3-21-2013

Cloud Computing Trace Characterization and Synthetic Workload Generation

Salvatore Capra

CLOUD COMPUTING TRACE CHARACTERIZATION AND SYNTHETIC WORKLOAD GENERATION

THESIS

Salvatore Capra, Civilian, USAF

AFIT-ENG-13-M-11

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENG-13-M-11

CLOUD COMPUTING TRACE CHARACTERIZATION AND SYNTHETIC WORKLOAD GENERATION

THESIS

Presented to the Faculty

Department of Electrical and Computer Engineering

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Cyber Operations

Salvatore Capra, BS

Civilian, USAF

March 2013

AFIT-ENG-13-M-11

CLOUD COMPUTING TRACE CHARACTERIZATION AND SYNTHETIC WORKLOAD GENERATION

Salvatore Capra, BS
Civilian, USAF

Approved:

_____	_____
Kenneth M. Hopkinson (Chairman)	Date
_____	_____
Jonathan W. Butts, Maj, USAF (Member)	Date
_____	_____
Kennard R. Laviers, Maj, USAF (Member)	Date

Abstract

This thesis researches cloud computing client initiated workloads. A heuristic presented in the work defines a process of workload trace characterization and synthetic workload generation. Analysis and characterization of a cloud trace provides insight into client request behaviors and statistical parameters. A versatile workload generation tool creates client connections, controls request rates, defines number of jobs, produces tasks within each job, and manages task durations. The test system consists of multiple clients creating workloads and a server receiving requests, all contained within a virtual machine environment. Statistical analysis verifies the synthetic workload experimental results are consistent with real workload behaviors and characteristics. This thesis provides researchers and developers with a lightweight process for characterizing and simulating cloud workloads.

Table of Contents

Page

Page

Page

List of Figures

Page

Page

List of Tables

Page

Page

List of Equations

Page

CLOUD COMPUTING TRACE CHARACTERIZATION AND SYNTHETIC WORKLOAD GENERATION

I. Introduction

General Issue

The future of cloud computing is moving toward a state in which we won't compute on local machines, but on highly automated data centers processing workloads in remote facilities. Commercial cloud services are becoming increasingly available, popular, complex, large, and difficult to administer and maintain. Current cloud computing research is vital to solving such demanding problem areas. Research topics such as autonomic systems, optimization, dynamic scalability, fault tolerance, virtual machine scheduling and releasing, hypervisor resource management, and clouds for rent/cost analysis, all rely on some form of workload input.

The accuracy of research results can vary considerably based on slight variations to the input. Trace files are client workloads, and serve as input to the cloud algorithm. Understanding and simulating realistic workload characteristics are imperative for making effective design decisions and adding value to research results. Generating realistic workloads, or trace files, can contribute to innovation in numerous areas of cloud computing.

Problem Statement / Objectives

The goal of this thesis is determining whether synthetically generated cloud workloads have characteristics statistically similar to real cloud traces. The simulated workloads, or synthetic traces, consist of characteristics of real trace files derived from

various forms of statistical analysis. This research is one part of an overall effort of improving autonomous management methods and resource provisioning in distributed systems.

Research Focus

This research develops a lightweight process for generating synthetic workloads using an open source load generator. It focuses on characterizing and simulating a publically available cloud workload, a trace file recently published in 2012 by Google. Synthetic workloads will ideally have statistically similar qualities compared to real traces.

Investigative Questions

What is new about this research? First, characterizing publically available workload traces is not new to the research community. Characterizing the Google cloud trace is new, and publications modeling the trace became available nearly the same time the trace became available. Google employees are involved with such early publications, giving researchers an early start. What is new about this research is using a new heuristic to simulate important characteristics of cloud traces using open source or free workload generation tools. The idea of generating realistic synthetic workloads is critical for researchers, especially those outside of private entities that do not own such trace files.

Why is this research relevant? This research is pertinent today because scientists and engineers developing and testing autonomic methodologies, optimization algorithms, and other cloud management issues, can utilize it. Researchers outside of private cloud

companies, such as academia, will have the information and tools necessary to feed their experiments with justifiably realistic workload inputs.

Literature Review

The next chapter discusses numerous aspects of cloud computing. The chapter defines cloud computing, compares clouds and grids, clarifies trace file availability, explores the Google trace, investigates a number of workload generation tools, and discusses related research.

Heuristic

Chapter 3 introduces a workload characterization and generation heuristic, which describes techniques that aid in the development of a synthetic workload. The heuristic is comprised of two main sections. It begins with workload analysis of particular trace file characteristics, such as job arrival rate and task duration. Next is a synthetic workload generator design that produces client-initiated workloads with characteristics similar to those of the analyzed trace file.

Methodology

Chapter 4 describes the methodological approach to this research effort. The methodology is as follows: obtaining appropriate traces, analyzing and characterizing the data, performing simulations utilizing the spiral development model, and comparing results for statistical similarities.

Analysis and Results

Chapter 5 begins with a statistical analysis of the Google trace, followed by a similar analysis of the synthetic workload. Next is a statistical comparison of similarities to the two traces. Chapter 4 ends with a scalability performance test of the workload generation tool.

Assumptions and Limitations

This research assumes publically available traces contain real data, and thus realistic characteristics. The data contained in the trace limits the results of this research. The small number of available traces also limits researchers. Even though these assumptions and limitations are restrictive, the results of this research are a vast improvement over predefined workloads, and provide researchers with the advantage necessary to justify their workloads.

II. Literature Review

Chapter Overview

This chapter begins by defining cloud computing. Second is a discussion on the limited availability of workload traces and data repositories. Next is comparing and contrasting cloud and grid computing technologies, followed by a description of the recently published Google cloud workload trace. The final section is a discussion on free and open source workload generation tools.

Cloud Computing

Cloud computing has numerous definitions within the scientific community. For the purpose of this research, the definition provided by the National Institute of Science and Technology (NIST) is appropriate: Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction [41].

Cloud Characteristics

Within the NIST definition, clouds display the following five essential characteristics [41]:

- On-demand Self-service. Consumers request and receive computing capabilities, such as server time and network storage, as needed automatically.
- Broad Network Access. A variety of devices such as smartphones, tablets, laptops, and workstations gain access to capabilities over the network.

- Resource Pooling. Integrated cloud computing resources serve multiple consumers, with different physical and virtual resources dynamically assigned and reassigned according to consumer demand. Examples of resources include storage, processing, memory, and network bandwidth.

- Rapid Elasticity. Capabilities are easily adapted to appropriate quantities proportionate with demand.

- Measured Service. Cloud systems automatically manage and optimize resources use by measuring, monitoring, and controlling services such as storage and processing.

Cloud Services

Cloud computing provides on-demand services and resources for consumers at three different levels:

- Software as a Service (SaaS). Applications and software running on cloud infrastructure support massive numbers of customers. Various interfaces, such as a web browser or other software interface, access applications remotely through the internet. [23,41].

- Platform as a Service (PaaS). Developers build, test, and deploy applications and software using an Application Programming Interface (API) environment. [23,28,41]

- Infrastructure as a Service (IaaS). The capability provided to the consumer is to provision hardware, software, and other computing resources [23,41]. Dynamically scalable raw infrastructure and associated middleware enable customers to run virtual machines [28,31]. For example, Amazon offers Elastic Compute Cloud (EC2) computing resources, which are available to the public for rent with a pay-per-use pricing model.

Cloud Traces

Recordings of application workload transactions, or traces, contain events such as request arrival time, job runtime, and other network-level traffic characteristics [3, 5, 35]. Obtaining real traces from cloud vendors is difficult, and publication of the traces is not

typical due to the proprietary nature of the data. Cloud computing vendors and/or the users who generate the traces consider them proprietary data. At the time of this research, there are only two known publically available cloud workload traces, both from Google.

Data Repositories

Limited Trace Availability

With very few publically available cloud traces offered, analysis of cloud traces is very limited. The Parallel Workloads Archive, The Grid Workloads Archive, and Failure Trace Archive all host numerous publically available real trace files. The traces may not have originated from a cloud platform, but there are enough similarities between clouds and grids to justify using grid workloads for cloud simulations. The next section, Cloud and Grid Computing Compared, compares the two computing platforms.

Publically Available Archives

The following list describes several well-known public workload trace archives. Researchers use the data within these repositories for countless studies and hundreds of publications.

- Parallel Workloads Archive. Numerous traces are publically available, the most recent being a workload of accounting records from the RIKEN Integrated Cluster of Clusters (RICC) installation in Japan [45]. RIKEN is a scientific research and technology institution of the Japanese government. The workload trace spans a period from May to Sep 2010, and represents 447,794 jobs. The Parallel Workloads Archive uses the Standard Workload Format for its trace file format.

- The Grid Workloads Archive. Numerous traces are publically available, the most recent being from the 2006 timeframe [27]. Traces over five years old are not used in this research for characterization purposes. The Grid Workloads Archive uses the Grid Workload Format for its trace file format.

- Failure Trace Archive. Numerous traces of parallel and distributed systems are publically available in this repository of system failure data [20]. The archive facilitates the design, validation, and comparison of fault-tolerant models and algorithms [20]. The SETI@home trace is the most recent available, and comes from a rather large distributed desktop grid system with approximately 230,000 nodes. The workload trace spans a period of 1.5 years from 2007-2009. The Failure Trace Archive uses Failure Trace Archive Format for its trace file format.

- MetaCentrum Data Sets. One trace from the Czech National Grid Infrastructure is publically available. This workload trace contains 103,656 jobs and spans a period from January to May 2009 [42]. MetaCentrum uses its own unique trace file format.

- Google Cloud Trace. Google recently published a limited production anonymized workload trace recorded in May 2011 that spans a period of approx 6 hours and 15 minutes with 5 min timestamps. It represents over 9000 jobs, each with multiple sub-tasks, totaling over 176,000 tasks [11]. This thesis research utilizes this trace for characterization and simulation experiments.

As computing technologies continue to change, the data contained in the traces that capture the workloads also change. Many of the traces in the above archives are over five years old, and consequently this research does not consider their use. Although there are few traces available, especially for academic research, the traces and archives described above do provide a diverse sampling of real workload traces. Varieties of statistical analyses characterize the traces. These derived characteristics are the foundation for building justifiably realistic synthetic traces.

Cloud and Grid Computing Compared

Cloud is the New Grid

Grid computing technologies primarily allow consumers to obtain processing power on demand. Cloud computing and grid computing are similar in the sense that both manage large datacenters and offer distributed computing resources to users [37].

It is no surprise that cloud computing and grid computing overlap in many aspects. Cloud computing evolved from grid computing and shares similar infrastructure. Building cloud environments on top of stable grid infrastructures is possible. In this scenario, grid services manage cloud virtual machines, as seen in the Nimbus project [23].

Figure 1 [23] displays the relationship between grids, clouds, and other computing environments. On the service oriented application side, Web 2.0 covers nearly the entire spectrum, and cloud computing lies at the large-scale side. Clusters and supercomputers are traditionally non-service application oriented. Finally, grid computing overlaps with all aforementioned computing environments. It covers both service and non-service applications, and is typically of lesser scale than supercomputers and clouds [23].

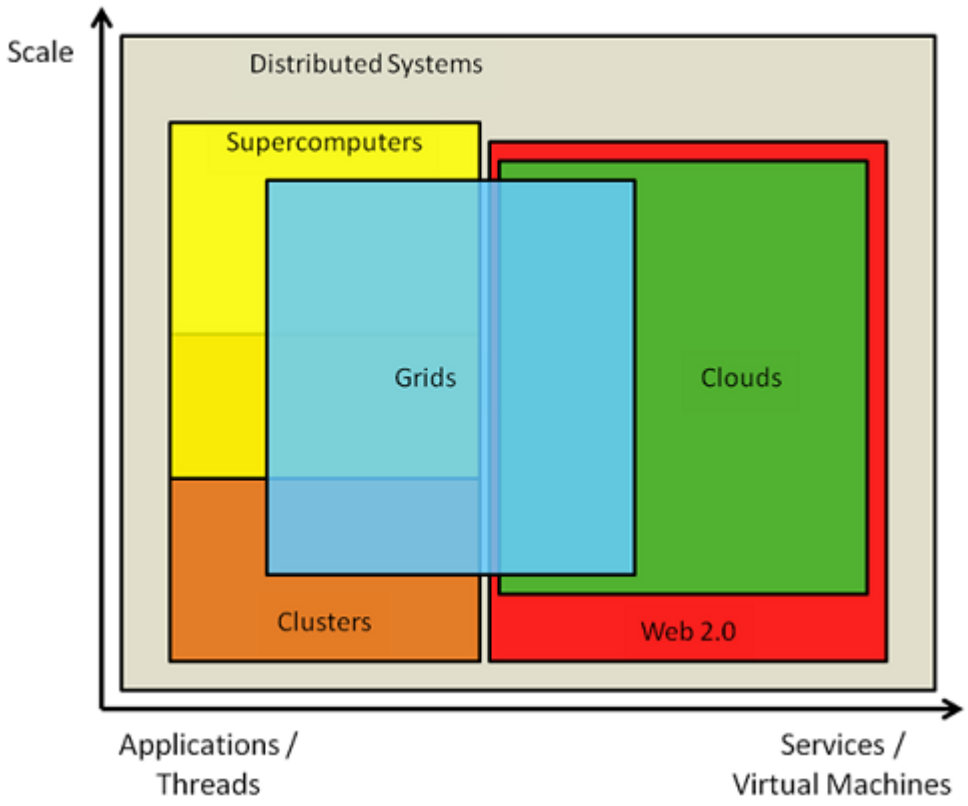

Figure 1: Computing Environment Relationships [23]

Cloud computing may be considered an extension or evolution of grid computing. Both share the same vision of reduced cost, increased reliability, and greater flexibility [13]. In addition, both share the concept of shifting computers from something we purchase and manage ourselves to something operated by third party utilities.

Cloud and Grid are Different Paradigms

Virtualization and Threads

Things are not what they used to be, especially when it comes to massive amounts of data and computing power. Large-scale commercial cloud systems contain thousands of computers and process millions of jobs in virtual machines. This virtualization is a

crucial component found in most clouds and allows for encapsulation and abstraction [13]. Virtualization is similar to the concept of threads used in grid systems, where multithreading allows for concurrent execution of the threads. The concept of dynamic scalability, or elasticity, is the ability to add and remove capacity and resources based on actual usage, made possible through virtualization. A disadvantage of virtualization is that it takes time to setup, and is major concern for efficient cloud utilization [7].

Figure 2 [6] shows an example of a grid system running a multithreaded music player application. In this simplified grid system, each user has a dedicated thread. Notice that any single failure could negatively affect all users in the system.

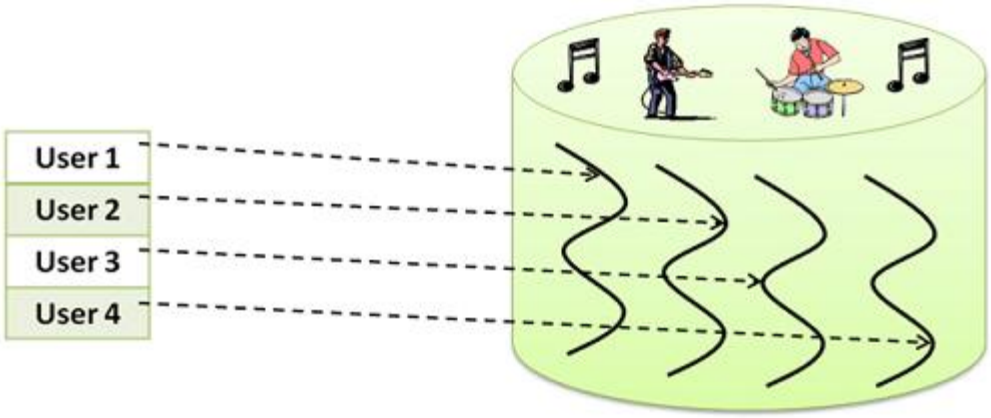

Figure 2: Multithreaded Music Player [6]

Figure 3 [6] shows an example of a cloud system running a single-threaded music player application. In this simplified cloud system, each user has a dedicated virtual machine. Notice that if the music player application fails, the impact is contained to one virtual machine and thus affects just one user.

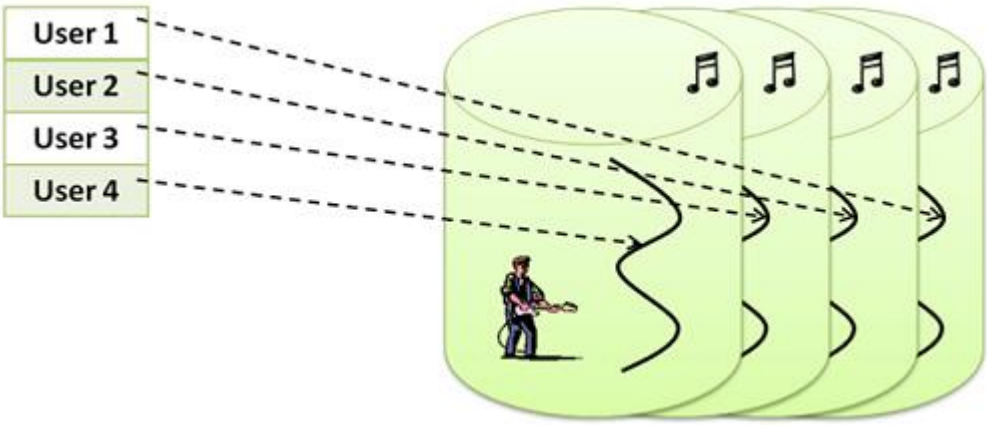

Figure 3: Virtual Machines with Single Threaded Music Player [6]

Clouds rely heavily on virtual machines, while grids typically do not. Clouds are massively scalable commercial systems consisting of hundreds of thousands of computers that consumers can access on-demand. Virtualization and sheer size are two of the biggest differences between clouds and grids. Even so, they share much commonality in vision, architecture, and technology.

Job Arrival

Job arrival rates in clouds are typically higher than grids and clusters. Table 1 shows the job arrival rate of some popular and well-studied public traces. Based on the data in Table 1, job submission frequency is much higher in clouds than that of grid or cluster systems.

Table 1: Jobs per Hour

	Auvergrid	RICC	ANL Intrepid	Google 1	Google 2
Type	Grid	Cluster	Cluster	Cloud	Cloud
Mean	48	122	11	1475	552

Table 1 compares job arrival rates of grids, clusters, and clouds. Perhaps a comparison of grid versus cluster computing is necessary. Clusters typically consist of several homogeneous computers (same hardware and OS) working together to solve a problem, and are controlled by a central resource manager [29]. On the other hand, grids consist of several heterogeneous networked computers (different OS and hardware), working together and utilizing spare computing power [29]. Clusters are typically housed together in a central location, while grids are distributed over a large area such as Local Area Network (LAN) or Wide Area Network (WAN) [29].

