Evolution and Heterogeneity of Cloud Clusters: Comparative Study Between Google and Alibaba Big Trace Data

Li Ruan, Xiangrong Xu, and Limin Xiao
State Key Laboratory of Software Development Environment
Beihang University
Beijing, China
ruanli@buaa.edu.cn

Feng Yuan, Yin Li
Institute of Software Application Technology
Guangzhou & Chinese Academy of Sciences
Guangzhou, China
yf@gz.iscas.ac.cn

Dong Dai
Department of Computer Science
University of North Carolina at Charlotte
Charlotte, NC, United States
dong.dai@uncc.edu

***Abstract*—With the booming of the scale and complexity of Cloud infrastructure, how to understand the temporal and spacial characteristics of cloud computing infrastructure in detail under real workloads becomes an increasing challenge. The most notable characteristics are evolution and heterogeneity. One challenge in learning from the trace data is that the traces have diversity and a trace analysis alone is insufficient to accurately prove the generality of a new technique, resulting in urgent requirements of comparative analysis of different traces. Another challenge is that the Cloud cluster trace is no doubt big multiview data now, resulting in difficulty in abstract views from tables.**

This paper studies the evolution and heterogeneity of Cloud clusters by comparatively studying the two most representative valuable big traces of Google trace, which is the most popular trace, and the Alibaba 2018 trace, which is the newest publicly released one. We first introduce a multiview based trace analysis framework. We then investigate the evolution and heterogeneity of cloud traces by characterizing how the jobs, tasks, machines and resources usage are managed. Quantitative comparative findings and inferences are presented which verified the effectiveness of the proposed method. To the best of our knowledge, we are the first to perform a comparative quantitative empirical study on these two trace using a multi-view based approach. Our multifaceted analysis and new findings not only reveal insights that we believe are useful for system designers, IT practitioners and users but also promote more new researches on big trace data comparative analysis in large-scale clusters.

***Index Terms*—Google trace, Alibaba trace, big data analysis, Cloud computing**

This work is by supported by the National Key R&D Program of China under Grant NO. 2017YFB0202004, the fund of the State Key Laboratory of Software Development Environment under Grant No. SKLSDE-2017ZX-10, and the National Science Foundation of China under Grant No. 61772053 and No.61572377. Guangzhou Science and Technology Projects (Grant Nos. 201807010052 and 201610010092), Nansha Science and Technology Projects (Grant No. 2017GJ001)

I. Introduction

The last decade has witnessed a surge of interest and commercial usage in cloud computing which offers high scalability and flexibility to meet the diverse computing and storage requirements. With the booming of the scale and complexity of Cloud infrastructure, how to effectively understanding the temporal and spacial state of cloud computing platform becomes an increasing interesting problem. The most notable characteristics are evolution and heterogeneity. For example, Barroso et al. [1] presents four challenges to address the Cloud characteristics: rapidly changing workloads, building balanced systems with imbalanced commodity components, curbing energy usage and maintaining high parallel efficiency in the presence of mismatches in performance and cost trends of hardware components. Addressing each challenge requires detailed understanding about *how a cloud computing infrastructure is utilized under real workloads and what hidden information about the jobs, tasks, machines and resources usage are hidden in these traces* first.

A. Challenges and Problems

George Amvrosiadis, et al. [2] have proposed and verified that one challenge in cluster workload traces analysis is that the traces have *diversity* and a trace analysis alone is insufficient to accurately prove the generality of a new technique. Therefore, there is urgent requirement to characterizing jobs, tasks, machines and resources usage by comparative analysis studies on traces to help cluster operators identify system bottleneck and figure out solutions for optimizing system-level and application-level performance.

On the other hand, despite intense research and development in the areas of cluster trace datasets, such as FaceBook trace [3], taobao data set [4], Nutanix1 private clouds trace [5], etc.. Most traces are private while the publicly available cluster big datasets remain relatively scarce [2]. The two most representative cloud computing dataset sources today are: the Google cluster trace (Google trace for short in the following) [6] which is collected in 2011 by Google and is the most popular trace by far [2]; the Alibaba trace 2018 (Ali2018 for short in the following) [7] is by far the newest public Cloud platform trace released in December 2018 by one of the top three Chinese Cloud platform providers called Alibaba.

