Data-driven Methods for Efficient Resource Allocation and Energy Management in Cloud Datacentres

by

Naghmeh Dezhabad

A Dissertation Submitted in Partial Fulfillment of the
Requirements for the Degree of

DOCTOR OF PHILOSOPHY

in the Department of Computer Science

Data-driven Methods for Efficient Resource Allocation and Energy Management in Cloud Datacentres

by

Naghmeh Dezhabad

Supervisory Committee

Dr. Sudhakar Ganti, Co-supervisor
(Department of Computer Science)

Dr. Gholamali Shoja, Co-supervisor
(Department of Computer Science)

Dr. Stephen W. Neville, Outside Member
(Department of Electrical and Computer Engineering)

Supervisory Committee

Dr. Sudhakar Ganti, Co-supervisor
(Department of Computer Science)

Dr. Gholamali Shoja, Co-supervisor
(Department of Computer Science)

Dr. Stephen W. Neville, Outside Member
(Department of Electrical and Computer Engineering)

ABSTRACT

Cloud providers aim to efficiently deliver various types of services on-demand to user jobs with diverse requirements and different desired performance metrics. Besides, cloud systems provide heterogeneous types of resources. Diversity of demands and variety of resources impose challenges like resource allocation, capacity planning and scheduling strategies. Among all available cloud-based solutions, spot instances are ideal for batch job execution with flexible completion times and failure tolerance. The combination of spot instance sizes and prices provides flexibility to execute jobs satisfactorily with regard to their computational requirements. However, the demand for each type of batch instance fluctuates over time and it brings a few challenges to provide adequate instances to match the workload demand. This makes it necessary to have a mechanism to dynamically decide on the best configuration of resources assignment to the submitted jobs.

The objective of this thesis work is to propose data-driven methods for efficient resource allocation in cloud environment. For this, we first present a new methodology to categorize cloud workload according to its resource demands. We employ a modified hierarchical clustering algorithm to determine the number of clusters with fairly homogeneous resource demand. Then, for each one of these demand profiles, we extract the number of incoming requests over time and predict the future number

of requests. We take these predictions into consideration for design of batch processing system. After this preliminary work, we expand the study by presenting a batch processing system. We use the result of clustering and prediction models with some changes, then complete the architecture by adding decision making component. We come up with an optimization-based solution to allocate spot resources for executing arriving batch jobs in the cloud platforms over time. The methodology manages the trade-off between the cost of renting spot instances and the user-centric quality of service in terms of job waiting time. Given a set of jobs and spot instances of various types, our algorithm selects the best combination of instances in each analysis window. In addition, the method takes into account the job arrival prediction and characteristics of input jobs to adapt the algorithm. We also investigate Alibaba datacentre workload for any unusual behavior in servers' resource usage pattern. Detecting these types of anomalies would help in making decisions for efficient resource allocation and increase performance. In another use case of resource usage prediction in datacentres, we study the applicability of resource usage prediction and its impact on optimal resource allocation where the power consumption is kept to a minimum, while the Service Level Agreement (SLA) will be met. The results show that using prediction techniques, we can estimate datacentre resource usage and consequently we are able to reduce the energy usage up to 56%.

Contents

List of Tables

List of Figures

ACKNOWLEDGEMENTS

I would like to express my utmost gratitude to my supervisors, Dr. Sudhakar Ganti and Dr. Gholamali Shoja for their patience, for being generous with their time and for great advice that helped me overcome the difficulties of research. The completion of this thesis would not have been possible without their significant support. I am truly indebted to them for their insightful comments and feedback.

The process of doctoral studies is more than just an intellectual pursuit. I am grateful to have made wonderful friends at University of Victoria. A special thanks goes to all my friends and schoolmates who I have spent lots of time with them through highs and lows of my PhD experience.

Last but not least, I owe more than thanks to my family, my parents and my sister, who are the biggest blessing in my life. They taught me my responsibilities as a human being. Without their inspiration, dedication, encouragement and support, I would not be able to succeed throughout my life.

Chapter 1

Introduction

In this data-driven era, handling Big Data has become a major challenge, since we are collecting massive amounts of complex data from people, actions, sensors, algorithms and the web. Cloud computing offers many benefits including faster access, lower costs and improved reliability and it's an optimal solution for dealing with big data issues such as storing, editing, retrieving, analyzing, maintaining and recovering. Large-scale distributed computing clusters, server virtualization and parallel processing are the most important properties that make cloud a technology with desired capability. Cloud providers try to efficiently deliver different types of service on demand to customers, such as software-as-a-service, platforms-as-a-service and infrastructure-as-a-service. To achieve this, they need a policy to dynamically provision the virtual computing resources and share them among the users. To be exact, resource management in the cloud datacentres signifies the decisions governing the allocation of processing power (or CPU time), memory, storage, network bandwidth etc. among different cloud clients. Efficient resource allocation may mean different things for cloud providers depending on their priorities and objectives, which may involve balancing the load, maximizing revenues, minimizing response time or the usage of server resources, bandwidth and electricity. In order to perform efficient resource management, we need to consider many issues such as resource mapping, resource selection, resource pricing, resource scheduling, resource estimation, workload prediction, application scaling, provisioning, and service analyzing. There are some concepts that we need to know about before describing our suggested problem. For instance, why we choose a specific dataset to characterize? What are the features of this dataset? Is there any unusual behavior in datacentre? Which attributes we consider in our classification algorithm?

On other hand, to serve the batch jobs, what kind of resources should be offered by cloud providers? What are the characteristics of spot instances and what are the pros and cons of using them compared to on-demand resources? How the predictability of resource usage can lead us to a better resource allocation? Following this, we are going to answer these questions and explain some essential subjects in detail. This dissertation is organized as follows. Since for any performance evaluation, we need a proper real-world dataset to test and explore our solutions, hence, in Sections 1.1, we describe Alibaba and Google datasets that are widely available for cloud workloads. Then we describe colocation in datacernters and the way it works. Batch workload and spot instances are explained. At the end of Chapter 1, research questions and dissertation outline are given. Chapter 2 reviews the related works. Chapter 3 describes the proposed method for workload modeling. In Chapter 4, we present a framework for allocation of spot instances to execute batch jobs. We describe the system model, its different components, and the decision making algorithm in this chapter. In Chapter 5, we study the energy management in datacentres and how the predictability of resource usage can impact on that. An analysis of anomaly detection and applying it on Alibaba dataset are given in Chapter 6. Finally, Chapter 7 concludes the dissertation and gives some suggestions for future works.

1.1 datacentre Workloads

For performance evaluation of any system, one of the major steps is to consider workload modeling. Evaluating a system with an incorrect workload will probably produce irrelevant and unreliable results. Size and scale of the workload, important features, and the level of details needed in workload depends on the goal of the evaluation [1]. One of the first and best datacentre datasets is Google Cluster Trace [2]. The first version published in 2011 and has been analyzed and various features extracted from that in various studies. It helped many researchers to improve cluster modeling and management systems. Some of these studies have been mentioned in Chapter 2. It consists of one month workload on more than 12000 machines on Google cluster. Data is classified in different tables, storing the information about machines, jobs and tasks. Google also released another trace from eight compute clusters in 2019 which includes more details about job submission, job usage and their master/worker relationships [3].

The other cluster trace that has been released in 2017 is Alibaba Cluster Data.

This is a resource usage dataset from Alibaba production cluster that runs both online services and batch jobs [4]. Alibaba dataset includes several tables for machines, batch jobs and online service jobs and also some attributes such as life-cycle events, reserved resources and actual usage. The first version of data has been collected from 1300 machines only in a 12-hour period. However the newer version includes around 4000 machines in a period of 8 days, published in 2018. They lately released a two-month workload traces of a GPU cluster with over 6500 GPUs in Alibaba Platform for AI [5].

Google Cluster and Alibaba Cluster has some key differences in terms of scheduler, server types, throughput, resource usage, etc. Google has about 10 types of servers in its trace as shown in Table 1.1. And Alibaba has published six types of servers, as shown in Table 1.2. We can see an obvious heterogeneity in google cloud platform, while Alibaba cloud platform is almost homogeneous.

Google cluster uses a scheduler called Borg [6], which is based on regular container. However, Alibaba has a scheduler base on its own container (Pouch). Pouch is a rich container and its internal application experience is similar to a virtual machine.

Task submission rate published in google trace is up to more than 400 tasks per second, of which repeated tasks account for the majority. The number of instances submitted in Alibaba cloud is nearly 10,000. So, Alibaba scheduler is capable of handling this large throughput. In addition, the number of re-submitted tasks in google trace reaches as many as 300 in one second sometimes, while the number of retried tasks in Alibaba trace is relatively small, which is less than 10.

Production-level jobs in google trace use more CPU time than non-production-level jobs. However, CPU used by online services on Alibaba cloud platform is less than that of batch instances. This may be due to the large number of batch loads on Alibaba cloud platform.

We can see elasticity feature of resource allocation in Alibaba cluster, since batch instances largely use the spare resources that containers reserved but not used. They also use resource overprovisioning and overbooking, which helps in achieving high elasticity. However the high elasticity may degrade the performance of online services. In this case, Alibaba cluster sets boundaries for resources used by batch tasks to guarantee the steady performance of both online services and batch tasks. This property has been called plasticity of resource allocation [7], [8].

Table 1.1: The normalized configuration of servers in Google cluster [9]

Number of machines	**Normalized CPU**	**Normalized Memory**
6732	0.50	0.50
3863	0.50	0.25
1001	0.50	0.75
795	1.00	1.00
126	0.25	0.25
52	0.50	0.12
5	0.50	0.03
5	0.50	0.97
3	1.00	0.50
1	0.50	0.06

Table 1.2: The configuration of Alibaba cloud platform

Number of machines	**CPU**	**Normalized memory**
736	64	0.6899697150104833
365	64	0.6900007765381927
199	64	0.6900059534594777
7	64	0.9999585846297208
5	64	1
1	64	0.5747883933424792

1.2 Datacentre Colocation

Colocation technology was initially created as a response to the growing business demand and increasing resource costs. Datacentre colocation can do this by reusing of existing resources to help a business meet the business requirements. Alibaba Group also started to use this technology to service their e-commerce requests. In addition to these online services, Alibaba provides large-scale offline computing services too. Since online and offline services have quite different resource usage patterns, there were initially two independent datacentres to provide resources to each of them. However, the problem was the low utilization percentage of the resources. Since the traffic of different services has different peak times and also, their response time requirements may not be the same, the use of resources can be prioritized among various services. Based on these two aspects, Alibaba explored the service colocation technology in a different direction. Colocation technology can be implemented in a server which hosts two different types of services sharing common resources as shown in Fig. 1.1.

Figure 1.1: Colocation on server

In 2015, Alibaba tested the first offline prototype of the technology and in 2016 they ran it in production level with 200 physical machines. It was up until 2018 that they consolidated latency insensitive batch jobs and latency critical online service jobs together deployed on a large scale with thousands of machines. Aside from Alibaba, workload colocation on a same cluster has been previously studied and adopted in other datacentres in order to improve utilization and reduce cost of infrastructure and electrical energy [10]-[14]. However, such consolidation brings new performance management challenges due to intrinsically different nature of a heterogeneous set of mixed workloads, ranging from scientific simulations to multitier transactional applications. The fact that different workloads have different demand, imposes the need for new scheduling mechanisms to manage colocated heterogeneous sets of applications.

Batch Jobs are computationally intensive long-running jobs (in the order of tens of minutes) whose performance is measured in terms of throughput and job completion times instead of latency. Applications like web crawling, index updates in search engines and web log analysis are some examples of that [12]. On the other hand, online service jobs as the name suggests, are live to action and need processing at that moment. The response time constraints of service jobs can be quite short (tens of milliseconds) and their failure recovery should be fast.

The overall architecture of colocation technology has three key features.

1. Integrating the resources from two services in a shared pool
2. Unified scheduling system for resource allocation
3. Implementing isolation and preemption during resource competition

This hierarchical architecture is shown in Fig. 1.2. The first underlying layer is infrastructure which is all the hardware and networking facilities. Next layer is resource layer which is shared for centralized management. On top of that, there is a scheduling layer. Sigma is the scheduler for online services and Fuxi is offline service scheduling system. However, there is a higher-level unified scheduler which manage allocation of resources into two other schedulers. The uppermost layer in colocated part is colocation management layer which is responsible for orchestrating and implementing service operation, managing physical resources, monitoring services and decision making.

There are two types of colocation scenarios. First, the online cluster provides resources for colocation, and online resources provide extra computing capability for offline services. In the second one, on the contrary, the offline cluster provides resources for colocation. Using either techniques, the resources that are not been fully utilized by one type of services, will be given to another type. Therefore, the overall utilization of machines would be improved by taking full advantage of resources. By using two different scheduling systems, Fuxi and Sigma [15], for batch and online jobs respectively, the performance was not efficient enough. The new online-offline colocation cluster running mixed jobs has increased the utilization from 10% to 40%, while the quality of service of online services hasn't been downgraded [16]. However, there are still challenges for job scheduling and resource allocation between two different job categories to increase utilization and provide better quality of service.

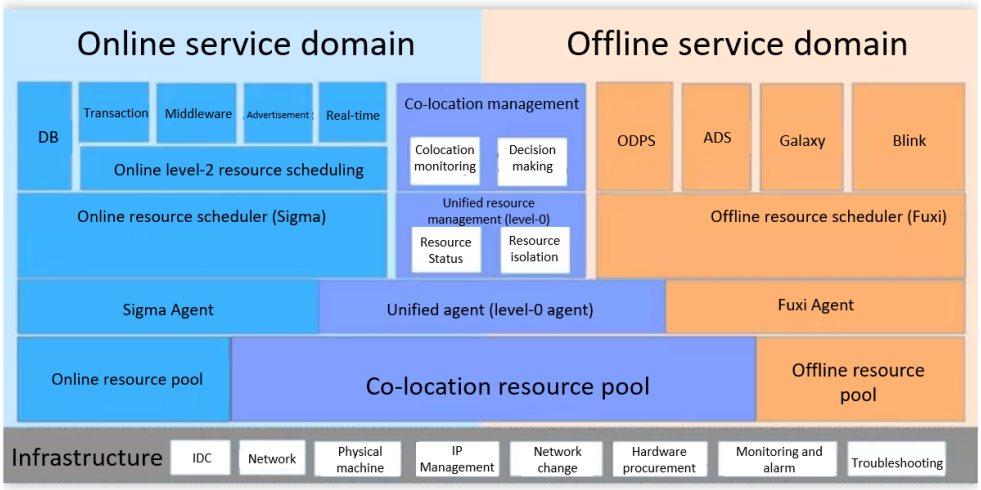


Figure 1.2: Colocation Architecture [16]

1.3 Batch workload

Alibaba datacentre colocated batch and online services running on the same hardware structure. However, they kept track of batch and online processes separately on different log files. Batch workload has been submitted to the system in the form of jobs. A batch job contains multiple tasks which execute different computing logic. Within a task, there are several instances. Instances are the smallest unit of the batch job which execute the same function on different input data. For a 12-hour period, there are 12951 batch jobs, 76989 tasks and more than 15 million instances in total. Batch workload in Alibaba trace is described by two tables: task table and instance table. In the task table, information about creation and modification time, number of instances per task and requested resources for each instance is given. We can find information about creation and completion time of instances, their status and the actual resource utilization in the instance table. In Chapter 3, we use batch instance trace *batch_instance.csv*, which has many different attributes. For this work, three attributes are considered: cpu usage, memory usage and job duration. Clustering using these attributes gives us the batch workload grouped into three profiles.

1.4 Spot Instances

To offer the cloud resources, most providers propose three pricing models: Reserved, on-demand and spot instances. Reserved instances provide users with the ability to

Table 1.3: Differences between Spot instances and On-Demand instances

Feature	**Spot Instances**	**On-demand Instances**
Launch time	Can only be launched immediately if the Spot Request is active and capacity is available	Can only be launched immediately if you make a manual launch request and capacity is available
Available capacity	If capacity is not available, the Spot Request continues to automatically make the launch request until capacity becomes available	If capacity is not available when you make a launch request, you get an insufficient capacity error (ICE)
Hourly price	The hourly price for Spot Instances varies based on demand	The hourly price for On-Demand Instances is static
Instance interruption	The Amazon EC2 Spot service can interrupt an individual Spot Instance if capacity is no longer available, the Spot price exceeds your maximum price, or demand for Spot Instances increases	You determine when an On-Demand Instance is interrupted (stopped or terminated)

reserve resources for a specific period of time and pay for the resources beforehand. In turn, they get discount on the hourly usage price. On-demand instances let users pay for the computing capacity by the hour depending on the resource usage of their application. Spot instances first introduced by Amazon EC2 and then offered by other Cloud vendors like Google (called Preemptible Virtual machine Instances) and Microsoft Azure (called low-priority virtual machines). Using them, you can bid for unutilized computing resources and save cost compared to other instance types. While submitting a Spot Instance request, you need to specify the instance type, availability zone, and the price you are willing to pay. If the spot price is below your bid price, you can use that instance until the spot price goes above your bid price. At that point, you get interrupted and the instance would be taken away. These characteristics make spot instances very suitable for batch processing, since the batch applications have the flexibility on completion time and tolerance for failures. In Table 1.3, we summarize the key differences between Spot instances and On-Demand instance [17].

One of the fundamental problems is scheduling and pricing of the spot instances. The providers try to dynamically change the price of each instance type to get the maximum revenue [18]-[20]. This is where the need for demand prediction of batch

jobs has emerged. To determine the price of an instance type in future, we need to consider the resources already promised and the future workload [21].

1.5 Research Goals

In this work, we studied data-driven methodologies to improve datacentre management. This includes the characterization of workload arriving to cloud datacentres, resource allocation and energy management. We propose an architecture for batch processing system. There are already batch processing systems available on cloud platforms like AWS (called AWS Batch service). However, our methodology takes the characteristics of the incoming jobs into account for resource assignment decision making and improves the performance of batch processing system. Having the knowledge about the arriving requests to the system would help the model to allocate proper types of spot instances and prevents underutilization and SLA violation. Another contribution of this work is using real-world dataset from a production cluster which gives us a real sense of jobs behavior in real production cluster.

More specifically, we addressed the questions and challenges mentioned bellow:

- Why we choose an specific dataset to characterize? What are the features of this dataset? Which attributes we consider for characterizing that?
- What is a batch workload? What makes it different from other type of workloads?
- What kind of resources can be offered by cloud providers to service the batch jobs? What are the characteristics of spot instances and what are the pros and cons of using them compared to on-demand resources?
- How machine learning helps us in the management of datacentres? Is there a way to save energy costs using ML techniques?

In what follows, we are going to provide methodologies, results and analysis to address these questions.

1.6 Dissertation Outline

The remainder of this dissertation is organized as follows:

Chapter 2 reviews the related works in three areas. First, the previous studies on workload characterization and modeling are given. Next, recent studies on scheduling of Spot instances have been mentioned. At last, we reviewed the researches on Energy management in datacentres.

Chapter 3 discusses a methodology for workload characterization using multiple dimensions. It also provides some insights about Alibaba dataset.

Chapter 4 further extends the workload characterization and uses that to design a framework for resource allocation. The framework optimally decides how to provide spot instances for batch jobs arriving in cloud datacentre.

Chapter 5 studies the energy management in datacentre and how the predictability of resource usage can impact that.

Chapter 6 investigates the anomalies in Alibaba datacentre at two levels; anomalies within a physical machine in different times of the day, and uneven load distribution among different physical machines. We also explain that how these anomalies affect the datacentre performance.

Chapter 7 concludes the dissertation and enumerates the paths for future work and further development of the allocation framework.

Chapter 2

Literature review

In this chapter, we are going to present the works that have been already done regarding the analysis and characterization of cloud workload in section 1. Also in section 2, we are providing some of the most recent and popular papers addressing the problem of spot instances price management.

2.1 Workload Characterization

There are several prior studies related to statistical analysis and characterization of datacentre workloads for in-depth understanding of the environment. Most of them analyzed Google cluster trace to reveal the underlying patterns of this trace. Abdul-Rahman et al. [22] analyzed Google trace to understand usage behavior. The investigation confirmed that user behavior in building applications and interacting with Google Cloud obeys the principles of the mice and elephant phenomenon, or the mass-count disparity. In other words, a small number of items (elephants) account for the majority of mass, whereas all of the small items together (mice) account for only a negligible proportion of the mass. Moreover, user behavior exposes the characteristics of heterogeneous mix of human, large web services, batch processing, and MapReduce computation. Sheng Di et al. [23] characterized Google workload at the application level and job levels and computed statistics about events and utilization. They also built a model that can simulate submitted jobs and tasks in a datacentre. Reiss et al. [24] addressed some new insights about Google trace. Characterizing and comparing resource requests and resource usage, along with discussion about Google task scheduling were their work's contribution. Alam et al. [25] deduced that the Google

trace jobs are trimodal in nature and profiled them into three major types to help scheduling decisions. Another work focused on machine level characterization and they revealed that majority of machines are homogenous in terms of cpu and memory capacity and only a small portion of machines have the largest cpu and memory [26]. Mishra et al. [27] developed a multi-level methodology of task classification and describes applications of task classification to capacity planning and task scheduling. Task classification provides a way to forecast application growth by tracking changes in resource consumption by task class. In recent years, newly published Alibaba trace became popular among researchers. Lu et al. [28] studied imbalances in the dataset in different aspects: a)Spatial imbalance which means the resource utilization is not uniformly distributed across machines and workloads. b)Temporal imbalance means the machine and workload resource usage is time-varying. c) Imbalances in proportion of multi-dimensional resource usage per workload, and d) imbalance in resource demand and task specifications between online service and offline batch jobs. Cheng et al. [29] analysed Alibaba trace from a different point of view. Their study addressed the challenges and complexity that Alibaba workload co-location imposes on the management system. Another related work is done to help designing effective resource management for cloud platform by powerful analysis of machine learning frameworks [7].

2.2 Scheduling and Pricing of Spot Instances

In addition to the previous papers, we studied some other works related to batch processing and Spot market in cloud. There are a number of papers that concentrated on the problem of resource pricing from cloud provider point of view and tried to maximize the providers revenue. Babaioff et al. [30] devised a framework that includes scheduling, pricing and demand prediction for the cloud resources. Their algorithm dynamically computes future resource prices based on supply and demand, where the demand includes both resources that are already committed to and predicted future requests, and schedules and prices the current request at the cheapest possibility. They have taken advantage of an online learning algorithm and implemented their work using public and private cloud systems, Azure Batch of Microsoft and Hadoop/YARN. In another similar project, Zhang et al. [18] modeled a similar problem as a constrained discrete-time optimal control problem and used Model Predictive Control (MPC) to find its solution. It is basically a technique to deal

with optimization problems in a dynamic setting. In their model, there is a resource controller which is responsible for controlling both the price and and capacity allocated to each Virtual Machine (VM) type. They also forecast the behaviour of future demand using an auto-regressive function. There is a group of papers Menache et al. [31] that applied online learning algorithm for sequential decision making task. They utilized a regret-minimizing online learning algorithm to provide the combination of Spot and on-demand instances for servicing batch jobs in order to balances the fundamental tradeoff between cloud computing costs and job due dates. Their algorithm dynamically adapts resource allocation by learning from its performance on prior job execution while incorporating history of spot prices and workload characteristics. They evaluated their work by simulation on a job trace from a datacentre cluster and Amazon EC2 spot market price. They proved that the algorithm quickly converges to a set of best performing policy and the average regret of the approach reaches to zero with time for a given dataset.

Stokely et al. [32] studied market-based resource provisioning in Google compute clusters, and presented a solution using ascending clock auction. However, the focus of these studies is to find appropriate mechanisms to achieve desired fairness and efficiency objectives, rather than allocating resources from suppliers perspective.

2.3 Energy Management in Datacentres

Managing power consumption of cloud servers is a fundamental problem in any modern cloud data centers. To address this problem, a broad range of techniques and approaches has been proposed through the years. Server consolidation is an approach to enhance cloud data center power efficiency. This technique which occurs through virtualizations, aims to increase the utilization of physical servers, hence decreasing the number of them and reducing the power consumption. As a matter of fact, servers even at a very low load, like 10% CPU utilization, consume around 50% of their peak power consumption [33]. Using virtual machine migration, VMs can be moved between physical machines and combine multiple workloads onto fewer servers, so idle servers can be switched off. Srikantaiah et al. [34] took the first step toward energy efficient consolidation, while providing required performance metrics. They addressed different challenges involved in the problem and applied a multi-dimensional bin-packing problem for allocating and migrating workloads to achieve energy optimality. They also studied the impact of multiple resources (like CPU and disk) utilization on

the performance and energy consumption of consolidated workloads. Results showed that increase in disk utilization can be a degrading factor in server performance. Besides, there is always an optimal combination of resource utilization that minimize energy per transaction of consolidated workload. Chen et al. [35] further improved the server consolidation method by adding spatially awareness into that, in which the workload will be placed into most suitable machines considering thermal distribution and cooling facilities. They proposed a reinforcement learning model to predict the thermal distribution resulting from different placement of incoming load and then picked the optimal one. The results showed 13%-18% saving in cooling energy.

There is a group of studies on power management that rely on workload prediction and scheduling, in which the efficient placement of workload would reduce the power consumption in servers. On an early work, Bradly et al. [36] developed a predictive technique for minimizing power consumption, server failure rate and client disconnection rate. The periodic behavior of e-commerce and web-servers' workload make the prediction achievable. Their proposed algorithm decides to turn on or off the servers in near future considering the current and predicted demand. If the CPU utilization of any resource exceeds a threshold, extra resources would be added. On the other hand, if the utilization of all servers are lower than the threshold and adding the load of one server won't make them over-utilized, then at least one server can be switched off. The implementation of the algorithm on the workload data from production servers showed 20% saving in energy with having less than 1% of unmet demand.

In another attempt, Goiri et al. [37] presented a dynamic job scheduling method to enable consolidation of workloads into smaller number of nodes, while maintaining quality of service. The power-aware scheduling algorithm gets different parameters like resource requirements of VMs, available resources offered by servers, servers' energy consumption and Service-level Agreement (SLA) constraints as input. The algorithm then assigns different scores for each possibility of server-VM allocation and tries to find the best possible schedule. They achieved a power consumption reduction of 15% as compared to typical policies. Machine learning methods have been extensively used for demand forecasting and energy management in previous works [6-8]. Farahnakian et al. [38] proposed a CPU usage prediction method based on history of machine usage, since CPU is the main consumer of energy in servers. They made use of linear regression model to identify the over and under-utilized hosts and minimize the energy consumption by migrating VMs and changing the servers to sleep mode from idle mode. In a similar study Hsieh et al. [39] did short-term future

CPU prediction of hosts for the purpose of economizing energy in cloud datacentres. They used the combination of current and future utilization as a measure for servers' workload. Gray-Markov prediction model have been used and applied on real-world workload trace [39]. Dabbagh et al. [40] developed a framework for prediction of future VM request and resource requirements using machine learning models. It consists of three components: data clustering with k-means, workload prediction using stochastic Wiener filter, and power management. The framework monitors physical machines in real time and effectively decides when physical machines need to be turned off. They used Google cluster trace for evaluation of their approach, considering only two types of resources, CPU and memory. The results demonstrated the energy management system achieved near-optimal energy efficiency.

