45
0.0800	2.76	2.98	2.62	8.17	4.99
0.2000	2.79	3.62	2.62	29.52	6.41
3.3000	4.98	20.12	5.02	304.42	0.95
TPCH Q14					
0.0260	1.41	1.52	1.35	7.77	4.55
0.0800	1.37	1.68	1.34	22.54	2.49
3.3000	2.57	11.37	2.63	342.72	2.36
TeraSort					
0.0260	5.38	4.29	5.23	20.32	2.79
0.0600	5.30	4.58	5.19	13.69	2.11
0.2000	5.34	5.76	5.18	7.88	2.93
3.3000	9.94	32.03	9.95	222.10	0.07
Word Count					
0.0260	10.68	8.92	10.78	16.45	0.96
0.2000	10.62	11.99	10.53	12.92	0.81
0.2400	10.58	12.69	10.56	19.97	0.21
3.3000	18.89	66.65	18.80	252.85	0.46

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

Completion Time Error for M1.Large Instances with 67% Accelerators by Bid Price

	Average Completion Time (hrs.)			Ave CT % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0260	2.18	0.92	2.13	58.03	2.58
0.0500	2.09	0.92	2.01	56.08	3.74
0.2000	2.08	0.92	1.93	55.96	7.40
3.3000	0.93	0.92	0.93	1.32	0.20
Grep					
0.0260	0.94	0.08	0.99	91.78	5.01
3.3000	0.23	0.08	0.23	65.82	0.00
Pi Est.					
0.0260	1.43	0.49	1.39	65.85	2.71
0.0300	1.42	0.49	1.38	65.65	2.83
0.2000	1.35	0.49	1.25	63.92	7.58
3.3000	0.50	0.49	0.50	1.85	0.20
TF-IDF Job1					
0.0260	3.38	1.47	3.41	56.41	0.93
0.2000	3.19	1.47	3.10	53.88	2.93
0.2410	3.10	1.47	3.03	52.44	2.33
3.3000	1.58	1.47	1.57	7.01	0.74
TPCH Q12					
0.0260	1.58	0.53	1.55	66.22	1.84
0.0500	1.54	0.53	1.46	65.40	4.88
0.2000	1.50	0.53	1.40	64.58	6.92
3.3000	0.62	0.53	0.62	13.69	0.17
TPCH Q14					
0.0260	1.17	0.30	1.16	74.23	0.86
0.0500	1.13	0.30	1.09	73.41	3.66
0.0800	1.08	0.30	1.05	72.10	2.31
3.3000	0.34	0.30	0.34	12.28	0.16
TeraSort					
0.0260	2.31	0.85	2.23	63.28	3.43
0.0540	2.23	0.85	2.12	61.92	4.92
0.2000	2.17	0.85	2.04	60.95	6.13
3.3000	1.07	0.85	1.07	20.78	0.15
Word Count					
0.0260	3.92	1.76	4.04	55.03	3.14
0.2000	3.75	1.76	3.67	52.95	2.01
0.2400	3.65	1.76	3.59	51.69	1.52
3.3000	1.84	1.76	1.82	4.38	1.14

