Is Sharing Caring? Analyzing the Incentives for Shared Cloud Clusters

Talha Mehboob
tmehboob@umass.edu
University of Massachusetts Amherst
Amherst MA, USA

Noman Bashir
nbashir@umass.edu
University of Massachusetts Amherst
Amherst MA, USA

Michael Zink
mzink@umass.edu
University of Massachusetts Amherst
Amherst MA, USA

David Irwin
deirwin@umass.edu
University of Massachusetts Amherst
Amherst MA, USA

ABSTRACT

Many organizations maintain and operate large shared computing clusters, since they can substantially reduce computing costs by leveraging statistical multiplexing to amortize it across all users. Importantly, such shared clusters are generally not free to use, but have an internal pricing model that funds their operation. Since employees at many large organizations, especially Universities, have some budgetary autonomy over purchase decisions, internal shared clusters are increasingly competing for users with cloud platforms, which may offer lower costs and better performance. As a result, many organizations are shifting their shared clusters to operate on cloud resources. This paper empirically analyzes the user incentives for shared cloud clusters under two different pricing models using an 8-year job trace from a large shared cluster for a large University system.

Our analysis shows that, with either pricing model, a large fraction of users have little financial incentive to participate in a shared cloud cluster compared to directly acquiring resources from a cloud platform. While shared cloud clusters can provide some limited reductions in cost by leveraging reserved instances at a discount, due to bursty workloads, realizing these reductions generally requires imposing long job waiting times, which for many users are likely not worth the cost reduction. In particular, we show that, assuming users defect from the shared cluster if their wait time is greater than 15× their average job runtime, over 80% of the users would defect, which increases the price of the remaining users such that it eliminates any incentive to participate in a shared cluster. Thus, while shared cloud clusters may provide users other benefits, their financial incentives are weak.

CCS CONCEPTS

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

KEYWORDS

Cloud computing; Provisioning policies; Trace analysis

ACM Reference Format:
Talha Mehboob, Noman Bashir, Michael Zink, and David Irwin. 2023. Is Sharing Caring? Analyzing the Incentives for Shared Cloud Clusters. In *Proceedings of the 2023 ACM/SPEC International Conference on Performance Engineering (ICPE '23), April 15–19, 2023, Coimbra, Portugal.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3578244.3583730

1 INTRODUCTION

Public cloud platforms, such as Amazon Web Services (AWS), Google Compute Engine (GCE), and Microsoft Azure, continue to grow rapidly [13] due to their many benefits, including low cost, pay-as-you-go billing, and scalability. These public cloud platforms are expected to continue to grow for the foreseeable future [6], as a substantial amount of computing is still done on private on-premises infrastructure. In general, such "on-prem" [10] computing is significantly more expensive [11] than cloud platforms, since cloud datacenters benefit from economies-of-scale and more statistical multiplexing [20]. While, in some cases, on-prem clusters are necessary due to regulatory or privacy [12] concerns, many have the potential to migrate to cloud platforms.

Given the high cost of deploying and operating on-prem computing clusters, large organizations generally operate them as shared clusters, such that users submit their jobs to a scheduler that either runs them immediately or queues them up for later execution [9]. Thus, by operating shared on-prem clusters, large organizations leverage some of the statistical multiplexing benefits of the cloud. Organizations determine the size of on-prem clusters based on their expected demand and cost constraints: the more resources, the lower the job waiting times but the higher the cost, and vice versa. As we discuss, workloads tend to be highly bursty with many periods of relative idleness interspersed with large bursts of jobs, which can cause long job waiting times for all but the highest-cost, most over-provisioned clusters. Of course, determining the size of an on-prem cluster [27], i.e., number of resources, is a key issue, since their workload intensity is not known *a priori*.

Given the advantages of public cloud platforms, many large organizations, such as Universities, are considering shifting their shared on-prem clusters to the public cloud. Cloud clusters are cheaper to both purchase and administer—with all administrative functions available remotely in software. There are multiple options when transitioning a shared on-prem cluster to the public cloud. The most

ICPE '23, April 15–19, 2023, Coimbra, Portugal

ACM ISBN 979-8-4007-0068-2/23/04…$15.00
https://doi.org/10.1145/3578244.3583730

basic option is "lift-and-shift", where organizations reserve similar resources as they have in their shared on-prem cluster and use the same management software, i.e., scheduler. Cloud platforms enable users to reserve virtual machine (VM) instances in advance for long periods, e.g., 1 or 3 years, at a discount, similar to how organizations purchase physical servers (often with a volume discount) for shared clusters. However, this approach suffers from the same drawback as above: organizations must determine the optimal number of VMs to reserve based on uncertain future workload demand. Purchasing too few resources causes high job waiting times, while purchasing too many resources incurs high costs.

Notably, cloud platforms offer other options beyond the simplistic "lift-and-shift" approach. In particular, users could abandon a shared cluster, and rent cloud VMs themselves on demand. While the per-hour cost of on-demand VMs is generally more (∼35-40%) [2] than that of highly utilized reserved VMs, by "flying solo," a user's jobs would experience no waiting time, as cloud platforms are provisioned to satisfy any on-demand requests. In addition, the per-hour cost of "flying solo" might not be higher if the shared cloud cluster is over-provisioned due to an inaccurate demand forecast, which causes its actual cost to exceed the optimal cost. In addition, there is also a middle ground where shared cloud clusters conservatively reserve some number of cloud resources, but to reduce waiting times during job bursts, use an autoscaler to automatically provision on-demand VMs to run queued jobs. This approach presents a configurable tradeoff between cost and job waiting time: the longer jobs are willing to wait, the lower the overall cost, since the cluster uses fewer high-cost on-demand VMs.

Importantly, even shared clusters have an internal pricing model that funds their operation. Since employees at many large organizations, especially Universities, have some budgetary autonomy over purchase decisions, shared clusters are increasingly directly competing with cloud platforms, which can potentially offer lower costs and better performance. While the cost for users "flying solo" and renting resources directly from the cloud is clear, the cost for users of a shared cloud cluster is based on clusters' internal pricing models, which are often complex and vary widely by institution. In this paper, we examine two pricing models at the different ends of a spectrum. Specifically, we examine i) a *socialist pricing model* that amortizes the per-hour cost of all reserved and on-demand resources, and charges users a single per-hour amortized price and ii) a *capitalist pricing model* that charges users the on-demand price if the scheduler runs their jobs on on-demand VMs and the discounted reserved price if it uses reserved VMs. As we show, both policies introduce incentives for a set of users to defect from the shared cluster, such that their cost for "flying solo" is similar to or less than when using the shared cluster while also yielding better performance. Ultimately, we analyze whether shared cloud clusters are viable given the incentives introduced by both internal pricing models at current cloud prices.

We conduct our analysis of the different cloud provisioning and pricing policies above using a large-scale longitudinal workload trace of a shared cluster from a large University system, which includes several large campuses. The trace covers 8 years of operation and includes 67 million job submissions for a shared cluster hosting a diverse workload of jobs submitted by researchers in the scientific, engineering, and medical fields.

