Work Load Imbalance Analysis on Alibaba Cluster Trace

Xiaomin Li
Texas State University
x_l30@txstate.edu

ABSTRACT

In the past decades, we have witnessed the rapid increase of data center capacity. With the booming of e-commerce, cloud services, modern data center need to efficiently handle and respond to a huge volume of data traffic. Meanwhile, it is also critical for data center owner to reduce the total cost for profitability. Improving system utilization is one of the strategies to reach a win-win situation. This project analyzes the real-world cluster trace released by Alibaba. Though some researches confirmed the success of Alibaba's mixed scheduling of time-sensitive online service jobs and time-insensitive batch jobs, which greatly improves both CPU and DRAM utilization. However, we can still notice some workload imbalance exists in the system. Such imbalances exacerbate the complexity and challenge of cloud resource management, which might incur severe wastes of resources and low cluster utilization. In this project, I mainly focus on analyzing the workload imbalance exist in the system. They are 1) Machine resource utilization imbalance 2) Imbalanced resource demand 3) Resource over-subscription 4) Long running batch jobs starvation.

CCS CONCEPTS

• **General and reference → Cross computing tools and techniques**.

KEYWORDS

Alibaba Trace, Imbalance, Container, Cloud Computing, Resource Efficiency

ACM Reference Format:
Xiaomin Li. 2019. Work Load Imbalance Analysis on Alibaba Cluster Trace. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Driven by the enormous growth of e-commerce and cloud service, the company like Amazon and Alibaba have become newly IT giants. According to the Netcraft report [2], Alibaba company has become the second largest cloud computing company in the world. The data centers of Alibaba process billions of transactions on a daily basis. For example, Alibaba online shopping platform sales over $25 billion of products in the 2017 Single's Day Shopping Festival. [3] The last year sealing amount was about $30.8 billion within 24 hours of the festival. [4] These tremendous amount of online transactions require powerful support of Alibaba datacenter.

The data center can handle these transaction are impressive but the investments to maintain the data center are also astonishing. One of the Alibaba's newest data centers in Hebei requires ¥18 billion capital investment plus millions of dollars annual electricity bills. This number could further grow as the number of servers, the consumed energy, the labor to manage the system, and the electricity price all increase. McKinsey reported that the cost of data centers accounts for approximately 25% of the total corporate IT budget. [18] Inefficient data centers may threaten profitability, despite the large number of users and dominating market share[18]. One of the primary reasons for data center inefficiency is the low utilization. The Gartner and McKinsey report indicated that the server utilization rate is merely 6% - 12% [18]for most enterprise data centers and another study showed that the utilization of Amazon AWS servers is not high either (7% - 17%)[11]. To address the inefficiency issue, virtualization and server consolidation have been widely used to boost utilization rate. However, co-running jobs, especially time-sensitive jobs, on the same server may interfere with each other and adversely affect performance. A single request failure (even delay) can be disastrous to user experiences and result in huge revenue loss. Alibaba has explored an innovative approach to alleviate this dilemma by scheduling time-sensitive online service jobs and time-insensitive batch jobs to the same machine [1] [24]In September 2017, Alibaba released a cluster trace file that uses the mixed scheduling strategy, which consists of 12,932 batch jobs and 11,076 online service jobs running on 1,313 machines over a 12-hour period[1]. Previous research explored the runtime status of the hybrid cluster, and showed several important insights about imbalance in the cloud.[14] [7] [12] Such imbalances exacerbate the complexity and challenge of cloud resource management.[15] In this project, I re-analyzed the trace data and found the following workload imbalance in the cluster. They are:

- Machine resource utilization imbalance : heterogeneous resource utilization across machines and workloads
- Imbalanced resource demands : greatly time-varying resource usage per machine
- Recourse over-subscription : per instance requested resource than needed
- Batch job starvation: long term batch jobs delayed response

2 RELATED WORK

2.1 Cloud Computing

Cloud computing nowadays is becoming more and more popular in industries for its convenient to use and cost-efficiency for end users. It is defined as a model for enabling ubiquitous, convenient,

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction. [16] The cluster machine scheduler strategies we talked about here are in the cloud computing category.

