Characterizing and Orchestrating VM Reservation in Geo-distributed Clouds to Improve the Resource Efficiency

Jiuchen Shi*, Kaihua Fu*, Quan Chen*, Changpeng Yang◇, Pengfei Huang◇, Mosong Zhou◇, Jieru Zhao*, Chen Chen*, Minyi Guo*
{shijiuchen,midway}@sjtu.edu.cn,chen-quan@cs.sjtu.edu.cn
{yangchangpeng,huangpengfei12,zhoumosong}@huawei.com
{zhao-jieru,chen-chen}@sjtu.edu.cn,guo-my@cs.sjtu.edu.cn
*Shanghai Jiao Tong University,◇ Huawei Cloud

ABSTRACT

Cloud providers often build a geo-distributed cloud from multiple datacenters in different geographic regions, to serve tenants at different locations. The tenants that run large scale applications often reserve resources based on their peak loads in the region close to the end users to handle the ever changing application load, wasting a large amount of resources. We therefore characterize the VM request patterns of the top tenants in our production public geo-distributed cloud, and open-source the VM request traces in four months from the top 20 tenants of our cloud. The characterization shows that the resource usage of large tenants has various temporal and spatial patterns on the dimensions of time series, regions, and VM types, and has the potential of peak shaving between different tenants to further reduce the resource reservation cost. Based on the findings, we propose a resource reservation and VM request scheduling scheme named **ROS** to minimize the resource reservation cost while satisfying the VM allocation requests. Our experiments show that ROS reduces the overall deployment cost by 75.4% and the reservation resources by 60.1%, compared to the tenant-specified reservation strategy.

CCS CONCEPTS

• **Computer systems organization → Cloud computing**.

Quan Chen and Minyi Guo are the corresponding authors.

SoCC '22, November 7–11, 2022, San Francisco, CA, USA

ACM ISBN 978-1-4503-9414-7/22/11…$15.00
https://doi.org/10.1145/3542929.3563490

KEYWORDS

Geo-distributed cloud, Resource reservation, Trace analysis

ACM Reference Format:
Jiuchen Shi, Kaihua Fu, Quan Chen, Changpeng Yang, Pengfei Huang, Mosong Zhou, Jieru Zhao, Chen Chen, Minyi Guo. 2022. Characterizing and Orchestrating VM Reservation in Geo-distributed Clouds to Improve the Resource Efficiency. In *Symposium on Cloud Computing (SoCC '22), November 7–11, 2022, San Francisco, CA, USA.* ACM, New York, NY, USA, 16 pages. https://doi.org/10.1145/3542929.3563490

1 INTRODUCTION

Public cloud providers often build multiple datacenters in different geographic regions [6, 13, 19, 22], to serve tenants at different locations. The hardware resources in a datacenter are often organized into resource pools, and cloud providers usually provide different virtual machine (VM) types for tenants to choose [3, 12, 17, 21]. These regions and VM types usually have different resource cost coefficients [3, 4, 12, 15, 17, 18, 21, 23], due to the different energy charges and VM performance.

These geo-distributed clouds often host large-scale applications, e.g., streaming video applications (like youtube) and social network applications (like twitter and facebook). The actual loads of such tenant-facing applications often change dynamically due to the unplanned load spike or diurnal load pattern [32, 37, 58], requiring different amounts of resources to achieve the strict Quality-of-Service (QoS) [59, 62]. To this end, current clouds (e.g., AWS, Azure, Google Cloud) allow the tenants to reserve VMs with preferred specifications in the preferred regions [5, 7, 16, 20, 24]. And the tenants often claim to reserve VMs according to their peak possible loads in the cloud regions that close to their end users.

In this paper, we first analyze the resource (CPU core and memory space) reservation patterns of these tenants in our in-production geo-distributed cloud with 17 regions[1].

[1]The traces are open-sourced via https://github.com/shijiuchen/HuaweiCloud-GeoVMTraces.

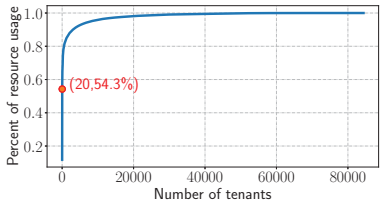


Figure 1: The aggregated resource usage of the tenants in the descending order in our geo-distributed cloud.

We observe that the tenant-specified resource reservation policy causes two main problems. First, a tenant often only uses all the reserved resources for a short time, the reserved resources result in the huge resource waste. For instance, in our geo-distributed cloud, at most 68.6% of the reserved resources are actually used in a region, while only 1.0% of the reserved resources are used in the worst case (Section 4.1). Second, the reservation is often placed on the expensive regions that close to the end tenants or the expensive VM types, resources on the expensive regions/VM types may be unnecessarily reserved (Section 4.2). It is beneficial to reserve "just-enough" resources on cheaper regions/VM types that are geographically close and have similar performance, while still ensuring the required service-level agreement (SLA).

Figure 1 shows the percentage of resource usage (*cores* × *hours*) of the 84379 tenants in our traces. As observed, the top 20 tenants use 54.3% of the all resource usages. While a few large tenants use a large percentage of the resources, by persuading them to allow flexible cross-datacenter VM reservation/scheduling with a lower price and ensured SLA, there is an opportunity to greatly improve the resource efficiency of the geo-distributed cloud.

It is nontrivial to take the above opportunity, as the resource usage of large tenants show different temporal (Section 5.1) and spatial (Section 5.2) patterns. The temporal pattern presents a tenant's resource usage amount over time. For instance, a video application [52] often has diurnal load pattern, and a social-network application [14] has bursty request pattern for the breaking news. The spatial pattern shows a tenant's resource usage pattern on different regions and VM types. For instance, a tenant may deploy VMs in multiple regions that close to the end-users for short response latency [9, 26, 43] or deploy on the VMs with various specifications that satisfy the performance requirement [2, 56].

Moreover, we also observe that some large tenants have complementary temporal and spatial resource usage patterns. When a tenant uses a small amount of resources (a large amount of resources are reserved actually), some other tenants use most of its reserved resources, and vice versa. This is because these tenants often run different applications, and their end users have different access behaviors. By orchestrating the resources of these complementary tenants in the same region of the geo-distributed cloud, the resource reservation cost can be further reduced.

Technically, three challenges should be resolved to improve the resource efficiency. First, the resource usage pattern of a tenant should be precisely predicted, so that we can reserve "just-enough" resources for the tenant. Second, to avoid the case that too many resources are unnecessarily reserved on the high cost regions or VM types, a cross-region resource orchestrator is required to carefully distributes the reserved resources of each tenant to different regions/VM types considering the tenants' spatial resource usage patterns. Last, an online scheduling mechanism is required to quickly allocate more resources to a tenant, as the load of an application may burst occasionally.

We therefore further propose a runtime scheme named **ROS** to orchestrate the resources in a geo-distributed cloud. ROS optimizes the resource orchestration of multiple tenants together based on their temporal and spatial resource usage patterns. ROS comprises a *load pattern predictor*, a *cross-region resource orchestrator*, and a *bursty-aware scheduler* to resolve the above challenges. The predictor uses different prediction methods to predict the resource usage of large tenants. Considering different costs of regions/VM types and the complementary of large tenants, the orchestrator takes the predicted load patterns as input, and orchestrates the reservation resources onto different regions/VM types, to reduce the overall deployment cost. The online scheduler schedules VM requests and compensates the bursty and irregular requests at runtime. We conduct simulations based on our traces, and the results show ROS reduces the overall deployment cost by 75.4% and the reserved resources by 60.1%. We have adopted ROS in our production geo-distributed cloud and show similar performance as the simulation.

We summarize the contributions as follows:

- **An open-sourced VM request dataset of our production geo-distributed cloud**. The dataset includes the VM request traces of the top 20 tenants in 4 months. To the best of our knowledge, it is the first VM request dataset of large tenants in a geo-distributed cloud.
- **The comprehensive analysis of the VM requests in our production-level geo-distributed cloud.** The insights obtained from the analysis identify the opportunity to reduce the resource reservation cost through cross-region reservation and scheduling.
- **The design of a resource orchestrating and scheduling scheme.** The scheme minimizes the resource reservation cost while satisfying the requests of VMs. The simulation results show it can reduce both the overall deployment cost and the reserved resources.

Our key contributions are the open-sourced traces and the comprehensive analysis on the VM usage patterns of large

tenants in our geo-distributed cloud. ROS is designed based on the analysis, which also has multiple technical novelties.

2 RELATED WORK

In this section, we discuss the current work on analyzing the traces of popular cloud providers and/or managing resources within or across datacenters.

2.1 Cloud Trace Analysis

Popular cloud providers released production traces of both the public and private clouds [10, 30, 39, 44, 51, 53]. As the number of the open-sourced traces is limited, we compare all the traces we know with ours. Table 1 makes the comparison.

In more detail, as for private clouds, Google provided a one month trace from its Borg cluster in 2011 [44], and published an extension version of the trace with 8 different clusters in 2019 [53]. These traces focused on the internal container-based workloads in their private cloud. Alibaba Cloud released a publicly accessible dataset in 2017 with 1300 machines in a period of 12 hours, which contains both the Latency-Critical (LC) and batch workloads [10, 39]. In 2018, Alibaba also published a larger scale dataset of the LC and batch workloads, which contains the DAG information of the batch workloads [30, 51]. Alibaba's traces also focus on their internal applications in a private cloud environment. In contract, we characterize the cloud VM workloads with the IaaS mode of the public cloud.

As for public clouds, Microsoft Azure open-sourced their first-party and third-party VM workloads over three months in 2017 and 2019 [25]. The work analyzed the temporal behaviors of the VMs to show their resource usage is predictable. Moreover, Microsoft Azure also published one VM request trace specifically for investigating their VM scheduling algorithms [31]. Compared with these traces, our traces focus on both the temporal and spatial characteristics of the resource usage of large tenants' VM requests in our geo-distributed cloud. This analysis perspective assists us to investigate how to reduce the deployment costs when make resource reservation for large tenants.