We use trace-driven simulations to evaluate the cost and waiting times for servicing this cluster's workload under the different cloud provisioning and internal pricing policies above, and use our results to analyze the incentives for users to participate in shared cloud clusters or to defect and "fly solo" by renting their own cloud resources to run jobs.

Our hypothesis is that cloud platforms already pass on most of their cost benefits to users in their on-demand price, and thus there is little financial incentive for large organizations to operate shared cloud clusters. As we show, many users will defect from a shared cloud cluster, since their cost to "fly solo" is similar and yields better performance. These defections in-turn raise the per-user cost of a shared cloud cluster, causing yet more users to defect. In evaluating our hypothesis, we make the following contributions.

Large-scale Workload Analysis. We analyze our large-scale workload trace to better understand users' job characteristics, including the number of submissions, runtimes, and burstiness. We highlight characteristics that impact the provisioning and pricing policies for shared cloud clusters.

Provisioning and Pricing Policy Incentives. We analyze the cost and job waiting times for different provisioning and pricing policies for shared cloud clusters, and discuss their impact on users' incentive to participate in a shared cluster or to defect and "fly solo."

Implementation and Evaluation. We implement a trace-driven job scheduling simulator, and evaluate the cost and job waiting time of our cloud provisioning and pricing policies on a large-scale 8-year workload trace. We quantify the cost and job waiting times for users under different provisioning and pricing policy combinations, which ultimately determines their incentive to defect from a shared cloud cluster. In particular, we show that, assuming users defect from the shared cluster if their wait time is greater than 15× their average job runtime, over 80% of the users would defect, which would increase the price of the remaining users such that it eliminates any incentive to participate in the shared cluster. Thus, while shared cloud clusters may provide users other benefits, their financial incentives are weak.

2 BACKGROUND

We provide background on transitioning on-prem shared clusters to cloud platforms, and discuss the various pricing models and job scheduling policies that affect user incentives.

2.1 Transitioning to Shared Cloud Clusters

Given the growing need for computation, large organizations, and especially Universities, are establishing large shared clusters to satisfy their users' demand. Critically, sharing compute resources and leveraging statistical multiplexing reduces users' computation costs [11]. Organizations purchase and operate these shared facilities on-premises, and have complete control over their operation, including the hardware, software, and operational policies, i.e., for prioritizing users. Typically, these clusters operate a job scheduler, such as Slurm [36] or Kubernetes [28], which schedules jobs that users submit based on a pre-defined policy. The size of a shared on-prem cluster is generally fixed, as adding new servers requires manually installing them in the cluster. As a result, jobs submitted to shared clusters can experience long waiting times if the number

Plan	Instance	Price/Hour	Cores	Memory	Discount
On-Demand	C6gd.16xLarge	$ 2.4576	64	128 GB	-
Reserved (3 years)	C6gd.16xLarge	$ 1.062	64	128 GB	57 %
Reserved (1 years)	C6gd.16xLarge	$ 1.548	64	128 GB	37 %

Table 1: *Amazon EC2 Pricing Model for On-demand and Reserved Instances.*

of submitted jobs exceeds the cluster's fixed resources. As we discuss, since jobs are often submitted in bursts, this can occur often even when most of the cluster remains idle most of the time.

Since cloud platforms offer a number of benefits for hosting a shared cluster, many organizations are considering transitioning their shared clusters to the cloud. Cloud platforms still provide organizations full control over their compute infrastructure, but without both i) the large upfront expenses for purchasing servers and provisioning space/cooling for them and ii) the ongoing maintenance costs. Instead, cloud platforms enable renting servers hosted in cloud datacenters, and paying for them incrementally over time. Importantly, cloud platforms offer servers under multiple different pricing models with different levels of discount. In particular, cloud platforms enable users to rent servers "on demand" for a per-hour price, or reserve servers for long periods, e.g., 1 or 3 years, for a discounted price (e.g., ~40-60% less).

The problem with renting on-demand servers for a shared cloud cluster is that it offers no cost benefit to users. Unlike an on-prem cluster, in this case, users can simply use the cloud platform directly to rent resources at the same cost as using the shared cloud cluster. However, shared cloud clusters can potentially provide a benefit to users by reserving servers at a discount. In this case, a shared cloud cluster can leverage statistical multiplexing among many users to offer resources at a lower overall cost than an individual user renting on-demand resources. In this paper, we evaluate whether this cost incentive is high enough to warrant deploying a shared cloud cluster based on user workloads and different pricing models.

2.2 Pricing Models

Cloud pricing models are simple for users, as users simply pay for their own resource usage. For example, in Amazon's Elastic Compute Cloud (EC2), users can either reserve servers of a particular type for 1 or 3 years for a fixed price, or rent them on-demand and pay a per-hour price based on their usage time. To illustrate, based on current prices in Table 1, renting a `C6dg.16xLarge` server, which offers 128 GB of memory and 64 cores, at the on-demand price for a year costs $21,528, at an hourly rate of $2.4576, while reserving it for a year costs around $13,560, at an hourly rate of $1.548. Thus, reserving the server provides a discount of 37% for one year, assuming the server is fully utilized for the entire year. If the server is utilized less than 63% of the time, then renting an on-demand server for only the utilized time period would be cheaper. While cloud platforms offer spot [1, 4] or preemptible VMs [8], which are cheaper than highly utilized fixed VMs, not all jobs can use them.

Unlike in the cloud, internal shared clusters generally have a wider variety of pricing models that distribute the burden of the cluster's cost across the users. While shared clusters may charge users a uniform price for the time their jobs run, they may also use other pricing models. As we discuss, when considering pricing models for shared cloud clusters there are two basic options: the shared cloud cluster can charge users a uniform rate that divides the clusters total cost by the total number of hours jobs run (based on the proportion of reserved and on-demand resources used), or it can charge users a variable rate based on their own usage of reserved and on-demand resources, which incur different costs.

2.3 Scheduling and Waiting Policies

In addition to cost, users also consider their performance, i.e., how long their jobs must wait to be scheduled, when determining whether to use a shared cluster. A job scheduler implements a scheduling policy that determines the order in which jobs run when resources become available. Most schedulers use simple scheduling policies, such as first-come-first-serve with various options, e.g., backfilling, priorities, etc., since optimal policies, such as shortest job first (SJF) [32], require accurate information on job runtimes that is generally not available [26]. Thus, in this paper, we assume a simple FCFS [32] scheduling policy. The interactions between scheduling, waiting, and cloud bursting are outside the scope of this paper. Instead, this paper's focus is on analyzing the incentives introduced by various provisioning policies and pricing models in conjunction with a common baseline scheduling policy.

