Imbalance in the Cloud: an Analysis on Alibaba Cluster Trace

Chengzhi Lu[1], Kejiang Ye[1,*], Guoyao Xu[1,2], Cheng-Zhong Xu[1], Tongxin Bai[1]
[1]*Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences*
[2]*Wayne State University*
{cz.lu, kj.ye, gy.xu, tx.bai, cz.xu}@siat.ac.cn

***Abstract*—To improve resource efficiency and design intelligent scheduler for clouds, it is necessary to understand the workload characteristics and machine utilization in large-scale cloud data centers. In this paper, we perform a deep analysis on a newly released trace dataset by Alibaba in September 2017, consists of detail statistics of 11089 online service jobs and 12951 batch jobs co-locating on 1300 machines over 12 hours. To the best of our knowledge, this is one of the first work to analyze the Alibaba public trace. Our analysis reveals several important insights about different types of *imbalance* in the Alibaba cloud. Such imbalances exacerbate the complexity and challenge of cloud resource management, which might incur severe wastes of resources and low cluster utilization. 1) Spatial Imbalance: heterogeneous resource utilization across machines and workloads. 2) Temporal Imbalance: greatly time-varying resource usages per workload and machine. 3) Imbalanced proportion of multi-dimensional resources (CPU and memory) utilization per workload. 4) Imbalanced resource demands and runtime statistics (duration and task number) between online service and offline batch jobs. We argue accommodating such imbalances during resource allocation is critical to improve cluster efficiency, and will motivate the emergence of new resource managers and schedulers.**

***Keywords*-Alibaba Trace, Imbalance, Container, Cloud Computing, Resource Efficiency,**

I. Introduction

Cloud datacenters usually comprise thousands of machines, providing highly reliable, efficient and scalable services. Examples of typical cloud services including web search, e-commerce systems, and social networks. With the increasing popularity of cloud and data center computation, users tend to share large hardware platforms. However, effective resource management is very important to guarantee both quality of service and high resource utilization [1][2][3][4][5][6][7][8].

Recent studies [9][10][11][12][13] revealed that most cloud facilities and commercial clusters are operating at low utilization. According to the data of Geithner and McKinsey several years ago, the global server utilization seems to be very low, which is only 6% to 12%. Even leveraging virtualization technology, the utilization is still below 17%. It probably incurs low cost-efficiency, energy-proportional and scalability challenges of clouds.

*Corresponding author

Co-locating online service and offline batch jobs on the same cluster is shown to be an efficient approach to improve cluster utilization in modern cloud data centers [14][15][2]. However, we find that Alibaba cluster reserved fix amounts of resources for online services rather than flexible allocations. Under such reservation mechanism, traditional co-locating strategy is ineffective because batch jobs could not leverage reserved idle resources of service jobs. Additionally, contention and interference on shared resources can cause latency spikes that violate the service-level objectives of service jobs. Ensuring quality of service (QoS) for latency-sensitive job is non-trivial in such environment.

By understanding the workload characteristics and machine utilization in large-scale cloud data centers, we could provide predictable knowledges to cluster manager. Through planning ahead and performing intelligent scheduling, we could improve resource efficiency and avoid such interferences.

In this paper, we perform a deep analysis on a newly released trace dataset by Alibaba in September 2017, covering 1300 servers over 12 hours [16]. Alibaba Cloud is one of the largest public cloud platforms in the world, processing millions of tasks across hundreds of data centers everyday. This trace includes runtime statistics of a hybrid cluster, on which online service and offline batch jobs are co-located.

To the best of our knowledge, this is one of the first work to analyze the public Alibaba trace. We explored runtime status of the hybrid cluster, and showed several important insights about **imbalance** in the cloud. Such imbalances exacerbate the complexity and challenge of cloud resource management. It includes:

- *Spatial Imbalance: heterogeneous resource utilization across machines and workloads.*
- *Temporal Imbalance: greatly time-varying resource usages per workload and machine.*
- *Imbalanced proportion of multi-dimensional resources (CPU and memory) utilization per workload.*
- *Imbalanced resource demands and runtime statistics (duration and task number) between online service and offline batch jobs.*

Many modern resource managers are designed under the assumption of ideal cluster environment. The commonly occurred imbalance phenomenons we found in Alibaba trace

