Implications of Overcommitment on Cloud Computing Workloads

by Ravi M. Shanbhag

B.E. in Computer Science & Engineering, June 2003, Manipal Academy of Higher Education
Master of Business Administration, December 2008, Texas A&M University
Master of Applied Statistics, May 2015, Pennsylvania State University

A Praxis submitted to

The Faculty of
The School of Engineering and Applied Science
of The George Washington University
in partial fulfillment of the requirements
for the degree of Doctor of Engineering

Aug 31, 2021

Praxis directed by

Shahryar Sarkani
Adjunct Professor of Engineering Management and Systems Engineering

John Fossaseca
Professional Lecturer of Engineering Management and Systems Engineering

The School of Engineering and Applied Science of The George Washington University certifies that Ravi M. Shanbhag has passed the Final Examination for the degree of Doctor of Engineering as of Aug 31, 2021. This is the final and approved form of the Praxis.

Implications of Overcommitment on Cloud Computing Workloads

Ravi M. Shanbhag

Praxis Research Committee:

Shahryar Sarkani, Adjunct Professor of Engineering Management and Systems Engineering, Praxis Co-Director

John Fossaseca, Professional Lecturer of Engineering Management and Systems Engineering, Praxis Co-director

J.P. Blackford, Professional Lecturer of Engineering Management and Systems Engineering, Committee Member

© Copyright 2021 by Ravi M. Shanbhag
All rights reserved

Dedication

Dedicated to my family who supported me while I spent many nights and weekends engrossed in this research. This would not have been possible without their patience, grace and understanding.

Acknowledgements

Undertaking a multi-year journey towards a Doctorate is easier said than done. Meaningful research is hard and making it practical is even harder. However, finishing this work has been a rewarding experience.

I want to offer my sincere appreciation to my advisors, Dr. Shahryar Sarkani, Dr. John Fossaseca for their support and encouragement. They guided me through navigating the hurdles and avoiding the dead-ends. They asked the tough and, in many ways, germane questions that helped me think through the problem holistically. This Praxis is in a superior shape because of their counsel and mentorship.

I felt lucky to be a part of an exceptionally talented cohort of Doctoral students who I could learn from on a regular basis. We faced the trials and tribulations together in our respective areas of research. Their friendships and a desire to enjoy each other's successes have left an indelible mark on my personal and professional journey. I thank them for bearing with me and for being a constant source of encouragement and support.

You seldom realize the kind of support you will need when you undertake a journey to do some of your best research work. My self-motivation and commitment were constantly bolstered by the resilience and encouragement from my family despite a raging pandemic around us all. It kept me going at a frantic pace for many years and sustained me in ways I had not realized I needed.

Thank You!

Abstract of Praxis

Implications of Overcommitment on Cloud Computing Workloads

In an overcommitted datacenter, the sum of resource requests for all tasks running in the datacenter is generally greater than the available capacity of that datacenter. Pushing overcommitment too far can cause increased task failures – so a balance is needed between the overcommitment rate and the task failure (task eviction) rate. The conditions under which overcommitment can be pushed higher to improve utilization without worsening the task failure rate can be understood better.

This Praxis creates explainable predictive models to effectively understand the impact of job arrivals, evictions and overcommitment rates on resource utilization using Generative Discreate Event Simulations. It further explores the implications of machine heterogeneity on resource (CPU/Memory) utilization in an overcommitted environment. The effect of interactions between job arrival rates and overcommitment rates in predicting eviction rates are also studied.

Results indicate that an overcommitted heterogenous cluster improves Memory and CPU utilization compared to a similar homogenous cluster when job arrival rates and eviction rates are held constant. We learn that CPU and Memory utilization can be predicted using job arrival rates, overcommitment rates and eviction rates. We also find that, at constant scheduling rates, eviction rate prediction can be improved by incorporating an interaction between job arrival rates and overcommitment rates. Finally, this research shows that a generative discrete event simulation approach can be used to model an overcommitted datacenter workload.

Table of Contents

Dedication ..... **iv**

Acknowledgements ..... **v**

Abstract of Praxis ..... **vi**

List of Figures ..... **x**

List of Symbols ..... **xiii**

List of Acronyms ..... **xiv**

Chapter 1. Introduction 1

1.1. Background 1

1.2. Research Motivation 2

1.3. Problem Statement 5

1.4. Thesis Statement 5

1.5. Research Objectives 5

1.6. Research Questions and Hypotheses 6

1.7. Scope and Research Limitations 7

1.8. Organization of Praxis 8

Chapter 2—Literature Review 10

2.1. Overcommitment in Cloud Computing 10

2.2. Multi-objective Scheduling Policies 11

2.2.1. Job Arrivals and Transitions 11

2.2.2. Resource Allocation and Scheduling (RAS) 13

2.3. Simulations for Modeling Cloud Workloads 15

2.3.1. Real-world Systems and Types 17
2.3.2. Discrete Event Simulations for Modeling Cloud Workloads 18
2.4. Summary 19
Chapter 3—Methodology 21
3.1. Introduction 21
3.2. Generative Modeling Flow Diagram 25
3.3. Discrete Event Simulation Modeling Software 27
3.4. Simulation Class Diagram 29
3.5. Simulation Parameter: Job States and Transitions 31
3.6. Simulation Parameter: Job Arrival Rates 33
3.7. Simulation Parameter: Overcommitment Rate 36
3.8. Simulation Parameter: Machine Heterogeneity and Distribution 39
3.9. Simulation Parameter: Bin Packing Algorithm 40
3.10. Other important scheduling policy considerations 48
Chapter 4—Results 50
4.1. Descriptive Characteristics of the Simulations 50
4.2. Evaluation Models for Memory and CPU Utilization 55
4.3. Evaluation Model for Job Eviction Rates 63
4.4. Summary 66
Chapter 5—Conclusions and Future Work 67
5.1 Conclusions 67
5.2 Research Contributions 68
5.3.1 Recommendations for Future Research 69

References 70

Appendix A – Trace Data Source 85

Appendix B – Simulated Data Sources 86

List of Figures

Figure 2-1: State transition diagram for jobs and tasks. Users can trigger submit, kill, and update transitions. (Source: (Verma, et al., 2015)) 12

Figure 2-2: A taxonomy of RAS algorithms and policies. (Source: (Ma, Chu, Zhao, & Ankhbayar, 2014)) 14

Figure 2-3: A Discrete System Example 17

Figure 2-4: A Continuous System Example 18

Figure 3-1: Descriptive Workload Modeling based on statistical analysis of the trace. (Feitelson, 2015) 22

Figure 3-2: Generative Workload Modeling based on analyzing user behavior in a production system. (Feitelson, 2015) 23

Figure 3-3: Generative Workload Modeling and subsequent Hypotheses Evaluation Flowchart 25

Figure 3-4: Mapping of original times to emitted times in the trace. (Wilkes, 2020) 26

Figure 3-5: Simplified Class Diagram for the Simulation 29

Figure 3-6: Job State Transitions in the Simulation 31

Figure 3-7: Probability Density Function of Exponential Distributions with different λ values. 34

Figure 3-8: Cumulative Distribution Function of Exponential Distributions with different λ values. 35

Figure 3-9: Job Resource Utilization over its run-time duration with an Overcommitment rate 36
Figure 3-10: Distribution of Job Arrival Rates and Overcommitment Rates in the Simulation 39
Figure 3-11: Distribution of Cumulative CPU and Memory Capacity across different simulations and Machine Heterogeneities 40
Figure 3-12: Initial System State example of Job Requests, Available Machines, and their respective resources capacities. 43
Figure 3-13: System State after the first job (Job 1) is scheduled 44
Figure 3-14: System state after the first 2 jobs (Job 1 and Job 2) are scheduled..... 45
Figure 3-15: System state after the first 5 jobs (Job 1 - Job 5) are scheduled 46
Figure 3-16: System state after all the jobs are scheduled..... 47
Figure 4- 1: Job creation histogram for each machine distribution type 50
Figure 4- 2: Job arrival rate and overcommitment rate by number of jobs created for each machine distribution type..... 51
Figure 4- 3: Density of % Jobs Completed and % Jobs Evicted for each machine distribution type 52
Figure 4- 4: Density of Median Memory vs. Median CPU Usage across simulations for each machine distribution type 53
Figure 4- 5: Histograms of the Coefficient of Variation (CV) for Memory and CPU Usage across simulations for each machine distribution type 54
Figure 4- 6: Effect Summary for the Median Memory Utilization Model 56

Figure 4- 7: Summary of Fit, Analysis of Variance and Parameter Estimates and Effect Tests for the Median Memory Utilization Model 58
Figure 4- 8: Prediction Expression for the Median Memory Utilization Model 59
Figure 4- 9: Effect Summary for the Median CPU Utilization Model 60
Figure 4- 10: Summary of Fit, Analysis of Variance and Parameter Estimates and Effect Tests for the Median CPU Utilization Model 61
Figure 4- 11: Prediction Expression for the Median CPU Utilization Model 62
Figure 4- 12: Effect Summary for the Eviction Rate Model with Job Arrival Rate and Overcommitment Rate Interaction 63
Figure 4- 13: Summary of Fit, Analysis of Variance, Parameter Estimates, Effect Tests and Cross validation results for the Eviction Rate Model 64
Figure 4- 14: Prediction Expression for the Eviction Rate Model 65
Figure 4- 15: Contour Profile for the Eviction Rate Model 65

List of Symbols

λ	Rate parameter of the exponential distribution

List of Acronyms

BPP	Bin Packing Problem
CPU	Central Processing Unit
CSP	Cloud Service Provider
CV	Coefficient of Variation
DAG	Directed Acyclic Graph
OOM	Out of Memory (Error)
QoS	Quality of Service
RAM	Random-Access Memory
RAS	Resource Allocation and Scheduling
SLO	Service Level Objective

Chapter 1. Introduction

1.1. Background

Cloud computing has gained broad enterprise adoption over the past decade and deploying workloads to cloud resources instead of captive datacenters is proving cost-effective for customers as well as Cloud Service Providers (CSPs) (Konstantinos, Persefoni, Evangelia, Christos, & Mara, 2015). Several CSPs like Google, Microsoft, and Amazon have made millions of dollars of investments in large scale globally distributed Cloud Computing Centers. As cloud computing has evolved, so have cloud architectures where customers deploy their workloads to multiple providers (multi-cloud) simultaneously to diversify from a single cloud provider (Varghese & Buyya, 2018). This creates fierce competition to keep prices low while offering high availability infrastructure services that are also sustainable in the long run.

Cloud computing offers huge benefits by harnessing the scalability and rapid provisioning capabilities of these CSPs. However, devising efficient workload management schemes that fully leverage the large capital investments and operational costs is still a major challenge (Zhang & Boutaba, 2014). Also, most computing environments are heterogenous where the hardware has different configurations (CPU, Memory, Disk etc.). These large datacenters also support mixed workloads with varied run-times and resource usages in the same datacenter (Lee & Katz, 2011). This complexity of matching the tasks optimally to the machines while minimizing machine idle-time is a complex problem resulting in significant underutilization of existing assets (Liu H. , 2011).

1.2. Research Motivation

Modern cloud datacenters are huge capital investments for Cloud Service Providers (CSPs) like Google, Microsoft, Amazon, and others. As cloud adoption among enterprises becomes the norm, the pressures on CSPs to provide scalable and reliable service guarantees become enormous. Customer satisfaction and retention is at stake. Furthermore, penalties for Service Level violations can hurt profitability. Offsetting these large capital investments by improving the utilization of these captive assets has long been a concern of CSPs.

CSPs can recoup millions of dollars from their investments by increasing utilization by a few percentage points (Verma, et al., 2015). Let us consider the example of Borg - a cluster management system developed by Google to manage batch and real-time jobs on large scale computing clusters (Burns, Grant, Oppenheimer, Brewer, & Wilkes, 2016). In 2011, Google released an extensive trace for one of its datacenters managed by Borg that logged activity for the month of May 2011 (Verma, et al., 2015). In that release, Google shared architecture details of Borg and presented in-depth usage pattern analyses using the trace data. The analyses revealed that the maximum memory utilization was less than 50% and maximum CPU utilization was as low as 60% (Reiss, Tumanov, Ganger, Katz, & Kozuch, 2012). The average utilizations over the trace period were much lower – memory was utilized at 53% and CPU was utilized at 40% (Tomás & Tordsson, 2013). More recent traces reveal that utilization has improved substantially but still lingers between 50% to 60% for both CPU and Memory across various cells with disparate workloads (Tirmazi, et al., 2020).

One reason for this is that users tend to request more resources than needed with an expectation of guaranteed execution and reliable completion of their workloads. Studies show that virtual machines have typically utilized between 35% and 55% of requested resources (Dabbagh, Hamdaoui, Guizani, & Rayes, 2015) resulting in CPU and memory to be reserved beyond the actual requirements, hence causing wastage.

Overcommitment (of CPU and Memory) has been deployed as a potential solution for improving utilization and reducing wastage (Ji, Da Li, Ji, & Li, 2018). When tasks arrive at the scheduler with a particular resource request – the scheduler anticipates that, during runtime, the task will utilize fewer resources than its original ask. For instance, let us assume we have a machine (Machine A) with 8 CPUs and 64 GB of Memory. If a task requests 2 CPUs and 16 GB of Memory the scheduler can allocate 1 CPU and 8 GB of Memory to such a task and place that on Machine A. This frees up the scheduler to place 8 more tasks with similar requests on the same machine. Thus, the overcommitment ratio will be 2 indicating that the total resource requests (CPU and Memory) of tasks on the machine was 2 times the capacity of that machine (i.e., 16 CPUs and 128 GB are requested against the 8 CPUs and 64 GB available on Machine A). This allows Machine A to execute more tasks than it would have otherwise if resources were allocated per the original ask. In other words, the machine is "overcommitted" if the tasks running on that machine end up utilizing what they originally requested.

A high overcommitment ratio can incase resource utilization but it does not come without its pitfalls. In the example above, if half the tasks utilize resources per their original request (vs. what was allocated), while the rest maximize their allocation, then collectively the 8 tasks will end up consuming all the available capacity of the machine.

CPUs can be throttled to a certain extent, but increased utilization of memory will cause Out of Memory (OOM) errors (Verma, et al., 2015). So, the scheduler must kill the task that is consuming more memory than its request (or allocation if the machine is out of capacity) and then restart it on a different machine. The task is said to be evicted and must be spawned again. All the work that was done prior to eviction must be redone.

When overcommitment occurs, the scheduler has a choice between the tasks it can evict. In the example above, if any one of the tasks consumes 1.5 CPUs then Machine A is overcommitted. However, 1.5 CPUs is still within the limits of the original ask by the task (2 CPUs). So, the scheduler must now decide which tasks to evict from Machine A and spawn again i.e., given all the tasks are still consuming fewer resources than their original request they are not violating their original ask limits. While it might appear that eviction and respawning is a wasteful exercise, (and it is for the task being evicted) large scale production cluster traces indicate that evictions are still rare compared to overall task completions (Tirmazi, et al., 2020). Significant research has been done to evaluate scheduling policy alternatives to determining the most effective way of identifying which tasks to evict and reschedule (Zhang, Shae, Zheng, & Jamjoom, 2012).

Getting the overcommitment ratio wrong can cause the entire data center to be overcommitted causing frequent task evictions and respawns. Memory overcommitment can be particularly detrimental to performance of tasks and completions (Williams, Jamjoom, Liu, & Weatherspoon, 2011). The ability to manage overcommitment effectively without performance degradation is likely going to be a crucial advantage to managing profitability while providing superior customer service in the increasingly competitive cloud computing landscape.

1.3. Problem Statement

Workload schedulers that overcommit resources (CPU/Memory) to improve hardware utilization for CSPs currently max out at 60% resulting in millions of dollars of wasted infrastructure investments.

Overcommitment is important to improving the utilization of cloud computing clusters and hence is widely used by CSPs. However, the scenarios in which it can be pushed to its limits are not well understood (Tirmazi, et al., 2020). Availability of a decision model to adjust overcommitment rates dynamically depending on changing workloads and infrastructure can help optimize the available hardware and hence maximize the capital investments in large cloud computing clusters (Gandhi, Chen, Gmach, Arlitt, & Marwah, 2011; Barroso & Hölzle, 2009).

1.4. Thesis Statement

Datacenter schedulers need a predictive model to adjust overcommitment rates to improve hardware utilization.

1.5. Research Objectives

- Understanding the relationships between job arrival rates, machine heterogeneity, overcommitment rates and resource (CPU/Memory) utilization while controlling eviction rates.
- Developing a Generative Discrete Event Simulation based predictive model to evaluate the relationships

- Demonstrating that the availability of such a predictive model will improve utilization as the relationships between dependent and independent variables become clear

1.6. Research Questions and Hypotheses

This praxis focuses on developing a decision model that could be utilized to understand the constraints within which overcommitment values could be altered to improve overall resource utilization. The following research questions must be answered to achieve the objectives:

1. Do heterogenous clusters improve memory utilization?
2. Do heterogenous clusters improve CPU utilization?
3. Does eviction rate prediction improve by incorporating an interaction between job arrival rates and overcommitment rates?

4: Can overcommitment rates be used to predict CPU and Memory utilization?

The following hypothesis were formulated to address the above questions

H_1: For constant job arrival, eviction and overcommitment rates, heterogenous clusters improve median memory utilization

H_2: For constant job arrival, eviction and overcommitment rates, heterogenous clusters improve median CPU utilization

H_3: Eviction rate prediction can be improved by incorporating an interaction between job arrival rates and overcommitment rates

H_4: CPU and memory utilization can be predicted using job arrival, job eviction and overcommitment rates

1.7. Scope and Research Limitations

Cloud workloads are impacted by many interrelated factors. Some of the important ones are scheduling policy, task mix (batch, real-time etc.), task priorities (high, low etc.), choice of bin-packing algorithm, machine downtimes, re-spawning policies etc. Real-world data that covers all combinations of these factors across extended runs of time is not easily available and, even if it were, meaningful assessment would be highly complex considering the enormity of the dataset (Buyya, et al., 2015). Furthermore, even though companies like Google, Alibaba, Microsoft have open sourced significant volumes of trace data from their cloud workloads (Tirmazi, et al., 2020; Guo, et al., 2019; Cortez, et al., 2017) their scheduling policies, bin-packing techniques, overcommitment rates, respawning techniques etc. are still proprietary (Chandini, Pushpalatha, & Boraia, 2016) and inaccessible for research purposes.

This research focuses on scheduling practices and policies of cloud operations that maximize resource utilization. It attempts to understand resource utilization challenges, especially the impact overcommitment can have on eviction rates in certain scenarios where task arrival rates and machine heterogeneity changes. It does so by simulating hundreds of discrete event scenarios where several factors are held constant while only task arrival rates, oversampling rates and machine heterogeneity are varied.

The research also makes several simulation design decisions for bin-packing, respawning, task runtimes, resource usage patterns, resource requests and varies them similarly (randomly) across all the simulations. Simulations assume uniform priority of all tasks and do not account for DAG dependencies (i.e., all arriving tasks are independent of each other). While the simulation could be altered to account for changes

to several of these factors, potentially any change in the simulation assumptions could influence the results and final conclusions.

This research contributes to an improved understanding of the effect of task arrival rates, machine heterogeneity and overcommitment rates on overall utilization while controlling for evictions. It defines a predictive model that can be used to influence scheduling policies for workloads that meet the assumption set used by the simulated data.

1.8. Organization of Praxis

The Praxis is divided into five chapters that cover the following areas:

Chapter 1: Introduction. The first chapter explains the complexities of managing large scale cloud datacenters and highlights the tradeoffs CSPs make between meeting their SLAs and maximizing their capital investments in hardware resources. This chapter offers an overview and the motivation for the problem, defines the objectives and approach, and lists the limitations of this body of research.

Chapter 2: Literature Review. This chapter offers a detailed review of the research relevant to this work. It covers cloud computing workloads and strategies for simulating cloud workloads.

Chapter 3: Research Methodology. This section presents the proposed methodology of using discrete event simulations for simulating cloud workloads. It explains the core python package (simpy) that is used and assesses several alternatives for algorithms that could be used for evaluating the simulated data and explains why linear regression is chosen as the best alternative for drawing final conclusions.

This chapter also describes the 2020 v3 Google trace data, exclusions that were made and how it was used to understand and setup the simulation parameters.

Chapter 4: Results. This section describes the simulation parameters, the generated synthetic data, and explains the summarizations that were created for analysis. It also explains the quality assurance tests that were performed to ensure reliability of the simulations. Finally, this section presents the results of the regression model and evaluates the parameters to understand the impacts the independent variables have on the dependent variables. Several graphical charts are provided to aid with a visual understanding of the resulting analyses.

Chapter 5: Conclusions and Future Work. This chapter discusses the overall assessments from the simulated data and the findings from the regression model evaluating the research hypotheses. This section also recommends possible future areas of research that can expand on this body of work and generalize these findings across several other scenarios.

Chapter 2—Literature Review

2.1. Overcommitment in Cloud Computing

Cloud computing has allowed individual users and enterprises to move computing workloads from their on-premise datacenters to remote servers managed by CSPs in a reliable and efficient manner. (Ji, Da Li, Ji, & Li, 2018). Simultaneously, competition between CSPs is fierce. As CSPs race to replace on-premise datacenters they've cut computing prices aggressively to attract customers. (Cohen, Keller, Mirrokni, & Zadimoghaddam, 2019; Durkee, 2010) However, utilization of computing clusters remains low ranging between 20%-40% often (Cheng, Anwar, & Duan, 2018). Improving resource utilization can allow the CSP to reduce costs significantly and create a competitive advantage.

Jobs arriving at a CSP request a specific resource requirement. However, their actual usage is not known until the job is scheduled. If the job scheduler allocates resources per the job's request (e.g., allocate exactly16 CPUs to a job that requested 16 CPUs on arrival) then the jobs that under-utilize their allocation will lock up computation capacity that could be allocated to other jobs. This creates an incentive to overcommit (i.e., allocate the limited resources such that the total requests exceed the physical capacity that is available). Because the actual usage is usually lower than the requests, this proves effective in spreading the finite, shared and expensive resources and support more job requests than otherwise (Nakazawa, Ogata, Seelam, & Onodera, 2017). However, when resource overcommitment leads to an actual overutilization of resources (compared to what's physically requested or available) the scheduler must make

decisions to evict or kill the jobs and then relocate them to a difference compute instance (Dabbagh, Hamdaoui, Guizani, & Rayes, 2015).

2.2. Multi-objective Scheduling Policies

A modern cloud datacenter is a complex system of interactions and interdependencies. The type and mix of workloads, the service level guarantees, the scheduling and provisioning policies, the type of hardware and software are just a few of the factors that determine the overall performance and utilization of a public cloud computing environment (Min, Park, Lee, Cho, & Kim, 2011). CSPs define scheduling policies that are applicable to their specific customers and their Quality of Service (QoS) objectives (Ma, Chu, Zhao, & Ankhbayar, 2014).

2.2.1. Job Arrivals and Transitions

Over the years a variety of different workloads have been migrated to the cloud by enterprises small and large. These range from single user interactive jobs (e.g., surveys) to highly interactive jobs that support multiple users (e.g., interactive gaming). There are cloud workloads where users generate a lot of data (e.g., Google Maps) while others that consume a lot of data (e.g., Netflix). These workloads utilize different types of cloud resources at different rates. For example, streaming video like YouTube consumes a lot of network bandwidth while e-mail applications consume network as well as persistent data-storage capabilities in the cloud (Mulia, et al., 2013). CSPs have taken varying approaches to processing these jobs. Some have co-located these jobs while others have bifurcated them into different computing clusters (Verma, et al., 2015).