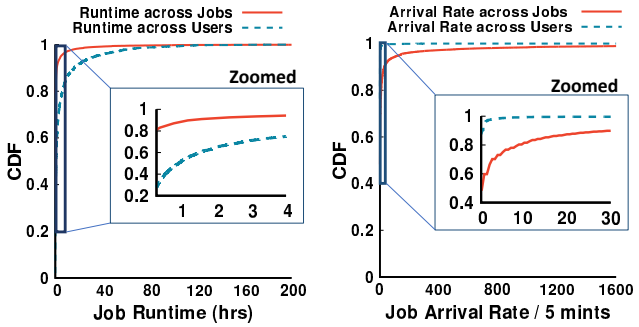
In addition to its scheduling policy, a shared cloud cluster's cost and performance is also a function of its waiting policy [17]. To provide a cost incentive to users, shared cloud clusters should provide a discount over the on-demand by reserving some resources. However, unlike a fixed-size on-prem cluster, a shared cloud cluster can dynamically increase its size by "bursting" [21, 29] into the cloud and renting on-demand servers when its demand is high, and jobs are experiencing excessively long waiting times. In this case, the waiting policy determines how long jobs wait before the cluster dynamically provisions an on-demand server to execute the job. As prior work shows [17], waiting policies offer a tradeoff between cost and job waiting time. The longer jobs wait for reserved resources, the higher the waiting time, but the lower the overall cost, since more jobs run on reserved resources than on-demand resources.

3 JOB WORKLOAD CHARACTERISTICS

The incentives for using a shared cloud cluster depend on its users' workload characteristics. We first analyze workload characteristics from an 8-year job trace from an on-prem shared cluster for a large University system. We use this trace in Section 6 to evaluate the performance of cloud provisioning and pricing policies in Section 4.

3.1 Workload Overview

Our job trace derives from an on-prem shared cluster comprised of ~14,300 cores and includes jobs from a wide range of computational disciplines, including the scientific, engineering, and medical research communities. Thus, the trace is highly representative of the type of large shared research cluster operated by Universities.



(a) Runtime (b) Arrival Rate

Figure 1: ***Cummulative Distribution Function (CDF) of (a) job runtime across jobs and users (on average) and (b) arrival rate per 5 minutes across jobs and users (on average) for our job trace spanning from 2014 to 2021.***

As we show, while the cluster's size is large relative to its average demand, users can experience significant wait times for their jobs due to the "burstiness" of job arrivals. Our trace covers an 8-year period from 2014 to 2021, and includes data on 67 million jobs submitted by 1790 distinct users. Each job entry contains a user ID, the maximum time limit for the job, its actual runtime, number of requested cores, CPU runtime, total memory required, job status, and submission time.

We analyze various aspects of this workload that impact the incentives for using a shared cloud cluster, including the aggregate and per-user job runtime distributions, job burstiness based on job arrival rates over time, and changes in both the number of distinct users and their total usage (in core-hours) over time. Many of our observations are general and have also been noted in prior work on analyzing the workload characteristics for large shared clusters. For example, in aggregate, our dataset is similar to a recently released Google trace, as we show in prior work [18]. However, other publicly-available datasets, e.g., released by Google, Azure, Alibaba, etc., differ in that they do not contain per-user information, and cover relatively short time periods, e.g., one month. As a result, these traces are not useful for much of our analysis, which requires per-user data over a long period of time. That said, given the aggregate similarities, our insights and analysis may be applicable to other similar large, general-purpose clusters.

3.2 Job Runtime Distributions

We analyze workload's job runtime distribution in aggregate and for each individual user. This analysis enables us to compare each user's average job runtime to the mean waiting time for reserved resources under the different provisioning policies in Section 4, which is important, as typically a job's waiting time should not be significantly longer than its runtime. For example, a 10-minute job is less likely to wait 24 hours to run, compared to a 48-hour job.

The CDF of the runtime for all jobs is shown in Figure 1a. As shown, nearly 94% of the jobs have a runtime of 4 hours or less, and almost 75% of the users have an average job runtime of 4 hours or less. This indicates that most of the jobs are relatively short, and that the majority of users have relatively small average job runtimes. Jobs with such short runtimes are likely more sensitive to

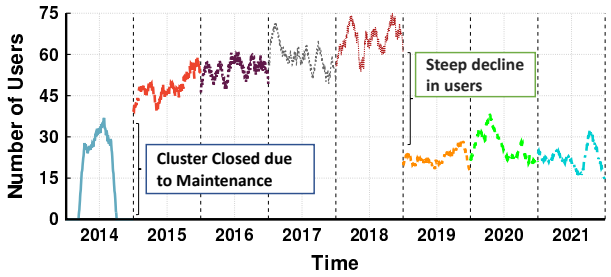


Figure 2: ***Moving average of the number of unique users using the shared cluster per day during 2014 to 2021.***

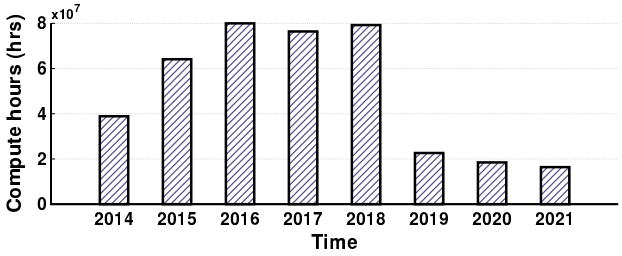


Figure 3: ***Total core-hours across all users per year.***

waiting for resources. Our analysis also shows that there are some users (~25%) with long job runtimes of several hundreds of hours.

Key takeaway. *Most of the jobs in our workload, and for each user, have short runtimes, e.g., well under 4 hours.*

3.3 Burstiness

The on-prem shared cluster that hosted our job trace is primarily used by the researchers from academia, where resource usage increases around well-defined deadlines, e.g., for research papers and proposals. Such burstiness of job arrivals has a significant effect on the cost and performance of a shared cloud cluster. To analyze the burstiness of job arrivals, we determine the number of jobs that are submitted within a 5-minute interval, and plot the CDF of the job arrival rate. Figure 1b shows the results. The solid line in Figure 1b shows that ~90% of the time, only 30 jobs or less arrive in a 5-minute interval. The remaining 10% represent intervals with much higher job arrivals (some with more than 200 jobs per interval), which indicates periods of extreme burstiness.

Key Takeaway. *Many users periodically submit large bursts of jobs, which can cause either high waiting times (in a shared on-prem cluster) or high costs (in a shared cloud cluster).*

3.4 Workload Variability

The benefit of our longitudinal trace is the ability to study how the utilization of a shared cluster varies over many years. Such data is vital to inform long-term decisions on resource provisioning. For example, a highly uniform workload where the number of users and the total utilized core-hours remains constant over time enables administrators to more accurately provision resources to meet demand, as we discuss in Section 6.2. In contrast, a highly variable demand makes it more challenging to accurately provision resources, and introduces a tradeoff between cost and waiting time. That is, provisioning more resources may reduce waiting times,

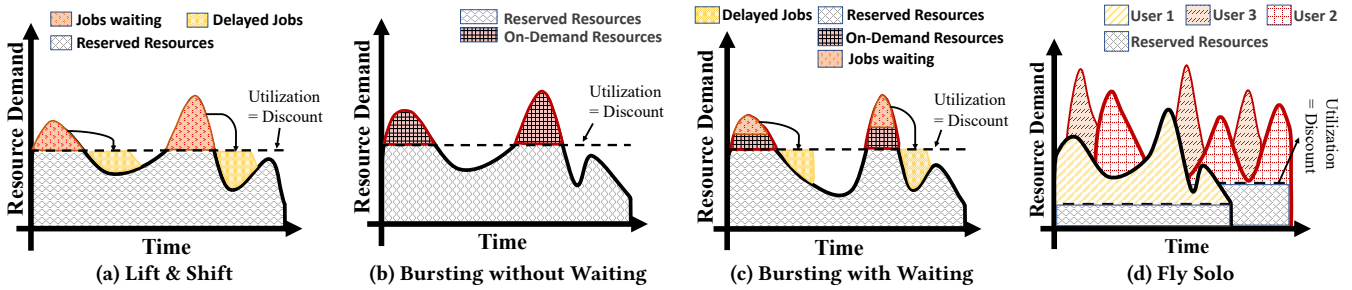


Figure 4: ***(a) All the jobs wait to get serviced on the fixed resources and no jobs execute on the on-demand resources. (b) On-demand resources are acquired by scheduler instantly without waiting for the fixed resources to be available again. (c) Jobs wait for the fixed resources to be available for a certain amount of time before going to on-demand resources. (d) User's provision resources on the public cloud on their own, without being part of the shared cluster.***

but it increases cost. In the extreme, on-prem clusters must provision resources to satisfy their peak workload demand (often at a prohibitively high cost) to ensure no waiting time.

