Minimizing Cost and Maximizing Performance for Cloud Platforms

Jashwant Raj Gunasekaran
The Pennsylvania State University

Abstract

We are witnessing the rapid growth of cloud computing with the proliferation of tenants adopting cloud for elasticity, availability, and flexibility for a plethora of applications. To efficiently cater for different tenant requirements, cloud providers have steadily evolved to offer a myriad of resource and service types which inherently complicates the cloud adoption process. On the other hand, the perpetuating growth of cloud tenants in turn impel providers to expand datacenters to cope with the tenant demand. The objective of this proposal is to maximize the performance and minimize the cost for both tenants and cloud providers, by providing efficient means of managing resource allocations for their applications. Towards this, the proposal comprises of three intertwined tasks. First, we start from a tenant perspective, with the first two tasks aimed at investigating the primary reasons for performance-cost inefficiency. Second, from a provider perspective, the third task investigates the primary reasons for performance-energy inefficiency in datacenters. All the three tasks can collectively improve the performance and cost efficiency of emerging applications in next generation cloud platforms.

CCS Concepts: • **Computer systems organization** → **Real-time systems**; • **Scheduling and Resource Management**;

Keywords: public-cloud, cost, performance, efficiency

ACM Reference Format:
Jashwant Raj Gunasekaran. 2020. Minimizing Cost and Maximizing Performance for Cloud Platforms. In *21st International Middleware Conference Doctoral Symposium (Middleware '20 Doctoral Symposium), December 7–11, 2020, Delft, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3429351.3431747

Middleware '20 Doctoral Symposium, December 7–11, 2020, Delft, Netherlands

ACM ISBN 978-1-4503-8200-7/20/12…$15.00
https://doi.org/10.1145/3429351.3431747

1 Introduction

Cloud computing has emerged as an attractive computing paradigm, where both hardware and software resources can be leased from a cloud provider to achieve better economy at scale, elasticity, availability and flexibility of resources for executing a wide range of scientific and enterprise applications. It has essentially become a foundation for digital business that encompass newer class of data intensive applications such as machine learning (ML) which have high computational demands along with Service Level Objective (SLO) requirements. Cloud computing bears the tremendous advantage, that frees tenants (customers) from investing upfront on clusters to cater peak loads, by offering on-demand resource provisioning. Furthermore, public clouds have steadily evolved to attract a large fraction of tenants by offering a more convenient pay-as-you-go service model. Owing to these prolific advantages, the customer-adoption for cloud continues to grow rapidly as tenants of varying sizes endure cloud as a first class deployment option. According to the 2020 cloud report by Flexera [4], 20% of enterprises spend more than $12 million per year on public clouds. Further, more than 50% of enterprise workloads and data are expected to be in a public cloud within 12 months.

On the other hand, to cope with the perpetuating demand of cloud tenants, there is a massive growth in public cloud spending both in terms of capital expenditure (CapEx) and operating expenditure (OpEx) [32]. Large-scale providers such as Amazon EC2 [10], Microsoft Windows Azure [6], Google Compute Engine (GCE) [18] and Alibaba Cloud [1] host tens of thousands of applications on a daily basis. According to Gartner [3] the projected revenues for different cloud services are more than 350 billion dollars. Therefore there is a profound growth, from both aspects of cloud-migration and cloud infrastructure spending.

2 The Problem

An illustration of the complex interaction between cloud tenants and providers spanning across physical servers, runtime systems and applications is shown in Figure 1. Due to the proliferation of cloud-bound tenants, it is well-known that cloud providers offers a plethora of resource types for tenants to choose from, which in-turn complicates the cloud adoption process. Specifically, the primary factors associated with the resources such as resource provisioning latency, resource demand estimation (autoscaling) and billing complexity, play a crucial role in determining tenant satisfaction. As a result, cloud-bound tenants face both a **"cost**