There are also other trace analysis works with other types of workloads for the public cloud. Microsoft Azure published two function traces in 2019 [46] and 2021 [60], respectively. Alibaba published their trace analysis about machine learning (ML) workloads and microservices in 2020 [57] and 2021 [40], respectively. Different from our traces, these traces are not about VM workloads with the IaaS mode.

2.2 Scheduling within/across Datacenters

In the aspect of task scheduling inside the datacenter, a series of works have been proposed to schedule VMs, containers, and functions, with the classification of centralized [25, 29, 50, 53, 54], distributed (two-level [33] and share state [8, 45, 61]), and hybrid [31, 55] scheduling architectures. Their common goal is to seek high scheduling quality under low latency scheduling decisions. These schedulers focused on task-level scheduling within the datacenter, so they cannot be applied to reserve resources and schedule VMs geographically.

Table 1: Comparisons between our traces and others

	Geo-distributed	VM Workload	Tenant Characteristic
Google Cluster [44, 53]			
Alibaba Cluster [39, 51]			
Alibaba GPU [57]			
Alibaba Microservice [40]			
Azure Function [46, 60]			
Azure VM [25, 31]		✔	
Ours	✔	✔	✔

In terms of geo-distributed task scheduling, most of the works focused on managing the network traffic among different datacenters. Yugong [35] managed the network traffic through project placement, table replication, and job outsourcing, to save the public network bandwidth. Taiji [11] assigned traffic objects geographically through modeling the network traffic routing as an assignment problem, to satisfy the service-level objectives. Gaia [34] eliminated insignificant communications between datacenters while maintaining the correctness of an ML algorithm. ELIS [47] and Nautilus [28] optimized the network traffic in public network when deploying microservices among datacenters and edges to achieve better service latency. However, none of these works took into account the resource reservation under different resource costs of datacenters, which aims at minimizing the computational resource reservation cost.

In addition, current cloud providers reserved resources based on the maximum required amount and instance locations specified by cloud tenants [5, 7, 16, 20, 24]. However, cloud tenants cannot fully utilize the reserved resources most of the time and the reservation positions may have high resource cost coefficients, resulting in huge resource reservation cost. Moreover, Narayanan et al. [42] proposed a geo-distributed capacity planning strategy to optimize the deployment cost, but orchestrated resources for each tenant independently, which leads to poor resource efficiency.

To conclude, prior researches on public cloud resource management and task scheduling to date lacked a thorough understanding of the key characteristics of large tenants' temporal and spatial patterns of large commercial providers.

3 BACKGROUND AND TERMINOLOGY

Our VM request traces include a total of the top 20 large tenants on 17 regions and dozens of VM types during 4 months of the year 2021. Each request specifies the tenant demand, e.g., the region and VM type. In this section, we introduce the background and terminology used in our paper. Figure 2 shows the overview of our geo-distributed cloud.

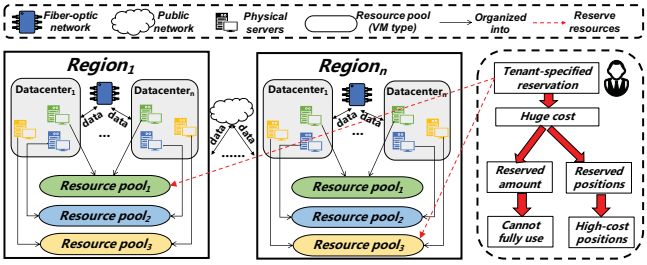


Figure 2: Overview of our geo-distributed cloud.

3.1 Cloud Regions and Datacenters

As shown in Figure 2, the datacenters of our cloud platform are built in multiple geographic positions. Regions in our cloud platform are partitioned based on their geographic positions. Public services, e.g., elastic computing, block storage, and VPC network are shared in the same region. The datacenters in a region are connected through high-speed fiber-optic network to meet cloud tenants' requirements for building high availability systems across datacenters.

As different datacenters are in the same geographic position, the spatial characteristics of large cloud tenants' resource usage lie in the selection of different regions. We use the resource usage of different regions as one aspect to analyze the spatial characteristics, and also reserve resources and schedule VM requests to different regions for large tenants. In this paper, we use ri to represent a specific region. Moreover, in order to facilitate the statistics in the later data analysis sections, we divide all regions into three region sets marked with *Region Set 1*, *Region Set 2*, and *Region Set 3*, and the geographical positions in each set are relatively close.

3.2 Resource Pools and VM Types

As shown in Figure 2, the heterogeneous hardware resources of each region are organized into multiple types of resource pools (marked with different colors), and each resource pool has the same type of physical servers from datacenters in this region. Different regions may have the same type of resource pool as they may have the same type of physical servers. Different resource pools have different capacities and different cost coefficients. Our cloud platform provides different types of VMs marked with different VM name for cloud tenants to choose according to their demand.

For an example VM named *"s2.medium.4"*, *"s2"* represents the type of physical servers the VM on, *"medium"* represents the number of CPU cores, and *"4"* represents the ratio of memory (GB) to CPU cores. Therefore, *"s2"* is corresponding to the type of the resource pool, which we name it as VM type. Since different large cloud tenants have their specific resource usage demand for different VM types, we take the resource usage on different VM types as another aspect to analyze the spatial characteristics. Moreover, combing the spatial characteristic of regions, our scheme aims at reserving resources and scheduling VM requests to different VM types of different regions. For different VM types, we use vi to represent a specific VM type.

3.3 Resource Reservation

The tenants sometimes need more resources than their daily usage due to some special cases, e.g., unplanned load spike and large-scale VM migrations. To handle these situations, current cloud providers provide reserved instances or services, and the reserved resource amount and reserved instance positions are all specified by the cloud tenants themselves [5, 7, 16, 20, 24], as shown in the right part of Figure 2.

These reservation methods ensure the high availability of resources for large tenants, but bring huge deployment cost to cloud providers. The reasons lie on: (1) cloud tenants cannot fully use the reserved resources most of the time, (2) tenant-specified reserved instance positions may have high resource cost coefficients. In this paper, we analyze the temporal and spatial patterns of large cloud tenants, and design corresponding methods to reduce the deployment cost of resource reservation. In the data analysis of this paper, we only show the statistical results of CPU resources since the space limitation. The results of memory resources show the similar patterns as the CPU results.

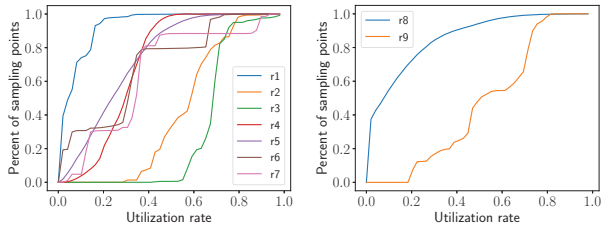
4 RESOURCE RESERVATION STATUS

In this section, we analyze the current situation of the resource reservation in our cloud. We first show the resource utilization of the reserved resources. Then, we present the resource usage distribution of different regions/VM types, and further propose the *acceptable spatial range*.

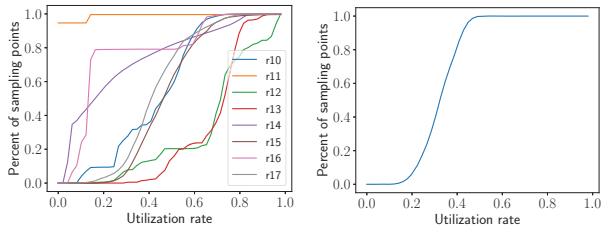
4.1 Utilization of Reserved Resources

To ensure the resources are available when large tenants need more resources than their normal usage (e.g., unplanned load spike), we reserve resources for them in advance. The amount of reservation resources is specified by large tenants themselves according to their maximum demand. However, most large tenants' daily resource usage cannot fully utilize the reserved resources at most of the time, which can result in a waste of resources.

To explore the degree of resource wasting, we sample the resource usage of different regions every 10 minutes, and calculate the resource utilization rate relative to the total reserved resources of all the large tenants in each region. Figure 3 reports the distribution of the resource utilization rate in all sampling points for all the regions. We present the results of the regions in *Region Set 1*, *Region Set 2*, and *Region Set 3* (defined in Section 3.1) in Figure 3(a), Figure 3(b), and Figure 3(c), respectively. We can find that most of the



(a) Regions in *Region Set 1*. (b) Regions in *Region Set 2*.



(c) Regions in *Region Set 3*. (d) Overall resource utilization rate.

Figure 3: The distribution of resource utilization rate.

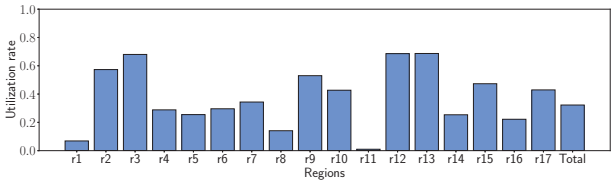


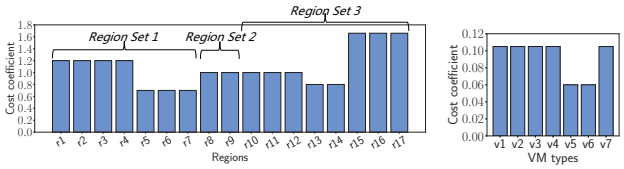
Figure 4: Average resource utilization rate relative to the reserved amount of different regions.

sampling points of the regions have low utilization rate relative to their reserved resources, although the distribution of different regions are not similar. The special curve of *r11* in Figure 3(c) is caused by: the resource usage amount is 0 for most of the sampling points, although we have reserved some resources on this region for large tenants. For the distribution of overall resource utilization rate shown in Figure 3(d), the sampling points are major distributed in the range of 20%-50%. Figure 4 shows the average utilization rate of different regions and the total utilization rate relative to their reserved resources. The average value is in the range of 1.0%-68.7% and the average total utilization rate is 32.3%. These results of low resource utilization prove that there is a large amount of resource waste for the reserved resources.

It is of great potential to reduce resource reservation waste if we reserve resources for large tenants according to their resource usage patterns. The exploration of the resource usage temporal patterns of large tenants is a necessity.

4.2 Resource Usage Distribution

In the geo-distributed cloud environment, different regions or VM types may have different resource cost coefficients.



(a) Regions. (b) VM types.

Figure 5: Cost coefficients of regions or VM types.