To illustrate cluster variability, Figure 2 plots the average number of users in a day, as a moving average with a window size of 30, which clearly shows an increasing and then decreasing trend in users over the trace period. The cluster started operation on April 1st, 2014 but was put under maintenance from September 9th 2014 to January 2015, and thus executed no jobs during this period. Between January 2015 and December 2018, we observe a steady increase in users, followed by a steep drop at the start of 2019. The drop roughly correlates with an updated pricing model for users. Figure 3, which plots the total core-hours over the year, shows a similar trend with the core-hours dropping substantially at the same time. The steep decline in users above motivates our analysis of the incentives for using a shared cloud cluster. Large shared on-prem clusters are already competing with cloud platforms at Universities, and other organizations, where users have some budgetary autonomy.

Key Takeaway. *Shared clusters may experience large variability, i.e., large increases and decreases, in their usage.*

4 PROVISIONING AND PRICING

There are many possible provisioning and pricing options when moving shared clusters to cloud platforms. Below, we analyze the cost and performance tradeoffs associated with each option, and highlight how they might affect users' incentives differently based on their own individual workload. In particular, the incentive for any individual user to participate in a shared cloud cluster is a function of the characteristics of their own workload and performance requirements, as well as the specific provisioning and pricing policy.

4.1 Resource Provisioning Policies

We define and analyze three resource provisioning policies for shared cloud clusters: lift-and-shift, cloud bursting, and flying solo. These policies represent points along a spectrum defined by the tradeoff between the amount of cloud resources a shared cluster reserves versus acquires on-demand.

4.1.1 Lift-and-Shift. The lift-and-shift policy is akin to moving a traditional static on-prem cluster to a cloud platform by simply reserving a fixed amount of resources in the cloud, rather than purchasing and installing them on-premises. This policy generally requires no changes to cluster schedulers, as it does not take advantage of the cloud's ability to rent resources on-demand. Thus, under bursty workloads, if resources are fully utilized, queued jobs must wait until other jobs complete and fixed resources become available before executing. Figure 4a illustrates the lift-and-shift policy, where all jobs must wait for fixed reserved resources, and the scheduler never acquires resources on-demand to execute jobs.

As in on-prem clusters, the lift-and-shift policy benefits users by acquiring many discounted reserved resources (rather than higher-cost on-demand resources), and leverages statistical multiplexing to increase their utilization. This can result in a lower normalized per-hour price for resources, i.e., the cost of reserved resources divided by the total time they are utilized, than individual users could get either by reserving their own resources or acquiring their own on-demand resources. Since most users' workload intensity is not high enough to make reserving any resources cost-effective, they can potentially leverage the shared cluster to execute their jobs for less than the on-demand price.

Unfortunately, lift-and-shift has at least two problems, which also manifest themselves in static on-prem clusters. Most importantly, periodic job bursts, where demand for resources substantially exceeds the supply, can result in long job waiting times for users. In addition, provisioning the number of fixed reserved resources is also challenging, since it represents a tradeoff between cost and waiting time and requires unknown future knowledge of the workload. Over-provisioning the shared cluster can eliminate any cost benefit it offered, while under-provisioning it can result in very long waiting times. Both scenarios can incentivize users to defect from a shared cloud cluster, and instead directly acquire resources from cloud. Finally, as we observed in §3, the longitudinal job trace we analyzed experienced significant changes in workload intensity each year, which illustrates the challenge of optimal provisioning.

4.1.2 Cloud Bursting. The cloud bursting policy builds on the lift-and-shift policy by enabling the cluster scheduler to dynamically acquire on-demand cloud resources to service jobs if fixed reserved resources are fully utilized. Below, we consider two cases of the cloud bursting policy: without job waiting and with job waiting.

Cloud Bursting without Job Waiting. Cloud bursting without waiting is typically referred to as autoscaling: in this case, if any job arrives to the scheduler and fixed reserved resources are fully utilized, then, as illustrated in Figure 4b, the scheduler dynamically acquires on-demand cloud resources to execute the job, and

releases them when the job is complete. A shared cloud cluster that utilizes cloud bursting without job waiting addresses the problem of excessively long waiting under bursty workloads from the lift-and-shift policy. While the potential for over- or under-provisioning the fixed reserved resources remains, shared cloud clusters that use cloud bursting generally need to reserve fewer fixed resources to minimize their cost (as some of the workload executes on on-demand resources). However, the discount this policy offers users is a function of the workload's characteristics. In particular, if the workload is highly bursty, then a high fraction of jobs will execute on on-demand resources, which will increase the overall cost. Since many batch workloads are bursty, as we show in §6, this policy offers little discount relative to users defecting from the shared cluster and acquiring on-demand resources themselves.

Cloud Bursting with Waiting. We can address the high cost of the cloud bursting without waiting by having the cluster scheduler force some jobs to wait for fixed reserved resources for some amount of time. As mentioned in §2, prior work defines a range of waiting policies that force jobs to wait based on the queue or job characteristics. In general, these policies define thresholds on job length and waiting time, and force jobs to wait if they are longer than some threshold, or if they would wait less than some threshold. Such policies provide a tradeoff, depending on the thresholds, between the lower cost and high waiting time of the lift and shift policy, and the higher cost and zero waiting time of the cloud bursting without waiting policy. Importantly, the optimal provisioning of fixed reserved resources will be a function of the thresholds. For example, if the waiting time threshold is infinite, then the scheduler will never acquire on-demand resources and the provisioning problem devolves to lift and shift, while, similarly, if the threshold is zero, then the provisioning problem devolves to cloud bursting without waiting. Figure 4c depicts cloud bursting with waiting.

In general, the longer the shared cloud cluster forces jobs to wait for resources, the higher the utilization of the fixed reserved resources, which are discounted, and lower the use of high-cost on-demand resources, which results in a lower overall cost. However, as shown in §3, since many jobs are small, e.g., 75% of users have an average job runtime <4 hours, even moderate absolute waiting times are often high relative to users' average job runtime. As we discuss in §6, this may incentivize users with small jobs to defect from a shared cluster and acquire cloud resources themselves unless the shared cluster offers a substantial discount.