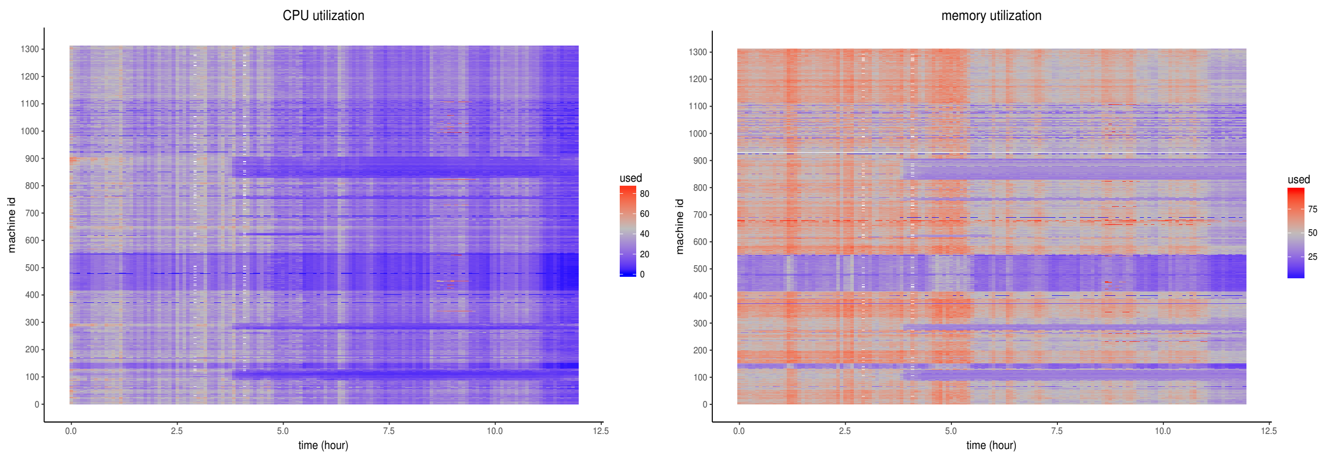

Figure 1. The heat maps of CPU and memory utilization of machines in the cluster. The white portion indicates the lack of data in the trace. Red color indicates high utilization while blue color indicates low utilization.

would lead to significant resource inefficiency and wastes. We argue it is critical to accommodate such imbalances during resource allocation to improve cluster efficiency. They will also motivate the emergences of new resource managers and schedulers.

II. The Dataset

Alibaba released a new dataset ClusterData201708 in September 2017, which contains a production cluster runtime information during 12 hours period, and includes 1.3k machines that run both *online service* and *offline batch jobs* [16]. 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 portions of data.

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 startime 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 lightweight 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, which is commonly used in modern resource managers [2][6][4][3][7][8]. And their cluster manager of Fuxi [8] 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 [5][13][17]. In the following sections, we introduced several imbalanced phenomenons in Alibaba cloud.

III. Imbalances of Machines

Figure 1 plots the resource utilization per machine in the cluster during 12 hours. The trace provided normalized CPU and memory usages information per sampling time for each machine. All the data are retrieved from “machine events” and “machine resource utilization” table.

We had an interesting observation that CPU utilizations of portion of machines (id from 400 to 600 and 900 to 1100) are always higher than others while their memory utilizations are relatively lower. And CPU utilizations of most machines are gradually increasing during cluster running while memory utilizations are decreasing. Thus we could always observe the highest CPU utilization and lowest memory utilization of machines at the end of trace period (from 11 to 12 hour). In contrast, CPU is always idle at the beginning (from 0 to 3.5 hour) while memory keeps high load.

It demonstrated that there exists significant **spatial imbalance** (heterogeneous resource utilization across machines)

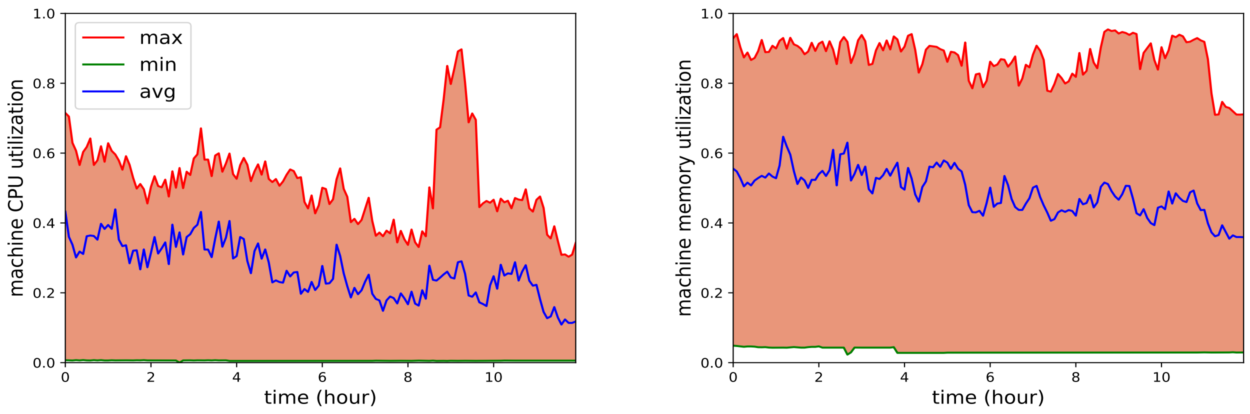

Figure 2. The CPU and memory utilization of machines during execution. The red line indicates the maximum utilization of all machines in the cluster, the blue one indicates the average utilization and green one means minimum utilization of all machines.

and **temporal imbalance** (time-varying resource usages per machine) of utilization for machines in cloud data center.

From Figure 2, we saw more fine-grained information of resource usages per machine. We summarized average, minimum and maximum utilization among 1300 machines at each sampling time. Both the CPU and memory usages are normalized

The average CPU utilization per machine is within 40% and maximum maintains about 60% along the sampling period. Average memory utilization per machine is within 60% and maximum about 90%. The green line plots utilizations of the machine whose utilization is the minimum among all machines per sampling time. Both CPU and memory utilization of such minimum usages are nearing zero. From hour 8 to 10, the maximum CPU utilization rapidly spikes, reaching over 90%, while the average CPU usages maintain stable. By comparing these huge gaps between minimum, average and maximum usages of machines, we observed tremendous spatial imbalance of utilization 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. Differ from CPU usages, memory usages maintain steady during that period. It also indicates **the proportion of multi-dimensional resources utilization (CPU and memory) of workloads is imbalanced**.