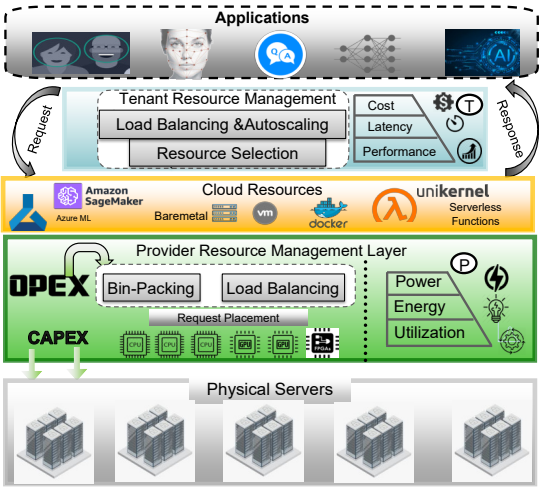


Figure 1. Cloud system stack depicting the different components for both tenants and provider.

wall" (not able to run cost-effectively on the cloud) and a **"performance wall"** (difficulty in choosing and extracting the best performance from the resources rented). Furthermore, these aspects are enlaced to each other – there is an inherent tension between keeping cost low of vs. maintaining the performance high.

On the other hand, from a provider's outlook, the increased spending by cloud-providers on Capex without paying attention to data-center utilization results in poor Opex efficiency. Especially for public cloud providers, overprovisioning servers in order to curtail tail latency [38, 41] during peak loads of tenant applications, further accentuates the under-utilization problem. Each server in a datacenter incurs a one-time cost of several thousands of dollars (Capex), while they continuously embrace Opex in terms of monthly electric utility bills for both the peak power draw as well as the aggregate energy use. Prolonged underutilization of servers leads to wasted resource and power, thereby reducing the energy efficiency of the entire datacenter [17, 30]. Therefore at the provider level, platform providers face a **"utilization wall"**(not able to fully utilize all cluster resources).

3 The Proposal

This proposal adopts a three pronged approach towards holistically addressing the above mentioned inefficiencies from both cloud tenant and provider perspectives. First, we start from a tenant perspective (**T**), with the first two tasks aimed at investigating the primary reasons for **performance-cost** inefficiency. More specifically, we focus on ML inference applications to improve the cost and performance in terms of accuracy and latency. Towards this, we characterize the recent public offering like serverless functions to help alleviate resource provisioning latency delays incurred by traditional VMs. Further to improve ML inference performance we focus on ensemble learning [8] techniques. Second, from a provider perspective (**P**), the third task investigates the primary reasons for **performance-energy** inefficiency in datacenters. Due to the rapid adoption of serverless functions [33], providers face critical challenges to ensure high server utilization while guaranteeing the fast startup latencies and instant scalability. Towards this, we plan to expose application characteristics that can be exploited by the provider to employ optimal resource management decisions.

Interplay: The three tasks are complementary to each other such that, the first two tasks can jointly optimize performance, cost and latency of hosting ML applications for tenants. The third task improves energy efficiency of datacenters while guaranteeing the same application performance for tenants that can be obtained from the first two tasks.

3.1 Related Work

There are several research works that optimize for the resource provisioning cost for various tenant applications in the public cloud. These works broadly fall into two categories: (i) tuning the auto-scaling policy based on changing needs [7, 9, 19, 24, 25, 34, 39], (ii) prediction-based proactive provisioning auto-scaling policy [19, 23, 24, 28, 35]. Complementary to these approaches, we propose to user serverless functions along with VMs for cost-effective autoscaling. Several works [11, 12, 29, 42] have tried to optimize ML inference serving cloud on cloud in terms of cost, latency and accuracy . The most recent works are InFaas [40] and Clipper [13]. InFaas is limited to single model selection and thus suffers from higher latencies to reach higher accuracy. Clipper leverages ensembling [8] but employs static model selection policies and hence consume more resources thereby incurs high cost. In this proposal we focus on bringing the cost of ensemble learning comparable to single model inference. On the provider front, a large body of work [14–16, 27, 31, 37] have looked at ensuring SLO guarantees for latency critical applications by developing sub-millisecond scheduling strategies using both distributed and hybrid schedulers. However, these policies are not entirely suitable for managing serverless platforms. Archipelago [36] is the most recent work that looks into serverless platforms, still it over-provisions containers due to non-batching nature.

4 Proposal Tasks

The individual tasks of this proposal are explained in detail below.