4.1.3 Flying Solo. Finally, the flying solo policy refers to a special case where a shared cluster acquires zero fixed reserved resources at a discount, i.e., there is no shared cloud cluster, which effectively requires users to acquire cloud resources themselves. Figure 4d illustrates provisioning for multiple users that are flying solo. In this case, users determine how many resources they reserve versus acquire on-demand based on their own individual workload's characteristics, as well as cost and performance requirements. Note that this approach does not benefit from any statistical multiplexing of low-price fixed reserved resources between users. As a result, individual users benefit much less from reserving low-price fixed resources: the typical cost-optimal approach for users is to reserve zero (or a small number of resources) and dynamically acquire on-demand resources to execute most jobs. This results in a higher price, often at or near the on-demand price, but zero waiting time, which is similar to the cloud bursting without waiting policy above.

4.2 Resource Pricing Policies

As discussed above, shared cloud clusters run jobs from many users on low-cost fixed reserved resources and, potentially, on higher-cost on-demand resources. Shared cloud clusters have multiple options for how to distribute these resources' cost across users. As discussed below, we consider two pricing policies at different ends of the economic spectrum.

4.2.1 Socialist Pricing Policy. We define a socialist pricing policy that evenly divides the total cost of the shared cluster by the total computation time, and charges all users a fixed price per unit time their jobs run on the cluster. We call this the cluster's *normalized price*. This pricing policy distributes the costs evenly across all users. We generally refer to a cluster's normalized price as a fraction of the on-demand price. So, for example, a normalized price of 0.9 means that users receive a 10% discount for using the shared cluster compared to flying solo and acquiring on-demand resources to execute their jobs. Since the socialist pricing policy does not charge users more if their jobs run on higher-cost on-demand resources, it advantages users with bursty workloads.

Under the cloud bursting provisioning policies, bursty users are more likely to cause the shared cluster to acquire on-demand resources to execute their jobs at a higher cost. However, under our socialist pricing policy, these costs are amortized across all users. Likewise, users with steadier workloads that are more likely to run on fixed reserved resources are effectively penalized, as they must pay a normalized price that is higher than the discounted price of the reserved resources. Thus, this policy weakens the financial incentive for steadier users to participate in a shared cloud cluster.

4.2.2 Capitalist Pricing Policy. The capitalist pricing policy charges users based on the resources their jobs actually execute on. Thus, if a user's job executes on an on-demand resource, then the user simply pays the corresponding on-demand price, while if the job executes on a fix reserved resource, then the user pays the discounted reserved price (normalized by utilization of the reserved resources). Unlike the socialist policy, the capitalist policy does not distribute costs across users. Thus, under the cloud bursting policies, users with a highly bursty workload will pay more than those with a steadier workload, since their jobs will be more likely to run on higher-cost on-demand resources. As a result, the capitalist policy weakens the financial incentive for bursty users to participate in a shared cloud cluster.

5 IMPLEMENTATION

We adapted and extended an open-source trace-driven job simulator [14], written in python, to evaluate the effect of the provisioning and pricing policies from the previous section [17]. Specifically, we augmented the simulator to compute cost based on the socialist and capitalist pricing policies for the different provisioning policies in the previous section. The simulator takes a job trace as input, with a specified provisioning and pricing policy, the number of fixed reserved resources, and the relative price of fixed reserved and on-demand resources, and simulates the scheduling of jobs on the fixed reserved and on-demand resources, if applicable. Each fixed

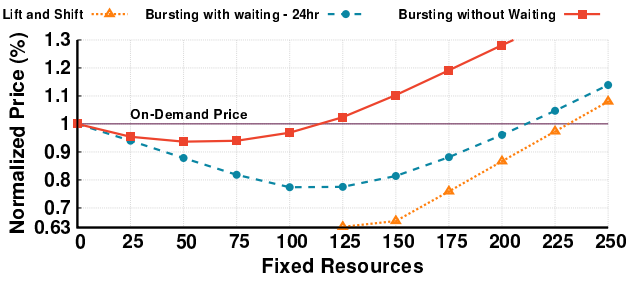


Figure 5: ***Normalized price of shared cloud cluster under different resource provisioning policies.***

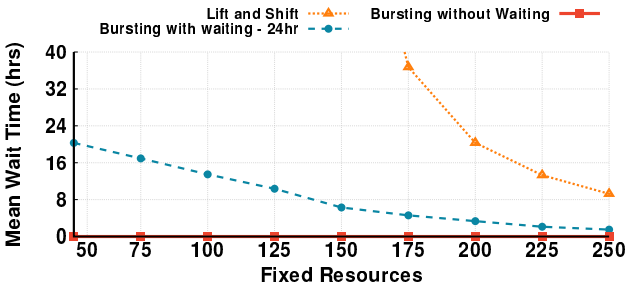


Figure 6: ***Mean job waiting time of shared cloud cluster under different resource provisioning policies.***

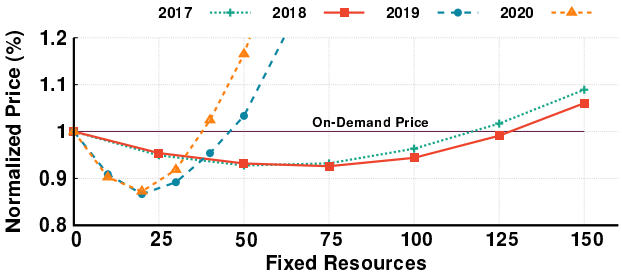


Figure 7: ***Normalized price as a function of fixed resources for each year of our job trace.***

resource includes some number of cores and memory, and each job in the trace requires some number of cores and memory. In our case, we set the relative price of fixed reserved resources to be 60% that of on-demand resources, which is roughly the same discount level offered by Amazon Web Services and Google Compute Platform.

Our simulator uses a best-fit heuristic to place jobs on reserved resources. Our simulator also assumes that, similar to most cloud platforms, multiple types of on-demand resources are available of different sizes, e.g., memory and cores, and that their price is proportional to their size. When selecting an on-demand resource to execute a job, the simulator selects the smallest (and thus cheapest) on-demand resource that is large enough to fit the job. We model the sizes and prices for fixed and on-demand resources using the `m5` family of general-purpose server instances offered by Amazon Web Services. We use a First-Come-First-Serve (FCFS) scheduling policy. While FCFS is non-optimal, recent work shows that combining FCFS with cloud bursting to bound wait time mitigates the advantage of optimal scheduling policies, such as Shortest Job First [18].

The simulator also implements various waiting policies depending on the provisioning scenario. Specifically, the lift-and-shift policy assumes an all-jobs-wait waiting policy, the cloud bursting policy without waiting assumes a no-jobs-wait policy, and the cloud bursting with waiting policy assumes a some-jobs-wait policy, which forces jobs to wait based on configurable thresholds on their runtime and waiting time. For our evaluation, jobs generally wait for fixed reserved resources if their waiting time is less than 24 hours or if their running time is greater than 3 minutes. Finally, the simulator computes each job's waiting time, whether it ran on fixed or on-demand resources, and the total cost of resources.