However, a common pattern among job arrivals is that they go through several state transitions throughout their lifecycle as shown in Figure 2-1.

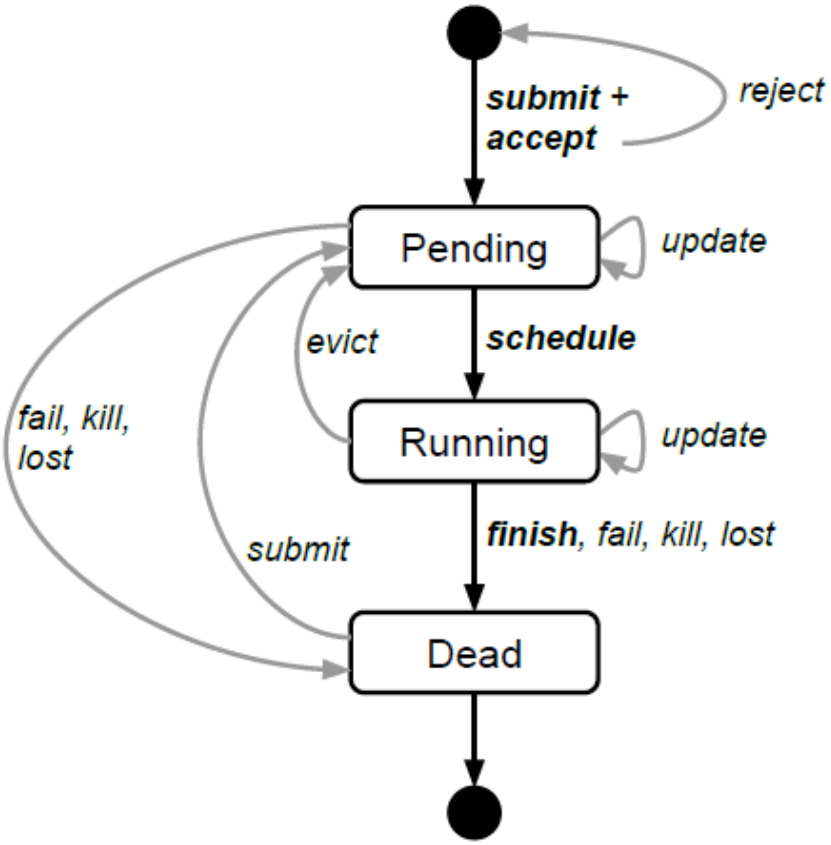

Figure 2-1: State transition diagram for jobs and tasks. Users can trigger submit, kill, and update transitions. (Source: (Verma, et al., 2015))

Many of these transitions could be user induced (e.g., a user might kill a job manually). Other transitions are managed by the scheduler depending on the state of the job and its interactions with the computational environment (e.g., a job that is consuming more resources than its allocated capacity might need to be evicted and rescheduled). Another job might need to be killed and restarted if the machine it is running or is taken down for maintenance. CSPs must balance the complexity of supporting these diverse

workloads, the job state transition complexities alongside Quality of Service (QoS) commitments for their customers while ensuring profitability given their massive capital investments in a highly competitive environment (Durkee, 2010).

2.2.2. Resource Allocation and Scheduling (RAS)

Resource Allocation and Scheduling (RAS) has been extensively studied. A variety of scheduling strategies have been offered by researchers such as (Chen, Zhang, Guo, Deng, & Guo, 2010), (Huang, et al., 2012), (Hammoud & Sakr, 2011), (Isard, et al., 2009), (Jia, Yang, Grundy, Keung, & Li, 2019), (Jin, Luo, Song, Dong, & Xiong, 2011), (Jin, Jun-zhou, Ai-bo, & Fang, 2011), (Rajan & Francis, 2015), (Tao, Zhang, Shi, & Chen, 2011), (Wang, Zeng, Tang, & Yao, 2012), (Wang, Zhang, Xu, He, & Guo, 2016), (Xiao & Wang, 2012), (Zhang, Feng, Feng, Fan, & Ming, 2011), (Zaharia, et al., 2010) to name a few. These researchers mentioned above have focused on increasing reliability, improving data-locality, energy efficiency and dynamic self-adoption of the scheduler. CSPs employ RAS policy alternatives appropriate to their QoS objectives and their commitments to their customers. An example taxonomy of Resource Allocation and Scheduling (RAS) Policy Alternatives is shown in Figure 2-2.

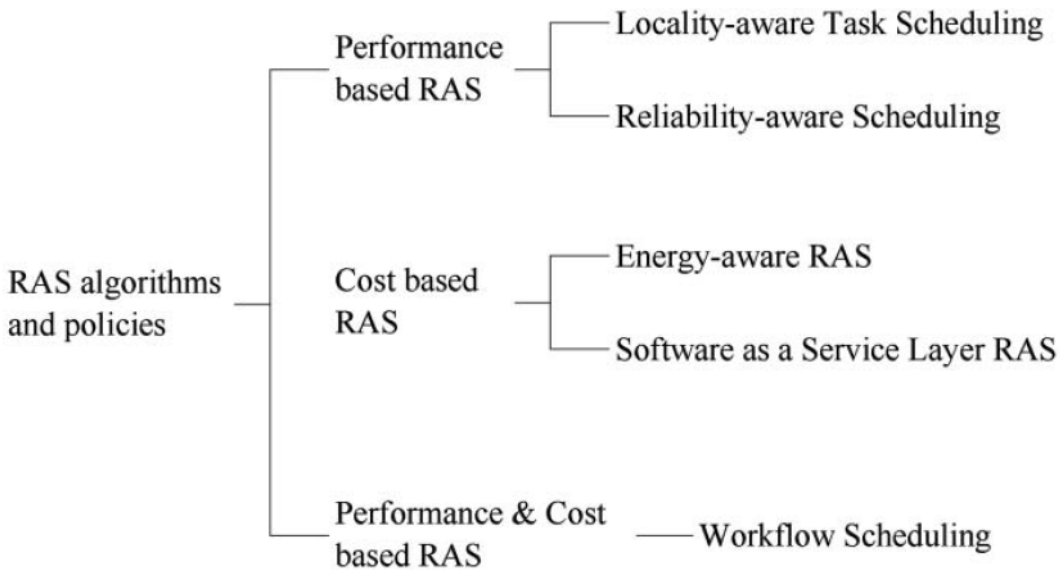

Figure 2-2: A taxonomy of RAS algorithms and policies. (Source: (Ma, Chu, Zhao, & Ankhbayar, 2014))

CSPs can employ RAS policies in a static or dynamic fashion. Static RAS policies establish a policy in advance and pre-schedule resources accordingly. Dynamic RAS policies, on the other hand, continuously monitor the state of the system and availability of resources and make allocation decisions. Both alternatives have their pros and cons. Some CSPs, use both static and dynamic RAS simultaneously (Varalakshmi, Ramaswamy, Balasubramanian, & Vijaykumar, 2011; Ma, Chu, Zhao, & Ankhbayar, 2014).

Other researchers like (Cohen, Keller, Mirrokni, & Zadimoghaddam, 2019), (Dabbagh, Hamdaoui, Rayes, & Guizani, 2016), (Ji, Da Li, Ji, & Li, 2018), (Takayama & Kourai, 2020), (Wang, Zhang, Xu, He, & Guo, 2016), (Zhang, Han, He, & Chen, 2017) have studied resource allocation and scheduling in overcommitted environments with a

focus on network-awareness or predicting migration patterns for jobs/virtual machines. Nakazawa et. al. studied performance degradation in clouds where memory is overcommitted (Nakazawa, Ogata, Seelam, & Onodera, 2017). Their research used containers that process two different types of workloads. The heavy workload was kept constant, and the light workload was varied to process requests at a fixed interval. They recommended that memory overcommitment can be beneficial to improving performance and suggested solutions for reducing performance degradation.

Most of these studies describe the effects of overcommitment on the cluster under a pre-identified set of workloads. They do not evaluate cluster performance using a combinations of variable machine configurations, variable job runtimes and variable task arrival rates. Also, they do not address how an overcommitment rate could be adjusted and the related effects as job arrival rates and machine heterogeneity changes. Finally, they use real-workloads and physical instances (Virtual Machines or Containers) for evaluating the scenarios without leveraging discrete event simulations.

Next section provides an overview of why simulations (discrete event simulations, in particular) are an effective tool for rapidly collecting large samples of observations that mimic real-world events.

2.3. Simulations for Modeling Cloud Workloads

Simulations attempt to mimic the behavior or function of a process or operation within a system (Banks, Carson, Nelson, & Nicol, 2013). They have been applied in many fields and are widely used in computer modeling and technology experiments for performance testing, and training & safety (Srinivasan, 2020). They are useful for

measuring the potential impact of a course of action and identifying ways to avoiding specific outcomes.

Simulations are preferred when the real course of action is perceived to be unachievable, undesirable, risky, or non-existent (Sokolowski & Banks, 2009). They are a powerful tool that can be used to model and compare alternative scenarios when decision makers are evaluating complex operations systems (Shannon 1998). Simulations offer finer control of various factors and help test new scenarios cost effectively compared to scaling up hardware instances on a CSP (Idziorek, 2010). Furthermore, simulations are particularly useful when complex interactions between system components make it difficult to predict the effect of making any changes (Carson, 2005). This complexity is exacerbated by the fact that different CSPs experience varying workloads, host different hardware and software stacks, commit to varied service level obligations, and support a diverse array of end-use and batch applications with different performance guarantees (Calheiros, Ranjan, Beloglazov, De Rose, & Buyya, 2011).

As computing power has become cheap and ubiquitous, simulation has become a widely used approach for areas such as control system design, scheduling, training, real-time control systems, predicting future performance etc. (Robinson, 2005; Heavey, et al., 2014). This added flexibility offered by simulations allows for assumptions to be adjusted for the problem that is being analyzed (Heavey, et al., 2014). When appropriate standards, software practices and tools are ensured simulation can offer much more reproducible way of testing alternative hypothesis during the design stage of any complex system (Buyya, Ranjan, & Calheiros, 2009; McDougal, Bulanova, & Lytton, 2016).

2.3.1. Real-world Systems and Types

A real-world system is a group of entities interacting with each other, independently or otherwise, to accomplish a particular set of objectives (Banks, Carson, Nelson, & Nicol, 2013). Systems can be affected by external changes in its environment and the set of allowable interactions depend on the purpose of the simulation study being conducted. Systems can be broadly classified in two categories:

- Discrete Systems: In a discrete system entities or events are thought of as moving from one time interval to another future discrete time interval in a chronological order. They compete with other entities in the system for the limited resources while doing so (Schriber, Brunner, & Smith, 2017). Customers arriving at a restaurant, cars arriving at a gas station (Figure 2-3), calls arriving in a customer service call center, cars passing through a road intersection, moviegoers arriving at a movie theater are some examples of discrete systems.

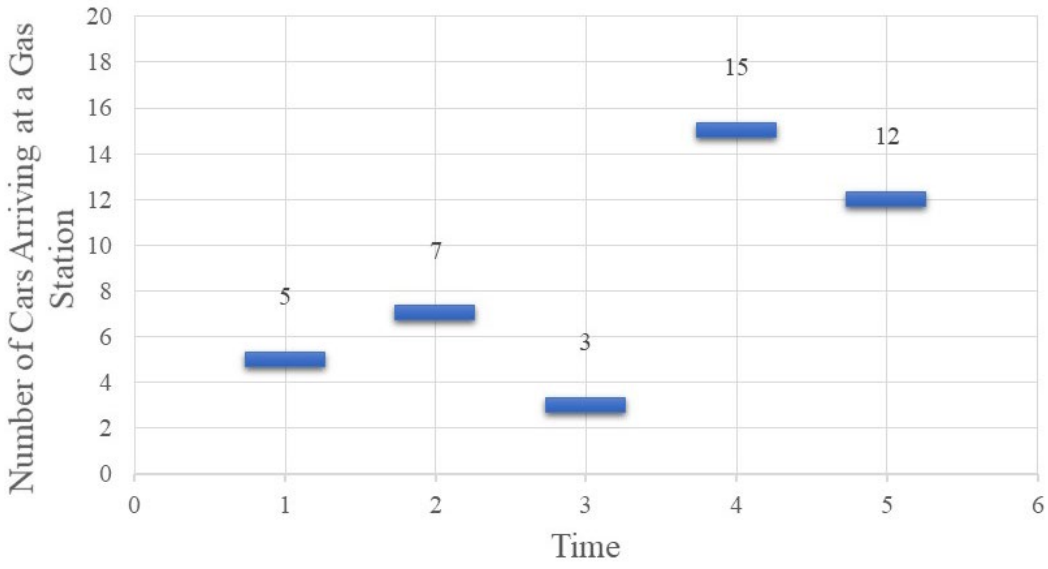

Figure 2-3: A Discrete System Example

- Continuous Systems: Entities/Events/Variables in a continuous system change their state continuously from one state (or value) to another throughout time. Temperature levels of a particular region, wind speed (
- Figure 2-4) and other weather patterns, and water level in a reservoir are examples of continuous systems.

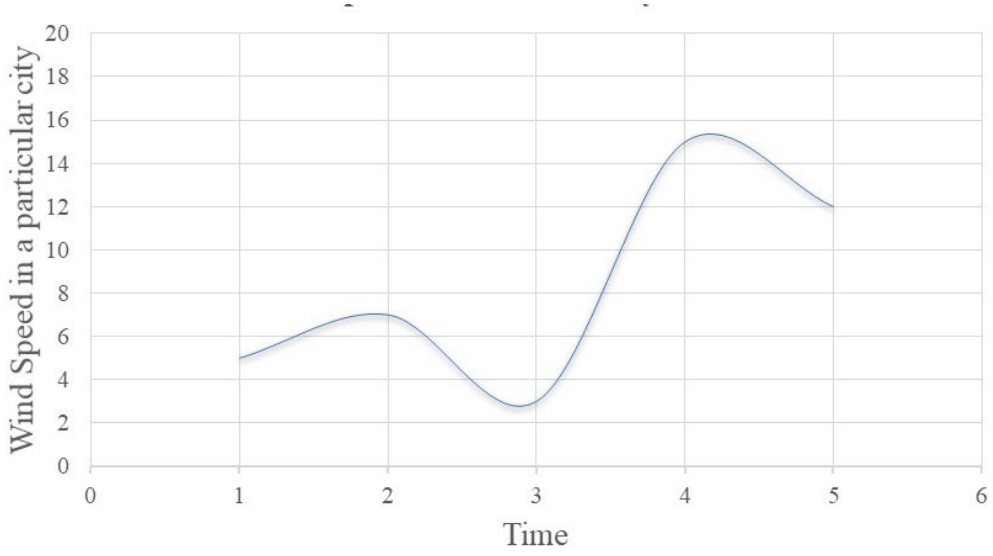

Figure 2-4: A Continuous System Example

2.3.2. Discrete Event Simulations for Modeling Cloud Workloads

Few systems can be thought of as wholly discrete or continuous. For purposes of analysis and modeling such systems we can consider one type of change (discrete or continuous) to be the predominant one and model them as such (Law, 2007). Real-world events often appear to occur concurrently (at the exact same time). However, it is a matter of resolution or precision of the time scale. For example, two customers arriving at a restaurant might appear to arrive at the same time. However, once the resolution of the

clock is increased (e.g., from hours to minutes or from minutes to seconds) then these seemingly continuous events can be treated as discrete. With higher precision of time (e.g., seconds to microseconds) even smaller time differences between events can be turned into discrete ones. Of course, this depends on the floating-point precision of the simulation software and the computing machinery the simulation runs on (SimPy, 2018). Given a single CPU core executes threads sequentially, simultaneous events can be discretized by executing them in the order in which they were created in the underlying computer data-structure utilized by the simulation software (Peschlow & Martini, 2007).

CloudSim, a Java based discrete-event simulation framework, has been widely adopted by several researchers like (Alves, et al., 2016), (Jung & Kim, 2012), (Humane & Varshapriya, 2015), (Bendechache, et al., 2019), (Kathiravelu & Veiga, 2014), (Rani & Kaur, 2017), (Pratap & Zaidi, 2018), (Santra & Mali, 2015), (Alves, et al., 2016), (Rawat, Dimri, Saroha, & Barthwal, 2016) for modeling cloud workloads over the past several years. This shows that Discrete Event Simulations are a practical approach for modeling and evaluating cloud computing workloads without incurring financial costs of procuring virtual machines or containers on the CSPs.

2.4. Summary

Resource Allocation and Scheduling (RAS) alternatives have been widely studied over the past several years. The studies acknowledge the complexity and the multi-objective, multi-constraint nature of the problem. Given these complexities, varied attempts have been made that create both simulated workloads as well as real workloads on a fixed set of machines. The studies have focused on improving a Quality of Service (QoS) attribute like Energy Utilization, Resource Wastage, Virtual Machine Swapping

etc. but have not addressed how the overcommitment rate gets affected by factors like job arrival rates and machine heterogeneity.

(Tirmazi, et al., 2020), in their review of the 2019 Google cluster trace, inquire about the extent and circumstances in which overcommitment can be pushed. This research attempts to answer that question in the context of variable job arrival rates and machine heterogeneity. This research also extends the work of (Nakazawa, Ogata, Seelam, & Onodera, 2017) by first using simulations rather than a fixed set of procured machines. This research, in contrast, focuses on the impact of job arrivals and machine heterogeneity on throughput and eviction rates when overcommitment rates change. Varying job arrival rates and machine heterogeneity are the norm in modern cloud workloads (Cheng, Anwar, & Duan, 2018) and hence this research adds to an understanding on how overcommitment rates can impact cluster performance as job arrival rates varied.

Chapter 3—Methodology

3.1. Introduction

Reproduction of cloud workloads is a complex task. Understanding the properties, behavior and interactions of such workloads is essential for capacity planning and cost optimization for a CSP's infrastructure (Calzarossa, Massari, & Tessera, 2016; Dezhabad, Ganti, & Shoja, 2019). Simulated workload models offer the benefit of being flexible enough to be adjusted to fit tightly controlled scenarios where certain variables are held constant while a few are varied (Magalhães, Calheiros, Buyya, & Gomes, 2015). Modeling can be also performed more efficiently to control for certain parameters only compared to logging and tracing a production workload that contains thousands or even millions of numbers (Feitelson, 2015).

CSP workloads are highly variable and only observable through trace logs made available to the public from some of the CSPs (Kavulya, Tan, Gandhi, & Narasimhan, 2010; Cheng, Anwar, & Duan, 2018). However, concerns around client confidentiality, organization policies and intellectual property make several parameters essential to modeling workloads unobservable in these traces (Alam, Shakil, & Sethi, 2016). In addition, CSPs workloads are also likely to change over time. For example, Mulia et. al. estimated that IT managers refresh about 25%-30% of their platforms every year yielding a heterogenous mix of hardware generations in each data center (Mulia, et al., 2013). Models that can study the relationship between dynamic job characteristics and their interactions with the environment are therefore effective in improving the Quality of Service (QoS) offered to customers (Moreno, Garraghan, Townend, & Xu, 2013).

Feitelson describes two types of workload modeling techniques: Descriptive Workload Models and Generative Workload Models (Feitelson, 2015). Both approaches rely on monitoring and tracing a real-world production system that is utilized by its respective users. The traces, generated over a defined period, log the important attributes of the entities that are interacting in the environment in question. The two modeling techniques are differentiated by the approach each takes to creating a synthetic job stream based on a real production system. The differences in the approaches are explained below starting with the Descriptive Workload Modeling approach in Figure 3-1.

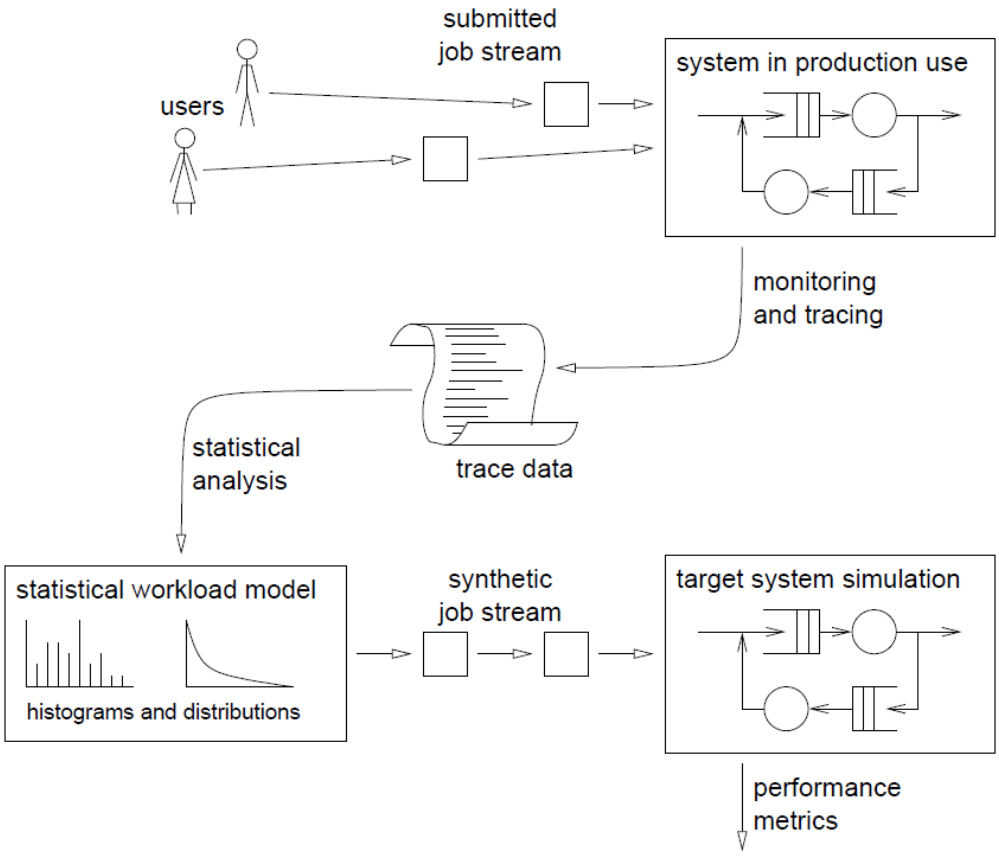

Figure 3-1: Descriptive Workload Modeling based on statistical analysis of the trace. (Feitelson, 2015)

This approach summarizes the production trace data by extracting its important statistical properties (shape of the distribution, moments of the distribution etc.) using exploratory data analyses and other statistical techniques. A synthetic workload is then created by sampling from the distributions and the necessary performance metrics are evaluated from the synthetic workload. Descriptive modeling uses statistical analyses to strike a balance between faithfully replicating features of the original data and abstracting complexity of the production system. Figure 3-2 depicts a Generative Workload Modeling approach.