Job and Task Durations

The work of Di et al [17] compares the Google cloud load versus grids. The general observation is that Google jobs are much shorter than grid jobs. For example, over 80% Google jobs' durations are under 1000 seconds, while over half of grid jobs are over 2000 seconds. In addition, approximately 94% of Google task executions complete within 3 hours, while only 70% of grid task executions complete within 12 hours [17]. This difference in task duration is mainly because Google jobs, such as keyword search, are inherently short duration and often real-time, while grid jobs are usually based on longer duration complex scientific problems. [17]

Cloud vs. Grid Significance

In summary, clouds have much shorter job durations but higher arrival rate as compared to grids and clusters. These are important characteristics that researchers must consider when creating synthetic workloads to drive their experiments.

Why be concerned with grid workload when cloud workloads are publically available? First, timing is an issue. The Google cloud trace publication occurred after the

start of this research. In addition, there are relatively few cloud traces compared to grid traces. Many publications and much research exist on grid traces. Due to the limited availability of cloud traces, researchers will continue to rely on the relatively larger number of grid and cluster traces for their work.

Google Cluster Usage Trace

This section introduces the trace data from a Google datacenter. The data in the trace is highly anonymized for confidentiality reasons. The Google trace is from May 2011 and contains 6 hours 15 minutes of data capture, 3.5 million entries (observations), over 9000 jobs, and over 176,000 tasks. The file is available for download in comma-separated values (CSV) format.

Google Cluster

Google datacenters contain clusters, or sets of racked machines connected by a high-speed network [49]. User requests arrive in the form of jobs, with each job containing one or more tasks. Tasks that belong to the same parent job have similar resource usage requirements [35]. Jobs are assigned unique 64-bit identifiers, which are never reused [49].

Scheduling Jobs and Tasks

Google tasks have a life cycle of four different states: unsubmitted, pending, running, and dead, as shown in Figure 4. State transitions are events that either change the state of the task or affect the scheduling state [49]. High priority tasks are scheduled before low priority ones, and first-come-first-serve (FCFS) applies to tasks with equal priorities.

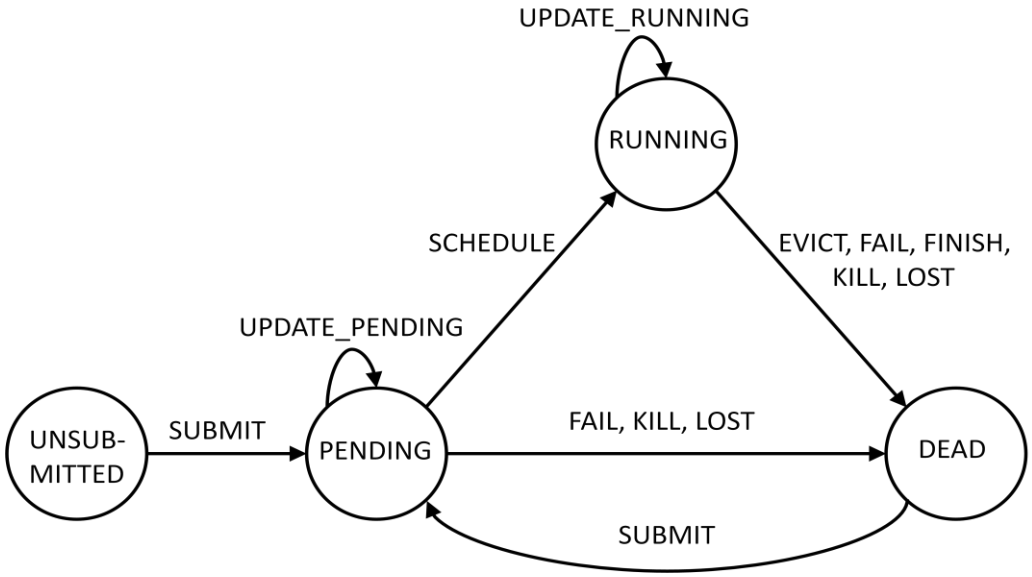

Figure 4: State Transitions [49]

It helps to understand the task lifecycle and state transition behavior prior to characterization and analysis of the trace data. Ultimately, researchers will understand the data to a level where it can be simulated using workload generation tools.

Workload Generation Tools

Numerous workload generation tools exist. Free or open source tools also exist. It is necessary to evaluate the available tools then choose one for simulations. At a minimum, the tool should have flexibility in request types, request rates, and tasks per job.

Rain Workload Generator

Rain is a statistics-based workload generation toolkit that uses distributions to model different workload classes [46]. It allows for delays between operation execution via cycle and think times. Rain assumes familiarity with workload generation and server

configuration/setup. The flexible and customizable workload characterization supports load variations. Rain has a Generator API for application specific load generators that target new systems and applications [46]. Rain Workload Toolkit is well suited for cloud workload generation. Unfortunately, documentation is minimal and lacks detail. Available tutorials are Olio and Raddit specific only. Consequently, Rain is not suitable for this research.

Olio Web 2.0

Olio is a Web 2.0 toolkit to aid in performance evaluations of web technologies. It is a Web 2.0 application that functions as a social event calendar. The toolkit also defines ways to drive load against the web application, which allows for performance measurements [44]. Olio is primarily for learning Web 2.0 technologies, evaluating the three implementations (PHP, Java EE, and RubyOnRails (ROR)), evaluating server technologies, and driving a load against the application to evaluate the performance and scalability of various platforms [44].

Olio has seven distinct operations that a workload can perform:

1. Homepage
2. Login
3. Tag Search
4. Event Detail
5. Person Detail
6. Add Person
7. Add Event

Olio is well documented, but assumes prerequisite knowledge with setup and operation of apache web servers and MySQL databases. Olio is not in itself a workload generator; it is the application that receives requests from a workload generator. It helps aid in server design decisions. The seven operations that Olio offers are all short duration tasks. This research effort requires a variety of task durations from seconds to hours; therefore, Olio is not an appropriate cloud server.

Cloudstone

Cloudstone is a multi-platform, multi-language performance measurement tool for Web 2.0 and Cloud Computing [14]. It deploys on an instance of the Amazon Elastic Cloud Computing (EC2) data center, and primarily measures database performance. Cloudstone uses Olio as a Web 2.0 application. Standalone deployment is possible using Olio and Faban Workload Generator [14]. Cloudstone is well suited for cloud performance measurement, but is not in itself a workload generator.

VMware VMmark

VMmark is a benchmark tool used to measure performance and scalability of applications running in virtualized environments [57]. VMmark has extensive hardware and software requirements compared to the aforementioned tools. VMmark enables users to measure, view, and compare virtual datacenter performance [5]. It utilizes two previously discussed toolkits, Rain and Olio. Overall, VMmark is well documented, but this research does not utilize VMmark due to time constraints.

Faban Workload Generator

Faban is a Markov-chain-based workload generator, and is widely used for server performance and load testing, also referred to as benchmarking [18]. It contains features

that measure and log key performance metrics, and automate statistics collection and reporting. Faban supports numerous servers such as Apache httpd, Sun Java System Web, Portal and Mail Servers, Oracle RDBMS, memcached, and others [18]. Perhaps the most important feature pertaining to this research is developers can build and modify realistic workloads.

Overall, Faban is well documented with manuals, tutorials, blogs, and other web documentation. Due to its distributed and scalable design, Faban is well suited for generating cloud computing workloads [18]. Consequently, Faban is the tool of choice for generating workloads in this research effort.

The R Project for Statistical Computing

R is a free and open source statistical analysis and graphics tool [31]. While R is not a workload generation tool, this research utilizes it extensively for data analysis, statistics collection, characterization, and graphics of cloud trace files.

Related Works

This research focuses on the analysis and synthesis of client-initiated workload characteristics contained within cloud trace files. The work most related to the work presented here is [58]. In [58], Wang et al. discuss analyzing and synthesizing realistic cloud workloads. The authors use a public 6-hour Google trace to design realistic cloud workloads, which drive the evaluation of Hadoop job schedulers and Hadoop shared storage system performance. The trace analysis focuses on job/task classification and resource utilization patterns. The authors attempt to predict future task and job behavior based on past information. MRPerf simulator is the workload generation tool utilized for

modeling MapReduce application performance. Wang et al. offer an algorithm for synthesizing realistic cloud traces based on pattern recognition, although the authors describe the results of the algorithm as a good first step towards workload generator development [58]. Unlike the work presented in this thesis, the authors assume statistical similarities between the analyzed trace and the synthesized workload.

Di et al. [17] describe a similar cloud workload analysis. The authors perform a limited characterization of job/task load and server load of a 29-day Google trace. The research presented in the article compares statistical similarities between the Google trace and grid traces with regard to client initiated workload and host load, with much of the focus on host load [17]. The authors claim significant difference between clouds and grids exist due to differences in user interaction and host applications.

Liu et al. [35] present characteristics of a 29-day Google trace, and focus on patterns of machine maintenance events and job/task behaviors. The authors study virtual machine management, job scheduling and processing, and cluster resource utilization. Liu et al. claim the Google trace discloses much information about how this particular Google cluster operates [35].

Chen et al. [11] developed a limited statistical profile of a 6-hour public Google trace. The authors cluster job types using the k-means clustering algorithm, and correlate job semantics and behavior [11]. Chen et al. claim the trace analysis provides system design insights, and make numerous implications regarding scheduling algorithms, cluster management, and capacity planning.

Reiss et al. [48] describe the scheduler request and utilization of a 29-day Google trace. The research characterizes cluster resource request, resource utilization, and

associated distributions. The authors show the overall trace consists of a large number of small requests, but a small number of large requests dominate its resource usage [48]. Reiss et al. claim they have found two scheduling characteristics that should be addressed in future scheduler designs: scheduler resource request and usage mismatches, and scheduling delays due to unrealistic task constraints [48].

Summary

This chapter defines cloud computing, discusses publically available traces, and highlights the differences between cloud and grid workloads. The Google trace data is a good fit for this thesis research for the following reasons: the Google trace is new, and most importantly, it represents a cloud workload. It may be the first of its kind ever made available to the public. Other companies and owners of cloud traces should follow suit for the benefit and advancement of cloud research. Faban workload generator is the tool of choice for this research for a few reasons: Faban is highly customizable, it has multiple levels of automation, and thankfully, it is well documented.

III. Workload Generation Heuristic

Chapter Overview

This chapter introduces a workload generation heuristic, which describes techniques that aid in the development of a synthetic workload. The heuristic may not generate optimized results like that of an algorithm, but it does provide reasonable results in an acceptable amount of time. Developers should adjust and fine-tune parameters as necessary to achieve desired results, characteristics, and distributions.

Workload Analysis

It is important that researchers become familiar with cloud workload trace properties and characteristics prior to generating a workload. First, researcher must locate and download the appropriate trace file. The following steps describe how to extract characteristics and properties from the trace as necessary for workload simulation. Use a statistical package, such as the R programming language, for computations and graphics.

This research effort focuses on four main categories of workload characteristics: Job Arrival Rate, Task Duration, Tasks per Job, and Running Tasks. Each category results in a statistical distribution, such as negative exponential or constant, as seen in the Google trace analysis in chapter 5. The characteristics of these distributions provide input to the synthetic workload design.

1. *Job Arrival Rate*
    a. Job Arrival Rate represents the number of unique connections to a server over time.
    b. Calculate the Job Arrival Rate using a statistical package, such as R, by tallying the number of unique jobs per interval of time. In the Google trace, for instance, each job is labeled with a unique *ParentID* number.

c. Using a statistical package, characterize the resulting distribution using the appropriate statistical methods, modeling, or curve fitting. For example, the Google trace Job Arrival Rate has a distribution that fluctuates around a constant average.
d. If the trace contains multiple jobs types, calculate the Job Arrival Rate and characterize the resulting distributions for each job type.

2. *Task Duration*
    a. Task Duration represents the time it takes for a server to process and respond to a client request.
    b. Find all occurrences of a unique task and note the time stamp information. Calculate Task Duration using a statistical package, such as R, by subtracting the smallest time stamp value from the largest time stamp value for each unique task in the trace. In the Google trace, for instance, each *ParentID* contains one or more tasks, labeled as *TaskID*. Each of these task entries contains time stamp information. Refer to Equation 1 for the Task Duration calculation.

$$TD = T_{final} - T_{init} \quad (1)$$

Where:
TD = Task Duration time
T_{final} = time stamp of last occurrence of unique task
T_{init} = time stamp of first occurrence of unique task

    c. Using a statistical package, characterize the resulting distribution using the appropriate statistical methods, modeling, or curve fitting. For example, the Google trace Task Duration has negative exponential distribution characteristics.
    d. If the trace contains multiple job types, calculate the Task Duration and characterize the resulting distributions for each job type.

3. *Tasks per Job*
    a. Tasks per Job represents the number of tasks contained within each job or unique client connection to the server.
    b. Tally the number of unique tasks within each unique job using a statistical package, such as R. In the Google trace, for instance, each *ParentID* is a job that contains one or more tasks, labeled as *TaskID*.

c. Using a statistical package, characterize the resulting distribution using the appropriate statistical methods, modeling, or curve fitting. For example, the Google trace Tasks per Job has negative exponential distribution characteristics.
d. If the trace contains multiple job types, calculate the Tasks per Job and characterize the resulting distributions for each type.

4. *Running Tasks*
    a. Running Tasks represents the presence of a task in a trace file. In the Google trace, for instance, a running task means the *TaskID* is present in the trace.
    b. Count the number of unique tasks per unit of time using a statistical package, such as R. In the Google trace, for instance, count the number of unique *TaskIDs* per five minute time interval.
    c. Using a statistical package, characterize the resulting distribution using the appropriate statistical methods, modeling, or curve fitting. For example, the Google trace Running Tasks has a distribution that fluctuates around a constant average.
    d. If the trace contains multiple job types, calculate the Running Tasks and characterize the resulting distributions for each job type.

Synthetic Workload Generator Design

This research utilizes Faban, a free and open source workload creation framework, as a synthetic workload generator. The characteristics and distributions from each of the four categories in the previous section determine the workload design parameters. The properties of the distributions help foster effective workload generator design decisions. It is important to understand the distribution properties prior to generating a workload.

1. Determine the total number of job launches. Within Faban, for instance, threads represent job launches, or unique client connection to the server.
    a. For a scaled down experiment, begin by reducing the actual cloud trace job launches by a factor of 100.
2. Establish the number of workload generation machines. Both Task Durations and Tasks per Job determine the number of workload generation machines. Multiple

machines are necessary due to limitations of Faban and may not be required for other workload generators.

a. Task Duration represents the time it takes for a server to process and respond to a client request.
b. Divide Task Duration into three categories (short is seconds, medium is minutes, long is hours). Begin by allocating one workload generation machine (computer or virtual machine) for each Task Duration category. For example, if a workload contains three Task Duration categories (short, medium, and long), the workload requires three workload generation machines. Keep in mind this number of machines can grow, depending on Tasks per Job.
c. Tasks per Job represents the number of tasks contained within each job or unique client connection to the server.
d. Divide Tasks per Job into categories. For example, 1-9 is small, 10-99 is medium, and 100 or more is large. The number of categories and category ranges may be altered as necessary per simulation requirements. If only one Tasks per Job category exists per Task Duration category, one Faban machine should suffice. For multiple categories, assign one category per machine.
e. Table 2shows an example requiring four Faban machines:

Table 2: Faban Machine Quantities

Task Duration	Tasks per Job	Faban Machines
Seconds	Small (1)	Machine 1
	Large (100)	Machine 2
Minutes	Small (1-4)	Machine 3
Hours	Small (1-4)	Machine 4

3. Set the number of job launches per workload generation machine. In Faban, for instance, threads represent job launches, which are set using the *scale* parameter. The number of job launches per workload generation machine affects the resulting workload distribution. Assign parameters on each workload generation machine such that the output mimics that of the distribution determined in the

workload analysis section. Choose a subset of critical points from the distribution in the workload analysis section that will result in similar distribution shapes.

a. Start with outliers, such as large (100) Tasks per Job, as seen in Table 2, Machine 2. In the Google trace, for instance, the number of clients requesting 100 Tasks per Job is relatively small compared to other categories. Assign a small number of jobs to the 100 Tasks per Job category, such as 1% of total jobs. For instance, in an overall Faban workload of 100 unique job launches, assign one thread to Machine 2.
b. Determine another workload generation machine that requires a small number of jobs. In the Google trace, for instance, the number of clients requesting Task Durations in the hour range is relatively small compared to other categories. Assign a small number of jobs to the hours Task Duration category, such as 5% of total jobs. For instance, in an overall Faban workload of 100 unique job launches, assign five threads to Machine 4.
c. Divide the remaining threads between the remaining Machines, as required to mimic the appropriate distribution. For example, assign 25 jobs to Machine 1 to simulate a portion of the workload with 25 client connections and one Task per Job.

4. Set the appropriate timing or delay parameter to define the number of Tasks per Job. In Faban, for instance, Tasks per Job cannot be set directly. Instead, define Tasks per Job indirectly using the *thinkTime* parameter. Increase the *thinkTime* parameter to decrease the number of tasks per job. This research develops the formula in Equation 2 to clarify the Tasks per Job calculation.

$$t_{steadystate} = TpJ\ (TD + TT) \quad (2)$$

Where:
$t_{steadystate}$ = total test duration time
TpJ = Tasks per Job
TD = Task Duration
TT = Think Time, simulates the amount of time that passes between tasks

5. Vary the workload arrival rate as necessary. Client-initiated workloads typically vary and are not flat. For example, the Google trace workload arrival rate fluctuates around a constant average. Workload generation tools may have a

feature that allows for workload variation. Faban, for instance, utilizes a load variation file to vary load patterns.

6. Workloads may require Task Duration category subsets, which provide more data points or resolution when simulating a distribution. For example, in Table 27, the minutes Task Duration category may require subcategories of 1, 5, and 20 minutes. In this case, multiple Task Durations are assigned to one machine. A feature may be present in the workload generation tool to accommodate this. In Faban, for instance, the *operationMix* parameter allows single threads to perform multiple separate operations, such as GET requests to multiple URLs. If such a feature is not available, assign one workload generation machine per subcategory.
    a. In Faban, a *Flat Mix* performs the same operation, such as a short duration HTML web request.
    b. In Faban, a *Probability Mix* will perform different operations based on probabilities assigned. Use this parameter to change the operation. For example, a workload may contain task durations that last minutes (1, 5, and 20 min), each at a unique URL. Assign probabilities as necessary for determining the number of times performing each operation.
7. Determine the total test duration, which is the workload runtime. In Faban, for instance, the *steadyState* parameter defines the total test duration.

Summary

This chapter defines a workload generation heuristic, which begins with a workload analysis of an actual trace file, and ends with a synthetic workload generation process. The synthetic workload output should have statistical similarities when compared to the characteristic results of the workload analysis. Developers should adjust the necessary parameters to obtain appropriate statistical similarities.