The third challenge is that the Cloud cluster trace, which contains temporal and spacial state information of the machines, systems and applications of large-scale clusters, is now no doubt in big multiview data era. For example, according to our statistical analysis of these two traces, Google trace, a 29-day 41G trace, amounts 670 thousand jobs, 25 million tasks, and 1.4 billion entries. Ali2018, a 8-day 49G trace, amounts 4 million 200 thousand jobs, 14 million tasks and 5 billion 600 million entries.

Unfortunately, although it is reported in [2] that there are more than 450 publications on Google traces [8] [9] [10] and some initial research work on Alibaba trace 2017 [11], to the best of our knowledge, the published research on the most newest Ali2018 has not been reported. Moreover, the research on the evolution and heterogeneity of Cloud Clusters based on a comparative multiview study between Google trace and Ali2018 still lacks.

B. Main Contributions

In this work, we develop a set of models, methods and algorithms to address the challenges. The key contributions are highlighted as follows:

- According to the published research, to the best of our knowledge, we are the first to perform a multi-view based comparative empirical study on Google cluster and Alibaba cluster 2018.
- We characterize the jobs, tasks, machines and resources usage through a comparative study and give multifaceted analysis. New findings reveal insights that we believe are useful for system designers, IT practitioners and users working on cluster management systems and promote the big trace data analysis in large-scale clusters.

C. Organization

The remainder of this paper is organized as follows: Section II gives a broad overview of some related work. We present a general multiview based cluster trace analysis framework to guide the comparative empirical study of Google Trace and Ali2018 Trace for multi-view trace data in Section III. The Global view, machine view, Jobs and tasks view based analysis especially from the resource usage, task execution and task failure point, together with the new findings based on comparative study are demonstrated in Section IV. The paper is concluded in Section V.

II. Related Work

Despite intense research and development in the areas of cluster trace datasets, such as FaceBook trace [3], taobao data set [4], Nutanix1 private clouds trace [5], Google trace and Alibaba trace, publicly available cluster workload big datasets remain relatively scarce [2]. The two most representative cloud computing dataset traces today are: the Google trace and Ali2018.

Reiss et al. [12] proposed that the machines and workload in Google trace are heterogeneous. The authors of [13] focused on the resource configuration of the machine and the execution process of the job, but lacks analysis of the task. Md. Rasheduzzaman et al. [14] used the k-means algorithm to cluster the jobs, indicating that more jobs are short-lived and low-memory. The above papers focus more on the workload of the system, but lack fine-grained analysis of resource utilization and job execution results. Chen et al. [15]and Jassas et al. [16] studied the task failure in Google trace and analyze some reasons for the failure of the task. They pointed out that task failure may be related to the time sensitivity of the task, task priority, and task resource request. However, they did not pay specific attention to the ratio of CPU resource request to memory resource request, which will also affect the task completion rate.

Although there are some analysis of Alibaba trace 2017 which is the older version of Ali2018, no one has analyzed Ali2018 yet. Wenyan Chen et al. [11] clustered and analyzed the workload of Alibaba trace 2017. Lu et al. [17] indicated that Alibaba trace 2017 imbalances exacerbate the complexity and challenges of cloud resource management, which can result in severe resource waste and low cluster utilization. However, Alibaba trace 2017 only contains 12 hours of cluster information and approximately 1.3k of machine information. However, Alibaba trace 2017 only contains 12 hours of cluster information and about 1.3k of machine information. Compared to the larger data size of Ali2018, Alibaba trace 2017's description of Alibaba cluster is not detailed enough to show some rules that need to be observed for a long time.

Although there are many cloud trace analysis on Google trace, Ali2017 trace, Facebook trace, etc., their analysis are based on one trace. George Amvrosiadis et al. [2] proposed that we should pay attention to the diversity of the trace analysis. They compared four new traces to demonstrate the problem of the overfitting of the previous work to Google's dataset characteristics. Existing research does not investigated the Ali2018 which is from Chinese biggest cloud can reflects the newest evolution and heterogeneity of cloud trace. Few studies been performed to compare two traces derived from actual production infrastructures from the multiple views of machines, jobs, tasks, etc.