4.1 Optimizing Latency and Cost

The *first dimension* of this proposal addresses the provisioning latency problem at scale that significantly affects the response latency of user-facing applications, without compromising on cost. The end goal of this proposal is to develop suitable resource selection policies to hide the provisioning latency from inference execution latency and at the

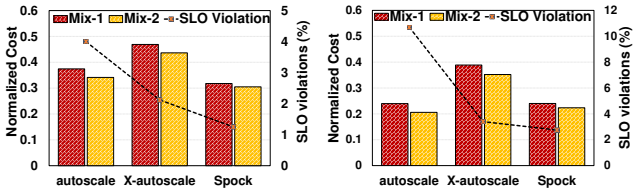
same minimize the cost hosting the applications. To this end, we investigate how using serverless functions (lambdas) can alleviate the shortcomings of VM-based auto-scaling. We identify two key insights; (i) provisioning VMs for the peak demands always leads to higher cost of deployment [22]. While, under provisioning VMs leads to severe SLO violations for queries [24, 38]. (ii) Using lambdas exclusively to deploy the web service would overcome the SLO violation problem, however doing so is not cost effective. Based on these insights, we propose a *Spock* [21] hybrid resource procurement system that exploits lambdas to be used with VMs.

4.1.1 Design We design *Spock* [21] which exploits lambda to be used along with existing VM-based autoscaling mechanism. In *Spock*, whenever an RM cannot find a free slot in a VM to serve a new request, it redirects the request to execute in a lambda function. Then, based on the request rate, the scaling policy decides to scale the number of VM instances required to satisfy the current load. Until the required number of VMs instances are spawned, any query which cannot be accommodated on VM is redirected to lambda functions.

4.1.2 Results We evaluate *Spock* for ML inference workload using Wiki web server traces to mimic data-center arrival rates. The resource procurement schemes used for comparison are: (i) deploying only via VMs with autoscaling (autoscale); (ii) using conservative over-provisioning with autoscaling i.e., acquire 1.5 times more resources than required (X-autoscale). With respect to these resource acquisition schemes, we perform evaluations using two scale-out policies namely (i) reactive and (ii) predictive. Figures 2a and 2b show that *Spock* reduces the cost and SLO violations compared to *autoscale* and X-autoscale under both the scaling policies for both the workload mixes. *Spock*, when compared to *autoscale*, reduces SLO violations by 68% and 74% for both the reactive and predictive scale-out policies, respectively. At the same time, *Spock* also reduces cost by 15%, when employing the reactive policy. The cost savings are not significant for the predictive policy because it inherently avoids VM over-provisioning, but results in more SLO violations.

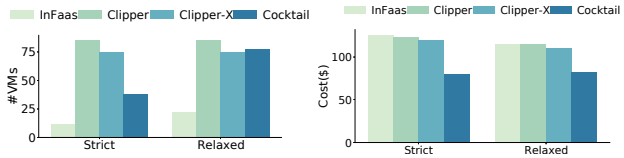
4.2 Optimizing Performance and Cost

Our first proposal addresses the latency and cost problem, while considering performance of ML inferences as a blackbox. Since the performance for inference is jointly determined using both accuracy and latency, the *second dimension* of this proposal targets to improve the accuracy at low latency for ML inference by leveraging ensemble learning, without increasing the resource usage. Towards this, we identify two key challenges with respect to ensemble learning [13]. First, ensemble learning is inherently resource hungry due to its distinctive nature of executing multiple inferences for a single request. Second, the increased resource footprint leads to excessive deployment costs. The main goal of this proposal is to reduce the resource-footprint of ensemble learning and further reduce the deployment cost of



(a) Reactive autoscaling. **(b)** Predictive autoscaling.

Figure 2. Cost savings and SLO violations of *Spock*.The cost is normalized to the scheme that only uses lambda.



(a) Number of VMs. **(b)** Cost Reduction.

Figure 3. Cost savings and VM reduction of *Cocktail* compared to three schemes.

hosting ensembles in public cloud. To this end, we propose *Cocktail*, which reduces the number of models used in an ensemble and employs novel autoscaling mechanisms to reduce the VMs provisioned the same. Further, we propose to use transient VMs to reduce the cost of deploying the ensemble framework. Transient VMs are 70-80% cheaper than standard VMs but can be interrupted at any time.