IV. Methodology

Chapter Overview

Analysis of the publically available Google cloud trace file for statistical characteristics focuses on client-initiated requests/workloads. This research uses the R statistical computing package for analysis of the Google trace. Some of the most important client workload characteristics include unique job launches, task duration, and tasks per job. Once the workload is analyzed and characterized, the results model a synthetic cloud workload. The resulting workload will have characteristics with statistical similarities to the Google trace. Figure 5 shows the overall methodological approach to this research effort.

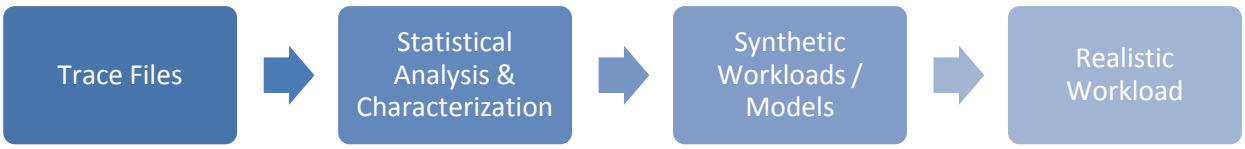

Figure 5: Methodology

Implementation of the spiral development process within the third step of Figure 5 facilitates synthetic workload generation and design. The spiral model encourages the addition new synthetic workload elements, as information becomes available. The earliest phase is quite simple with a focus on exploration and learning. Each iteration adds a level of complexity to the workload generator. The final iteration contains the necessary elements and parameters for production of a justifiably realistic synthetic workload.

Google Trace Statistical Analysis and Characterization

Google trace file statistical analysis results provide the foundation for modeling and generating synthetic workloads. Although the file is highly anonymous, there is an abundance of information available for both client and server side characterizations. Table 3 describes the parameters necessary for statistical analysis. The R statistical analysis package parses the Google trace file, analyzes the data, and plots the results.

Table 3: Google Trace Characteristics

Parameter	Description
Google Trace File	Publically available Google Trace File
Job Arrival Rate	Measure the number of jobs in 5 min intervals. Jobs are assigned unique identification numbers within the trace file.
Task Duration	Measure the task durations. All tasks are assigned unique identification numbers within the trace file.
Tasks per Job	Measure the tasks per job. Jobs are assigned unique identification numbers within the trace file.
Memory and CPU Consumption	Measure/plot the server-side memory and CPU usage.
Number of Running Tasks Job Type 0, 1,2, 3	Measure/plot the number of running tasks for each job type.
Task Duration by Job Type Job Type 0, 1, 2, 3	Measure/plot the task durations for each job type.
Tasks per Job by Job Type Job Type 0, 1, 2, 3	Measure/plot the tasks per job for each job type.

Characterizing and modeling the Google trace file provides the essential information for designing the synthetic workload. The synthetic workload will have statistically similar distributions.

Spiral Development

Synthetic Workload Generation Phase 1

Phase 1 of the spiral development process explores the Faban software and establishes a simple HTTP workload between a client and server. Faban offers a command-line utility called Faban Http Bench (fhb), which primarily tests the throughput of a single GET or POST request emulating some number of clients [18]. Faban's fhb utility provides a simple command line interface that automatically creates and compiles an HTTP driver from the command line arguments [18]. The HTTP driver executes, and a results summary is printed. Scalability within fhb is limited to one client machine, one Java Virtual Machine (JVM), and one URL, as seen in Figure 6. Fhb generates a workload by instantiating multiple client threads, each thread runs independently and maintains its own statistics.

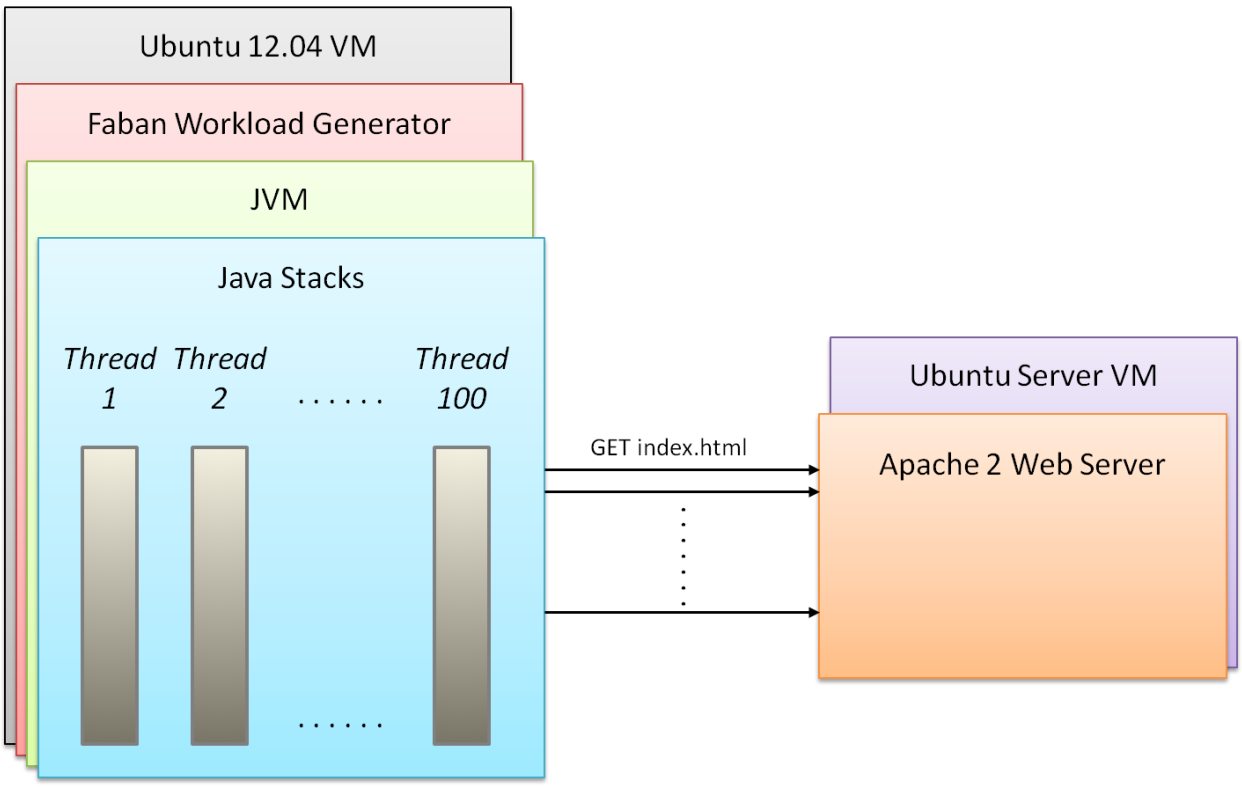

Figure 6: Spiral Development Phase 1

The fhb utility has numerous command line arguments to specify parameters and customize the workload. Table 4 lists the relevant options for creating a simple workload. More options are available beyond those listed in Table 4, located at faban.org. Phase 2 of the spiral development process investigates many of these options.

Table 4: fhb Options

Parameter	Description	Example
-J	Pass standard JVM option.	-J -Xmx600m, sets max JVM heap size
-r	rampUp/steady/rampDown	-r 60/300/60, time in sec for each interval
-W	thinkTime in milliseconds	-W 1000, wait one sec between client requests
-c	numThreads	-c 100, simulates 100 unique client connections

Once fhb is successfully communicating between client and server, and a solid understanding of all the options listed in Table 4 exists, the next phase in the spiral development process may begin. The fhb command line is a preliminary tool with limited functionality, and is not suitable subsequent development phases.

Synthetic Workload Generation Phase 2

Phase 2 of the spiral development process integrates the Faban configuration file. Rather than passing fhb options via command line, the configuration file contains all the parameters that control the workload. Users invoke an Extensible Markup Language (XML) encoded file from the command line. The XML configuration file offers more parameters than fhb, thus promoting the development of complex workloads. For example, a configuration file may include workloads with multiple URLs and varying load.

Threads

Each thread represents a unique connection to the server -- that is, each thread is a client in a logical sense [18]. Understanding the concept of threads, and how Faban uses them, is important. The number of threads equals the number of clients, which is also equal to the number of jobs. Each thread simulates one client or job. For example, launching 50 threads simulates 50 unique clients or jobs. In addition, each thread utilizes a unique port number. For example, 25 threads contain 25 unique port numbers.

Java Virtual Machines

JVM heap size limits the size of the workload. The JVM can and will run out of memory. The following workload characteristics affect the amount of memory used within the JVM.

- Number of jobs
- Number of tasks per job
- Number of running tasks

JVMs not allocated enough memory to perform the workload throw an error. To correct this issue, increase the amount of memory that the JVM uses by changing the appropriate argument. For example, change the argument -Xmx600m to -Xmx2048m. This will increase the heap size from 600 MB to 2 GB. The operating system and underlying hardware limit the max heap size.

Cycle Time and Think Time

Cycle Time and Think Time are timing delays that help regulate workloads. These timing parameters are important because they allow developers to regulate significant properties such as server load and task per job. Without them, threads would continuously run back-to-back tasks with no delay in between. Although Cycle Time and Think Time are similar, there is a subtle difference between the two parameters.

Cycle time represents the inter-arrival time between successive requests arriving at the server [55]. The frequency of the requests remains the same, even if the server is slows down, thus causing task duration or response time to increase. A large response time, while cycle time remains the same, increases load and degrades performance on the server [55]. Figure 7 shows one thread with three tasks per job. Notice the cycle time remains the same, even when the task duration varies.

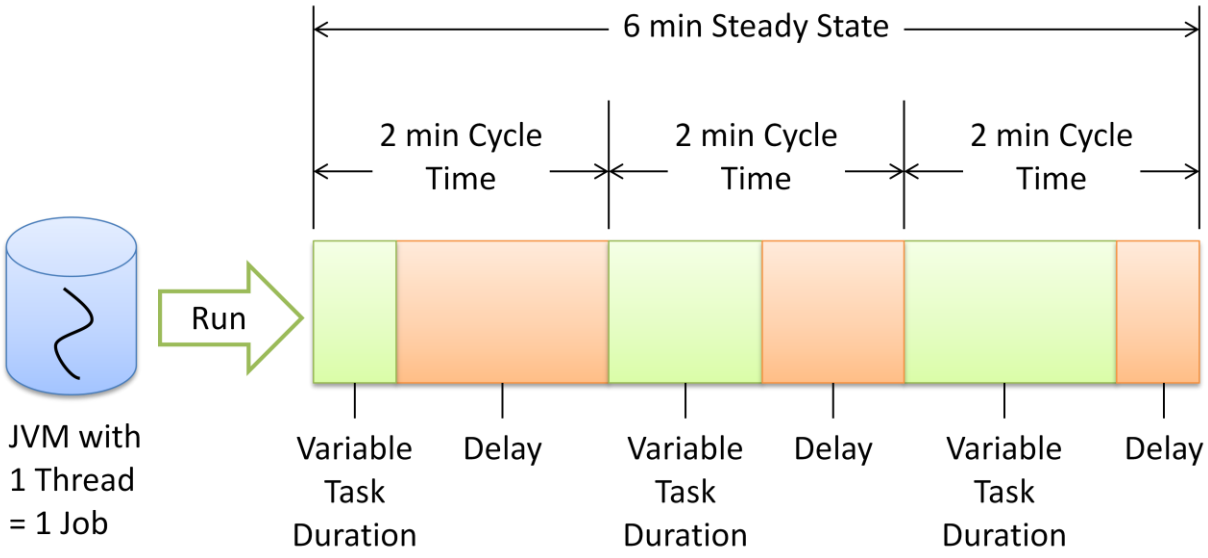

Figure 7: Cycle Time - Three Tasks per Job

Think Time represents the time interval that passes while a user reviews data presented on their screen and decides what to do next [55]. Developers can emulate Think Time in the configuration file. Closed systems typically use Think Times, where known client population and client interaction with the server exist [18]. For cloud computing synthetic workload generation, Think Time applies in a slightly different manner. Think Time helps define the number of tasks per job. In Figure 8, one thread runs for a steady state duration of 6 min, the task duration is 1 min (time it takes for the server to process and respond to the client requested task), and Think Time is set 2 min. The result is a single thread (one job) that runs two separate tasks, i.e. two tasks per job.

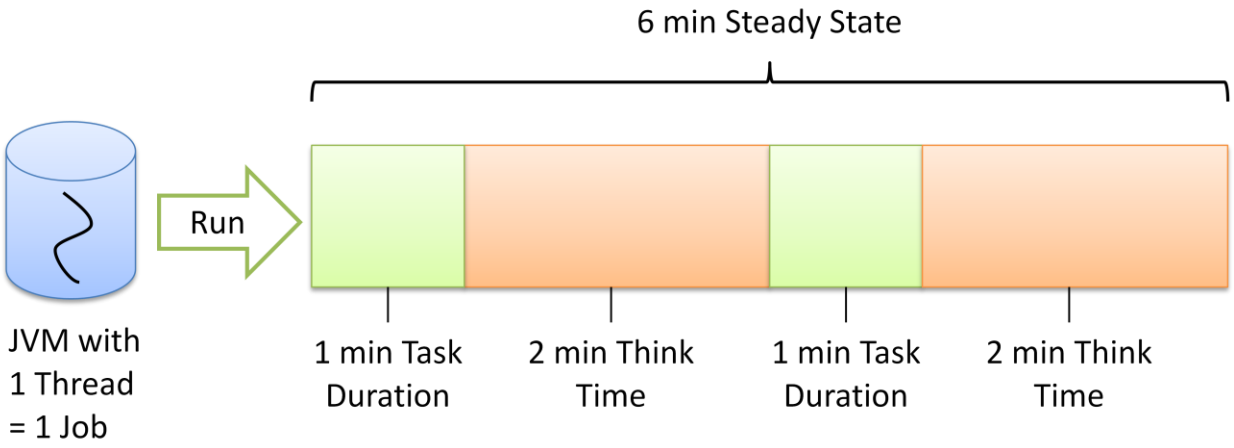

Figure 8: Think Time - Two Tasks per Job

Workload generation experiments utilize Think Time as opposed to Cycle Time for time delay parameters. Preliminary exploration during Phase 2 of the spiral development process shows Cycle Time does not perform as advertised. Results are inconsistent and at times unpredictable. Therefore, Think Time is the preferred time delay parameter used for all Faban synthetic workload generation experiments.

Load Variation

Constant load patterns are user or application requests that do not vary, and can be useful when stress testing a web application or server. On the other hand, variable workloads are important because real world workloads do have variation, especially public clouds. Fortunately, sound provisioning of data centers and clouds allow them to handle such variation. Server loads vary based on user interaction and requests. Varying the scale of the workload is necessary to simulate real user requests. Load variation is also useful for testing how well elastic cloud management techniques adapt to load variation [18].

Faban has a load variation feature for scaling workloads. A workload generation run receives a load variation file at submission time [18]. Creation of the requested driver threads for the maximum load occurs at the beginning of the run [18]. Extra threads remain idle until needed, and return to an idle state between and after uses [18].

The load variation file is an extensionless file that contains load level records, one load per line. Each record is a comma-separated pair of values in the structure <runtime in sec>,<thread count> as seen in Figure 9. The runControl element of the configuration file calls the variable load file. The example in Figure 9 shows a load of 500 threads for 300 seconds, followed by a load of 700 threads for 600 seconds, and finally a load of 600 threads for 300 seconds. Note the <*scale*> element of the configuration file represents the number of threads in the workload, and is equal to the largest thread count value in the variable load file. Also, note the <*steadyState*> element is equal to the sum of the runtime values in the variable load file.

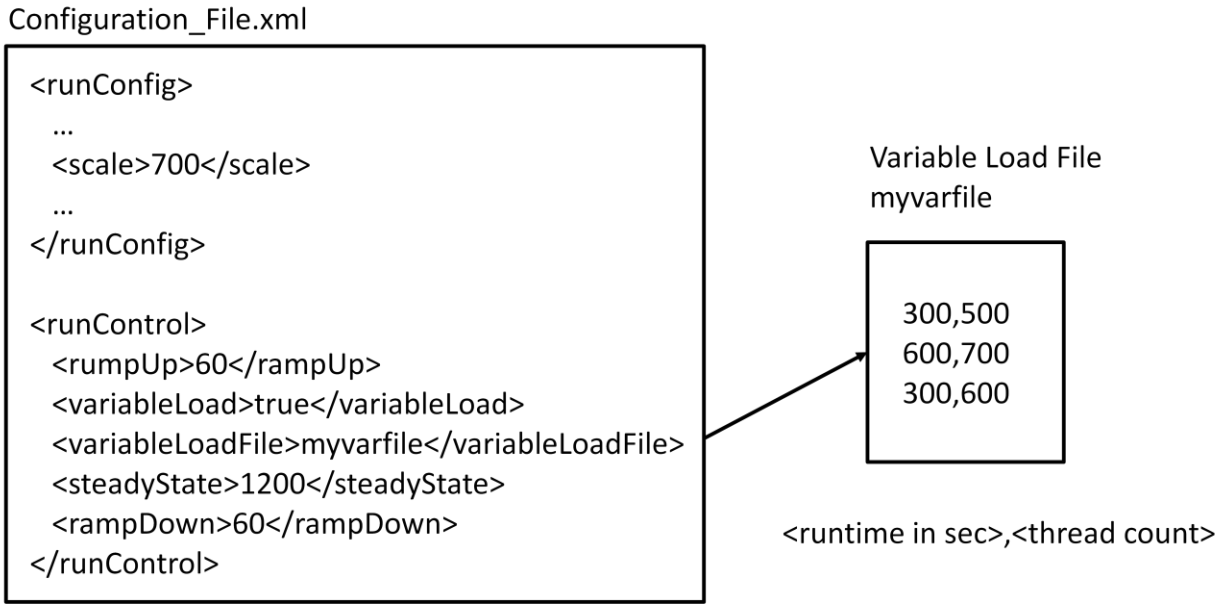

Figure 9: Variable Load File

Operation Mix

User interactions with cloud servers vary in load, and the type of operation varies. Users and applications send a variety of requests to servers. When simulating real workloads, it is necessary to emulate this type of client request variation. Sending identical HTTP GET/POST requests for the same URL is insufficient and does not represent real workloads. Fortunately, the Faban workload generator offers several different operation mixes [18]:

- Flat Mix - chose the next operation based on assigned probability
- Matrix Mix - maintain state and chose next operation based on current operation and probability ratio (Markov chain model)
- Fixed Sequence - call operations in sequence
- Flat Sequence Mix - select fixed sequences based on assigned probability

Phase 2 of the Spiral Development explores a few critical elements of realistic workloads. Threads simulate the number of unique client connections to the server. Think Times help define the number of tasks per job. The variable load file specifies the number of threads and execution duration, which allows for workload variation. Finally, the operation mix diversifies the type and sequence of operations. Figure 10 illustrates the utilization of these parameters.