Figure 6: Resource usage distribution of regions in different region sets.

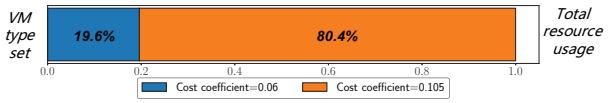


Figure 7: Resource usage distribution of VM types.

Therefore, it is expected to distribute the tenant requests to the regions or VM types with smaller cost coefficients, which can reduce the overall deployment cost.

We first explore large tenants' resource usage distribution of different regions. Figure 5(a) shows the different resource cost coefficients of different regions. A region's cost coefficient is its maintenance cost (e.g., the cost for real estate and power) normalized to the cost of region r8. We can find there are 2 different values (1.2 and 0.7) in *Region Set 1*, 1 value (1.0) in *Region Set 2*, and 3 different values (1.0, 0.8, and 1.66) in *Region Set 3*, respectively. Since regions in the same *Region Set* are geographically close, most of the VM requests in the regions of *Region Set i* can be scheduled to any region within this range without affecting the response time (caused by geographical positions). Figure 6 shows large tenants' current resource usage distribution with different resource cost coefficients in the regions of *Region Set 1*, *Region Set 2*, and *Region Set 3*, respectively. We can observe that only 30.2% and 14.0% or the resources are distributed to the low-cost regions in *Region Set 1* and *Region Set 3*, respectively. Therefore, current distribution of tenant resource usage does not tend to low-cost regions.

Moreover, we also explore the resource usage distribution of different VM types. Figure 5 shows the resource cost coefficients of 7 kinds of different VM types. The cost coefficient of a VM type is its selling price with 1vCPU and 2GiB memory. We can find that these 7 VM types have 2 different resource coefficient values (0.06 and 0.105). Moreover, after

communicating with large tenants, we have learned that most of their workloads can be conducted on any one of the 7 VM types without having different execution performance. Figure 7 shows large tenants' current resource usage distribution with different resource cost coefficients on the 7 kinds of VM types. Similar to the result of the regions, we can also find that only 19.6% of the resources are distributed to low-cost VM types. This result proves that current distribution of tenant resources does not tend to low-cost VM types.

It's of great potential to reduce the deployment cost through reserving resources or scheduling VMs on the low-cost regions/VM types. Therefore, we need to explore the spatial (region/VM type) resource usage patterns of large tenants.

4.3 Defining Acceptable Spatial Ranges

The inter-datacenter scheduling is more complex than the intra-datacenter scheduling, as datacenters have different cost coefficients and network latencies. Combing the observations in Section 4.2, we determine the specific *acceptable spatial ranges* on resource usage for each large tenant through negotiating with them in advance. This helps us to reserve tenants' resources on low-cost regions/VM types under their SLA of network latency and VM performance.

The *acceptable spatial ranges* consider the demand of network latencies and VM performance of large tenants, and thus includes two meanings. First, the tenant's reserved resources or VM requests on regions of *ri*, ..., *rj* can be reserved or scheduled within these regions arbitrarily, due to these regions are geographically close and the network latencies are acceptable for tenants. For instance, different region sets *Region Set 1*, *Region Set 2*, and *Region Set 3*. Second, the tenant's reserved resources or VM requests on VM types of *vi*, ..., *vj* can be reserved or scheduled within these VM types arbitrarily, due to these VM types all have acceptable performance for this tenant's workloads which satisfies the SLA demand of performance. For instance, the 7 VM types used in Figure 7.

Our resource orchestrating and scheduling scheme considers the *acceptable spatial ranges* of large tenants into resource reservation and request scheduling, to minimize the deployment cost while satisfying large tenants' required SLA.

4.4 Main Takeaways

From the analysis of reserved resource utilization and distribution, we now reiterate our 2 observations.

First, the resource reservation for large tenants causes huge resource waste. It is desired to reserve resources for tenants on demand by analyzing their resource usage patterns and predicting their recent required resource amount, rather than based on the tenant-specified maximum value at all times. Second, current resource usage distribution of large tenants are not aware of the cost of regions and VM types. Specifically, less than 30% of the large tenants' resource usage are distributed on the low-cost regions or VM types. Therefore, it has great potential to reserve resource or schedule VM requests to low-cost regions and VM types for reducing the overall deployment cost.

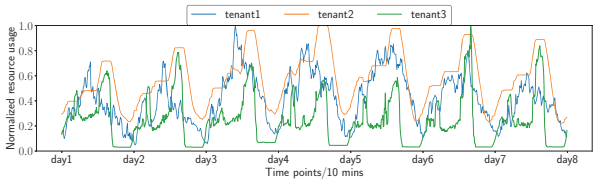


Figure 8: The resource usage of three tenants with the diurnal pattern in one week. A tenant's resource usage is normalized to its maximum value.

To conclude, no matter reducing the reserved resources or reserving resources in low-cost regions/VM types, the foundation lies on the analysis of large tenant's resource usage patterns. This motivates us to analyze the resource usage patterns of large tenants in Section 5.

5 TEMPORAL AND SPATIAL RESOURCE USAGE PATTERNS

In this section, we analyze the resource usage patterns of large tenants, classify the patterns into multiple types, and propose the corresponding prediction methods. Then, we break down the resource usage of large tenants into regions and VM types to explore the spatial characteristics. Furthermore, we explore the temporal and spatial potential of peak shaving among different tenants.

5.1 Temporal Usage Patterns

The analysis of different temporal patterns of large tenants can help us to predict their resource usage, and then reserve resources based on the predicted results to reduce the resource waste. In this section, we explore the temporal patterns of our 20 large tenants and classify them into 4 different types, i.e., the diurnal load pattern, persistent usage pattern, bursty usage pattern, and irregular usage pattern.

5.1.1 Diurnal usage pattern. The diurnal pattern of resource usage often shows typical day and night mode. We find that 6 of the 20 top large tenants show obvious diurnal temporal patterns, and Figure 8 shows the resource usage of the 3 example large tenants in one week.

Except for the obvious day and night pattern, we can find that the peak time of the 3 example tenants is not the same. This brings the potential of reducing the reserved resources through peak shaving between different large tenants. In addition, *tenant3* has a small peak and a large peak at forenoon and night, respectively. This reminds us that the diurnal tenants may have multiple peaks in a day. For diurnal tenants,

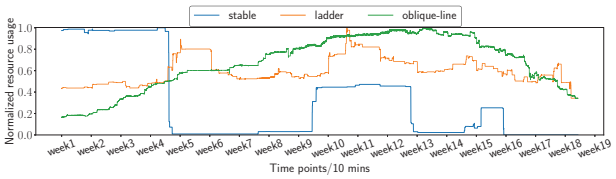


Figure 9: The resource usage of three tenants with the persistent pattern in four months. A tenant's resource usage is normalized to its maximum usage value.

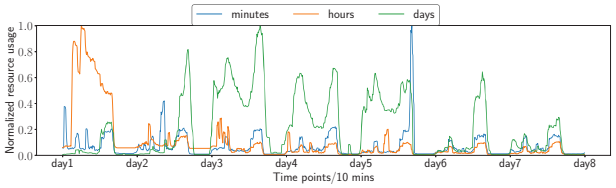


Figure 10: The resource usage of three example bursty patterns in one week. Each case's resource usage is normalized to its maximum value.

we also find that there is an overall resource usage upward and downward in the long-term trend.

Combining the short-term and long-term patterns of diurnal tenants, the resource usage can be predicted by using time series prediction methods, e.g., LSTM [48] and ARIMA [49].

5.1.2 Persistent usage pattern. The resource usage with a persistent pattern usually has fewer changes at a long period of time and maintains unchanged within one day. We find 11 of the 20 large tenants can be classified into the persistent type. Figure 9 shows the results of 3 example large tenants with the persistent usage pattern across 4 months. The results show that the long-term patterns of the 3 tenants can be further classified into 3 subtypes, i.e., the stable (5 tenants), ladder (3 tenants), and oblique-line (3 tenants) sub-types.

The resource usage of the stable sub-type has little change and remains stable for a long period. Therefore, its resource usage can be easily predicted by calculating the average daily resource usage in the previous period (e.g., one week). The ladder-type has many small ladder switches of resource usage in a small period of time (e.g., between adjacent two days). We can reserve resources for it by using the maximum value in the previous period (e.g., one week). The long-term trend of the oblique-line sub-type can be seen as a continuous upward or downward oblique line with a stable slope. For this sub-type, we can use the linear regression method [41] to predict the resource usage of the next day based on the resource usage of the previous week.

As the resource usage of persistent tenants shows fewer changes in the long term, the usages can be predicted using average and maximum values in the previous period, and linear regression analysis methods.

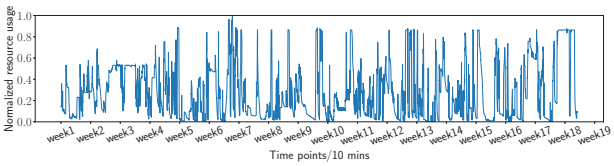


Figure 11: The resource usage of a tenant with the irregular pattern across four months. The resource usage is normalized to the maximum value.

5.1.3 Bursty usage pattern. For the large tenants in our cloud, besides the regular diurnal resource usage pattern, unpredictable extremely high resource usage may happen. We find 2 of the 20 large tenants have bursty cases. We explore the different bursty cases for them as shown in Figure 10.

The bursty cases all have the diurnal resource usage pattern at normal time, but use about 3-7 times of normal resource usage at bursty time. Moreover, we can also observe that the 3 bursty cases have different bursty duration, i.e., tens of minutes, a few hours, and a few days. To sum up, this usage pattern has the unpredictability in bursty happening time, bursty resource amount, and bursty duration.

Since the unpredictability of bursty situations, it is hard to predict the resource usage and reserve enough resources in advance. Therefore, the satisfaction of the bursty resource usage has to rely on online scheduling and compensation. For instance, for the bursty load in a region, we can improve the reserved resources when monitoring it at runtime, and apply other regions' resources if local resources are insufficient. Moreover, since the bursty duration is unpredictable, we need to monitor the duration and reduce the reserved resources after the bursty duration, so as to reduce the waste of resource reservation. At last, since the tenants with bursty cases still show diurnal patterns at the normal time, we can use methods in Section 5.1.1 to predict the normal usage.