6 EVALUATION

We use the simulator above to evaluate the user incentives of the different provisioning and pricing policies from §4 on the longitudinal job trace analyzed in §3. We analyze the incentives for each pricing policy to understand whether users are incentivized to participate in a shared cloud cluster, or if they would defect. Most of our analyses focus on the year 2016, as a representative year, unless otherwise stated.

6.1 Lift-and-Shift Policy

Figure 5 shows the normalized price using the socialist pricing policy with the lift-and-shift provisioning policy as a function of provisioning different numbers of fixed resources. The graph also shows the normalized price of the cloud bursting policies, which we discuss below. Recall that the normalized price is the price users pay per unit time as a function of the on-demand price; thus, a normalized price of 1 is equivalent to paying the on-demand price. The graph shows that lift-and-shift's static provisioning generally offers a lower price than the cloud bursting policies, and reaches a minimum price of 60% the on-demand price at 125 fixed resources—at this point, the fixed resources are 100% utilized.

By contrast, the cloud bursting policies have a higher normalized cost because they use some higher-cost on-demand resources to service a fraction of the workload. However, Figure 6 shows the tradeoff in waiting time: the lift-and-shift waiting time is more than a week (not pictured) at the optimal 125 fixed resources. Even when reserving 225 resources, the average job waiting time is 14 hours, and the normalized price is near 1, offering nearly zero discount.

Pricing and Defection Analysis. Since jobs execute on fixed reserved resources under lift-and-shift, the socialist and capitalist pricing policies are equivalent. Our results show that reserving resources using lift-and-shift is not a viable strategy for a shared cloud cluster, since users are incentivized to defect due to very high waiting times caused by bursty workloads, as they can always use on-demand resources without incurring any waiting time.

6.2 Cloud Bursting Policy

We evaluate cloud bursting both without and with waiting.

6.2.1 Cloud Bursting without Waiting. Figure 5 shows that cloud bursting without job waiting results in a substantially higher cost than the other policies. The minimum price occurs at 50 reserved resources, and results in a normalized cost under the socialist pricing policy of ~94% of servicing the workload using on-demand resources, or a ~6% discount. While small, this approach does offer a modest 6% discount for users along with a guarantee of no job waiting, as shown in Figure 6. The problem is that achieving even this small discount requires the shared cloud cluster to optimally provision the fixed resources. As Figure 5 shows, the

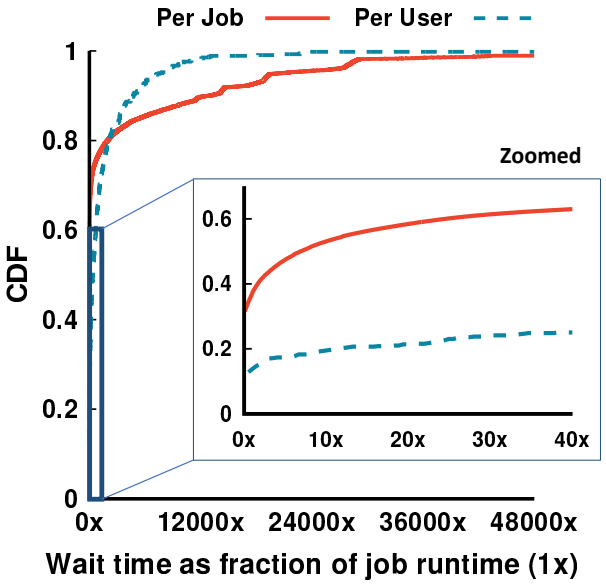


Figure 8: ***CDF of average waiting time (as a factor of job running time) across users and jobs.***

discount reduces, and is ultimately eliminated, if fixed resources are over- or under-provisioned. If over-provisioned too many reserved resources, the normalized price can actually exceed the cost of servicing the workload using on-demand resources.

Thus, shared cloud clusters are generally incentivized to under-provision the fixed resources to prevent high costs, which, in practice, would likely reduce the small discount even further. Figure 7 shows the normalized price across 4 different years of our job trace as a function of the number of fixed reserved resources, and shows that the optimal number varies significantly from year-to-year, and is difficult to predict. For example, if a shared cloud cluster reserved 75 resources in 2019 based on the optimal amount from 2018, the cluster's normalized price would be much higher than the on-demand price, since the optimal fixed resources in 2020 was many fewer resources due to changes in the workload.

Pricing and Defection Analysis. As shown above, the socialist pricing policy provides a small discount for users at the risk of potentially paying a high price if the cluster over-provisions resources. In contrast, the capitalist pricing policy results in different normalized prices for different users, ranging between near the maximum discount of 60% to near 0% discount (for highly bursty users). In general, large users with bursty workloads, which is ~7% of the users in our trace, pay a higher than the average normalized price (since many of their jobs execute on on-demand resources), while smaller users with fewer jobs receive a higher discount (since their jobs tend to run on reserved resources).

Under optimal provisioning, users have no incentive to defect from a shared cloud cluster without waiting, as they all receive some, albeit potentially small, discount. That said, given the generally small discount, there is also not a strong incentive for users to participate, especially when factoring in the risk of a higher cost due to non-optimal provisioning.

6.2.2 Cloud Bursting with Waiting. Figure 5 also shows the normalized price of cloud bursting with job waiting. In this case, the

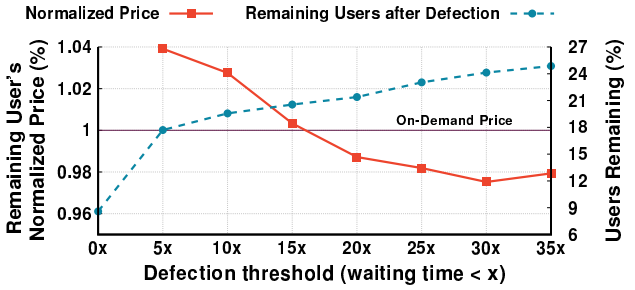


Figure 9: ***Normalized price (left y-axis) and fraction of participating users (right y-axis) as a function of a defection threshold, which is defined in terms of waiting time as a factor of a user's average job run time.***

scheduler dynamically acquires on-demand resources to run any job that would wait more than 24 hours to run on the fixed resources. The figure shows that this policy significantly reduces the normalized price compared to cloud bursting without waiting. In particular, at the optimal number of reserved resources, the normalized price is ~0.77, and thus offers a ~23% discount relative to using on-demand resources (or a 15% increase relative to no waiting).

However, Figure 6 shows that this approach also substantially increases job waiting time: at the optimal price, the waiting time is 13.5 hours. Recall from §3 that most jobs are small. Figure 8 plots the CDF of average waiting time (as a factor of job running time) across jobs and users for the case above. The graph shows i) that 50% of jobs must wait at least 10× their run time, and ii) that 20% of users have an average job waiting time more than 15× their average job run time. While the waiting time decreases if the cluster provisions more fixed resources, the discount does as well.