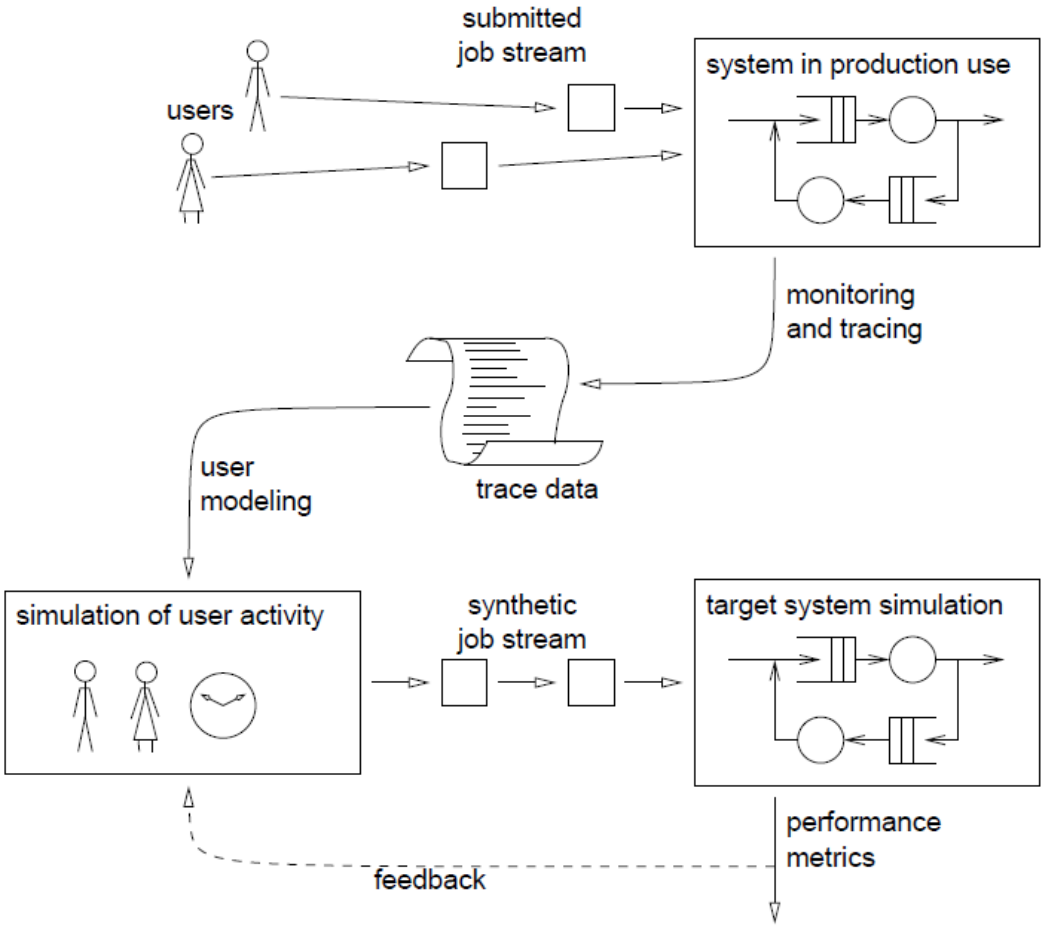

Figure 3-2: Generative Workload Modeling based on analyzing user behavior in a production system. (Feitelson, 2015)

Generative modeling takes a different approach to creating a target simulation for evaluation. Instead of modeling the workload through statistical properties or fitting distributions to mimic the data this approach models the process that created the production trace. This approach focuses on simulation of user (or entity) behavior that created the production trace instead. For e.g. In a call center environment, we might want to simulate the type of end-user calls that arrive during different days of the week or different hours of the day. The expectation is that if the user emulation is done accurately then the relevant distributions of attributes under study will be generated automatically.

Because we are modeling user behavior, a Generative Modeling approach helps us alter the simulated user behavior and thus manipulate scenarios that might not be observable in a production environment. For e.g. In a well-managed production call center environment, we might not observe system outages during business hours. Simulating the effect of business hour system outages on users that abandon the calls would not be possible because of lack of observable data points. Such simulations can be performed using a Generative Model that accurately depicts user call patterns. This has the added advantage of allowing us to study relevant interactions between the users/entities and the system/environment within which they are operating.

The challenge in both the approaches is ensuring the targeted simulation derived from the statistical summaries or the user activity can capture the specific workload feature being evaluated. The goal is not a complete reproduction of the production environment. This research uses a Generative Workload Modeling approach.

3.2. Generative Modeling Flow Diagram

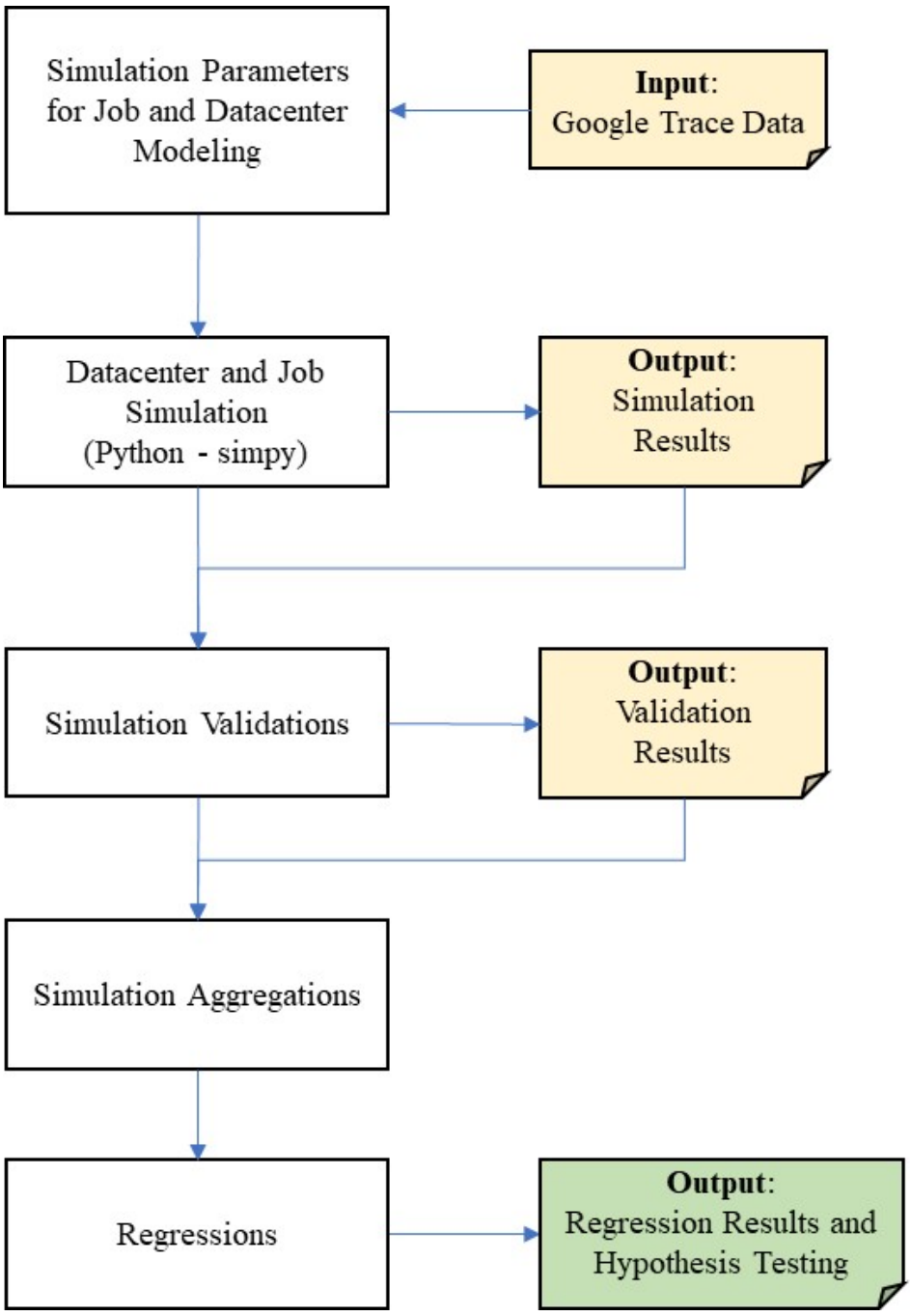

Figure 3-3: Generative Workload Modeling and subsequent Hypotheses Evaluation Flowchart

The modeling flow diagram shown in Figure 3-3 explains how the generative simulation modeling approach was utilized and the output evaluated to answer the specific research questions. Publicly available Google cluster trace data (version 2019) were used for identifying appropriate simulation parameters for this research. The trace data spans the entire month of May 2019 and includes job scheduling events and interactions from 8 different Google Borg cells. Each Borg cell is a collection of machines (several thousands) grouped together as one unit and shared by several users. The 2019 trace covers 96.4k machines, 7 different hardware platforms, 21 different machine shapes with about 12k machines per cell (Tirmazi, et al., 2020). Figure 3-4 shows a mapping of original to emitted times (0-offset timestamps for each record) for the trace. This large dataset is 2.4 TB in compressed form and is supplied in BigQuery Tables on Google Cloud (Wilkes, 2020).

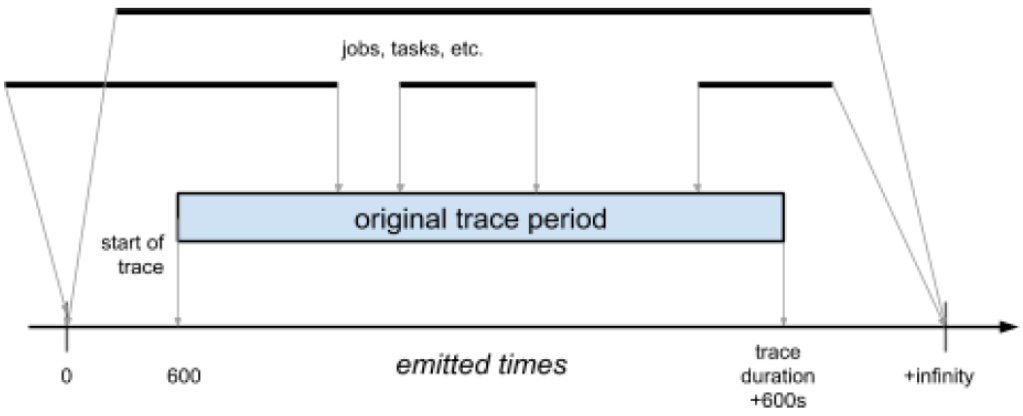

Figure 3-4: Mapping of original times to emitted times in the trace. (Wilkes, 2020)

Job arrival patterns, machine heterogeneity compositions and overcommitment rate are the relevant parameters for this research. However, parameters like bin-packing

algorithm (deciding which jobs are assigned to which machines) and overcommitment rate are not observable in the Google trace. In this research, the bin-packing algorithm was held constant across all the simulations. The overcommitment rate was picked randomly from a specified range for each simulation. Once an overcommitment rate is set for a simulated run it is held constant for the duration of the simulation.

The simulation modeling software and the class diagram for the simulated entities/objects are described in Section 3.33 and Section 3.4 respectively. Once the simulation results are generated, data validations are performed against the raw simulated data to ensure the results meet the quality criteria necessary for further evaluations. Once all validation tests have passed, the raw simulation results are further summarized to create simulation level aggregated analytical datasets.

The target variables that are evaluated in this research include CPU and memory utilization, job scheduling rates, job eviction rates and job wait times. These variables are created as a part of the data aggregations from the raw simulated data. Linear Regression is used to evaluate the effect of the parameters in question on the target variables. Finally, the regression model results are assessed to evaluate the hypotheses and answer the research questions outlined in Section 1.6.

3.3. Discrete Event Simulation Modeling Software

Real-world simulations require a complex set of system interactions where processes execute in parallel, and entities interact with other entities simultaneously. Instead of creating completely new simulation languages, modern approaches have chosen to create libraries that can be invoked from traditional foundational programming

languages like Java, C++, Python etc. (Matloff, Introduction to discrete-event simulation and the simpy language, 2008). Therefore, the developers and researchers do not need to learn a completely new programming language and can focus on the programming paradigm instead while utilizing all the foundational benefits and robustness of the native programming language.

Simpy, a Python based process-oriented discrete-event simulation package, was used for the simulations. Simpy makes the difficult task of simulation programming simpler by creatively using the power of Python Generators (Scherfke, 2014). Generators allow a serial system to simulate a parallel processing system by freezing the state of a function and allowing execution to begin at a later point in time from the point where the function had exited last (Schemenauer, Peters, & Hetland, 2001; VanRossum & Drake, 2010). This allows simulation and tracking of asynchronous, multi-agent interactions possible using a process-oriented framework (Oujezsky & Horvath, 2016).

Simpy has been used successfully across diverse domains. It has been used for simulating energy supply chains (Pinho, Coelho, & Boaventura-Cunha, 2016), manufacturing processes (Castillo, 2006), inventory optimization (Holden, 2017), channel allocation in wireless cellular networks (Cao & Ro, 2012), IoT applications for smart-cities (Karanjkar, Tejasvi, & Amrutur, 2019), moving-target defenses of computer networks (Xiong, Ma, & Cui, 2019) to list a few applications. In addition, entire simulation frameworks have been built using Simpy (Dagkakis, Heavey, Robin, & Perrin, 2013; Tinini, dos Santos, Figueiredo, & Batista, 2020). Several advanced features available in the package allow for efficiently modeling entity interactions without excessive programmatic burden (Matloff, Advanced features of the SimPy language,

2008) making it an effective framework for creating complex simulations. Finally, the convenience of it being a Python package offers an intuitive usage which offsets processing speed issues caused by Python being an interpreted language (Castillo, 2006).

3.4. Simulation Class Diagram

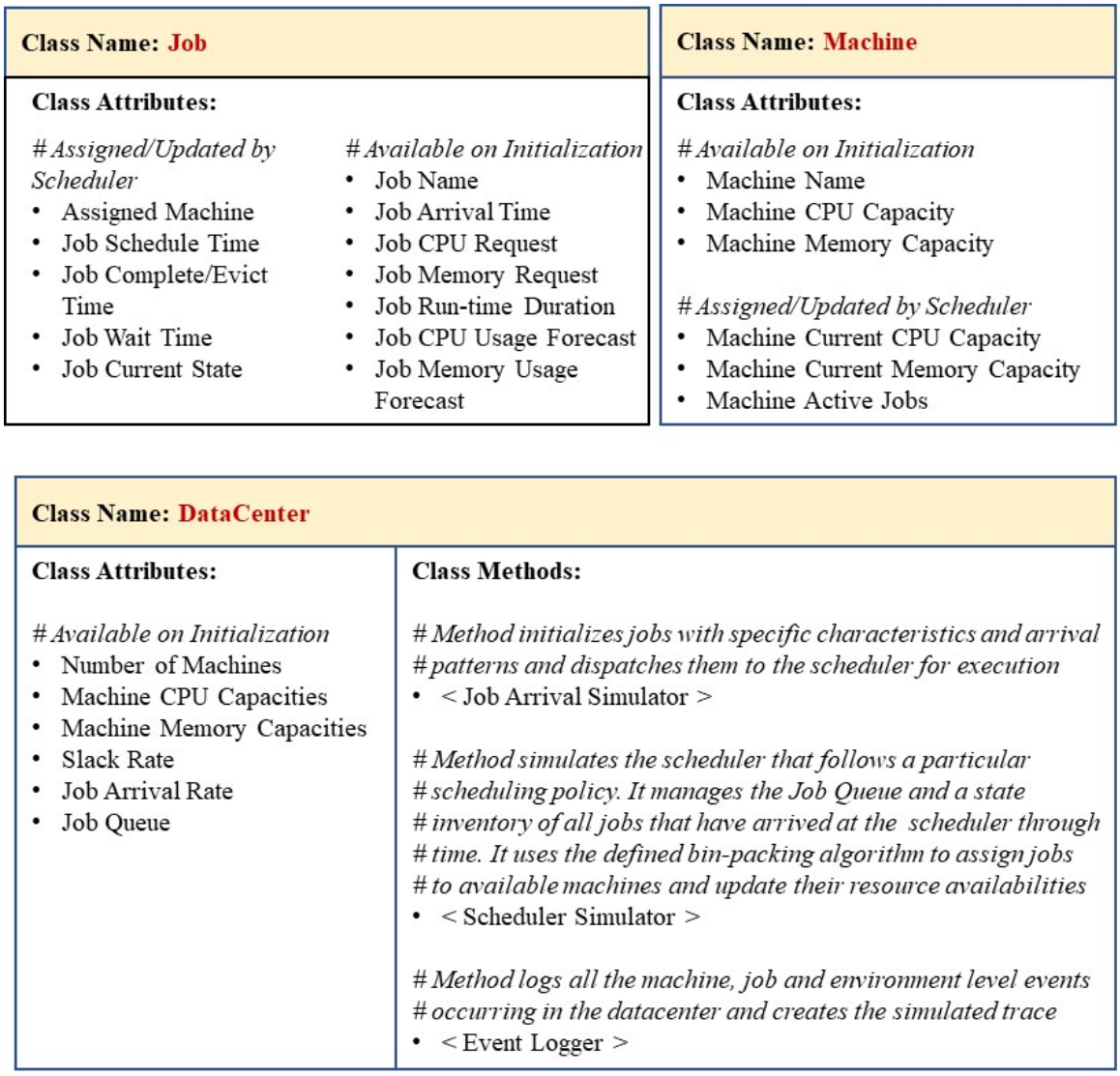

Figure 3-5: Simplified Class Diagram for the Simulation

Figure 3-5 describes the 3 key Classes involved in the simulation. The Job Class is used to create the Job objects with specific attributes. Similarly, the Machine Class is used to create the Machine objects that reside in the data center. The DataCenter Class hosts 50 machines and implements all essential methods for the simulation.

Several static attribute values of objects (e.g., Job Name, Job Arrival Time, Machine Initial Capacity etc.) are set at initialization and represent the state of the object at inception. States change as the entities interact with each other and their environment. The DataCenter object tracks these changes using attribute values that are dynamically changed as the system state changes (e.g., Job Schedule Time, Machine Current CPU Capacity, Machine Active Jobs, etc.). It also maintains a Job Queue to track the current state (as well as history) for every job that is created since the beginning of the simulation. The state transitions shown in Figure 3-6 are tracked and maintained by the Datacenter. It also maintains a complete history of the state of every machine (resource capacity) at regular time intervals.

This detailed logging of events and entity interactions allows us to create a comprehensive trace of job arrivals, completions, and evictions alongside the resource capacities of every machine in the DataCenter at regular time intervals. Finally, the scheduler method simulates the scheduling policy described in the next several sections including the bin-packing algorithm that assigns jobs to machines. The simulations were each run for a total of 2000 time units. The first 500 time units were excluded from the evaluations to allow for a sufficient job load to be processed by the simulation and to let the environment reach a steady state.

The next several sections describe the key simulation parameters that were modeled as a part of this research. The focus was on building the most parsimonious and manageable model while allowing for variation and fine-grained control on the parameters of interest.

3.5. Simulation Parameter: Job States and Transitions

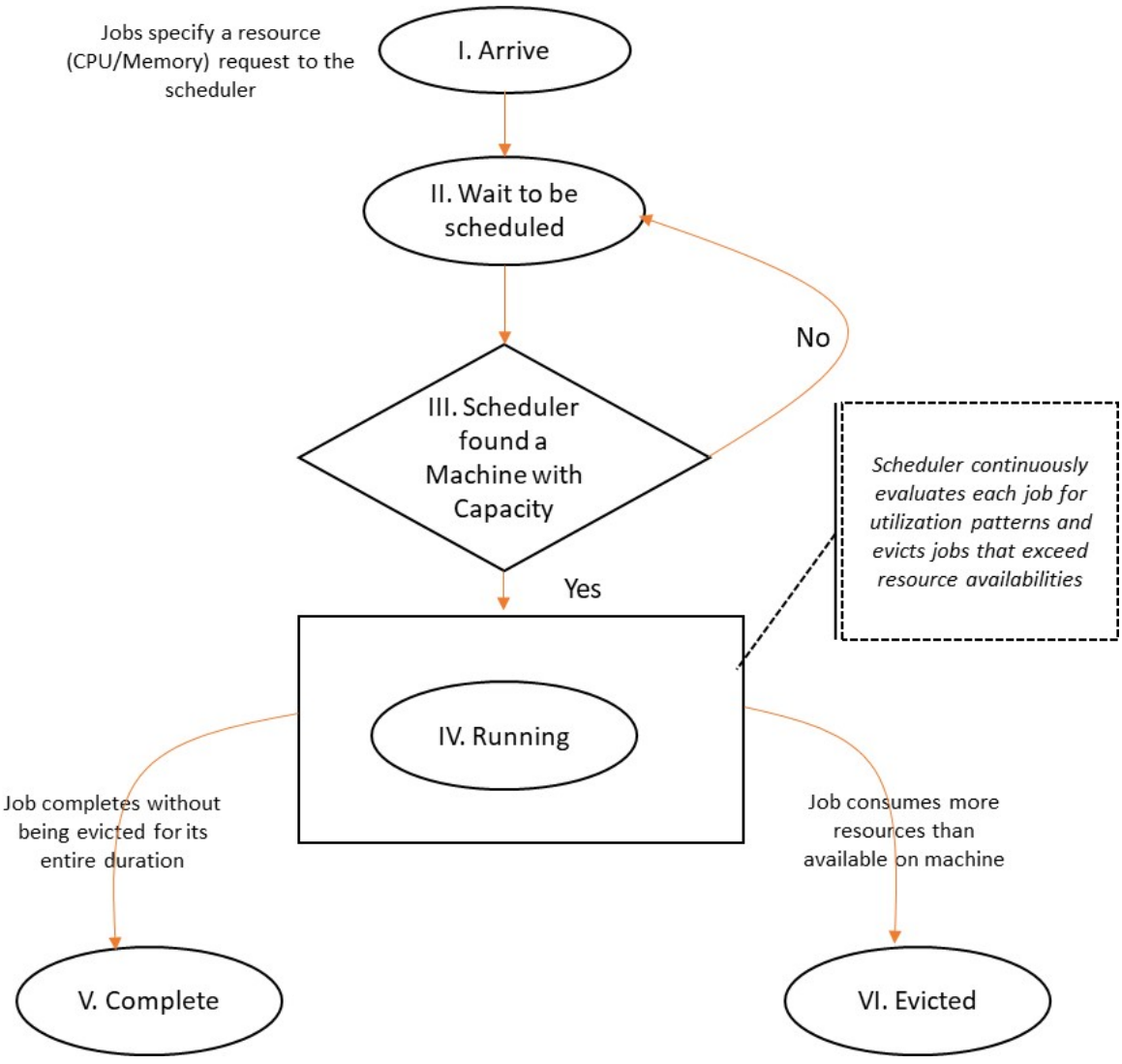

Figure 3-6: Job State Transitions in the Simulation

Figure 3-6 shows the various states jobs can take throughout their life cycle. Jobs arrive in the datacenter ready to be scheduled. As they arrive, jobs specify a specific resource request for both CPU and Memory. CPU and Memory utilization request for any job are independent of each other. The scheduler (a method of the DataCenter class) attempts to schedule the job immediately without delay. Job scheduling delays are caused if the scheduler cannot find any machine (from the ones available in the datacenter) that would fit the job resource requirements. Other than machine availability, the scheduler does not face any other internal delays (maintenance, overload etc.) in processing requests. If a machine is not available, the job waits until one becomes available. Jobs that are waiting are given priority over new jobs i.e., the scheduler follows a first-come first-served scheduling policy. A First-Fit Bin Packing approach is used as explained in Section 3.9.