By the newly emerged Virtualized network technology, the ability to manage the network got independent from physical equipment. Therefore, the recent studies on power efficiency have moved toward this path. A group of papers focus on the problem of optimal virtual network function placement as a way to minimize energy consumption. For example, Yang et al. [41], addressed the problem of VNF placement with the goal of minimizing the cost, energy consumption, network load and latency in cloud datacentres. Their heuristic algorithm deploys VNFs at the least number of servers, and aggregates service function chain (SFC) requests on the least amount of paths. In another work, Zhang et al. [42] investigated the joint optimization of Virtual Network Function (VNF) placement and traffic steering with the goal of minimizing energy consumption. They devised a power consumption model for servers and physical links for SFC deployment. The power consumption in an Network functions virtualization (NFV) consists of two parts. First, the power consumption of servers hosting VNF instances and second, the power consumption of physical transmitting links. They assumed the consumption of servers is mainly related to CPU utilization of those machines. Then, they formulated the problem as an Integer Programming model and found the near-optimal solution. The final results showed 14.08% saving in power consumption. There are a couple of papers which investigated the impact of SFC orchestration on energy efficiency in NFVs. Kim et al. [43] designed an SFC construction and reconfiguration algorithm to guarantee QoS requirement and minimize power consumption. They first find the shortest path between source and destination of service using Dijkstra's algorithm, then decides where to allocate NFVs to virtual machines along the path to minimize energy. The algorithm is also able to reconfigure SFC path when the energy consumption of idle servers exceeds a given threshold.

Sun et al. [44] also proposed an energy-aware algorithm for orchestrating online SFC requests. They formulated the problem as an integer programming model in order to find the optimal model. Then, a low complexity heuristic-based algorithm has been used to solve this NP-hard problem. They have proved that this method consumes less energy than the benchmark algorithms.

Chapter 3

Workload Modeling

In previous chapters, we reviewed research done on workload characterization as well as scheduling and price management for Spot instances. In this chapter, we discuss the procedural steps used for extracting features of Alibaba trace. After defining the problem, the methodology used for workload characterization and further analysis on the dataset is presented. A comparison between characteristics of Google and Alibaba datasets is also presented at the end.

3.1 Methodology

The step by step methodology used in this study is depicted in Fig. 3.1. As a first step, to select workload dimensions, we collected data from batch instance table with *determinated* status which means that their execution is successfully completed. For each completed instance, there are two measured metrics: instance start time and end time. The difference between these two values gives us the time an instance spent in the system called instance duration. Along with this attribute, we consider two others for clustering: real cpu usage which shows the actual number of cpus the instance is using and real memory usage which is the average normalized memory of the instances. Selection of workload dimensions depends on the application of the workload characterization. These three parameters together represent batch instances' demand for the resources. One instance can have less cpu or memory usage, but its long execution duration in the system imposes a high resource usage. The ultimate goal of the algorithm is to categorize all batch instances into three profiles: low-demand, medium-demand and high-demand instances. So we decided to go with

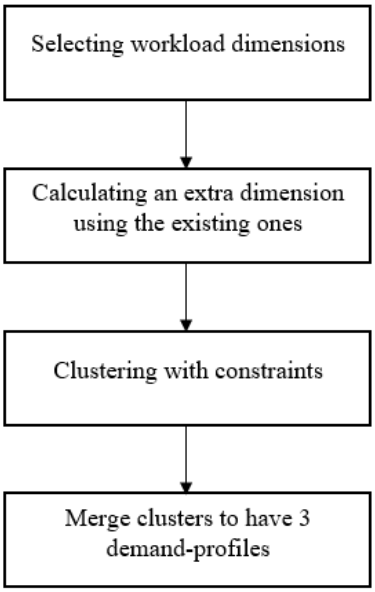


Figure 3.1: Methodology stages

three groups in this work, although it could be any numbers.

Feature creation is part of machine learning engineering process that implies human intervention in creatively mixing the existing features. The task is performed by mathematical operations on existing features to generate new derived features with more predictive power than the originals. For this purpose, we first propose a formula which derives a new feature called instance's demand, based on three other parameters, CPU, memory and length. In other words, it can be interpreted as the distance of any point from the center, in a three-dimensional Euclidean space with the axes labeled as CPU, Memory and Length. This formula is desirable for computing demand, because if all attributes are low, the result for demand would be low and, if one attribute value gets high, the demand gets a high value, which means it compensates the other ones. It also gives equal weight and significance to all three parameters. The demand is defined as shown in Eq. 3.1:

$$d = \sqrt{cpu^2 + mem^2 + len^2} \tag{3.1}$$

For calculating the extra demand dimension from the three existing dimensions, we

need to normalize them before applying Eq. 3.1, since their values are not of the same units. Generally, variables that are measured at different scales do not contribute equally to the model fitting and learning function and might end up creating a bias. Also, K-Means algorithm which uses Euclidean distance as a measure, can get highly influenced by the differing scale of features. Min-Max normalization is a way to linearly transform data to fit to an interval $[x_{min}, x_{max}]$ as shown in Eq. 3.2. It preserves the relationships among the original data values and performs very well in cases without outliers.

$$\tilde{x}_i = \frac{x_i - x_{min}}{x_{max} - x_{min}} \tag{3.2}$$

So, after removing outliers data, we normalized all parameters to the range [0,1] and then calculate demand based on Eq. 3.1. Next, in order to determine two boundaries for dividing the space into three categories as low, medium and high demand, we fit different distributions to the empirical data in order to find the best fit. Then using the inverse function of identified distribution's CDF, called Quantile function, we can find two values for demand that split the whole dataset equally into 3 categories. The cpu and memory usage in the trace are normalized, and we used normalized values to calculate the demand. Since we do not have the actual values to set the boundaries based on them, we just split all the instances into three equal groups. We devised an algorithm to fit the best distribution on the data. The results shows that the best fit in this case would be a Log-Laplace distribution as demonstrated in Fig. 3.2. The estimation algorithm uses Sum of Squares Error (SSE) to select the best fit. Then, we calculated Sum of Squares Total (SST), and R-squared based on that to evaluate goodness-of-fit. The test gives 0.8630 value for R-squared which shows the estimation is valid.

Log-Laplace distribution is known for its power tail. It can be derived by combining two power laws and has power tails at zero and infinity. Power law is defined as a functional relationship between two variables in which exceeding one variable, say x, for large values is proportional to $x^{-\alpha}$. The Log-Laplace distribution is denoted by $\mathcal{LL}(\delta, \alpha, \beta)$, in which the quantity δ is a scale parameter, and α and β are the tail parameters at $x \to \infty$ and $x \to 0^+$. The probability density function and cumulative distribution function are given by Eq. 3.3 and 3.4, respectively:

$$f(x) = \frac{1}{\delta}\frac{\alpha\beta}{\alpha+\beta}\begin{cases} \left(\frac{x}{\delta}\right)^{\beta-1} & \text{for } 0 \le x < \delta \\ \left(\frac{\delta}{x}\right)^{\alpha+1} & \text{for } x \ge \delta \end{cases} \tag{3.3}$$

$$F(x) = \begin{cases} 0 & \text{for } x < 0 \\ \frac{\alpha}{\alpha+\beta}\left(\frac{x}{\delta}\right)^{\beta} & \text{for } 0 \le x < \delta \\ 1 - \frac{\beta}{\alpha+\beta}\left(\frac{\delta}{x}\right)^{\alpha+1} & \text{for } x \ge \delta \end{cases} \tag{3.4}$$

Using the quantile value, which is inverse cumulative distribution function, we can split all the instances into equal groups, each containing the same fraction of total population:

First quantile ($F(x) = p = 1/3$): demand= 0.25

Second quantile ($F(x) = p = 2/3$): demand=0.44

By setting these two boundaries, called low-medium bound ($b1$) and medium-high bound ($b2$), and dividing the space into three regions, instances are equally distributed, with respect to the demand.

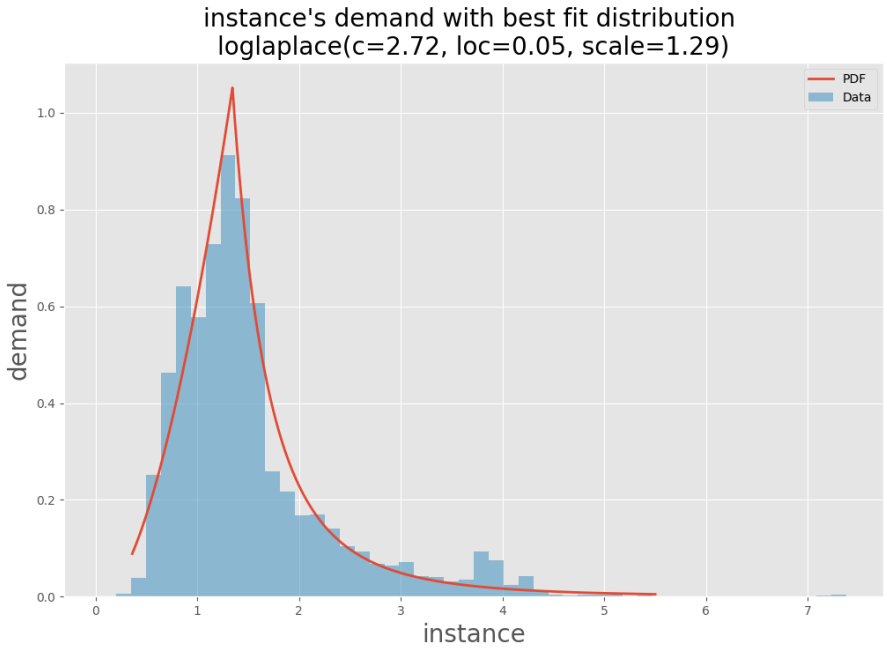


Figure 3.2: Demand distribution

3.2 Clustering with constraints

Clustering is normally categorized as an unsupervised learning. However, if there is some background knowledge available about the data, it can be considered as a semi-supervised learning. The knowledge can be either partial label information or some other constraints. There are two types of constraints between instances: must-link (two instances have to be in the same clusters) and cannot-link (two instances have to be in different clusters). To solve these kinds of problems with constraints, one approach is to change the similarity measure based on the given constraints. In a hierarchical clustering method, each point is being considered as a cluster in the initialization step. During the next steps, samples with the most similarity are grouped together, and then it merges two closest clusters until a single cluster remains. The similarity of clusters can be calculated in various ways from a distance matrix. For the hierarchical clustering, we have used a common method called Agglomerative Clustering, with the parameters given in Table 3.1. "Linkage" is criterion determines which distance to use between sets of observation. The algorithm will merge the pairs of cluster that minimize this criterion. Here, we have chosen "ward" that minimizes the variance of the clusters being merged. Affinity is the metric used to compute the linkage and can be "euclidean", "l1", "l2", "manhattan", "cosine", or "precomputed". Here, "euclidian" has been selected for affinity. Connectivity matrix defines for each sample the neighboring samples following a given structure of the data. We set that parameter as None, since the algorithm is unstructured in clustering algorithm.

Using Euclidean distance, the distance matrix can be defined as 3.5:

$$D_{A,B} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2 + (z_A - z_B)^2} \tag{3.5}$$

In clustering with constraint, given the initial constraints about the data, we can alter the distance matrix in a way that the distance between two must-link pairs decreases and cannot-link pairs increases. In our method, we use each instance demand as a prior knowledge to get a more efficient clustering and set some cannot-link constraints as below:

1. Instances with demand less than $b1$ cannot be in the same cluster as instances

Table 3.1: Parameters in Hierarchical Clustering Algorithm

parameter name	**parameter value**
Affinity	euclidean
Linkage	ward
Number of clusters	[3,14]
Connectivity matrix	None

with demand greater than $b1$.

2. Instances with demand more than $b2$ cannot be in the same cluster as instances with demand less than $b2$.

To apply these constraints on the distance matrix, for any cannot-linked pairs (A, B), we update the equivalent element of the distance matrix, $D_{A,B}$, given by Eq. 3.6:

$$D_{A,B} = D_{A,B} + 1 \tag{3.6}$$

In order to select the suitable number of clusters, we measured the SSD parameter. SSD (sum-of-squared distance) shows the error when each point of data is represented by its corresponding cluster center. This parameter has been calculated after clustering by applying the condition in Eq. 3.6. The value of SSD is a way to validate the clustering algorithm. The smaller the SSD is, the centroids of that clustering is a better representation of the cluster instances.
In Table 3.2, SSD values are shown as a function of number of sets between K=3 and K=15. We also calculated rate of change which is defined in Eq. 3.7:

$$rate\ of\ change = \frac{(SSD\ of\ K_{i-1} - SSD\ of\ K_i) * 100}{SSD\ of\ K_i} \tag{3.7}$$

As shown in Table 3.2, when number of clusters, "K" increases, SSD decreases monotonically. However, for selecting K, there should be a balance between reducing errors and maintaining low computation overhead. Based on that, we suggest to set $K = 7$ which has a high change rate called knee-point as shown in Fig. 3.3:

We compared the results of constrained clustering algorithm with a simple hierarchical clustering algorithm on Alibaba dataset. Each instance demand can be illustrated as the distance between its corresponding location on the 3D space to

Table 3.2: SSD and its rate of change for different # of clusters (K)

# of clusters	**SSD**	**Rate of change**
3	718.7	-
4	623.4	13.26
5	565.3	9.31
6	518.9	8.20
7	328.9	36.61
8	289.1	12.10
9	270.5	6.43
10	246.3	8.94
11	231.7	5.92
12	226.4	2.28
13	215.1	4.99
14	209.1	2.79

Figure 3.3: Change of SSD by increasing number of clusters

zero. We call this the radius. As we can see in Fig. 3.4a, applying the constraints, inter-cluster instances have more similarity with regards to their demands. A proof of this statement lies in is inter-cluster variance. It means that if we measure the variance of demands for all the instances within a cluster in Fig. 3.4a, it would be lower than Fig. 3.4b. This implies that in the next level, when we merge clusters to create demand profiles, the instances within a profile would be more relevant based on their demands.

Table 3.3, for each cluster shows inter-cluster demand variance comparison between unmodified (variance1) and modified (variance2) clustering algorithm. In the last row, we calculate the average variance for all 7 clusters. The clustering algorithm with constraints decreases 40% in terms of average inter-cluster variance.

One last step of our methodology is to identify demand profiles. In Fig. 3.4a, two yellow surfaces show the demand boundaries $b1$ and $b2$ which divide the whole space into three demand profiles. Two closest clusters to the center (green and orange) can be combined and considered as low demand. Medium demand profile also contains two clusters (dark red and light red) and the other three outer clusters are in high

Table 3.3: Inter-cluster variance for two methods of clustering

cluster #	**variance1**	**variance2**
0	0.4681	0.0376
1	0.1152	0.1230
2	0.5265	0.2510
3	0.8692	0.2512
4	0.4442	0.4501
5	0.2922	0.4457
6	0.6435	0.4438
Average	0.4798	0.2860

demand profile.

Using this method, for any upcoming instance, we can predict the instance cluster first and based on that, we can have an estimation of the demand in terms of cpu, memory and time duration.

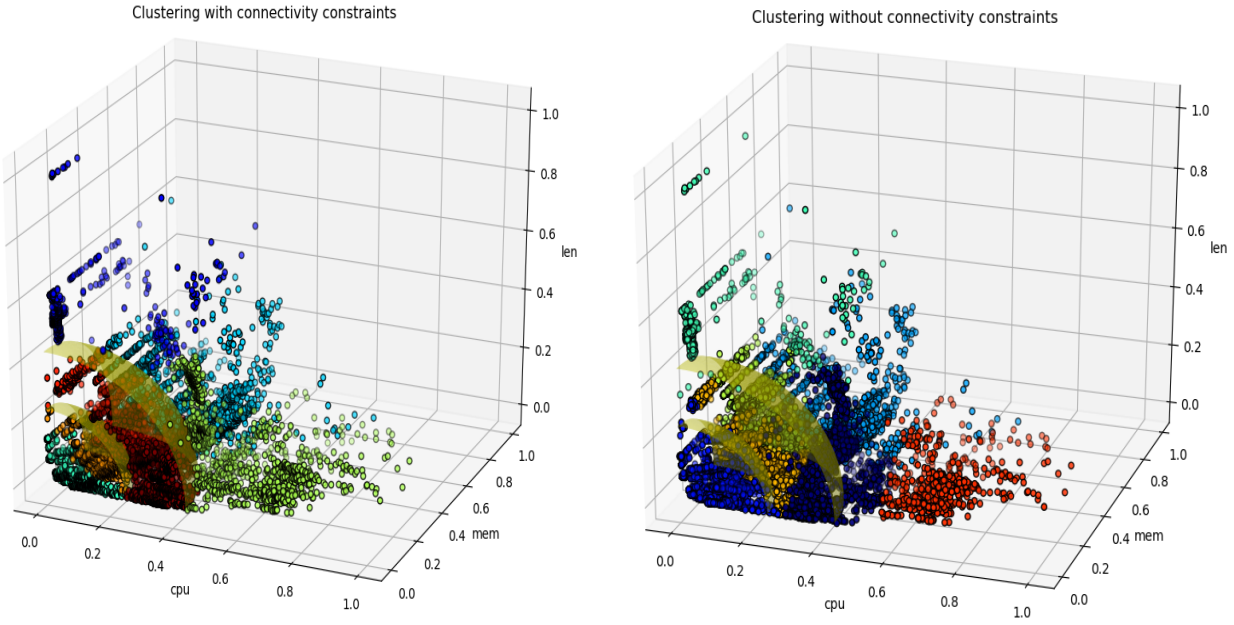
In the next step, we extracted the arrival rates of instances with different demand profiles. These time series for low, medium and high demand instances have been illustrated in Figure 3.5, 3.6 and 3.7, respectively. As expected, a majority of these instances have low resource demand, but the number of medium and high demand instances are close together. It also can be observed in the figures that the rate of requests are higher in the latter half of the day. These insights can be used for better prediction of future demand.

3.3 Characterization of Alibaba Dataset

In this section, we first explain the basics of mass-count disparity and show how it is beneficial to have a simple representation about the disparity among many small items and few large items. Then we use this technique to extract parameters of Alibaba dataset that affect system performance.

3.3.1 Mass-Count Disparity

One approach to investigate the statistical features of heavy tailed distributions is to use mass-count disparity. It consists of mass and count curves. Here, count is CDF function of distribution and shows how many items are less than a specific



(a) Clustering with constraints (b) Clustering without constraints

Figure 3.4: Comparison between two clustering algorithms

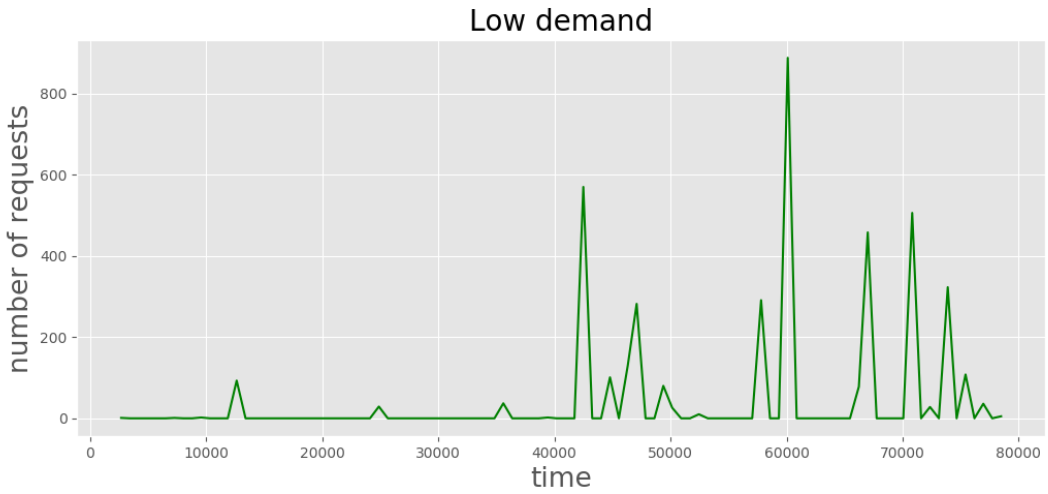


Figure 3.5: The arrival rate for low demand instances

value. However, mass function weights each item by its number and creates another distribution specifying the probability that a unit of mass belongs to an item. By comparing these two curves, it can be identified whether the data follows Pareto principle or heaviness of the tail. Larger vertical distance between mass and count curves indicates more Pareto like distribution. This is usually measured in terms of

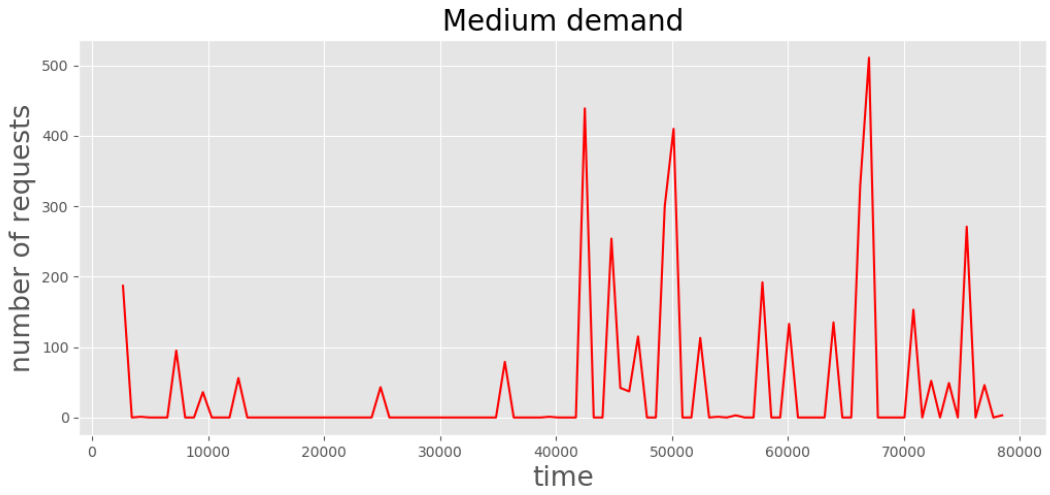


Figure 3.6: The arrival rate for medium demand instances

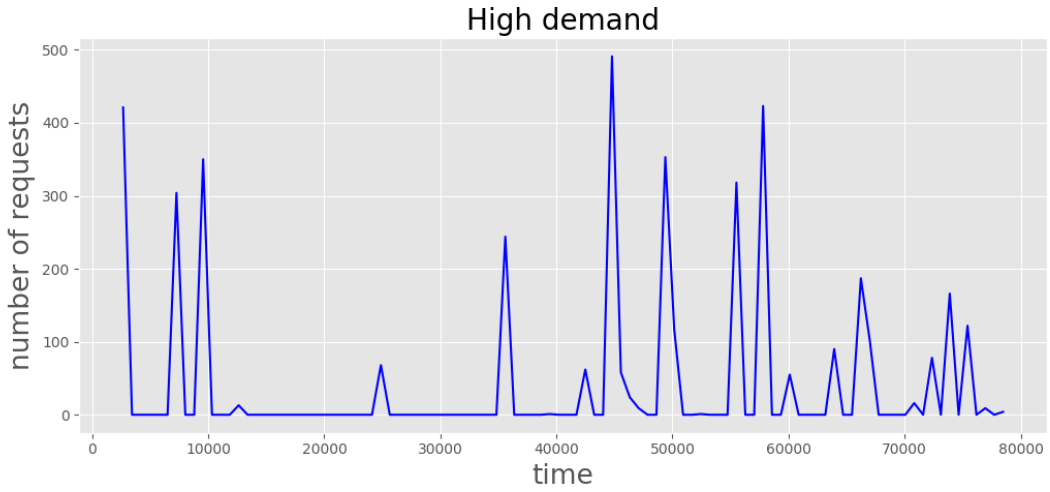


Figure 3.7: The arrival rate for high demand instances

a Gini coefficient, called joint ratio. Joint ratio finds the point in the graphs where the sum of the two CDFs is 1 and indicates the percentage X such that $X\%$ of items accounts for $Y = 100 - X\%$ of mass, and $Y\%$ of items account for $X\%$ of the mass. Smaller X implies heavier tail of the distribution. Below, the count and mass functions, $F_c(x)$ and $F_m(x)$ respectively are formulated as Eq. 3.8 and 3.9:

$$F_c(x) = \Pr(X < x) \; = \int_{-\infty}^{x} f(t) \; dt \tag{3.8}$$

$$F_m(x) = \frac{\int_0^x tf(t)\ dt}{\int_0^\infty tf(t)\ dt} \tag{3.9}$$

Where $f(x)$ is the probability density function.

The horizontal distance between medians of two distributions called median-median distance (*m-m distance*), is another parameter to measure how much larger the tail items are. The larger the distance is, the heavier the tail of the distribution. Since the absolute values depends on the units used, it usually expressed as a ratio or log of the ratio. Here, we use the ratio as defined in Eq. 3.10 to make it independent of measurement unit.

$$m-m\ distance\ ratio = \frac{m-m\ distance\ \times 100}{observation\ max - observation\ min} \tag{3.10}$$

A more extreme demonstration of mass-count disparity is the 0/50 rule which is defined by two metrics. The first one is $N_{1/2}$ or N-half, which quantifies the percentage of items from the tail needed to account for half of the mass. The second is $W_{1/2}$, and quantifies the total mass of the bottom half of the items [45], [46].

3.3.2 Analysis of Alibaba dataset using Mass-Count Disparity

In this section, we have applied mass-count disparity on a few metrics of Alibaba dataset. We have used the one-week version of dataset to extract these fine-grained specifications [4].

The first parameter we've investigated is length of instances in batch workload, or the time it takes to execute a batch instance. It is defined as the duration between instance submission time and completion time. We calculated this from batch instance table, only for instances that get terminated by considering the status of instance in "status" column. As demonstrated in Fig. 3.8, the analysis shows that the vertical distance between mass and count curves is small and the joint ratio which is the divergence between these two distributions is not significant. So, we cannot consider the distribution of instance length as a very long tail distribution where a very small portion of instances has the biggest length. However, we observe that Alibaba batch tasks are quite short tasks, since about 75% of jobs lengths are shorter than 100 seconds. So we consider 100 seconds as a boundary between short and long instances.

Similarly, Fig. 3.9 illustrates that 90% of batch instances belong to only 10% of tasks and 10% of batch instances belong to 90% of tasks. The plot also indicates that about 65% of jobs each just have only one instance and over 80% of jobs have no more than 10 instances.

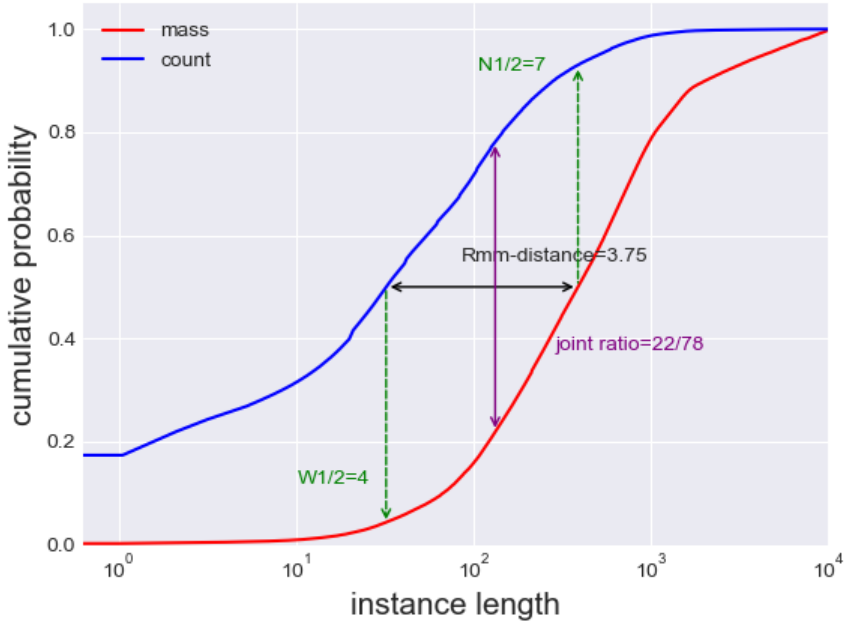


Figure 3.8: Mass-count analysis of instance length

In Fig. 3.10, we assessed the number of tasks per batch jobs. This parameter shows a higher Pareto principle or "mice and elephants" phenomenon, where a small number of jobs (9%) account for 91% of the tasks and all of the other 91% of jobs account for only 9% of total tasks.