Additionally, we observed severe wastes and **resource inefficiency** of CPU and memory resources in cluster. However, due to relatively low maximum usages of machines, CPU utilization has the opportunity to be greatly improved through comprehensively understanding workloads' resource demands and making proper reservations. Nevertheless, improving memory utilization is challenging since job performance is sensitive to the relatively high maximum usages of machines. Simply decreasing the reservations to improve cluster memory efficiency might lead to serious performance degradation due to thrashing. Recent study [13] proposed one solution by making better demands estimations. We argue the cloud data center needs new resource managers and schedulers to improve cluster resource efficiency by avoiding above imbalanced and low utilization.

IV. Imbalances of Workloads

In the trace, workloads are classified into two categories. One is long-term service job, another is short-term batch job. Each service instance belongs to one job, and is running within a Linux container for 12 hours. While each instance of batch jobs belongs to one task, and is running for seconds or minutes. Multiple tasks compose of one batch job. Detail runtime statistics of batch workloads are shown in Table I.

Each job commonly has several tasks, but the maximum one has 156. There are three types of status for batch tasks, including normally terminated, failed and waiting due to preemption. Most tasks are normally terminated, while over 2000 are waiting. The majority of tasks own hundreds of instances, while some has an extremely large number of 64486. The corresponding average durations of instances and tasks are 129 and 192 seconds respectively. The maximum durations are 29558 and 29585 seconds, while both of the minimum durations are less than 1 second. By diving into the task execution information, we found the longest task that ran over 8 hours was consisting of several longest instances that were executing at the same time. Thus their maximum durations are similar.

In contrast, each service job consists of only one long-term instance. There are totally 11089 service instances (jobs) running for the whole 12 hours (43200 seconds). It illustrates the **imbalanced numbers and durations of runtime instances for service and batch jobs.** By leveraging such imbalanced knowledges, one could schedule and co-locate batch and service instances in a more efficient way.

Table I
STATISTICS OF BATCH JOBS

Status	Number
Failed tasks	1126
Terminated tasks	67013
Waiting tasks	8847
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
Total instances	15186017
Total tasks	76986
Total jobs	12951
Average instance duration	129 (seconds)
Maximum instance duration	29558 (seconds)
Mimimum instance duration	$\leq$ 1 (seconds)
Average task duration	192 (seconds)
Maximum task duration	29585 (seconds)
Mimimum task duration	$\leq$ 1 (seconds)

To make efficient resource reservations, it is necessary to understand the workload characteristics and demands. We studied the distribution of resource requests, actual usages and corresponding utilization in the following subsections.

A. Imbalances of Resource Demands

1) Batch Workloads: For each job, we summarized its average requested and used CPU numbers per task in Figure 3. We accumulated the CPU and normalized memory requests of all instances within a task. And accumulated requested resources of all tasks within the same job, then divided by corresponding task numbers to get the average values. However, the scale of the normalized memory size per task would not be from 0 to 1, which we ignored.

We could see most of the batch jobs requested 1 to 100 cores of CPU for each task, while the maximum requested number is more than 1000. In contrast, we observed most jobs used 0.01 to 1 core CPU per task while very few used more than 100 cores. Additionally, we observed many jobs are waiting for resources while few jobs occupied overwhelming cluster CPU resource (more than 100 cores per job). Such **spatial imbalance** of CPU usages across workloads probably leads to the bottleneck of cluster throughput, while exacerbating inefficiency of resources. New scheduling algorithms are essential to accommodate imbalanced loads and demands of workloads.

In Figure 4 and Figure 5, we summarized the average requested and used CPU numbers per instance for each task, as well as normalized memory sizes. We accumulated all the CPU numbers and normalized memory sizes of instances within a task. And retrieve the average value through dividing the sum by corresponding task's instance numbers.

Most tasks request either 0.5 or 1 core CPU per instance, while few requests 6 or 8 cores (hard to distinguish in figure). The used CPU numbers are mainly between 0.1 and 0.7. Small portion of tasks' average used CPU numbers per instance are between 0.7 and 1.2. As we can see, most tasks' instances are operating at half of the CPU utilization (used to requests). Due to the mechanism of resource reservation, cloud datacenters are suffering severe inefficiency and wastes of resources.

From Figure 5, the majority of tasks requested normalized memory sizes between 0.05 and 0.15 per instance. While they commonly used 0.001 to 0.05 sizes. However, since the requested and used memory sizes per instance are normalized independently in two separate tables, it is not accurate to observe memory utilizations by comparing them directly. It's shown that most tasks are consuming only small portions of memory, while few occupied the majorities. It confirms the existences of **spatial imbalance** across workloads, and highlights the motivation to design new allocation mechanisms to handle complexity of scheduling.

2) Service Workloads: Each service instance is running within one Linux container for 12 hours. Figure 6 shows the average, maximum and minimum ratio of resource used to requests of all instances at each sampling time. It indicates the average time-varying utilization of CPU and memory per service instance.