4.2.1 Design The overall design of Cocktail is two fold. First from the model ensemble part, we characterized the accuracy *vs.* latency of ensemble models and identified that we can prudently select a subset of available models under a given latency to achieve the desired accuracy. We leverage this in *Cocktail*, to design a novel dynamic model selection policy, which ensures the accuracy with significantly reduced number of models. The dynamic policy downscales the number of models, if more than $N/2 + 1$ models vote for the same inference. Further, to minimize the bias in predictions from multiple models, *Cocktail* employs a novel per-class weighted majority voting policy, which can effectively breaks ties, thereby minimizing the accuracy loss.

Second from a resource management part, we show that uniformly scaling resources for all models in the ensemble leads to over-provisioning of resources. To minimize over-provisioning, we build a distributed proactive weighted auto-scaling policy that utilizes the *importance sampling* technique to proactively allocate resources to every model. The weights are determined by frequency in which a particular model is chosen for requests with respect to other models in the ensemble. The weights are multiplied with predicted load to scale instances for every model pool. Further, *Cocktail* leverages transient VMs as they are cheaper, to drastically minimize the cost for hosting model-serving infrastructure in a public cloud. Note that, the resource procurement scheme proposed in *Spock* can be employed in

Cocktail to avoid SLO violations when starting new spot instances. We implement a prototype of *Cocktail* using both CPU and GPU instances on AWS EC2 [5] platform and extensively evaluate it using different request-arrival traces.

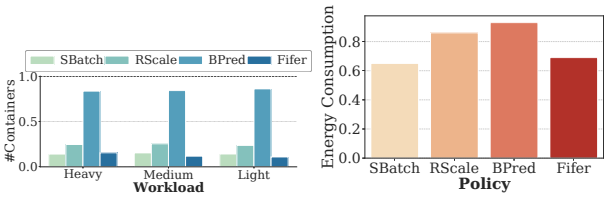
4.2.2 Results We compare *Cocktail* against (i) *InFaas* [40] which is our baseline that employs single model selection policy, (ii) *Clipper* [13] which uses static full model selection policy, and (iii) *Clipper-X* which is an enhancement to *Clipper* with a naïve dynamic model selection that does not utilize the vote-frequency based policy enforced in *Cocktail*.

Figure 3b plots the cost savings of *Cocktail* when compared to *InFaas*, *Clipper* and *Clipper-X* policies. It can be seen that, *Cocktail* is up to 1.4× more cost effective than *InFaas* for *Strict* workload. In addition, *Cocktail* reduces cost by 25% and 20% compared to *Clipper* and *Clipper-X* policies, owing to its dynamic model selection policy, which minimizes the resource footprint of ensembling. Figure 3a plots the reduction in the number of VMs used by all four schemes. It can be seen that, both *Cocktail* and *Clipper-X* spawn 49% and 20% fewer VMs than *Clipper*.

4.3 Improving Datacenter Energy Efficiency

A large fraction of the user-interactive applications are being hosted on serverless platforms, owing to its monetary rewards in terms of pay-per-event billing. The *third dimension* of this proposal identifies the inefficiencies in existing serverless platform providers in terms of scheduling and resource management (RM) for microservices (functions). We specifically focus on function chains where multiple stages of an applications (for instance complex ML inference pipelines) are connected as a series of inter-linked services. Since, infinite scalability is a key aspect of serverless computing, serverless providers inherently overprovision servers to cope with the scaling demand. This work identifies two key reasons that lead to resource over-provisioning. First, the existing RM policies are imperceptive to the total end-to-end SLO of the function-chain leading to sub-optimal resource scaling. Second, the RMs employ one-to-one mapping of requests to containers. This inherently leads to excessive number of containers being provisioned when handling a sudden burst of requests. To this end, we propose *Fifer* [20], a stage-aware resource provisioning and management of function chains for serverless platforms. We plan to leverage the "leftover slack" between function chains to design efficient batching-based resource management policies thereby increasing the overall cluster utilization.

4.3.1 Design We design *Fifer*, to take advantage of slack between function chains, towards calculating the batch-size (queue length) to determine the optimal number of requests that could be queued at every stage. *Fifer* is inherently *stage aware*, such that it proportionally allocates slack to every function stage of an application proportional to its execution time, and independently decides the scale-out threshold for every stage. This inherently minimizes the number



(a) Containers norm. to `Bline`. **(b)** Energy Savings norm. to `Bline`.