The bursty cases of large tenants are hard to predict, and we mainly rely on online scheduling and compensation to satisfy the bursty resource usage demand.

5.1.4 Irregular usage pattern. Some large tenants (3 of the 20 large tenants) show irregular resource usage in our trace data. Figure 11 shows an example large tenant of the irregular type across 4 months. We can find it does not have a determined pattern no matter short-term or long-term. The irregular resource usage is hard to predict and we cannot make precise resource reservation in advance.

Therefore, we make some simple predictions, and mainly rely on the online scheduling and compensation to handle it.

5.1.5 Main takeaways. Based on the observations of the temporal patterns, we can classify them into 4 different types, i.e., diurnal, persistent, bursty, and irregular types.

For different types of resource usage patterns, we need to use the corresponding methods to predict, and further make resource reservations based on the prediction results in advance. Moreover, since the diurnal tenants would have different peak time, it's potential to reduce the resource reservation through peak shaving between them. For the bursty type and irregular type, since the unpredictability of them, the online scheduling and compensation within or across the regions are of significance to satisfy their requirement.

5.2 Spatial Usage Patterns

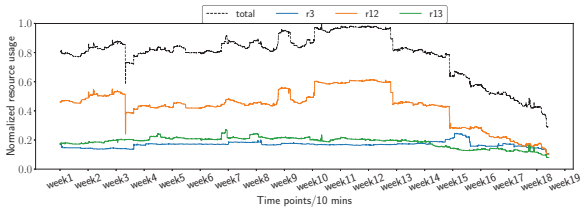
Different regions/VM types have different resource costs, and large tenants have different *acceptable spatial ranges* (Section 4.2 and 4.3). These spatial characteristics motivate us to explore the potential differences in resource usage predicting and reservation on spatial. Therefore, based on the temporal characteristics of large tenants' resource usage, we further break down the resource usage into different regions and VM types, to explore spatial patterns of them.

5.2.1 Using a single major region/VM type. From the statistics of the 20 tenants, we find 10/20 and 5/20 of the large tenants mainly use one region and VM type for their VMs, respectively, while the resource usage on other regions and VM types is negligible.

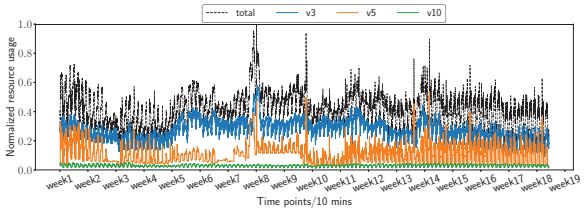
For this kind of tenant, we can focus on the resource usage prediction of their mainly used one region/VM type. Similar prediction methods of the temporal pattern can be used to predict the resource usage of this region/VM type. The other regions/VM types often show little and stable resource usage, so we can reserve resources on them with a fixed value (e.g., an average of previous data) for the large tenants.

The spatial (region/VM type) patterns of these large tenants show few differences from their temporal patterns.

5.2.2 Using multiple regions/VM types with stable usage division. Some large tenants use multiple regions/VM types with stable resource usage division, i.e., the proportion of resource usage between different regions/VM types maintains stable. Figure 12(a) shows an example tenant who uses 3 regions. We can find that *r12* has a similar trend with the total resource usage, and the other two regions' resource usage is nearly stable and equal to each other. Moreover, we can find that the proportion of resource usage in the 3 regions is fairly stable. Figure 12(b) shows an example tenant who uses 3 VM types. We can find *v3* and *v5* has the similar resource usage trend with the total resource usage, and *v10* maintains stable fluctuation. Moreover, similar observations to the regions can be achieved that the resource usage proportion of the 3 VM types approximately remains stable. From statistics of the 20 large tenants, 2/20 and 6/20 large tenants use multiple regions and VM types with stable resource usage division.



(a) The tenant that uses multiple regions with stable mode.



(b) The tenant that uses multiple VM types with stable mode.

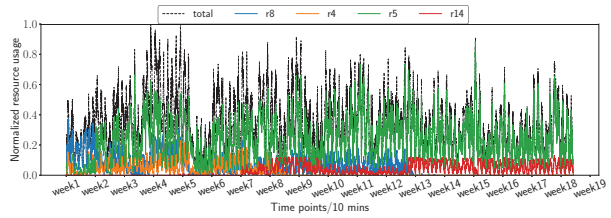
Figure 12: The resource usage of two tenants that use multiple regions/VM types with stable mode across 4 months. All the resource usage is normalized to the maximum value for each tenant.

For this kind of tenant, as the resource usage division of different regions/VM types remains stable with each other, we can easily achieve the resource usage of each region/VM type by using tenants' temporal prediction results to multiply the proportion of each region/VM type.

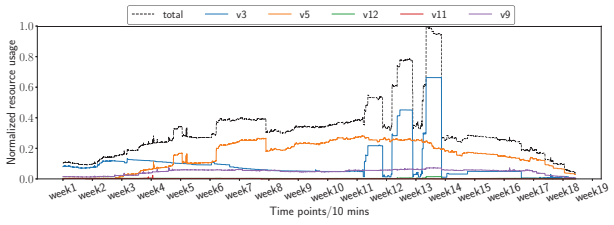
To sum up, although this kind of tenant uses multiple regions/VM types, their spatial (region/VM type) patterns still do not have much difference from the temporal patterns.

5.2.3 Using multiple regions/VM types with dynamic division. Most of the large tenants use multiple regions/VM types with changing resource usage division, i.e., the proportion of resource usage between different regions/VM types is changing frequently. Figure 13(a) shows an example tenant who uses 4 regions, and the resource usage proportion of them changes frequently. As the resource usage breakdown of each week shown in Figure 14(a), the resource proportion is $r8 > r5 \approx r4 > r14 \approx 0$ in week1, $r5 > r8 > r4 > r14 \approx 0$ during week2 to week6, $r5 > r8 > r4 > r14$ in week7, $r5 > r8 > r14 > r4 \approx 0$ during week8 to week12, and $r5 > r14 > r8 \approx r4 \approx 0$ from week13 . Moreover, the resource usage in different regions shows a variety of patterns, which are not necessarily the same as the overall temporal pattern. The results of Figure 13(b) and Figure 14(b) also shows the similar observations on the VM type spatial pattern. From our statistics of the 20 large tenants, most of the tenants belong to this type of spatial pattern, with 8/20 and 9/20 for the region and VM type spatial pattern, respectively.

This kind of tenant has many differences in their spatial patterns with the minimum granularity (region/VM type)

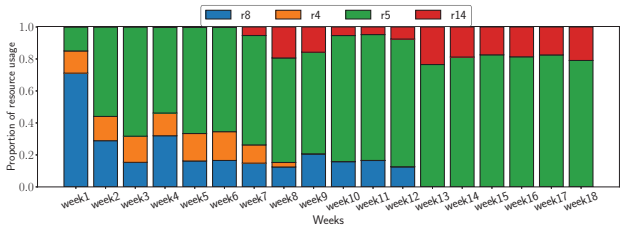


(a) The tenant that uses multiple regions with changing mode.

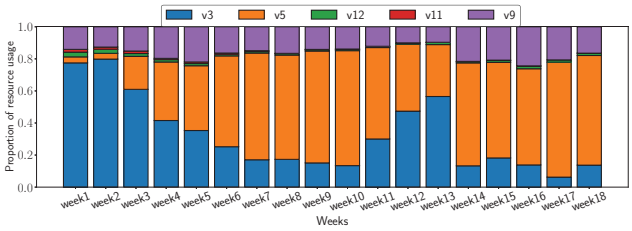


(b) The tenant that uses multiple VM types with changing mode.

Figure 13: The resource usage of two tenants that use multiple regions/VM types with changing mode across 4 months. A tenant's resource usage is normalized to its maximum value.



(a) The resource usage breakdown of each week for Figure 13(a).



(b) The resource usage breakdown of each week for Figure 13(b).

Figure 14: The resource usage breakdown of the two tenants in Figure 13.

from their overall temporal patterns. The reasons include: (1) their resource usage proportion among different regions/VM types changes frequently, and (2) their resource usage of different regions/VM types shows a variety of patterns. Moreover, since different large tenants have their own *acceptable spatial ranges* which are more complex than the minimum spatial granularity (region/VM type), their spatial patterns can be more different from their temporal patterns.

We may divide the spatial granularity according to the *acceptable spatial ranges* of different tenants, and conduct resource usage prediction and resource reservation on the corresponding spatial granularity, so as to better reduce the cost of resource reservation.

5.2.4 Main takeaways. Based on the analysis of resource usage on different regions/VM types, we find most of the tenants use multiple regions/VM types with dynamic resource usage divisions, which have two spatial characteristics.

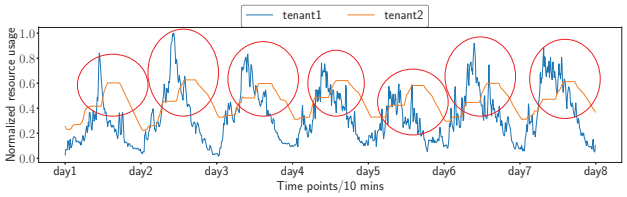
First, the resource usage of different regions/VM types shows a variety of patterns that are different from their temporal patterns of overall resource usage. Second, the resource usage division among different regions/VM types changes frequently over time. Considering the more complex *acceptable spatial ranges* of different large tenants (stated in Section 4.3), the spatial patterns can be more different from their temporal patterns. These spatial characteristics do not hold for centralized clouds, as the resource usage patterns of baremetals in a datacenter are often identical for a tenant.

The spatial characteristics suggest us to conduct the resource usage prediction and resource reservation on tenants' corresponding *acceptable spatial ranges*, to achieve better prediction results and more efficient resource reservation.