Monetary Cost Error for M1.Large Instances with 67% Accelerators by Bid Price

	Average Cost ($)			Ave Cost % Error	
Bid	Simulated	HDFS Pred.	Forecast	HDFS Pred.	Forecast
Co-Occurrence					
0.0260	3.86	2.68	3.64	30.73	5.77
0.0500	3.73	3.12	3.52	16.45	5.55
0.2000	3.88	5.87	3.48	51.40	10.24
3.3000	12.36	62.70	12.38	407.11	0.10
Grep					
0.0260	0.65	0.23	0.66	64.96	1.53
3.3000	2.91	5.31	3.03	82.22	4.03
Pi Est.					
0.0260	1.91	1.42	1.73	25.45	9.67
0.0300	1.89	1.46	1.72	22.73	9.10
0.2000	1.98	3.12	1.68	58.00	14.91
3.3000	6.52	33.36	6.62	411.38	1.48
TF-IDF Job1					
0.0260	6.91	4.30	6.89	37.72	0.33
0.2000	6.75	9.43	6.50	39.63	3.74
0.2410	6.69	10.64	6.56	59.13	1.87
3.3000	21.11	100.78	21.00	377.36	0.55
TPCH Q12					
0.0260	2.31	1.55	2.15	32.64	6.96
0.0500	2.30	1.81	2.10	21.35	8.58
0.2000	2.38	3.41	2.09	43.08	12.16
3.3000	8.15	36.40	8.22	346.90	0.89
TPCH Q14					
0.0260	1.23	0.88	1.11	28.31	9.82
0.0500	1.22	1.02	1.09	16.25	10.51
0.0800	1.21	1.20	1.09	0.22	9.45
3.3000	4.46	20.57	4.57	361.57	2.53
TeraSort					
0.0260	4.19	2.47	3.92	40.91	6.37
0.0540	4.09	2.95	3.82	27.93	6.59
0.2000	4.13	5.42	3.79	31.26	8.32
3.3000	14.24	57.95	14.26	307.01	0.12
Word Count					
0.0260	8.28	5.15	8.46	37.81	2.20
0.2000	8.15	11.28	7.94	38.43	2.56
0.2400	8.08	12.69	8.00	57.12	1.02
3.3000	24.51	120.60	24.33	392.13	0.70

Note * = Forecasted optimal price bid, ** = Simulated optimal price bid

References

Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., & Tsafrir, D. (2011). Deconstructing Amazon EC2 Spot Instance Pricing. *IEEE International Conference on Cloud Computing Technology and Science* (pp. 304-311). IEEE. doi: 10.1109/CloudCom.2011.48

Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., & Tsafrir, D. (2013). Deconstructing Amazon EC2 Spot Instance Pricing. *ACM Trans. Econ. Comput., 1*(3), 1-20. doi: 10.1145/2509413.2509416

Agmon Ben-Yehuda, O., Ben-Yehuda, M., Schuster, A., & Tsafrir, D. (2014). The rise of RaaS: the resource-as-a-service cloud. *Commun. ACM, 57*(7), 76-84. doi: 10.1145/2627422

Amazon Web Services Inc. (2014). Amazon EC2 Instances. 11/25/2014, from http://aws.amazon.com/ec2/instance-types/

Amazon Web Services LLC. (2011). Amazon Elastic Compute Cloud. Retrieved 10/1/2011, from http://aws.amazon.com/ec2/

Apache Software Foundation. (2012). Hadoop Distributed File System. Retrieved 4/19/2012, from http://hadoop.apache.org/hdfs/

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., . . . Zaharia, M. (2010). A view of cloud computing. *Commun. ACM, 53*(4), 50-58. doi: http://doi.acm.org/10.1145/1721654.1721672

Barr, J. (2014). AWS Price Reduction. Retrieved 9/26/2014, from http://aws.amazon.com/blogs/aws/aws-price-reduction-42-ec2-s3-rds-elasticache-and-elastic-mapreduce/

Bicer, T., Chiu, D., & Agrawal, G. (2011). MATE-EC2: a middleware for processing data with AWS. *Proceedings of the 2011 ACM international workshop on Many task computing on grids and supercomputers* (pp. 59-68). ACM. doi: 10.1145/2132876.2132889

Bresnahan, J., Keahey, K., LaBissoniere, D., & Freeman, T. (2011). Cumulus: an open source storage cloud for science. *Proceedings of the 2nd international workshop on Scientific cloud computing* (pp. 25-32). 1996115: ACM. doi: 10.1145/1996109.1996115

Chen, J., Wang, C., Zhou, B., Sun, L., Lee, Y., & Zomaya, A. (2011). Tradeoffs Between Profit and Customer Satisfaction for Service Provisioning in the Cloud. *Proceedings of the 20th international symposium on High performance distributed computing* (pp. 229-238). ACM. doi: 10.1145/1996130.1996161