In a real-world environment, jobs continue to utilize the resources they need and the CSPs bill the consumer for their respective utilization (Kilcioglu, Rao, Kannan, & McAfee, 2017). CSPs that overcommit resources leverage historical usage data from similar jobs for creating a predicted job resource usage (Bashir, et al., 2021). To simulate such conditions, jobs have several latent variables that are unknown to the scheduler but essential for the simulation:

- Job Runtime: Jobs have a duration variable indicative of the length of time the job is expected to run
- Job Resource Utilization Pattern: Jobs are assigned a random utilization pattern for both CPU and Memory for the duration of their runtimes. This utilization pattern is created such that the maximum resource utilization

(CPU/Memory) never exceeds the original request. In other words, a job is expected to use resources at or below its original resource ask throughout its entire runtime. Real-world schedulers forecast resource utilization for jobs and use that for deciding the overcommitment rate. The actual usage of the job might come in lower or above the forecast. This job resource utilization pattern is created to mimic the actual run-time utilization in a production environment and hence the scheduler is not aware of it in advance. At each time epoch, the scheduler allocates cpu/memory as defined by the resource utilization pattern

Throughout the simulation, the scheduler tracks jobs continuously and transitions them from one state to another as the necessary conditions are met. A job that can be scheduled will be scheduled without delay. A job that has run for its designated duration will be marked as complete and the resources released from the machine it was scheduled on. Similarly, a job will be evicted if it is found to be utilizing resources more than what were available on the machine. It is important to note that jobs could be evicted even if they are utilizing fewer resources than their original request. This occurs because the scheduler initially allocated fewer resources than the jobs' original request. This is explained in Section 3.7 where the overcommitment rate is described further.

3.6. Simulation Parameter: Job Arrival Rates

Trace evaluations of Google Borg indicates that job arrivals follow an exponential distribution (Tirmazi, et al., 2020). A similar job arrival pattern was observed in a dataset that tracked 6 months of activity on a 600-node cluster at Facebook (Chen, Ganapathi,

Griffith, & Katz, 2010). An exponential distribution has a parameter λ, called the rate parameter, that indicates the passage of time between subsequent events. As λ increases, the gap between subsequent events is larger. Figure 3-7 shows changes in the probability density functions as the value of λ changes for the exponential distribution. Similarly, Figure 3-8 shows the cumulative distribution function for the same λ values. Therefore, a datacenter that supports a job arrival pattern with a higher λ gets scheduling requests at a faster pace than one with a lower λ.

The simulations assume an exponential arrival pattern for incoming job scheduling requests. As Figure 3-10 shows, the values of λ are varied randomly between the [1.0,50.0] range for the simulations.

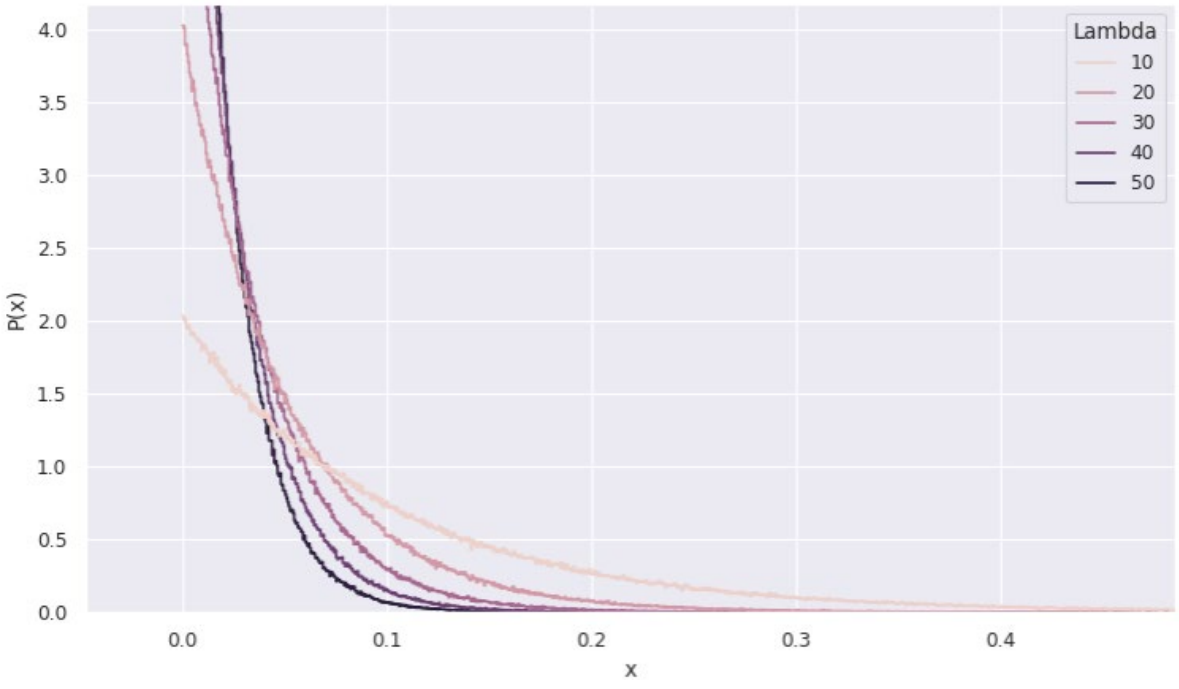

Figure 3-7: Probability Density Function of Exponential Distributions with different λ values.

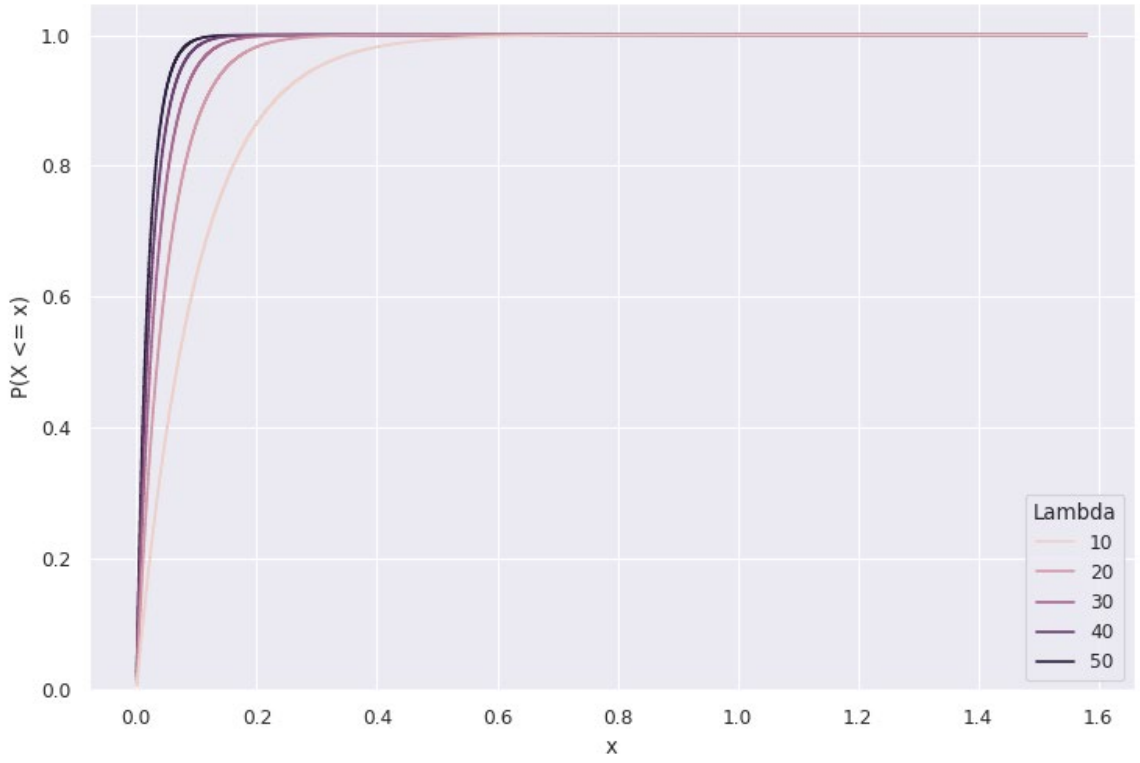

Figure 3-8: Cumulative Distribution Function of Exponential Distributions with different λ values.

A fixed number of machines (50) for each simulation are used thereby fixing the resource capacity at the start of the simulation. Hence, when all things are held equal, for a simulation, a faster arrival rate (i.e., higher λ) is likely to have higher utilization of resources. In conditions where resources are utilized fully a higher λ will cause higher wait times for jobs as newly arriving jobs wait for the scheduler to identify and allocate resources on a machine with corresponding free capacity.

3.7. Simulation Parameter: Overcommitment Rate

The following diagram explains how the overcommitment rate affects resource utilization in the context of one job.

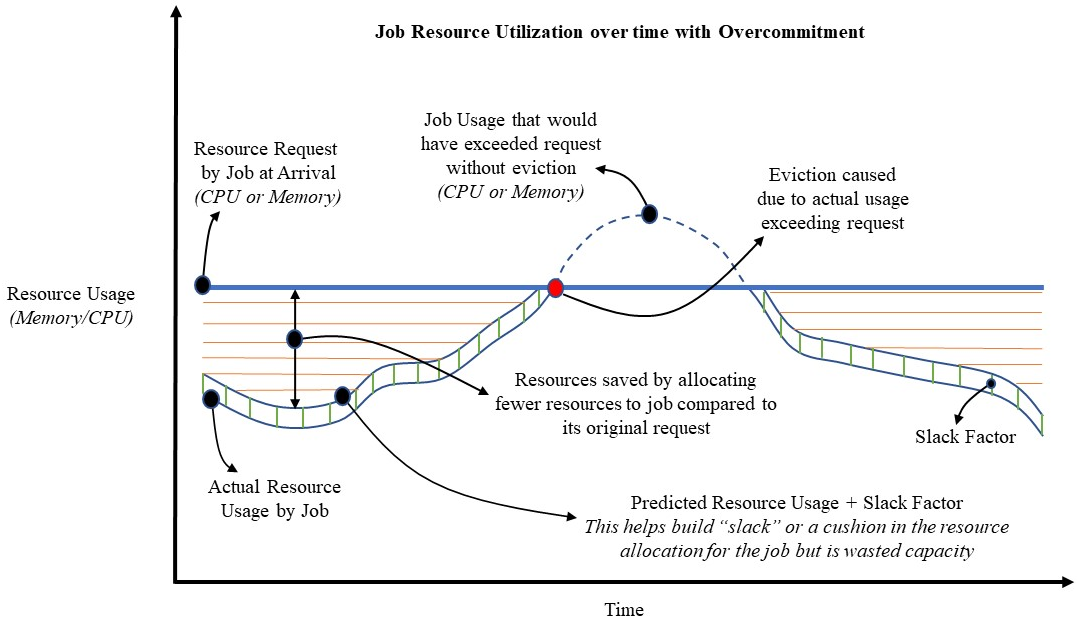

Figure 3-9: Job Resource Utilization over its run-time duration with an Overcommitment rate

The X-axis represents passage of time since arrival and the Y-axis represents units of CPU/Memory that are consumed by the job. It is applicable for both CPU and Memory requests as these resources are requested and managed independently of each other.

As mentioned in Section 3.5, jobs request a fixed capacity of resources (CPU and Memory) on arrival. When jobs arrive, schedulers create an estimate (forecast) for the resource utilization depending on history of similar jobs, environment conditions, Service Level Objectives (SLOs) etc. (Bashir, et al., 2021). To account for any prediction errors

in the forecasts, the resource estimate is padded with a slack factor. The addition of this slack factor allows for an error tolerance in situations where the actual resource usage (during runtime) for a job exceeds the forecast. This allows schedulers to overcommit resources without causing excessive job evictions. When the run-time resource usage is below the forecast, the job is allocated resources more than its utilization even in an overcommitted environment.

The logic for allocating resources can thus be presented as follows:

$$\text{Allocated Resources at time t} = \text{Max}(\text{Forecasted Resource Usage at time t} + \text{“Slack Factor”}, \text{Requested Resources at time t}) \quad (1)$$

The overcommitment rate is an inverse of the slack factor. The higher the slack the lower the overcommitment and vice versa. A scheduling policy that employs a low overcommitment rate will utilize a high slack factor and therefore adjust the Total Forecasted resource usage up. This policy will end up allocating all the resources to the job that were original requested. Alternatively, a scheduling policy that employs a high overcommitment rate will utilize a low slack factor and therefore allocate resources tracking closely with the forecasted usage.

A scheduler that does not overcommit resources, allocates a fixed quantity of resources that were originally requested for the duration of the job. This causes resources to be locked up and wasted (underutilized) if the job ends up utilizing fewer resources during runtime. Schedulers that follow overcommitment policies utilize historical data and predict a resource utilization pattern that is different from a fixed utilization pattern as requested by the incoming job (Bashir, et al., 2021). Furthermore, the utilization

pattern is continuously adjusted over the duration of the job runtime. This allows the scheduler to dynamically allocate resources lower than the original ask (Liu & Yu, 2018).

In the simulation, the scheduler allocates a maximum of requested capacity even if it predicts that the job will consume more resources than the original request. The decision on what to do when the job usage exceeds its original request depends largely on the Service Level commitments, Billing policies, workload types etc. managed by the CSP (Bashir, et al., 2021; Gkatzikis & Koutsopoulos, 2013). For simplicity of the simulation, it is assumed that the actual resource usage of jobs does not vary from the predicted resource usage. The simulations were conducted over a range of values for λ and the overcommitment rate as shown in Figure 3-9. Overcommitment rates were uniformly varied in the [1.0, 4.0] range while λ was uniformly varied in the [1.0, 50.0] range across all the simulations.

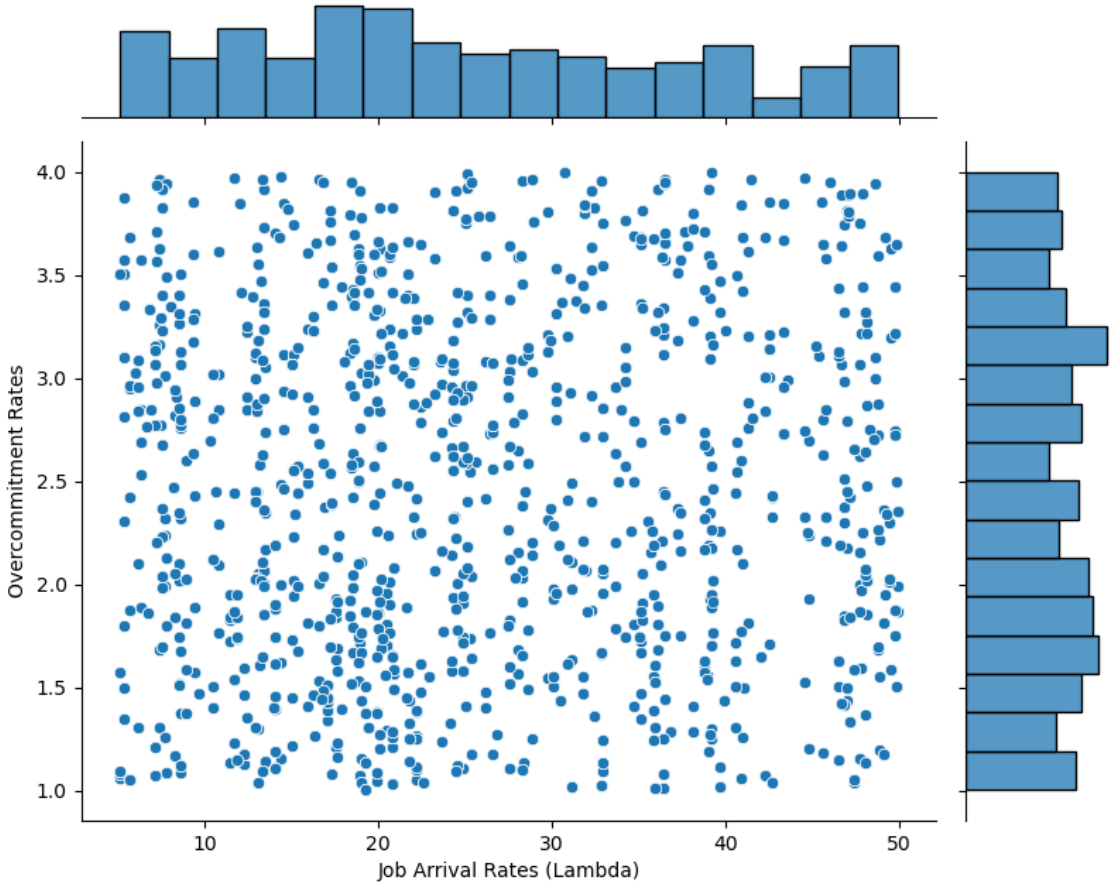

Figure 3-10: Distribution of Job Arrival Rates and Overcommitment Rates in the Simulation

3.8. Simulation Parameter: Machine Heterogeneity and Distribution

Two different machine capacity heterogeneities (CPU and Memory) were considered across the 50 machines. The distributions were chosen to have the same mean.

- Gaussian: The resources capacities are picked from a Gaussian distribution with a mean of 3 and a standard deviation of 1
- Homogenous: All the machines have fixed capacity of 3 resource units. This case models the scenario of zero heterogeneity.

Figure 3-11 shows a distribution of the total datacenter capacity for both resources (CPU and Memory) across all the simulations. Note that the mean of all the distributions is 150 units (50 machines * 3 units average per machine) as expected.

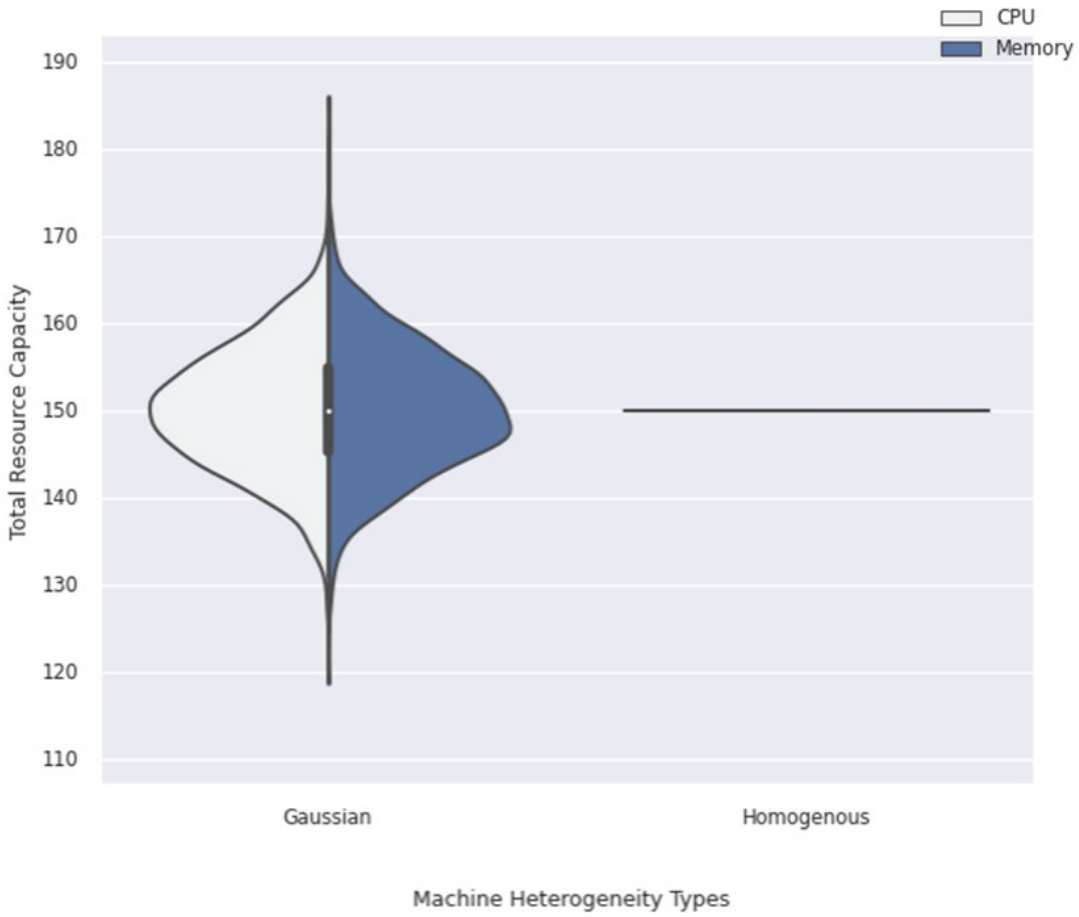

Figure 3-11: Distribution of Cumulative CPU and Memory Capacity across different simulations and Machine Heterogeneities

3.9. Simulation Parameter: Bin Packing Algorithm

As jobs arrive at a scheduler it must follow a policy for picking the machines (with free resources) where a particular incoming job would be scheduled. This process is called Bin-Packing and the corresponding problem is called the Bin Packing Problem

(BPP). It is one of the oldest and well-researched optimization problems in the field of computer science (Johnson, Near-optimal bin packing algorithms, 1973; Johnson, Fast algorithms for bin packing, 1974; Martello & Toth, 1990; Lee & Lee, 1985). The BPP algorithm receives n items of size x_1, x_2, x_3 …. x_n and attempts to allocate these to an infinite number of bins each with a fixed capacity. A bin is empty unless it is in use by one or more item(s) it contains. This is an NP-hard problem where the algorithm aims to minimize the number of bins that are used to fit the n items (Karp, 1972; Coffman, Garey, & Johnson, 1984).

There are several variations of this problem. In an *off-line* version all the items to be allocated are assumed to be known in advance. In an *on-line* version the items arrive one by one and hence each arriving item with size x_s needs to be allocated to a bin with a free capacity of at least x_s immediately. The on-line algorithm has no knowledge of the items that are yet to arrive. The goal, of minimizing the number of bins that are used, is still the same in the on-line version (Seiden, 2002).

Extensive research has been done in the field of Bin Packing over the years (Bang-Jensen & Larsen, 2012; Hasan, Kaabi, & Harrath, 2019). Several exact and approximate variants have been proposed (Tadic, et al., 2019; Hasan, Kaabi, & Harrath, 2019; György, Lugosi, & Ottucsák, 2010).

In a cloud computing cluster, the scheduler is not aware of the entire workload arriving in advance. Instead, the jobs arrive one by one and hence on-line algorithms are utilized (Tang, Li, Ren, & Cai, 2016). There are several on-line versions of on-line bin packing. Some common approaches are listed below:

- First-Fit: Place the arriving item on the lowest indexed bin (machine) that has availability. If it does not fit, then open the next indexed bin that has availability to fit the item
- Next-Fit: Place the arriving item into the current bin (if it has the necessary capacity). If not, then open a new bin and fit the item. This new bin becomes the current bin
- Best-Fit: Place the arriving item into a bin where this item fits leaving the smallest open space. If not, then open a new bin and place the item there
- Worst-Fit: Place the arriving item into a bin where this item fits leaving the largest open space. If not, then open a new bin and place the item there

Several other variants like Any-Fit, Almost Any-Fit, Harmonic-Fit, Modified Harmonic-Fit etc. that have been proposed (Delorme, Iori, & Martello, 2016) . They have focused on improving the execution time complexity as well as finding optimal solutions for both offline and online nature of this problem (Coffman, Garey, & Johnson, 1984). We follow a First-Fit Bin-packing approach for this research because of its implementation simplicity. As we will hold the bin-packing approach constant across all simulations, time performance for the algorithm is not an area of focus for this research. The following sequence of diagrams explain how First-Fit works in detail.