Most service instances stably used less than 10% CPU resources they requested during executions. However, there were always some portions of instances consuming 60% to 90% resources (red maximum line), while some used near-to-zero cores (green minimum line). Such **spatial** and **temporal imbalances** across service instances make it knotty to make proper reservations. Balancing the trade-offs between performance and resource efficiency would be the principal challenge for cluster managers. The normalized average memory utilization is stably 45%, while maximum keeps 79% and minimum maintains 1%. Unlike resources of CPU, it is shown that there are opportunities to make better reservations to improve memory utilization [13].

Differ from the time-varying average utilizations of all instances in Figure 6 (spatial average), Figure 7 plots the CDF of instances' average CPU and memory utilizations of 12 hours (temporal average). The traces provides average CPU and memory utilization every 5 minutes per instance. We list the maximum and minimum values of 5-minute average utilization during 12 hours per instance, and plot the CDF. Similarly, we add up all 5-minute utilizations of 12 hours, and dividing it by intervals ($12 * 60 \div 5 = 144$) to achieve the average CDF curve.

There are 50% of instances whose average CPU utilization of 12 hours reached up to 0.05, maximum 0.2 and minimum 0.02. There are even 90% of instances whose maximum CPU utilization of 12 hours only reached up to 0.4, which illustrates extremely huge wastes of reserved CPU numbers. Unlike the idle of most CPU cores, memory utilization is a little bit higher. There were 50% of instances reaching about 0.45, 0.5 and 0.35 respectively. Comparing with

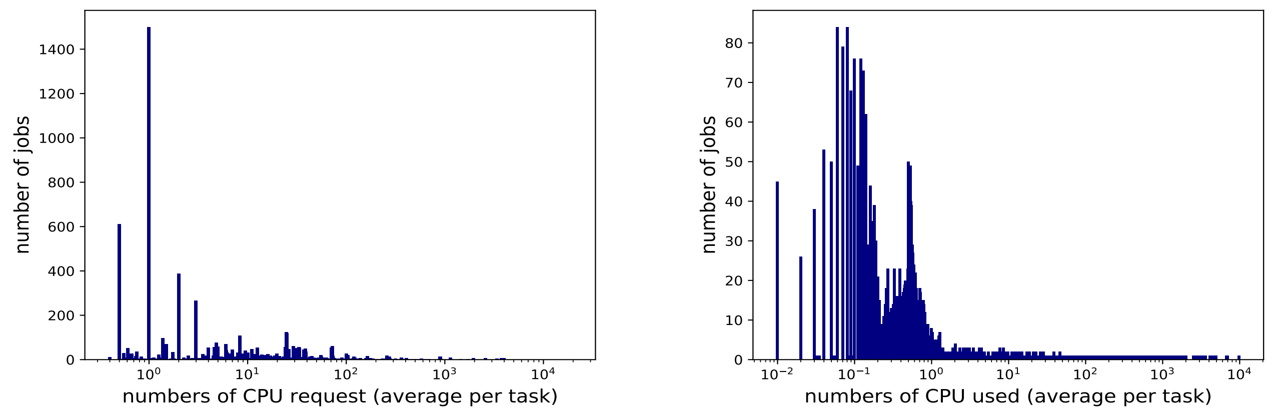

Figure 3. Job counts by average CPU request numbers (left) and average CPU used numbers (right) per task. Note the log-scale on the plot's x-axis.

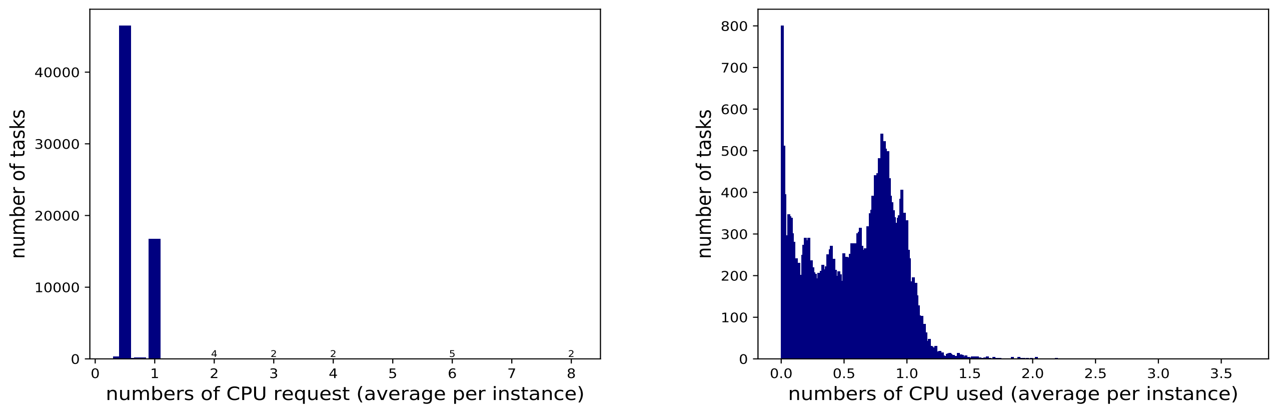

Figure 4. Task counts by average CPU request numbers (left) and used numbers per instance (right). Note the log-scale on the plot's x-axis.