III. Case Study Process Design

We propose a general multiview based trace data analysis framework. A graphical illustration of the proposed framework is given in Fig. 1 in Section III-A and the multivew based trace analysis process is presented in Section III-B to facilitate the understanding of our comparative study.

TABLE I: Trace, machine and job statistics of Google trace and Ali2018

View	Items	Google trace 2011	Ali2018 2018
Trace overview	Trace data size	41G	49G
	Trace sampling time	29 days	8 days
	Timestamp unit	Microsecond	Second
	Number of jobs	670 thousand	4.2 million
	Number of task	25 million	14 million
	Number of entries	1.4 billion	5.6 billion
Machine information analysis	Number of machines	12583	4043
	Machine states	Add, Remove, Update	Using, Import installing
	Machine attribute	CPU, Memory capacity	CPU, Memory, net capacity
Job analysis	Job record form	There is a separate job table	From the task table
	Time sensitive task	Distinguish by scheduling class	Batch tasks, Online task
	Relationship between tasks in a job	Not recorded	Recorded by DAG
	Task execution process	Not recorded	One or more instances
	Entries loss handling	Virtual record	Unprocessed
User	Recorded user information	Yes	No

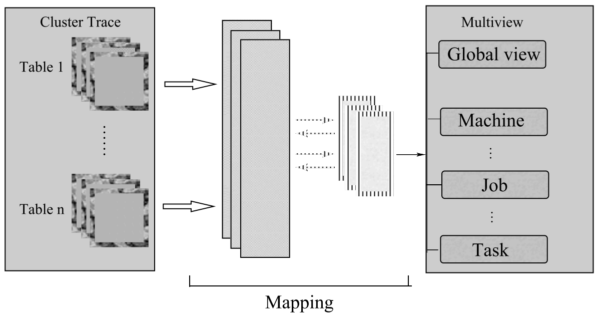


Fig. 1: Multiview based trace data analysis.

A. Multiview based trace analysis framework

The cloud cluster trace is no doubt big multiview data now which contains implicit associations among jobs, tasks, resources, failures among trace tables . To investigate the evolution and heterogeneity of the Cloud Cluster, how to organize such multiview data and abstract the features from the big trace data are the first obstacle.

We propose a multiview based trace statistical analysis framework by using an example in Fig. 1. Following this framework, we need to analyze the cluster trace based on its various tables, establish the mapping and abstract the feature of trace based on the mapping, formulate the global views and other sub views, and finally perform statistical analysis based on the views and give out the new findings.

B. Multiview based trace analysis Process

Based on the framework in Fig. 1, we now present the details of multivew based trace analysis process. The steps are highlighted as the follows:

1) Analyze and compare the trace statistics and data structure based on the overall view.
2) Establish the views based on the trace features and expert knowledge. For example, in Google trace and Ali2018, machine, job, task, resource usage are the most important views for large-scale clusters.
3) Analyze the mapping relationship between the views and the trace tables.
4) Perform the data analysis based on the trace tables from the temporal and spatial views.
5) Visualize the trace analysis data analysis results.
6) Comparatively analyze the evolution and heterogeneity among different traces.
7) Analyze findings and give inferences.

Using Google trace and Ali2018 as an example, the analysis process results will be shown in the following Section IV.

IV. Case Study Findings and Results Analysis

In this section, based on the framework and process in section III, we first classify the data table of Google and Ali2018 based on alobal view in Section IV-A. Machine view, Job view and Task view based comparative analysis and new findings are introduced in the following sections.

A. Global view based data tables classification

Google trace is 41G before decompression. It contains 6 tables, *machine events*, *machine attributes*, *job events*, *task events*, *task constraints*, *task usage*. Ali2018 is 49G when uncompressed with 6 tables of *machine meta*, *machine usage*, *container meta*, *container usage*, *batch task* and *batch instance*.