2.2 Cluster trace studies

In 2011, Google open-sourced the first publicly available cluster trace data[5] spanning several clusters. Reiss et al.[17] study the heterogeneity and dynamicity properties of the Google workloads. Other works[9][13][23] focus their studies on different aspects of the Google trace. Alibaba, the largest cloud service provider in China, released their cluster trace[1] in late 2017. Different from the Google trace, the Alibaba trace contains information about the two co-located container and batch job work-loads, facilitating better understanding of their interactions and interferences. Lu et al.[14] perform characterization of the Alibaba trace to reveal basic workload statistics. Our study is focused on providing a unique and microscopic view about how the co-located workloads interact and impact each other.

2.3 Cluster Management systems

A series of state-of-the-art cluster management systems (CMSs) support co-located long-running and batch services. Monolithic CMSs such as Borg [20](and its open-sourced implementation Kubernetes[10]) and Quasar[8] use a centralized resource scheduler for performing resource allocation and management. Two-level CMSs such as Mesos[8] adopts a hierarchical structure where a Level-0 resource manager is used to jointly coordinate multiple Level-1 application-specific schedulers. Other CMSs achieve low-latency scheduling by using approaches such as shared-state parallel schedulers[19].

3 ALIBABA SCHEDULER POLICIES

Alibaba's CMS resembles the architecture of a two-level CMS, as shown in Figure 1. The logical architecture consists of three component: (1) a global Level-0 controller. (2) a Sigma [7] scheduler that manages the long- running workload, and (3) a Fuxi[25] scheduler that manages the batch workloads. Sigma is responsible for scheduling online service containers[6] for the production jobs. These online services are user-facing, and typically require low latency and high performance. Sigma has been used for large-scale container deployment purposes by Alibaba for several years. It has also been used during the Alibaba Double 11 Global Shopping Festival. Fuxi, on the other hand, manages non-containerized non-production batch jobs. Fuxi is used for vast amounts of data processing and complex large-scale computing type applications. Fuxi employs data-driven multi-level pipelined parallel computing framework, which is compatible with MapReduce, Map-Reduce-Merge, and other batch programming modes. The Level-0 mechanism coordinates and manages both types of workloads (Sigma and Fuxi) for coordinated global operations. Particularly the Level-0 controller performs four important functionalities. (1) manages colocation clusters, (2) performs resource matching between each scheduling tenant, (3) strategies for everyday use and use during large-scale promotions, and (4) performs exception detection and processing.

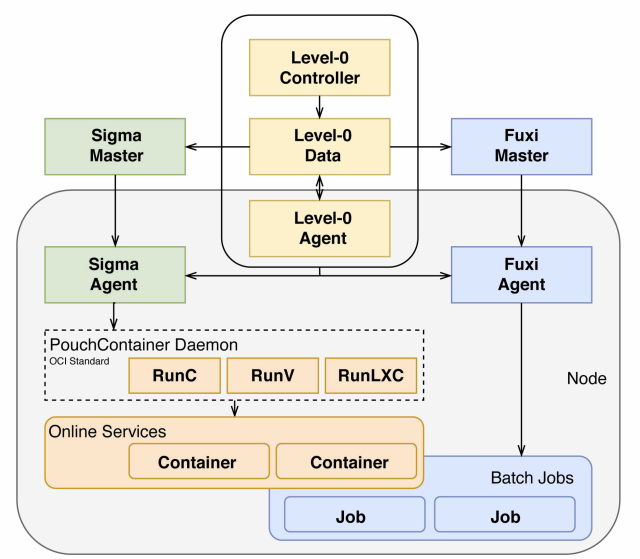

Figure 1: Alibaba co-locating scheduler. Each server machine has an instance of Sigma agent and Fuxi agent which coordinate with the Level-0 management component to enable Sigma and Fuxi schedulers to work together.