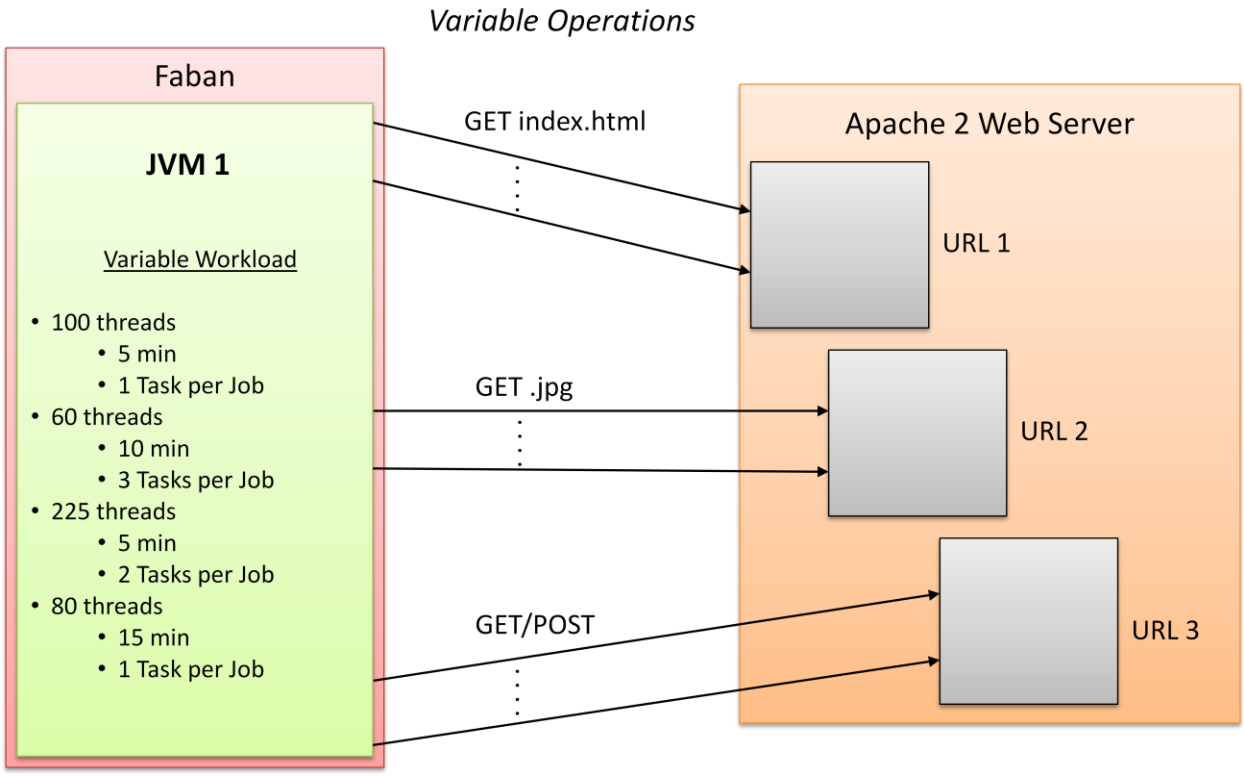

Figure 10: Spiral Development Phase 2

Synthetic Workload Generation Phase 3

The third and final phase of the spiral development includes multiple machines running simultaneous Faban workloads. The heuristic from chapter 3 of this determines the number of workload generation machines, as seen in Table 2. Each machine will simulate workloads with a unique set of characteristics. For example, one machine produces workloads that request short duration tasks and contain a small number of tasks per job, while another machine produces workloads that request long duration tasks and contain one task per job, as seen in Figure 11. The goal of phase 3 is to generate an

overall workload that has statistical similarities to the Google trace. Developers must have a good understanding of cloud trace characteristics prior to generating a workload.

The heuristic provides a framework for dividing the appropriate workloads between workload generation machines. Developers characterize the resulting workload output and compare for statistical similarities to the analyzed trace, then adjust input parameters for desired results. Number of job launches and tasks per job are categorical parameters commonly adjusted when simulating distributions and fitting curves.

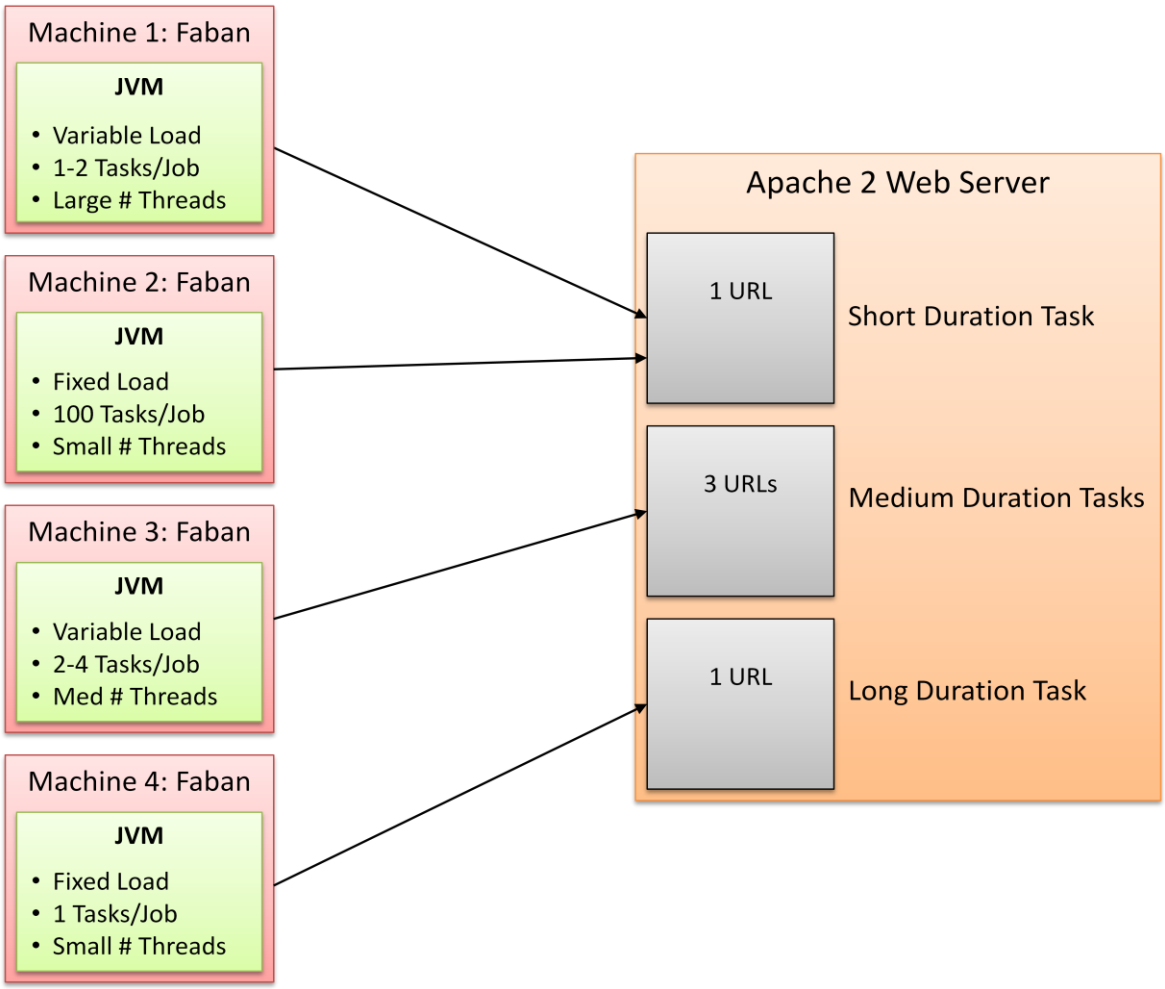

Figure 11: Spiral Development Phase 3

Characteristics Experiment Setup

The spiral development process leads up to this point where Google trace characteristics are simulated. The following sections document the parameters and settings for virtual machines, as well as the Faban configuration file.

Virtual Machine Configurations

Performance of all experiments occurs in a virtual computing environment using VMware Workstation. Utilizing virtual machines consolidates resources and eliminates the need for separate networked computers. Five virtual machines, one server acting as the cloud and four clients simulating hundreds of users and a variety of workloads, are able to communicate within a private network. The network adapter hardware setting is set to "Bridged" mode, which allows multiple virtual machines to talk to each other within the private network. Utilization of the ping command verifies communication between multiple host machines. The server and four clients utilized in the experiment have configurations with operating systems, software, and network settings, as seen in Table 5 and Table 6 below.

Table 5: Server Configuration

Parameter	Description
Operating System	Ubuntu Server 12.04.1 LTS
Web Server	Apache/2.2.22 (Ubuntu)
Web Server Administration	Webmin 1.580
Private IPv4 Address	192.168.1.109

Table 6: Client Machine Configurations

Parameter	Description
Operating System	Ubuntu Desktop 12.04 LTS
Java	Java SE Development Kit 7
Workload Generator	Faban 1.0.2
Private IPv4 Address	Client 1: 192.168.1.119 Client 2: 192.168.1.115 Client 3: 192.168.1.118 Client 4: 192.168.1.117

Faban Parameters

Machine 1

Machine 1 simulates a workload with 15 client connections, task duration less than one minute, and one task per job. The task is a HTML GET request to the same URL.

Table 7: Faban Parameters Machine 1

Parameter	Value	Description
Steady State	3600 sec	Run experiment for 1 hour.
Think Time	299000 ms	Wait 299 sec after each task. Increase Think Time to decrease number of tasks per thread.
Variable Load Threads	0,3,1,0,2,1,0,2,3,0,2,2	Number of threads for the specified Variable Load Duration. Each thread simulates a new user or job.
Variable Load Duration	300 sec	5 min interval
Mix	Flat Mix	HTML Web Request (1 URL)
Num Threads	3	Number of Unique Threads in JVM. Increase the number of threads to increase number of simulated users.
Num Tasks		Measure the number of tasks.
Task Duration		Measure the task duration.
Tasks / Job		Measure the number of tasks per thread. Threads are assigned unique port numbers.

Machine 2

Machine 2 simulates a workload with 1 client connection, short task duration less than one minute, and 100 tasks per job. The task is a HTML GET request to the same URL.

Table 8: Faban Parameters Machine 2

Parameter	Value	Description
Steady State	3600 sec	Run experiment for 1 hour
Think Time	35500 ms	Wait 35.5 sec after each TCP session. Increase Think Time to decrease number of tasks per thread.
Variable Load	N/A	No load variation
Mix	Flat Mix	HTML Web Request (1 URL)
Num Threads	1	Number of Unique Threads in JVM.
Num Tasks		Measure the number of tasks.
Task Duration		Measure the task duration.
Tasks / Job		Measure the number of tasks per thread. Threads are assigned unique port numbers.

Machine 3

Machine 3 simulates a workload with nine client connection, task durations in the 5 to 30 min range, and 1-4 tasks per job. The tasks are three file downloads from three different URLs.

Table 9: Faban Parameters Machine 3

Parameter	Value	Description
Steady State	3600 sec	Run experiment for 1 hour.
Think Time	250000 ms, 250000 ms, 149500 ms, 99500 ms, 74500 ms	Wait after each operation. Increase Think Time to decrease number of tasks per thread.
Variable Load	7,4,1,7,6,4,3,2,4,9,3,5	Number of threads for the specified Variable Load Duration. Each thread simulates a new user or job.

Variable Load Duration	300 sec	5 min interval
Mix	Probability Mix (0.4, 0.4, 0.1, 0.05, 0.05)	5 operations with respective Think Times and probability of occurrence.
Num Threads	9	Number of Unique Threads in JVM. Increase the number of threads to increase number of simulated users.
Num Tasks		Measure the number of tasks.
Task Duration		Measure the task duration.
Tasks / Job		Measure the number of tasks per thread.

Machine 4

Machine 4 simulates a workload with threads representing five client connections; the task duration is in the 1-hour range, and one task per job. The task is not a file download as seen on Machine 3. It is a series of delays on the web server simulating a long duration task. Large file downloads in the GB range create system instabilities within this small virtual environment, therefore large downloads are avoided.

Table 10: Faban Parameters Machine 4

Parameter	Value	Description
Steady State	3600 sec	Run experiment for 1 hour.
Think Time	3600000 ms	Wait 1 hour after each server request. Increase Think Time to decrease number of tasks per thread.
Variable Load	N/A	No load variation
Mix	Flat Mix	Server Request (1 URL)
Num Threads	5	Number of Unique Threads in JVM. Increase the number of threads to increase number of simulated users.
Num Tasks		Measure the number of tasks.
Task Duration		Measure the task duration.
Tasks / Job		Measure the number of tasks per thread. Threads are assigned unique port numbers.

Scalability Experiment Setup

The scale of the actual Google workload is much larger than that of the characteristics experiment. The experiment reduces the number of unique job launches by a factor of 120. The intention is to show statistical similarities in workload distributions. The question may arise; can Faban handle the scale of a real cloud workload? The scalability experiment will provide an answer to that question.

Faban Parameters

Max Tasks per Job

The experiment will verify Faban is capable of producing thousands of job requests from one individual thread. The test parameters come from the max tasks per job from the Google trace.

Table 11: Max TPJ

Parameter	Value	Description
Steady State	300 sec	Run experiment for 5 min.
Think Time	60 ms	Wait 60 msec after each task. Increase Think Time to decrease number of tasks per thread.
Mix	Flat Mix	HTML Web Request (1 URL)
Num Threads	1	Number of Unique Threads in JVM. Increase the number of threads to increase number of simulated users.
Num Tasks		Measure the number of tasks.
Task Duration		Measure the task duration.
Tasks / Job		Measure the number of Server Requests per thread. Threads are assigned unique port numbers.

Mean Tasks per Job

This portion of the experiment will verify if Faban is capable of producing hundreds of job requests per thread from hundreds of individual threads. The mean tasks per job test parameters come from a section within the Google trace.

Table 12: Mean TPJ

Parameter	Value	Description
Steady State	300 sec	Run experiment for 5 min.
Think Time	575 ms	Wait 575 msec after each task. Increase Think Time to decrease number of tasks per thread.
Mix	Flat Mix	HTML Web Request (1 URL)
Num Threads	120	Number of Unique Threads in JVM. Increase the number of threads to increase number of simulated users.
Num Tasks		Measure the number of tasks.
Task Duration		Measure the task duration.
Tasks / Job		Measure the number of Server Requests per thread. Threads are assigned unique port numbers.

Summary

The methodology of this research effort begins with choosing an appropriate trace file, followed by analysis and characterization of a publically available Google trace. Next is modeling the workload, and ending with the production of a synthetic workload with characteristics similar to that of the Google trace.

Developers must understand the complex Faban software prior to beginning synthetic workload generation. Due to the complex nature, the spiral development model is applied. Consecutive phases include additional parameters and modification of these parameters until particular synthetic workload characteristics result. There are two

primary experiments using the Faban workload generation toolkit. The first and primary experiment simulates the Google trace characteristics, while the second experiment verifies the scalability of the Faban software. The purpose of the scalability test is to verify the Faban toolkit can recreate a large number of client requests as seen in real workloads.

V. Analysis and Results

Chapter Overview

This chapter is comprised of four main areas. First are the characteristic results from the statistical analysis of the Google trace. Second is the analysis and characterization of the synthetic workload simulation. Next is a statistical comparison of the Google trace and synthetic trace. This chapter ends with a Faban scalability results summary.

Results of Google Trace Analysis

The Google trace contains important characteristics necessary to simulate a client workload. Data from job rate of arrival, server memory and CPU core usage, task durations, tasks per job, and number of running tasks is contained within the trace. Extracting and characterizing this data by utilizing the heuristic in chapter 3 produces distributions that are essential for generating synthetic traces with statistical similarities.

Job Launches by Job Type

The heuristic in chapter 3 provides a framework for job launch characterization. Part 1 in the *Workload Analysis* section of the heuristic produces particular job arrival rate characteristics of the Google trace. The Google trace has four different job types (0, 1, 2, and 3). Figure 12 contains a tally of unique job launches for each job type in five-minute intervals.

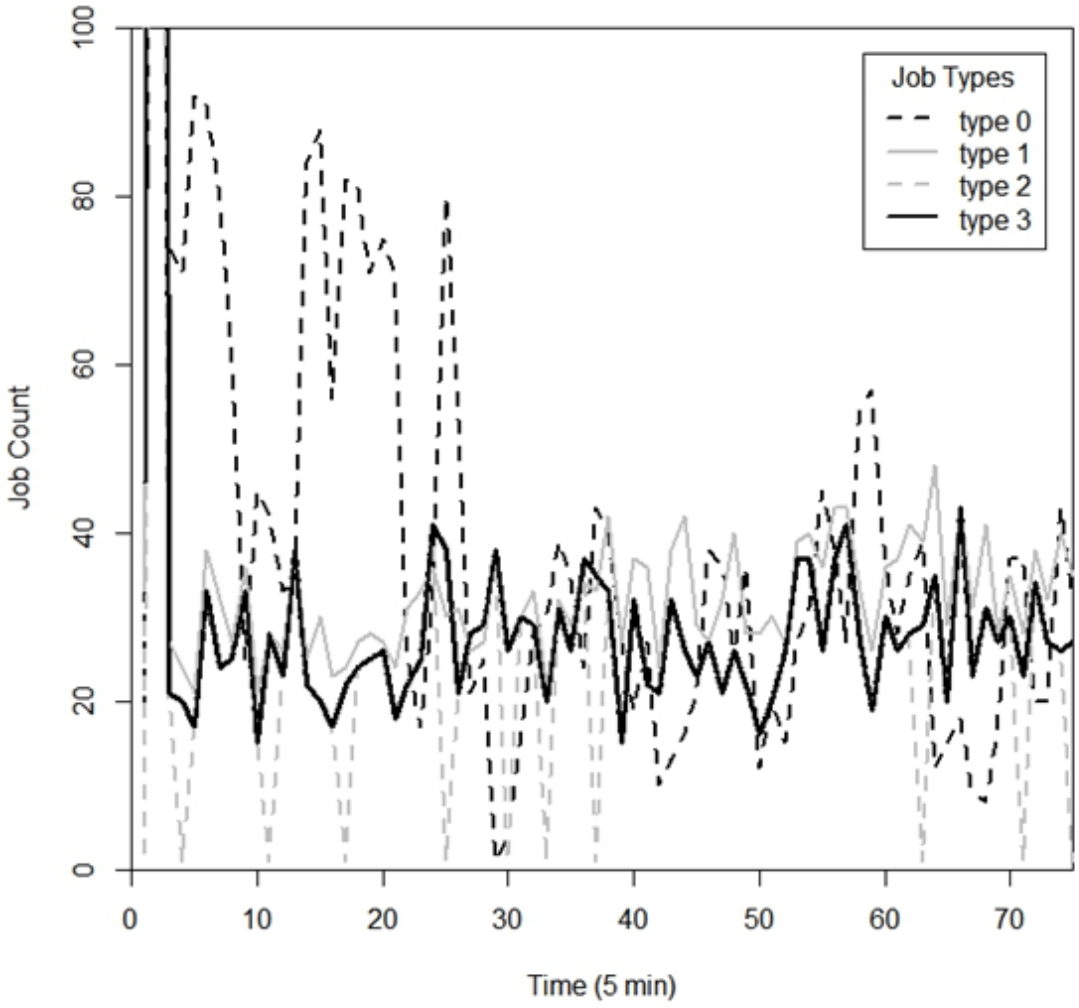

Figure 12: Google Cluster Unique Job Launches

Table 13 presents a statistical summary of unique job launches, or job arrivals.