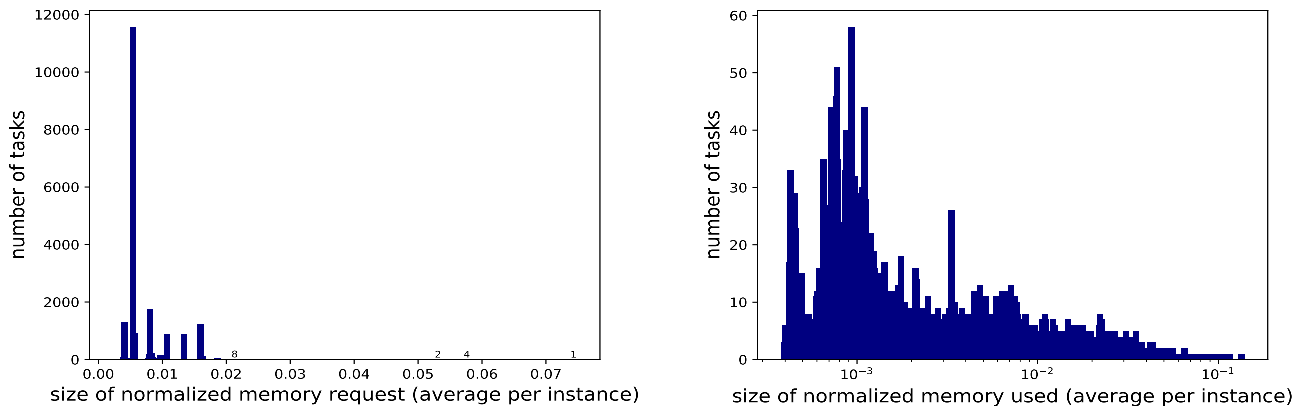

Figure 5. Task counts by normalized average memory request sizes (left) and used sizes (right) per instance. Note the log-scale on the plot's x-axis.

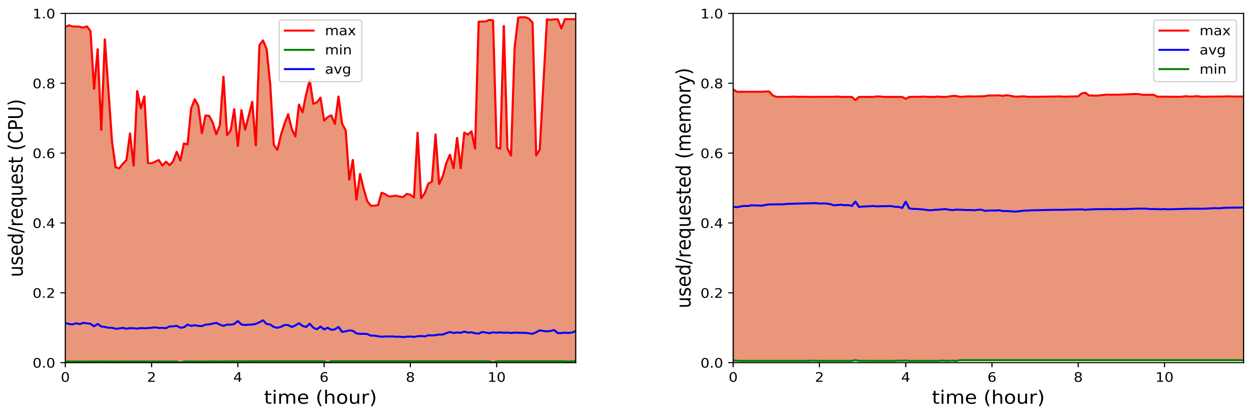

Figure 6. The ratio of used CPU and memory to requested per sampling time. The red, blue, green lines indicate the maximum, average and minimum ratio of all service instances respectively.

Figure 6, it identifies **imbalanced proportion between CPU and memory utilization** for service instances. Resource allocation strategy should take such imbalance into account, and design better fair share algorithm of multi-dimensional resources.

Users always tend to over-provision resources to guarantee SLA for latency-sensitive production services. However, such extremely low utilizations would lead to incredible high costs for large-scale cloud data center. Meanwhile, online service jobs reserved and hold resources forever, which might cause imbalanced cluster loads (hot spots) or job starvation due to insufficient resources on constrained hosts. By considering the results of Section IV-A1 and Table I, batch and online service jobs are shown to have serious **imbalanced instance numbers, resource utilization and duration.** Modern schedulers could take above runtime phenomenons of hybrid cluster into account, and adopts sophistical co-locating strategy to avoid imbalance and maximize resource efficiency.

Figure 8 shows the container's moving hourly average of CPU load. We find that the maximum CPU load is less than 60 percents, and the average CPU load is lower than 10%, which means most of containers use less than 10 percents of the CPU. From the beginning time of sampling, the maximum Linux CPU load is high, over 50 percents. But after less than 1 hour, the maximum load dropped drastically, and then the maximum CPU load keeps stable and occasionally fluctuating. As for the average CPU load, it is almost zero, which means most of machines are idle in the cluster. The imbalance of CPU load causes the severe resource waste.

The top of Figure 8 shows the hourly average and maximum CPU loads of all service instances. We observed the maximum CPU loads are below 60% while the average are below 10%. Additionally, CPU loads are about 60% at beginning and drastically drop to 20% one hour later. Afterwards, the maximum loads fluctuate over time while average ones keep stably few. Most instances are idle in cluster. We could see obvious **spatial and temporal imbalances** across service workloads, which increase the complexity of scheduling.