Different types of workloads were running on separate clusters before 2015, since when Alibaba has been making effort to co-locate them on shared clusters.

4 ALIBABA CLUSTER DATASET

Alibaba released a new dataset Cluster Data 2017 in September 2017, which contains a production cluster run-time information during 12 hours period, and includes 1.3k machines that run both online service and offline batch jobs[1] The data is motivated to address the low utilization and resource inefficiency challenges of Alibaba cluster when co-locating online services and batch jobs.

There are three types of data in the trace: machine utilization and runtime information of both batch and online service workloads. For confidentiality reasons, portion information in the trace is obfuscated.

Machine utilization is described as two tables: the "machine events" table and the "machine resource utilization" table. Capacities reflect the normalized multi-dimension physical capacity per machine. Each dimension (CPU cores, RAM size) is normalized independently.

Batch workloads are described as two tables: "instance" table and "task" table. The user submits a batch workload in the form of Job (which is not included in the trace). Each job consists of multiple tasks, each forming a DAG according to the data dependency. They are consisting of multiple instances and executing different computing logics. Instance is the smallest scheduling unit of batch workload. All instances within a task execute exactly the same binary code with identical multi-resource demands, but processing different portion of data.

Table 1: Statistics of Batch jobs

Status	Number
Total instances	15186017
Total tasks	76986
Total jobs	12951
1 in 40,000	Unexplained usage
Average instance number per task	152
Maximum instance number per task	64486
Minimum instance number per task	1
Average task number per job	6
Maximum task number per job	156
Minimum task number per job	1
Average instance duration	129 seconds
Maximum instance duration	29558 seconds
Mimimum instance duration	< 1 second
Average task duration	192 seconds
Maximum task duration	29585 seconds
Mimimum task duration	< 1 second

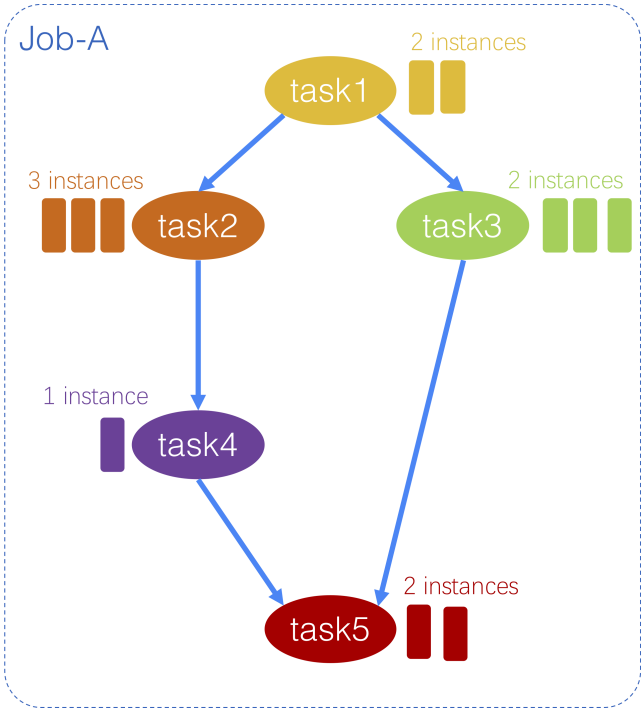

Figure 2: Batch job DAG

Online service jobs are described by two tables: "service instance event" and "service instance usage". The trace includes only two types of instance events. One event for creation, and another for finish. Event of creation records the start time of a service instance, and event of remove indicates the finish of an service instance. Each instance is the smallest scheduling unit and running in a light-weight virtual machine of Linux container (LXC). It could also be regarded as a complete service job. Either instances of batch or service workloads express their resource demands in the form of reservation. And their cluster manager of Fuxi leverages admission-control strategy for resource allocation. The combination of above two mechanisms is regarded to be the essential cause of low cluster utilization and resource inefficiency in recent studies[22][21][8]. In the following sections, I introduced several imbalanced phenomenons found from Alibaba cluter trace.