Table 13: Job Launches

Parameter	Job Arrival Rate (5 min intervals)	Job Arrival Rate (1 hr intervals)
Min	57	1315
Max	195	1641
Mean	120	1428
Standard Deviation	25	121

Normalized CPU and Memory Consumption

The Google trace contains CPU and Memory consumption measurements. Even though client-side synthetic trace simulation does not use these measurements directly, it is important to understand server side characteristics. Correlation and covariance statistics show how CPU and memory are related.

Figure 13 shows a strong visual correlation between memory and CPU core usage. The covariance and correlation data prove memory and CPU usage have a relationship. This data represents measurements of the cloud computing system, not the clients. In addition, the memory and CPU usage fluctuate around a constant average, which may imply a relatively constant load.

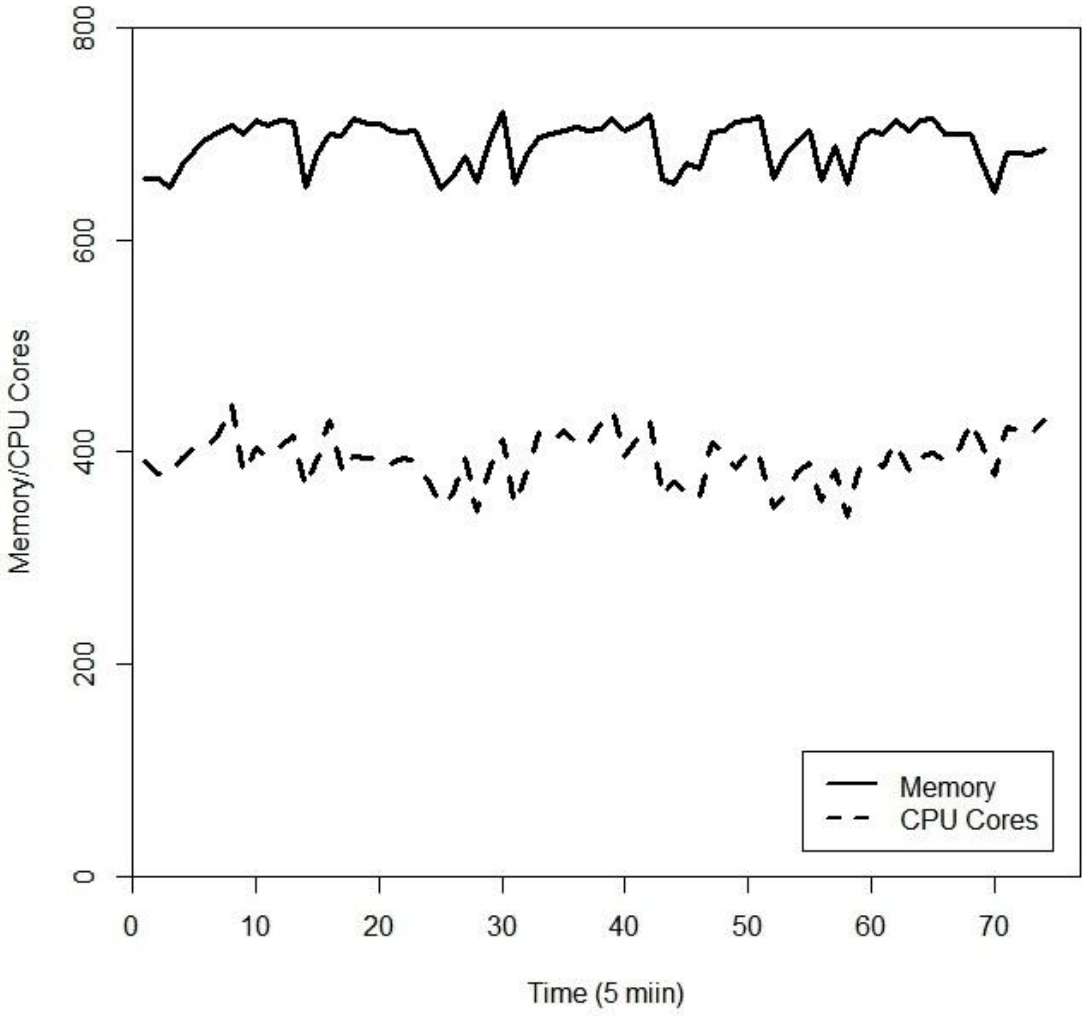

Figure 13: Google Cluster Normalized Memory and CPU Consumption

Pearson's Product-Moment Correlation

The Pearson product-moment correlation coefficient measures the strength of linear dependence between memory and CPU consumption. Table 14 contains the correlation value of the relationship.

Table 14: Correlation Between Memory and CPU

Correlation Coefficient (r)	Degrees of Freedom	p-value	95% Confidence Interval	r^2
0.6668	73	8.8e-11	0.517 to 0.777	0.445

The correlation coefficient value r is equal to 0.6668, which indicates a positive relationship between memory and CPU consumption. The square of the coefficient is equal to the percent of the variation in one variable that relates to the variation in the other. The square of the coefficient equals 44.5%. The r^2 value is > 25%, which is a strong effect size [15]. In addition, the P-value is less than 5%, so it is statistically significant.

Covariance

The covariance of the two data sets is 334.3756. It indicates a positive linear relationship between the two variables. Since the covariance is > 0, there is a tendency for large values of memory consumption to be associated with large values of core consumption, and vice versa.

Task Duration

The heuristic in chapter 3 provides a framework for task duration characterization. Part 2 in the *Workload Analysis* section of the heuristic produces particular task duration characteristics of the Google trace. The bar graph in Figure 14 shows a negative exponential distribution of task durations. All task duration calculations result from Equation 1. The majority of tasks is on the left side of the plot, and represents tasks that run for seconds and minutes. More specifically, the first 12 data points account for 73% of all tasks. The first 15 entries account for 20% of all data points and 73.5% of all tasks.

This ratio is similar to the 80/20 rule or Pareto principle, a specific type of long tailed distribution. The exception is the spur that appears at the end of the negative exponential distribution. The spur represents full-length duration tasks (greater than the length of the trace), and is removed from this plot. Had the trace recorded indefinitely, it is assumed the spur would not be present, and the distribution would likely taper off into a longer tail. The value of the spur is 35,206 tasks.

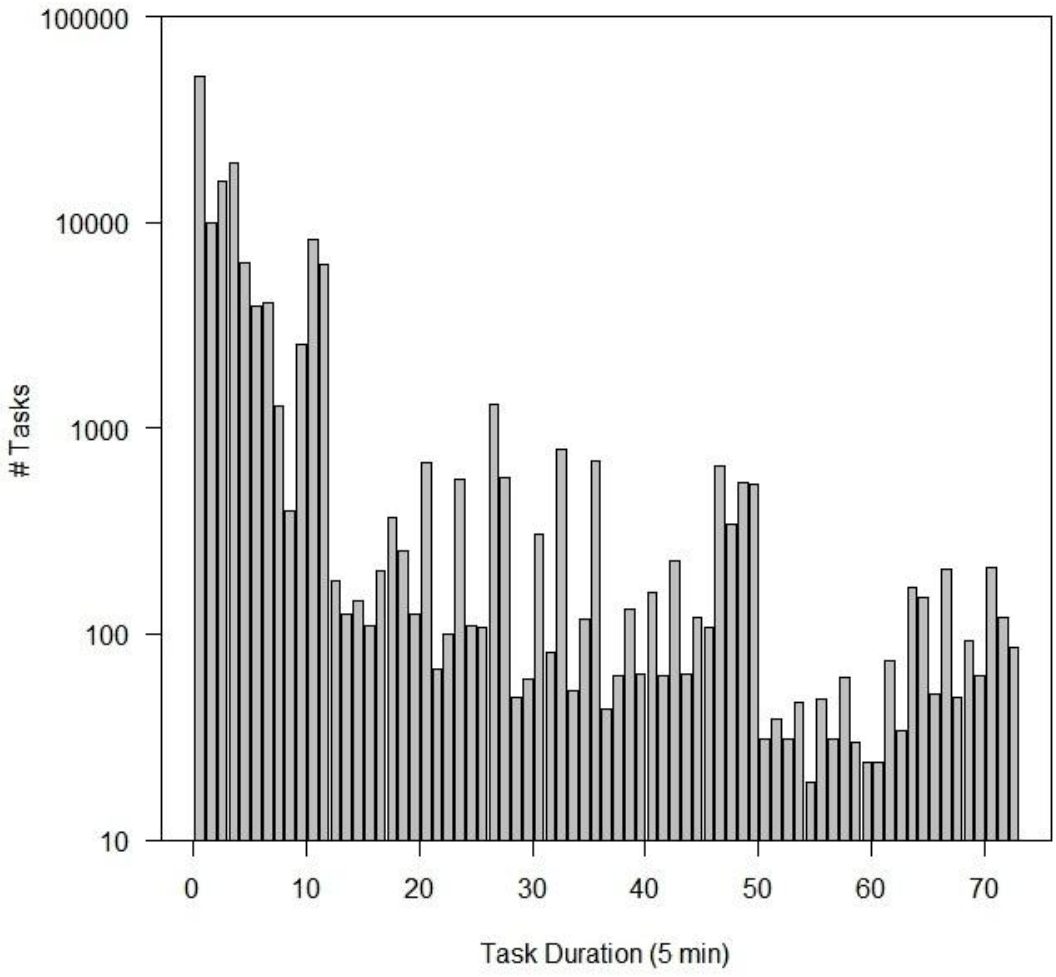

Figure 14: Google Cluster Task Duration

The decay rate estimates throughout this chapter are a result of the formula in Equation 3 below [43]. The decay rate estimate for Google task duration from Figure 14 is -6.9%.

$$P_t = P_o e^{-rt} \quad (3)$$

Where:
P_t = the final quantity at time t
P_o = the initial quantity at time t = 0
e = mathematical constant ≈ 2.71828
r = the rate of decay
t = time or number of periods

Figure 3 displays task durations without regard to job type. Plotting task duration for individual job types may provide additional details and insight into workload behavior. All four job types appear to have qualities of a negative exponential distribution, as seen in Figure 15.

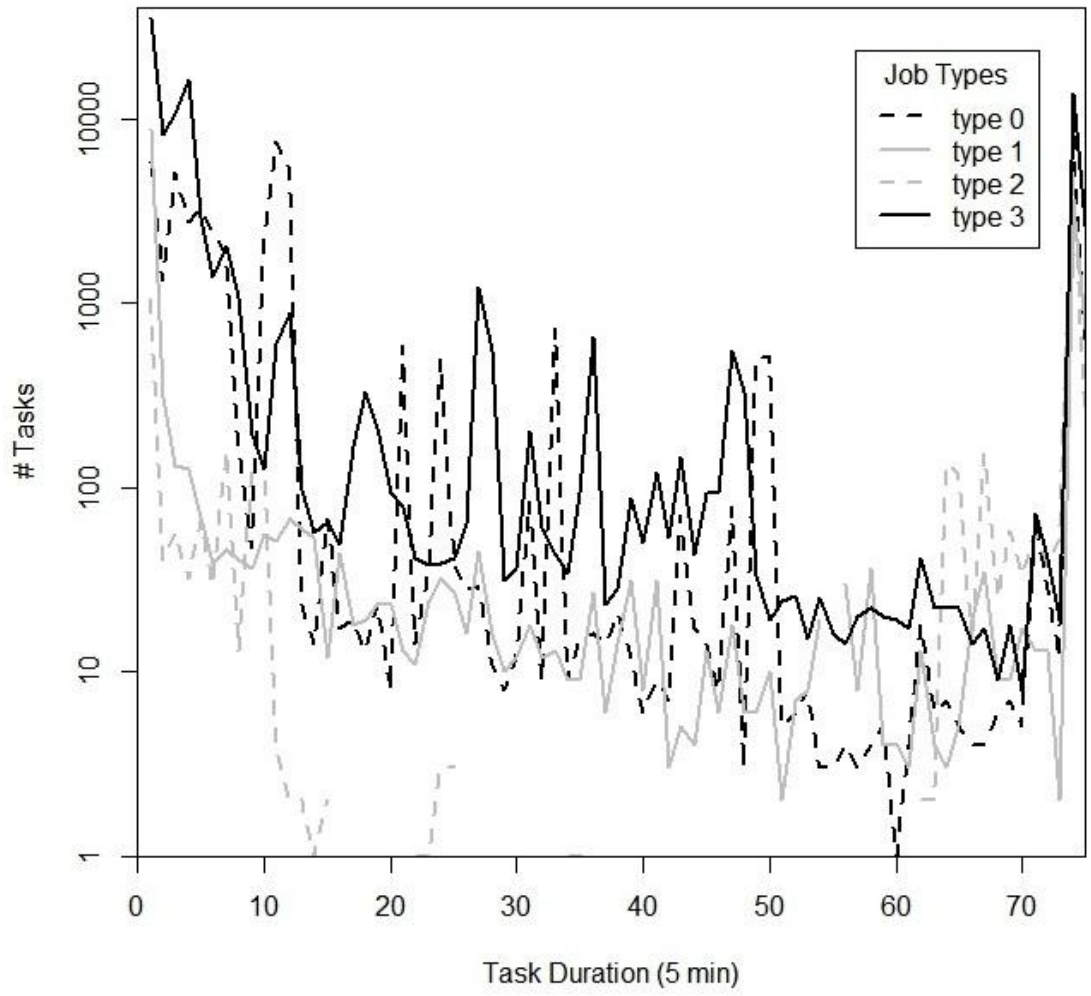

Figure 15: Google Cluster Task Duration by Job Type

A closer look at the different job types reveals all job types have short and long task durations that range from seconds to hours, as seen in Table 15. This characteristic is important when designing the synthetic trace.

Table 15: Task Durations: % of Total Jobs

	All Jobs	Job 0	Job 1	Job 2	Job 3
Seconds	29%	35%	12%	59%	13%
Minutes	44%	43%	63%	7%	9%
Hours	7%	6%	7%	6%	9%
Full-Length	20%	16%	18%	28%	69%

Three job types (0, 1, 2) all follow a negative exponential distribution. The exception is Job Type 3, which follows a U-shaped distribution, with the maximum frequencies at the two extremes of the range, as seen in Figure 16.

Figure 16 does not show the spurs representing full-length duration tasks. Had the trace recorded indefinitely, it is assumed the spurs would not be present, and the distribution would likely taper off into a longer tail. The values of the spurs are in Table 16.

Table 16: Full Length Tasks

	Job 0	Job 1	Job 2	Job 3
Task Count	16389	9123	4240	5454

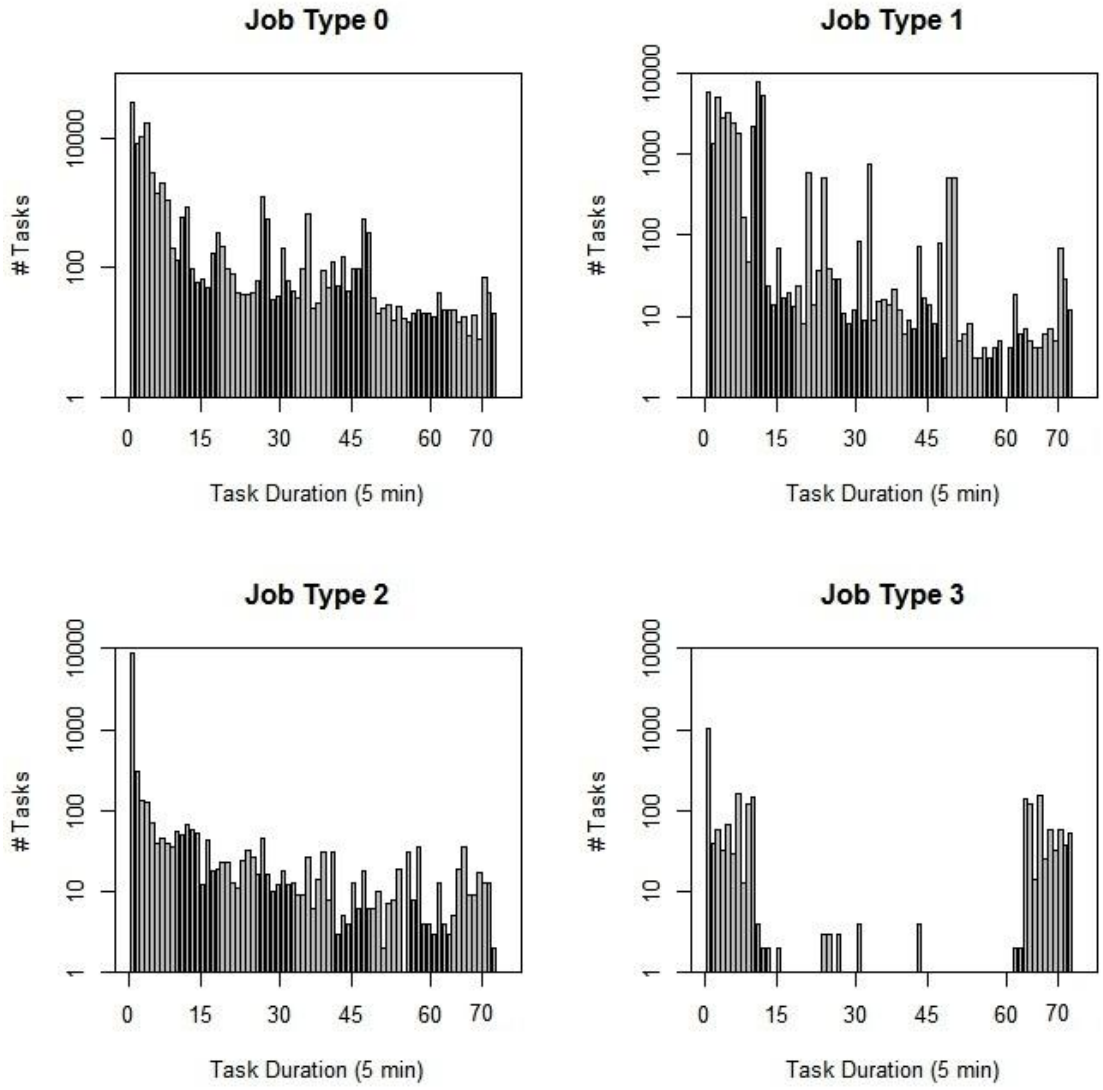

Figure 16: Google Cluster Bar Plot Task Duration by Job Type

Table 17: Task Duration Decay Rate

	Job 0	Job 1	Job 2	Job 3
Decay Rate	-8.3%	-6.7%	-7.8%	N/A

All four Job Types in Figure 16 have short and long task durations. This characteristic is important when designing the synthetic trace because it may not be necessary to simulate individual job types. All four job types have tasks that last seconds, minutes, and hours. Consequently, the simulation is simplified by lumping the categories together, thus eliminating the need to separate and simulate individual job type categories.