As shown in Figure 3-12, consider a scenario where there are 8 jobs that arrive at the scheduler sequentially space in time with different resource needs. We have 3 machines available with similar capacities for the same resource. This is the initial state at time t=0 when no jobs have arrived, and nothing has been scheduled. Each machine has 100% availability for the resource in question.

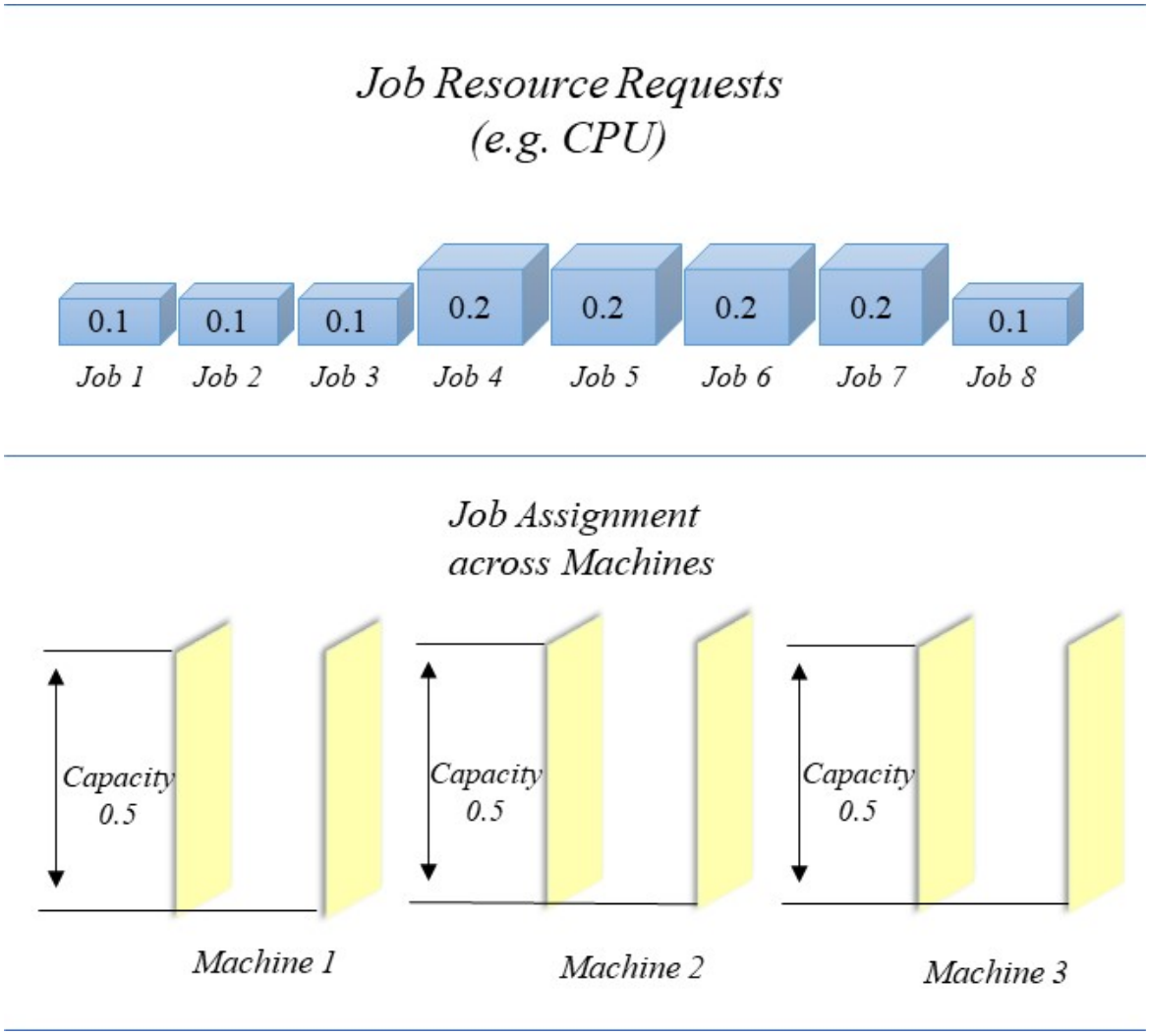

Figure 3-12: Initial System State example of Job Requests, Available Machines, and their respective resources capacities.

When Job 1 arrives requesting 0.1 units of the resource request, the scheduler finds the first machine that can meet the resource request. The scheduler scans the available machines in a specific order. We assume that the machines are scanned sequentially in a numeric order every time a job arrives. In this case the scheduler finds the first machine in the scan order - Machine 1 to have available capacity and proceeds to schedule the job on it. The state of the system after Job 1 is scheduled is shown in Figure

3-13. Now Machine 1 has 0.4 units of resource capacity left while the other 2 machines are still unutilized.

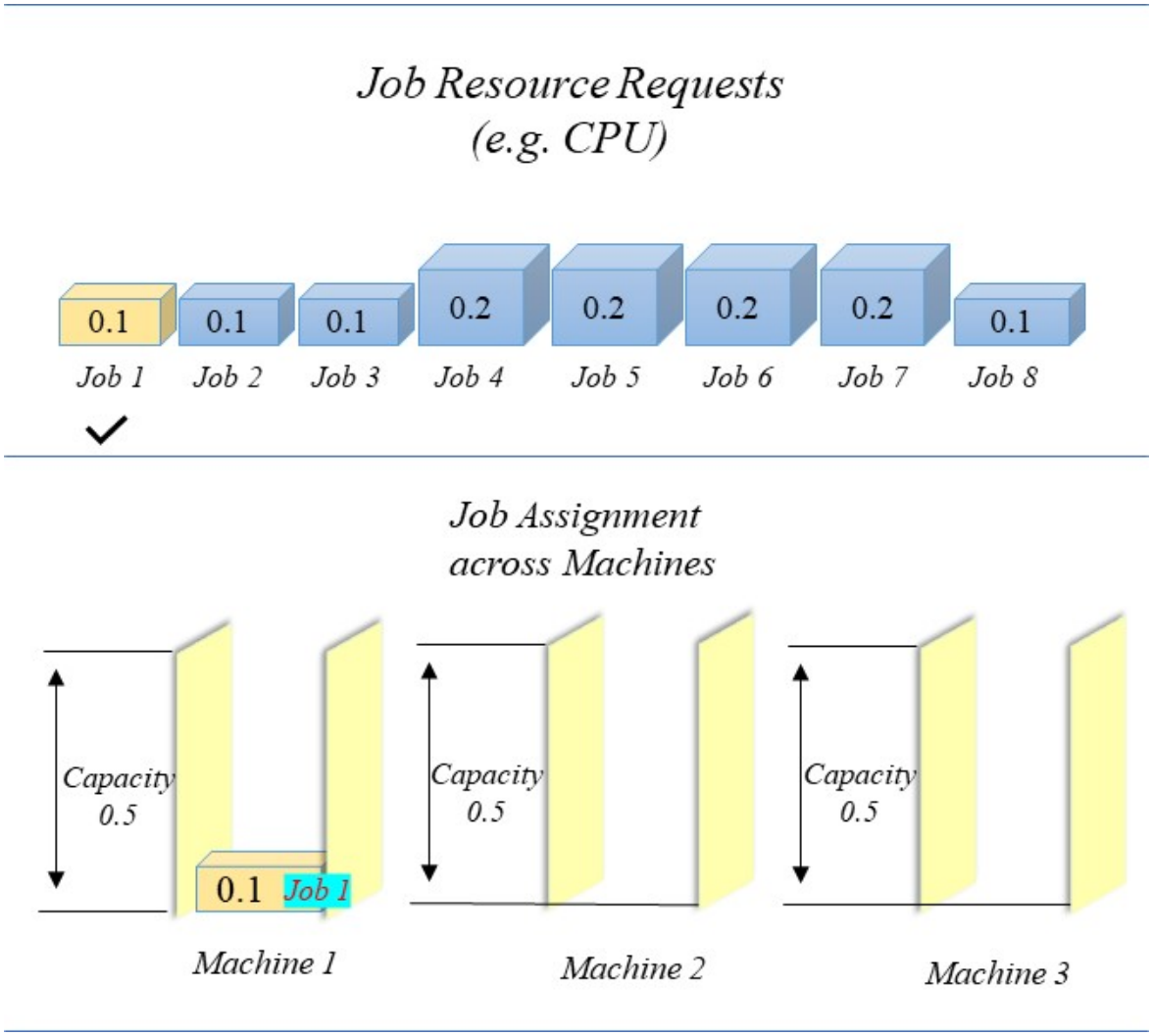

Figure 3-13: System State after the first job (Job 1) is scheduled

When Job 2 arrives, it requests 0.1 units of resource from the scheduler. Because any of the 3 available machines have sufficient resources to process this job the scheduler can choose to fit the job on any of the machines. However, because the scheduler is following a First-Fit bin packing approach, it chooses the first available machine with commensurate capacity. Figure 3-14 depicts the scenario after Job 2 has been scheduled.

Now Machine 1 has residual capacity of 0.3 units while the other 2 machines are still unutilized.

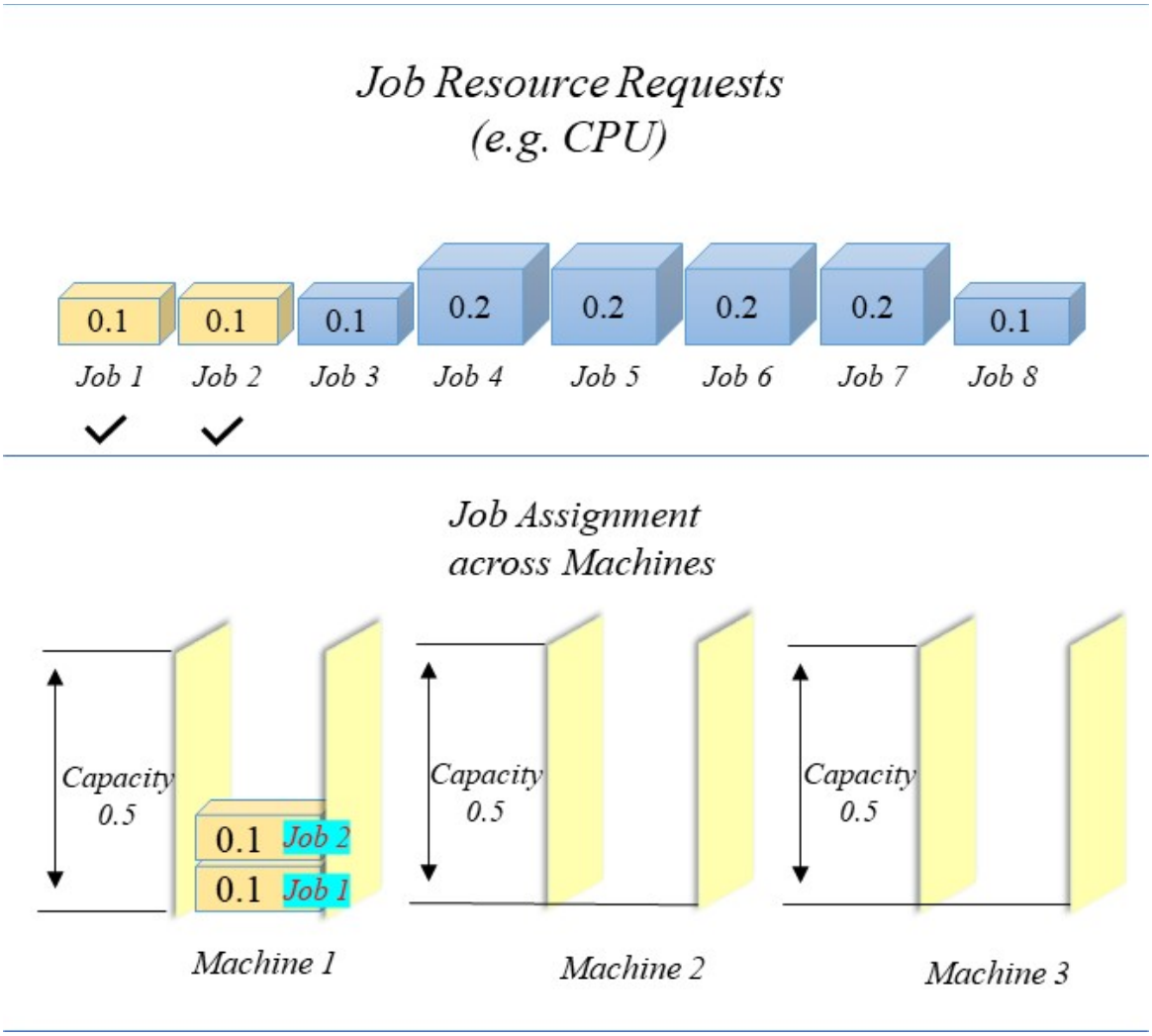

Figure 3-14: System state after the first 2 jobs (Job 1 and Job 2) are scheduled

Figure 3-15 shows the system state after the first 5 jobs (Job 1 – 5) have arrived and have been allocated resources on the machines. Note that Job 5 gets placed on Machine 2 because Machine 1 does not have any available resource capacity and

Machine 3 is scanned after Machine 2 (and Machine 1) do not have sufficient resources to schedule the incoming job.

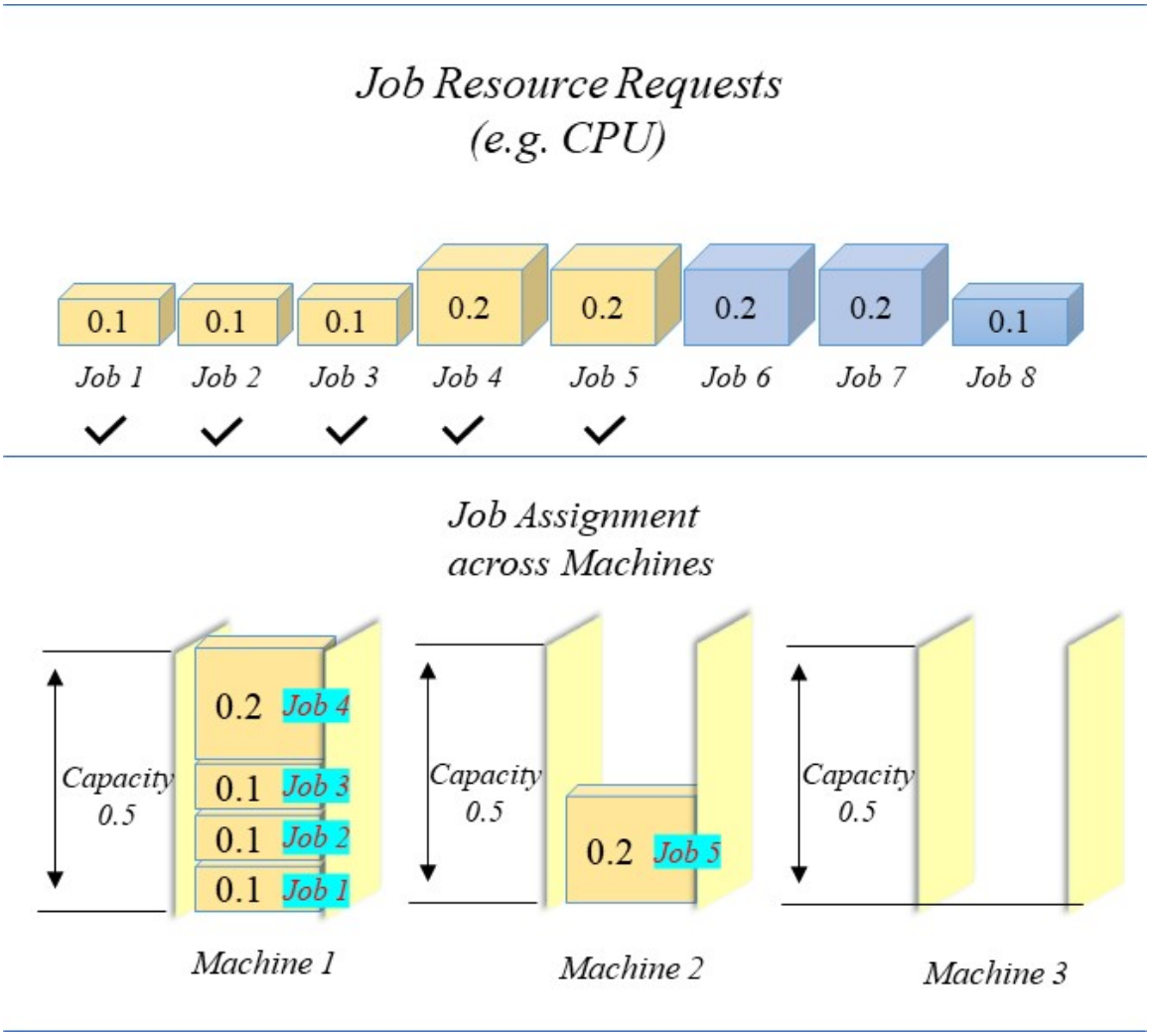

Figure 3-15: System state after the first 5 jobs (Job 1 - Job 5) are scheduled

Figure 3-16 shows the system state after all 8 jobs have been scheduled. Because the first-fit bin packing approach, new machines are utilized only if other machines that are currently under use have no capacity. For e.g., when Job 7 arrives requesting 0.2 resource units, although Machine 2 has free capacity of 0.1 units it is not sufficient to

accommodate the 0.2 units that Job 7 needs. Therefore, Job 7 needs to be scheduled on a new machine – Machine 3 leaving 0.1 units of unused capacity on Machine 2.

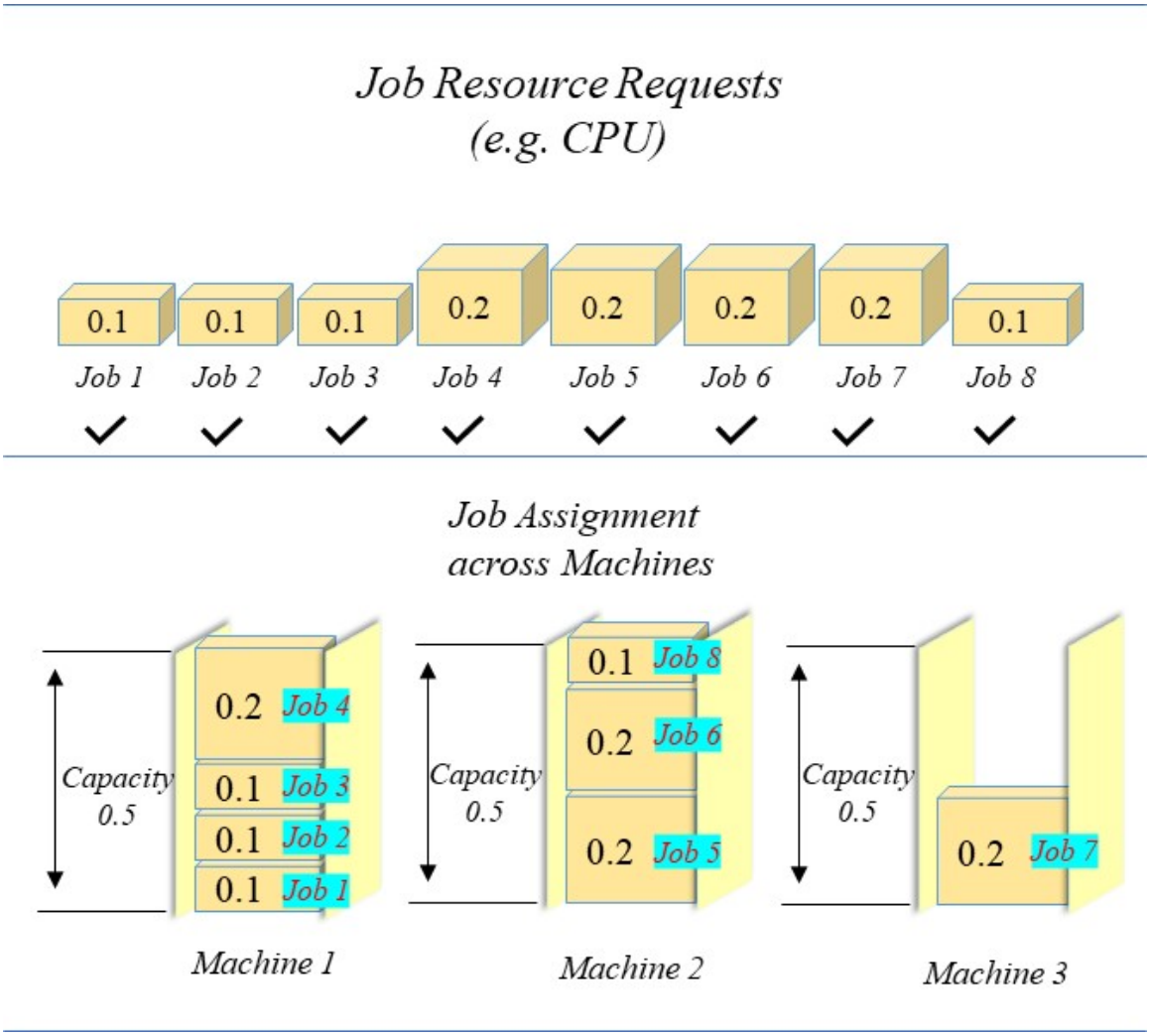

Figure 3-16: System state after all the jobs are scheduled

When Job 8 finally arrives requesting 0.1 units of the resource, the scheduler finds that all the machines are actively processing jobs. Applying the First-Fit approach, the scheduler finds the first available machine with sufficient capacity – Machine 2 and schedules Job 8 on it even though Machine 3 has sufficient capacity to process Job 8 as

well and the previous job (Job 7) was placed on Machine 3. This bin packing approach ensures that new machines are utilized only when all 'active' machines (i.e., machines that are currently processing tasks) have no available capacity. We assume that the jobs run-times are sufficiently long that no job completes before all 8 have arrived and made a scheduling request.

3.10. Other important scheduling policy considerations

The following scheduling policy assumptions were made in the simulations:

- In production environments, end users often terminate jobs (both batch and real-time) (Tirmazi, et al., 2020). These jobs are then resubmitted to the scheduler and then get treated as any other new job based on the SLA commitments. User induced state transitions were not evaluated.
- All jobs are assumed to be real-time.
- Jobs are scheduled on a first come first served basis (i.e., all jobs are assumed to have the same priorities and the scheduler does not give preferential treatment to any job).
- Jobs do not exceed their requested Memory and CPU allocation. In a production environment, jobs could exceed their requested allocation and the scheduler might continue running them if there is available free resource capacity.
- Evicted jobs are not rescheduled again.

- CPU and Memory are overcommitted by the same factor. Both these resources could be overcommitted at different rates depending on the scheduling policy that is being utilized.
- A machine's CPU capacity is assumed to be fixed i.e., CPUs are not overclocked.

Chapter 4—Results

4.1. Descriptive Characteristics of the Simulations

This section describes the key characteristics of the simulations that evaluate the validity of the parameters.

Figure 4- 1 shows a distribution profile of job creation frequencies across the simulations for both the machine distribution types.