Based on our framework in Fig. 1, we first classify the data structure of the tables in the Google trace and Ali2018 based on the views so as to establish the mapping relationship. Based on the classification results in See Table I, some overall findings and inference from a global data view are summarized in the following as an example which we think is useful for further job and task scheduling, resource usage, etc.:

- Google trace and Ali2018 have relatively same-scale of trace data size while Googles duration is four times that of Ali2018.
- Ali2018 has 4.2 million jobs and 14 million tasks, while Google has 670 thousand jobs and 1.4 billion tasks, we can infer that the number of tasks in each job (3.33 tasks per job) in Ali2018 is ten times smaller than that (37.31 tasks per job) in Google trace. Based on the above analysis, we can conclude that the jobs in Google is, averagely speaking, divided in to a more fine-grained smaller tasks during the execution.

TABLE II: Machine resource capacity configuration CPU_{cap}, Mem_{cap}) in Google trace

CPU_{cap} \ Mem_{cap}	$[0, 0.25)$	$[0.25, 0.5)$	$[0.5, 0.75)$	$[0.75, 1]$	total
0.25	126 (1%)	0	0	0	126(1%)
0.5	3924 (31.17%)	6732 (53.48%)	1003 (7.97%)	5 (0.04%)	11664 (92.66%)
1	0	3 (0.02%)	0	795 (6.32%)	798 (6.34%)
total	4050 (32.17%)	6735 (53.50%)	1003 (7.97%)	800 (6.36%)	12588 (100%)

(CPU_{cap} denotes CPU Capacity and Mem_{cap} denotes Memory Capacity)

- Google trace has more than three times of machines than Ali2018.
- From Ali2018, the DAG relationship can be somewhat reconstructed while Google trace has not reported such relationship.
- User information can not be revealed in Ali2018 while Google trace can do.

Besides the overall view based analysis, in the following section, we will investigation the evolution and heterogeneity of Clouds by comparatively characterizing their jobs, tasks, etc..

B. Analysis of the machine

Analyzing the cluster trace from the view of the machine is the basis to understanding the resource configuration and update in the cluster.This section investigates the resource usage information in the view of machines by the record of *machine event*.

1) Machine configuration analysis: At the first step, we analyze the the difference of the configurations between Google trace and Ali2018. The resource configuration of 12,583 machines was recorded in Google trace. The CPU capacity of all machines in Google trace is normalized into three types: 0.25, 0.5 and 1. For the normalized memory capacity, we classified it into four types: [0, 0.25), [0.25, 0.5), [0.5, 0.75), [0.75, 1]. Next, through statistical analysis of the *machine event* table, the distribution of the number of machines of the resource capacity is shown in Table II and Fig. 2.

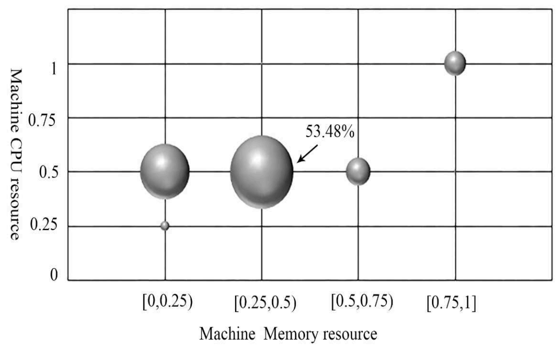


Fig. 2: Machine record number distribution of various resource capacities in Google trace. Bubble size reflects the number of machines

As can be seen from Table II, in Google trace, the majority (92.66%) of the machines are configured with the CPU capacity of 0.5. From the perspective of memory resources, the number of machines with memory resources in [0.25,0.5) is the highest, accounting for 53.48%. We can conclude that in Google trace, on the one hand, the machine's CPU and memory capacity values are diverse, on the other hand, more than half of the machines have a relatively identical resource capacity configurations whose CPU capacity is 0.5 and memory capacity is [0.25,0.5).