Pricing and Defection Analysis. The likelihood of a user defecting from a shared cloud cluster in the case above depends on their sensitivity to both price and performance. Some users might be willing to tolerate high wait times for a discount, while others might not. Importantly, though, when users defect, they increase the normalized price that other users pay, since they reduce the total number of users over which the shared cluster amortizes its cost. To understand this dynamic, Figure 9 plots a defection threshold on the x-axis, as a function of waiting time (as a factor of a user's average job runtime), the normalized price on the left y-axis, and the percentage of remaining users in the cluster on the right y-axis. As the threshold increases, more users participate in the shared cloud cluster because they are willing to wait longer, and the normalized price goes down. However, even at excessively high thresholds, the fraction of participation is low (<30%), which significantly diminishes the discount (down to just a few percent). This occurs because, as shown in §3, most of the jobs are small, and thus their wait times are quite high relative to their job length.

As shown, if the defection threshold is less than 15×, the incentives collapse and there is no incentive to participate, since the price of a shared cluster becomes too high. Under the capitalist pricing policy, the incentives collapse even faster. Figure 10 shows the price for every user in order under capitalist pricing. As shown, roughly 22% of the, generally larger and burstier, users actually pay more than the average normalized price in this case. These users would thus defect earlier than under the socialist pricing policy.

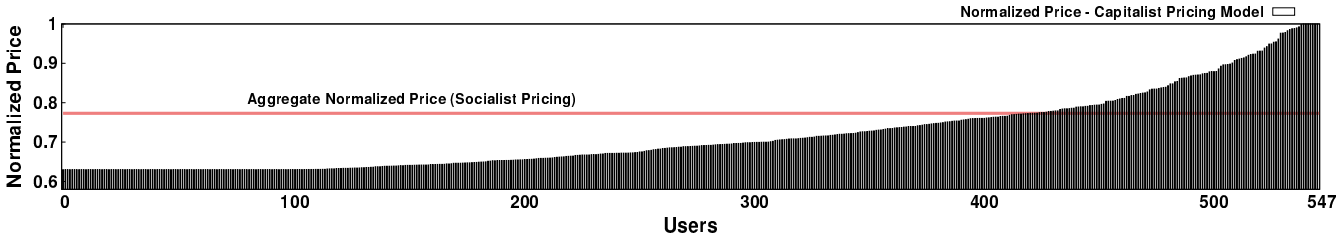


Figure 10: ***Each user's normalized price (in order) under capitalist pricing and cloud bursting with waiting.***

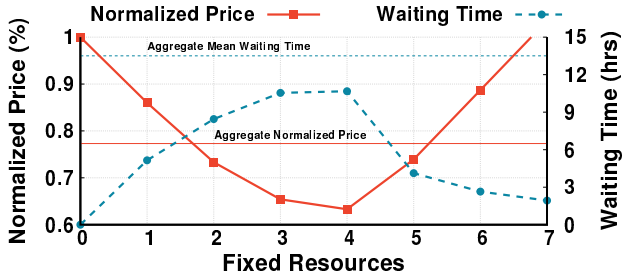


Figure 11: ***Normalized price (left y-axis) and waiting time (right y-axis) as a function of fixed reserved resources for a representative user "flying solo."***

6.3 Flying Solo

The analyses above indicate that there is little financial incentive to operate shared cloud clusters, and motivates users to "fly solo" and acquire cloud resources themselves to service their own workload. In this approach, users can still potentially make use of discounted fixed reserved resources if their workload is large enough to have some continuous baseload. For example, Figure 11 shows one such user from our job trace. This user's minimum price when flying solo is 4 fixed reserved resources, which offers a lower price than the cloud bursting with waiting scenario above, while also offering a much lower waiting time. Thus, even if this user were tolerant of long wait times (and thus would participate in the cluster above), flying solo is a better option in terms of both price and waiting time. However, only ~2% of users in our trace fall into this category.

The rest of the users fall into one of two broad categories: small users (46%) and bursty users (52%). Reserving resources is not cost-effective for either category, since neither has a baseload. Based on our analysis above, small users that infrequently submit small jobs may be sensitive to long waiting times, and thus might prefer on-demand resources. The bursty users submit larger numbers of jobs in bursts, and get the most benefit from a shared cloud cluster, especially under the socialist pricing policy (which distributes the cost of handling job bursts using on-demand resources across all users). These users might also be more tolerant of long waiting times, since their jobs are submitted in large bursts.

Pricing and Defection Analysis. Based on the analysis above, roughly 48% of users receive a lower waiting time (and in a few cases also a lower price) compared to using a shared cloud cluster (using the cloud bursting with waiting policy). These users are unlikely to participate in a shared cloud cluster. The remaining 52% can get a cost benefit from using a shared cloud cluster, but this is generally at the expense of the other 48%. If many users defect, the price increase eliminates a shared cloud cluster's incentives.

7 RELATED WORK

Our work is related to a variety of prior work on cloud scheduling, bursting, and cost optimization. However, in general, our work differs from this prior work in its focus on the incentives for shared cloud clusters, rather than a particular policy for scheduling, bursting, or allocating resources.

Cloud Scheduling. There has been a substantial amount of work on optimizing the provisioning of fixed reserved resources on cloud platforms. Much of this work makes strong assumptions about the job workload distribution, i.e., that it is continuous and uniform, rather than consisting of discrete jobs. As a result, many of the approaches adopt dynamic and linear programming-based solutions [23, 24, 31, 34, 35]. Our work differs in that we analyze a specific, large-scale job trace that is representative of other large-scale batch traces. We compute optimal prices directly using the trace, rather based on an idealized analytical model of the workload.

Cloud Bursting. Our analysis is also related to prior work on resource provisioning, job scheduling, and autoscaling for hybrid clouds, which execute jobs on fixed on-prem resources but periodically burst into the cloud [3, 19, 21, 22, 25, 30]. Similarly, shared cloud clusters can also periodically burst by acquiring on-demand resources to execute queued jobs. However, again, our work does not propose particular policies for provisioning or scheduling, but rather analyzes the cost and waiting time for multiple different resource provisioning and pricing policies for a shared cloud cluster to understand users' participation incentives.

Cloud Cost Optimization. Finally, there is also significant work on cloud cost optimization, which focuses on selecting the type and number of VM instances specific applications require to minimize their cost for a given level of performance [5, 7, 16, 33]. These approaches often use machine learning to profile jobs' resource usage and then match them with the minimum size and cost VM that best aligns with their resource usage profile. Our work is orthogonal to this work. In general, job schedulers require users to specify the resources their jobs require. In our analysis, we assume these resource requests are accurate, although jobs may utilize fewer resources. While improving resource efficiency may change the absolute numbers in our analysis, it is unlikely to change the broader insight of our work, which is that there are weak-to-no incentives for shared cloud clusters given the current reserved discount.

8 CONCLUSION

Large organizations have long operated shared cloud clusters to reduce costs by taking advantage of statistical multiplexing among users. Many of these organizations are, or are considering, moving these shared clusters to cloud platforms. Our analysis shows that

there is little financial benefit from statistical multiplexing of discounted fixed reserved resources on shared cloud clusters, and that most users are incentivized to directly acquire resources from the cloud, since it provides a lower waiting time (and sometimes a lower cost). Thus, while there may be, potentially numerous, other reasons for hosting shared cloud clusters, i.e., ease-of-administration, there is little-to-no financial benefit.