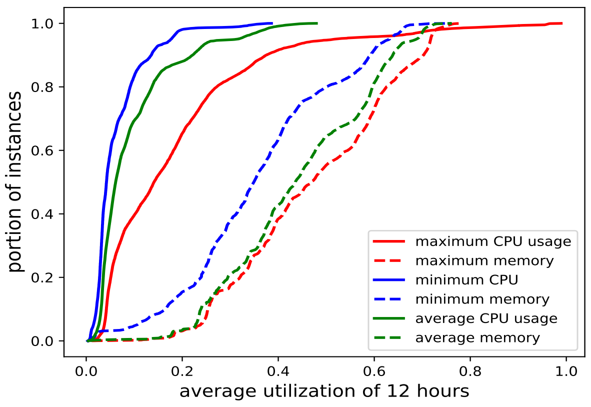

Figure 7. CDF of instances' average resource utilizations of 12 hours.

The bottom of Figure 8 displays the hourly average and maximum CPU loads of all machines. The fluctuation trends of both maximum and average loads are similar to containers'. However, the gaps between highest and lowest usages are even bigger. At beginning, the machine's maximum CPU loads are even over 1. While the minimum ones are still close to 0. The average CPU loads are about 20%. It confirmed the existences of **spatial and temporal imbalances** across machines.

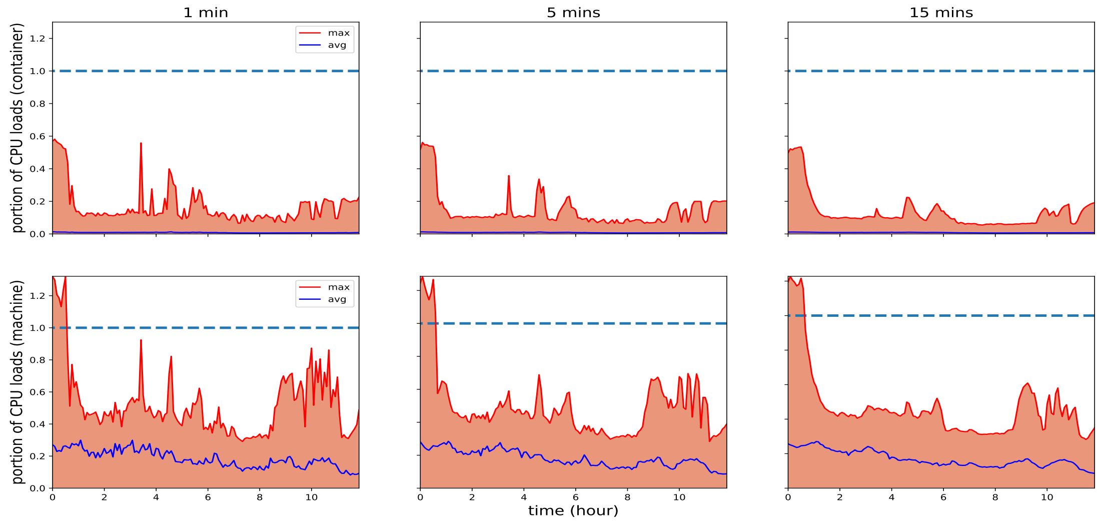

Figure 8. Hourly average and maximum CPU loads of all service instances (top) and machines (bottom). The sampling interval is one minute, five minutes, and fifteen minutes from left to right. The dashed line represents the total capacity of each machine.

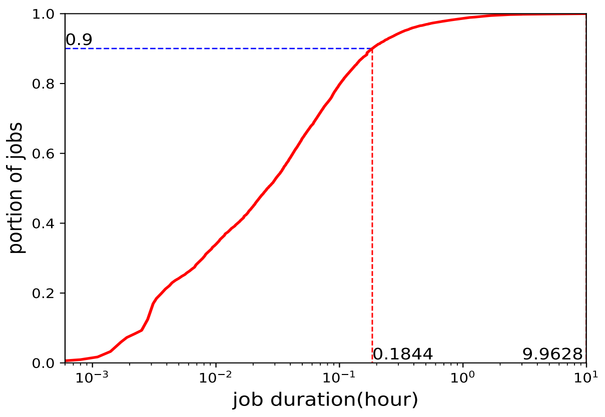

Figure 9. CDF of job durations. We only count the terminated jobs.

B. Imbalances of Batch Job Durations

Figure 9 plots the duration distributions of batch jobs. We exploited the difference between durations of earliest created task and latest finished task within the same job, to indicate job running time. 90% of jobs run less than 0.19 hours, while the longest one is running up to 10 hours. In detail, over 12481 jobs run less than half of an hour and over 12705 jobs run less than 1 hour. Short jobs overwhelmingly occupied the cluster. It also identifies the **imbalances of job durations.** One could take these phenomenons into account, and leverage proper scheduling algorithm such as SJF (Shortest Job First) to speedup executions of short jobs, while maximizing cluster makespan.