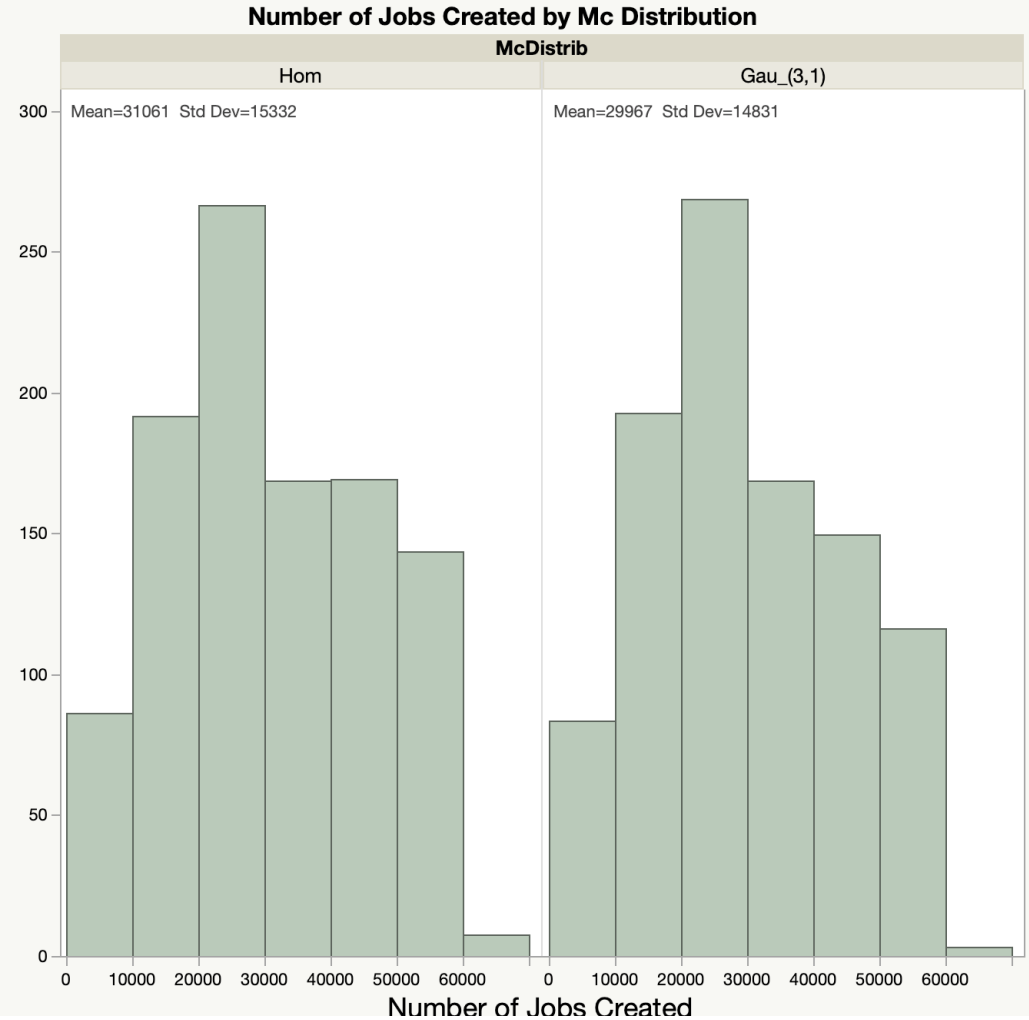

Figure 4- 1: Job creation histogram for each machine distribution type

The distribution of job creation across both the machine types is almost identical and the means and standard deviations (shown in the Figure) are similar as well. To ensure that the simulations are not biased and that they cover an appropriate range of values. Figure 4- 2 evaluates the distribution of overcommitment rate and the job arrival rates across both the machine heterogeneities.

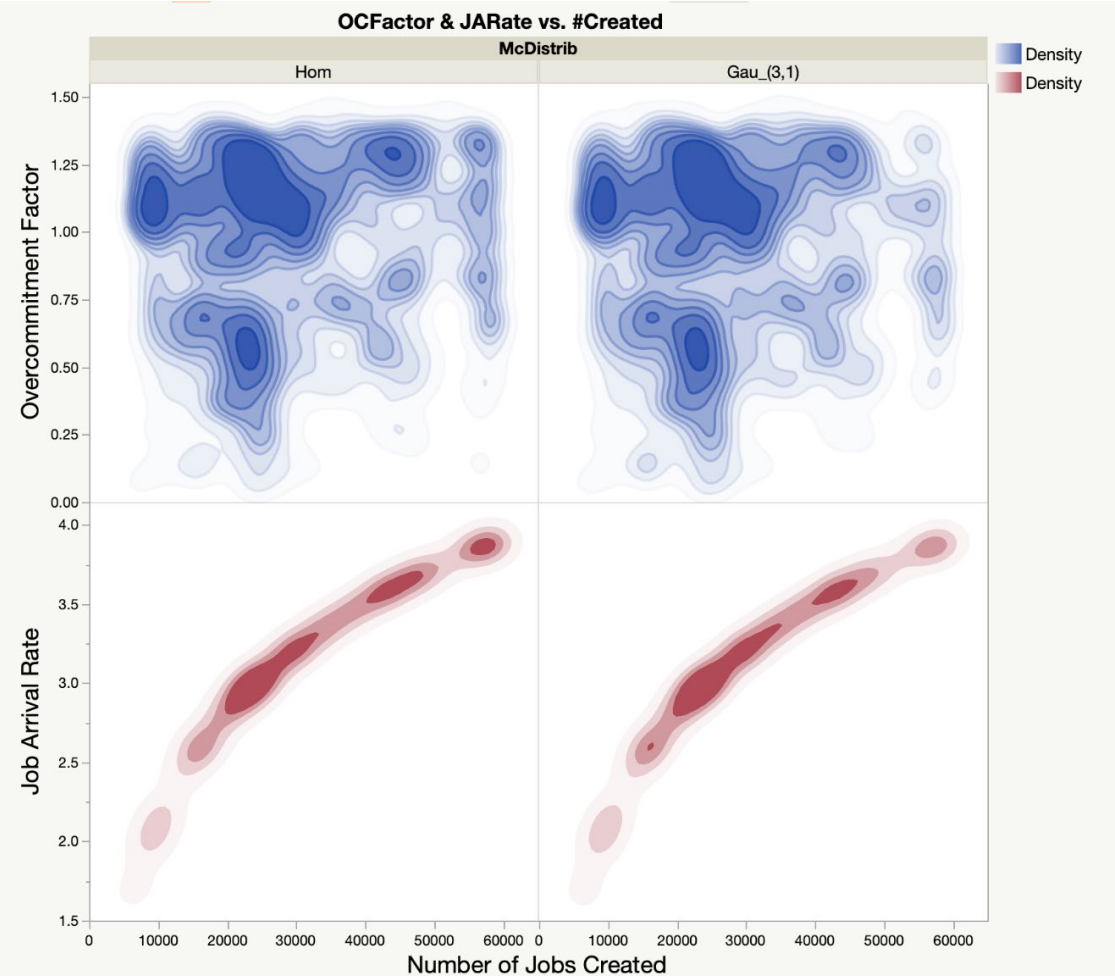

Figure 4- 2: Job arrival rate and overcommitment rate by number of jobs created for each machine distribution type

Note that the Job Arrival Rates and Overcommitment rates are natural log transformed for ease of visualization. The figure demonstrates that, just like the job

creation frequencies, overcommitment rates and job arrival rates that were sampled for each simulation are the same across both machine distribution types.

Figure 4- 2 also indicates that as the number of jobs created increases exponentially as the job arrival rates increase. This is what we would expect given we are modeling an exponential arrival rate between subsequent jobs. This indicates that the simulations are being conducted identically for both machine distribution profiles.

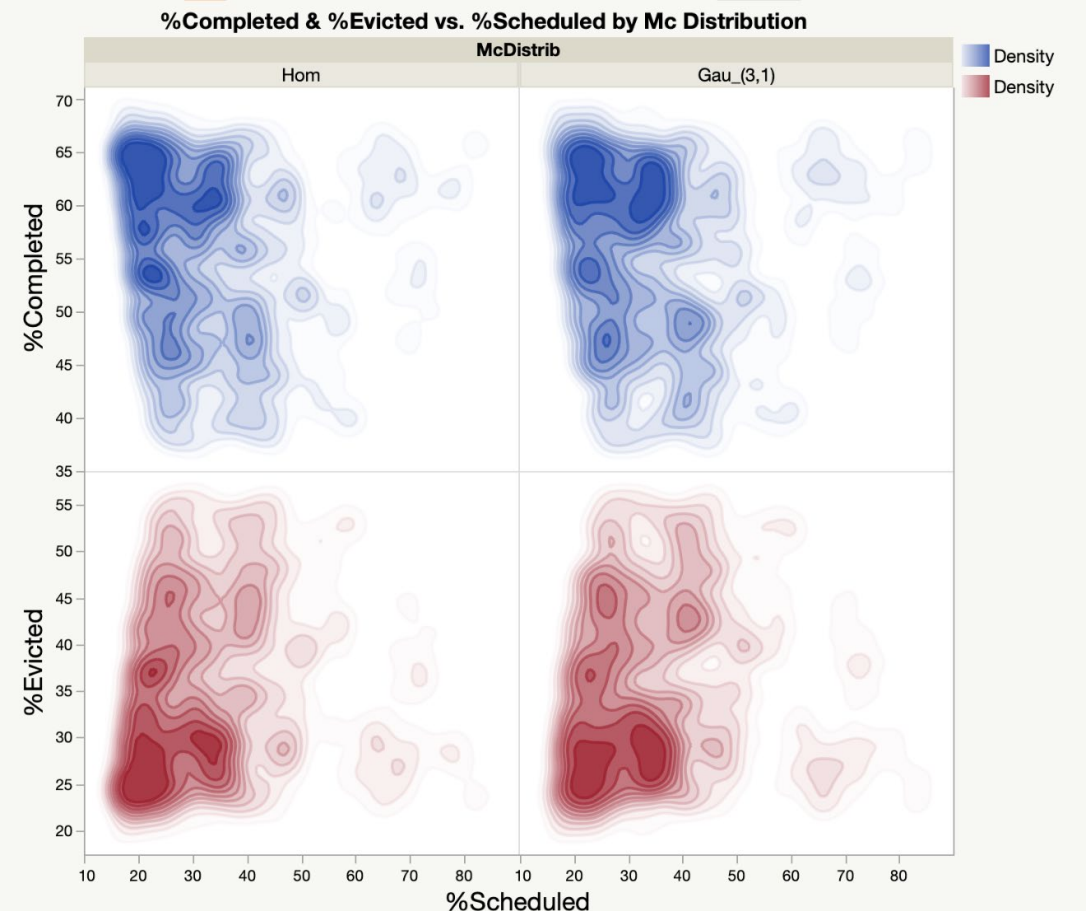

Figure 4- 3: Density of % Jobs Completed and % Jobs Evicted for each machine distribution type

Once the simulations are underway the percentage of jobs completed and evicted, as a fraction of scheduled, are dependent on how the jobs interact with the environment (machines in the datacenter) and the scheduling policy decisions (bin-packing, evictions in case of overutilization etc.). Although we are varying the overcommitment rates and job arrival rates similarly, we should expect the percentage completion and eviction profiles to be different for both machine heterogeneities. This is shown in Figure 4- 3.

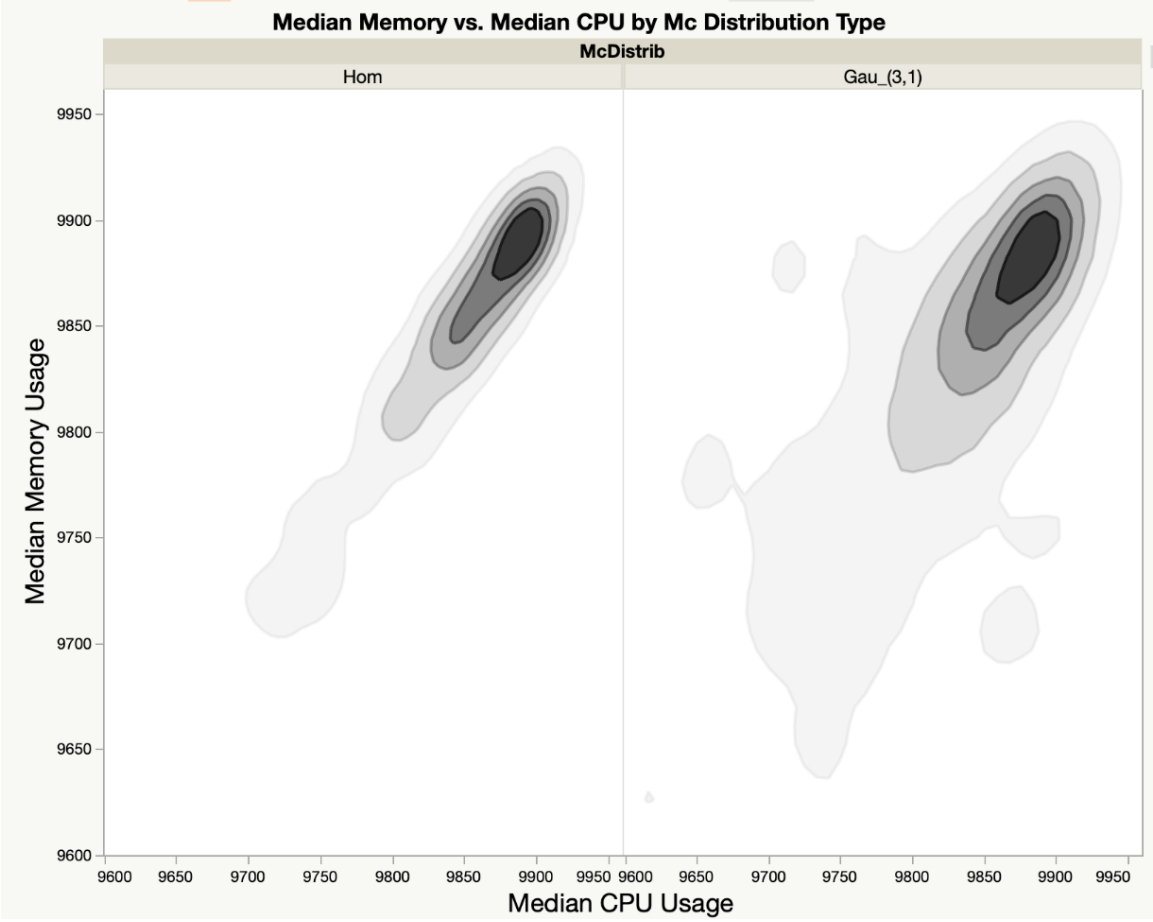

Figure 4- 4: Density of Median Memory vs. Median CPU Usage across simulations for each machine distribution type

Figure 4- 4 shows the relationship between median Memory and median CPU usage for both the machine heterogeneities. For each simulation, the median memory and median CPU usage for the entire duration is calculated and the relationship between the two is plotted on this chart. The darker areas represent a higher density of simulations that achieve that combination of median CPU/Memory utilization and vice versa. The Gaussian cluster achieves more darker contours thus visually indicating that it is achieving a higher resource utilization than the homogenous cluster on the left.

We also evaluate the Coefficient of Variation (CV) for Memory and CPU Usage. The coefficient of variation is defined as the ratio of the standard deviation to the mean of the variable in question. It is unitless and explains the variability in relation to the mean of the measure in question (CPU or Memory).

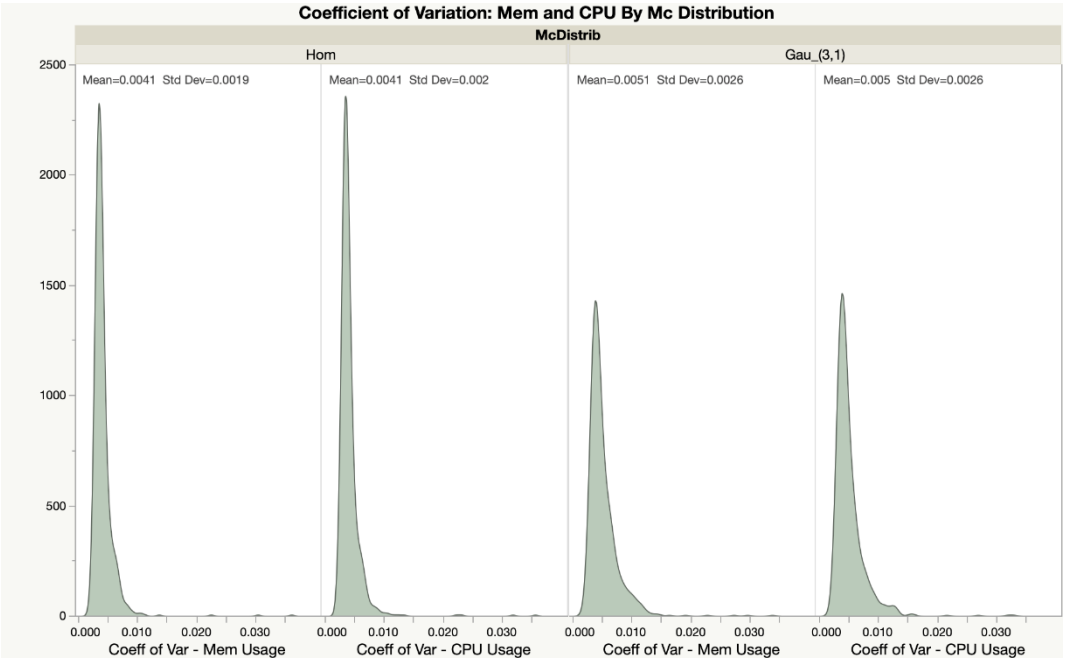

Figure 4- 5: Histograms of the Coefficient of Variation (CV) for Memory and CPU Usage across simulations for each machine distribution type

The histograms of the CVs for Memory and CPU utilization for each of the machine distribution types are shown in Figure 4- 5. The CV histograms in Figure 4- 5 indicate that dispersion frequencies of Memory and CPU usage is similar within the machine distribution. However, the dispersion frequencies are quite different when compared across the machine distribution types. These figures help us understand that the simulations are performing as we would expect – i.e., they are holding the simulation parameters constant but are able to get varying results for the parameters of the study.

Next, we study the model results including the effect summaries and the prediction equation that address the research questions and hypotheses described in Section 1.6.

4.2. Evaluation Models for Memory and CPU Utilization

The first two research questions of interest are if the heterogenous cluster improves machine utilization (Memory and CPU) compared the to the homogenous cluster. Correspondingly, two hypotheses - H_1 and H_2, were posed. H_1 formulated that for constant job arrival, eviction and overcommitment rates, heterogenous clusters improve median memory utilization. H_2 formulated the same for and median CPU utilization. The results (detailed in this section below) show that for constant job arrival, eviction and overcommitment rates the heterogenous cluster improves median memory and CPU utilization compared to the homogenous cluster. This satisfies H_1 and H_2, respectively.

We noted in Section 4.1 that the simulations across the two machine distributions behave differently from each other. However, a simulation within the distribution type is like another within the same machine distribution type. i.e., the distribution type seems to

be impacting the results. Thus, we need to account for the impact of machine distribution type on CPU and Memory utilization across all the simulations while controlling for other variables (like eviction rates). Hierarchical linear models were used because they allow us to leverage the variances that are shared by observations of related data (Woltman, Feldstain, MacKay, & Rocchi, 2012).

Two independent models were built to understand CPU and Memory Utilization. Both models used Median Memory (or CPU) as the target while Job Arrival rates, % Job Evictions, Overcommitment rate and Machine Distribution were predictors. The continuous variables were nested within the two levels of the machine distribution types to study the effect of machine distribution types for different levels of job arrival and overcommitment rates. To control for any training biases both models were cross-validated with 75% data used for training purposes and 25% held out for cross-validation.

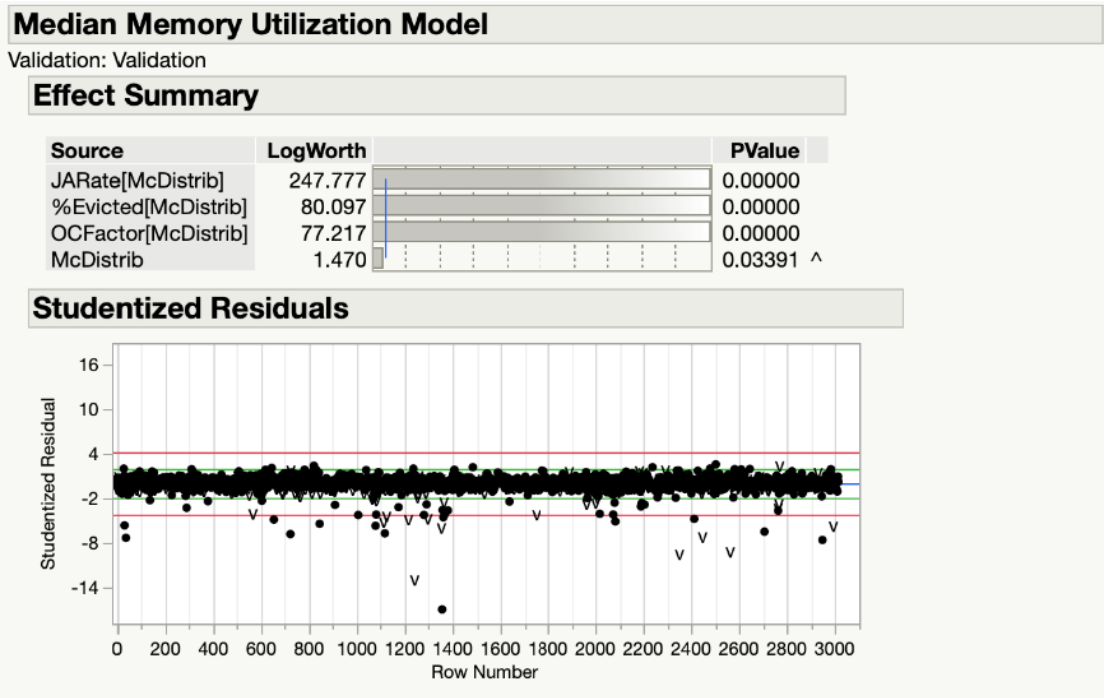

Figure 4- 6: Effect Summary for the Median Memory Utilization Model

We first look at the overall effect summary for the median memory model in Figure 4- 6. The following variable names are reflected in the model

- JARate = Job Arrival Rate used in the simulation
- OCFactor = Overcommitment Rate used in the simulation
- McDistrib = Machine Distribution used in the simulation
- %Evicted = % of Jobs Evicted in the simulation
- %Scheduled = % of Jobs Scheduled in the simulation

The p-values of the effect summary show that all parameters (job arrival rate, overcommitment rate, % evictions and machine distribution type) are significant. Secondly, the studentized residuals do not exhibit any pattern and seem to be normally distributed around a 0 mean.

Figure 4- 7 shows a Summary of Fit for the model. The overall model has a 0.81 Adjusted-R^2 and the F Ratio is significant for the Analysis of Variance indicating that the overall model has significant predictive capability. The cross-validation results show a 0.73 Adjusted-R^2 on the validation dataset which is a bit lower than on the training data but still significant for the overall model . The Parameter Estimates table shows all the parameter coefficient estimates including the upper and lower 95% confidence levels. The confidence limits do not include 0 and the p-values indicated by Prob >|t| are significant indicating the predictor variables that are contributing to the model fit. Furthermore, the effect tests show that all the predictor variables included in the model are significant. These results indicate that the job arrival rate, overcommitment rate, % evictions and machine heterogeneity can be used to predict the median Memory usage effectively.