5 ALIBABA CLUSTER WORKLOADS

In the trace, workloads are classified into two categories. One is long-term service job, another is short-term batch job.

Each long-term service instance belongs to one job, and is running within a Linux container for 12 hours. One short-term batch job has multiple tasks and each task is made up of multiple instances. Each task or instance is running for seconds or minutes. A batch workload is in the form of a job, each job can be viewed as a DAG (Figure 2). Tasks form a DAG according to the data dependency can not run in parallel and all the instances within a task execute the same instruction but different data chunk.

The detail run-time statistics of batch jobs are shown in Table 1. There are totally 11089 service instances(jobs) running for the whole 12 hours.

6 IMBALANCE ANALYSIS

6.1 Machine resource utilization imbalance

Figure 3 plots the resource utilization information per machine. Stand from the perspective of cluster machines' workloads, it summarized the average, minimum and maximum utilization of CPU, memory and disk among 1313 machines at each sampling time. In the data trace, the sampling time is 300 seconds (every 5 minutes). In the figure, all the usage are normalized as 1. The green lines represent the maximum usage, the blue lines represent the average and the orange lines mean the minimum utilization.

The average CPU utilization per machine is around 40% in the first half of the time period and about 30% in the second half. The maximum CPU utilization is 20% higher than the average except high spikes appeared at 8 - 10pm, which is reaching over 90% whereas the average CPU utilization maintain stable. The maximum memory usage is about 90% and the average is around 60%. For the disk usage, it is quite stable than the other two measurements. The peak value is over 90% and average is under 40% and there were two fluctuation at 2pm and 3pm when the value went down. And we can also observe that within the whole time period, some machines in the cluster are idle at each time point and we can see from the Figure 3 that the minimum values are all around 0. By comparing these huge gaps between minimum, average and maximum usages of machines, we can observed tremendous machine resource utilization imbalance in cluster. It demonstrated that cloud data centers need new schedulers to balance the load and avoid hot spot of machine utilization, so as to improve cluster efficiency.

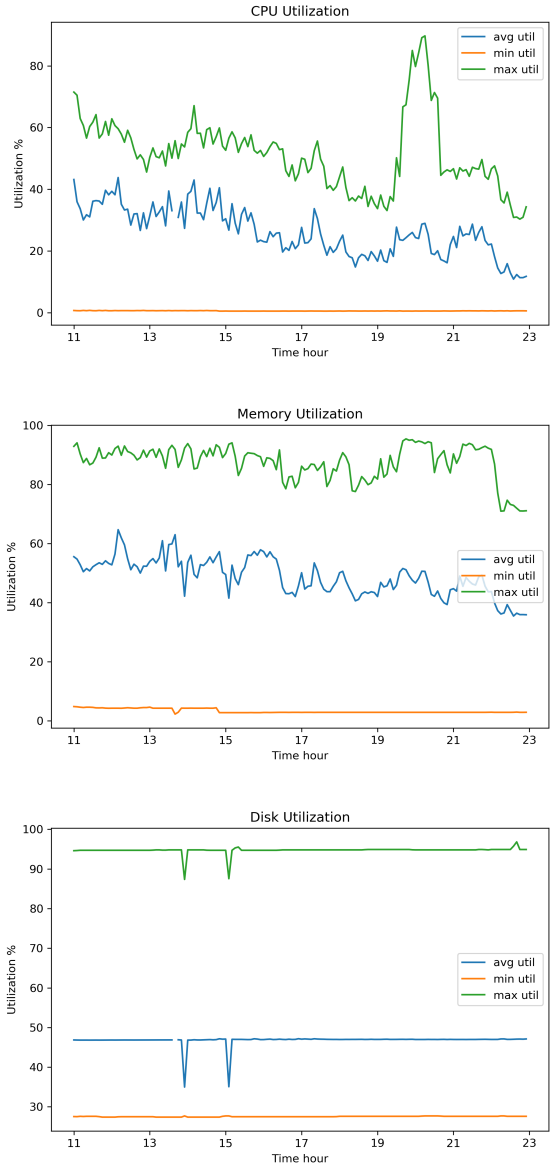

Figure 3: The CPU, Memory and Disk utilization of machines during 12 hour time period. The green line indicates the maximum utilization of all machines in the cluster, the blue lines indicate the average utilization and orange lines means the minimum utilization of all machines.