Acknowledgements. This work is funded, in part, by National Science Foundation grants CNS-1908536, CNS-1925464 and CNS-2213636. We also acknowledge and thank the Research Computing team at the UMass Medical School for providing access to the longitudinal batch traces from the UMass Green High Performance Computing Cluster (GHPCC) [15].

REFERENCES

[1] 2022. Amazon EC2 Spot Instances. https://aws.amazon.com/ec2/spot/.
[2] 2022. AWS - Discounts on Reserving Resources. https://aws.amazon.com/ec2/pricing/reserved-instances/pricing/.
[3] 2022. AWS ParallelCluster Auto Scaling. https://docs.aws.amazon.com/parallelcluster/latest/ug/autoscaling.html.
[4] 2022. Azure Spot Virtual Machines. https://azure.microsoft.com/en-us/products/virtual-machines/spot/.
[5] 2022. Cloud Cost Optimizer. https://research.redhat.com/blog/research_project/cloud-cost-optimizer/.
[6] 2022. Cloud Growth in Future. https://www.globenewswire.com/news-release/2022/05/06/2437934/0/en/Cloud-Computing-Market-to-Grow-at-a-CAGR-of-11-until-2028-BlueWeave-Consulting.html.
[7] 2022. Curator. https://github.com/operate-first/curator/.
[8] 2022. Google Preemptible Virtual Machines. https://cloud.google.com/compute/docs/instances/preemptible.
[9] 2022. Kubernetes on AWS. https://aws.amazon.com/kubernetes/.
[10] 2022. On-Prem Computing. https://www.techslang.com/definition/what-is-on-premises/.
[11] 2022. On-Prem Computing, Expensive than Cloud. https://www.executech.com/insights/the-cloud-vs-on-premise-cost-comparison/.
[12] 2022. Privacy and Regulatory on On-Prem Computing. https://www.cleo.com/blog/knowledge-base-on-premise-vs-cloud.
[13] 2022. Rapid Growth of Cloud. https://www.capacitymedia.com/article/2afswwuvis94wy12r320w/news/google-cloud-growing-45-a-year-with-azure-at-40-says-canalys.
[14] 2023. Job Simulator. https://github.com/sustainablecomputinglab/waitinggame/tree/master/simulator.
[15] 2023. University of Massachusetts Green High Performance Computing Cluster. http://wiki.umassrc.org/wiki/index.php/MainPage.
[16] Abdullah Alzaqebah, Rizik Al-Sayyed, and Raja Masadeh. 2019. Task Scheduling Based on Modified Grey Wolf Optimizer in Cloud Computing Environment. In *2019 2nd International Conference on new Trends in Computing Sciences (ICTCS)*. IEEE.
[17] Pradeep Ambati, Noman Bashir, David Irwin, and Prashant Shenoy. 2020. Waiting Game: Optimally Provisioning Fixed Resources for Cloud-Enabled Schedulers. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*. IEEE.
[18] Pradeep Ambati, Noman Bashir, David Irwin, and Prashant Shenoy. 2021. Good Things Come to Those Who Wait: Optimizing Job Waiting in the Cloud. In *Proceedings of the ACM Symposium on Cloud Computing*.
[19] Tekin Bicer, David Chiu, and Gagan Agrawal. 2011. A Framework for Data-Intensive Computing with Cloud Bursting. In *IEEE International Conference on Cluster Computing*. IEEE.
[20] Kavitha Chandra. 2003. Statistical Multiplexing. *Wiley Encyclopedia of Telecommunications* 5 (January 2003).
[21] Tian Guo, Upendra Sharma, Prashant Shenoy, Timothy Wood, and Sambit Sahu. 2014. Cost-Aware Cloud Bursting for Enterprise Applications. *ACM Transactions on Internet Technology (TOIT)* (2014).
[22] Tian Guo, Upendra Sharma, Timothy Wood, Sambit Sahu, and Prashant Shenoy. 2012. Seagull: Intelligent Cloud Bursting for Enterprise Applications. In *USENIX Annual Technical Conference*.
[23] Yu-Ju Hong, Jiachen Xue, and Mithuna Thottethodi. 2011. Dynamic Server Provisioning to Minimize Cost in an IaaS Cloud. In *Special Interest Group on Measurement and Evaluation (SIGMETRICS)*.
[24] Menglan Hu, Jun Luo, and Bharadwaj Veeravalli. 2012. Optimal Provisioning for Scheduling Divisible Loads with Reserved Cloud Resources. In *IEEE International Conference on Networks (ICON)*.
[25] Sriram Kailasam, Nathan Gnanasambandam, Janakiram Dharanipragada, and Naveen Sharma. 2010. Optimizing Service Level Agreements for Autonomic Cloud Bursting Schedulers. In *International Conference on Parallel Processing Workshops*. IEEE.
[26] Michael Kuchnik, Jun Woo Park, Chuck Cranor, Elisabeth Moore, Nathan DeBardeleben, and George Amvrosiadis. 2019. This is Why ML-driven Cluster Scheduling Remains Widely Impractical. *Technical Report* (2019).
[27] Tania Lorido-Botran, Jose Miguel-Alonso, and Jose A Lozano. 2014. A Review of Auto-Scaling Techniques for Elastic Applications in Cloud Environments. *Journal of Grid Computing* (2014).
[28] Marko Luksa. 2017. *Kubernetes in Action.* Simon and Schuster.
[29] Michael Mattess, Christian Vecchiola, Saurabh Kumar Garg, and Rajkumar Buyya. 2011. Cloud Bursting: Managing Peak Loads by Leasing Public Cloud Services. In *Cloud Computing: Methodology, Systems, and Applications*. CRC Press.
[30] Shuangcheng Niu, Jidong Zhai, Xiaosong Ma, Xiongchao Tang, and Wenguang Chen. 2013. Cost-effective Cloud HPC Resource Provisioning by Building Semi-Elastic Virtual Clusters. In *The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC)*.
[31] Siqi Shen, Kefeng Deng, Alexandru Iosup, and Dick Epema. 2013. Scheduling Jobs in the Cloud using On-demand and Reserved Instances. In *International European Conference on Parallel and Distributed Computing (Euro-Par)*.
[32] Abraham Silberschatz, Peter B Galvin, and Greg Gagne. 2018. *Operating System Concepts, 10e Abridged Print Companion.* John Wiley & Sons.
[33] Jose Luis Lucas Simarro, Rafael Moreno-Vozmediano, Ruben S Montero, and Ignacio Martín Llorente. 2011. Dynamic Placement of Virtual Machines for Cost Optimization in Multi-Cloud Environments. In *2011 International Conference on High Performance Computing & Simulation*. IEEE.
[34] Ruben Van den Bossche, Kurt Vanmechelen, and Jan Broeckhove. 2015. IaaS Reserved Contract Procurement Optimisation with Load Prediction. *Future Generation Computer Systems* (2015).
[35] Wei Wang, Baochun Li, and Ben Liang. 2013. To Reserve or Not to Reserve: Optimal Online Multi-Instance Aquisition in IaaS Clouds. In *International Conference on Autonomic Computing (ICAC)*.
[36] Andy B Yoo, Morris A Jette, and Mark Grondona. 2003. Slurm: Simple linux Utility for Resource Management. In *Job Scheduling Strategies for Parallel Processing*. Springer.