In addition, the large proportion of short jobs give us opportunities to improve quality of co-locating choices in hybrid cluster. We have more opportunities to select other proper jobs to avoid interferences and contentions between batch and service jobs. A scheduler that adequately exploiting such imbalances could greatly improve cluster efficiency and guarantee SLA for service jobs.

C. Discussion

Due to the reservation mechanism and imbalanced phenomenons in Alibaba cloud data center, co-locating service and batch jobs is ineffective to improve cluster efficiency. In the future, one could leverage flexible allocations of containers and knowledges of imbalances to greatly improve resource efficiency in hybrid cluster.

In addition, by considering data locality, the imbalance phenomenons would be aggravated during scheduling. How to make proper resource allocation and scheduling decisions to balance the trade-offs between imbalance relief, data locality and SLA (performance) is challenging. It also becomes our future research direction.

V. Related Work

Google released a 29-day trace of over 25 million tasks across 12,500 heterogeneous machines in 2011 [18]. There are several important works on analyzing Google trace from different perspectives. Zhang et al., focused on characterizing run-time task resource usages of CPU, memory and disk [19]. Reiss et al., characterized cluster resource requests, distributions, and the actual resource utilizations. They found heterogeneity and dynamics are two important characteristics [12][20]. Liu et al., characterized how the machines in cluster are managed and when the workloads submitted during a 29-day period behave. They focus on the frequency and pattern of machine maintenance events, job and task-level workload behaviors, and how the overall cluster resources are utilized [21]. Abdul-Rahman et al., considered user behaviors in composing applications from the perspective of topology, maximum requested computational resources, and types of workloads [22]. Sharma et al., focused on the task placement constraints in Google compute cluster and developed methodologies for incorporating task placement constraints and machine properties into performance benchmarks of large compute clusters [23]. Di et al., compared the differences between a Google data center and other Grid/HPC systems, focus on loads of jobs and machines [24].

While other works use machine learning method, such as k-means clustering, to study the workload characteristics. Mishra et al., described an approach to workload classification based on k-means and its application to the Google Cloud Backend [25]. Di et al., computed the valuable statistics about task events and resource utilization for Google applications, based on various types of resources (such as CPU, memory) and execution types (e.g., whether they can run batch tasks or not). They also classified applications via a K-means clustering algorithm with optimized number of sets, based on task events and resource usage [26]. Chen et al., identified common groups of jobs by k-means clustering. They also did correlation analysis between job semantics and job behavior, leading to helpful perspectives on capacity planning and system tuning [27].

While our work is one of the first analysis on Alibaba trace, which is released in September 2017. Furthermore, we analyze this dataset from a new perspective and find several interesting *imbalance* phenomena in the cloud.

VI. Conclusions

Understand machine characteristics and workload behaviors in large-scale cloud data centers is critical to maximize cluster resource efficiency. In this paper, we performed a deep analysis on a newly released trace dataset by Alibaba Group in September 2017, covering 1300 servers over 12 hours. To the best of our knowledge, this is one of the first work to analyze the Alibaba public trace.

We explored detail runtime characteristics of a hybrid cluster that co-locates both online service and offline batch jobs, and discovered several interesting insights about *imbalance* in the cloud. Such imbalances exacerbate the complexity and challenges of cloud cluster management, incurring severe resource inefficiency. We argue accommodating imbalances of both machines and workloads is critical to cluster efficiency, and will motivate the design and emergences of new resource managers and schedulers.

Acknowledgment

This work is supported by the National Key Research and Development Program of China (Grant No. 2016YFB1000204), NSFC (Grant No. 61702492, U1401258), Research Program of Shenzhen (JCYJ20170307164747920, JSGG20150512145714248), Natural Science Foundation of Guangdong Province (Grant No. 2014A030313687, 2015B010129011).

References

[1] L. A. Barroso, J. Clidaras, and U. Hölzle, “The datacenter as a computer: An introduction to the design of warehouse-scale machines,” *Synthesis lectures on computer architecture*, vol. 8, no. 3, pp. 1–154, 2013.

[2] A. Verma, L. Pedrosa, M. Korupolu, D. Oppenheimer, E. Tune, and J. Wilkes, “Large-scale cluster management at google with borg,” in *Proceedings of the Tenth European Conference on Computer Systems*. ACM, 2015, p. 18.

[3] M. Schwarzkopf, A. Konwinski, M. Abd-El-Malek, and J. Wilkes, “Omega: flexible, scalable schedulers for large compute clusters,” in *Proceedings of the 8th ACM European Conference on Computer Systems*. ACM, 2013, pp. 351–364.

[4] B. Hindman, A. Konwinski, M. Zaharia, A. Ghodsi, A. D. Joseph, R. H. Katz, S. Shenker, and I. Stoica, “Mesos: A platform for fine-grained resource sharing in the data center.” in *NSDI*, vol. 11, 2011, pp. 22–22.

[5] G. Xu, C.-Z. Xu, and S. Jiang, “Prophet: Scheduling executors with time-varying resource demands on data-parallel computation frameworks,” in *Proceedings of the 13th IEEE International Conference on Autonomic Computing*, ser. ICAC ’16. IEEE, 2016, pp. 45–54.