6.2 Imbalanced resource demands

I calculate the resource(CPU cores and memory percentage) amount requested and reserved by long-term service jobs per machine, Figure 4. From the figure, we can observe that for most of the machine-level CPU allocation are equal to 64, which is the total number of CPU cores of one machine. And for the memory allocation, each machine has been requested about 80% - 90% of the total memory. However, we can still find a few of the machine been overbooked

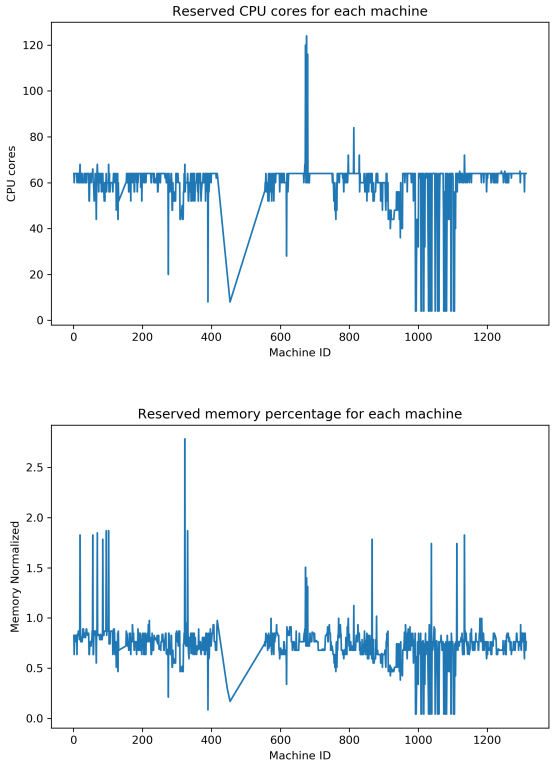

Figure 4: Distribution of reserved resources at container creation time across the cluster

and some are under burden which cause the imbalanced resource allocation on cluster machine. (The weird pattern showed from machines which ID is 1000 - 1100 are caused by data missing. For the confidential issues, the Alibaba datacenter erased some data here before releasing them.)

6.3 Resource over-subscription

To better understand the observed resource usage imbalance, I compared service jobs reserved CPU and memory capacity with the actual usage, as shown in Figure 5 and Figure 6. We can observe that: (1) Most of the service jobs request 4-6 or 8-10 CPU cores. (2) Almost all the service jobs request memory usage under 20%. (3) Most of the service jobs actually only used under 20% of CPU resource compare to allocated. (4) Average memory usage is under 50%. From the listed observation above, we can say that the service jobs resource over-subscription or over-provisioning are commonly existed in Alibaba cluster. Because service jobs hold resource once it allocated, the resource over-subscription will cause some long running batch jobs can not get enough resource which probably leads to the bottleneck of improving cluster throughput.

6.4 Batch job starvation

I used the last task finished timestamp subtract the first task creation timestamp for each batch job to calculate the execution time. The