TABLE III: Homogeneous analysis of Google trace and Ali2018

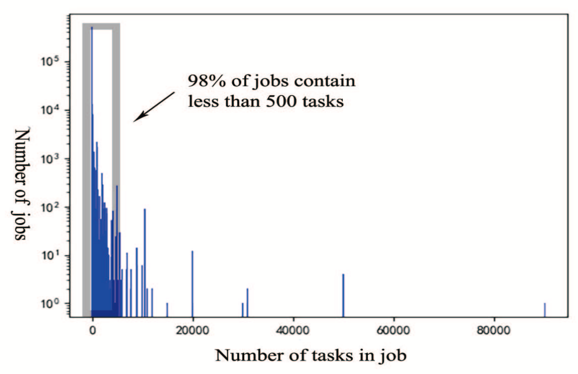
Trace \ Items	CPU	Memory	Homogeneous
Google trace	0.25, 0.5, 1	[0, 0.25), [0.25, 0.5) [0.5,0.75), [0.75, 1]	No
Ali2018	96 cores	100	Yes

To analyze the machine characteristics in Ali2018, we first compute the numbers of deduplicated *machine_id* in table *machine_meta* and find that Ali2018 has a total of 4043 machines. Next, we focus on the question whether Ali2018 has a fine-grained resource management schema as Google trace does by analyzing the CPU's and memory's capacity in Ali2018.

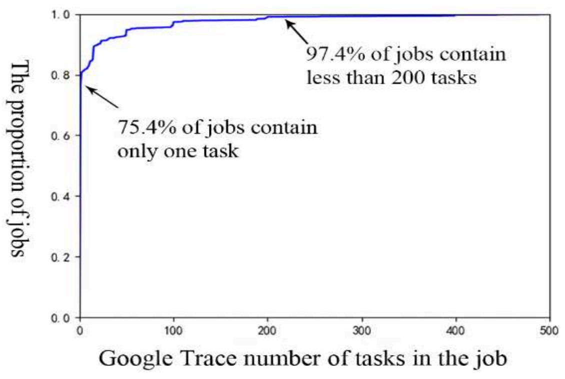
We choose the record *mem_size* as the memory capacity index, and the record *cpu_num* as the CPU capacity index . We find that every machine is configured with the CPU capacity of 96 cores and the normalized memory capacity of 100. i.e., each machine in Ali2018 is configured with the identical CPU and Memory capacity.

The comparative machine configurations based on the above results are summarized in Table III. From Table III and the above comparative analysis, we can deduce that the management of the CPU capacity and the memory resource in Ali2018, which has the same resource capacity configuration, is more coarse-grained than Google trace whose resource capacity configurations are diverse. i.e., Google trace is heterogeneous and Ali2018 is homogeneous in terms of resource capacity configurations.

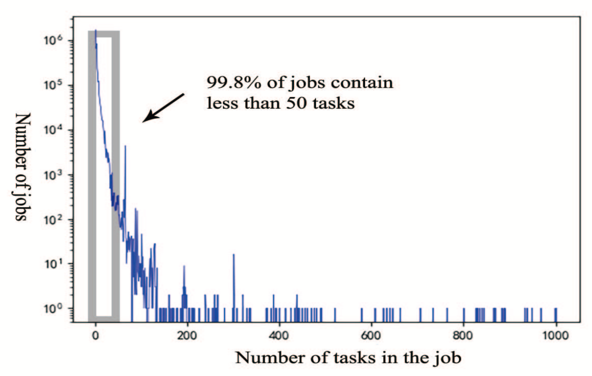
2) Machine events: There are three types of machine events in Google trace, namely *Add*, *Remove* and *Update*, as shown in Fig. 3. The *Add* event is when the machine joins the cluster and can perform the task. There are 21,443 *Add* records in Google trace, far more than the other two types of records. Because at the beginning of the observation time, all the machines in the cluster will have one *Add* record. Remove event refers to the machine failure or other reasons to exit the cluster. There are



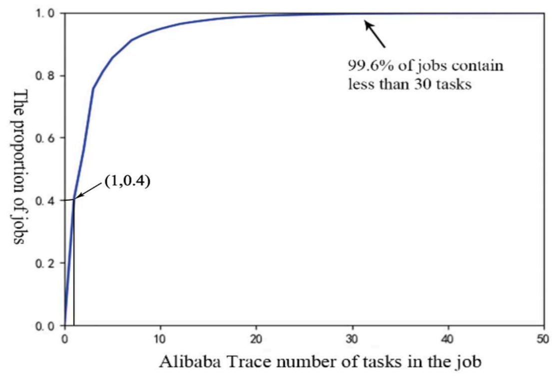
(a) Frequency distribution of jobs for the specified number of tasks included in Google trace