Considering Fig. 3.8 and Fig. 3.9, we can find that 90% of tasks include 100 instances and 75% of instances have the length of less than 100 seconds. From that, it can be inferred that about 70% of instances have been executed in less than 3 hours. However, the remaining 30% of instances that are located in system for more than 3 hours are highly computing-intensive instances.

Comparing the three plots in Fig. 3.8, Fig. 3.9 and Fig. 3.10, it can be seen that the distribution of the number of tasks per job is extremely non-uniform and a large majority of jobs only account for very few tasks. Moreover, considering the area between two curves, the larger is this area, the more likely that the distribution

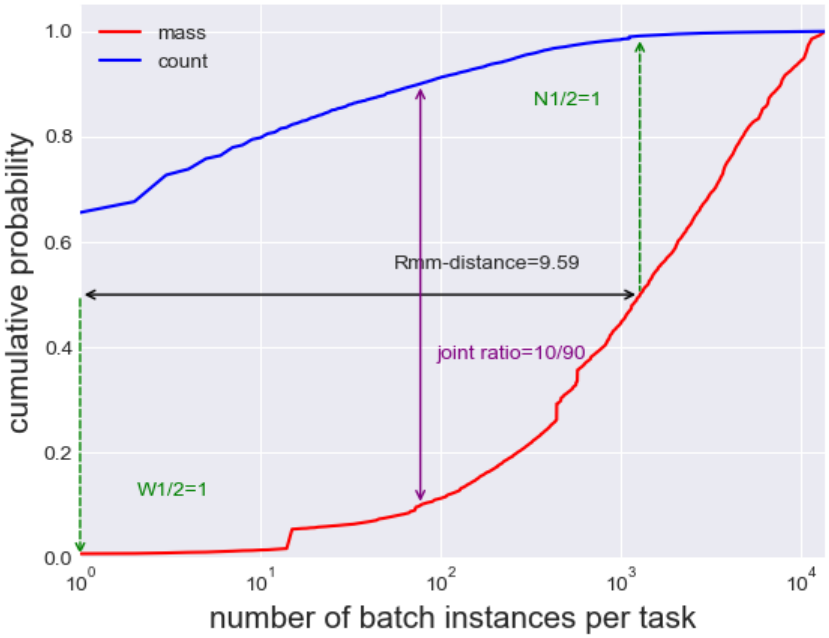


Figure 3.9: Mass-count analysis of number of instances per task

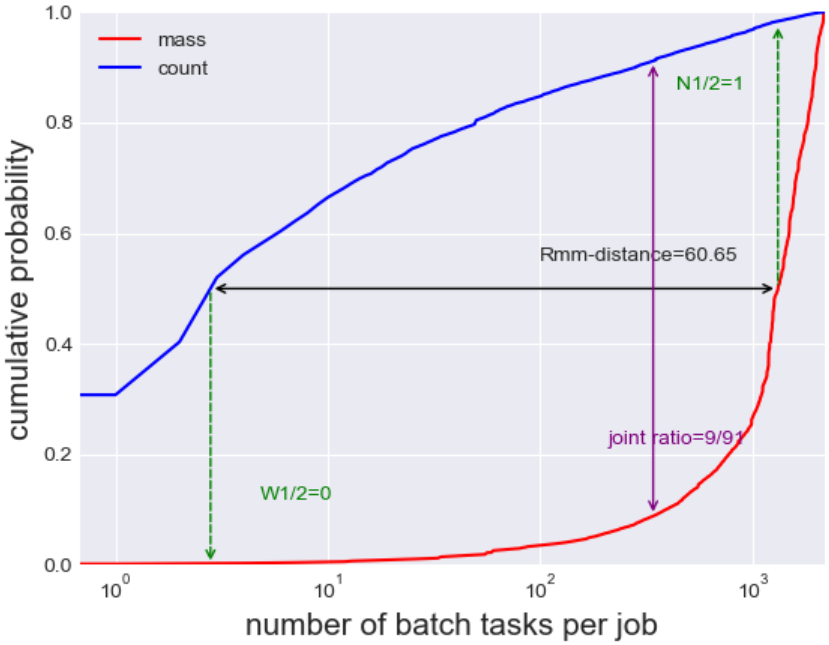


Figure 3.10: Mass-count analysis of number of tasks per job

is power-law. Therefore, the distribution of number of instances per task in Fig. 3.9 and number of tasks per job in Fig. 3.10 are expected to show power-law features.

3.3.3 Comparison of Alibaba with Google workload

The other metrics we studied are per task cpu usage and memory usage, using mass-count disparity analysis for both Alibaba and Google dataset in one week period. The utilization of service instance is presented as a fraction of the requested resource. The maximum utilization is 100 (full usage). For instance, if a task's cpu usage is 10, it means it's using 10% of requested cpu.

In terms of joint ratio, Alibaba tasks shows more Pareto characteristics; and as reflected in the shape of the plots for Alibaba, its usage distribution is more heavy-tailed. However, we cannot see any such tail heaviness in Google task usage.

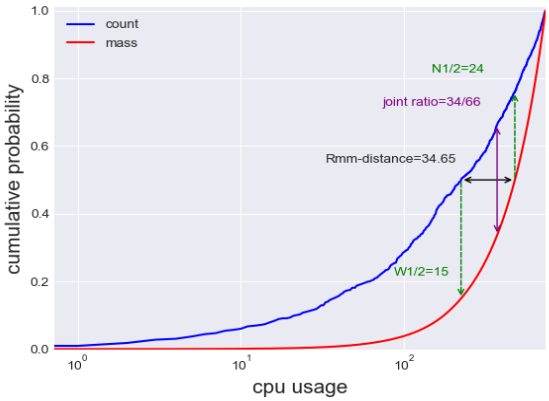
We can also see the 0/50 rule for Alibaba plots. As shown in Fig. 3.11b and 3.11d, half of the tasks are so small that together they account for a negligible fraction of the cpu/memory. At the same time, half of the cpu/memory is occupied by a very small fraction of tasks, which are very large.

Mainly, Alibaba was more successful regarding the resource planning in comparison to Google. As shown, 70% of Google tasks require more than planned cpu(usage is more than 100%). However, in Alibaba datacentre, only for 10% of tasks, the planned cpu in not adequate. This condition is less severe for memory utilization where 20% of Google tasks and 10% of Alibaba tasks require extra memory. Based on this observation, cpu provisioning seems to be more critical and problematic for datacentres.

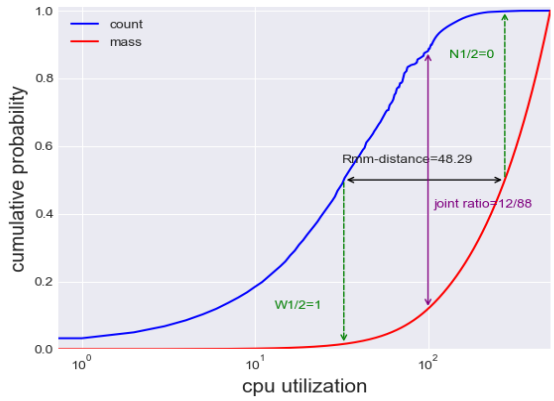
3.4 Summary

It's a big challenge for cloud management middleware to handle heterogeneous workloads. They need techniques to maximize mixed workload performance while providing service differentiation based on high-level performance goals.

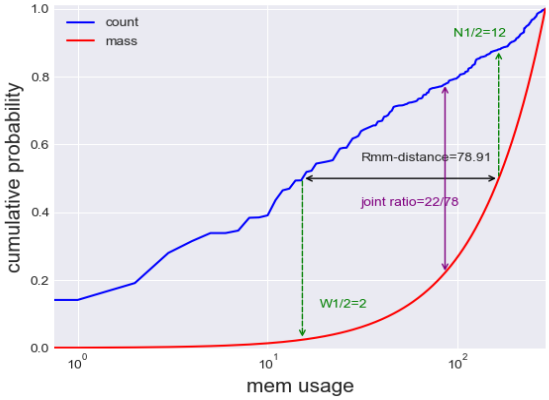
In this chapter, we computed the valuable statistics about job events and resource utilization for Alibaba dataset, based on various types of resources (such as CPU, memory) and execution types. Number of tasks per jobs have been observed with an extremely typical Pareto principle. Using mass-count disparity analysis allowed us to focus on the portion of workload that is responsible for the most of the mass, and



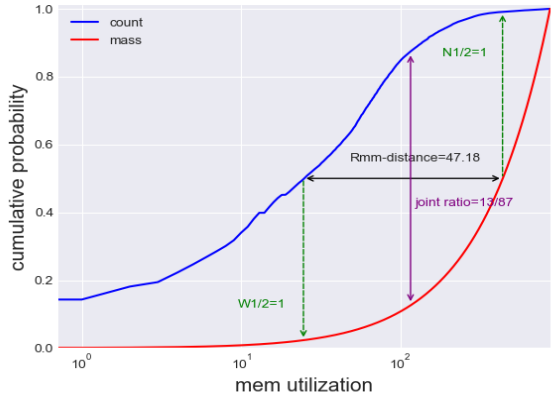
(a) Mass-count disparity for Google task cpu usage



(b) Mass-count disparity for Alibaba task cpu usage



(c) Mass-count disparity for Google task memory usage



(d) Mass-count disparity for Alibaba task memory usage

Figure 3.11: Comparison of Google and Alibaba Workload

disregard the portion that has less effect.

On the other hand, we applied a clustering based demand profiling to the incoming tasks. We classified instances via a hierarchical clustering algorithm with optimized number of sets, based on the extra variable we defined, namely demand. In the end we had three demand profiles for the workload. And for each demand type, we found the arrival rate over time from the Alibaba one-day dataset. Using these insights, cloud providers can allocate resources in a better way to the batch users in order to meet their performance requirements. At the same time, they can set price models for their resources in order to improve the total revenue for the system.

Chapter 4

A Decision Making Framework for Allocating Spot Instances to Execute Batch Workload in Cloud

4.1 Introduction

Among all available cloud-based solutions, spot instances are ideal for batch job execution with flexible completion times and failure tolerance. The combination of spot instance sizes and prices provides flexibility to execute jobs satisfactorily with regards to their computational requirements. However, the demand for each type of batch instance fluctuates over time and it raises some challenges to provide adequate instances to match the workload demand. This makes it necessary to have a mechanism to dynamically adjust the number of instances and assign them to the incoming jobs.

In this chapter, we propose a framework for allocation of resources (Spot instances) to execute arriving batch jobs in order to let providers allocate their resources more efficiently. It also reduces the cost of computing resources for users, while providing their required Quality of Service (QoS). In other words, it manages the trade-off between the cost of renting spot instances and the user-centric quality of service in terms of job waiting time. Given a set of jobs and spot instances of various types, our algorithm selects the best combination of instances in each analysis window. In addition, the method takes into account the job arrival prediction and characteristics of input jobs to adapt the algorithm.

This whole process consists of the following steps. First, incoming batch requests

are profiled based on their resource utilization. For each job profile, we extract the arrival rate per time and then predict future demand using machine learning and deep learning models which have been elaborated in Section 4.4.1. Based on future prediction, we assign the configuration of resources to the jobs with the help of an optimization problem. The goals here are maximizing the profit and increasing customer satisfaction by improving quality of service for users in terms of job waiting time.

4.2 System Model

The proposed framework consists of different modules which have been described in detail below. We also depicted these model in Fig. 4.1

Module 1: (Online monitoring): This module monitors and measures different system parameters, such as job's arrival time, start and end time, jobs' resource utilization and number of available resources in the pools. These parameters are used to model the system and to characterize the workload.
Module 2: (Workload profiling): In this module, the clustering is performed offline on the historical data stored by the monitoring module. The number of job arrivals for each cluster is extracted and sent to the next module for prediction.
Module 3: (Future workload forecasting): This module is responsible for predicting the number of arrivals per job class for the next time steps.
Module 4: (Decision making): This module dynamically executes the algorithm to decide on the allocation of resources to the incoming jobs. To do this, the measured metrics, workload characteristics and arrival prediction would be taken into account.

4.3 Problem Definition

For batch jobs, the Quality-of-Service is usually determined by job waiting time. Because the amount of time a user's job will wait in any one batch queue can significantly impact the overall time a user waits from job submission to job completion. Waiting time can be estimated by using queuing analysis by modeling the computing system as a multi-server queue (M/M/c) for each resource pool as shown in Fig. 4.2.

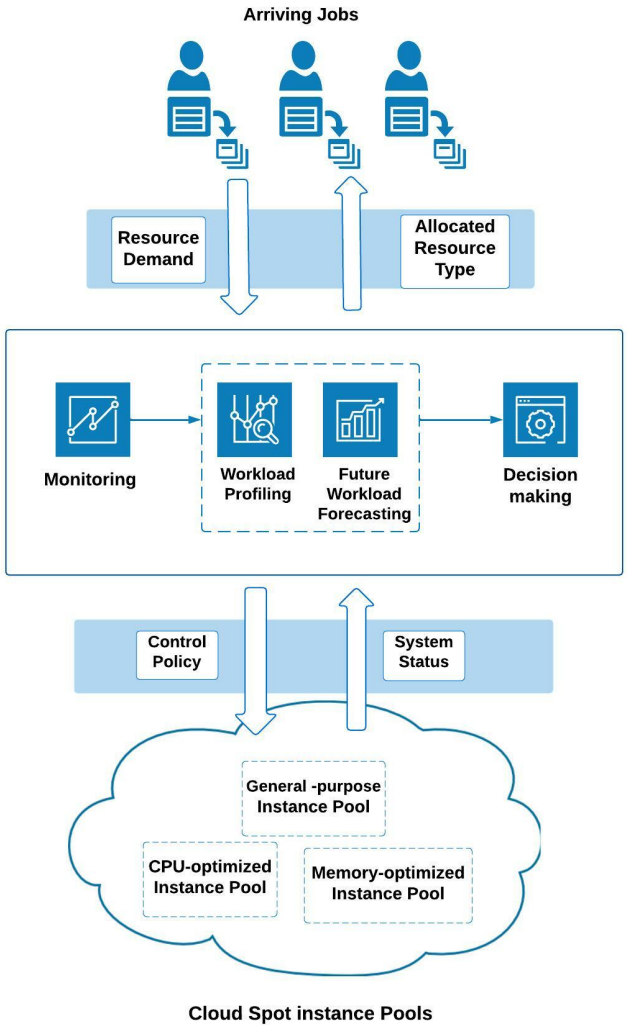


Figure 4.1: System Model

There is a separate spot instance pool for each VM category. As for the implementation, the spot instance pools are created from the resources which are not being used by dedicated instances. We assume only large and extra-large instances are available in each pool. More details about this queue system will be explained in later sections. Each instance type has a different price and availability. We aim to find a mechanism to manage the trade-off between the cost of renting spot instances and the user-centric quality-of-service for arriving jobs (in terms of job waiting time). In the next sections, we will provide the details of this mechanism and its formulation.

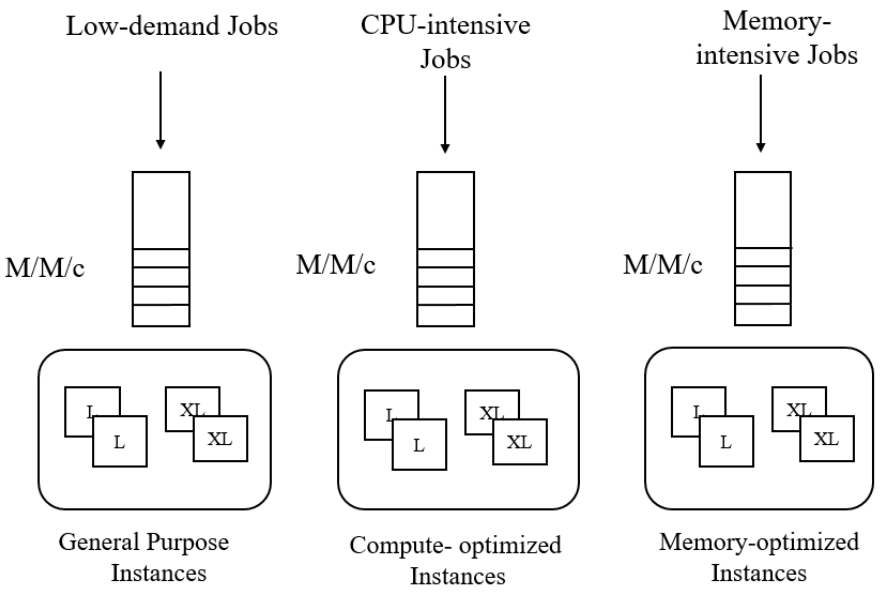


Figure 4.2: Queue Model of Resource pools

4.4 Data Analysis and Prediction

In this section, we discuss the data analysis and prediction on Alibaba datacentre trace [4]. Alibaba datacentre keeps track of batch and online processes separately on different log files. Batch workload has been submitted to the system in the form of jobs. A batch job contains multiple tasks which execute different computing logic. Within a task, there are several instances. Instances are the smallest unit of the batch job which execute the same function on different input data. Batch workload in Alibaba trace is described by two tables: task table and instance table. In the task table, information about creation and modification time, number of instances per task and requested resources for each instance is given. We can find information about creation and completion time of instances, their current status and the actual resource utilization in the instance table. In order to help with the decision making, we classified batch workload by using K-means clustering algorithm. The methodology for clustering follows the previous method we had utilized in [47], except in this case, we only considered two attributes, CPU utilization and memory utilization. To find the optimal number of clusters, Silhouette analysis has been applied for cluster numbers in the range [2, 7]. In this method, the quality of clustering is measured in terms of average distance between clusters. The silhouette plot shows how close each

Table 4.1: Silhouette Score for Different Number of Clusters

Number of Clusters(k)	Average Silhouette Score
2	0.58
3	0.61
4	0.42
5	0.45
6	0.50
7	0.43

point in one cluster is to points in neighboring clusters. Silhouette score is formulated as below:

$$SilhouetteScore = \frac{(x - y)}{max(x, y)} \tag{4.1}$$

Where x is the nearest cluster distance and y is mean intra-cluster distance. The score varies between -1 and 1. A value close to +1 means that the instance is far away from neighboring clusters. Score 0 indicates that the instance is on the boundary of two clusters and negative scores means that the instances assigned to the incorrect cluster [48]. The silhouette score calculated for each cluster number is presented in Table 4.1. As shown, the score is maximum when the number of clusters is 3 (k=3). Fig. 4.3 depicts silhouette plots for (k=3). As we can see, there is no cluster with average silhouette score below 0.61, that confirms our decision to choose 3 as the optimum number of clusters.

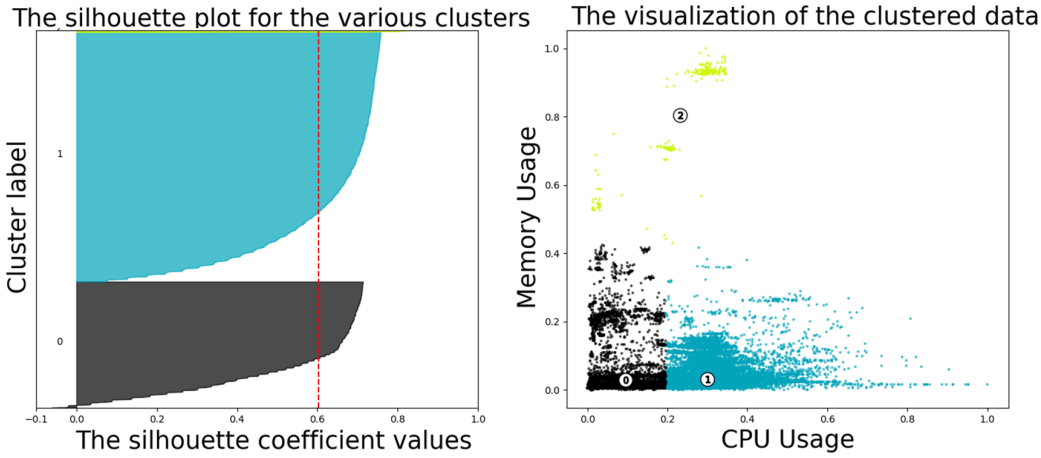


Figure 4.3: Silhouette analysis for K-means clustering with 3 clusters (K=3) on sample data

In Fig. 4.3, we also show the dispersion of normalized clustered instances on a 2-D space. Cluster number 0 includes instances with relatively low CPU and memory usage. In cluster 1, instances have higher CPU usage and below average memory usage. For the third cluster memory usage is very high while CPU usage is low. We name these three clusters as low-demand, CPU-intensive and memory-intensive instances, respectively. Furthermore, it should be noted that the number of instances grouped into each category varies. The majority of instances belong to CPU-intensive class (60%), while just a small percentage (3%) are memory-intensive and the remaining is low-demand instances. The un-normalized version of samples is also depicted in Fig. 4.4 in order to give real sense of resource usages. As mentioned in the Alibaba dataset schema, CPU usage is measured as percentage, which means 100 is one core. So, the maximum number of cores has been used by an instance is around 3. Memory is also normalized to the largest memory size of all machines in the Alibaba Cluster, which is 384GB. So, the maximum memory usage for batch instances is around 16GB.

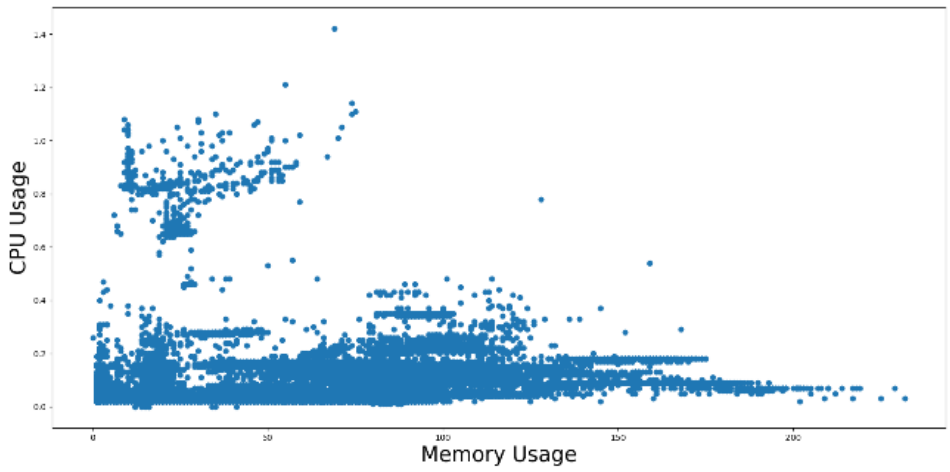


Figure 4.4: Un-normalized resource usage for instances in Alibaba Cluster

In the next step, we calculated the average number of arrivals for every 15 minutes from each cluster as a time series. We are going to use these time series to predict the future number of arrivals of each class.

Instead of providing a single estimate of future arrival rate in the next time steps, we decided to provide a prediction interval for arrival rate. This prediction method is called quantile regression. As described by Koeneker and Hallock [2], the standard

approach in regression is to estimate the conditional mean of the data by drawing a line which minimizes distance between the line and all of the points on the graph. In contrast, the goal in quantile regression is to estimate the conditional median, or any other quantile. To do so, we instead weight the distances between points on the graph and our regression line based on the selected quantile. For example, if we selected the 90th quantile to estimate, we'd fit a regression line so that 90% of the data points are below the line and 10% are above. So, in overall, single point prediction minimizes the squared error loss function, but in the quantile regression method, the goal is to minimize the quantile loss for a certain quantile. Quantile loss is a function which varies with the prediction quantile. For higher quantiles, more negative errors are penalized more while for lower quantiles, more positive errors are penalized. Quantile loss function is defined as:

$$Quantile\ loss = \begin{cases} \alpha.\varepsilon & if\ \varepsilon \geq 0 \\ (\alpha - 1).\varepsilon & if\ \varepsilon < 0 \end{cases} \tag{4.2}$$

Where α is the required quantile and has a value between 0 and 1. And ε is the prediction error, defined as the difference between predicted value (y) and real value $f(x)$ [49].

$$\varepsilon = y - f(x) \tag{4.3}$$

4.4.1 Prediction Models

We have selected a wide variety of linear and tree-based machine learning methods and also deep learning techniques for time series prediction. First, we train these models on 80% of the time series data and then tested on the remaining 20%. The figures 4.5 to 4.10 show the trained model applied on the entire time series data. Below, we explain each model in details.

- **Ordinary Least Square:** Ordinary Least Square (OLS) is the most commonly used statistical procedure for regression analysis which minimizes the sum of squared errors. The methods draws a straight line as close as possible to the data points. Although it predicts the mean rather than the median, we can still calculate prediction intervals from it based on standard errors and the inverse normal CDF: In this study, we use OLS as a baseline approach and it creates linear and parallel quantiles around the mean as shown in Fig. 4.5.

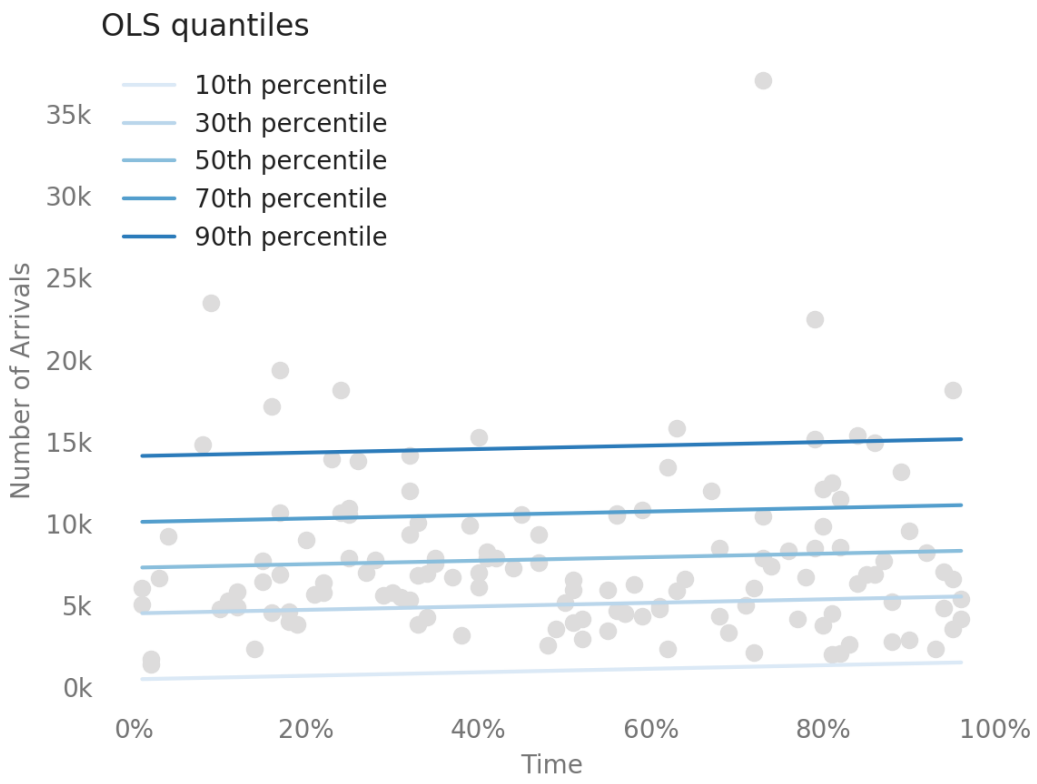


Figure 4.5: Prediction with Ordinary Least Square

- **Linear Quantile Regression:** Linear quantile regression is similar to least-squares regression in that both look into the linear relationship between a response variable and one or more independent or explanatory variables. However, whereas least-squares regression is concerned with modelling the mean of the response variable, quantile regression models the median and other quantiles of the response variable and minimizes quantile loss. As quantile regression allows multiple quantiles to be modelled it can allow for a more comprehensive analysis of the data to be carried out compared to least-squares regression where only the mean is considered. This potentially enables more insight into the data and any underlying relationships, in addition, it will tend to be less sensitive to large outlying observations. As shown in Fig. 4.6, quantile regression does not have OLS's parallel trend assumption, while still imposing linearity.