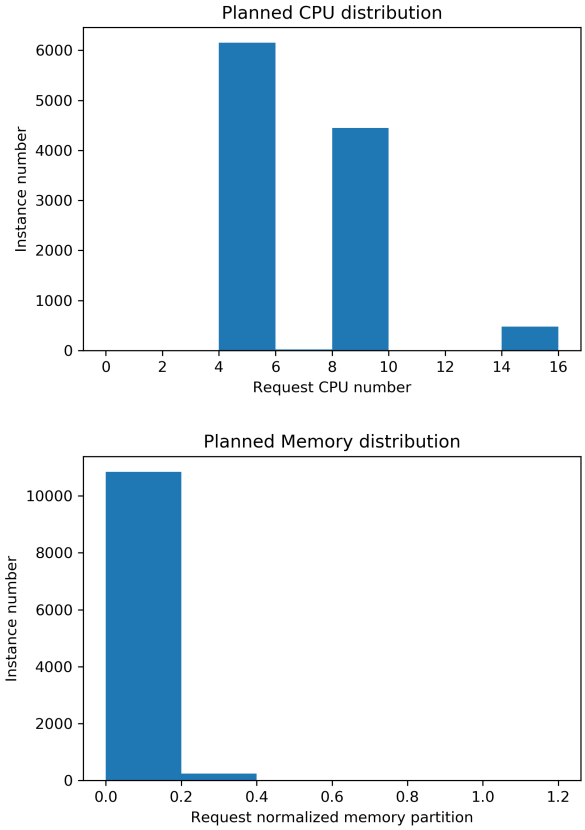

Figure 5: Service jobs request CPU cores and memory percentage

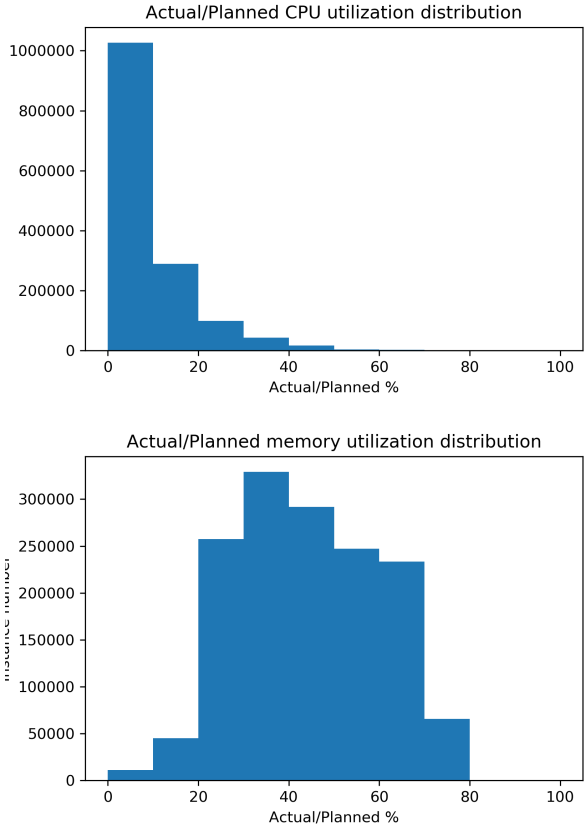

Figure 6: Actually used percentage of CPU and memory of service jobs

longest batch job executed 12950 seconds, the shortest finished within 1 second and the average execution time of all batch jobs is 6475.5 second. Short jobs overwhelmingly occupied the cluster which will cause long running batch jobs wait too long to finish execution. (Figure 7)

7 CONCLUSION

In this project, I analyzed Alibaba cluter batch jobs and service jobs workload and illustrated four different types of imbalance found in trace data. Due to such resource reservation mechanism and imbalanced phenomenon observed here, we can draw the conclusion that Alibaba's co-locating scheduler is ineffective to improve cluster efficiency. A better scheduler is needed to solve the dilemma of the quality of service and efficiency of data center.