Chohan, N., Castillo, C., Spreitzer, M., Steinder, M., Tantawi, A., & Krintz, C. (2010). See spot run: using spot instances for mapreduce workflows. *Proceedings of the 2nd USENIX conference on Hot topics in cloud computing* (pp. 7-7). USENIX Association. Retrieved from http://www.usenix.org/events/hotcloud10/tech/full_papers/Chohan.pdf

Chun, B. N., & Culler, D. E. (2002). User-Centric Performance Analysis of Market-Based Cluster Batch Schedulers. *IEEE International Symposium on Cluster Computing and the Grid* (pp. 30-30). Retrieved from http://doi.ieeecomputersociety.org/10.1109/CCGRID.2002.1017109

Dean, J., & Ghemawat, S. (2008). MapReduce: simplified data processing on large clusters. *Commun. ACM, 51*(1), 107-113. doi: 10.1145/1327452.1327492

Ghemawat, S., Gobioff, H., & Leung, S.-T. (2003). The Google file system. *Proceedings of the nineteenth ACM Symposium on Operating Systems Principles* (pp. 29-43). New York, NY: ACM. doi: 10.1145/945445.945450

Herodotou, H. (2011). *Hadoop Preformance Models* (Report No. CS-2011-05). Retrieved from the Duke University website: http://www.cs.duke.edu/starfish/files/hadoop-models.pdf

Herodotou, H. (2012). *Automatic tuning of data-intensive analytical workloads.* (Doctoral dissertation). Available from ProQuest Dissertations & Theses database. (UMI No. 3504075)

Herodotou, H., & Babu, S. (2011). Profiling, what-if analysis, and cost-based optimization of MapReduce programs. *Proceedings of the VLDB Endowment, 4*(11), 1111-1122.

Herodotou, H., Dong, F., & Babu, S. (2011). No one (cluster) size fits All:Automatic cluster sizing for data-intensive analytics. *Proceedings of the 2nd ACM Symposium on Cloud Computing* (pp. 1-14). ACM. doi: 10.1145/2038916.2038934

Kavulya, S., Tan, J., Gandhi, R., & Narasimhan, P. (2010). An analysis of traces from a production MapReduce cluster. *Proceedings of the 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing* (pp. 94-103). IEEE Computer Society. doi: 10.1109/CCGRID.2010.112

Lee, G., Chun, B.-G., & Katz, R. H. (2011). Heterogeneity-aware resource allocation and scheduling in the cloud. *3rd USENIX Workshop on Hot Topics in Cloud Computing* (p. 1). USENIX Association. Retrieved from http://www.usenix.org/events/hotcloud11/tech/final_files/Lee.pdf

Li, M., Subhraveti, D., Butt, A. R., Khasymski, A., & Sarkar, P. (2012). CAM: a topology aware minimum cost flow based resource manager for MapReduce applications in the cloud. *Proceedings of the 21st international symposium on High-Performance Parallel and Distributed Computing* (pp. 211-222). ACM. doi: 10.1145/2287076.2287110

Liu, H. (2011). Cutting MapReduce Cost with Spot Market. *Proceedings of the 3rd USENIX Workshop on Hot Topics in Cloud Computing* (pp. 1-5). USENIX Association. Retrieved from http://www.usenix.org/event/hotcloud11/tech/final_files/Liu_Huan.pdf

Meng, X., Isci, C., Kephart, J., Zhang, L., Bouillet, E., & Pendarakis, D. (2010). Efficient resource provisioning in compute clouds via VM multiplexing. *Proceeding of the 7th international conference on Autonomic computing* (pp. 11-20). ACM. doi: 10.1145/1809049.1809052

Mesnier, M., Wachs, M., Sambasivan, R., Zheng, A., & Ganger, G. (2007). Modeling the relative fitness of storage. *Proceedings of the 2007 ACM SIGMETRICS international conference on Measurement and modeling of computer systems* (pp. 37-48). ACM. doi: 10.1145/1254882.1254887

Palankar, M. R., Iamnitchi, A., Ripeanu, M., & Garfinkel, S. (2008). Amazon S3 for science grids: a viable solution? *Proceedings of the 2008 international workshop on Data-aware distributed computing* (pp. 55-64). ACM. doi: 10.1145/1383519.1383526

Schad, J., Dittrich, J., & Quiané-Ruiz, J.-A. (2010). Runtime measurements in the cloud: observing, analyzing, and reducing variance. *Proc. VLDB Endow., 3*, 460-471.