Tasks per Job

The heuristic in chapter 3 provides a framework for tasks per job characterization. Part 3 in the *Workload Analysis* section of the heuristic produces particular tasks per job characteristics of the Google trace. Tasks belong to jobs, and jobs may have multiple tasks. In the Google trace, some jobs contain one task while others contain thousands of tasks. Table 18 summarizes these characteristics and shows a wide range of tasks per job.

Table 18: Tasks per Job Characteristics

Parameter	Tasks per Jobs
Min	1
Max	4880
Decay Rate	-2.7%

Tasks per Job Distribution

The long tail statistical distribution has a high number of occurrences followed by a low number of occurrences, which gradually fades off in an asymptotic curve [43]. The events that occur at the far end of the tail have a very low probability of occurrence. A large share of the population (number of data points) lies in the tail.

Table 19 shows the few points that dominate the left side of the graph. For example, the first two data points (1 and 2 tasks per job) account for approx 81.7% of the total number of jobs. In addition, 86% of all jobs have five tasks or less. The first five data points account for just 2.1% of the total number of data points, but account for 86% of all jobs. In summary, most jobs contain few tasks, a few jobs contain thousands of tasks, and much of the population lies in the tail. Thus, the tasks per job characteristics of the trace data follow a long tail statistical distribution. Figure 17 shows the long tail negative exponential distribution. Notice that both axes use a logarithmic scale. A linear scale on the x-axis would result in a much steeper curve of the exponential decay.

Table 19: Small # Tasks per Job

# Tasks Per Job	# Jobs	% of Total Jobs
1	6746	73.2%
2	782	8.5%
3	174	1.9%
4	97	1.1%
5	121	1.3%

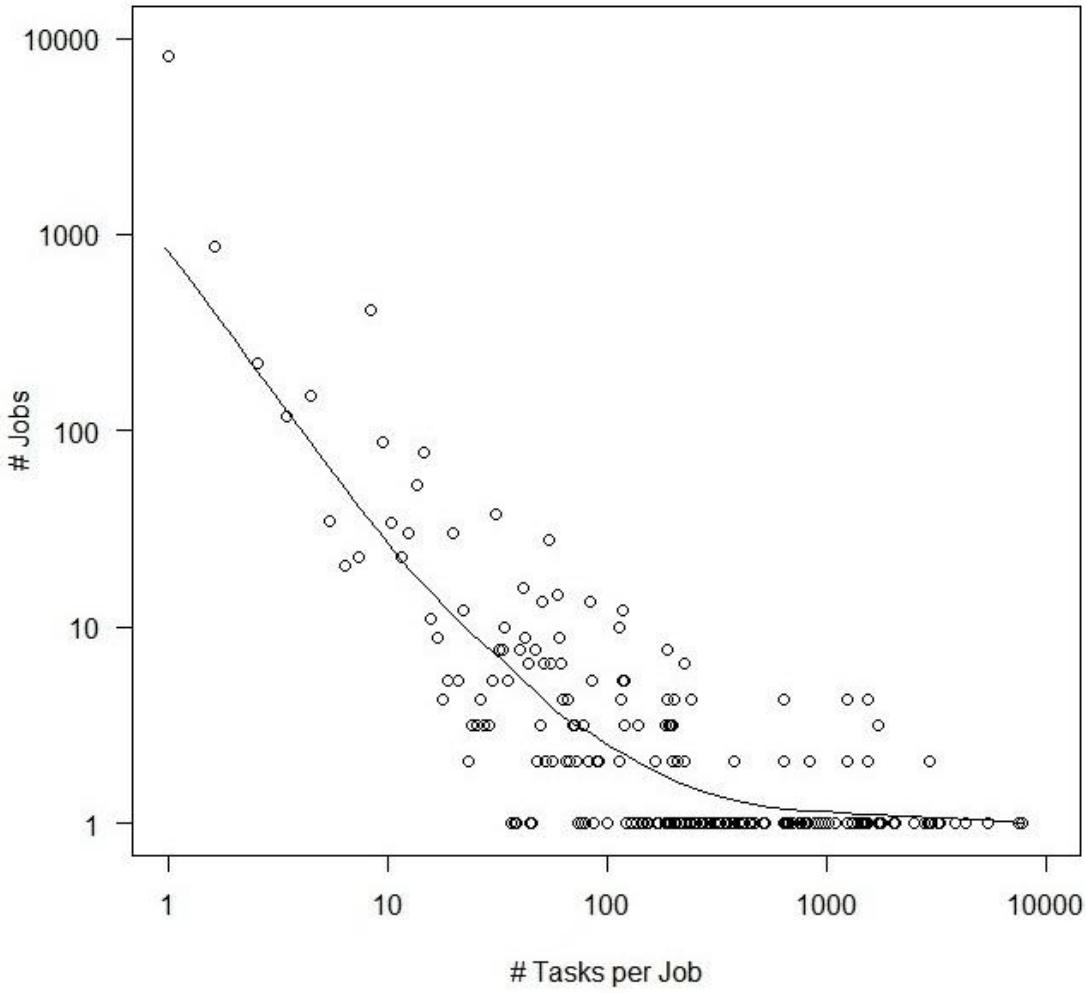

Figure 17: Google Tasks per Job w/ Non Linear Regression Fit

Google 2 trace follows a very similar task per job distribution, as seen in [35]. Liu [35] claims the jobs with a few tasks, rather than a few jobs with many tasks, drives the overall system throughput of the Google 2 trace. This is because jobs with one task dominate the left side of the plot, as seen in Figure 17. The only two cloud traces publically available at the time of this writing follow this characteristic: most jobs contain few tasks.

Smoothing

The *scatter.smooth* command in R is a smoothing function fitted by the LOESS algorithm, a locally weighted polynomial regression model [51]. The LOESS function allows the tracing of a smooth curve through a plot, as seen in Figure 17. The polynomial is fit to a subset of the data, using weighted least squares, giving more weight to the nearest points and less weight to points further away, as in the k-nearest neighbor algorithm [12]. The object of the nonlinear nonparametric regression fit is to estimate the regression function *f( )* directly [24]. The LOESS algorithm attempts to fit the model in Equation 4.

$$y_i = f(x_i) + \varepsilon_i \quad (4)$$

Where:
f = unspecified regression function
x_i = corresponding data point
ε_i = random error

Tasks per Job by Job Type

Figure 17 shows tasks per job without regard to job type. Plotting tasks per job for individual job types may provide additional details and insight into workload behavior. All four Job Types appear to have some qualities similar to that of a negative exponential distribution, especially Job Types 0 and 1, as seen in Figure 18.

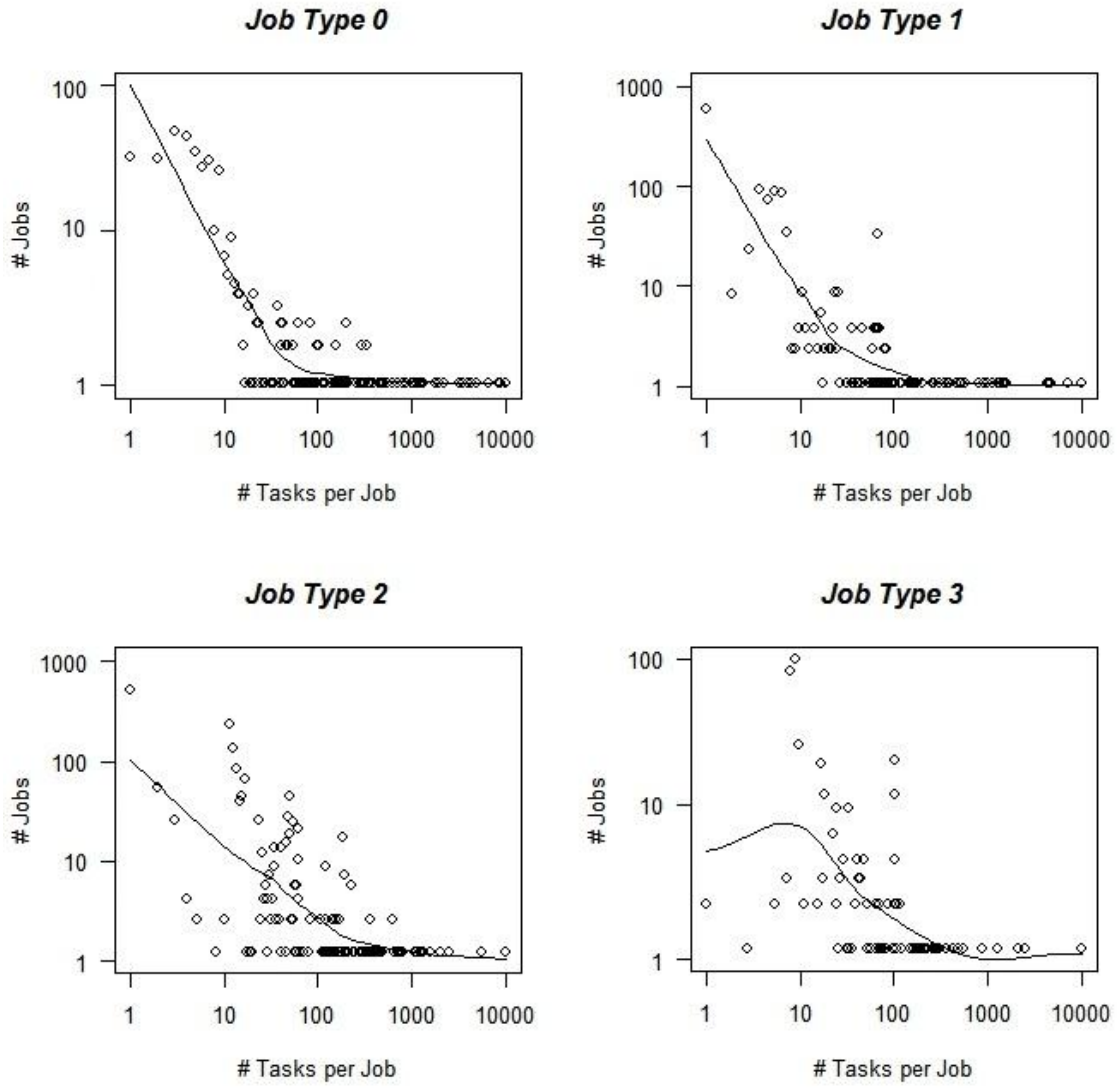

Figure 18: Google Tasks per Job by Job Type with Smoothing

All four Job Types in Figure 18 contain numerous values for tasks per job, ranging from 1 to thousands. This characteristic is important when designing the synthetic trace because it may not be necessary to simulate individual job types. Consequently, the simulation is simplified by lumping the categories together, thus eliminating the need to separate and simulate individual job type categories, similar to the results of the task

duration analysis.

Table 20: Tasks per Job Decay Rate

	Job 0	Job 1	Job 2	Job 3
Decay Rate	-0.4%	-0.5%	-1.1%	-0.3%

Running Tasks

The heuristic in chapter 3 provides a framework for running tasks characterization. Part 4 in the *Workload Analysis* section of the heuristic produces particular running tasks characteristics of the Google trace. Figure 19 shows a time series of the number of running tasks. The definition of "running" is the presence of a task in the trace [11]. Idle tasks with 0 normalized cores are present in the trace and therefore counted. Although the line plots of each of the four job types are a different shape, some display similar behaviors [11]. For example, Job Types 2 and 3 both have near constant number of running tasks. In addition, Job Types 0 and 1 both have running jobs that fluctuate around a constant average. Table 21 contains a summary of the mean and standard deviation for number of running tasks within each job type.

Table 21: Running Task Mean and Standard Deviation

	Job 0	Job 1	Job 2	Job 3
Mean	22296	14329	4902	6173
Standard Deviation	1384	1569	46	28

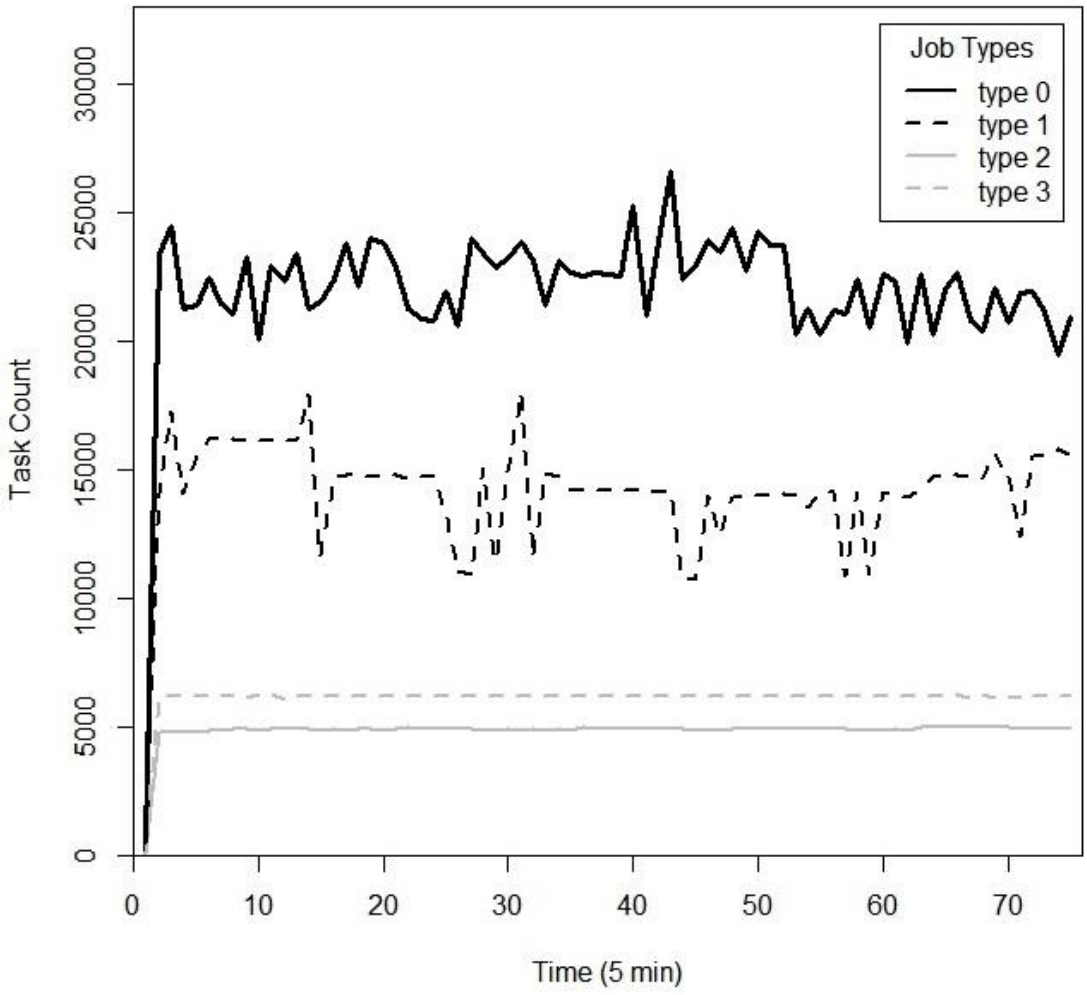

Figure 19: Google Number of Running Tasks by Job Type

Summary of Google Trace Analysis

The number of unique job launches is variable, but fluctuates around a constant average. Server memory and CPU consumption are highly correlated and fluctuate around a constant average. Task durations have a negative exponential distribution and follow the Pareto principle. Tasks per job also have a negative exponential distribution and follow the Pareto principle. Many jobs with a few tasks and short task durations, rather than a

few jobs with many tasks and long task durations, determine the overall system throughput [35]. Finally, the number of running tasks is steady or fluctuates around a constant average.

Synthetic Workload Generation Results

The synthetic workload design replicates certain characteristics of the Google trace, such as task duration and tasks per job. This section analyzes important characteristics of the synthetic trace, and compares them to the Google trace.

Job Launches

Figure 20 shows the number of unique job launches from the synthetic workload experiment. The fluctuation in job launches is due to the number of threads on each client machine as well as the variable load file. Notice the *Total Unique Job Launches* fluctuates around a constant average, similar to the behavior of the Google trace.

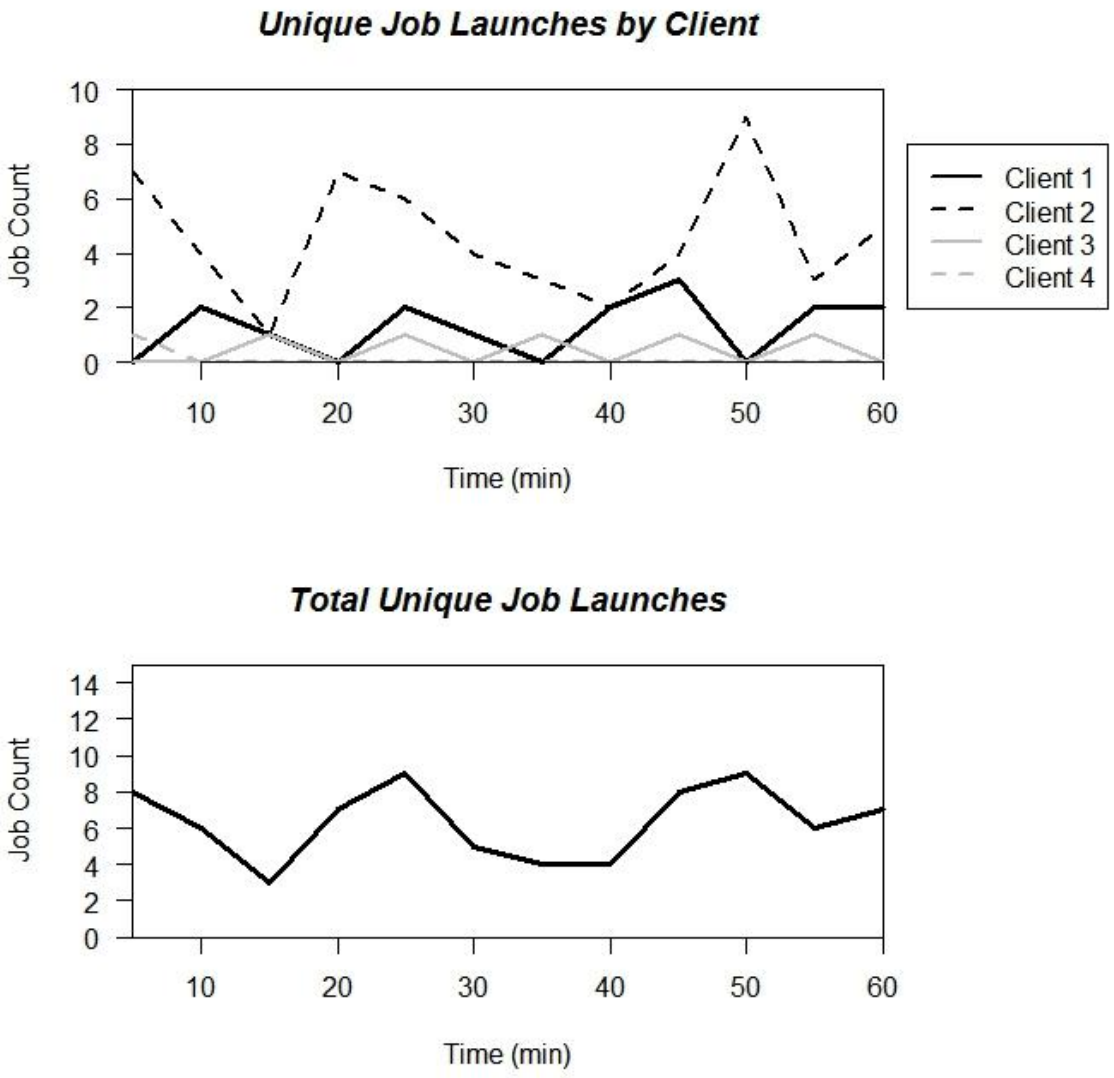

Figure 20: Faban Job Launches

Task Duration

The simulation eliminates the full duration tasks found in the Google trace, and adjusts the three remaining categories of seconds, minutes, and hours accordingly. All proportions are preserved. As a result, the simulation task durations as a percent of total jobs are as follows in Table 22.