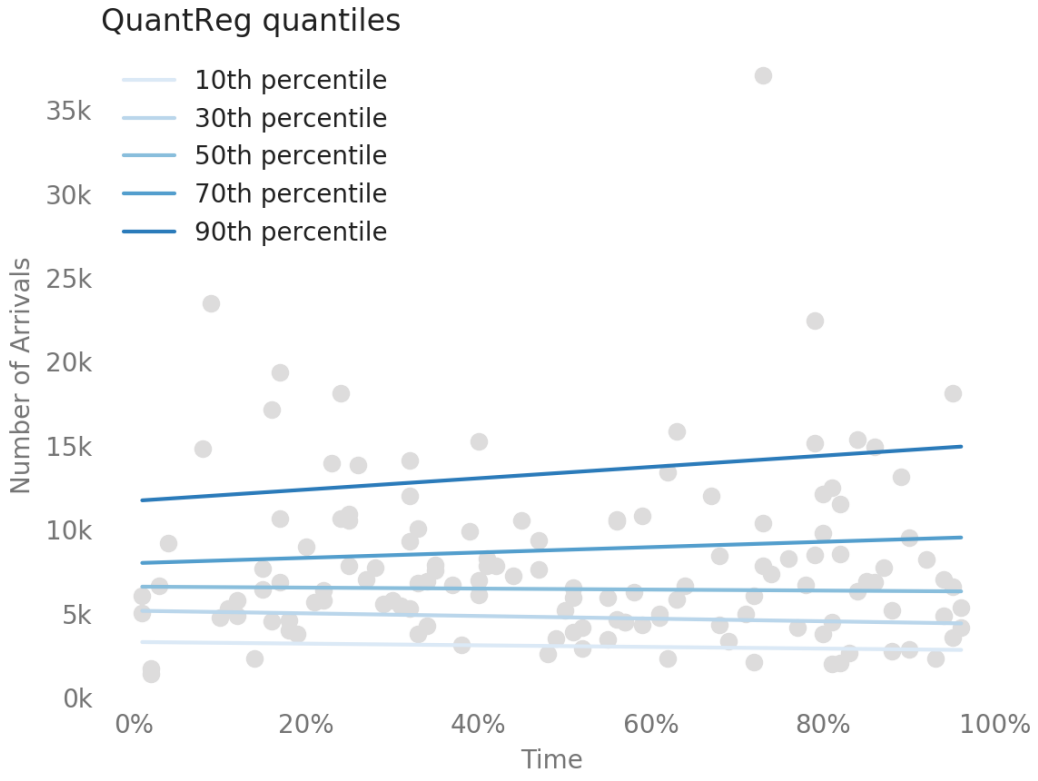


Figure 4.6: Prediction with Linear Quantile Regression

- **Random Forest:** Unlike two previous methods, Random Forest is nonlinear model which generally does not predict quantiles. However we can get prediction quantiles from it with some adjustments. The idea is that instead of recording

the mean value of response variables in each tree leaf in the forest, record all observed responses in the leaf. The prediction can then return not just the mean of the response variables, but the full conditional distribution $P(Y \leq y|X = x)$ of response values for every x. Using the distribution, the prediction intervals for new instances can be simply created by using the appropriate percentiles of the distribution. Fig. 4.7 is a bit jumpy which can be a sign of overfitting. That's because random forests are more efficient when used for high-dimensional data.

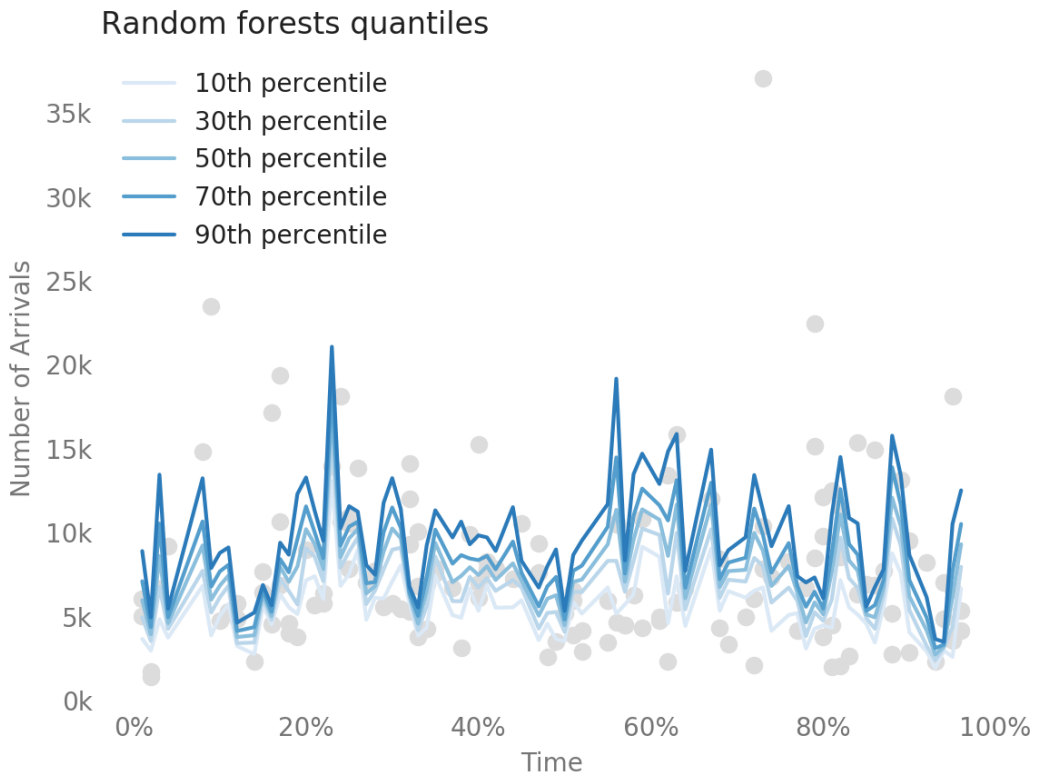


Figure 4.7: Prediction with Random Forest

- **Gradient Boosting:** Gradient Boosting is based on an optimization algorithm called Gradient Descent. In each iteration, it builds many small models like decision trees to reduce the errors from previous round sequentially. Although, the standard gradient boosting only provides the mean prediction, the algorithm can be used to create prediction intervals if we set the loss function(the function optimized by the model) as quantile. Then we're able to get predictions corresponding to percentiles. Fig. 4.8 demonstrates better prediction compared

to random forest. However, Gradient Boosting tends to produce intervals that are very distant from real values, as we can see for the 90th percentile.

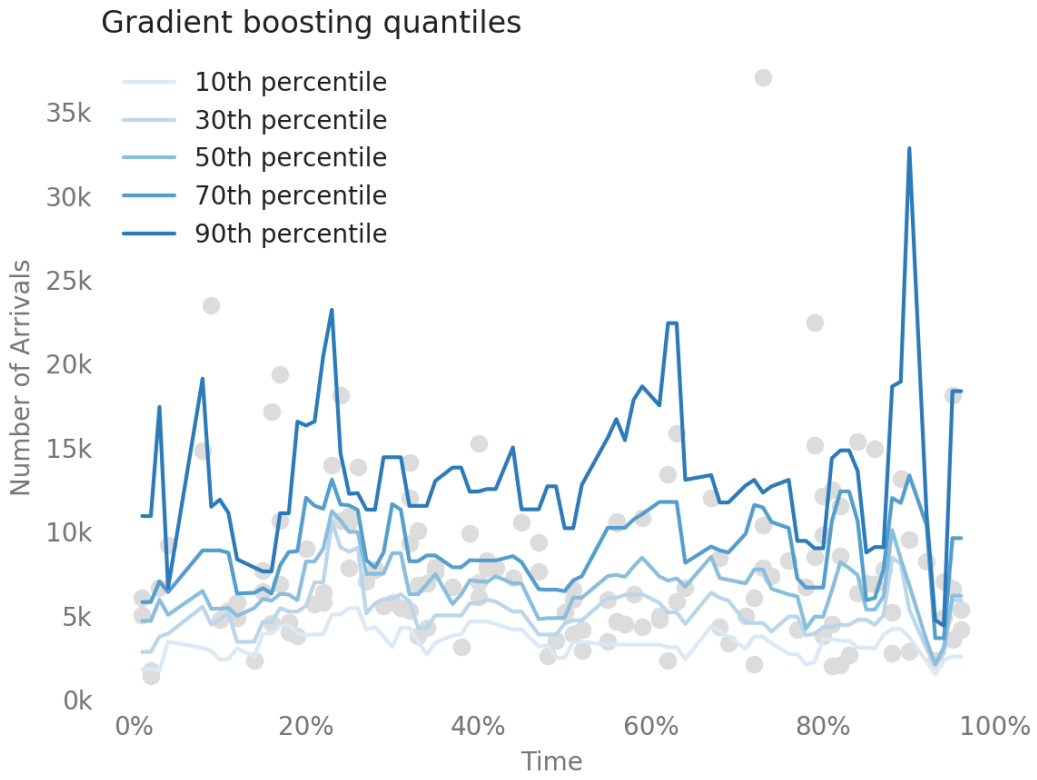


Figure 4.8: Prediction with Gradient Boosting

- **Keras:** The first deep learning model we have used is Keras. We can use neural networks to predict quantiles by passing the quantile loss function. The quantile prediction result shows more like linear behaviour as shown in Fig. 4.9.
- **TensorFlow:** The last method has been used is TensorFlow, another deep learning model. despite Keras, which requires seperate fitting of models per quantile, with Tensorflow, we can fit any numer of quantiles simultaneously and there would be a common learning across the quantile predictions as shown in Fig. 4.10.

To evaluate the prediction using aforementioned methods, for each cluster data, we calculated per quantile loss and also the average loss over all quantiles. The losses have been separately compared for each cluster in Fig. 4.13. Since we had five quantiles, we have five quantile losses for each observation. The plot reveals that

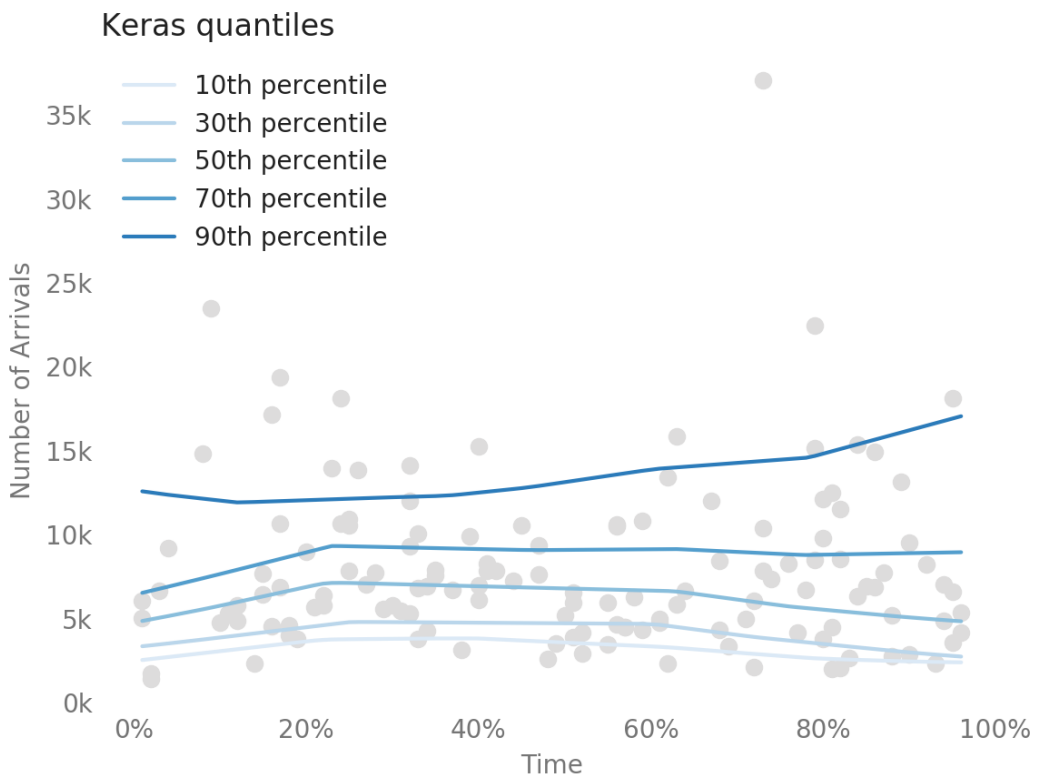


Figure 4.9: Prediction with Keras

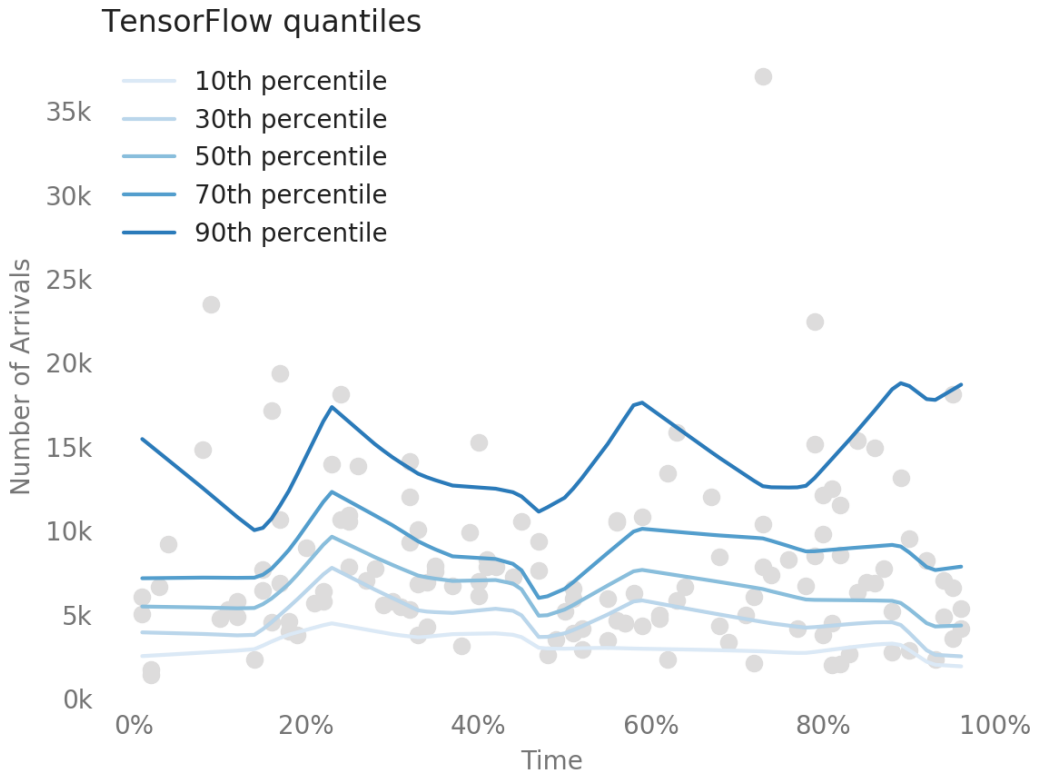


Figure 4.10: Prediction with Tensorflow

tree-based models like random forest had a poor performance for any quantile, but gradient boosting did its best for lower quantiles. Also based on the total quantile loss, it seems that gradient boosting method has the least quantile loss on the average for all clusters.

4.5 Resource Allocation

In this section, we introduce an optimization-based solution to allocate resources, in terms of spot and On-demand instances, for executing batch jobs arriving at the cloud. The clustering step was performed offline on historical data and we then predicted the number of arrivals for each job class for the next time step. However, the resource assigning decision is made on-the-fly, based on parameters that are constantly monitored, measured or calculated in every analysis window. All system parameters and decision variables that are involved in the problem are given in Table III.

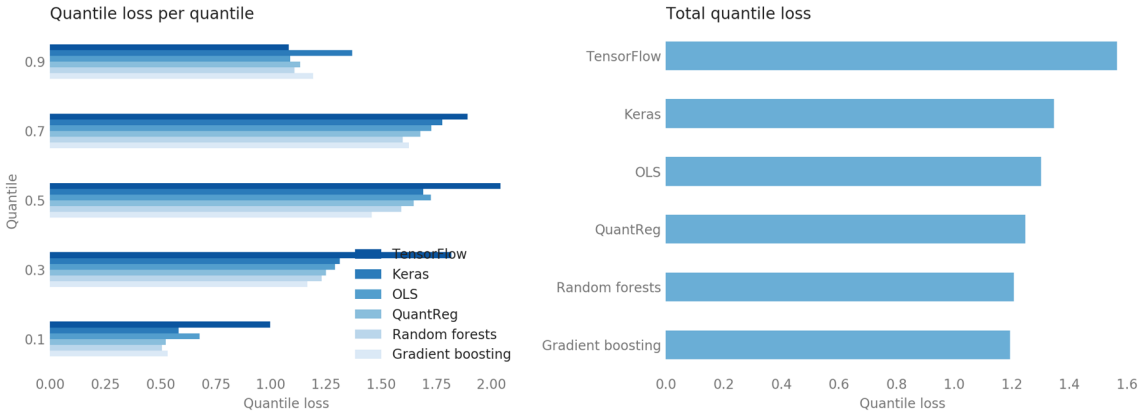


Figure 4.11: Quantile loss and total loss for cluster 1

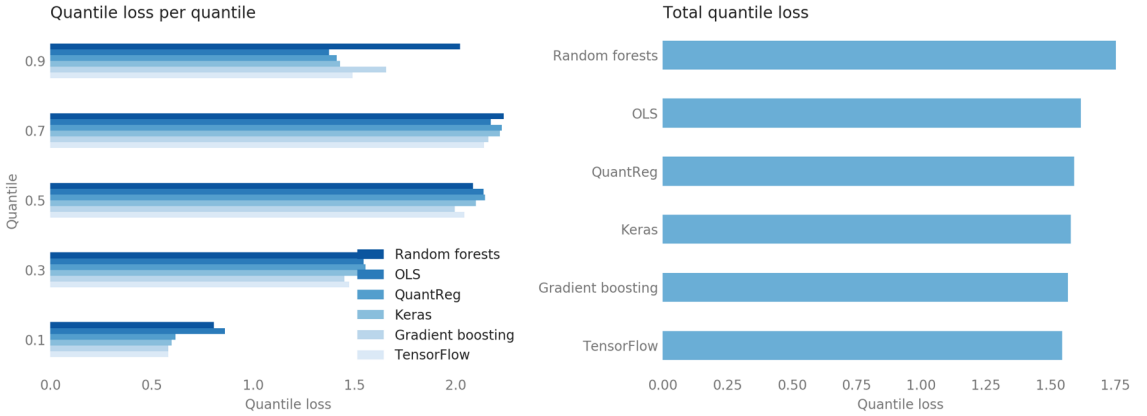


Figure 4.12: Quantile loss and total loss for cluster 2

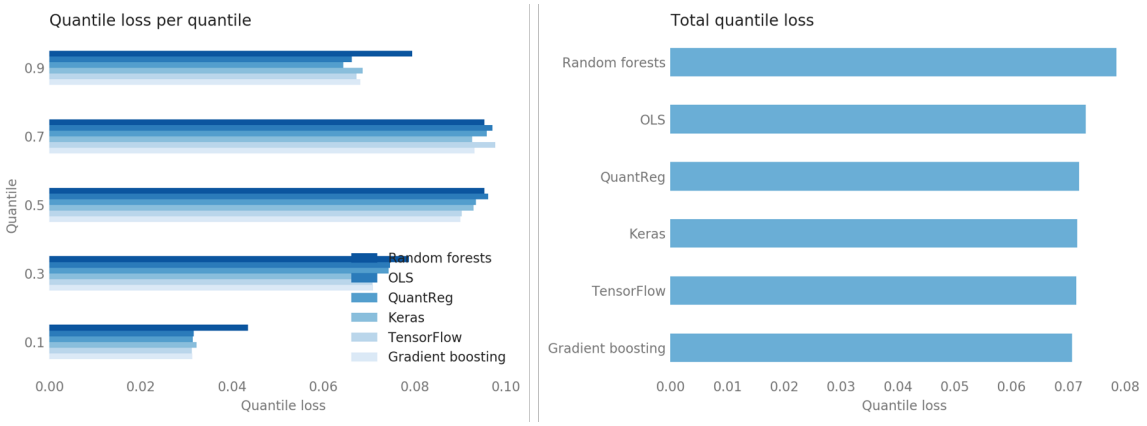


Figure 4.13: Quantile loss and total loss for cluster 3

Table 4.2: System Parameters and Decision Variables

Variable/Parameter	**Description**
τ	Analysis window
c	Number of job classes
t	Number of resource types
C_r	CPU requirement for each job class
M_r	Memory requirement for each job class
C_s	CPU of each instance type
M_s	Memory of each instance type
N	Number available spot instances from each type
P_s	Spot instances price
P_d	On-demand instances price
Q	Number of requests at the beginning of analysis window
L	Number of jobs arriving at each queue
A	Number of arriving jobs of each class
W	Queue waiting time
$S_{c \times t}$	Allocation Matrix

In each analysis window (τ), we have a number of arriving jobs belonging to different classes where each job needs a combination of resources to be executed. An optimization problem has been devised to find the optimum number of resources for the arriving jobs to meet their requirements. We discuss that in the next section. The complete resource assignment algorithm is given in below.

Algorithm 1 Optimization-based Resource Assignment Algorithm

Input: Set of job requests with their resource requirements
Output: Optimal number of spot instances, on-demand instances and the cost

procedure *JobClassification*
  $c \leftarrow$ *number of job classes*
  $C \leftarrow$ *CPU requirement of job classes*
  $M \leftarrow$ *Memory requirement of job classes*
end procedure
procedure *Prediction*
  $A \leftarrow$ *predict number of arrival jobs in the next time step*
end procedure
procedure *ResourceAssignment*(A)
  for $\tau \leq T$ **do**
    $P_s \leftarrow$ *Price of spot instance type at the moment*
    $P_d \leftarrow$ *Fixed price of on-demand instance type*
    $N \leftarrow$ *Number of available instances in the pool*
    $Q \leftarrow$ *Queue length of each resource pool*
    $S \leftarrow$ *Optimal number of allocated spot instances with solving Opt. problem*
    if $S \leq N$ **then**
      $ConfigurationCost \leftarrow S \times P_s$
    else
      $D \leftarrow S - N$
      $ConfigurationCost \leftarrow S \times P_s + D \times P_d$
    end if
    return $ConfigurationCost$
    $\tau \leftarrow \tau + 1$
  end for
end procedure
call $JobClassification$
call $Prediction$
call $ResourceAssignment$

4.5.1 Optimization Problem

We have developed a discrete optimization function. By solving the optimization problem, we aim to minimize the negation of profit, that is summation of cost and waiting time, such that total number of resources assigned from each type should be less than or equal to the number of available resources. Total CPU and memory allocated to a job should satisfy the jobs' requirements. Waiting time in the queue (W) is a function of queue length and the queue length itself is the total number of jobs arriving to each queue in each window (L) in addition to the number of jobs that are already in the queue (Q).

The objective function is defined as below:

$$min \sum (\alpha \times \ cost + \beta \times \ waiting\ time) = min \sum_{i=1}^{t} \alpha P_i L_i + \beta W_i \tag{4.4}$$

Where:

$$W_i = k(Q_i + L_i) \qquad \text{for i} = 1, 2, \ldots, \text{t} \tag{4.5}$$

$$L_i = S_{ij}.\ A_j \qquad \text{for i} = 1, 2, \ldots, \text{t} \tag{4.6}$$

Subject to:

$$S_{ij}.A_j \leq N_i \qquad \text{for i} = 1, 2, \ldots, \text{t} \tag{4.7}$$

And:

$$Cs_i.S_{ij} \geq Cr_j \tag{4.8}$$

for all jobs in low demand and CPU intensive class

$$Ms_i.S_{ij} \geq Mr_j \tag{4.9}$$

for all jobs in low demand and Memory intensive class

$$S_{i,j} \in \{0, 1\} \tag{4.10}$$

4.5.2 Solving optimization problem

The only variable which we have control over is the assignment matrix $S_{i,j}$. The elements of this matrix can only be zero or one. One means that the job class in the equivalent column can be assigned to the resource type in the corresponding row. Besides, the objective function and all the constraints are linear. All of this conditions would change the optimization problem as a linear Integer programming problem, where the complexity is NP-hard [50]. So, the feasible solution for these type of problems are approximate solutions.

Among various methods of solving a discrete optimization problem (shown in Fig. 4.14), we have selected three methods. The Branch and Bound method and two metaheuristic methods: Simulated Annealing (SA) and Genetic Algorithm (GA). The advantage of this family of solutions is that they aim to escape the local minima and drive the search towards a global minimum. The details of each method are explained below.

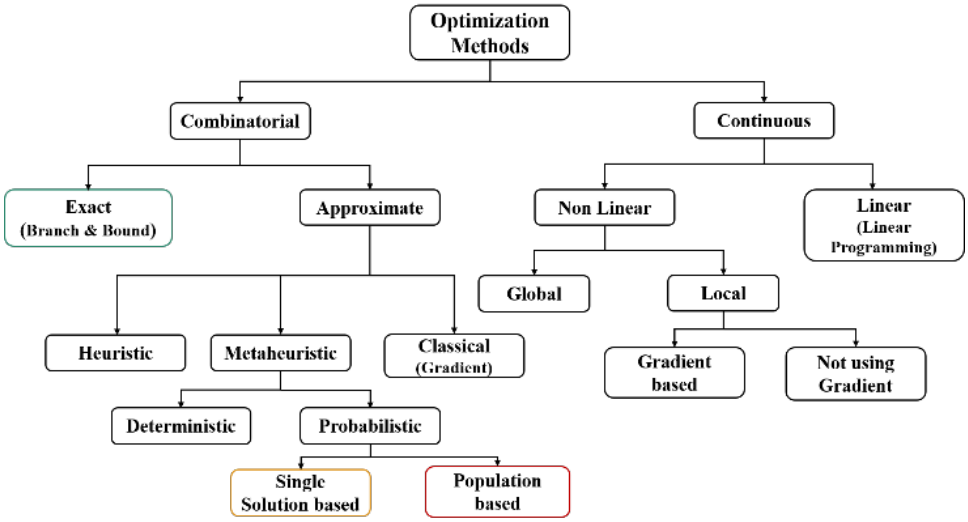


Figure 4.14: Methods for solving a discrete optimization problem[51]

1. **Branch and Bound:** Branch and Bound is first proposed by Land and Doig [52]. It is the most common general strategy for discrete optimization problems. This exact method is suitable for a general integer programming problems and can be tailored for different problem classes. The method is based on a "divide and conquer" principle: the problem is partitioned or divided into subproblems and bounds for the objective value are calculated for each subproblem. Branch

and Bound method searches the complete space of solutions for the optimal solution for a given problem. All candidate solutions will be divided into several subsets of solutions. Then we can systematically evaluate these smaller subsets till the best solution is found.