Schwarzkopf, M., Murry, D. G., & Hand, S. (2012). The Seven Deadly Sins of Cloud Computing Research. *4th USENIX Workshop on Hot Topics in Cloud Computing* (p. 1). USENIX Association. Retrieved from https://http://www.usenix.org/conference/hotcloud12/seven-deadly-sins-cloud-computing-research

Seltzer, M., Krinsky, D., Smith, K., & Zhang, X. (1999). The Case for Application-Specific Benchmarking. *Proceedings of the The Seventh Workshop on Hot Topics in Operating Systems* (pp. 102-107). IEEE Computer Society. doi: 10.1109/HOTOS.1999.798385

Sharma, V., & Srinivasan, D. (2007). Evolutionary computation and economic time series forecasting. *Evolutionary Computation, 2007 IEEE Congress on* (pp. 188-195). doi: 10.1109/CEC.2007.4424471

Stokely, M., Winget, J., Keyes, E., Grimes, C., & Yolken, B. (2009). Using a market economy to provision compute resources across planet-wide clusters. *Proceedings for the IEEE International Parallel and Distributed Processing Symposium* (pp. 1-8). IEEE. doi: 10.1109/IPDPS.2009.5160966

Verma, A., Cherkasova, L., & Campbell, R. (2011). ARIA: automatic resource inference and allocation for mapreduce environments. *Proceedings of the 8th ACM international conference on Autonomic computing* (pp. 235-244). ACM. doi: 10.1145/1998582.1998637

Wang, G., Butt, A. R., Monti, H., & Gupta, K. (2011). Towards Synthesizing Realistic Workload Traces for Studying the Hadoop Ecosystem. *International Symposium on Modeling, Analysis, and Simulation of Computer Systems* (pp. 400-408). doi: 10.1109/MASCOTS.2011.59

Wang, G., Butt, A. R., Pandey, P., & Gupta, K. (2009, 21-23 Sept. 2009). A simulation approach to evaluating design decisions in MapReduce setups. *IEEE International Symposium on Modeling, Analysis & Simulation of Computer and Telecommunication Systems* (pp. 1-11). doi: 10.1109/MASCOT.2009.5366973

Wieder, A., Bhatotia, P., Post, A., & Rodrigues, R. (2010). Conductor: orchestrating the clouds. *Proceedings of the 4th International Workshop on Large Scale*

Distributed Systems and Middleware (pp. 44-48). ACM. doi: 10.1145/1859184.1859197

Wieder, A., Bhatotia, P., Post, A., & Rodrigues, R. (2012). Orchestrating the deployment of computations in the cloud with conductor. *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (pp. 27-27). Berkeley, CA: USENIX Association. Retrieved from https://http://www.usenix.org/conference/nsdi12/orchestrating-deployment-computations-cloud-conductor

Zaharia, M., Borthakur, D., Sarma, J., Elmeleegy, K., Shenker, S., & Stoica, I. (2010). Delay scheduling: a simple technique for achieving locality and fairness in cluster scheduling. *Proceedings of the 5th European conference on Computer systems* (pp. 265-278). ACM. doi: 10.1145/1755913.1755940

Zaharia, M., Chowdhury, M., Das, T., Dave, A., Ma, J., McCauley, M., . . . Stoica, I. (2012). Resilient distributed datasets: a fault-tolerant abstraction for in-memory cluster computing. *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation* (pp. 2-2). USENIX Association.

Zaharia, M., Konwinski, A., Joseph, A. D., Katz, R., & Stoica, I. (2008). Improving MapReduce performance in heterogeneous environments. *Proceedings of the 8th USENIX conference on Operating systems design and implementation* (pp. 29-42). Berkeley, CA: USENIX Association.