Table 22: Synthetic Task Durations as % of Total Jobs

Task Duration	% of Total Jobs
Seconds	21.0%
Minutes	72.4%
Hours	6.6%

Figure 21 shows a bar plot of the task durations from the synthetic workload. In addition to the bar plot, a Loess smoothing curve is fitted to the data. The curve displays a negative exponential distribution, similar to that of the Google trace.

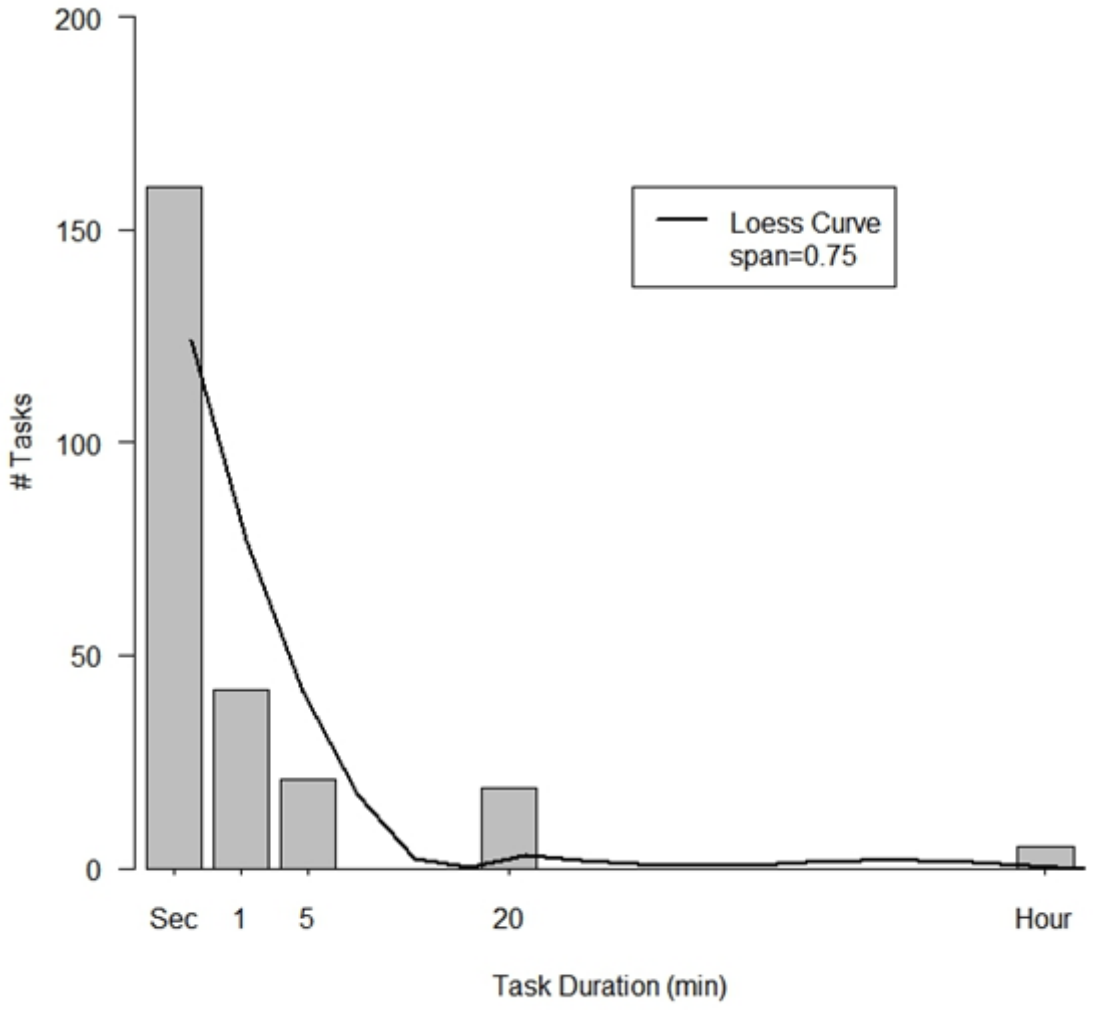

Figure 21: Faban Task Duration with Smoothing

Figure 22 shows a bar plot of the tasks per job from the synthetic workload. In addition to the bar plot, a Loess smoothing curve is fitted to the data. The curve displays properties of a negative exponential distribution, similar to that of the Google trace.

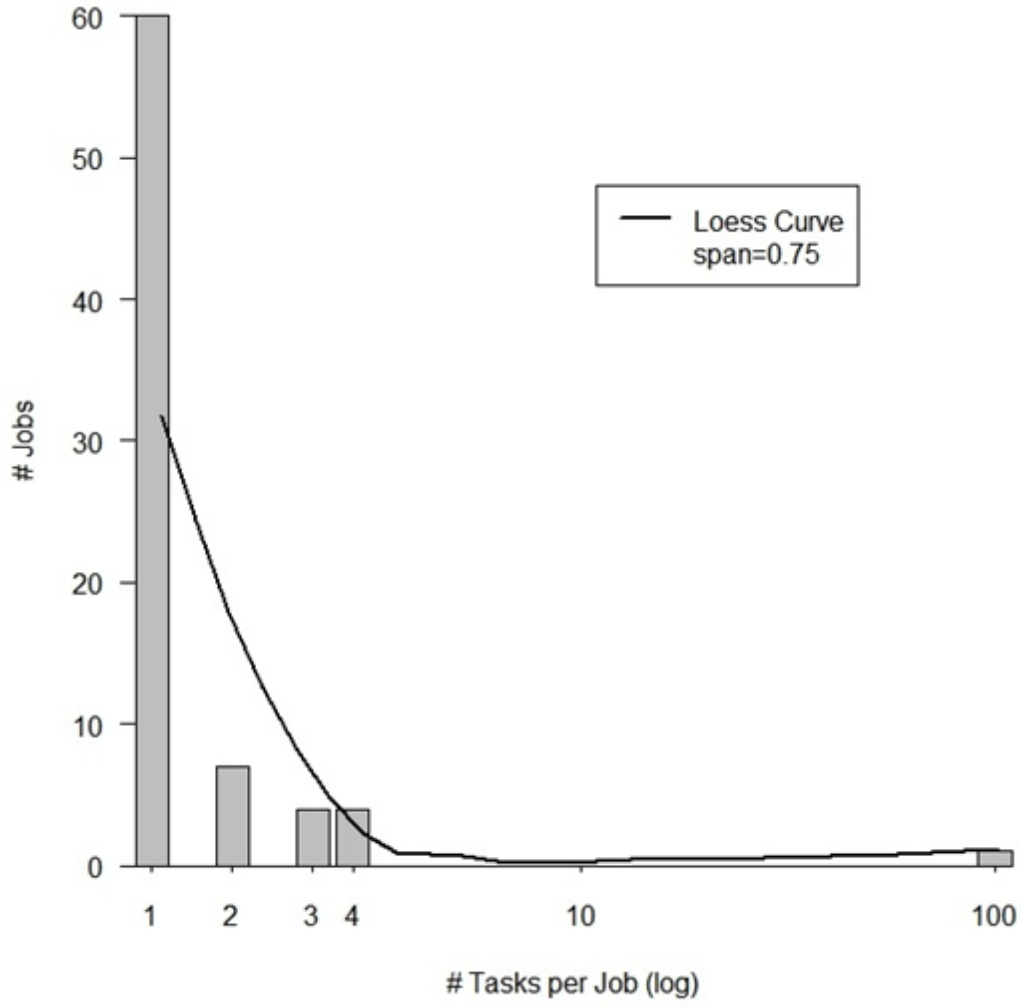

Figure 22: Faban Tasks per Job with Smoothing

The design of the synthetic workload contains a small number of tasks per job. The data in Table 23 follows the Pareto principle closely: 78.9% of all jobs have one task per job, quite close to the 80/20 ratio.

Table 23: Small # Tasks per Job

# Tasks Per Job	# Jobs	% of Total Jobs
1	60	78.9%
2	7	9.2%
3	4	5.3%
4	4	5.3%
100	1	1.3%

Synthetic Workload and Google Trace Compared

Task Length

Task duration is one of the primary characteristics simulated in the synthetic workload experiment, although the server and not the client workload determine it. Figure 23 visually compares the Cumulative Distribution Function (CDF) of both the Google and synthetic workloads. The CDFs appear to have very similar characteristics, both of which are exponentially distributed.

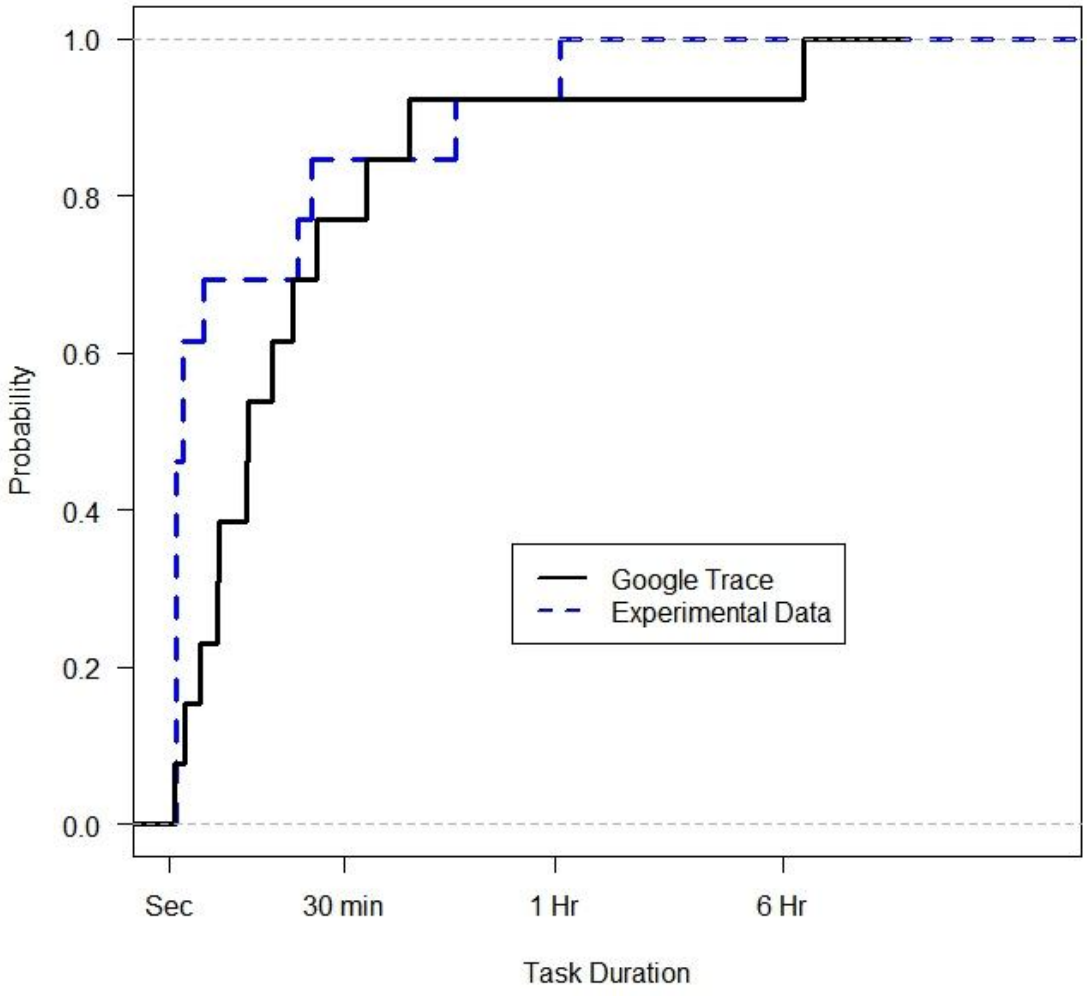

Figure 23: Cumulative Distributions of Task Durations - Google versus Synthetic

Tasks per Job

The number of tasks per job is another primary characteristics simulated in the synthetic workload experiment. Figure 24 visually compares the Cumulative Distribution Function (CDF) of the Google and synthetic workloads. The CDFs appear to have very similar characteristics, both of which are exponentially distributed.

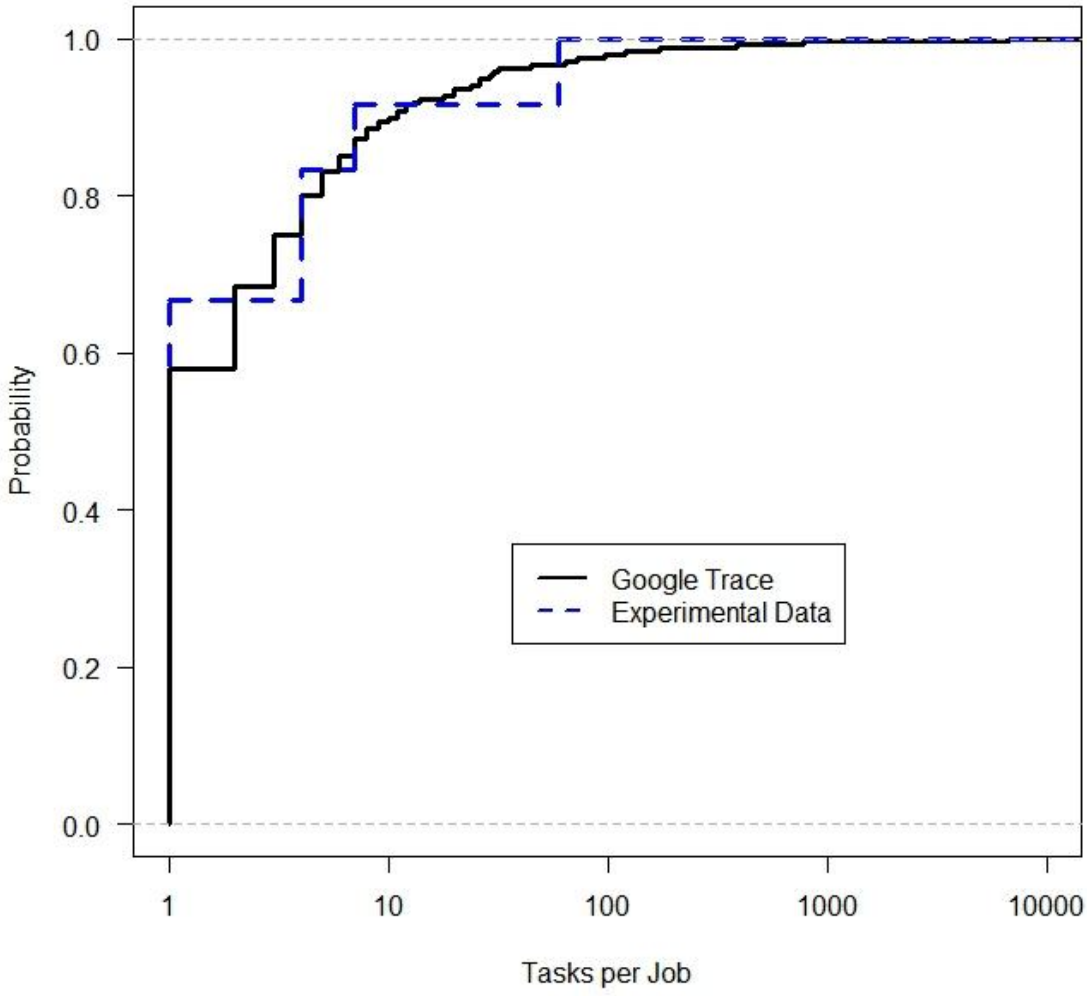

Figure 24: Cumulative Distributions of Tasks per Job - Google versus Synthetic

Correlation

Tasks per Job and Task Duration are important characteristics in the simulated trace data. Table 24 describes the statistical significance of the comparison between Google trace data and synthetic workload.

Table 24: Pearson's Product-Moment Correlation

Parameter	Correlation Coefficient (r)	Degrees of Freedom	p-value	95% Confidence Interval	r^2
Tasks per Job	0.999	2	0.0010	0.95016 to 0.99998	0.998
Task Duration as % Total Jobs (sec, min, hr)	0.916	1	0.2636	N/A	0.839
Task Duration by Task Count (see Table 25)	0.995	2	0.0053	0.7657 to 0.99990	0.990

Correlation coefficients provide an indication of strength of relationship. The Null Hypothesis (H_0): $r = 0$, states there is no 'true' relationship. The Alternative hypothesis (H_1): $r \neq 0$, states the relationship is real. A small p value points to strong evidence.

Regarding tasks per job, there is a 0.1% chance that the relationship under test is due to random sampling variability. The p-value is less than 5%, so it is statistically significant. The r^2 value is > 25%, which is a strong effect size [15]. Actually, the r^2 value is nearly equal to 1, with 1 being a perfect goodness of fit. Consequently, we reject the null hypothesis H_0.

For Task Duration as % of Total Jobs, a minimum of 2 degrees of freedom (n-2) is required for a confidence interval. The data sets are from Table 15 and Table 22. The three categories of seconds, minutes, and hours are not enough to meet this criteria. Even so, the r value indicates there is some relationship. The p value is not small enough to indicate statistically significant evidence. The task duration r^2 value is > 25%, which is a strong effect size [15]. Even though the data points to some relationship, it is not statistically significant, and we fail to reject H_0.