2. **Simulated Annealing:** Simulated Annealing is a single solution based metaheuristic method. Kirkpatrick et al. [53] introduced the idea of simulated annealing to the field of combinatorial optimization. The key idea of this approach is using metropolis model that shows how in the (physical) annealing process, particles of a solid arrange themselves into thermal equilibrium at a given temperature. In the optimization problem, there is a controlling parameter called temperature. It starts from a high temperature (it's like a random walk) and decreases progressively. When the temperature is low it is like a greedy search. The advantage of Simulated Annealing is that it allows "hill-climbing" moves in order to avoid local optimal solutions. The algorithm starts with a arbitrary state. In the search space, moving from a solution to its neighbor is 100% acceptable, if the move improves the objective function, and if not, it is acceptable with some probability. This probability depends on how bad the move is. The pseudocode for Simulated Annealing is given in the Algorithm 2 below [54].

3. **Genetic Algorithm:** Genetic Algorithm has been developed by John Holland and his collaborators in 1975 [55]. It is a population based metaheuristic method which uses techniques inspired from evolution to find an optimal or near-optimal solution towards a problem. Like in evolution, genetic algorithm's process is random, however the level of randomization can be changed. First, a population of chromosomes are initialized randomly and the fitness of each chromosome in population is computed. Based on the fitness value, two chromosome are selected and crossover operator is applied on them. After that, the uniform mutation operator is also applied with a specific probability to produce an offspring. The new offspring after evaluation would be replaced to the population. This process would be repeated until new population is generated. Also, different stopping criteria can be used and it's highly problem specific. First, when we reach maximum (predefined) number of iterations (generations), the algorithm will stop and give the best solution. Second, the algorithm will proceed until there has been no improvement in the solution and it reaches to a predefined convergence. A combination of these criteria can be used too. We

Algorithm 2 Simulated Annealing Algorithm

Input: Temperature T
Boltzmann's constant k
reduction factor c

Output: the best solution

Initialize parameters
Generate the initial solution S
$bestSolution \leftarrow s$
while $T < T_{final}$ **do**
  $iter \leftarrow 0$
  while $iter < MaxIterations$ **do**
    Generate solution S' in the neighborhood of S
    $Delta \leftarrow f(S')$
    if $delta \leq 0$ **then**
      $S \leftarrow S'$
      if $f(S') \leq f(BestSolution)$ **then**
        $BestSolution \leftarrow S'$
      else$r = \text{Random}(0,1)$
        if $r < \exp \frac{-delta}{k \times T}$ **then**
          $S \leftarrow S'$
        end if
      end if
    end if
    $iter \leftarrow iter + 1$
  end while
  Decrease the temperature periodically $T \leftarrow c \times T$
end while

show the pseudo-code of genetic algorithm in below [56].

Algorithm 3 Genetic Algorithm

Input: *Population of individuals*
Probability of crossover(P_c)
Probability of mutation(P_m)

Output: *the best individuals*

$t \leftarrow 0$
Create an initial population (Pop(0))
for *each individual in the population* (P_0) **do**
    Evaluate individuals using fitness function
end for
while *stop condition reached* **do**
    Select individuals for new population Pop(t)
    Perform crossover operation with probability of crossover (P_c)
    Perform the mutation of individuals with probability of mutation (P_m)
    Evaluate individuals
    Replace old population with new ones
    $t \leftarrow t + 1$
end while
return the best solution in population

4.6 Simulation Scenario

In this section, we want to utilize the prediction results for assigning the resources by solving an optimization problem. In a scenario, we assume that every batch job consists of 100 batch instances. Accordingly, the average CPU and memory usage for three classes of jobs would be as shown in Table 4.3. For the test, we used Amazon EC2 instances as the target resources. The Linux instance types and prices for the US-east region for a specific time of the day (Feb. 17, 2020, 3pm PST) are available in Table 4.4.

We have considered two criteria for evaluation of our algorithm.

1. Quality of solution: How close is the solution to the optimal solution? Is the quality dependent on the size or scale of the problem?

2. Processing time: or time consumption of algorithm that means how long it takes for different algorithms to converge and give the solution?

Table 4.3: Resource Requirements for different types of jobs

Job Class	**CPU(Core)**	**Memory(GB)**
Low-demand	60	40
CPU intensive	300	50
Memory Intensive	100	160

Table 4.4: Characteristics and price of different spot instance types

Category	**Type**	**CPU core**	**Memory (GB)**	**On-demand Price (per hour)**	**Spot Price (per hour)**
General purpose	m5.large	2	8	0.096	0.020
General purpose	m5.xlarge	4	16	0.192	0.0399
Compute-optimized	c5.large	2	4	0.085	0.0202
Compute-optimized	c5.xlarge	4	8	0.17	0.0417
Memory-optimized	r5.large	2	16	0.126	0.0209
Memory-optimized	r5.xlarge	4	32	0.252	0.042

3. Cost of solution: the cost here means how much we have to pay compared to the maximum cost calculated from different solutions. The maximum cost gets 100% and the other ones have been calculated as a percentage of the maximum cost. The cost also can be translated to the energy consumption. Because the bigger resource you have, you are using more energy and paying higher price.

As shown, the on-demand price for some resources is 5 times more than the equivalent spot instances. In addition to the huge difference between the on-demand price and spot price, we can see the price also varies for different types of spot instances. Thus, by assigning the optimized types of instances to the incoming jobs, we can save on computing costs and have superior performance. The cost saving would be particularly significant specially for batch jobs with long run times.

We predict the arrival rate for the next 15 minutes after extracting the number of job arrivals every 15 minutes. We also consider the 90th quantile of prediction, which means that there is a 90% chance that the actual number of arrivals is below the prediction, while there is only a 10% chance that the number is above. The results are presented in Fig. 9 to Fig. 11 below by applying the three aforementioned optimization methods

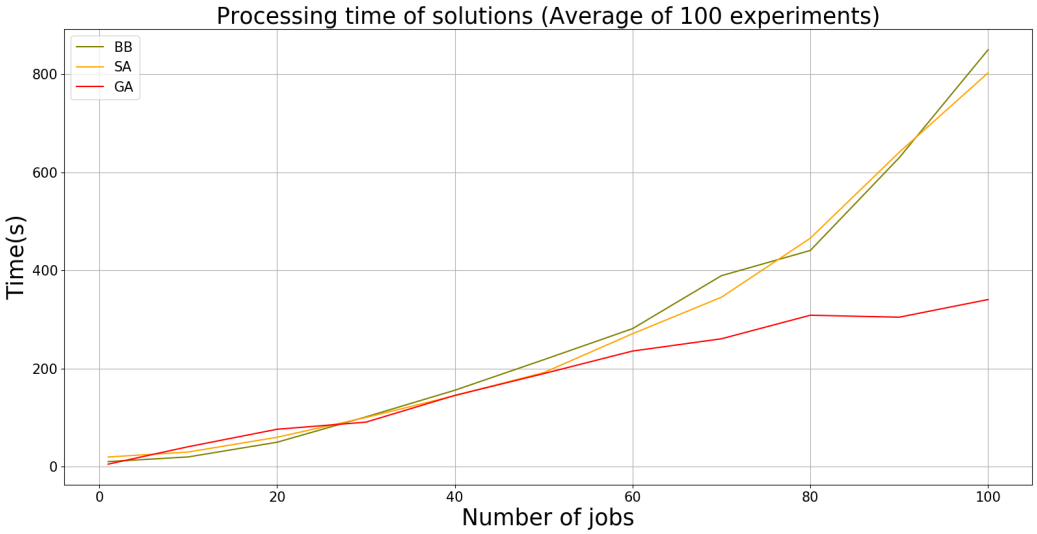


Figure 4.15: Processing time of different optimization solutions

Based on the simulation result, we show that for large scale problems, our meta-heuristic approach outperforms the exact method (Branch and Bound) in terms of

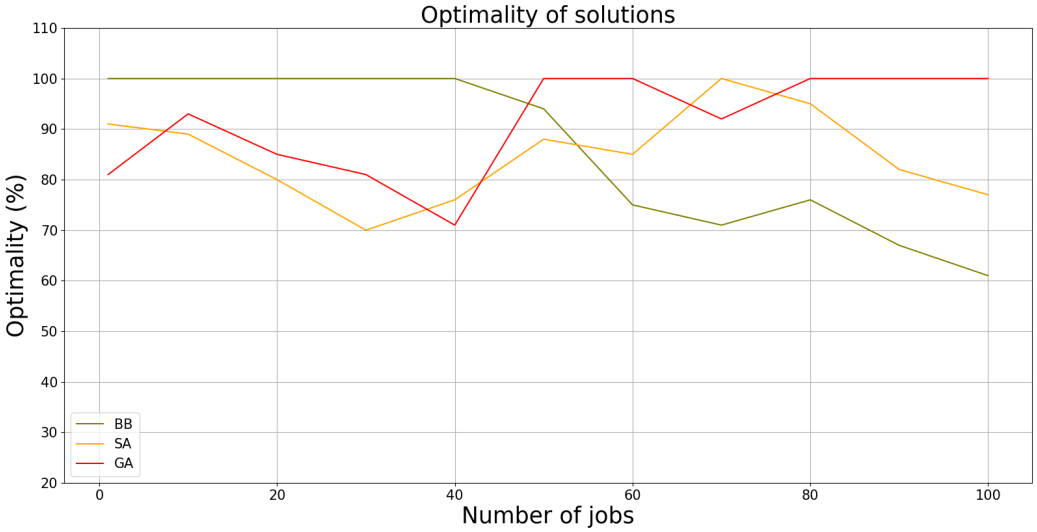


Figure 4.16: Optimality of different optimization solutions

total profit, while providing faster convergence. Additionally, our planning-based approach is able to generate reliable solutions that are cheaper and able to meet quality of service. For small scale problems when the number of jobs increase, the exact method gives the best quality solution in terms of optimality. However, for the larger scale problems, heuristic-based solutions outperform the exact method due to the exact method's inability to find the minimum in a specific number of iterations, consigning it to the local minima. Considering the graph of cost and optimality, there are some cases where the solution gives optimal results, but not the minimum cost. It's because the algorithm considers the waiting time as well. Customers would pay a bit more for faster service. In addition, it takes more for the exact solution to converge. Overall, we show that our Genetic Algorithm approach outperforms the other methods in terms of total revenue, while it provides fast convergence.

We also calculated the complexity of the optimization-based resource assignment algorithms when Genetic Algorithm has been applied. In order to do that, first we need to calculate the complexity of objective function which can be done by counting the number of operations which would be explained in bellow:

$$W_i = k(Q_i + L_i) = k(Q_i + S_{ij}.\ A_j) ===> Total\ Operations = O(tc) + O(t) = O(tc) \tag{4.11}$$

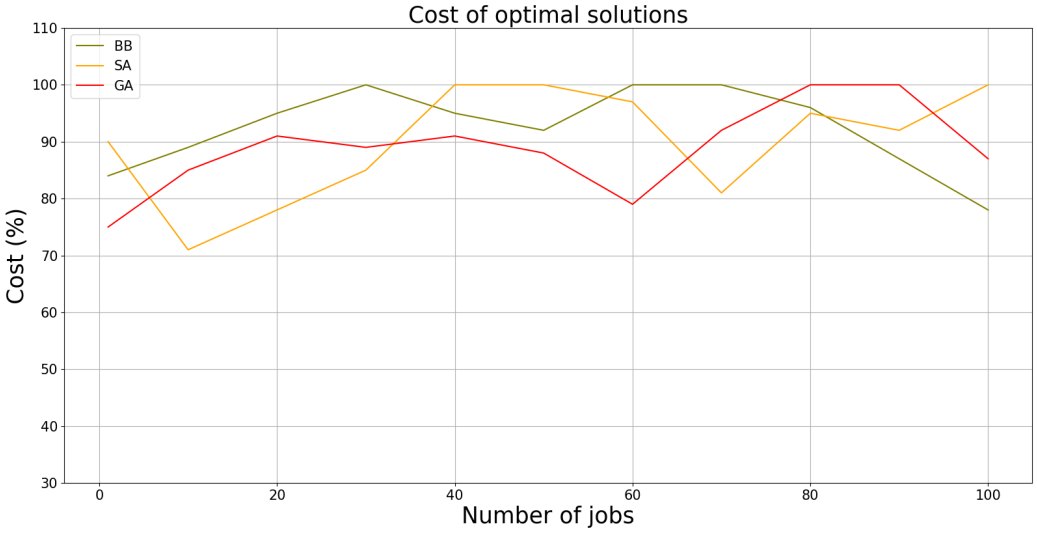


Figure 4.17: Cost of different optimization solutions

$$S_{ij}.A_j \leq N_i ===> Total\ Operations = O(tc) + O(t) = O(tc) \tag{4.12}$$

$$Cs_i.S_{ij} \geq Cr_j ===> Total\ Operations = O(tc) + O(c) = O(tc) \tag{4.13}$$

$$Ms_i.S_{ij} \geq Mr_j ===> Total\ Operations = O(tc) + O(c) = O(tc) \tag{4.14}$$

where t is the number of resource types and c is the number of job classes

The complexity of Genetic Algorithm, then can be presented as complexity of fitness function multiplied by number of population n, and number of generations (iterations) g. Therefore the complexity of this algorithm would be $n \times g \times t \times c$.

4.7 Results and Discussion

Batch jobs mostly require high computational resources but have flexibility in their completion time and have low latency requirements. These characteristics make them a suitable option to run on spot instances which may have interruptions but are priced lower than on-demand instances. This study presented an end-to-end mechanism to allocate spot instances to batch jobs arriving at the cloud. We used real batch workload from Alibaba datacentre trace. After analyzing and clustering the dataset, we found a trimodal behavior in the cloud batch jobs and we grouped them into three distinct classes. Adopting various regression, machine learning and deep learning methods, we predicted the future arrival of incoming jobs separately for each job class. Then we implemented an optimization-based solution for assigning the most suitable type of the spot instances to the batch workload in order to reduce the cost and also meet the jobs resource requirements. An optimal revenue maximizing resource assignment is generally NP-hard and heuristic methods are considered more practical in large scale problems.

This study mostly concentrated on the assignment of spot instances without considering their pricing fluctuation. Pricing of spot instances is a fundamental problem faced by providers. They attempt to dynamically adjust the price of each instance type with changes in to supply and demand in order to maximize revenue and increase efficiency. Whenever demand is lower and the system is under-utilized, the provider decreases the price to attract more customers. On the other hand, in the times of datacentre over-utilization, it's beneficial to raise the price. Devising a pricing algorithm and combining it with the scheduling process is the focus of our future work.

In addition, datacentre providers usually offer the spare computing resources as spot instances. Scheduling of these resources requires monitoring of entire system in different levels and despite all consideration, unexpected events and anomalies in system behaviour happen sometimes. Early identification of these anomalies is an important factor for providing better services. We would investigate this topic in the following chapters. The proposed method in this chapter, provides the best configuration of resources based on the resource demand of the batch jobs. The algorithm tries to improve the job wait times in the queue before processing, while minimizing the cost of resources for the users. It means that the offered resource type are the best option for job requirement and it reduces the underutilization in them. So, users with a group of jobs submitted to the system would significantly

save on their computing expenses. This also can be interpreted as saving in energy in datacentres, since fewer number of underutilized resources would cut down the datacentre power usage. There are various other ways to save energy consumption in datacentres like consolidation and migration which are discussed in more details in the next chapter.

Chapter 5

Energy Management in datacentres

5.1 Introduction

In this chapter we study the applicability of resource prediction and its impact on optimal resource allocation where the Service Level Agreement (SLA) will be met while the power consumption is kept to the minimum. As we saw in previous chapters, the main characteristics of resource usage behavior in Alibaba datacentre is that it is fluctuating and being periodic. On a daily cycle, there are peak hours and there are off-hours. These depend on what services the Data Center provides, which hour of the day it is and which parts of the world uses the services in the Data Center. All these parameters affect the nature and behavior of resource utilization.

In Fig. 5.1, we show the average cpu utilization among all servers in Alibaba datacentre over a week period. This is an example, however we believe this is a typical behavior in most datacentres serving in one time zone. It can be seen from the picture that a daily cycle can reach up to 80% of maximum utilization and down to below 20%. This picture also reveals that the utilization is periodic on a daily basis and this implies that it's also predictable.

In recent years, Machine learning (ML) has been advancing to the point that nearly every device leverages some kind of learning algorithms. This was driven mainly by the increase in compute and the availability of datasets. Machine learning algorithms can be trained on big datasets with large specialized computing hardware to produce best models. Those then can be used in products and services to run the models with less effort and fewer resources. Combining ML with datacentre datasets, we can come up with good resource allocation models.

It has been suggested that 2% of the globlal electricity is consumed by dataenters [57]. Although many techniques were adopted for more efficient use, it is estimated that 50% of the used power is actually wasted [58]. As shown above, resource utilization goes through a daily cycle where there are peak and off hours. This pattern for CPU utilization has been also identified in other datacentres like AM2 datacentre [59], although for bigger datacentres, the behavior can be more complex since they cover a wider geographical area. Regarding to the cyclic behavior of datacentres, we can come up with a mechanism that conserves energy used by preemptively turning-on and off equipment. Network functions virtualization (NFV) has been introduced to allow dynamic behavior in datacentre and network deployments. The programmability of such frameworks allows the concentration of resources in smaller number of physical server/containers. This enables better efficiency and power management. For example, at off-hours the services can be migrated to a small number of server racks which will allow non-operational rack to be shutdown through a management software. This will also cut the energy needed to operate the servers and to cool off that part of datacentre.

In the this work, we will combine those ideas to study the applicability of prediction models on datacentre resources. In a similar work, Alutaibi et al. [60] showed the predictability of network traffic (GÉANT Network). Here we are interested in allocating enough resources to meet the Service Level Agreement (SLA), while saving energy in datacentre.

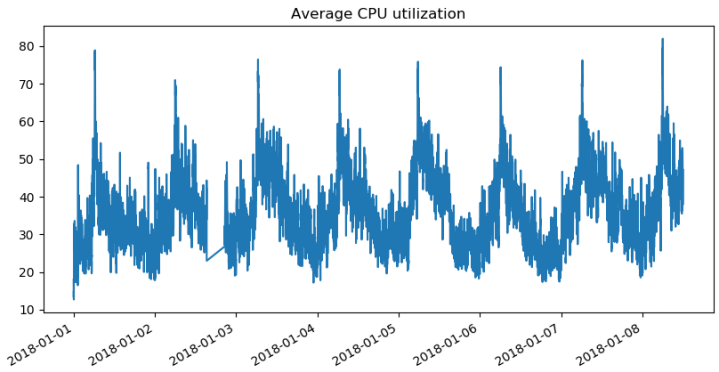


Figure 5.1: Average CPU utilization of Servers in Alibaba Datacentre

5.2 Resource Prediction

Machine learning has been advancing rapidly in the last decade due to the increase in computational power and the proliferation of data. Researchers and Data Scientists have created an ever increasing number of models. Each model can claim its superiority in fitting a certain data or description of a problem. Some are very well suited for a class of problems. One thing that is certain in this field is that better models are always replacing older ones. This is in part because of the availability of new descriptive data, bigger feature space, better tuning of model hyper-parameter, etc. That's why in what follows we evaluate our approach on several prediction models.

The normal testing of the goodness-of-fit of a model is presented by the difference between the predicted value and the tested data. Although this is the most common measure of model performance, recently the focus has been shifted to produce models that can give upper and lower bounds on the predicted data. It is much more meaningful to say "the annual wage of a professional python programmer is between 100K-127K dollars" than to say "the annual wage for a professional python programmer is 110K". The first statement is more meaningful because it gives a better insight into the data and shows the possibility of getting a wage in the upper percentile. Also, it shows estimation, uncertainty, and distribution of the data. It is even more meaningful to quantify those limits by saying "the 90th percentile of the wage is 127k".

Any model can describe a real process in the following way:

$$h_i = f(x_i; \Theta) + e_i \tag{5.1}$$

where the function f is the estimate of the real values $\hat{f}$. For regression, the observed values x_i combined with the model parameters Θ produce the estimation of the model h_i. e_i represents the error induced in the model. It has been the norm to treat the error as an independently and identically distributed (iid) value. This implies that it has variance σ^2 and a normal distribution of $\mathcal{N}(0, \sigma^2)$ [61][62]. A minimization of the cost function as shown in Eq. 5.2 will give the least square estimate of the model parameters[63].

$$C(\Theta) = \sum_{i=1}^{n} (h_i - y_i)^2 \tag{5.2}$$

where h_i is the process being modeled, y_i is the estimated model output, and n is

the set of data collected to describe the process. In linear regression we have:

$$y_i = \Theta_2 x_i + \Theta_1 + \epsilon \tag{5.3}$$

With the same error assumptions stated above, the minimization of the errors through the minimization of the likelihood estimate of the coefficients (or the variance of the maximum likelihood of the estimation) is as follows:

$$\hat{\sigma} = \underset{\Theta_1,\Theta_2}{\operatorname{argmin}} \frac{N-1}{N} \sum_{i=1}^{n} (y_i - \Theta_1 - \Theta_2 x_i)^2 \tag{5.4}$$

As a result of our assumption that the error is normally distributed, we find the Prediction Interval (PI) of the linear model from its proprieties as:

$$\begin{aligned} Upperlimit &= y + \sigma.\, \mathrm{z}_{\alpha/2} \\ Lowerlimit &= y - \sigma.\, \mathrm{z}_{\alpha/2} \end{aligned} \tag{5.5}$$

From this, we get for example the 95% prediction interval as:

$$[\hat{\mu} - 2\hat{\sigma}, \hat{\mu} + 2\hat{\sigma}]$$

This solution is limited by its definition. The model should be linear. The error is assumed to be iid and normally distributed. This formulation doesn't describe the majority of problems. If we look at the problem as an optimization of residuals, we find a formulation that can extend beyond linear models. From statistics, we know that the mean is the minimization of squared sum of residuals, while the median is the minimization of the absolute sum of the residuals. From now on, we can assume that minimization of the sum of asymmetrically weighted absolute residuals is a quantile [64]. This is accomplished by giving different weights to negative and positive values. The most straightforward implementation of this loss function is shown in Eq. 5.6

$$C(\beta_i|\alpha) = \begin{cases} \alpha\beta_i & \text{if } \beta_i \geq 0, \\ (\alpha - 1)\beta_i & \text{if } \beta_i < 0, \end{cases} \tag{5.6}$$

where α is the quantile we are seeking and β_i can be described as the remainder after subtracting predicted h_i value from the observed value y_i as in Eq. 5.7.

$$\beta_i = h_i - y_i \tag{5.7}$$

Fig. 5.2 shows the 90% quantile and how the deviation from the correct value (error) causes different behavior in the negative compared to that in the positive. In high quantiles (like 90%), more negative errors are penalized more and have bigger impact. In our implementation we will use quantile loss to establish upper bounds on the predicted values.

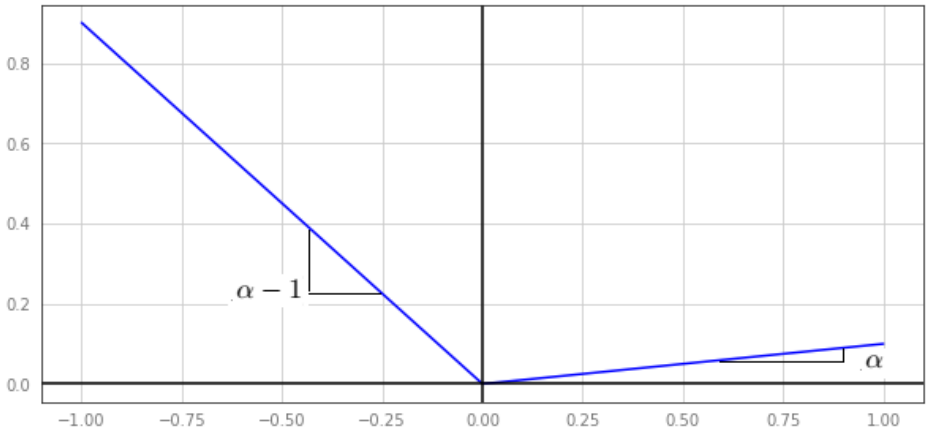


Figure 5.2: 90th quantile error impact on quantile loss.

5.3 Quantiles and Service Level Agreement

From the previous section, we can see that calculating a quantile out of the data, is an attempt to put a line through the data. This line is separating the data into two parts which are α and $1 - \alpha$. For example, a 90% quantile bound have 10% of the observed data above the quantile while 90% are under this quantile. Since dataenters serve multiple clients with different service priorities and needs, Service Level Agreements (SLA) are established between the client and the service provider. Most SLAs have strict availability requirements. This with other factors cause service providers to over-provision their resources to the point where only 10% of the resources are utilized. Quantiles are by definition a percentage guarantee that a percent of the the observed data lay below the quantile. With this, we can be assured that it will satisfy the needed SLA (in a perfect model). However, the model accuracy can vary widely and hidden model features can affect the outcomes. Taking those factors into account (in addition to others that we will mention in next sections) is very important when implementing resource allocation prediction.

5.4 Dynamic Allocation of Resources

A typycal server can waste up to 70% of the power consumed by a fully utilized server [65]. Using programmable NFVs to dynamically turn-on or turn-off servers depending on the predicted load, we can introduce better power management and efficiency. Consolidating servers to smaller number of containers/racks has been used widely to conserve power in datacentres [66]. Using prediction models we propose migrating virtual servers and services to the needed number of resources. Then we can dynamically turn-off idle devices. The conservation of power process does not need to completely shut down those devices. Some devices have lower power consuming modes that can be used. One aspect we need to guarantee is that the SLAs are not violated during this operation. Introducing a buffer with the prediction will mitigate the bursts of usage demands.

5.5 Prediction Models

To cover a wider range of models, we looked into the applicability of the most popular linear and non-linear prediction models. We did not consider ARIMA[67] and its variations because those models are built for univariate data. We will start by describing linear models then tackle tree-based and neural networks models.

5.5.1 Ordinary Least Squares (OLS)

Linear Quantile Regression is the most basic form of regression as described in Eq. 5.3. The the error is assumed normal and has the iid property. It is essentially a regression of the median 50% of the loss function in Eq. 5.6 [68]. OLS is another form of linear regression model. It minimizes the mean instead of the median. To change the prediction from the mean to the median (and then quantile), we can add the result of the prediction (mean) to the value of the inverse transform CDF of the median (or quantile). The reliance on the assumption that the errors are normally distributed highly affect OLS. Outliers can have a greater impact on OLS estimate [69]. Linear models produce good results when the stated conditions hold. They are flexible to changes and their time complexity is cheap ($O(p^2n + p^3)$, where p is the number of features [70]). Also, they benefit greatly from adding lag (supervised data) to the training data.