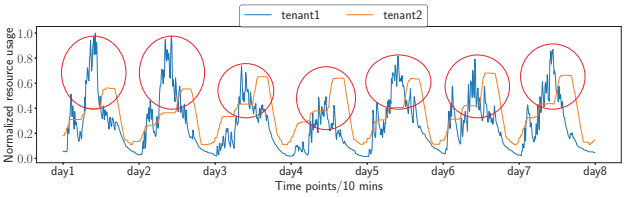
5.3 The Potentials of Improving the Resource Reservation Efficiency

Based on our previous analysis of large tenants' temporal patterns in Section 5.1, we have observed that some tenants present the diurnal patterns of resource usage. Moreover, different large tenants may have different peak time in one day, which brings the potentials to reduce the reserved resources. Therefore, we explore both the temporal and spatial potentials of peak shaving among tenants in this section.

5.3.1 Temporal potential. We first explore the temporal potential between different large tenants. Figure 15(a) shows two example tenants' resource usage on the same region and same VM type during a week. As the red ellipses marked in this figure, we can find that the resource usage of the two tenants shows different peak time. The *tenant1* always achieves the peak load from forenoon to noon, while *tenant2* always experiences peak load during the night. It's an obvious complementary resource usage between these two tenants. For this example, we make resource reservations for both the two tenants together each day, i.e., peak shaving with each other, and then calculate its ratio relative to the method which reserves resources for each tenant separately. The results in Figure 16(a) show that the resource usage of each day in this week can be reduced, with an average value

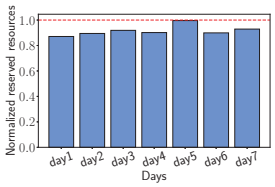


(a) The example of the temporal potential. The two tenants are on the same region *r4* with the same VM type *v5*.

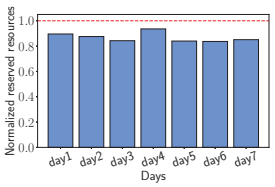


(b) The example of the spatial potential. One tenant is on the VM type *v5* of region *r14*, while the other is on the VM type *v1* of region *r10*.

Figure 15: Examples to show the temporal and spatial (region/VM type) potential of peak shaving. The red ellipses mark the complementary resource usage.



(a) The normalized reserved resources corresponding to the example of Figure 15(a).



(b) The normalized reserved resources corresponding to the example of Figure 15(b).

Figure 16: The normalized reserved resources of peak shaving relative to separately reserve for each tenant.

of 8.4%. From this example, we can see the temporal potential for peak shaving among different tenants, and the resource usage among other tenants also shows similar patterns.

5.3.2 Spatial potential. We further explore the peak shaving potential of resource usage on spatial in different regions and VM types. Figure 15(b) shows an example of resource usage of two tenants in two regions using two VM types. In detail, *tenant1* is on region *r14* with the VM type *v5* while *tenant2* is on region *r10* with the VM type *v1*. As the red ellipses marked in this figure, the resource usage curve of the two large tenants has different peak time similar to the temporal potential. The *tenant1* always achieves the peak load at the forenoon, while *tenant2* always achieves the peak load at night. So if we schedule the requests of the two tenants into the same region using the same VM type, the resource usage can be further reduced since the peak shaving. We take the same method of Figure 16(a) to calculate the reserved resource ratio, Figure 16(b) shows that the resource usage on each day can be reduced more, and with a better average value of 13.2%. The better-reserved resource reduction results prove that the potential of peak shaving is enhanced when we take the spatial factor into consideration.

5.4 Insights from Usage Pattern Analysis

From the comprehensive analysis of temporal and spatial usage patterns of large tenants in our production-level geo-distributed cloud, we now summarize our 3 key insights.

First, there are different types of temporal patterns of large tenants, and we need to use the appropriate prediction method for the corresponding temporal pattern. Moreover, since some temporal patterns are unpredictable (e.g., bursty usage pattern), it is necessary to schedule and compensate their VM requests at runtime.

Second, most of the tenants have obvious spatial patterns with: (1) there is a variety of resource usage patterns in different regions and VM types, and (2) the resource usage divisions between regions and VM types change frequently over time. These spatial patterns suggest us to conduct resource usage prediction and resource reservation on large tenants' corresponding *acceptable spatial ranges*, to achieve better prediction results and resource reservation cost reduction.

Third, we observe that the diurnal large tenants have both the temporal and spatial potential of reducing the reserved resources through peak shaving between them. Combing the analysis in Section 4.2, these observations suggest us to orchestrate resources on low-cost regions/VM types while considering the peak shaving potentials among tenants.

Combing all the insights, we are motivated to propose a resource reservation and VM scheduling scheme. It consists the resource usage predicting, resource orchestrating, and online scheduling and compensation, to reduce the overall deployment cost while satisfying large tenants' VM requests.

6 THE ROS METHODOLOGY

Based on above analysis, we propose a resource orchestrating and scheduling scheme named **ROS**. In this section, we first present the overview of ROS, followed by the design details of each part in ROS. Then, we evaluate ROS's effectiveness in reducing the deployment cost and resource reservation. Lastly, we discuss the lessons learned from the traces.

6.1 Overview

Figure 17 shows the design overview of ROS. ROS comprises a *load pattern predictor*, a *cross-region resource orchestrator*, and a *bursty-aware scheduler*. The predictor identifies tenants' behaviors and estimates the resource usage on each tenant's *acceptable spatial ranges* (Section 6.2). Based on the estimated load pattern, the orchestrator allocates resources for the day

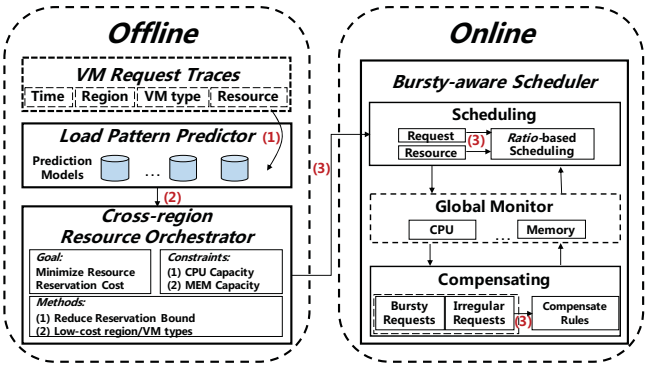


Figure 17: The design overview of ROS.

to minimize overall deployment costs (Section 6.3). Orchestration equivalents to distributing tenants' resource usage to each VM type of each region, and deciding resource reservation bounds correspondingly. At runtime, the scheduler responds to tenants' requests according to the orchestration, and handles the unplanned requests like sudden requests due to the load burst (Section 6.4).

ROS tends to orchestrate the resources for the tenants daily as the tenants often have diurnal load patterns. The design logic of ROS is also applicable for other time period. During a day, ROS works in the following steps.

1) In the beginning of the day, the predictor identifies each tenant's resource usage pattern, and uses the corresponding prediction methods to predict the resource usage curve for the day on each tenant's acceptable spatial ranges. The challenging part here is that the tenants have various resource usage patterns thus require different prediction models.

2) Once receiving the prediction results, ROS orchestrates the predicted resource usage onto different VM types in different regions, and allocates resource reservation bounds accordingly. The challenging part is that ROS has to consider not only the amount of resources in each region but also the cost differences of different regions and different VM types.

3) At runtime, the scheduler allocates the resources and monitors the resource demand of different VM types on different regions. If the resource demand exceeds the reservation bound, the scheduler compensates the insufficient resource reservation. Moreover, the scheduler reduces the reservation bounds when the resource demand drops.

We implement and deploy ROS in our production cloud. The achieved gain is similar to the results in Section 6.5.

6.2 Load Pattern Prediction

Based on our analysis in Section 5.2, the resource usage patterns of large tenants can be divided into 4 types: *diurnal pattern*, *persistent pattern*, *bursty pattern*, and *irregular pattern*. For the high accuracy, ROS uses different models to predict the load variation in different patterns. Specifically, ROS predicts the resource usage (CPU and memory) of each tenant in each *acceptable spatial range*.

To predict the load pattern of a tenant, we first classify the load pattern using K-means [27] and mass-count disparity [1] algorithms. The predictor then selects the corresponding prediction models, conduct the prediction based on the trace of previous one week, and obtains the resource usage time curve of the tenant on the next day. The prediction models are trained based on the historical resource requests of these large tenants and are updated incrementally to capture the new pattern variations. For a new tenant in our cloud, we reserve enough VMs as specified by the tenant in the first month. During this period, we collect the resource usage data every 10 minutes and train the parameters in the models.

For the diurnal load pattern, a three-layer LSTM-based model is used to capture the load variation with time. In detail the model contains two LSTM units [48] and one full-connected (FC) layer at the end. We provide a unique LSTM model for each tenant on each of its *acceptable spatial range*, and the input of the LSTM is the resource usage time curve of previous one week, and the output is the resource usage time curve of the next day.

For the persistent load pattern, the average value calculation, maximum value calculation, and the linear regression (LR) analysis methods are adopted for the stable, ladder, and oblique-line sub-types, respectively. We also provide a unique prediction model for each tenant on each of its *acceptable spatial range* for this load pattern. Expect the prediction model, the input and output of LR are the same as LSTM. The input of average/maximum value calculation methods is same to LSTM, and we use the average/maximum value of the input as the resource usage amount of each time point in the next day to generate the resource usage time curve.

The resource requirements of the tenants with bursty and irregular load patterns are hard to predict. Therefore, for these load patterns, we use the LSTM to make a basic resource usage prediction, and rely on the online scheduling to quickly satisfy the unplanned resource requests.

For the predictable large tenants with diurnal and persistent load patterns, the R-squared value of the real load prediction is 0.865. Predicting the load pattern of a tenant completes in 300ms.

6.3 Cross-region Resource Orchestration

We design the orchestrator based on two observations. 1) Within the *acceptable spatial ranges* of a specific large tenant, some regions or VM types have lower resource costs (Section 4.2). We can directly reduce the deployment cost if ROS makes resource reservation on them for the tenant. 2) The resource usage among diurnal tenants may be complementary. As analyzed in Section 5.3, there are both temporal

and spatial potentials for reducing the resource reservation through peak shaving among large tenants.