Figure 4. *Fifer*: Container Reduction and Energy Savings.

of containers used in every stage. Further, to reduce SLO violations from cold-starts resulting from proactive scaling, we design a novel *LSTM-based [26] prediction model*, which provides fairly accurate request arrival estimations even when there are large dynamic variations in the arrival rate. We implement *Fifer* as a part of the Brigade serverless workflow framework [2] in a 80-core Kubernetes cluster and extensively evaluate it with synthetic traces and comprehensive real-world traces.

4.3.2 Results Figure 4a show the average number of containers spawned for different RMs across a mix of pipelined ML inference workloads. It is evident that *Fifer* spawns the least number of containers on average compared to all other schemes except *SBatch*. This is because *SBatch* used fixed number of containers even for changes in request load, thus suffers from more SLO violations. Figure 4b plots the cluster-wide energy as an average of energy consumed across all nodes in the cluster measured over intervals of 10 seconds for the entire workload duration. It can be seen that *Fifer* is 30.75% more energy efficient than the *Bline* (for heavy workload-mix). This is because *Fifer* can accurately estimate the number of containers at each stage, thereby consolidating all active containers to fewer nodes. The energy savings are a result of servers only consuming idle power.

5 Concluding Remarks

Cloud and datacenters are becoming a quintessential crux of computing systems. Owing to the increased demand of tenants to host applications in cloud, datacenters are expanding rapidly both in terms of size and types of resources being offered. This proposal is aimed at understanding and addressing some of the key bottlenecks in improving the performance-cost efficiency for both cloud tenants and providers. As a part of future work, on we plan to optimize hosting ML training applications in cloud. Further, we plan to understand resource management for dynamic function-chains in serverless platforms.

Acknowledgements

I am grateful to my advisors Dr. Chita Das and Dr. Mahmut Kandemir for their continuous guidance and support throughout my PhD. This research was supported in part by NSF grants #1931531, #1955815, #1908793 and NSF Chameleon Cloud project CH819640.

References

[1] 2020. Alibaba Cloud. https://alibaba.com/cloud.

[2] 2020. Brigade-workflows. https://brigade.sh/.

[3] 2020. Cloud Computing Market Projected To Reach $411B By 2020. https://https://www.gartner.com/en/newsroom/press-releases/2019-11-13-gartner-forecasts-worldwide-public-cloud-revenue-to-grow-17-percent-in-2020.

[4] 2020. Flexera 2020 State of the Cloud Report. https://info.flexera.com/SLO-CM-REPORT-State-of-the-Cloud-2020.

[5] Amazon. 2020. EC2 pricing. https://aws.amazon.com/ec2/pricing/.

[6] Microsoft Azure. 2020. Serverless Functions. https://azure.microsoft.com/en-us/services/functions/.

[7] Ataollah Fatahi Baarzi, Timothy Zhu, and Bhuvan Urgaonkar. 2019. BurScale: Using Burstable Instances for Cost-Effective Autoscaling in the Public Cloud. In *Proceedings of the ACM Symposium on Cloud Computing*. Association for Computing Machinery, New York, NY, USA.

[8] William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. 2018. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 9368–9377.

[9] Andrew Chung, Jun Woo Park, and Gregory R. Ganger. 2018. Stratus: Cost-aware Container Scheduling in the Public Cloud. In *SoCC*.

[10] Amazon Elastic Compute Cloud. 2011. Amazon web services. *Retrieved November* (2011).

[11] Daniel Crankshaw, Peter Bailis, Joseph E Gonzalez, Haoyuan Li, Zhao Zhang, Michael J Franklin, Ali Ghodsi, and Michael I Jordan. 2014. The missing piece in complex analytics: Low latency, scalable model management and serving with velox. *arXiv preprint arXiv:1409.3809* (2014).