For Task Duration by Task Count, there is a 0.5% chance that the relationship under test is due to random sampling. The p-value is less than 5%, so it is statistically significant. The r^2 value is > 25%, which is a strong effect size [15]. Actually, the r^2 value is nearly equal to 1, with 1 being a perfect goodness of fit. Consequently, we reject the null hypothesis H_0.

Table 25: Task Duration Counts

Trace	1sec to < 5min	5 min	20 min	Hour
Google	51,182	9918	6381	181
Synthetic	202	21	19	5

Covariance

Table 26 shows the covariance values for task duration and tasks per job. Both values, task duration and tasks per job, indicate a positive linear relationship between the two data sets. In summary, the distributions have similar behaviors and change together.

Table 26: Covariance

Parameter	Value
Task Duration	732.0
Tasks per Job	1259.9

Summary of Synthetic Workload Generation Results

The number of unique job launches is variable, but fluctuates around a constant average. Task durations and tasks per job are both negative exponentially distributed. The tasks per job parameter has an 80/20 characteristic, where nearly 80% of jobs contain a single task. In addition, cumulative distribution function plots visually compare the Google trace to the synthetic workload. The distributions show a strong relationship.

Correlation and covariance values show statistical significance, or lack thereof, when comparing similarities of task duration and tasks per job.

Results of Scalability Test

This experiment scales down the size of the Faban synthetic workload. The scalability test determines if the workload generator can produce a load equivalent to that of the largest surges in the Google trace.

Max Tasks per Job

The max tasks per job value from the Google trace is 4880, as seen in Table 18. The test verifies if Faban is capable of creating a load with 1 thread and 4880 jobs. The experiment runs for 300 sec with a 60 ms think time. The result of the test is a throughput of 4983 tasks. Consequently, Faban is capable of simulating a single user requesting thousands of tasks.

Mean Tasks per Job

The mean tasks per job value from the Google trace is 19, as seen in Table 18. Faban is capable of generating such a load for a sustained amount of time. However, a requirement may exist for generation of a larger load, or surge. The Google trace does contain a surge with a mean tasks per job value of 521. The mean number of threads in the trace is 120. The test verifies if Faban is capable of creating a load with 120 threads and 521 tasks per thread, totaling 62,520 tasks. The experiment runs for 300 sec with a 575 ms think time. The throughput is 62,447 tasks. Consequently, Faban is capable of simulating hundreds of users each requesting hundreds of tasks.

Summary of Scalability Test

The scalability test shows Faban can produce large synthetic workloads with load sizes comparable to publically available cloud traces. Faban's distributed design makes it very well suited for generating large loads, perhaps simulating thousands of users each requesting thousands of tasks. The addition of new client workload machines increases output capacity. The limiting factor is the JVM and underlying hardware, as described in Chapter 3.

VI. Conclusions and Recommendations

Chapter Overview

This thesis researches cloud trace characterization and synthetic workload generation. Very few publically available cloud traces exist, only two published by Google at the time of this writing. Numerous workload generation tools are available, and Faban is the tool of choice for this study. Synthetic traces can contain justifiably real characteristics, as shown in Chapter 4. The information contained in this thesis will assist future researchers who require cloud workloads without using full-blown data sets.

Conclusions of Research

Publically available trace files bound research of this nature. More specifically, the data contained in the Google trace limits the synthetic workload results of this research. Even so, this is a vast improvement over predefined workload with no statistical justifications. The information in this thesis provides researchers with a lightweight heuristic for generating synthetic workloads using an open source load generator. In addition, this research provides Google cloud workload characteristics and methodologies that justify statistical similarities.

Traces

Numerous publically available trace archives exist, and many contain workload traces from computer system technologies such as grids and clusters. Clouds and grids have much commonality, with the main differences being: clouds use virtual machines while grids use threads, job and task durations are shorter in clouds than grids, and clouds have a higher job arrival rate than grids. Characterizing the qualities of grid trace files

within the research community is not new. Publications characterizing the Google trace and the Google trace itself both appeared at nearly the same time, likely because Google employees are involved with the publications. Even so, the Google cloud trace is the first known publically available cloud trace, and simulating its characteristics is new to academia and the research community.

The Google workload trace is anonymous and thus has limitations, but still contain much useful information. One must become familiarized and understand the data prior to characterizing and modeling it. Some of the more important characteristics for simulation purposes are job types, job launches and request rates, running tasks, tasks per job, and task durations. The goal is generating a synthetic workload with statistically significant similarities to the Google trace.

Workload Generation

Faban is a free web-benchmarking tool that is well suited for cloud generation. It is scalable to meet large workload demands found in clouds. There is a learning curve to the software tool, so the spiral development process or similar procedure is supportive when generating synthetic workloads. Developers can create different workload distributions, such as exponential or normal, by modifying Faban configuration files. Think time and variable load parameters are particularly important for shaping the distributions.

Real vs. Synthetic Trace

The number of job launches in the Google trace is variable and fluctuates around a constant average. Server memory and CPU consumption are highly correlated, and values fluctuate around a constant average. Task durations and task per job both have

negative exponential distributions and follow the Pareto principle. Jobs with a few tasks and short task durations, rather than a few jobs with many tasks and long task durations, determine the overall system throughput [35]. Finally, the number of running tasks is steady or fluctuates around a constant average.

The number of unique job launches in the synthetic trace is variable, but fluctuates around a constant average, similar to the Google trace. Task durations and tasks per job are both negative exponentially distributed. The tasks per job parameter has an 80/20 characteristic, where nearly 80% of jobs contain a single task. In addition, cumulative distribution function plots visually compare the Google trace to the synthetic workload. The distributions show a strong relationship. Correlation and covariance values prove statistical significance when comparing characteristics of task duration and tasks per job. The overall result of the synthetic workload generation is a strong positive relationship exists between the Google trace and the synthetic workload.

Significance of Research

This research effort proves the heuristic from chapter 3 successful. By gathering trace data, analyzing and understand the data, characterizing and modeling it, and finally generating a synthetic workload, researchers have the foundation needed for justifying realistic characteristics and proving statistical significance. This research effort provides a stepping-stone for engineers and researchers who require a cloud workload. Autonomous cloud management techniques and virtual machine optimization are significant and relevant research topics that necessitate realistic workloads. The heuristic

in this thesis provides vital information for creating justifiably realistic synthetic cloud workloads.

Recommendations for Future Research

Algorithm

Develop an algorithm from the heuristic presented in chapter 3 to formalize the lightweight synthetic workload generation process. Formalizing the heuristic can optimize cloud workload generation, and ultimately further cloud research.

Supplementary Synthetic Data

Generating more data at a larger scale can lead to an improved statistical analysis. In particular, more variety in task duration and tasks per job can lead to a better distribution fit. An ideal fit to the Google trace has a long tail, which requires more data points in the tail, for both task durations and tasks per job. Task durations are more a bit more complex because the durations rely on the time it takes the server to complete the task, hence the need to design additional server-side tasks.

Additional Cloud Traces

As new cloud traces become available to the public, researchers must analyze and characterize the data. This leads to improvements in quality of synthetic workloads, elasticity and optimization algorithms that manage cloud servers, and ultimately the services provided by cloud vendors. Owners of cloud traces should follow Google's initiative and publish their data for the good of the cloud computing community.

Cloud Workload Generation Tools

Although Faban is well suited for cloud synthetic workload generation, there may be superior tools available. One software package that sounds particularly interesting is VMmark by VMware, a virtualization platform benchmarking tool. Its primary purpose measures datacenter performance, but also includes built-in load generation tools. At a minimum it seems worthy of exploring, but does require a very high level of prerequisite knowledge in the field of virtual machine administration. In addition, VMmark is a rather large system that encompasses numerous smaller systems. Consequently, researchers must consider the extensive hardware and software requirements of the system.

Summary

Overall, this work analyzes real traces and simulates those characteristics in a synthetic workload. Statistical analysis proves the relationships and similarities. Specifically, the synthetic task duration and tasks per job distributions mimic that of the Google trace. The ability to create justifiably realistic workloads furthers cloud research and is not in current literature.

Bibliography

[1] A. Ali-Eldin, M. Kihl, J. Tordsson and E. Elmroth. Efficient provisioning of bursty scientific workloads on the cloud using adaptive elasticity control. Presented at Proceedings of the 3rd Workshop on Scientific Cloud Computing Date. 2012, .

[2] A. Andrzejak, D. Kondo and S. Yi. Decision model for cloud computing under sla constraints. Presented at Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2010 IEEE International Symposium on. 2010, .

[3] A. Bahga and V. K. Madisetti. Synthetic workload generation for cloud computing applications. *Journal of Software Engineering and Applications 4(7),* pp. 396-410. 2011.

[4] A. Beitch and D. A. Patterson. Rain: A workload generation toolkit for cloud computing applications. *Rain* 2010.

[5] T. Benson, A. Akella and D. A. Maltz. Network traffic characteristics of data centers in the wild. Presented at Proceedings of the 10th Annual Conference on Internet Measurement. 2010, .

[6] K. P. Birman, *Guide to Reliable Distributed Systems: Building High-Assurance Applications and Cloud-Hosted Services.* Springer, 2012.

[7] R. Buyya, C. S. Yeo, S. Venugopal, J. Broberg and I. Brandic. Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Comput. Syst. 25(6),* pp. 599-616. 2009.

[8] E. Caron, F. Desprez and A. Muresan. Forecasting for grid and cloud computing on-demand resources based on pattern matching. Presented at Cloud Computing Technology and Science (CloudCom), 2010 IEEE Second International Conference on. 2010, .

[9] D. Chappell. Introducing the windows azure platform. *David Chappell & Associates White Paper* 2010.

[10] Y. Chen, A. S. Ganapathi, R. Griffith and R. H. Katz. Towards understanding cloud performance tradeoffs using statistical workload analysis and replay. *University of California at Berkeley, Technical Report no.UCB/EECS-2010-81* 2010.

[11] Y. Chen, A. S. Ganapathi, R. Griffith and R. H. Katz. Analysis and lessons from a publicly available google cluster trace. *University of California, Berkeley, CA, Tech.Rep* 2010.

[12] W. S. Cleveland. *Visualizing Data* 1993.

[13] W. S. Cleveland and S. J. Devlin. Locally weighted regression: An approach to regression analysis by local fitting. *Journal of the American Statistical Association 83(403),* pp. 596-610. 1988.

[14] Cloudstone. Available: http://radlab.cs.berkeley.edu/wiki/Projects/Cloudstone.

[15] J. Cohen. A power primer. *Psychol. Bull. 112(1),* pp. 155. 1992.

[16] M. D. De Assunção, A. Di Costanzo and R. Buyya. Evaluating the cost-benefit of using cloud computing to extend the capacity of clusters. Presented at Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing. 2009.

[17] S. Di, D. Kondo and W. Cirne. Characterization and comparison of google cloud load versus grids. 2012.

[18] Faban. Available: http://faban.org/.

[19] Faban Harness and Benchmark Framework. Available: http://java.net/projects/faban/.

[20] Failure Trace Archive. Available: http://fta.inria.fr.

[21] D. Feitelson. Workload modeling for computer systems performance evaluation. *Book Draft, Version 0.36* 2012.

[22] D. Feitelson. Workload modeling for performance evaluation. *Performance Evaluation of Complex Systems: Techniques and Tools* pp. 114-141. 2002.

[23] I. Foster, Y. Zhao, I. Raicu and S. Lu. Cloud computing and grid computing 360-degree compared. Presented at Grid Computing Environments Workshop, 2008. GCE'08. 2008, .

[24] J. Fox. *An R and S-Plus Companion to Applied Regression* 2002.

[25] A. Ganapathi, Y. Chen, A. Fox, R. Katz and D. Patterson. Statistics-driven workload modeling for the cloud. Presented at Data Engineering Workshops (ICDEW), 2010 IEEE 26th International Conference on. 2010, .

[26] S. Genaud and J. Gossa. Cost-wait trade-offs in client-side resource provisioning with elastic clouds. Presented at Cloud Computing (CLOUD), 2011 IEEE International Conference on. 2011, .

[27] Grid Workloads Archive. Available: http://gwa.ewi.tudelft.nl.

[28] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer and D. H. J. Epema. Performance analysis of cloud computing services for many-tasks scientific computing. *Parallel and Distributed Systems, IEEE Transactions on 22(6),* pp. 931-945. 2011.

[29] Journal of Theoretical and Applied Information Technology. Available: http://www.jatit.org/.

[30] J. O. Kephart and D. M. Chess. The vision of autonomic computing. *Computer 36(1),* pp. 41-50. 2003.

[31] C. Kleineweber, A. Keller, O. Niehorster and A. Brinkmann. Rule-based mapping of virtual machines in clouds. Presented at Parallel, Distributed and Network-Based Processing (PDP), 2011 19th Euromicro International Conference on. 2011, .

[32] D. Kluscek and H. Rudov. The importance of complete data sets for job scheduling simulations. Presented at Job Scheduling Strategies for Parallel Processing. 2010, .

[33] D. Kondo, B. Javadi, A. Iosup and D. Epema. The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems. Presented at Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on. 2010, .

[34] Y. Liu, N. Bobroff, L. Fong and S. Seelam. Workload management in cloud computing using meta-schedulers. *IBM Research* 2009.

[35] Z. Liu and S. Cho. Characterizing machines and workloads on a google cluster. Presented at Parallel Processing Workshops (ICPPW), 2012 41st International Conference on. 2012, .

[36] V. Makhija, B. Herndon, P. Smith, L. Roderick, E. Zamost and J. Anderson. VMmark: A scalable benchmark for virtualized systems. *VMware Inc, CA, Tech.Rep.VMware-TR-2006-002, September* 2006.

[37] E. P. Mancini, M. Rak and U. Villano. Perfcloud: Grid services for performance-oriented development of cloud computing applications. Presented at Enabling Technologies: Infrastructures for Collaborative Enterprises, 2009. WETICE'09. 18th IEEE International Workshops on. 2009, .

[38] N. Matloff. *The Art of R Programming* 2011.

[39] M. Mattess, C. Vecchiola and R. Buyya. Managing peak loads by leasing cloud infrastructure services from a spot market. Presented at High Performance Computing and Communications (HPCC), 2010 12th IEEE International Conference on. 2010, .

[40] J. M. McCune, D. A. Fisher and A. D. Andrews. Trust and trusted computing platforms. 2011.

[41] P. Mell and T. Grance. The NIST definition of cloud computing. *National Institute of Standards and Technology 53(6),* pp. 50. 2009.

[42] MetaCentrum Data Sets. Available: http://www.fi.muni.cz/~xklusac/workload;.

[43] J. S. Milton and J. C. Arnold. *Introduction to Probability and Statistics: Principles and Applications for Engineering and the Computing Sciences* 2002.

[44] Olio Web 2.0. Available: http://incubator.apache.org/olio/index.html.

[45] Parallel Workloads Archive. Available: http://www.cs.huji.ac.il/labs/parallel/workload/.

[46] Rain Workload Toolkit. Available: https://github.com/yungsters/rain-workload-toolkit/wiki.

[47] F. L. Ramsey and D. W. Schafer. *The Statistical Sleuth: A Course in Methods of Data Analysis* 2012.

[48] C. Reiss, R. H. Katz and M. A. Kozuch. Towards understanding heterogeneous clouds at scale: Google trace analysis. 2012.

[49] C. Reiss, J. Wikes and J. Hellerstein, "Google cluster-usage traces: format + schema," .

[50] V. Ricci. Fitting distributions with r. *Contributed Documentation Available on CRAN* 2005.

[51] N. Sematech. *Engineering Statistics Handbook* 2006.

[52] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat and C. R. Das. Modeling and synthesizing task placement constraints in google compute clusters. Presented at Proceedings of the 2nd ACM Symposium on Cloud Computing. 2011, .

[53] B. Speitkamp and M. Bichler. A mathematical programming approach for server consolidation problems in virtualized data centers. *Services Computing, IEEE Transactions on 3(4),* pp. 266-278. 2010.

[54] P. Teetor. *R Cookbook* 2011.

[55] TestnScale. Available: http://www.testnscale.com/docs/;.

[56] A. Totok. Exploiting service usage information for optimizing server resource management. 2006.

[57] VMWare VMmark. Available: http://www.vmware.com/products/vmmark/requirements.html.

[58] G. Wang, A. R. Butt, H. Monti and K. Gupta. Towards synthesizing realistic workload traces for studying the hadoop ecosystem. Presented at Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), 2011 IEEE 19th International Symposium on. 2011, .

[59] J. Wilkes and C. Reiss. Google cluster data. Available: http://code.google.com/p/googleclusterdata/wiki/ClusterData2011_1.

[60] Q. Zhang, J. L. Hellerstein and R. Boutaba. Characterizing task usage shapes in Google's compute clusters. *Proc.of Large-Scale Distributed Systems and Middleware (LADIS 2011)* 2011.

[61] Y. Zhang, Z. Wang, B. Gao, C. Guo, W. Sun and X. Li. An effective heuristic for on-line tenant placement problem in SaaS. Presented at Web Services (ICWS), 2010 IEEE International Conference on. 2010, .

[62] X. Zhu, D. Young, B. Watson, Z. Wang, J. Rolia, S. Singhal, B. McKee, C. Hyser, D. Gmach, R. Gardner, T. Christian and L. Cherkasova. 1000 islands: An integrated approach to resource management for virtualized data centers. *Cluster Computing 12(1)*, pp. 45-57. 2009.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

PLEASE DO NOT RETURN YOUR FORM TO THE ABOVE ADDRESS.

1. REPORT DATE *(DD-MM-YYYY)*	2. REPORT TYPE	3. DATES COVERED *(From – To)*
21-03-2013	Master's Thesis	Sep 2011 - Mar 2013

4. TITLE AND SUBTITLE
Cloud Computing Trace Characterization and Synthetic Workload Generation

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Capra, Salvatore, Civilian, USAF

5d. PROJECT NUMBER
N/A

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENG-13-M-11

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Intentionally Left Blank

10. SPONSOR/MONITOR'S ACRONYM(S)

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRIBUTION STATEMENT A: APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

This thesis researches cloud computing workload characteristics and synthetic workload generation. A heuristic presented in the work guides the process of workload trace characterization and synthetic workload generation. Analysis of a cloud trace provides insight into client request behaviors and statistical parameters. A versatile workload generation tool creates client connections, controls request rates, defines number of jobs, produces tasks within each job, and manages task durations. The test system consists of multiple clients creating workloads and a server receiving request, all contained within a virtual machine environment. Statistical analysis verifies the synthetic workload experimental results are consistent with real workload behaviors and characteristics.

15. SUBJECT TERMS
Cloud Computing, Trace File, Synthetic Workload Generation, Workload Characterization

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT	b. ABSTRACT	c. THIS PAGE			Dr. Kenneth Hopkinson (ENG)
U	U	U	UU	97	19b. TELEPHONE NUMBER *(Include area code)* (937) 255-6565, x 4579 kenneth.hopkinson@afit.edu