For VM reservation, the SLA is satisfied if a tenant can obtain the VMs with the required performance and network latency. The SLA is considered through the *acceptable spatial ranges* in our work. Since a tenant often has determined acceptable spatial ranges, reserving resources out of these ranges may result in the SLA violation. For instance, orchestrating a live video service's resource in a region far away from the end users may result in high network latency. Some computation-intensive workloads can be deployed on multiple CPU-enhanced VM types, but will harm the performance when deploying on the memory-enhanced VM types.

Therefore, the orchestrator determines a tenant's resource reservation with each resource type in each of its acceptable spatial ranges. The resource type means a specific VM type in a region. We model the resource orchestrating as a single objective optimization problem in Equation 1. In the optimization problem, the optimal objective function is defined as finding the minimum total deployment cost. The deployment cost is the sum of the maximum resource usage of each resource type during the day multiplied by its cost coefficient. To be more specific, the inputs of the orchestrator are the predicted resource usage of large tenants achieving from the predictor, and the outputs are the orchestrating schemes and the reservation bounds for each resource type. We provide a unique model to orchestrate large tenants' required resources on each of the acceptable spatial range.

Suppose there are m tenants in an acceptable spatial range, and their required resources can be orchestrated onto n resource types. We use a matrix, $Ratio$, to represent the orchestrating scheme. In the matrix, $ratio_{ij}$ indicates the resource usage percentage of the tenant i on the j-th resource type. In Equation 1, $f_i(t)$ is the weighted average resource usage time curve of tenant i's CPU and memory resource usage provided by the predictor as $f_i(t) = \alpha \times CPU_i(t) + \beta \times MEM_i(t)$. α and β is the relative cost of per unit CPU and memory, respectively. Moreover, C_j represents the cost coefficient of the specific resource type j, which is calculated by the region's cost coefficient times the VM type's cost coefficient.

$$\text{Minimize:} \quad C_{total} = \Sigma_{j=1}^{n} C_j \times max(\Sigma_{i=1}^{m}(f_i(t) \times ratio_{ij})|_{t=0}^{24h})$$
$$\text{s.t.} \begin{cases} \forall i, \Sigma_{j=1}^{n} ratio_{ij} = 1 & i = 1 \cdots m \\ \forall j, max(\Sigma_{i=1}^{m} CPU_{ij}(t)|_{t=0}^{24h}) < Cap_{CPU}^{j} & j = 1 \cdots n \\ \forall j, max(\Sigma_{i=1}^{m} MEM_{ij}(t)|_{t=0}^{24h}) < Cap_{MEM}^{j} & j = 1 \cdots n \end{cases} \tag{1}$$

Note that, the peak shaving is embodied in calculating the maximum resource usage of each resource type. While other orchestrators handle each tenant independently (e.g., Narayanan et al. [42]), Equation 1 optimizes the reservations of multiple tenants together, enabling spatial peak shaving across regions. The optimization problem has 3 constraints.

Table 2: The variables used in Section 6.3

Varible	Varible Description
C_j	The cost coefficient of the *resource type j*
$ratio_{ij}$	The resource percentage of tenant i on *resource type j*
$f_i(t)$	The weighted average resource usage curve of tenant i
$CPU_{ij}(t)$	Tenant i's CPU usage time curve on *resource type j*
$MEM_{ij}(t)$	Tenant i's memory usage time curve on *resource type j*
Cap_{CPU}^{j}	The CPU capacity of *resource type j*
Cap_{MEM}^{j}	The memory capacity of *resource type j*

First, for each large tenant, the sum of the resource ratio needs to be equal to 1, i.e., the required resource usage for this tenant is satisfied. Second, the CPU and memory usage for each *resource type* cannot exceed its capacity. By solving the optimization problem, the optimal orchestration matrix $Ratio$ is obtained. Table 2 summarizes the variables in Equation 1.

ROS orchestrates the reserved resources in 2.5 minutes with about 118,000 VMs on a cloud with 17 regions and 7 VM types. The time drops exponentially if we group tenants/regions/VM types.

6.4 Bursty-aware Resource Scheduling

Even if the offline prediction and orchestration may be inaccurate occasionally for the tenants that have bursty or irregular loads, the bursty-aware scheduling is able to modify the reservation bounds for the inaccurate resource reservation.

When a VM request is received, ROS schedules the request to one of the resource type according to the $Ratio$ matrix and current resource distribution of this tenant. If the scheduler finds that the resource usage in one resource type exceeds the current resource reservation bound, the compensation mechanism is activated to adjust the reservation bound.

The adjustment of resource reservations is constrained by three rules. 1) If the current resource type still has idle resources, the scheduler increases the resource reservation bound until it achieves the current required resource usage. 2) If there are no idle resource in the current resource type, the scheduler selects the lowest-cost resource type with enough idle resources within the acceptable spatial range, and raises its resource reservation bound to satisfy the resource demand. 3) If the resource usage of a resource type is lower than the reservation value during the specified time t, the scheduler gradually reduces its reservation. The scheduler reduces the reservations in the reverse order of raising the bounds.

Since the major increase of reserved resources is caused by the bursty load, we define t according to the minimum duration (tens of minutes) of the bursty load (Section 5.1.3). Therefore, we define t as 30 minutes to be aware of the end of bursty loads quickly. Some previous researches [38, 46] have shown that the load can be regarded as being dropped when load level is below a predetermined value for 30 minutes.

The functionality of our bursty-aware resource scheduling is similar to autoscaling. However, comparing with current autoscaling policies (e.g., K8S's [36]), ROS introduces the acceptable spatial ranges (regions and VM types) for different tenants, which enables auto-scaling with the lowest cost.

6.5 Evaluation of ROS

We evaluate ROS using the open-sourced 4 months trace dataset. We use the first month of traces to train the predictor, and use the rest 3 months of traces to evaluate ROS.

As mentioned in Section 3.1, the regions are divided into three region sets (denoted by *Region Set 1*, *Region Set 2*, *Region Set 3*) based on the geographic positions. The three region sets have 7, 2, 8 regions, respectively. After negotiating with the large tenants, we extract 7 typical VM types ($v1 \cdots v7$). Large tenants' VM requests can be safely switched between them. The set of ($v1 \cdots v7$) is seen as the acceptable spatial range in terms of the VM types.

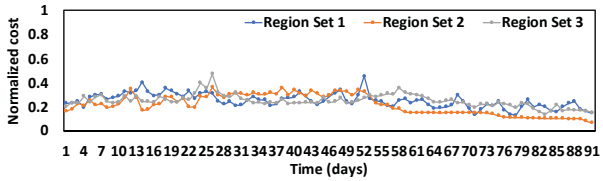
We first compare ROS with the resource reservation strategy currently used by most cloud providers, i.e., reserve resources based on the tenant-specified amount and positions. We also compare ROS with a state-of-the-art baseline [42], which is a geo-distributed capacity planning strategy. We adapt its cost minimization model into our scenario, which is aware of different cost coefficients of datacenters, but orchestrate resources for each tenant independently. We name this baseline [42] as Base in the following experiments.

Moreover, we conduct another experiment to compare ROS with a variation of ROS without the orchestrator (*ROS-wo*), for exploring the effectiveness of the orchestrator. The ROS-wo reserves resources for large tenants equal to their maximum required resources in each day on the corresponding regions and VM types. We use the real request usage results in the trace dataset to replace the prediction results as the input to both ROS and ROS-wo in this experiment, to focus only on the effectiveness of the orchestrator.

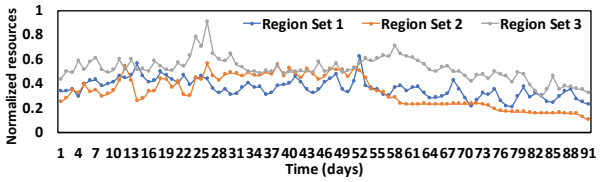
6.5.1 Reducing the deployment cost. Figure 18(a) shows the normalized deployment cost of ROS to the tenant-specific strategy for each day of the three region sets. As observed, ROS reduces the deployment cost on all the days. As the blue bars shown in Figure 19(a), ROS can reduces the deployment cost of the three region sets and total by 74.9%, 78.3%, 75.0%, and 75.4%, compared with the tenant-specific strategy.

Figure 18(b) shows the normalized reserved resources of ROS to the baseline, which we can observe the reserved resources are reduced for each day on the three region sets. As the blue bars shown in Figure 19(b), the reserved resources of the three region sets and total are reduced by 63.5%, 66.2%, 47.7%, and 60.1%, compared with the tenant-specific strategy.

Three reasons result in the deployment cost reduction of ROS compared with tenant-specified reservation. Firstly,

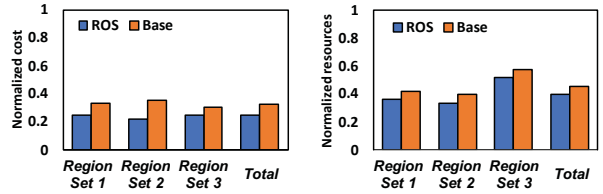


(a) Deployment cost.



(b) Reserved resources.

Figure 18: The deployment cost and reserved resources of ROS normalized to the tenant-specific strategy of three regions sets in each day.



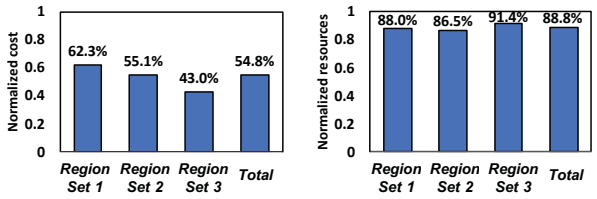
(a) Deployment cost. (b) Reserved resources.

Figure 19: The overall deployment cost and reserved resources of ROS and Base normalized to tenant-specified strategy.

ROS reserves resources based on the predicted tenants' resource usage, rather than the fixed amount of reserving resources specified by the tenants. Secondly, ROS orchestrates the tenants' reserved resources with different patterns (e.g., staggering peak with each other) into the same region or VM type, which can further reduce the reservation bound. Thirdly, ROS tends to orchestrate tenants' reserved resources to low-cost regions or VM types.

Moreover, as shown in Figure 19(a) and Figure 19(b), ROS can also reduce the deployment cost and reserved resouces compared with Base [42]. Statistically, ROS can reduce the total deployment cost and reserved resources by 24.7% and 12.2%, respectively. The baseline performs poor because it orchestrates the resources for each tenant independently. This leads to the lack of temporal and spatial potentials to reduce reserved resources by peak shaving among tenants.