Summary of Fit

RSquare	0.811953
RSquare Adj	0.811096
Root Mean Square Error	30.06213
Mean of Response	9841.683
Observations (or Sum Wgts)	1543

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	7	5989810.4	855687	946.8375
Error	1535	1387228.4	904	Prob > F
C. Total	1542	7377038.7		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|	Lower 95%	Upper 95%
Intercept	9841.7486	0.770248	12777	<.0001*	9840.2378	9843.2595
McDistrib[Hom]:(OCFactor-0.85834)	360.36053	37.122	9.71	<.0001*	287.54533	433.17573
McDistrib[Gau_(3,1)]:(OCFactor-0.85834)	580.86668	33.2099	17.49	<.0001*	515.7251	646.00826
McDistrib[Hom]:(%Evicted-35.7912)	15.106029	1.505738	10.03	<.0001*	12.152508	18.05955
McDistrib[Gau_(3,1)]:(%Evicted-35.7912)	24.082731	1.354102	17.79	<.0001*	21.426644	26.738817
McDistrib[Hom]:(JARate-3.08945)	79.467344	2.632001	30.19	<.0001*	74.304646	84.630043
McDistrib[Gau_(3,1)]:(JARate-3.08945)	75.575649	2.703481	27.95	<.0001*	70.272743	80.878555
McDistrib[Hom]	1.6352463	0.770248	2.12	0.0339*	0.1243973	3.1460954

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
OCFactor[McDistrib]	2	2	361639.0	200.0809	<.0001*
%Evicted[McDistrib]	2	2	376814.5	208.4770	<.0001*
JARate[McDistrib]	2	2	1530091.0	846.5404	<.0001*
McDistrib	1	1	4073.3	4.5072	0.0339*

Crossvalidation

Source	RSquare	RASE	Freq
Training Set	0.8120	29.984	1543
Validation Set	0.7343	37.943	466

Figure 4- 7: Summary of Fit, Analysis of Variance and Parameter Estimates and Effect Tests for the Median Memory Utilization Model

The prediction equation for the overall median memory utilization model is shown in Figure 4- 8. The “Uno3” machine distribution level represents the homogenous machine cluster while the “Gau3_1” machine distribution level represents the heterogenous gaussian distribution. The prediction expression represents the hierarchical

model design where the continuous predictors are nested within the two machine heterogeneity levels. This design and the magnitude of the coefficients allows us to understand the effects of change in one variable while holding all the others constant.

Prediction Expression

$$
\begin{aligned}
&9841.7486326 \\
&+\text{Match}(\text{McDistrib})\begin{pmatrix} \text{"Uno3"} & \Rightarrow (\text{OCFactor} - 0.8583441015) \bullet 360.3605304 \\ \text{"Gau3_1"} & \Rightarrow (\text{OCFactor} - 0.8583441015) \bullet 580.86667886 \\ \text{else} & \Rightarrow . \end{pmatrix} \\
&+\text{Match}(\text{McDistrib})\begin{pmatrix} \text{"Uno3"} & \Rightarrow (\%\text{Evicted} - 35.791170632) \bullet 15.106028878 \\ \text{"Gau3_1"} & \Rightarrow (\%\text{Evicted} - 35.791170632) \bullet 24.082730769 \\ \text{else} & \Rightarrow . \end{pmatrix} \\
&+\text{Match}(\text{McDistrib})\begin{pmatrix} \text{"Uno3"} & \Rightarrow (\text{JARate} - 3.0894505226) \bullet 79.467344132 \\ \text{"Gau3_1"} & \Rightarrow (\text{JARate} - 3.0894505226) \bullet 75.575648833 \\ \text{else} & \Rightarrow . \end{pmatrix} \\
&+\text{Match}(\text{McDistrib})\begin{pmatrix} \text{"Uno3"} & \Rightarrow 1.6352463329 \\ \text{"Gau3_1"} & \Rightarrow -1.635246333 \\ \text{else} & \Rightarrow . \end{pmatrix}
\end{aligned}
$$

Figure 4- 8: Prediction Expression for the Median Memory Utilization Model

The first set of coefficients for the equation pertain to the overcommitment rate. At a similar overcommitment rate the homogenous cluster has a smaller coefficient

(360.3605) than the heterogenous cluster (580.8667) indicating that the heterogenous cluster achieves a higher median memory utilization at a similar overcommitment rate while eviction rates and job arrival rates are held constant.

Similar results were observed for the Median CPU Utilization Model as well.

Figure 4- 9 shows the Effect summary for the model where all predictors are significant and the studentized residuals are randomly distributed around a 0 mean. Figure 4- 10 shows the Model Fit Summary with an adjusted-R^2 of 0.7625 on the Training data and 0.7616 on the Validation data. The Parameter Estimate indicate that all the nested variables at all levels are significant, and the 95% confidence limits do not include 0.

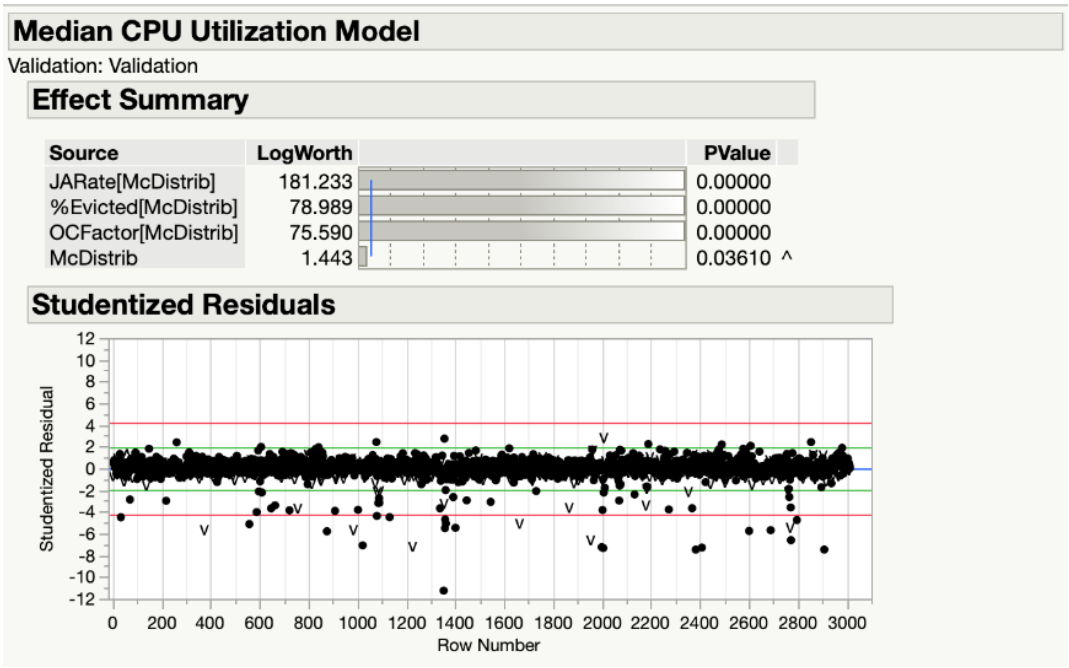

Figure 4- 9: Effect Summary for the Median CPU Utilization Model

Summary of Fit

RSquare	0.763662
RSquare Adj	0.762584
Root Mean Square Error	35.83952
Mean of Response	9840.168
Observations (or Sum Wgts)	1543

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	7	6370883.5	910126	708.5609
Error	1535	1971663.6	1284	Prob > F
C. Total	1542	8342547.1		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|	Lower 95%	Upper 95%
Intercept	9840.297	0.918275	10716	<.0001*	9838.4958	9842.0982
McDistrib[Hom]:(OCFactor-0.85834)	425.65589	44.25617	9.62	<.0001*	338.84695	512.46484
McDistrib[Gau_(3,1)]:(OCFactor-0.85834)	683.72974	39.59224	17.27	<.0001*	606.06915	761.39033
McDistrib[Hom]:(%Evicted-35.7912)	17.76033	1.795113	9.89	<.0001*	14.239196	21.281463
McDistrib[Gau_(3,1)]:(%Evicted-35.7912)	28.54169	1.614336	17.68	<.0001*	25.375153	31.708228
McDistrib[Hom]:(JARate-3.08945)	78.53496	3.137824	25.03	<.0001*	72.380086	84.689834
McDistrib[Gau_(3,1)]:(JARate-3.08945)	70.794231	3.22304	21.97	<.0001*	64.472204	77.116259
McDistrib[Hom]	1.9262389	0.918275	2.10	0.0361*	0.1250325	3.7274454

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
OCFactor[McDistrib]	2	2	501887.0	195.3671	<.0001*
%Evicted[McDistrib]	2	2	527240.7	205.2365	<.0001*
JARate[McDistrib]	2	2	1424335.3	554.4441	<.0001*
McDistrib	1	1	5652.0	4.4002	0.0361*

Crossvalidation

Source	RSquare	RASE	Freq
Training Set	0.7637	35.746	1543
Validation Set	0.7616	33.682	466

Figure 4- 10: Summary of Fit, Analysis of Variance and Parameter Estimates and Effect Tests for the Median CPU Utilization Model

Like the Median Memory Model, the effect tests demonstrate that the predictor variables included in the CPU model are significant as well. Hence, the job arrival rate, overcommitment rate, % evictions and machine heterogeneity can be used to predict the median CPU usage effectively. Finally, Figure 4- 11 shows the Prediction Expression for the Median CPU Model where the coefficient of the nested Overcommitment rate for the Homogenous cluster ("Uno3") is lower (425.65598) than the coefficient for the

Heterogenous cluster (“Gau3_1”) indicating that the Gaussian Heterogenous cluster achieves improved CPU Utilization compared to the Homogenous cluster for similar overcommitment rates while job arrival rates and % of evictions are held constant.

Prediction Expression

$$
\begin{aligned}
&9840.2969649 \\
&+\text{Match}\left(\text{McDistrib}\right)\begin{pmatrix}\text{"Uno3"} & \Rightarrow \left(\text{OCFactor} - 0.8583441015\right)\cdot 425.655894 \\ \text{"Gau3_1"} & \Rightarrow \left(\text{OCFactor} - 0.8583441015\right)\cdot 683.72973738 \\ \text{else} & \Rightarrow .\end{pmatrix} \\
&+\text{Match}\left(\text{McDistrib}\right)\begin{pmatrix}\text{"Uno3"} & \Rightarrow \left(\%\text{Evicted} - 35.791170632\right)\cdot 17.760329957 \\ \text{"Gau3_1"} & \Rightarrow \left(\%\text{Evicted} - 35.791170632\right)\cdot 28.541690191 \\ \text{else} & \Rightarrow .\end{pmatrix} \\
&+\text{Match}\left(\text{McDistrib}\right)\begin{pmatrix}\text{"Uno3"} & \Rightarrow \left(\text{JARate} - 3.0894505226\right)\cdot 78.534959695 \\ \text{"Gau3_1"} & \Rightarrow \left(\text{JARate} - 3.0894505226\right)\cdot 70.794231239 \\ \text{else} & \Rightarrow .\end{pmatrix} \\
&+\text{Match}\left(\text{McDistrib}\right)\begin{pmatrix}\text{"Uno3"} & \Rightarrow 1.9262389418 \\ \text{"Gau3_1"} & \Rightarrow -1.926238942 \\ \text{else} & \Rightarrow .\end{pmatrix}
\end{aligned}
$$

Figure 4- 11: Prediction Expression for the Median CPU Utilization Model

4.3. Evaluation Model for Job Eviction Rates

We next address the third research question and the corresponding hypothesis noted in Section 1.6.. H_3 formulated that eviction rate prediction can be improved by incorporating an interaction between job arrival rates and overcommitment rates. To model this, besides the main effects of job arrival rate and overcommitment rate, an interaction effect of overcommitment and job arrivals was included. % jobs scheduled was included to control for the job scheduling rate.

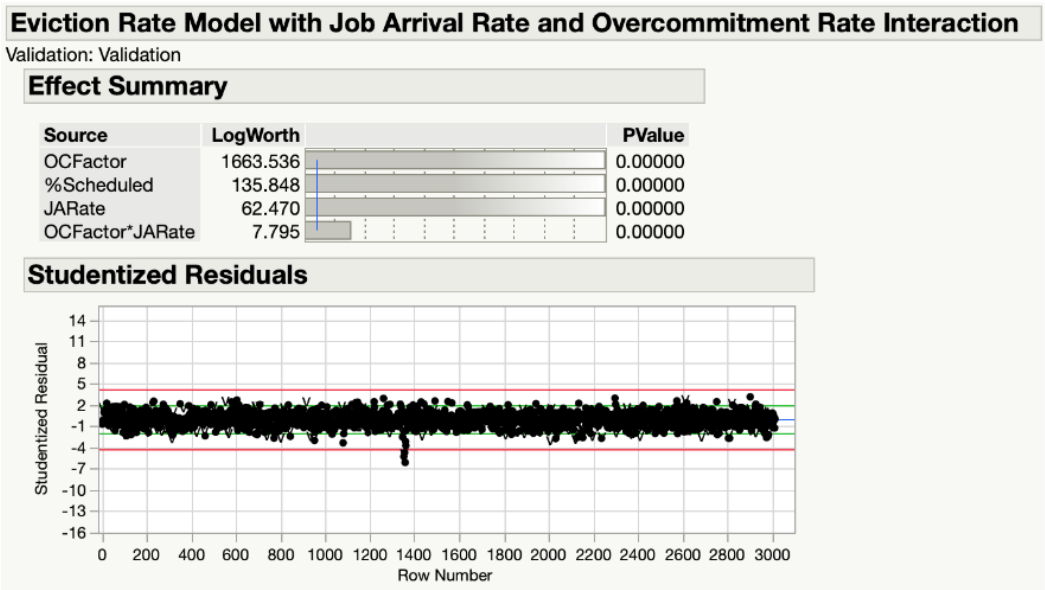

Figure 4- 12: Effect Summary for the Eviction Rate Model with Job Arrival Rate and Overcommitment Rate Interaction

Figure 4- 12 shows that the effect summary for the model has all significant effects and overcommitment rate has the highest LogWorth. The studentized residuals are normally distributed around a 0 mean. Figure 4- 13 shows a Summary of Fit with all significant parameter coefficient estimates and predictor p-values (Prob >|t|). The model

has a 0.99 Adjusted-R^2 with a significant F Ratio for the Analysis of Variance indicating significant predictive capability for the overall model.

Summary of Fit

RSquare	0.995968
RSquare Adj	0.995958
Root Mean Square Error	0.607364
Mean of Response	35.70626
Observations (or Sum Wgts)	1543

Analysis of Variance

Source	DF	Sum of Squares	Mean Square	F Ratio
Model	4	140151.86	35038.0	94981.92
Error	1538	567.35	0.368891	Prob > F
C. Total	1542	140719.22		<.0001*

Parameter Estimates

Term	Estimate	Std Error	t Ratio	Prob>|t|	Lower 95%	Upper 95%
Intercept	70.487783	0.65176	108.15	<.0001*	69.209351	71.766216
OCFactor	-25.5231	0.054254	-470.4	<.0001*	-25.62952	-25.41668
JARate	-2.482646	0.141206	-17.58	<.0001*	-2.759622	-2.20567
(OCFactor-0.85834)*(JARate-3.08945)	-0.406385	0.071543	-5.68	<.0001*	-0.546716	-0.266053
%Scheduled	-0.135972	0.004926	-27.60	<.0001*	-0.145634	-0.12631

Effect Tests

Source	Nparm	DF	Sum of Squares	F Ratio	Prob > F
OCFactor	1	1	81638.308	221307.5	<.0001*
JARate	1	1	114.031	309.1189	<.0001*
OCFactor*JARate	1	1	11.903	32.2660	<.0001*
%Scheduled	1	1	281.086	761.9760	<.0001*

Crossvalidation

Source	RSquare	RASE	Freq
Training Set	0.9960	0.60638	1543
Validation Set	0.9959	0.60820	466

Figure 4- 13: Summary of Fit, Analysis of Variance, Parameter Estimates, Effect Tests and Cross validation results for the Eviction Rate Model

Figure 4- 14 shows the prediction equation with the coefficients of all the predictors including the interaction factor and Figure 4- 15 shows the contours of the % Completion rate against overcommitment rate and Job arrival rate changes at a fixed Scheduling rate of 37%.

Prediction Expression

70.487783391

+ -25.52310091 • OCFactor

+ -2.482645793 • JARate

+ (OCFactor − 0.8583441015) • ((JARate − 3.0894505226) • -0.406384597)

+ -0.135971803 • %Scheduled

Figure 4- 14: Prediction Expression for the Eviction Rate Model

The contour plot visually exhibits the existence of interaction between the two terms. Different job arrival and overcommitment rates achieve the same completion rate while scheduling rate is held constant. The visual contour plot representation, in addition to the significance of the interaction term in the model, satisfies H_3.

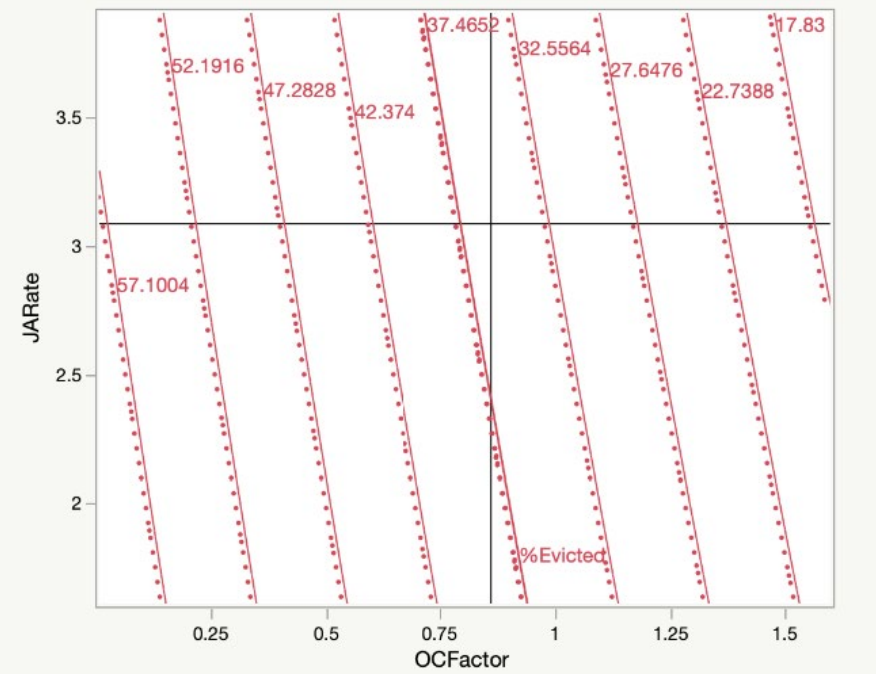

Figure 4- 15: Contour Profile for the Eviction Rate Model

4.4. Summary

The cross-validated model residuals do not exhibit any patterns and the effect sizes, and the F-tests are significant. This indicates that a regression model is an appropriate explainable predictive model to understand the relationships between overcommitment rates, job arrival rates, machine heterogeneity and machine utilization. The overall models described in Section 4.2 and the prediction expressions described in Figure 4- 8 and Figure 4- 11 address the fourth research question posed in Section 1.6. H_4 is thus satisfied indicating that CPU and memory utilization can be predicted using job arrival, job eviction and overcommitment rates.

As measured by the weights of the parameter coefficients in the median memory and CPU utilization model, the heterogenous cluster achieves a 2.16% improvement in Memory and 2.48% improvement in CPU utilization over the homogenous cluster when job arrival rates, eviction rates and overcommitment rates are held constant. Although the magnitude of the improvement is not a focus of the research, the ability of the heterogenous cluster to outperform the homogenous cluster for both CPU and Memory is the important finding.

Finally, we observe that eviction rates can be effectively predicted by a model that includes an interaction term of overcommitment rates and job arrival rates. This indicates that the two interact with each other to affect eviction rates when scheduling rates are held constant.

Chapter 5—Conclusions and Future Work

5.1 Conclusions

Given the massive investments CSPs must make in their datacenters, their ability to maintain SLOs without incurring penalties is key to achieving profitability, maintaining a competitive advantage, and recouping their investments. Upgrading their hardware (servers and network) is seldom an option for achieving improved performance compliance with QoS guarantees (Sahlin & Mazzuchi, 2012). Hence, understanding workload characteristics is essential as CSPs are challenged with improving utilization of deployed hardware using overcommitment as a important scheduling policy option.

Although availability of trace data from Google, Alibaba, Yahoo etc. has made research in the areas of understanding computing workloads possible, these traces lack several proprietary scheduling policy attributes like bin-packing approach and overcommitment rates. So, researchers are left with simulating workloads using physical hardware that is time-consuming, cost-prohibitive, and too varied to be appropriately configured in a physical environment (Heavey, et al., 2014).

This analysis shows that once scheduling policy decisions are set, generative discrete event simulations are an effective tool for simulating an overcommitment datacenter environment with different machine heterogeneities. The results of the CPU and Memory utilization models indicate that if job arrival rates and eviction rates are held constant then as the overcommitment rate increases, the heterogenous cluster with a Gaussian distribution of machine capacities achieves a higher utilization for both CPU and Memory compared to the homogenous cluster where all machine capacities

(CPU/Memory) are the same. Furthermore, this research demonstrates that if scheduling rates are held constant then job arrival rates and overcommitment rates interact with each other to affect job eviction rates. Thus, for a fixed scheduling rate achieved by a datacenter a CSP can use the interaction model to derive a likely eviction rate as job arrival rates and overcommitment rates are varied.

5.2 Research Contributions

This research demonstrates that although the publicly available cloud data traces (like Google Borg, Alibaba etc.) do not contain information about overcommitment rates or bin packing policies, the trace characteristics can be used to create a generative simulation model that accurately represents key interactions in a datacenter under specific scheduling policies.

The generative simulation modeling approach allows for creating cost-effective large-scale simulations that demonstrate that a heterogenous cluster outperforms a homogenous cluster in achieving higher CPU and memory usage. This is an important aspect for CSPs to understand as machines are constantly upgraded or replaced in a datacenter. Monitoring and managing machine heterogeneity are essential to achieving high utilization in an overcommitted datacenter while maintaining low eviction rates.

CSPs need to establish SLAs around acceptable eviction rates and make certain performance guarantees to clients based on their workloads. This research has generated interpretable linear models where the prediction expressions can be used as decision-making tools for setting overcommitment values that can achieve high utilization at fixed eviction rates while job arrival rates vary.

5.3.1 Recommendations for Future Research

This research relies heavily on the flexibility of the Python simpy framework that allows for modeling complex real-world interactions using discrete event simulations. Future research evaluations could focus on expanding this body of work into other variations of scheduling policies that include specific performance guarantees for different workloads.

Additionally, future research could evaluate the impact to utilization by simulating a mix of batch and real-time jobs with user-induced state transitions. This can be further extended to incorporate job Directed Acyclic Graphs (DAGs) and dependencies. Finally, the impact of overcommitment could be studied when alternative Bin-packing algorithms are used.

References

Alam, M., Shakil, K. A., & Sethi, S. (2016). Analysis and clustering of workload in google cluster trace based on resource usage. *2016 IEEE Intl conference on computational science and engineering (CSE) and IEEE Intl conference on embedded and ubiquitous computing (EUC) and 15th Intl symposium on distributed computing and applications for business engineering (DCABES)*, (pp. 740–747).