[6] V. K. Vavilapalli, A. C. Murthy, C. Douglas, S. Agarwal, M. Konar, R. Evans, T. Graves, J. Lowe, H. Shah, S. Seth, B. Saha, C. Curino, O. O’Malley, S. Radia, B. Reed, and E. Baldeschwieler, “Apache hadoop yarn: Yet another resource negotiator,” in *Proceedings of the 4th Annual Symposium on Cloud Computing*, ser. SoCC ’13. ACM, 2013, pp. 5:1–5:16.

[7] E. Boutin, J. Ekanayake, W. Lin, B. Shi, J. Zhou, Z. Qian, M. Wu, and L. Zhou, “Apollo: Scalable and coordinated scheduling for cloud-scale computing,” in *Proceedings of the 11th USENIX Conference on Operating Systems Design and Implementation*, ser. OSDI ’14. USENIX Association, 2014, pp. 285–300.

[8] Z. Zhang, C. Li, Y. Tao, R. Yang, H. Tang, and J. Xu, “Fuxi: a fault-tolerant resource management and job scheduling system at internet scale,” in *Proceedings of the VLDB Endowment*, ser. VLDB ’14. VLDB Endowment, 2014, pp. 1393–1404.

[9] C. Delimitrou and C. Kozyrakis, “Quasar: resource-efficient and qos-aware cluster management,” in *Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS ’14. ACM, 2014.

[10] S. A. Jyothi, C. Curino, I. Menache, S. M. Narayanamurthy, A. Tumanov, J. Yaniv, Í. Goiri, S. Krishnan, J. Kulkarni, and S. Rao, "Morpheus: Towards automated slos for enterprise clusters," in *Proceedings of the 12th USENIX Symposium on Operating Systems Design and Implementation*, ser. OSDI '16. USENIX Association, 2016, p. 117.

[11] K. Rajan, D. Kakadia, C. Curino, and S. Krishnan, "Perforator: eloquent performance models for resource optimization," in *Proceedings of the 7th ACM Symposium on Cloud Computing*, ser. SoCC '16. ACM, 2016, pp. 415–427.

[12] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Towards understanding heterogeneous clouds at scale: Google trace analysis," *Intel Science and Technology Center for Cloud Computing, Tech. Rep*, p. 84, 2012.

[13] G. Xu and C.-Z. Xu, "Prometheus: Online estimation of optimal memory demands for workers in in-memory distributed computation," in *Proceedings of the 2017 Symposium on Cloud Computing*, ser. SoCC '17. ACM, 2017, p. 655.

[14] Y. Yan, Y. Gao, Y. Chen, Z. Guo, B. Chen, and T. Moscibroda, "Tr-spark: Transient computing for big data analytics," in *Proceedings of the 7th ACM Symposium on Cloud Computing*, ser. SoCC '16, 2016, pp. 484–496.

[15] W. Chen, J. Rao, and X. Zhou, "Preemptive, low latency datacenter scheduling via lightweight virtualization," in *2017 USENIX Annual Technical Conference (USENIX ATC 17)*. USENIX Association, 2017, pp. 251–263.

[16] "Alibaba trace," *https://github.com/alibaba/clusterdata*, 2017.

[17] C. Delimitrou and C. Kozyrakis, "Quasar: Resource-efficient and qos-aware cluster management," *ACM SIGPLAN Notices*, vol. 49, no. 4, pp. 127–144, 2014.

[18] "Google trace," *https://github.com/google/cluster-data*, 2011.

[19] Q. Zhang, J. L. Hellerstein, and R. Boutaba, "Characterizing task usage shapes in googles compute clusters," in *Large Scale Distributed Systems and Middleware Workshop (LADIS'11)*, 2011.

[20] C. Reiss, A. Tumanov, G. R. Ganger, R. H. Katz, and M. A. Kozuch, "Heterogeneity and dynamicity of clouds at scale: Google trace analysis," in *Proceedings of the Third ACM Symposium on Cloud Computing*. ACM, 2012, p. 7.

[21] Z. Liu and S. Cho, "Characterizing machines and workloads on a google cluster," in *Parallel Processing Workshops (ICPPW), 2012 41st International Conference on*. IEEE, 2012, pp. 397–403.

[22] O. A. Abdul-Rahman and K. Aida, "Towards understanding the usage behavior of google cloud users: the mice and elephants phenomenon," in *Cloud Computing Technology and Science (CloudCom), 2014 IEEE 6th International Conference on*. IEEE, 2014, pp. 272–277.

[23] B. Sharma, V. Chudnovsky, J. L. Hellerstein, R. Rifaat, and C. R. Das, "Modeling and synthesizing task placement constraints in google compute clusters," in *Proceedings of the 2nd ACM Symposium on Cloud Computing*. ACM, 2011, p. 3.

[24] S. Di, D. Kondo, and W. Cirne, "Characterization and comparison of cloud versus grid workloads," in *Cluster Computing (CLUSTER), 2012 IEEE International Conference on*. IEEE, 2012, pp. 230–238.

[25] A. K. Mishra, J. L. Hellerstein, W. Cirne, and C. R. Das, "Towards characterizing cloud backend workloads: insights from google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, pp. 34–41, 2010.

[26] S. Di, D. Kondo, and F. Cappello, "Characterizing cloud applications on a google data center," in *Parallel Processing (ICPP), 2013 42nd International Conference on*. IEEE, 2013, pp. 468–473.

[27] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available google cluster trace," *EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2010-95*, vol. 94, 2010.