[12] Daniel Crankshaw, Gur-Eyal Sela, Corey Zumar, Xiangxi Mo, Joseph E. Gonzalez, Ion Stoica, and Alexey Tumanov. 2018. InferLine: ML Inference Pipeline Composition Framework. *CoRR* abs/1812.01776 (2018). arXiv:1812.01776 http://arxiv.org/abs/1812.01776

[13] Daniel Crankshaw, Xin Wang, Guilio Zhou, Michael J. Franklin, Joseph E. Gonzalez, and Ion Stoica. 2017. Clipper: A Low-Latency Online Prediction Serving System. In *14th USENIX Symposium on Networked Systems Design and Implementation (NSDI 17)*. USENIX Association, Boston, MA, 613–627. https://www.usenix.org/conference/nsdi17/technical-sessions/presentation/crankshaw

[14] Pamela Delgado, Diego Didona, Florin Dinu, and Willy Zwaenepoel. 2016. Job-aware Scheduling in Eagle: Divide and Stick to Your Probes. In *Proceedings of the Seventh ACM Symposium on Cloud Computing*.

[15] Pamela Delgado, Florin Dinu, Anne-Marie Kermarrec, and Willy Zwaenepoel. 2015. Hawk: Hybrid datacenter scheduling. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 499–510.

[16] Christina Delimitrou, Daniel Sanchez, and Christos Kozyrakis. 2015. Tarcil: Reconciling Scheduling Speed and Quality in Large Shared Clusters. In *Proceedings of the Sixth ACM Symposium on Cloud Computing* (Kohala Coast, Hawaii) *(SoCC '15)*. ACM, New York, NY, USA.

[17] Sina Esfandiarpoor, Ali Pahlavan, and Maziar Goudarzi. 2015. Structure-aware online virtual machine consolidation for datacenter energy improvement in cloud computing. *Computers & Electrical Engineering* 42 (2015), 74–89.

[18] Google. 2020. Cloud Functions. https://cloud.google.com/functions/docs/,February2018.

[19] Arpan Gujarati, Sameh Elnikety, Yuxiong He, Kathryn S. McKinley, and Björn B. Brandenburg. 2017. Swayam: Distributed Autoscaling to Meet SLAs of Machine Learning Inference Services with Resource Efficiency. In *USENIX Middleware Conference*.

[20] Jashwant Raj Gunasekaran, Prashanth Thinakaran, Nachiappan C.Nachiappan, Mahmut Taylan Kandemir, and Chita R. Das. 2020. Fifer: Tackling Resource Underutilization in the Serverless Era. In *USENIX Middleware Conference*.

[21] Jashwant Raj Gunasekaran, Prashanth Thinakaran, Mahmut Taylan Kandemir, Bhuvan Urgaonkar, George Kesidis, and Chita Das. 2019. Spock: Exploiting serverless functions for slo and cost aware resource procurement in public cloud. In *IEEE CLOUD*.

[22] Jashwant Raj Gunasekaran, Prashanth Thinakaran, Cyan Subhra Mishra, Mahmut Taylan Kandemir, and Chita R. Das. 2020. Towards Designing a Self-Managed Machine Learning Inference Serving System inPublic Cloud. arXiv:2008.09491 [cs.DC]

[23] Rui Han, Moustafa M. Ghanem, Li Guo, Yike Guo, and Michelle Osmond. 2014. Enabling Cost-Aware and Adaptive Elasticity of Multi-Tier Cloud Applications. *Future Gener. Comput. Syst.* 32, C (March 2014), 82–98.

[24] Aaron Harlap, Andrew Chung, Alexey Tumanov, Gregory R. Ganger, and Phillip B. Gibbons. 2018. Tributary: spot-dancing for elastic services with latency SLOs. In *ATC*.

[25] Aaron Harlap, Alexey Tumanov, Andrew Chung, Gregory R. Ganger, and Phillip B. Gibbons. 2017. Proteus: Agile ML Elasticity Through Tiered Reliability in Dynamic Resource Markets. In *Eurosys*.

[26] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* (1997).

[27] Konstantinos Karanasos, Sriram Rao, Carlo Curino, Chris Douglas, Kishore Chaliparambil, Giovanni Matteo Fumarola, Solom Heddaya, Raghu Ramakrishnan, and Sarvesh Sakalanaga. 2015. Mercury: Hybrid centralized and distributed scheduling in large shared clusters. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*. 485–497.