Alves, D. C., Batista, B. G., Filho, D. M., Peixoto, M. L., Reiff-Marganiec, S., & Kuehne, B. T. (2016). CM Cloud Simulator: A Cost Model Simulator Module for Cloudsim. *2016 IEEE World Congress on Services (SERVICES)*, (pp. 99-102). doi:10.1109/SERVICES.2016.20

Bang-Jensen, J., & Larsen, R. (2012). Efficient algorithms for real-life instances of the variable size bin packing problem. *Computers & Operations Research, 39*, 2848–2857.

Banks, J., Carson, J. S., Nelson, B. L., & Nicol, D. M. (2013). *Discrete-event system simulation: Pearson new international edition.* Pearson Higher Ed.

Barroso, L. A., & Hölzle, U. (2009). The datacenter as a computer: An introduction to the design of warehouse-scale machines. *Synthesis lectures on computer architecture, 4*, 1–108.

Bashir, N., Deng, N., Rządca, K. M., Irwin, D., Kodakara, S., & Jnagal, R. (2021). Take it to the Limit: Peak Prediction-driven Resource Overcommitment in Datacenters.

Bendechache, M., Svorobej, S., Endo, P. T., Mario, M. N., Ares, M. E., Byrne, J., & Lynn, T. (2019). Modelling and Simulation of ElasticSearch using CloudSim.

2019 IEEE/ACM 23rd International Symposium on Distributed Simulation and Real Time Applications (DS-RT), (pp. 1-8). doi:10.1109/DS-RT47707.2019.8958653

Burns, B., Grant, B., Oppenheimer, D., Brewer, E., & Wilkes, J. (2016). Borg, Omega, and Kubernetes: Lessons learned from three container-management systems over a decade. *Queue, 14*, 70–93.

Buyya, R., Ramamohanarao, K., Leckie, C., Calheiros, R. N., Dastjerdi, A. V., & Versteeg, S. (2015). Big data analytics-enhanced cloud computing: Challenges, architectural elements, and future directions. *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*, (pp. 75–84).

Buyya, R., Ranjan, R., & Calheiros, R. N. (2009). Modeling and simulation of scalable Cloud computing environments and the CloudSim toolkit: Challenges and opportunities. *2009 International Conference on High Performance Computing & Simulation* (pp. 1-11). IEEE.

Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A., & Buyya, R. (2011). CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and experience, 41*, 23–50.

Calzarossa, M. C., Massari, L., & Tessera, D. (2016). Workload characterization: A survey revisited. *ACM Computing Surveys (CSUR), 48*, 1–43.

Cao, Y., & Ro, C.-W. (2012). Dynamic channel allocation control with thresholds in wireless cellular networks using simpy. *International Journal of Contents, 8*, 19–22.

Carson, J. S. (2005). Introduction to modeling and simulation. *Proceedings of the Winter Simulation Conference, 2005.*, (pp. 8–pp).

Castillo, V. (2006). Parallel simulations of manufacturing processing using simpy, a python-based discrete event simulation tool. *2006 Winter Simulation Conference*, (pp. 2294–2294).

Chandini, M., Pushpalatha, R., & Boraia, R. (2016). A Brief study on Prediction of load in Cloud Environment. *International Journal of Advanced Research in Computer and Communication Engineering.–2016.–5 (5).–pp*, 157–162.

Chen, Q., Zhang, D., Guo, M., Deng, Q., & Guo, S. (2010). Samr: A self-adaptive mapreduce scheduling algorithm in heterogeneous environment. *2010 10th IEEE International Conference on Computer and Information Technology*, (pp. 2736–2743).

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

Cheng, Y., Anwar, A., & Duan, X. (2018). Analyzing alibaba's co-located datacenter workloads. *2018 IEEE International Conference on Big Data (Big Data)*, (pp. 292–297).

Coffman, E. G., Garey, M. R., & Johnson, D. S. (1984). Approximation algorithms for bin-packing—an updated survey. In *Algorithm design for computer system design* (pp. 49–106). Springer.

Cohen, M. C., Keller, P. W., Mirrokni, V., & Zadimoghaddam, M. (2019). Overcommitment in cloud services: Bin packing with chance constraints. *Management Science, 65*, 3255–3271.

Cortez, E., Bonde, A., Muzio, A., Russinovich, M., Fontoura, M., & Bianchini, R. (2017). Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. *Proceedings of the 26th Symposium on Operating Systems Principles*, (pp. 153–167).

Dabbagh, M., Hamdaoui, B., Guizani, M., & Rayes, A. (2015). Efficient datacenter resource utilization through cloud resource overcommitment. *2015 IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, (pp. 330–335).

Dabbagh, M., Hamdaoui, B., Rayes, A., & Guizani, M. (2016). An energy-efficient VM prediction and migration framework for overcommitted clouds. *IEEE Transactions on Cloud Computing, 6*, 955–966.

Dagkakis, G., Heavey, C., Robin, S., & Perrin, J. (2013). ManPy: An open-source layer of DES manufacturing objects implemented in SimPy. *2013 8th EUROSIM Congress on Modelling and Simulation*, (pp. 357–363).

Delorme, M., Iori, M., & Martello, S. (2016). Bin packing and cutting stock problems: Mathematical models and exact algorithms. *European Journal of Operational Research, 255*, 1–20.

Dezhabad, N., Ganti, S., & Shoja, G. (2019). Cloud Workload Characterization and Profiling for Resource Allocation. *2019 IEEE 8th International Conference on Cloud Networking (CloudNet)*, (pp. 1–4).

Durkee, D. (2010). Why cloud computing will never be free. *Communications of the ACM, 53*, 62–69.

Feitelson, D. G. (2015). *Workload modeling for computer systems performance evaluation.* Cambridge University Press.

Gandhi, A., Chen, Y., Gmach, D., Arlitt, M., & Marwah, M. (2011). Minimizing data center SLA violations and power consumption via hybrid resource provisioning. *2011 International Green Computing Conference and Workshops*, (pp. 1–8).

Gkatzikis, L., & Koutsopoulos, I. (2013). Migrate or not? Exploiting dynamic task migration in mobile cloud computing systems. *IEEE Wireless Communications, 20*, 24–32.

Guo, J., Chang, Z., Wang, S., Ding, H., Feng, Y., Mao, L., & Bao, Y. (2019). Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces. *2019 IEEE/ACM 27th International Symposium on Quality of Service (IWQoS)*, (pp. 1–10).

György, A., Lugosi, G., & Ottucsák, G. (2010). On-line sequential bin packing. *Journal of Machine Learning Research, 11*.

Hammoud, M., & Sakr, M. F. (2011). Locality-aware reduce task scheduling for MapReduce. *2011 IEEE Third International Conference on Cloud Computing Technology and Science*, (pp. 570–576).

Hasan, J., Kaabi, J., & Harrath, Y. (2019). Multi-objective 3D bin-packing problem. *2019 8th International Conference on Modeling Simulation and Applied Optimization (ICMSAO)*, (pp. 1–5).

Heavey, C., Dagkakis, G., Barlas, P., Papagiannopoulos, I., Robin, S., Mariani, M., & Perrin, J. (2014). Development of an open-source discrete event simulation cloud enabled platform. *Proceedings of the Winter Simulation Conference 2014*, (pp. 2824–2835).

Holden, L. (2017). Inventory Optimization Using a SimPy Simulation Model.

Huang, Q., Su, S., Li, J., Xu, P., Shuang, K., & Huang, X. (2012). Enhanced energy-efficient scheduling for parallel applications in cloud. *2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*, (pp. 781–786).

Humane, P., & Varshapriya, J. N. (2015). Simulation of cloud infrastructure using CloudSim simulator: A practical approach for researchers. *2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM)*, (pp. 207-211). doi:10.1109/ICSTM.2015.7225415

Idziorek, J. (2010). Discrete event simulation model for analysis of horizontal scaling in the cloud computing model. *Proceedings of the 2010 Winter Simulation Conference*, (pp. 3004–3014).

Isard, M., Prabhakaran, V., Currey, J., Wieder, U., Talwar, K., & Goldberg, A. (2009). Quincy: fair scheduling for distributed computing clusters. *Proceedings of the ACM SIGOPS 22nd symposium on Operating systems principles*, (pp. 261–276).

Ji, S., Da Li, M., Ji, N., & Li, B. (2018). An online virtual machine placement algorithm in an over-committed cloud. *2018 IEEE International Conference on Cloud Engineering (IC2E)*, (pp. 106–112).

Jia, R., Yang, Y., Grundy, J., Keung, J., & Li, H. (2019). A highly efficient data locality aware task scheduler for cloud-based systems. *2019 IEEE 12th International Conference on Cloud Computing (CLOUD)*, (pp. 496–498).

Jin, J., Luo, J., Song, A., Dong, F., & Xiong, R. (2011). Bar: An efficient data locality driven task scheduling algorithm for cloud computing. *2011 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing*, (pp. 295–304).

Jin, J.-H., Jun-zhou, L. U., Ai-bo, S. O., & Fang, D. O. (2011). Adaptive delay scheduling algorithm based on data center load analysis. *Journal on Communications, 32*, 47.

Johnson, D. S. (1973). *Near-optimal bin packing algorithms.* Ph.D. dissertation, Massachusetts Institute of Technology.

Johnson, D. S. (1974). Fast algorithms for bin packing. *Journal of Computer and System Sciences, 8*, 272–314.

Jung, J., & Kim, H. (2012). MR-CloudSim: Designing and implementing MapReduce computing model on CloudSim. *2012 International Conference on ICT Convergence (ICTC)*, (pp. 504-509). doi:10.1109/ICTC.2012.6387186

Karanjkar, N., Tejasvi, P. C., & Amrutur, B. (2019). A simpy-based simulation testbed for smart-city IoT applications. *Proceedings of the International Conference on Internet of Things Design and Implementation*, (pp. 273–274).

Karp, R. M. (1972). Reducibility among combinatorial problems. In *Complexity of computer computations* (pp. 85–103). Springer.

Kathiravelu, P., & Veiga, L. (2014). Concurrent and distributed cloudsim simulations. *2014 IEEE 22nd International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems*, (pp. 490–493).

Kavulya, S., Tan, J., Gandhi, R., & Narasimhan, P. (2010). An analysis of traces from a production mapreduce cluster. *2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing*, (pp. 94–103).

Kilcioglu, C., Rao, J. M., Kannan, A., & McAfee, R. P. (2017). Usage patterns and the economics of the public cloud. *Proceedings of the 26th International Conference on World Wide Web*, (pp. 83–91).

Konstantinos, K., Persefoni, M., Evangelia, F., Christos, M., & Mara, N. (2015). Cloud Computing and Economic Growth. *Proceedings of the 19th Panhellenic Conference on Informatics* (pp. 209–214). New York, NY, USA: Association for Computing Machinery. doi:10.1145/2801948.2802000

Law, A. M. (2007). *Simulation modeling and analysis* (4th ed. ed.). Boston: McGraw-Hill.

Lee, C. C., & Lee, D.-T. (1985). A simple on-line bin-packing algorithm. *Journal of the ACM (JACM), 32*, 562–572.

Lee, G., & Katz, R. H. (2011). Heterogeneity-Aware Resource Allocation and Scheduling in the Cloud. *HotCloud, 11*, 4–8.

Liu, H. (2011). A measurement study of server utilization in public clouds. *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*, (pp. 435–442).

Liu, Q., & Yu, Z. (2018). The elasticity and plasticity in semi-containerized co-locating cloud workload: a view from alibaba trace. *Proceedings of the ACM Symposium on Cloud Computing*, (pp. 347–360).

Ma, T., Chu, Y., Zhao, L., & Ankhbayar, O. (2014). Resource allocation and scheduling in cloud computing: policy and algorithm. *IETE Technical review, 31*, 4–16.

Magalhães, D., Calheiros, R. N., Buyya, R., & Gomes, D. G. (2015). Workload modeling for resource usage analysis and simulation in cloud computing. *Computers & Electrical Engineering, 47*, 69–81.

Martello, S., & Toth, P. (1990). Bin-packing problem. *Knapsack problems: Algorithms and computer implementations*, 221–245.

Matloff, N. (2008). Advanced features of the SimPy language. *Advanced Features of the SimPy Language.-2008*.

Matloff, N. (2008). Introduction to discrete-event simulation and the simpy language. *Davis, CA. Dept of Computer Science. University of California at Davis. Retrieved on August, 2*, 1–33.

McDougal, R. A., Bulanova, A. S., & Lytton, W. W. (2016). Reproducibility in computational neuroscience models and simulations. *IEEE Transactions on Biomedical Engineering, 63*, 2021–2035.

Min, O., Park, C., Lee, J., Cho, J., & Kim, H. (2011). Issues on supporting public cloud virtual machine provisioning and orchestration. *13th International Conference on Advanced Communication Technology (ICACT2011)*, (pp. 270–273).

Moreno, I. S., Garraghan, P., Townend, P., & Xu, J. (2013). An approach for characterizing workloads in google cloud to derive realistic resource utilization

models. *2013 IEEE Seventh International Symposium on Service-Oriented System Engineering*, (pp. 49–60).

Mulia, W. D., Sehgal, N., Sohoni, S., Acken, J. M., Stanberry, C. L., & Fritz, D. J. (2013). Cloud workload characterization. *IETE Technical Review, 30*, 382–397.

Nakazawa, R., Ogata, K., Seelam, S., & Onodera, T. (2017). Taming performance degradation of containers in the case of extreme memory overcommitment. *2017 IEEE 10th International Conference on Cloud Computing (CLOUD)*, (pp. 196–204).

Oujezsky, V., & Horvath, T. (2016). Case study and comparison of SimPy 3 and OMNeT++ Simulation. *2016 39th International Conference on Telecommunications and Signal Processing (TSP)*, (pp. 15–19).

Peschlow, P., & Martini, P. (2007). A discrete-event simulation tool for the analysis of simultaneous events. *Proceedings of the 2nd international conference on Performance evaluation methodologies and tools*, (pp. 1–10).

Pinho, T. M., Coelho, J. P., & Boaventura-Cunha, J. (2016). Forest-based supply chain modelling using the SimPy simulation framework. *IFAC-PapersOnLine, 49*, 90–95.

Pratap, R., & Zaidi, T. (2018). Comparative Study of Task Scheduling Algorithms through Cloudsim. *2018 7th International Conference on Reliability, Infocom Technologies and Optimization (Trends and Future Directions) (ICRITO)*, (pp. 397-400). doi:10.1109/ICRITO.2018.8748514

Rajan, R. A., & Francis, F. S. (2015). Design principles of request scheduler in cloud. *2015 International Conference on Computer Communication and Informatics (ICCCI)*, (pp. 1–6).

Rani, E., & Kaur, H. (2017). Study on fundamental usage of CloudSim simulator and algorithms of resource allocation in cloud computing. *2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*, (pp. 1-7). doi:10.1109/ICCCNT.2017.8203998

Rawat, P. S., Dimri, P., Saroha, G. P., & Barthwal, V. (2016, 3). Power consumption analysis across heterogeneous data center using CloudSim. *2016 3rd International Conference on Computing for Sustainable Global Development (INDIACom)*, (pp. 1-5).

Reiss, C., Tumanov, A., Ganger, G. R., Katz, R. H., & Kozuch, M. A. (2012). Heterogeneity and dynamicity of clouds at scale: Google trace analysis. *Proceedings of the third ACM symposium on cloud computing*, (pp. 1–13).

Robinson, S. (2005). Discrete-event simulation: from the pioneers to the present, what next? *Journal of the Operational Research Society, 56*, 619–629.

Sahlin, J. P., & Mazzuchi, T. (2012). Optimizing QoSin distributed systems/cloud computing architectures. *International Journal of Computer Applications, 975*, 8887.

Santra, S., & Mali, K. (2015). A new approach to survey on load balancing in VM in cloud computing: Using CloudSim. *2015 International Conference on Computer, Communication and Control (IC4)*, (pp. 1-5). doi:10.1109/IC4.2015.7375671

Schemenauer, N., Peters, T., & Hetland, M. L. (2001). PEP 255–Simple generators in Python. *PEP 255–Simple generators in Python*.

Scherfke, S. (2014). Discrete-event simulation with SimPy. *OFFIS–Institute for Information Technologie, USA*.

Schriber, T. J., Brunner, D. T., & Smith, J. S. (2017). Inside discrete-event simulation software: How it works and why it matters. *2017 Winter Simulation Conference (WSC)*, (pp. 735–749).

Seiden, S. S. (2002). On the online bin packing problem. *Journal of the ACM (JACM), 49*, 640–671.

SimPy, T. (2018). Overview—SimPy 3.0. 11 documentation. *Simpy documentation, https://simpy. readthedocs. io/en/latest/(2017, accessed 16 April 2018). Google Scholar*.

Sokolowski, J., & Banks, C. (2009, 1). Principles of Modeling and Simulation: A Multidisciplinary Approach. doi:10.1002/9780470403563

Srinivasan, B. (2020, 9). Words of advice: teaching enzyme kinetics. *The FEBS journal*. doi:10.1111/febs.15537

Tadic, L., Afric, P., Sikic, L., Kurdija, A. S., Klemo, V., Delac, G., & Silic, M. (2019). Analysis and Comparison of Exact and Approximate Bin Packing Algorithms. *2019 42nd International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, (pp. 919–924).

Takayama, T., & Kourai, K. (2020). Optimization of Parallel Applications under CPU Overcommitment. *2020 IEEE 44th Annual Computers, Software, and Applications Conference (COMPSAC)*, (pp. 51–60).

Tang, X., Li, Y., Ren, R., & Cai, W. (2016). On first fit bin packing for online cloud server allocation. *2016 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, (pp. 323–332).

Tao, Y., Zhang, Q., Shi, L., & Chen, P. (2011). Job scheduling optimization for multi-user MapReduce clusters. *2011 Fourth International Symposium on Parallel Architectures, Algorithms and Programming*, (pp. 213–217).

Tinini, R. I., dos Santos, M. R., Figueiredo, G. B., & Batista, D. M. (2020). 5GPy: A SimPy-based simulator for performance evaluations in 5G hybrid Cloud-Fog RAN architectures. *Simulation Modelling Practice and Theory, 101*, 102030.

Tirmazi, M., Barker, A., Deng, N., Haque, M. E., Qin, Z. G., Hand, S., . . . Wilkes, J. (2020). Borg: the next generation. *Proceedings of the Fifteenth European Conference on Computer Systems*, (pp. 1–14).

Tomás, L., & Tordsson, J. (2013). Improving cloud infrastructure utilization through overbooking. *Proceedings of the 2013 ACM Cloud and Autonomic Computing conference*, (pp. 1–10).

VanRossum, G., & Drake, F. L. (2010). *The python language reference.* Python Software Foundation Amsterdam, Netherlands.

Varalakshmi, P., Ramaswamy, A., Balasubramanian, A., & Vijaykumar, P. (2011). An Optimal Workflow Based Scheduling and Resource Allocation in Cloud. In A. Abraham, J. Lloret Mauri, J. F. Buford, J. Suzuki, & S. M. Thampi (Ed.), *Advances in Computing and Communications* (pp. 411–420). Berlin: Springer Berlin Heidelberg.

Varghese, B., & Buyya, R. (2018). Next generation cloud computing: New trends and research directions. *Future Generation Computer Systems, 79*, 849–861.

Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015). Large-scale cluster management at Google with Borg. *Proceedings of the Tenth European Conference on Computer Systems*, (pp. 1–17).

Wang, J., Zhang, H., Xu, Z., He, W., & Guo, Y. (2016). A scheduling algorithm based on resource overcommitment in virtualization environments. *2016 First IEEE International Conference on Computer Communication and the Internet (ICCCI)*, (pp. 439–443).

Wang, W., Zeng, G., Tang, D., & Yao, J. (2012). Cloud-DLS: Dynamic trusted scheduling for Cloud computing. *Expert Systems with Applications, 39*, 2321–2329.

Wilkes, J. (2020). Google cluster-usage traces v3. *Technical report at https://github.com/google/cluster-data*.

Williams, D., Jamjoom, H., Liu, Y.-H., & Weatherspoon, H. (2011). Overdriver: Handling memory overload in an oversubscribed cloud. *ACM SIGPLAN Notices, 46*, 205–216.

Woltman, H., Feldstain, A., MacKay, J. C., & Rocchi, M. (2012). An introduction to hierarchical linear modeling. *Tutorials in quantitative methods for psychology, 8*, 52–69.

Xiao, J., & Wang, Z. (2012). A priority based scheduling strategy for virtual machine allocations in cloud computing environment. *2012 International Conference on Cloud and Service Computing*, (pp. 50–55).

Xiong, X., Ma, L., & Cui, C. (2019). Simulation Environment of Evaluation and Optimization for Moving Target Defense: A SimPy Approach. *Proceedings of the 2019 the 9th International Conference on Communication and Network Security*, (pp. 114–117).

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

Zhang, Q., & Boutaba, R. (2014). Dynamic workload management in heterogeneous cloud computing environments. *2014 IEEE Network Operations and Management Symposium (NOMS)*, (pp. 1–7).

Zhang, W., Han, S., He, H., & Chen, H. (2017). Network-aware virtual machine migration in an overcommitted cloud. *Future Generation Computer Systems, 76*, 428–442.

Zhang, X., Feng, Y., Feng, S., Fan, J., & Ming, Z. (2011). An effective data locality aware task scheduling method for MapReduce framework in heterogeneous environments. *2011 International Conference on Cloud and Service Computing*, (pp. 235–242).

Zhang, X., Shae, Z.-Y., Zheng, S., & Jamjoom, H. (2012). Virtual machine migration in an over-committed cloud. *2012 IEEE Network Operations and Management Symposium*, (pp. 196–203).

Appendix A – Trace Data Source

- Source: Google Borg Cluster Traces v3
- Timespan: 8 Borg Datacenter Cells on 1-May-2019
- Format: Google BigTable dataset
- Reference: https://github.com/google/cluster-data/blob/master/ClusterData2019.md

Dataset Name	**Brief Description** *(Extracted from the Google Traces v3 Documentation)*	**Num. of records**
Machine Events	Each machine is described by one or more records in the machine event table. The majority of records describe machines that existed at the start of the trace.	~640,000
Machine Attributes	Machine attributes are key-value pairs representing machine properties, such as the kernel version, clock speed, presence of an external IP address, etc. Instances can specify constraints on machine attributes	~29M
Collection & Instance Events	The CollectionEvents table and InstanceEvents table describe the life cycle of collections and instances respectively. All the tasks within a job usually execute the same binary with the same options and resource request. However, some programs need to run different types of tasks with different resource requirements.	~64M
Instance Usage	Usage values from a series of non-overlapping measurement windows for each instance. The windows are typically 5 minutes (300s) long, although may be shorter if the instance starts, stops, or is updated within that time period. The measurements may extend for up to tens of seconds after an instance is terminated, or (rarely) for a few minutes.	~60B

Appendix B – Simulated Data Sources

- Source: Generative Discrete Event Simulation Model Output
- Timespan: 2000 time units
- Counts: ~2000 Simulations with 50+ machines each
- Format: Json and CSV
- Simulation Level Attributes: Overcommitment Rate, Job Arrival Rate, Machines per datacenter

Name	Brief Description	Num. of records
Job Creations	Each job is described by its name, creation time, cpu/memory requested, forecast for actual job usage (unknown to the scheduler)	~111M
Job Schedule	Dataset includes information about when the job was scheduled, the machine it was scheduled on, the wait time before being scheduled, the residual machine capacity after scheduling	~33M
Machine State	Contains the state (cpu and memory capacity) of every machine in the datacenter at regular time epochs when discrete events are dispatched	~30M
Job Eviction/Completions	Dataset includes information the final state of the job (eviction or completion) and the time it was evicted	~22M