6.5.2 Effectiveness of the resource orchestrator. Figure 20(a) shows the overall deployment cost of ROS normalized to ROS-wo. Under the effect of orchestrator, we can find the



(a) Deployment cost (b) Reserved resources

Figure 20: The normalized overall deployment and reserved resource of ROS relative to ROS-wo.

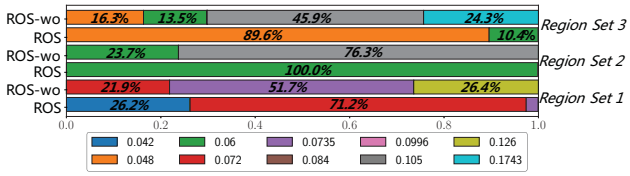


Figure 21: The resource distribution on different VM types of different regions in the 3 region sets.

deployment cost of the three region sets and total are reduced by 37.7%, 44.9%, 57.0%, and 45.2% on average, respectively.

Figure 20(b) shows the overall reservation resource of ROS normalized to ROS-wo. We can observe that the reservation resource of the three region sets and total are reduced by 12.0%, 13.5%, 8.6%, and 11.2% on average, respectively. Our orchestrator allocates large tenants' required resources with staggering peak into the same region or VM types, which can further reduce the resource reservation.

Figure 21 shows the resource distribution percentage on different resource types in the three region sets. One resource type represents one VM type of one region. The labels represent all the resource cost coefficients in the three region sets, which is calculate by the product of region cost coefficient and VM type cost coefficient. Since some resource types have the same cost coefficient, we find out the different values in all the three region sets, and calculate the resource distribution percentage on different cost coefficients for each region set. As observed, ROS distributes most resources on the resource types with low cost coefficients, while ROS-wo distributes part or lot of resources on the resource types with high cost coefficients. The orchestrator tends to reserve resources to low-cost regions or VM types, which can directly reduce the overall resource reservation cost.

6.6 Lessons Learned

By analyzing the resource request traces in our production geo-distributed cloud, we have learned several lessons.

Lesson 1: The tenants, even the top tenants, do not understand their resource requirements in detail. They often tell us to reserve enough VMs with their frequently used types on a region close to the end users, and do not try other VM types or other regions with similar performance and lower price. Suggesting them multiple applicable VM types and multiple regions may help the tenants to reduce the cost, and help us to improve the utilization of the entire geo-distributed cloud.

Lesson 2: It is better to consider the resource reservation from the cloud provider side. A cheapest resource reservation policy for a tenant may not be optimal any more considering the reservation of other tenants. In this case, in order to optimize the entire resource and cost efficiency, it is better to perform the reservation by the cloud provider as a tenant does not know the potential resource usage of other tenants.

Lesson 3: It is possible to build datacenters at where with low costs and improve the utilization through adaptive VM orchestration. Our in-production usage of ROS shows that there are no complains on this new resource reservation policy. We can safely build some datacenters at where with low costs and offload some VMs from the "hot" expensive datacenters to the cheaper datacenters. The running efficiency of the entire cloud can be greatly improved in this way.

Lesson 4: There are opportunities to further optimize the resource utilization by carefully selecting the provided VM types. From our statistics, we find that the VMs with different types have various reservation rates. By tuning the types according to the workloads of the main tenants, the resource utilization and efficiency could be further improved.

7 CONCLUSION

This paper makes a thorough study of the production VM request traces in our geo-distributed cloud. The characterization shows that the resource usage of large tenants has various temporal and spatial patterns, and has the potential of peak shaving between different tenants to further reduce the resource reservation cost. Based on the findings, we propose a resource reservation and VM request scheduling scheme named ROS to minimize the resource reservation cost while satisfying the VM allocation requests. ROS comprises a load pattern predictor, a cross-region orchestrator, and a bursty-aware scheduler. The predictor predicts the resource usage of large tenants. The orchestrator orchestrates the predicted resources of large tenants to different regions/VM types. The scheduler schedules VM requests and compensates the special requests at runtime. Via simulations conducted on our VM traces, we show ROS can reduce the overall deployment cost by 75.4% and the reserved resources by 60.1%.

8 ACKNOWLEDGEMENTS

This work is partially sponsored by the National Natural Science Foundation of China (62022057, 61832006), and Shanghai international science and technology collaboration project (21510713600).

REFERENCES

[1] Omar Arif Abdul-Rahman and Kento Aida. 2014. Towards understanding the usage behavior of Google cloud users: the mice and elephants phenomenon. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*. IEEE, 272–277.

[2] AWS. 2021. *Determine appropriate EC2 instance type for your workload.* Retrieved June 7, 2022 from https://aws.amazon.com/premiumsupport/knowledge-center/ec2-instance-choose-type-for-workload/

[3] AWS. 2022. *Amazon EC2 Instance Types.* Retrieved June 7, 2022 from https://aws.amazon.com/ec2/instance-types/

[4] AWS. 2022. *Amazon EC2 On-Demand Pricing.* Retrieved June 7, 2022 from https://aws.amazon.com/ec2/pricing/on-demand/?nc1=h_ls

[5] AWS. 2022. *On-Demand Capacity Reservations.* Retrieved June 7, 2022 from https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/ec2-capacity-reservations.html

[6] AWS. 2022. *Regions and Availability Zones.* Retrieved June 7, 2022 from https://aws.amazon.com/about-aws/global-infrastructure/regions_az/

[7] Microsoft Azure. 2022. *On-demand Capacity Reservation.* Retrieved June 7, 2022 from https://docs.microsoft.com/en-us/azure/virtual-machines/capacity-reservation-overview

[8] Eric Boutin, Jaliya Ekanayake, Wei Lin, Bing Shi, Jingren Zhou, Zhengping Qian, Ming Wu, and Lidong Zhou. 2014. Apollo: Scalable and Coordinated Scheduling for Cloud-Scale Computing. In *11th USENIX Symposium on Operating Systems Design and Implementation*. 285–300.

[9] Shuang Chen, Christina Delimitrou, and José F Martínez. 2019. Parties: Qos-aware resource partitioning for multiple interactive services. In *Proceedings of the 24th International Conference on Architectural Support for Programming Languages and Operating Systems*. 107–120.

[10] Yue Cheng, Zheng Chai, and Ali Anwar. 2018. Characterizing co-located datacenter workloads: An alibaba case study. In *Proceedings of the 9th Asia-Pacific Workshop on Systems*. 1–3.

[11] David Chou, Tianyin Xu, Kaushik Veeraraghavan, Andrew Newell, Sonia Margulis, Lin Xiao, Pol Mauri Ruiz, Justin Meza, Kiryong Ha, Shruti Padmanabha, et al. 2019. Taiji: managing global user traffic for large-scale internet services at the edge. In *Proceedings of the 27th ACM Symposium on Operating Systems Principles*. 430–446.

[12] Alibaba Cloud. 2021. *Compute optimized instance families.* Retrieved June 7, 2022 from https://partners-intl.aliyun.com/help/en/elastic-compute-service/latest/compute-optimized-instance-families

[13] Alibaba Cloud. 2021. *Regions and zones.* Retrieved June 7, 2022 from https://www.alibabacloud.com/help/en/basics-for-beginners/latest/regions-and-zones

[14] Alibaba Cloud. 2022. *Case studies of Sina Weibo.* Retrieved June 5, 2022 from https://www.alibabacloud.com/help/en/function-compute/latest/sina-weibo

[15] Alibaba Cloud. 2022. *Price Calculator.* Retrieved June 7, 2022 from https://www.alibabacloud.com/pricing-calculator#/commodity/vm_intl

[16] Alibaba Cloud. 2022. *Reserved intances overview.* Retrieved June 7, 2022 from https://www.alibabacloud.com/help/en/elastic-compute-service/latest/reserved-instances-overview

[17] Google Cloud. 2022. *About machine families.* Retrieved June 7, 2022 from https://cloud.google.com/compute/docs/machine-types

[18] Google Cloud. 2022. *Compute Engine pricing.* Retrieved June 7, 2022 from https://cloud.google.com/compute/all-pricing

[19] Google Cloud. 2022. *Global Locations - Regions & Zones.* Retrieved June 7, 2022 from cloud.google.com/about/locations

[20] Google Cloud. 2022. *Reservations of Compute Engine zonal resources.* Retrieved June 7, 2022 from https://cloud.google.com/compute/docs/instances/reservations-overview

[21] Huawei Cloud. 2022. *ECS Types.* Retrieved June 7, 2022 from https://support.huaweicloud.com/intl/en-us/productdesc-ecs/en-us_topic_0035470096.html

[22] Huawei Cloud. 2022. *HUAWEI CLOUD Regions and Service Endpoints.* Retrieved June 7, 2022 from https://developer.huaweicloud.com/intl/en-us/endpoint

[23] Huawei Cloud. 2022. *Price Calculator.* Retrieved June 7, 2022 from https://www.huaweicloud.com/intl/en-us/pricing/index.html

[24] Huawei Cloud. 2022. *Reserved Instance Overview.* Retrieved June 7, 2022 from https://support.huaweicloud.com/intl/en-us/usermanual-ecs/en-us_topic_0177964530.html

[25] Eli Cortez, Anand Bonde, Alexandre Muzio, Mark Russinovich, Marcus Fontoura, and Ricardo Bianchini. 2017. Resource central: Understanding and predicting workloads for improved resource management in large cloud platforms. In *Proceedings of the 26th Symposium on Operating Systems Principles*. 153–167.

[26] Weihao Cui, Han Zhao, Quan Chen, Hao Wei, Zirui Li, Deze Zeng, Chao Li, and Minyi Guo. 2022. DVABatch: Diversity-aware Multi-Entry Multi-Exit Batching for Efficient Processing of DNN Services on GPUs. In *2022 USENIX Annual Technical Conference*. 183–198.

[27] Sheng Di, Derrick Kondo, and Franck Cappello. 2013. Characterizing cloud applications on a Google data center. In *2013 42nd International Conference on Parallel Processing*. IEEE, 468–473.