5.5.2 Gradient Tree Boosting (GTB)

Gradient Tree Boosting is a branch of decision trees that uses ensemble learning to do classification and regression. Decision trees tend to overfit the training data and for that Random Forests (RF) [71] and GTB was proposed to train on different parts of the data (using bagging). The idea is to create several subsets of data from training sample chosen randomly with replacement. Then, each collection of subset data is used to train their decision trees. So, we will have an ensemble of different models. Average of all the predictions from different trees ($T_b(x)$) are used which is more robust than a single decision tree.

$$F(x) = \frac{1}{B}\sum_{b=1}^{B} T_b(x) \tag{5.8}$$

To find the prediction interval (PI) using RF, we are faced with the same problem in OLS. In [72] the authors suggested building a distribution out of all leaf observations (not only mean). Similar to RF, GTB combines multiple estimators to produce predictions. But rather than using averaging to combine the results of individual trees, GTB uses boosting to combine weaker models to produce stronger ensemble. The model is built in a greedy additive way.

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x) \tag{5.9}$$

where $F_m(x)$ is the approximation function, h_m is estimation minimization function of the loss (L) and γ_m is a decision tree at step m fit to the pseudoresiduals of the loss function which is given in below [73].

$$\gamma_m = argmin_h \sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x_i)) \tag{5.10}$$

Tree based models vary greatly and one can be strong in one aspect while being weak in another. We choose GTB because it deals with weak learners (smaller trees) to reduce bias. It has been suggested that GTB produces better results because of its versatility, simplicity, faster form, and ability to incorporate automatic predictor selection [74]. The computational complexity for GTB is $O(npn_{trees})$, where n_{trees} is the number of trees [70].

5.5.3 Long Short-Term Memory (LSTM)

A field of Machine Learning that has become very popular in recent years is Neural Networks (NN). It has become a very prominent method and research intensive. The simplicity, fast convergence, flexibility, and adaptability to non-linear processes are some of the advantages of NN. There are many types of NN architectures each claiming to solve one type of problem better than the others. The simplest form of NN is the Feedforward Neural Network (FNN) shown in Fig. 5.3, where the information moves from the input to the output while passing by an activation function (usually sigmoid). There are many types of FNN (Autoencoder, Convolutional, etc).

In FNN, the NN consists of input nodes where the data to be trained or predicted is fed. It is followed by one or more hidden perceptron (nodes) that each gets activated or not depending on the sum of the products of the weights connected to it. This is done in sequence until the information propagates to the output. In training the difference between the predicted value and the actual is used to adjust the weights backward (Backpropagation). The sigmoid function as shown in Eq. 5.11, is used for activation instead of a step function because it has a continuous derivative which is very helpful in the backpropagation phase. However, the step function is non-differentiable at $x = 0$ and it has zero derivative elsewhere. This means that gradient descent won't be able to make a progress in updating the weights. Also, the objective of the neural network is to learn values of the weights and biases so that the model could produce a prediction as close as possible to the real value. In order to do this, as in many optimization problems, we'd like a small change in the weight or bias to cause only a small corresponding change in the output from the network. By doing this, we can continuously tweaked the values of weights and bias towards resulting the best approximation. Having a function that can only generate either 0 or 1 won't help us to achieve this objective.

$$f(x) = \frac{1}{1 + e^{-x}} \tag{5.11}$$

FNN has benefited from the advances in the field in recent years. Especially from advances in drop out, regularization, optimization, loss functions, and fast processing [76]. Also the sheer amount of available data made it easier for researchers to experiment with different implementations and techniques.

We can change the NN loss function and use equation 5.6 directly as our loss function. This limits our experimentation with other loss functions but it should be

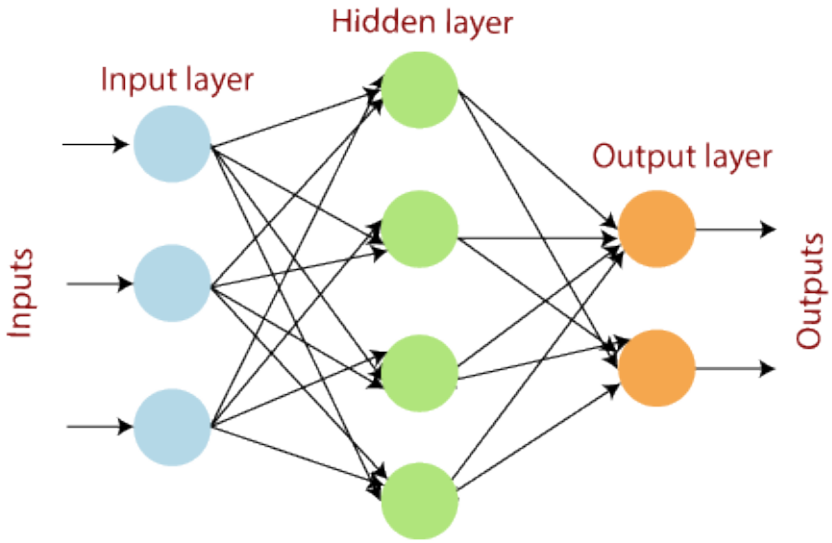


Figure 5.3: Simple Feedforward Neural Network.

sufficient to our purpose.

The promising results of FNN and its variants, opened the door for the research and development to tackle other architectures. One of those was Recurrent Neural Network (RNN). In this architecture, the internal nodes have memory of the previous inputs. This state is kept hidden but influences the output of the node. RNN have been proven to be better predictors of sequences, handwriting[77], and speech recognition[78]. A variant of RNNs is LSTM where internal structures are introduced to solve the vanishing and exploding gradient problems[79]. The internal structures are called gates and they are:

- Input gate - This gate accepts the new inputs and through an activation Sigmoid allows or prevents the input to influence the internal state of the LSTM. Then it uses a weighing function (usually tanh) to push the input internally.

$$\begin{aligned} i_t &= Sigmoid(W_i \cdot [S_{t-1}, x_t] + \alpha_i) \\ \hat{C}_t &= tanh(W_c \cdot [S_{t-1}, x_t] + \alpha_c) \end{aligned} \tag{5.12}$$

  Where S_{t-1} is the state of the previous node.

- Forget gate - This gate acts as a suppressor to weaker states from previous and current time slot. Both inputs are run through another sigmoidal function (10).

$$F_i = Sigmoid(W_f \cdot [S_{t-1}, x_t] + \alpha_f) \tag{5.13}$$

- Output gate - Similar to the input gate, the output gate consists of a sigmoid function followed by a weighing function. This influences the next step $t + 1$ where the first function decides on which information to be passed and how important it is (11).

$$\begin{aligned} O_t &= Sigmoid(W_o \cdot [S_{t-1}, x_t] + \alpha_o) \\ S_t &= O_t * tanh(C_t) \end{aligned} \tag{5.14}$$

LSTM's ability to keep information in its memory allowed it to be a good predictor for sequenced data, time series , and context sensitive training[80].

5.6 Data and Experiments

Data sets containing datacentre usage records are rare, especially those that show resource usage per unit of time. One of the popular ones is Alibaba dataset. The dataset includes an eight day collection of various matrices on different resources usage. These include CPU, Memory, Disk utilization of machines and also containers. We chose these three resources as our prediction target. The data was sanitized out of outliers. Since we like to capture the daily cycle, we created three new features, which are Minute of the hour, Hour of the day, and Day of the week. These have been extracted out of the none periodic absolute time given. Each resource was predicted separately. We sampled usage data every 15 minutes and used that for prediction. So, the x axis in the following figures shows 15 minute time steps.

The applicability of a model has been typically measured with the Mean Square Error (MSE) or Mean Absolute Error (MAE). In our study we value the satisfaction of the SLA more. Also, the maximization of resource usage is an important metric. But the most important metric is how much energy was saved using the prediction mechanism. This will introduce a delicate balance between trying to over-allocate resources to guarantee the resource requirements, and to under-allocate resources to save on energy.

To find model parameters, we employed a technique used in data science called grid search[81]. For the model learning rate we chose 0.1. For the tree model we chose

2000, 9, 3 as the number of estimators, the minimum sample leaf, and maximum depth, respectively.

Similar to the work in [82], we found that constructing a neural net with more than one hidden layer will give a decaying improvement to the results of less than 1%. We chose the number of input nodes as 100.

We are going to first show the prediction results of the CPU data. This will give us insight on the efficiency and the guarantee of SLAs. Then we will show the prediction results of the other resources. Finally, we will discuss improvements and suggestions for the employed techniques.

5.6.1 Predicting upper Quantiles of the CPU Utilization

From Fig. 5.1, we can see that the CPU data is very periodic which implies that this cycle is predictable. In our experiments we tested a good number of models but only the results of the best performing ones are shown. We don't claim that these are the optimal prediction results. There exists better models and better parameters but using them will show how prediction of resources is possible and useful.

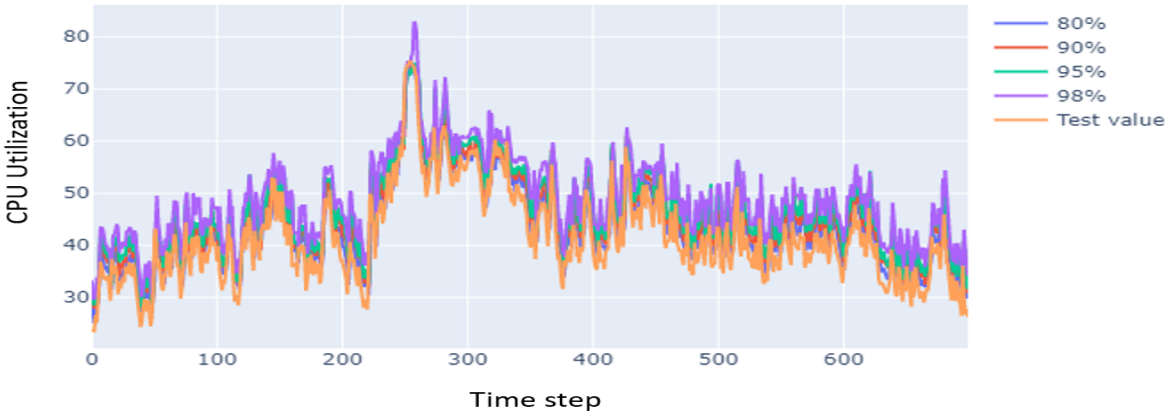
Since we are trying to fulfill the SLA we chose to predict four quantiles that are usually used. These are the 80%, 90%, 95%, and 98%. We trained the data on the first seven days then tested it on the eighth day. Although the data doesn't represent a wider range, we believe that the results will be similar. Figure 5.4 shows the quantiles predicted compared to the test value using different models. The figure shows how each predicted quantile builds an upper bound that tries to capture a quantile amount or more below it.

The quantile loss for each model is represented in Fig. 5.5 and Fig. 5.6. These show the total quantile loss and the loss per quantile, respectivly. Since the models are predicting the utilization, the loss is a percentage loss out of 100. We see that the linear model has the least loss compared to the other models. Although there are differences in prediction between the implemented models, the loss in general is very small to a point were we can say that all are reasonable predictors and are very similar.

5.6.2 Energy prediction and SLA

Since resources in datacentres are turned on/off as discrete units, it is important for us to treat the prediction discrete as well. For simplicity, we assumed each one percent of

(a) Ordinary Least Square



(b) Gradient Tree Boosting

(c) Long Short-term Memory

Figure 5.4: CPU Quantile Prediction

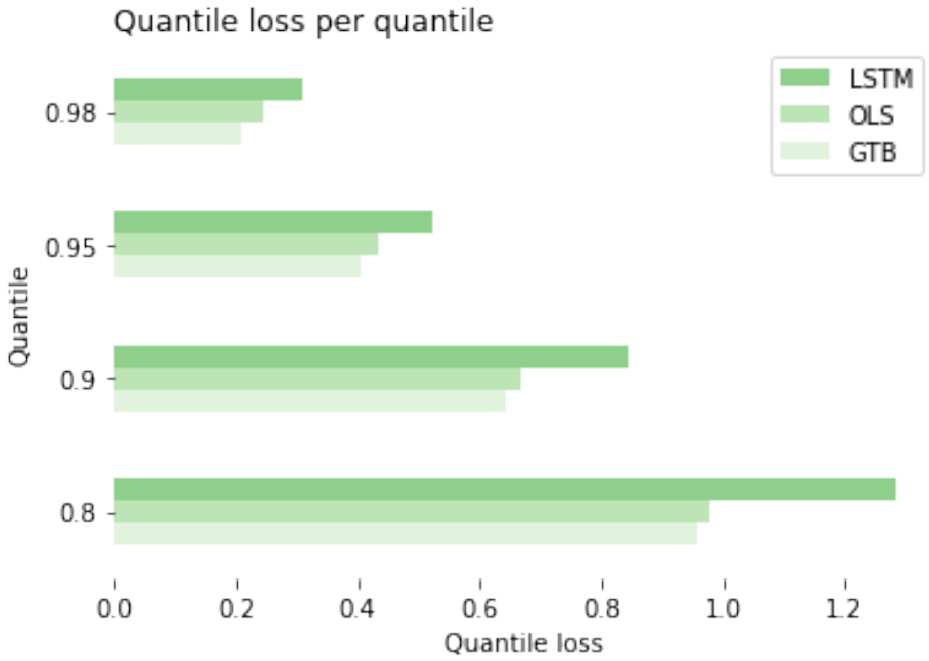


Figure 5.5: Detailed Quantile loss.

the utilization (out of 100) to be one resource. This means that predictions should be rounded-up to represent resource prediction. This assumption is very representative of a medium size datacentre. Figure 5.7 shows the 90% quantile prediction of OLS rounded (in bars) to the test value. Since it is the prediction of the 90% quantile, we can expect that 90% of the test data is below this boundary and a 10% above. As a matter of fact, the number of test data that went above this boundary for this particular example were 7.5% (due to rounding up). This in essence satisfies the SLA demanded of the prediction. Table 5.1 shows SLA compliance for all prediction models with the quantiles for the CPU dataset.

Table 5.1: SLA compliance

	Quantiles			
	80%	90%	95%	98%
OLS	87.43	92.48	95.49	97.47
GTB	78.71	89.44	93.94	97.56
LSTM	84.22	89.35	94.49	97.30

Although there is a small deviation from the target value, we can see that the prediction is very close. We attribute this error to the small dataset and the inherent

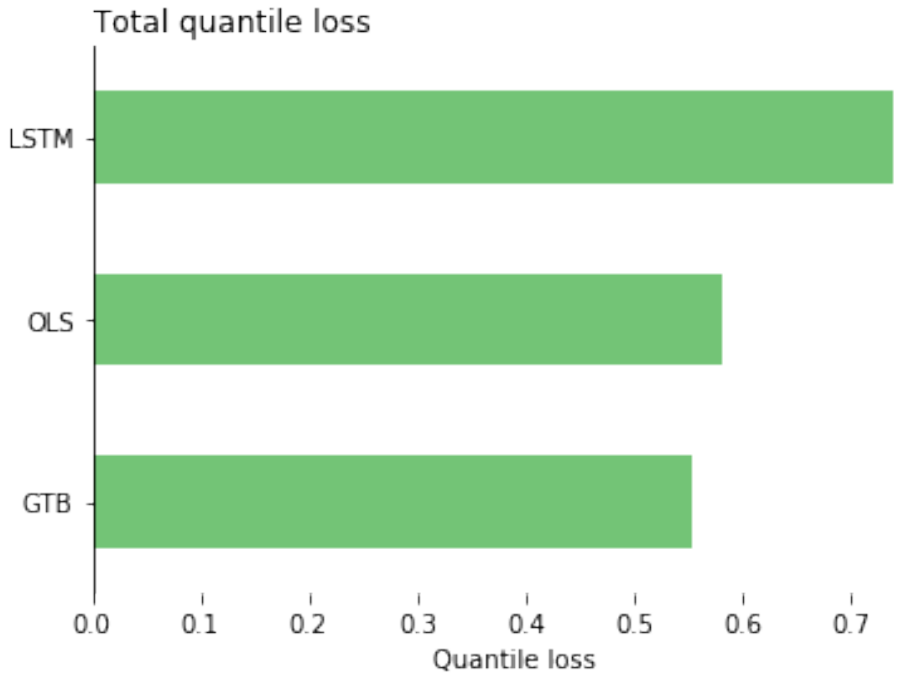


Figure 5.6: Average Quantile loss.

nature of prediction in general.

5.6.3 Buffer and power saved

It has been shown [83] that CPU utilization has a direct relationship with power consumption. From that we can assume that any conserved resources (CPU) can be translated to a direct conservation of energy. Saying that our results and conclusions are a prove that dynamic allocation of resources is undermined.

From Figure 5.7 we can see how much we can save on energy if we adopt the prediction scheme. The exact average predicted cut is 56% (average unutilized resources). The average extra predicted resources above the test data is 7.5% which means the resources are on average utilizing 92.5% of the predicted CPU resources. Since we are looking for compliance rather than accuracy, we can introduce a buffer. The buffer can act like an extra guarantee and to absorb bursts in demand [84] (an inherent Poisson characteristic of datacentre traffic and usage). If we assume a 20% buffer to CPU allocation we get a 36% more efficient energy use and a compliance of 100% (we got no test values that went above the prediction plus the buffer in the CPU simulation).

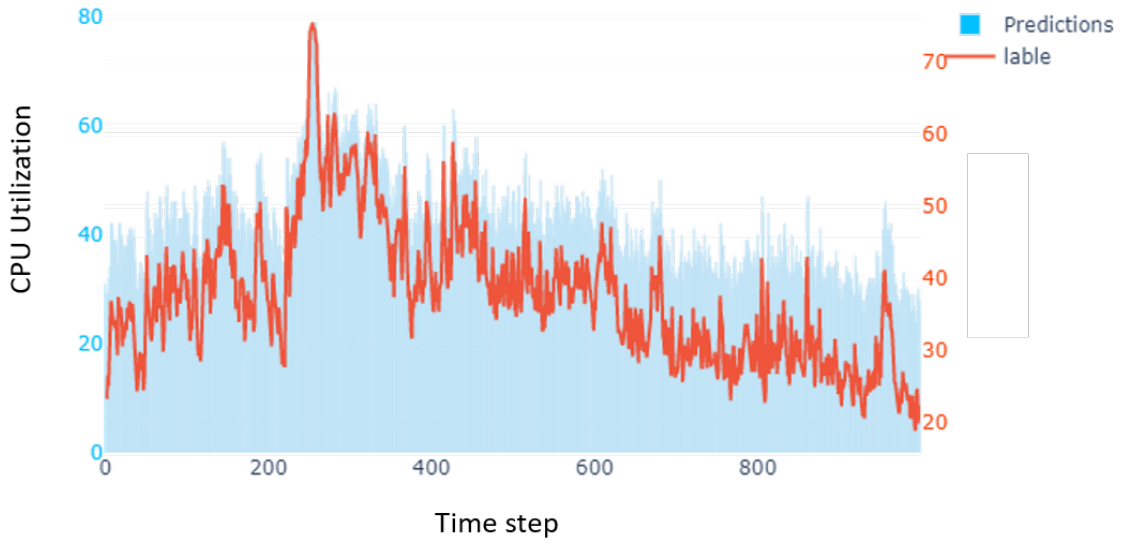


Figure 5.7: Prediction Rounded compared to tested.

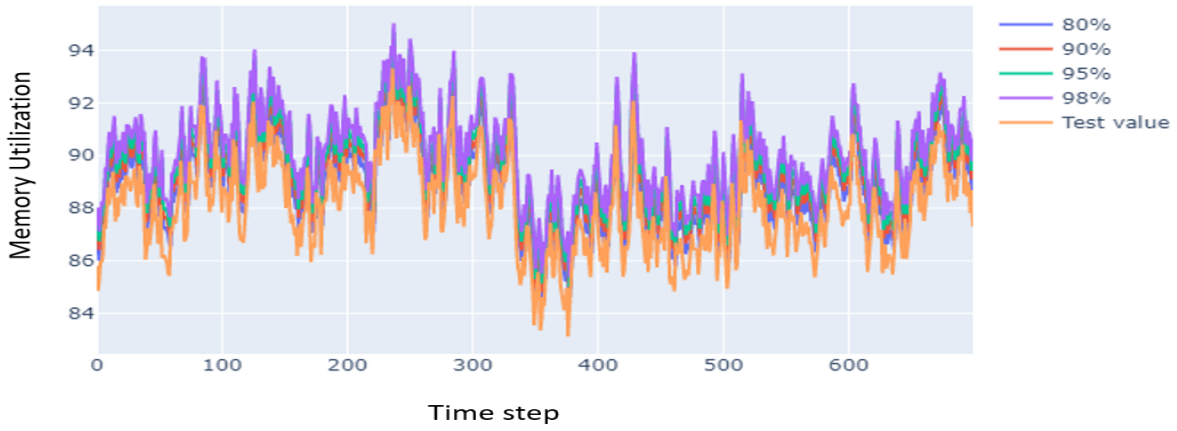
5.6.4 Other resources

In Table 5.2 we summarize the results of the memory and disk i/o model quantile prediction. We noticed that the memory data has less predictability (periodic behavior) and thus the efficiency was the least among the three resources studied. We attribute this to the fact that operating systems can address more memory than that allocated physically. This and other techniques employed like paging can give a poor representation of the memory used. Also, processes can stay idle in memory without being removed or used for arbitrary time periods.

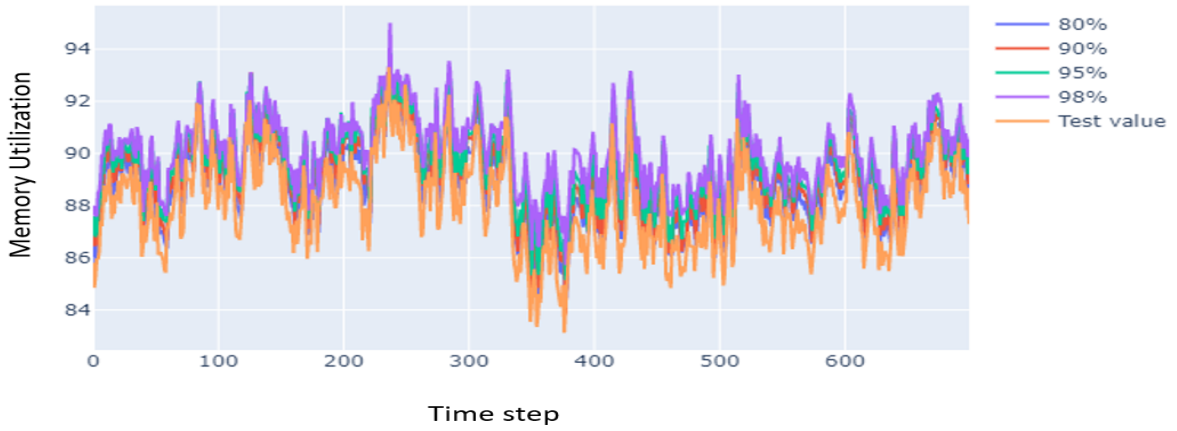
Table 5.2: Memory and Disk SLA compliance

	Quantiles							
	Memory				Disk i/o			
	80%	90%	95%	98%	80%	90%	95%	98%
OLS	93	97.1	98.6	99.4	91.2	94.7	96.2	97.4
GTB	91.3	95.2	97.3	98.7	86.4	93.2	96.7	98.4
LSTM	94.7	97	98	98.3	57.1	77.3	86.1	98.3

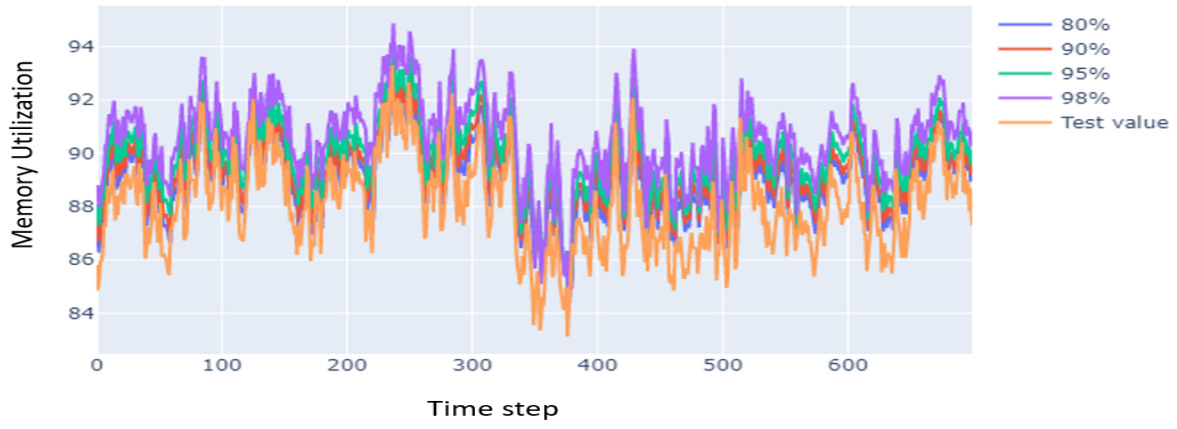
Fig. 5.8 and 5.9 show the memory and disk prediction using the three models used. We can see tight bounds on all quantiles. Table 5.2 shows the average quantile loss of all models ran through the three resources studied. The table gives a good



(a) Ordinary Least Square

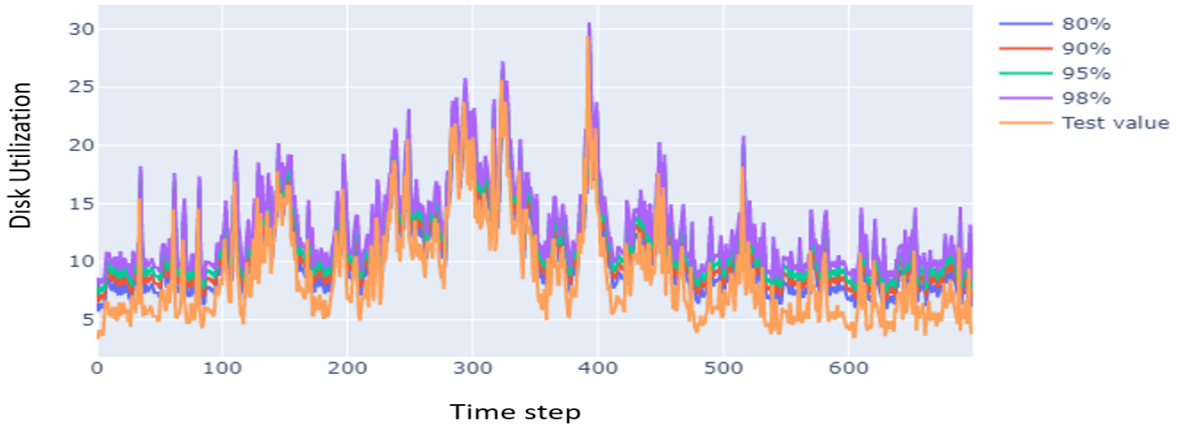


(b) Gradient Tree Boosting

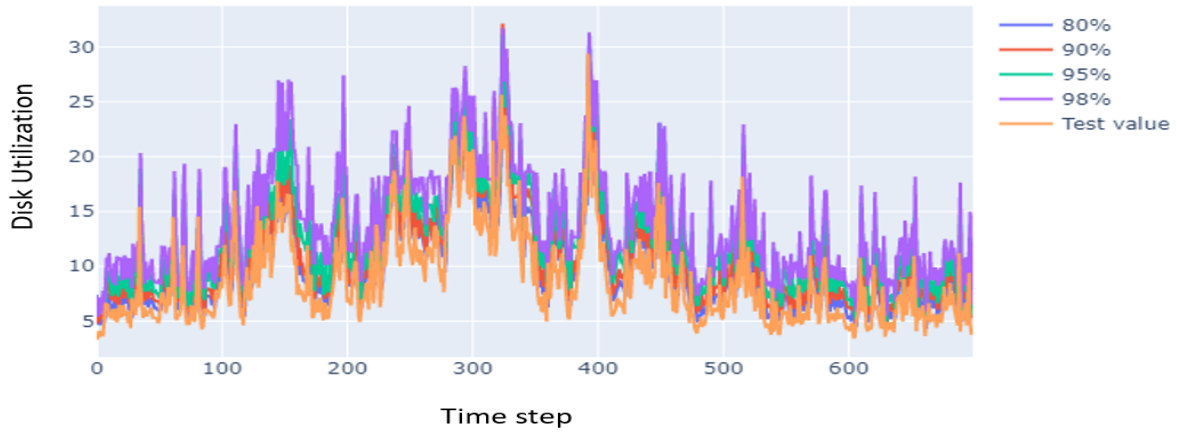


(c) Long Short-term Memory

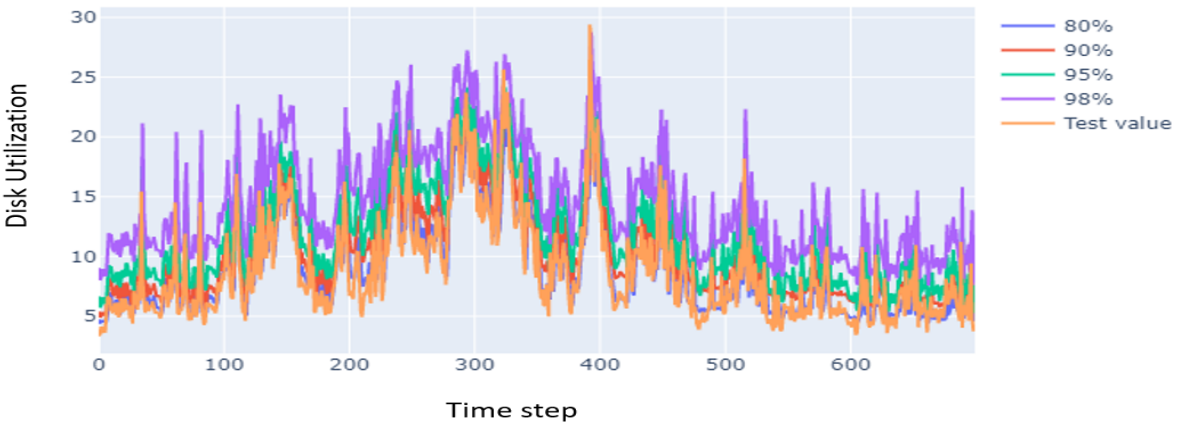
Figure 5.8: Memory Quantile Prediction



(a) Ordinary Least Square



(b) Gradient Tree Boosting



(c) Long Short-term Memory

Figure 5.9: Disk Quantile Prediction

overlook on the performance of the models.