Figure 7: Execution time of all batch jobs, x-axis represents the job id, y-axis represent execution time

REFERENCES

[1] [n. d.]. *Alibaba cluster data.* https://github.com/alibaba/clusterdata.
[2] [n. d.]. *Cloud wars alibaba.* https://news.netcraft.com/archives/2017/08/22/cloud-wars-alibaba-becomes-2nd-largest-hosting-company.html.
[3] [n. d.]. *Cloud wars alibaba.* https://www.businessinsider.com/alibaba-singles-day-total-sales-2017-2017-11.
[4] [n. d.]. *Cloud wars alibaba.* https://www.cnbc.com/2018/11/11/alibaba-singles-day-2018-record-sales-on-largest-shopping-event-day.html.
[5] [n. d.]. *Google cluster workload traces.* https://github.com/google/cluster-data.
[6] [n. d.]. *Pouch - An Efficient Enterprise-class Rich Container Engine.* https://github.com/alibaba/pouch.
[7] Yue Cheng, Zheng Chai, and Ali Anwar. 2018. Characterizing co-located datacenter workloads: An Alibaba case study. *arXiv preprint arXiv:1808.02919* (2018).
[8] Christina Delimitrou and Christos Kozyrakis. 2014. Quasar: resource-efficient and QoS-aware cluster management. In *ACM SIGARCH Computer Architecture*

News, Vol. 42. ACM, 127–144.

[9] Sheng Di, Derrick Kondo, and Walfredo Cirne. 2012. Characterization and comparison of cloud versus grid workloads. In *2012 IEEE International Conference on Cluster Computing*. IEEE, 230–238.

[10] Inc Kubernetes. [n. d.]. Kubernetes-production-grade container orchestration.

[11] Huan Liu. 2011. A measurement study of server utilization in public clouds. In *2011 IEEE Ninth International Conference on Dependable, Autonomic and Secure Computing*. IEEE, 435–442.

[12] 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*. ACM, 347–360.

[13] Zitao Liu and Sangyeun Cho. 2012. Characterizing machines and workloads on a Google cluster. In *2012 41st International Conference on Parallel Processing Workshops*. IEEE, 397–403.

[14] C. Lu, K. Ye, G. Xu, C. Xu, and T. Bai. 2017. Imbalance in the cloud: An analysis on Alibaba cluster trace. In *2017 IEEE International Conference on Big Data (Big Data)*. 2884–2892. https://doi.org/10.1109/BigData.2017.8258257

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

[16] Peter Mell, Tim Grance, et al. 2011. The NIST definition of cloud computing. (2011).

[17] Rajesh Nishtala, Hans Fugal, Steven Grimm, Marc Kwiatkowski, Herman Lee, Harry C Li, Ryan McElroy, Mike Paleczny, Daniel Peek, Paul Saab, et al. 2013. Scaling memcache at facebook. In *Presented as part of the 10th {USENIX} Symposium on Networked Systems Design and Implementation ({NSDI} 13)*. 385–398.

[18] Revolutionizing Data Center Energy Efficiency 2018. *McKinsey Report*. Revolutionizing Data Center Energy Efficiency.

[19] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, and John Wilkes. 2013. Omega: flexible, scalable schedulers for large compute clusters. (2013).

[20] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, and John Wilkes. 2015. Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 18.

[21] Guoyao Xu and Cheng-Zhong Xu. 2017. Prometheus: Online estimation of optimal memory demands for workers in in-memory distributed computation. In *Proceedings of the 2017 Symposium on Cloud Computing*. ACM, 655–655.

[22] Guoyao Xu, Cheng-Zhong Xu, and Song Jiang. 2016. Prophet: Scheduling executors with time-varying resource demands on data-parallel computation frameworks. In *2016 IEEE International Conference on Autonomic Computing (ICAC)*. IEEE, 45–54.

[23] Qi Zhang, Mohamed Faten Zhani, Shuo Zhang, Quanyan Zhu, Raouf Boutaba, and Joseph L Hellerstein. 2012. Dynamic energy-aware capacity provisioning for cloud computing environments. In *Proceedings of the 9th international conference on Autonomic computing*. ACM, 145–154.

[24] 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. *Proceedings of the VLDB Endowment* 7, 13 (2014), 1393–1404.

[25] 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. *Proc. VLDB Endow.* 7, 13 (Aug. 2014), 1393–1404. https://doi.org/10.14778/2733004.2733012