[28] Kaihua Fu, Wei Zhang, Quan Chen, Deze Zeng, and Minyi Guo. 2021. Adaptive resource efficient microservice deployment in cloud-edge continuum. *IEEE Transactions on Parallel and Distributed Systems* 33, 8 (2021), 1825–1840.

[29] Ionel Gog, Malte Schwarzkopf, Adam Gleave, Robert NM Watson, and Steven Hand. 2016. Firmament: Fast, centralized cluster scheduling at scale. In *12th USENIX Symposium on Operating Systems Design and Implementation*. 99–115.

[30] Jing Guo, Zihao Chang, Sa Wang, Haiyang Ding, Yihui Feng, Liang Mao, and Yungang Bao. 2019. Who limits the resource efficiency of my datacenter: An analysis of alibaba datacenter traces. In *2019 IEEE/ACM 27th International Symposium on Quality of Service*. IEEE, 1–10.

[31] Ori Hadary, Luke Marshall, Ishai Menache, Abhisek Pan, Esaias E Greeff, David Dion, Star Dorminey, Shailesh Joshi, Yang Chen, Mark Russinovich, et al. 2020. Protean: VM allocation service at scale. In *14th USENIX Symposium on Operating Systems Design and Implementation*. 845–861.

[32] Kim Hazelwood, Sarah Bird, David Brooks, Soumith Chintala, Utku Diril, Dmytro Dzhulgakov, Mohamed Fawzy, Bill Jia, Yangqing Jia, Aditya Kalro, et al. 2018. Applied machine learning at facebook: A datacenter infrastructure perspective. In *2018 IEEE International Symposium on High Performance Computer Architecture*. IEEE, 620–629.

[33] Benjamin Hindman, Andy Konwinski, Matei Zaharia, Ali Ghodsi, Anthony D Joseph, Randy Katz, Scott Shenker, and Ion Stoica. 2011. Mesos: A Platform for Fine-Grained Resource Sharing in the Data Center. In *8th USENIX Symposium on Networked Systems Design and Implementation*.

[34] Kevin Hsieh, Aaron Harlap, Nandita Vijaykumar, Dimitris Konomis, Gregory R Ganger, Phillip B Gibbons, and Onur Mutlu. 2017. Gaia:Geo-Distributed Machine Learning Approaching LAN Speeds. In *14th USENIX Symposium on Networked Systems Design and Implementation*. 629–647.

[35] Yuzhen Huang, Yingjie Shi, Zheng Zhong, Yihui Feng, James Cheng, Jiwei Li, Haochuan Fan, Chao Li, Tao Guan, and Jingren Zhou. 2019. Yugong: Geo-distributed data and job placement at scale. *Proceedings of the VLDB Endowment* 12, 12 (2019), 2155–2169.

[36] Kubernetes. 2022. *Horizontal Pod Autoscaling.* Retrieved September 28, 2022 from https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/

[37] Zijun Li, Jiagan Cheng, Quan Chen, Eryu Guan, Zizheng Bian, Yi Tao, Bin Zha, Qiang Wang, Weidong Han, and Minyi Guo. 2022. RunD: A Lightweight Secure Container Runtime for High-density Deployment and High-concurrency Startup in Serverless Computing. In *2022 USENIX Annual Technical Conference*. 53–68.

[38] Zijun Li, Linsong Guo, Quan Chen, Jiagan Cheng, Chuhao Xu, Deze Zeng, Zhuo Song, Tao Ma, Yong Yang, Chao Li, et al. 2022. Help Rather Than Recycle: Alleviating Cold Startup in Serverless Computing Through Inter-Function Container Sharing. In *2022 USENIX Annual Technical Conference*. 69–84.

[39] Qixiao Liu and Zhibin Yu. 2018. The elasticity and plasticity in semi-containerized co-locating cloud workload: a view from alibaba trace. In *Proceedings of the ACM Symposium on Cloud Computing*. 347–360.

[40] Shutian Luo, Huanle Xu, Chengzhi Lu, Kejiang Ye, Guoyao Xu, Liping Zhang, Yu Ding, Jian He, and Chengzhong Xu. 2021. Characterizing microservice dependency and performance: Alibaba trace analysis. In *Proceedings of the ACM Symposium on Cloud Computing*. 412–426.

[41] Dastan Maulud and Adnan M Abdulazeez. 2020. A review on linear regression comprehensive in machine learning. *Journal of Applied Science and Technology Trends* 1, 4 (2020), 140–147.

[42] Iyswarya Narayanan, Aman Kansal, and Anand Sivasubramaniam. 2017. Right-sizing geo-distributed data centers for availability and latency. In *2017 IEEE 37th International Conference on Distributed Computing Systems*. IEEE, 230–240.

[43] Tirthak Patel and Devesh Tiwari. 2020. Clite: Efficient and qos-aware co-location of multiple latency-critical jobs for warehouse scale computers. In *2020 IEEE International Symposium on High Performance Computer Architecture*. IEEE, 193–206.

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

[45] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. 2013. Omega: flexible, scalable schedulers for large compute clusters. In *Proceedings of the 8th ACM European Conference on Computer Systems*. 351–364.

[46] Mohammad Shahrad, Rodrigo Fonseca, Íñigo Goiri, Gohar Chaudhry, Paul Batum, Jason Cooke, Eduardo Laureano, Colby Tresness, Mark Russinovich, and Ricardo Bianchini. 2020. Serverless in the wild: Characterizing and optimizing the serverless workload at a large cloud provider. In *2020 USENIX Annual Technical Conference*. 205–218.

[47] Jiuchen Shi, Jiawen Wang, Kaihua Fu, Quan Chen, Deze Zeng, and Minyi Guo. 2022. QoS-awareness of Microservices with Excessive Loads via Inter-Datacenter Scheduling. In *2022 IEEE International Parallel and Distributed Processing Symposium*. IEEE, 324–334.

[48] Xingjian SHI, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-kin Wong, and Wang-chun WOO. 2015. Convolutional LSTM Network: A Machine Learning Approach for Precipitation Nowcasting. In *Advances in Neural Information Processing Systems*, Vol. 28.

[49] Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. 2018. A comparison of ARIMA and LSTM in forecasting time series. In *2018 17th IEEE international conference on machine learning and applications*. IEEE, 1394–1401.

[50] Chunqiang Tang, Kenny Yu, Kaushik Veeraraghavan, Jonathan Kaldor, Scott Michelson, Thawan Kooburat, Aravind Anbudurai, Matthew Clark, Kabir Gogia, Long Cheng, et al. 2020. Twine: A unified cluster management system for shared infrastructure. In *14th USENIX Symposium on Operating Systems Design and Implementation*. 787–803.

[51] Huangshi Tian, Yunchuan Zheng, and Wei Wang. 2019. Characterizing and synthesizing task dependencies of data-parallel jobs in alibaba cloud. In *Proceedings of the ACM Symposium on Cloud Computing*. 139–151.

[52] Ye Tian, Min Zhao, and Xinming Zhang. 2017. *Internet Video Data Streaming: Energy-saving and Cost-aware Methods*. Springer.

[53] Muhammad Tirmazi, Adam Barker, Nan Deng, Md E Haque, Zhijing Gene Qin, Steven Hand, Mor Harchol-Balter, and John Wilkes. 2020. Borg: the next generation. In *Proceedings of the fifteenth European conference on computer systems*. 1–14.

[54] Alexey Tumanov, Timothy Zhu, Jun Woo Park, Michael A Kozuch, Mor Harchol-Balter, and Gregory R Ganger. 2016. TetriSched: global rescheduling with adaptive plan-ahead in dynamic heterogeneous clusters. In *Proceedings of the 7th European Conference on Computer Systems*. 1–16.

[55] Luping Wang, Qizhen Weng, Wei Wang, Chen Chen, and Bo Li. 2020. Metis: Learning to schedule long-running applications in shared container clusters at scale. In *International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 1–17.

[56] Jon Welling. 2022. *How to Choose the Best Virtual Machine For Your Workload in Azure*. Retrieved June 7, 2022 from https://www.cbtnuggets.com/blog/certifications/microsoft/how-to-choose-the-best-virtual-machine-for-your-workload-in-azure

[57] Qizhen Weng, Wencong Xiao, Yinghao Yu, Wei Wang, Cheng Wang, Jian He, Yong Li, Liping Zhang, Wei Lin, and Yu Ding. 2022. MLaaS in the Wild: Workload Analysis and Scheduling in Large-Scale Heterogeneous GPU Clusters. In *19th USENIX Symposium on Networked Systems Design and Implementation*. 945–960.

[58] Juncheng Yang, Yao Yue, and KV Rashmi. 2020. A large scale analysis of hundreds of in-memory cache clusters at Twitter. In *14th USENIX Symposium on Operating Systems Design and Implementation*. 191–208.

[59] Wei Zhang, Quan Chen, Kaihua Fu, Ningxin Zheng, Zhiyi Huang, Jingwen Leng, and Minyi Guo. 2022. Astraea: towards QoS-aware and resource-efficient multi-stage GPU services. In *Proceedings of the 27th ACM International Conference on Architectural Support for Programming Languages and Operating Systems*. 570–582.

[60] Yanqi Zhang, Íñigo Goiri, Gohar Irfan Chaudhry, Rodrigo Fonseca, Sameh Elnikety, Christina Delimitrou, and Ricardo Bianchini. 2021. Faster and Cheaper Serverless Computing on Harvested Resources. In *Proceedings of the ACM SIGOPS 28th Symposium on Operating Systems Principles*. 724–739.

[61] Zhuo Zhang, Chao Li, Yangyu Tao, Renyu Yang, Hong Tang, and Jie Xu. 2014. Fuxi: a fault-tolerant resource management and job scheduling system at internet scale. In *Proceedings of the VLDB Endowment*, Vol. 7. VLDB Endowment Inc., 1393–1404.

[62] Han Zhao, Weihao Cui, Quan Chen, Youtao Zhang, Yanchao Lu, Chao Li, Jingwen Leng, and Minyi Guo. 2022. Tacker: Tensor-CUDA Core Kernel Fusion for Improving the GPU Utilization while Ensuring QoS. In *2022 IEEE International Symposium on High-Performance Computer Architecture*. IEEE, 800–813.