5.7 Results and Discussion

As we can see, the differences between the selected prediction models are very small. The results show that the predictability of the resources usage is high, as if choosing any prediction model didn't matter much (we have tried others and they gave nearly similar results). Consequently, using such techniques has an impact on reducing energy and cost. The next challenge to such an implementation is to concentrate used resources in fewer physical devices. This will enable resources to be concentrated for more efficient cooling and maintenance operations. Also, since on average 56% of the resources are off, we can assume that their lifetimes are doubled. Of course, in a real implementation other factors should be taken into consideration like, power-on lag times, buffer for sudden spikes in demand, resources allocation optimization, and fault tolerance of management of resources and cooling. In the next chapter we will discuss the use of anomaly detection for improving quality of service, reducing power consumption and increasing revenues.

Chapter 6

Anomaly Detection for Performance Monitoring in Cloud

6.1 Introduction

Anomaly detection is an effective means of identifying unusual or unexpected events and measurements. The term "unexpected" here can be interpreted as statistically improbable in the system. Therefore, before detecting any anomalies, we need a comprehensive knowledge of system's baseline performance and behavior. Like many other areas, anomaly detection also applies to the field of cloud computing in different ways, from the detection of various types of intrusions to the detection of hardware failures and performance anomalies. Monitoring the entire system at different levels, such as performance metrics and resource workloads is a way to early identification of anomalies. This is critical for getting high-quality services, saving in power consumption and increasing the revenues.

In this work, we present an anomaly detection system based on Cloud server usage behaviour analysis at different levels, to detect any misuse of cloud instances. Prediction-based and Neural Net-based methodologies are used to analyse the normal usage of Alibaba cloud physical machines and then detect abnormalities.

6.2 Behavior Analysis of Alibaba Servers

The cpu usage of physical machines fluctuates over time and has a regular daily pattern, very high during the day than other times. This traffic pattern is consistent in

almost all servers. The cluster nodes that belong to the same geographical zone have synchronous ups and downs. Moreover, comparing the cpu usage of a sample machine and a sample container running on the same machine in Fig. 6.1 and Fig. 6.2 demonstrates that the periodicity of cpu usage in machines comes from online jobs running on them. Both graphs have peaks at the same time and fluctuate synchronously. Since in Alibaba cloud online traffic mostly belongs to their e-commerce platform which is obviously higher during daytime than night time. This makes the total cpu usage of machines high during daytime. Another fact about e-commerce platforms is that their traffic goes extremely high during a seasonal promotion, even hundreds of times of a normal day. And, the system must be capable of handling this amount of workload. So, we need a schema that ensures the offline service occupies few resources most of the time, and more resources are consumed by the online service. Looking at the behavior of server usage in Alibaba datacentre, we think that providing extra resources as Spot instances, can be a solution for responding to fluctuating traffic in datacentre. Spot instances can be released whenever needed and be ready for high traffic situation. Having spot instances, there is no need for separate schema for high and low traffic situation. The spot instance scheduler would take care of all of that.

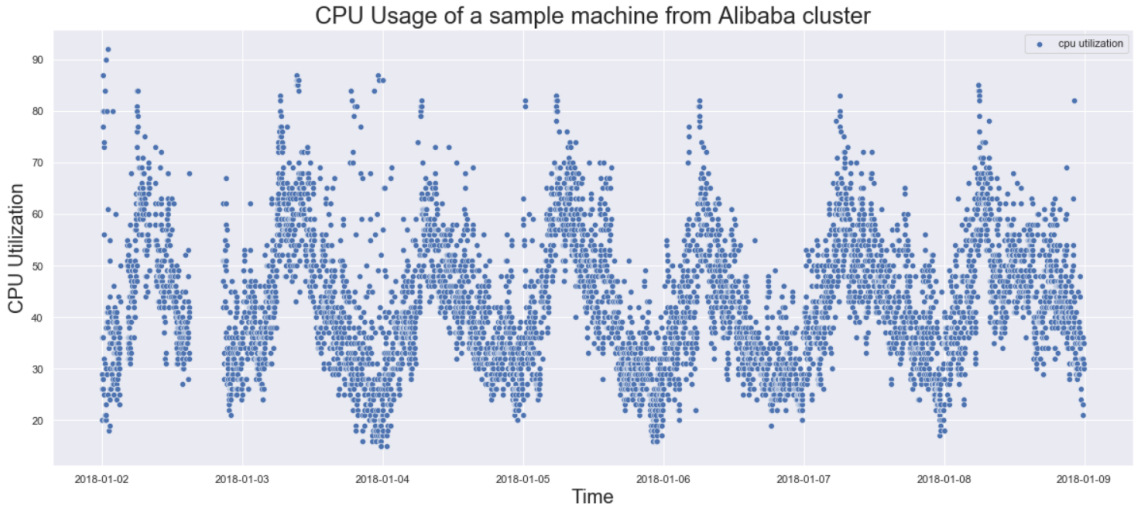


Figure 6.1: CPU Usage of a Sample Machine from Alibaba Cluster

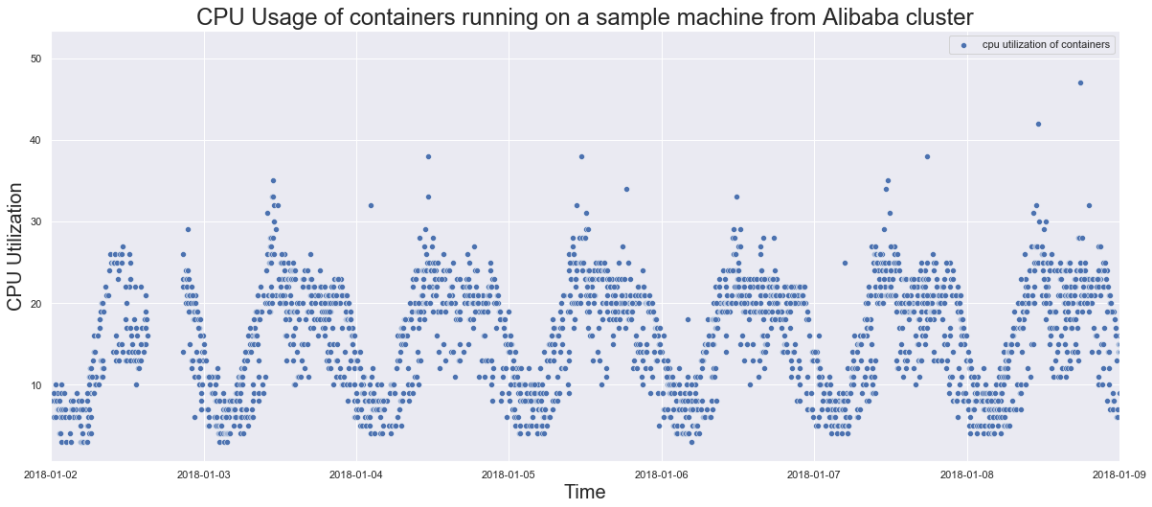


Figure 6.2: CPU Usage of Containers running on a sample machine from Alibaba Cluster

6.3 Anomaly Detection Methods

There is not one specific model that performs well in all anomaly detection problems. Here, we describe the fundamental methodologies which are mainly utilized for detecting anomalies on timeseries in an unsupervised way.

- **Probability Based Approaches:** One of the most straightforward probability methods is Z-score. Z-score stands for the number of standard deviation that sample is far away from mean of distribution. The idea is that the we assume each feature fits a normal distribution and samples with higher Z-score are most likely to be an anomaly. One of the disadvantages of this approach is that it assumes the feature fits a normal distribution which is not true in all cases.

- **Dimension Reduction Based Approaches:** Dealing with high dimensional data could be challenging and there are several approaches to reduce the number of variable which is called dimension reduction. Principal Component Analysis (PCA) is one of the main techniques used for dimension reduction. It helps to cover most of the variance in data with a smaller dimension by extracting eigenvectors that have largest eigenvalues. Therefore, it is able to keep most of the information in the data with a very smaller dimension. While using PCA in anomaly detection, it decomposes data into a smaller dimension and then it reconstructs data from the decomposed version of data again. Anomalous

samples will exhibit more loss or reconstruction error than normal ones.

- **Forecasting Based Approaches:** Using these methodologies, we predict the future values of the features and if forecasted values are out of confidence interval, those samples are considered as anomalies. And the accuracy of the forecasting model directly affects the success of anomaly detection. There are multiple forecasting models such as ARIMA and LSTM, however a popular model developed by Facebook and designed for predicting time series is fbprophet. This module specifically caters to stationarity and seasonality, and can be tuned with some hyper-parameters.

- **Neural Network Based Approaches:** An Autoencoder is a special type of neural network, mainly used for feature extraction and dimension reduction. However, it can be used for anomaly detection. Autoencoder has two seperate parts: encoding and decoding. First, main features are extracted and patterns in data are revealed using encoding. Then each sample is reconstructed by decoding. The reconstruction error will be minumum for normal samples. On the other hand, the model is not able to reconstruct abnormal samples, resulting a high reconstruction error. So, the reconstruction errors are used as the anomaly scores, the higher reconstruction error a sample has, the more likely it is to be an anomaly. Since layers of autoencoders can be composed of LSTMs at the same time, dependencies in sequential data like time series can be captured. That makes Autoencoder very convenient for time series.

6.4 Anomaly Detection in Alibaba Datacentre

Quick discovery of anomalies in a cluster is very important. It helps to locate bottlenecks, troubleshoot problems and improve utilization. Investigating the patterns in resource usage of physical machines in Alibaba datacentre during a week revealed different levels of abnormal behaviors. We have studied several different physical machines and the online and offline jobs running on them. And we found two levels of anomaly among them and the possible reasons that can cause them in the system which are described as below:

- Anomalies in resource (cpu and memory) utilization of physical machines. There are a few number of machines with very low utilization which is caused by

uneven co-located workload distribution. The goal of finding these anomalies are to redirect the workload to the under-utilized machines to optimize energy consumption. Comparing Fig. 6.3 and Fig. 6.4, we can observe the difference between cpu usage of two sample servers. While the usage in machine #1 is 50% and reaches up to 80%, the cpu usage for machine #2 is barely above 25% in the same time period. This pattern can be seen in memory usage of the same machines in Fig. 6.5 and Fig. 6.6. Very different patterns of memory usage in these two servers are obvious.

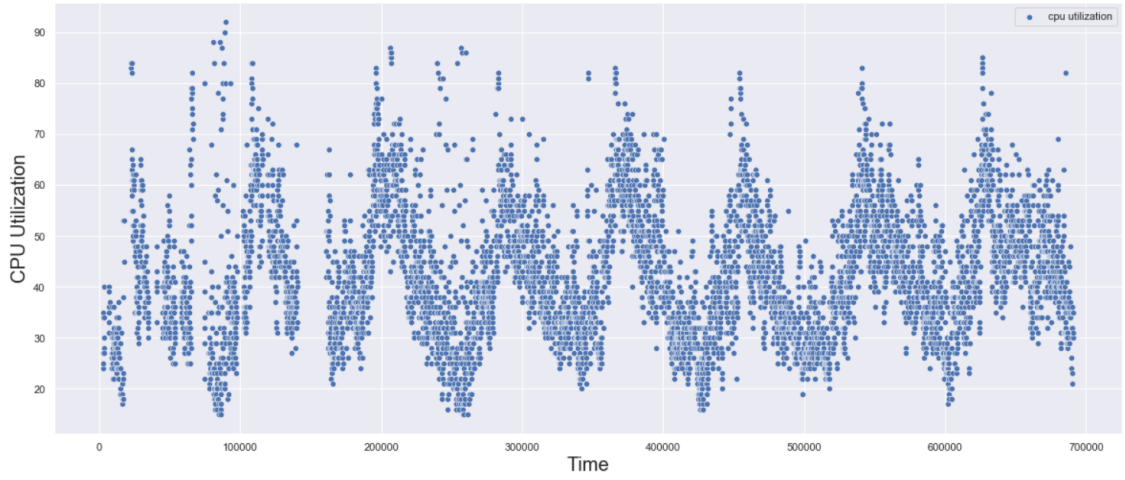


Figure 6.3: CPU Usage of Machine #1 from Alibaba Cluster

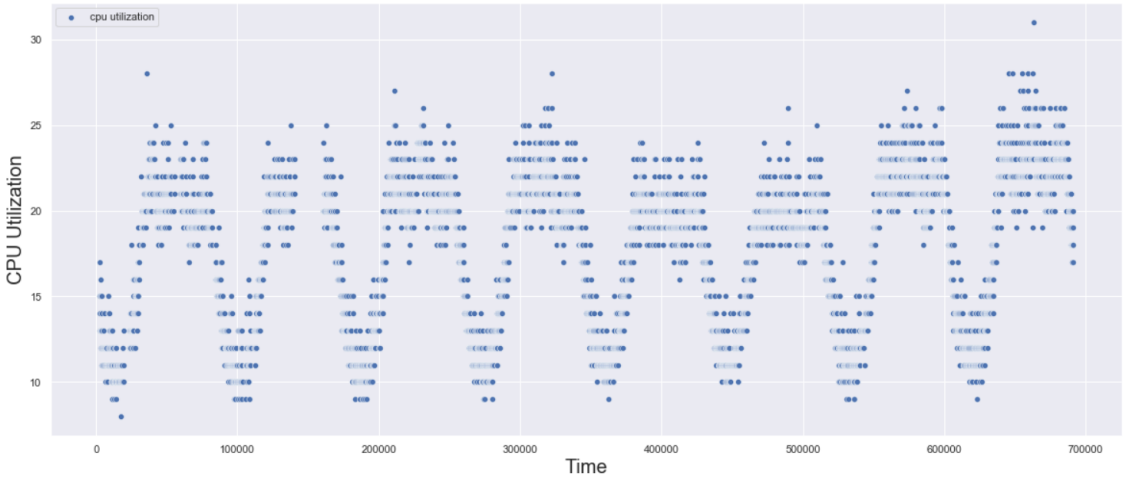


Figure 6.4: CPU Usage of Machine #2 from Alibaba Cluster

- Abnormal and unusual events in the daily pattern of resource usage that could

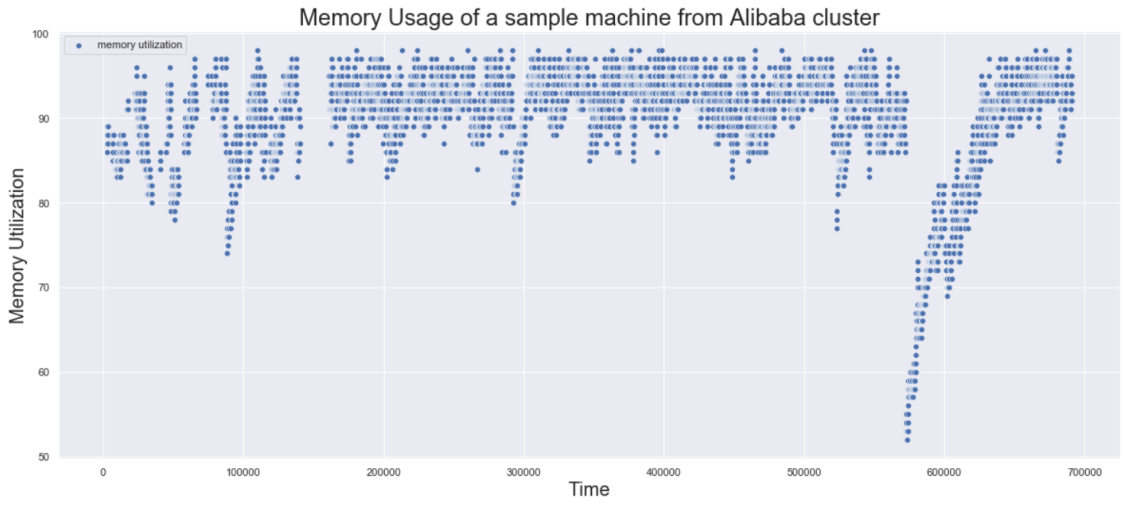


Figure 6.5: Memory Usage of Machine #1 from Alibaba Cluster

be because of system failure. Real-time detection of these group of anomalies are crucial for the system in order to be able to provide enough resources for the arriving jobs. This would lead to increasing performance and saving in the energy. To detect this type of anomaly in usage of resources, we applied two models on cpu usage of machines, which as mentioned before has a periodic behaviour on a daily basis. We compared the results of anomaly detection using prophet model [85] and Autoencoder model [86] in Fig. 6.7 and Fig. 6.8. As we can see, Prophet model considers any high value points in utilization during low-time of the day as an anomaly. These unpredicted usage in an unexpected time of the day is called Unexpected high usage and can degrade the performance of the system. While Autoencoder model could not detect these unusual behavior. It works like a threshold based method and identifies any utilization above a threshold value as an anomaly. So, in the Alibaba case, where the cpu usage of servers is periodic, Prophet model outperforms the Autoencoder method. We also applied the same models on cpu usage of server in Google cluster, where despite Alibaba it is not periodic. Prophet works better in that case too.

6.5 Solutions for Improving Performance

By collocation of offline batch jobs and online jobs, Alibaba improved resource utilization. Here, we are proposing a method to increase resource utilization even more

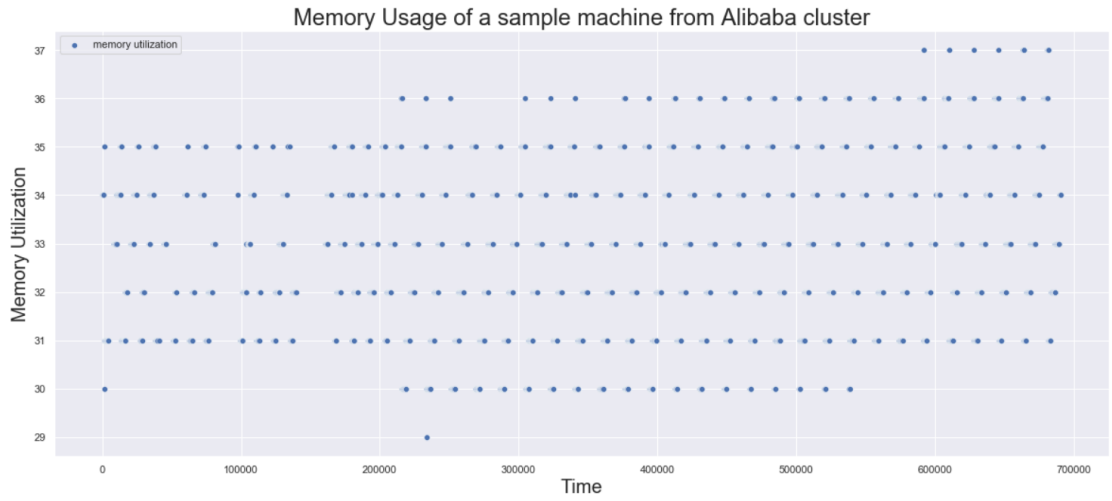


Figure 6.6: Memory Usage of Machine #1 from Alibaba Cluster

than that by offering the extra resources as spot instances. We investigated the CPU usage of servers across the Alibaba cluster and the results shows that there is a periodicity along this usage as a result of collocation. However, the Memory utilization is very high for the most machines. Fig. 6.11 is a box-and-whisker plots that shows the distribution of cpu and memory usage among all servers in Alibaba datacentre. As we can see, memory is fully occupied, with more that 90% on average but cpu is under-utilized. The average cpu usage is less than 40%. So, first suggestion is that the scheduling algorithm can be changed in a way that utilizes cpu more than memory since cpu is the bottleneck for most applications. Migration of the VM could be also efficient. This would increase the utilization of servers and leads to saving power consumption. A simple calculation shows that an increase in utilization from U_1 to U_2, without taking other factors into account, would lead to saving $N \times (U_2 - U_1)/U_2$, where N is the initial number of servers. For example, if we can increase the average utilization by 5%, from 35% to 40% on 200 servers, we can save 25 of them.

On the other hand, this unused capability of the servers can be offered as spot instances and to serve even more batch jobs which are CPU intensive. This would improve the system performance significantly as the nature of spot instances are compatible for serving batch computing. As the number of online jobs increase, spot instances would be released and get back to serve these online jobs. Since we know the pattern in usual server CPU utilization, pricing for the spare resources would be also easier. For instance, during that time of the day that online service traffic and cpu utilization of servers are low, we can schedule executing low priority, cpu intensive

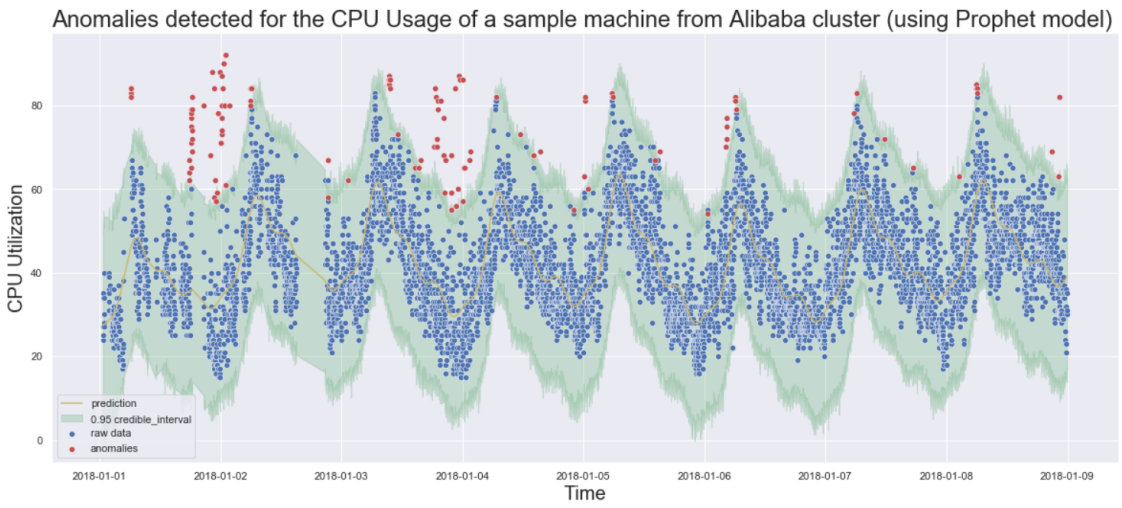


Figure 6.7: CPU Usage of a Sample Machine from Alibaba Cluster

batch jobs. Therefore, we should be aware of any anomalies during these times, which would degrade the accuracy of our estimation. A monitoring and predicting tool for tracking resource usage on physical machine is necessary to report any unusual event.