[28] Adithya Kumar, Iyswarya Narayanan, Timothy Zhu, and Anand Sivasubramaniam. 2020. The Fast and The Frugal: Tail Latency Aware Provisioning for Coping with Load Variations. In *Proceedings of The Web Conference 2020 (WWW '20)*. Association for Computing Machinery, New York, NY, USA.

[29] Yunseong Lee, Alberto Scolari, Byung-Gon Chun, Marco Domenico Santambrogio, Markus Weimer, and Matteo Interlandi. 2018. PRETZEL: Opening the Black Box of Machine Learning Prediction Serving Systems. In *13th USENIX Symposium on Operating Systems Design and Implementation (OSDI 18)*. USENIX Association, Carlsbad, CA, 611–626. https://www.usenix.org/conference/osdi18/presentation/lee

[30] I. Narayanan, D. Wang, A. Mamun, A. Sivasubramaniam, H. K. Fathy, and S. James. 2017. Evaluating energy storage for a multitude of uses in the datacenter. In *2017 IEEE International Symposium on Workload Characterization (IISWC)*. 12–21. https://doi.org/10.1109/IISWC.2017.8167752

[31] Kay Ousterhout, Patrick Wendell, Matei Zaharia, and Ion Stoica. 2013. Sparrow: distributed, low latency scheduling. In *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*. ACM, 69–84.

[32] Khalid Rafique, Abdul Wahid Tareen, Muhammad Saeed, Jingzhu Wu, and Shahryar Shafique Qureshi. 2011. Cloud computing economics opportunities and challenges. In *Broadband Network and Multimedia Technology (IC-BNMT), 2011 4th IEEE International Conference on*. IEEE, 401–406.

[33] Mohammad Shahrad, Rodrigo Fonseca, Inigo 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 (USENIX ATC 20)*. USENIX Association, 205–218. https://www.usenix.org/conference/atc20/presentation/shahrad

[34] Prateek Sharma, David Irwin, and Prashant Shenoy. 2017. Portfolio-Driven Resource Management for Transient Cloud Servers. *Proc. ACM Meas. Anal. Comput. Syst.* 1, 1, Article 5 (June 2017), 23 pages.

[35] Prateek Sharma, Stephen Lee, Tian Guo, David Irwin, and Prashant Shenoy. 2015. Spotcheck: Designing a derivative iaas cloud on the

spot market. In *Proceedings of the Tenth European Conference on Computer Systems*. 1–15.

[36] Arjun Singhvi, Kevin Houck, Arjun Balasubramanian, Mohammed Danish Shaikh, Shivaram Venkataraman, and Aditya Akella. 2019. Archipelago: A Scalable Low-Latency Serverless Platform. *arXiv preprint arXiv:1911.09849* (2019).

[37] P. Thinakaran, J. R. Gunasekaran, B. Sharma, M. T. Kandemir, and C. R. Das. 2019. Kube-Knots: Resource Harvesting through Dynamic Container Orchestration in GPU-based Datacenters. In *CLUSTER*.

[38] Bhuvan Urgaonkar, Prashant Shenoy, Abhishek Chandra, Pawan Goyal, and Timothy Wood. 2008. Agile Dynamic Provisioning of Multi-tier Internet Applications. *TAAS* (2008).

[39] Cheng Wang, Bhuvan Urgaonkar, Neda Nasiriani, and George Kesidis. 2017. Using Burstable Instances in the Public Cloud: Why, When and How? *SIGMETRICS* (June 2017).

[40] Neeraja J. Yadwadkar, Francisco Romero, Qian Li, and Christos Kozyrakis. 2019. A Case for Managed and Model-Less Inference Serving. In *Proceedings of the Workshop on Hot Topics in Operating Systems*. Association for Computing Machinery, New York, NY, USA. https://doi.org/10.1145/3317550.3321443

[41] Jeong-Min Yun, Yuxiong He, Sameh Elnikety, and Shaolei Ren. 2015. Optimal aggregation policy for reducing tail latency of web search. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 63–72.

[42] Chengliang Zhang, Minchen Yu, Wei Wang, and Feng Yan. 2019. MArk: Exploiting Cloud Services for Cost-Effective, SLO-Aware Machine Learning Inference Serving. In *ATC*.