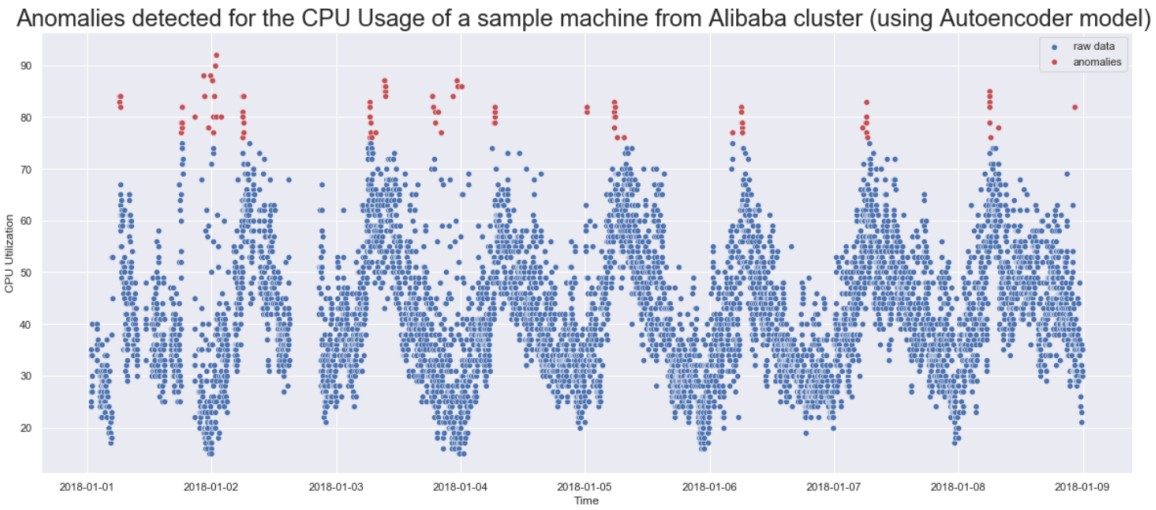


Figure 6.8: CPU Usage of a Sample Machine from Alibaba Cluster

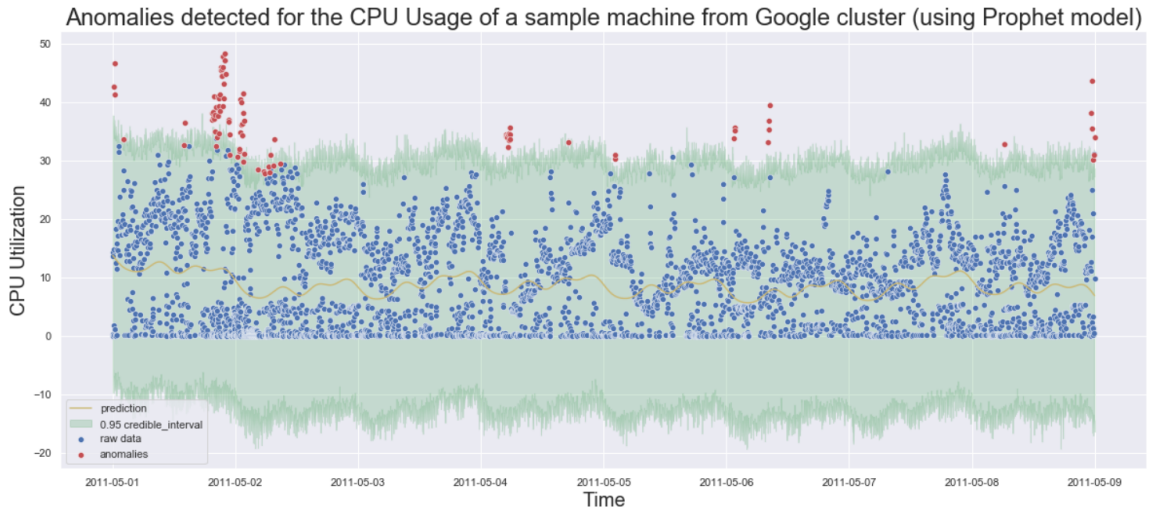


Figure 6.9: CPU Usage of a Sample Machine from Alibaba Cluster

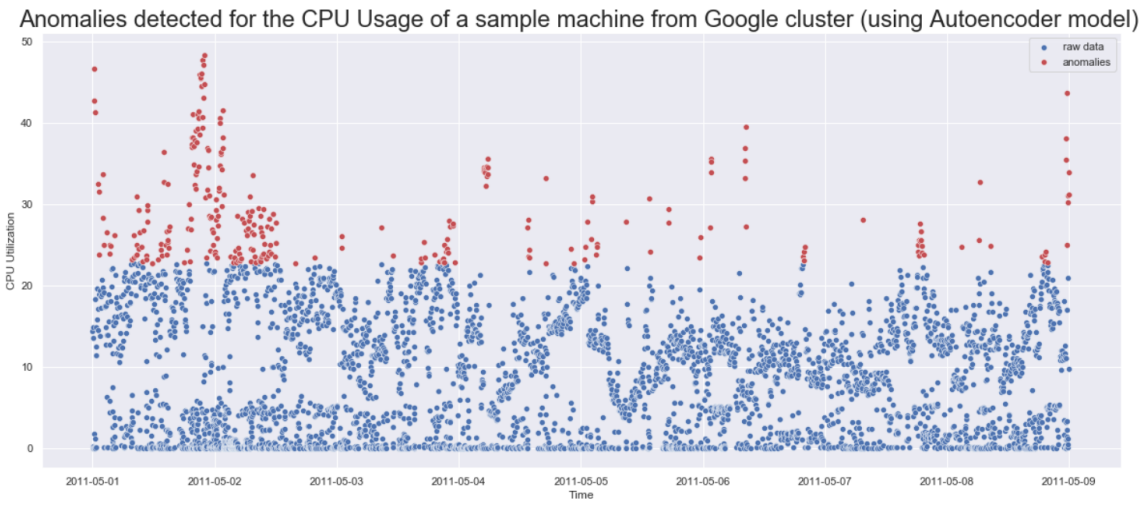


Figure 6.10: CPU Usage of a Sample Machine from Alibaba Cluster

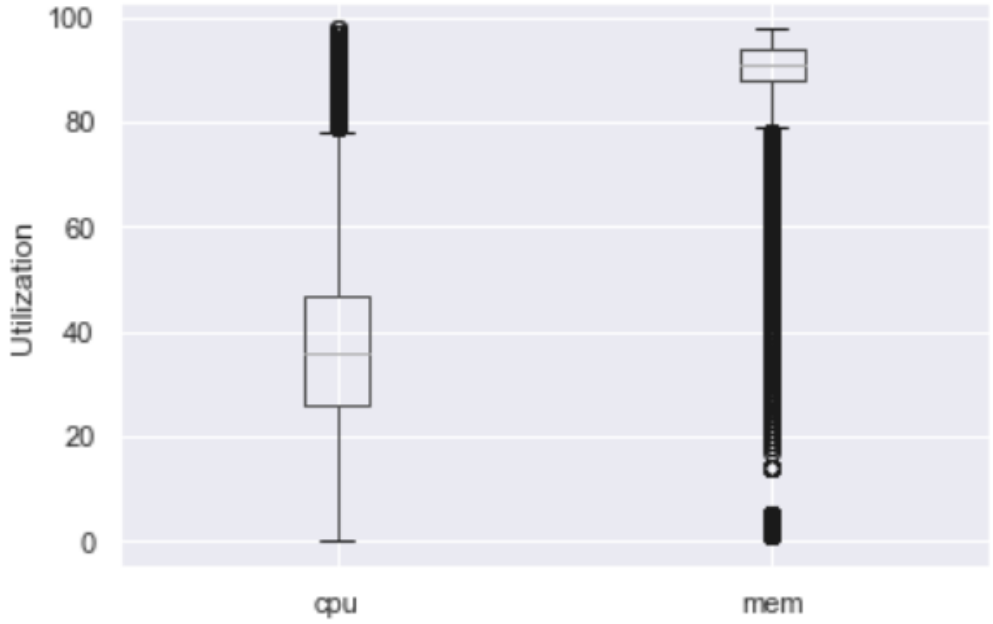


Figure 6.11: Distribution of Resource Usage among physical machines in Alibaba Cluster

Chapter 7

Conclusions

Analysing Alibaba cluster trace reveals important information about the datacentre behavior when it collocates online and batch jobs. This insight can be used to understand and improve scheduling and resource allocation strategies. We carefully extracted valuable statistics about job events and resource utilization in Alibaba datacentre. Using mass-count disparity analysis allowed us to focus on the portion of workload that is more important and disregard the portion that has less effect. In addition, we classified the incoming workload based on their resource demand. Then we improved the workload classification method and applied it in order to assign incoming batch jobs to machines. Instead of regular on-demand resources, we chose spot instances with lower cost for serving batch jobs. We further analyzed Alibaba dataset in machine level and investigated it for anomalies in the resource utilization. This would helpful for decision making about allocation of resources. At the end, we used the predictability of resource usage in datacentre to reduce energy consumption, while SLA requirements are met.

In this work, we have focused on Alibaba dataset and the insights and analysis comes from that. It's mostly because of its specific dimensionality and granularity in different levels, the colocation technology used on their datacentre, and the periodic behavior of servers' resource usage over time which makes it highly predictable. However, we believe that the methodology used in this work can be generalized to other datasets, even for non-periodic ones. Because we are not looking into exact prediction of resource usage, we only need the prediction intervals and estimation of upper and lower bounds. Therefore, the proposed method can be applied on different datasets by changing prediction models or only adjusting prediction parameters.

The main contributions of this work can be summarized as follows:

1. We devised a classification algorithm for workload characterization. We classified instances via a hierarchical clustering algorithm with optimized number of sets, based on the extra variable we have defined, namely demand.

2. We proposed an end-to-end mechanism to allocate spot instances to the arriving batch jobs. The framework first classifies the incoming workload, then predicts the future arrival for each job class. Using this prediction and adopting a optimization-based solution, we decided on assigning the batch workload to the most suitable type of spot instances in order to meet the jobs resource requirements and reduce the cost and consequently the power consumption.

3. We have examined different physical machines in Alibaba datacentre and the online and offline jobs running on them. We found two levels of anomaly among them. First, huge difference in resource utilization between some physical machines which can be caused by uneven co-located workload distribution. Second, unusual behavior in the daily pattern of resource usage on the same machine which can be because of system failure. Real-time detection of these group of anomalies are crucial for the system in order to be able to provide enough resources for the arriving jobs and will lead to increasing performance and saving in the energy.

4. We examined the predictability of resource usage in a datacentre environment. Using quantile regression and machine learning models, we predicted the cpu usage. Since the cpu utilization is linearly correlated with power consumption, we showed that the energy consumption can be cut down by 56% by adopting our prediction schema.

7.1 Future Work

For future work, we suggest suggest some extension of the work either in algorithm or in implementation in order to improve the system model performance and efficiency.

1. For the decision making part, we can take advantage of various decision making algorithms. The algorithm should takes into account prior history of user demand and the prediction for future demand. Markov decision process(MDP) is one of existing methods. Applying MDP, we need to define the reward function

as the difference between the the value obtained from executing a job and the total cost paid for that execution.

2. To solve the problem of assigning resources to jobs, another option is using an online learning algorithm to balance between provider revenue and user satisfaction in term of quality of service. It is be able to learn actively while the requests are arriving. To do this, we first define each jobs in terms of some parameters. The learning algorithm starts from a set of policies, it continuously upgrades per-policy weights based on their performance on job execution to find the optimum policy.

3. Forecasting the future incoming request for each type of virtual machine using auto-regression function.

4. Using the combination of on-demand and spot instances to to service batch jobs to address the trade-off between computation cost and performance.

5. As an implementation suggestion, we can use Amazon fleet in which by having the ability to run across multiple pools, you reduce your application's sensitivity to price spikes that affect a pool or two.

Bibliography

[1] Dror G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*. 1st edition, Cambridge University Press, New York, NY, USA: 2015.

[2] Borg cluster traces from Google. Contribute to google cluster-data development by creating an account on GitHub. Google, 2018.

[3] Google Cluster Workload Traces 2019, Available at: https://research.google/tools/datasets/google-cluster-workload-traces-2019/

[4] Cluster data collected from production clusters in Alibaba for cluster management research, Available at: https://github.com/alibaba/clusterdata.

[5] Alibaba GPU Cluster Trace, Available at: https://github.com/alibaba/clusterdata/tree/master/cluster-trace-gpu-v2020

[6] Abhishek Verma, Luis Pedrosa, Madhukar R. Korupolu, David Oppenheimer, Eric Tune, and John Wilkes, *Large-scale Cluster Management at Google with Borg*, 10th European Conference on Computer Systems, pp. 1-17, 2015.

[7] Li Deng, Yu-Lin Ren, Fei Xu, Heng He, and Chao Li, *Resource Utilization Analysis of Alibaba Cloud*, Intelligent Computing Theories and Application, pp. 183-194, 2018.

[8] Liu, Qixiao, and Zhibin Yu, *The elasticity and plasticity in semi-containerized co-locating cloud workload: a view from Alibaba trace*, ACM Symposium on Cloud Computing, pp. 347-360. 2018.

[9] Reiss, Charles, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch, *Heterogeneity and dynamicity of clouds at scale: Google trace analysis*, ACM symposium on cloud computing, pp. 1-13. 2012.

[10] Christina Delimitrou and Christos Kozyrakis, *Workload Modeling for Computer Systems Performance Evaluation*, ACM Transaction on Computer Systems, vol. 31, no. 4, pp. 12:1-12:34, 2013.

[11] Pawl Janus and Krzysztof Rzadca, *SLO-aware Colocation of Data Center Tasks Based on Instantaneous Processor Requirements*, Symposium on Cloud Computing, pp. 256-268, 2017.

[12] Baris Aksanli, Jagannathan Venkatesh, Liuyi Zhang, and Tajana Rosing, *Utilizing Green Energy Prediction to Schedule Mixed Batch and Service Jobs in Data Centers*, 4th Workshop on Power-Aware Computing and Systems, pp. 1-5, 2011.

[13] Christina Delimitrou, Christos Kozyrakis, *Quasar: Resource-efficient and qos-aware cluster management*, 19th International Conference on Architectural Support for Programming Languages and Operating Systems, pp. 127–144, 2014.

[14] Jason Mars, Lingjia Tang, Robert Hundt, Kevin Skadron, Mary Lou Soffa, *Bubble-up: Increasing utilization in modern warehouse scale computers via sensible co-locations*, 44th Annual IEEE/ACM International Symposium on Microarchitecture, pp. 248–259, 2011.

[15] Maximizing CPU Resource Utilization on Alibaba's Servers, Available at: https://102.alibaba.com/detail/?id=61

[16] The Evolution of Large-Scale Co-Location Technology at Alibaba, available at: https://www.alibabacloud.com/blog/the-evolution-of-large-scale-co-location-technology-at-alibaba_595595.

[17] Amazon Elastic Compute Cloud User Guide for Linux Instances, Available at: https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-ug.pdfusing-spot-instances

[18] Qi Zhang, Quanyan Zhu, Raouf Boutaba, *Dynamic Resource Allocation for Spot Markets in Cloud Computing Environments*, Fourth IEEE International Conference on Utility and Cloud Computing, pp. 178-185, 2011.

[19] Xu, Hong, and Baochun Li, *Dynamic cloud pricing for revenue maximization*, IEEE Transactions on Cloud Computing 1, no. 2, pp. 158-171, 2013.

[20] Wang, Wei, Ben Liang, and Baochun Li, *Revenue maximization with dynamic auctions in IaaS cloud markets*, IEEE/ACM 21st International Symposium on Quality of Service (IWQoS), pp. 1-6, 2013.

[21] Dedicated, On-Demand, Reserved and Spot— Demystifying the Terminology of AWS Instances, available at: https://www.virtana.com/blog/demystifying-terminology-aws-instances/

[22] Omar A. Abdul-Rahman and Kento Aida, *Towards Understanding the Usage Behavior of Google Cloud Users: The Mice and Elephants Phenomenon*, IEEE 6th International Conference on Cloud Computing Technology and Science, pp. 272-277, 2014.

[23] Sheng Di, Derrick Kondo, and Frank Cappello, *Characterizing and modeling cloud applications/jobs on a Google data center* Journal of Supercomputing, vol. 69, no. 1, pp. 139-160, 2014.

[24] Charles Reiss, Alexey Tumanov, Gregory R. Ganger, Randy H. Katz, and Michael A. Kozuch, *Towards understanding heterogeneous clouds at scale: Google trace analysis*, pp. 21, 2012.

[25] Mansaf Alam, Kashish A. Shakil, and Shuchi Sethi, *Analysis and Clustering of Workload in Google Cluster Trace Based on Resource Usage*, IEEE Intl Conference on Computational Science and Engineering (CSE), pp. 740-747, 2016.

[26] Zitao Liu and Sangyeun Cho, *Characterizing Machines and Workloads on a Google Cluster*, 41st International Conference on Parallel Processing Workshops, pp. 397-403, 2012.

[27] Asit K. Mishra, Joseph L. Hellerstein, Walfredo Cirne, and Chita R. Das, *Towards characterizing cloud backend workloads: insights from Google compute clusters*, ACM SIGMETRICS Performance Evaluation Review archive Vol. 37, Issue 4, pp. 34-41, 2010.

[28] Chengzhi Lu, Kejiang Ye, Guoyao Xu, Cheng-Zhong Xu, and Tongxin Bai, *Imbalance in the cloud: An analysis on Alibaba cluster trace*, IEEE International Conference on Big Data (Big Data), pp. 2884-2892, 2017.

[29] Yue Cheng, Zheng Chai, and Ali Anwar, *Characterizing Co-located Datacenter Workloads: An Alibaba Case Study*, Distributed, Parallel and Cluster Computing, pp. 1-3, 2018.

[30] Moshe Babaioff, Yishay Mansour, Noam Nisan, Gali Noti, Carlo Curino, Nar Ganapathy, Ishai Menache, Omer Reingold, Moshe Tennenholtz, Erez Timnat, *ERA: A Framework for Economic Resource Allocation for the Cloud*, 26th International Conference on World Wide Web Companion, Pages 635-642, 2017.

[31] Ishai Menache, Ohad Shamir, Navendu Jain, *On-demand, Spot, or Both: Dynamic Resource Allocation for Executing Batch Jobs in the Cloud*, 11th International Conference on Autonomic Computing, pp. 177-187, 2014.

[32] Murray Stokely, Jim Winget, Ed Keyes, Carrie Grimes, and Benjamin Yolken, *Using a market economy to provision compute resources across planet-wide clusters*, IEEE International Symposium on Parallel and Distributed Processing (IPDPS), pp 1-8, 2009.

[33] Chen, Gong, Wenbo He, Jie Liu, Suman Nath, Leonidas Rigas, Lin Xiao and Feng Zhao, *Energy-Aware Server Provisioning and Load Dispatching for Connection-Intensive Internet Services*, NSDI, Vol. 8, pp. 337-350, 2008.

[34] Srikantaiah, Shekhar, Aman Kansal, and Feng Zhao. *Energy aware consolidation for cloud computing*, 2008.

[35] Chen Hui, Mukil Kesavan, Karsten Schwan, Ada Gavrilovska, Pramod Kumar, and Yogendra Joshi, *Spatially-aware optimization of energy consumption in consolidated data center systems*, International Electronic Packaging Technical Conference and Exhibition, Vol. 44625, pp. 461-470, 2011.

[36] Bradley, David J., Richard E. Harper, and Steven W. Hunter, *Workload-based power management for parallel computer systems*, IBM Journal of Research and Development 47, no 5.6, pp. 703-718, 2003.

[37] Goiri, Inigo, Ferran Julia, Ramon Nou, Josep Ll Berral, Jordi Guitart, and Jordi Torres, *Energy-aware scheduling in virtualized Data Centers*, IEEE International Conference on Cluster Computing, pp. 58-67, 2010.

[38] Farahnakian, Fahimeh, Pasi Liljeberg, and Juha Plosila, *LiRCUP: Linear regression based CPU usage prediction algorithm for live migration of virtual machines in data centers*, 39th Euromicro conference on software engineering and advanced applications, pp. 357-364, 2013.

[39] Hsieh, Sun-Yuan, Cheng-Sheng Liu, Rajkumar Buyya, and Albert Y. Zomaya, *Utilization-prediction-aware virtual machine consolidation approach for energy-efficient cloud data centers*, Journal of Parallel and Distributed Computing 139, pp. 99-109, 2020.

[40] Mehiar Dabbagh, Bechir Hamdaoui, Mohsen Guizani, and Ammar Rayes, *Energy-efficient cloud resource management*, IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), pp. 386-391, 2014.

[41] Yang, Ke, Hong Zhang, and Peilin Hong. *Energy-aware service function placement for service function chaining in data centers*, IEEE Global Communications Conference (GLOBECOM), pp. 1-6, 2016.

[42] Zhang, Xiaoning, Zhichao Xu, Lang Fan, Shui Yu, and Youyang Qu, *Near-optimal energy-efficient algorithm for virtual network function placement*, IEEE Transactions on Cloud Computing, 2019.

[43] Kim, Siri, Yunjung Han, and Sungyong Park. *An energy-aware service function chaining and reconfiguration algorithm in NFV*, IEEE 1st International Workshops on Foundations and Applications of Self* Systems (FAS* W), pp. 54-59, 2016.

[44] Sun, Gang, Yayu Li, Hongfang Yu, Athanasios V. Vasilakos, Xiaojiang Du, and Mohsen Guizani, *Energy-efficient and traffic-aware service function chaining orchestration in multi-domain networks*, Future Generation Computer Systems 91, pp. 347-360, 2019.

[45] Dror G. Feitelson, *Metrics for mass-count disparity*, 14th IEEE International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems (MASCOTS). IEEE Computer Society, pp. 61–68, 2006.

[46] Mark E. Crovella, *Performance Evaluation with Heavy Tailed Distributions*, Workshop on Job Scheduling Strategies for Parallel, pp. 1-9, 2001.

[47] Naghmeh Dezhabad, Sudhakar Ganti, and Golamali Shoja, *Cloud Workload Characterization and Profiling for Resource Allocation*, IEEE 8th International Conference on Cloud Networking (CloudNet), pp. 1-4, 2019.

[48] Leonard Kaufman and Peter J. Rousseeuw, *Finding Groups in Data: An Introduction to Cluster Analysis*, John Wiley Sons, Vol. 344, 2009.

[49] K. F. Hallock and R. Koenker, *Quantile regression*, Journal of Economic Perspective. Vol. 15, pp. 143–156, 2001.

[50] Alexander Schrijver, *Theory of linear and integer programming*, John Wiley Sons, 1998.

[51] Ncibi, Kais, Tarek Sadraoui, Mili Faycel, and Amor Djenina, *A Multilayer perceptron artificial neural networks based a preprocessing and hybrid optimization task for data mining and classification*, International Journal of Economic Finance Managment 5, pp. 12-21, 2017.

[52] Ailsa H. Land and Alison G. Doig, *An Automatic Method of Solving Discrete Programming Problems*, In 50 Years of Integer Programming 1958-2008, Springer, pp. 105-132, 2010.

[53] Kirkpatrick, Scott, C. Daniel Gelatt, and Mario P. Vecchi, *Optimization by simulated annealing*, science 220, no. 4598, pp. 671-680, 1983.

[54] Richard W. Eglese, *Simulated annealing: a tool for operational research*, European Journal of Operational Research, vol. 46, no 3, pp. 271–281, 1990.

[55] John H. Holland, *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, Michigan, 1975, re-issued by MIT Press 1992.

[56] David E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning*, 1st Edition, Addison-Wesley Longman Publishing Co., USA, 1989.

[57] Rabih Bashroush, Andy Lawrence, *Beyond PUE: Tackling IT's Wasted Terawatts*, Available at: https://uptimeinstitute. com/beyond-puetackling-it's-wasted-terawatts, 2020.

[58] Arman Shehabi, Sarah Josephine Smith, Dale A Sartor, Richard E Brown, Magnus Herrlin, Jonathan Koomey, Eric Masanet, Nathaniel Horner, Inês Azevedo, and William Lintner, *United States Data Center Energy Usage Report*, 2016.

[59] Kilcioglu Cinar, Justin M. Rao, Aadharsh Kannan, and R. Preston McAfee, *Usage patterns and the economics of the public cloud*, 26th International Conference on World Wide Web, pp. 83-91, 2017.

[60] Ahemd Alutaibi and Sudhakar Ganti, *Network Traffic Prediction using Quantile Regression with linear, Tree, and Deep Learning Models*, 45th IEEE Conference on Local Computer Networks (LCN), pp. 421-424, 2020.

[61] Nikolaos Pandis, *Linear regression*, Statistics and Research Design, Vol. 149, pp. 431-434, 2016.

[62] Joos Korstanje, *Assumptions of linear regression*, available at: https://towardsdatascience.com/assumptions-of-linear-regression-fdb71ebeaa8b

[63] Shrestha, Durga L., and Dimitri P. Solomatine, *Machine learning approaches for estimation of prediction interval for the model output*, Neural Networks 19, no 2, pp. 225-235, 2006.

[64] K. F. Hallock and R. Koenker, "Quantile regression", Journal of Economic Perspective. Vol. 15, pp. 143–156, 2001.

[65] Carlo Mastroianni, Michela Meo, Giuseppe Papuzzo, *Probabilistic consolidation of virtual machines in self-organizing cloud data centers*, IEEE Transaction on Cloud Computing 1, no.2, pp. 215-228, 2013.

[66] Jacob Leverich, Matteo Monchiero, Vanish Talwar, Parthasarathy Ranganathan, and Christos Kozyrakis, *Power management of Data Center workloads using per-core power gating*, Computer Architecture Letters, vol. 8, no. 2, pp. 48-51, 2009.

[67] George E.P. Box, and Gwilym M. Jenkins, *Time Series Analysis, Forecasting and Control*, Holden-Day, pp. 199-201, 1970.

[68] Koenker, Roger, and Gilbert Bassett Jr. *Regression quantiles*, Econometrica: journal of the Econometric Society, pp. 33-50, 1978.

[69] Choi, Seung-Whan, *The effect of outliers on regression analysis: regime type and foreign direct investment*, Quarterly Journal of Political Science 4, no. 2, pp. 153-165, 2009.

[70] Computational Complexity of Learning Algorithms, available at: https://www.thekerneltrip.com/machine/learning/computational-complexity-learning-algorithms/

[71] Leo Breiman, *Random Forests*, Machine Learning, 45(1), pp. 5-32, 2001.

[72] Jerome Friedman, Hastie Trevor, and Robert Tibshirani, *The elements of statistical learning*, Vol. 1, No. 10, Springer series in statistics, 2001.

[73] Meinshausen, Nicolai and Greg Ridgeway, *Quantile regression forests*, Journal of Machine Learning Research 7, no. 6, pp. 983-999, 2006.

[74] Friedman, Jerome H., *Greedy function approximation: a gradient boosting machine*, Annals of statistics, pp. 1189-1232, 2001.

[75] Ogutu, Joseph O., Hans-Peter Piepho, and Torben Schulz-Streeck, *A comparison of random forests, boosting and support vector machines for genomic selection*, In BMC proceedings, Vol. 5, No. 3, pp. 1-5, BioMed Central, 2011.

[76] Gu, Jiuxiang, Zhenhua Wang, Jason Kuen, Lianyang Ma, Amir Shahroudy, Bing Shuai, Ting Liu, *Recent advances in convolutional neural networks*, Pattern Recognition 77, pp. 354-377, 2018.

[77] Mikolov, Tomas, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpu, *Recurrent neural network based language model*, 11th annual conference of the international speech communication association, vol. 2, no. 3, pp. 1045-1048. 2010.

[78] Graves, Alex, Marcus Liwicki, Santiago Fernández, Roman Bertolami, Horst Bunke, and Jürgen Schmidhuber, *A novel connectionist system for unconstrained handwriting recognition*, IEEE transactions on pattern analysis and machine intelligence 31, no. 5, pp. 855-868, 2008.

[79] Gers, Felix A., Jürgen Schmidhuber, and Fred Cummins, *Learning to forget: Continual prediction with LSTM*, Neural computation 12, no. 10, pp. 2451-2471, 2000.

[80] Felix A. Gers and Jurgen Schmidhuber, *LSTM recurrent networks learn simple context free and context sensitive languages*, IEEE Transactions on Neural Networks , vol. 12, no. 6, pp. 1333-1340, 2001

[81] Bergstra, James, and Yoshua Bengio, *Random search for hyper-parameter optimization* Journal of machine learning research, Vol. 13, no. 2, pp. 281-305, 2012.

[82] Azzouni, Abdelhadi, and Guy Pujolle, *NeuTM: A neural network-based framework for traffic matrix prediction in SDN*, IEEE/IFIP Network Operations and Management Symposium, pp. 1-5, 2018.

[83] Fan, Xiaobo, Wolf-Dietrich Weber, and Luiz Andre Barroso, *Power provisioning for a warehouse-sized computer*, ACM SIGARCH computer architecture news, no. 2, pp. 13-23, 2007.

[84] Sparsh, Mittal and Zhao Zhang, *EnCache: Improving cache energy efficiency using a software-controlled profiling cache*, in IEEE International Conference On Electro/Information Technology, USA, 2012.

[85] Facebook Prophet Model, Available at: https://facebook.github.io/prophet/

[86] Ian Goodfellow, Yoshua Bengio, and Aaron Courville, *Deep learning*, MIT press, 2016.