(b) CDF for jobs with the specified number of tasks in Google trace



(c) Frequency distribution of jobs for the specified number of tasks included in Ali2018



(d) CDF for jobs with the specified number of tasks in Ali2018

Fig. 4: Google trace and Ali2018 job and task distribution diagram

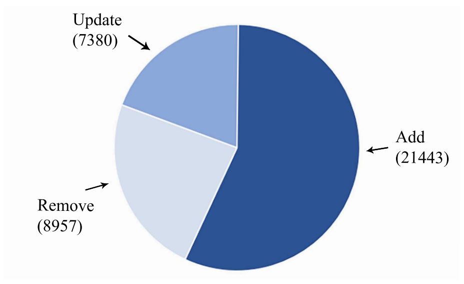


Fig. 3: The event distribution map of each machine in Google trace.

a total of 8957 Remove records in Google trace. The Update record is a change to an available resource on the machine, with a total of 7,380 records in trace. By analyzing the number of Add and Remove records, it can be concluded that some machines rejoin the cluster after being removed.

In Ali2018, the machine has only two states, *Using* and *Import*. In almost all the records, the state of the machine is *Using*, and only the machine numbered 2739 has produced 5 records with the state of $Import$. The $Import$ records may be caused by a machine configuration failure.

Finding 1: Google trace has more fine-grained resource and event management mechanism than Ali2018. Google trace is heterogeneous and Ali2018 is homogeneous in terms of resource capacity configurations. There are three types of events in Google trace, which are *Add*, *Remove* and *Update*. However, in Ali2018, the majority of entries, except for 5 entries, has single event type like *Using*. We can infer that Google platform has a much preciser machine state change management approach than Ali2018.

C. Jobs and tasks characteristic analysis

This section focuses on the static characteristic of jobs and tasks in Google trace and Ali2018, including the size of the jobs, job resource usage, etc.

1) The large and small jobs identification in terms of its tasks: Through the statistics of the tasks in the table *task events* of Google trace, we find that the size of the jobs in Google trace is quite different.For example, job 15590910 contains only 1 task while job 644725309 is composed of 90050 tasks in Google trace. Compared with Google trace, the size difference between the jobs in Ali2018 is smaller, for example the job numbered J1651717 with the largest amount of tasks only has 1002 tasks.

Fig. 4 shows the number of jobs of various sizes in Google trace and Ali2018. As can be seen from the Fig. 4(a) and Fig. 4(b), about 75% of the jobs in Google trace are composed of only one task, and the jobs containing less than 500 tasks account for more than 98% of the total number of jobs. From Fig. 4(c) and Fig. 4(d), we can see that, in Ali2018, 40% jobs in the total of 14 million jobs has only one task , while 99.6% jobs contain less than 30 tasks. i.e., the jobs in Ali2018 tends to be small jobs with small tasks. When scheduling jobs in a cluster, the feature that small jobs account for the majority should be fully considered to avoid affecting the overall scheduling efficiency.

Finding 2: Most of the jobs in Google trace and Ali2018 contain only a few tasks. In Google Trace, about 98% of jobs contain less than 500 tasks. Approximately 99.6% of jobs contain less than 30 tasks in Ali2018.

Algorithm 1 Algorithm for calculating ($RatioCPU_{Ali}$ and $RatioCPU_{Google}$) of Ali2018 and Google trace

Input: N(the number of top jobs), CPU resource usage of jobs in Ali2018 and Google trace
Output: $RatioCPU_{Ali}$ and $RatioCPU_{Google}$ of Ali2018 and Google trace

```
1: Sort the jobs in descending order by comparing the resource usage in Ali2018 and Google trace respectively
2: Calculate the total CPU resource usage ($TotalCPU_{Ali}$ and $TotalCPU_{Google}$) of all jobs in Ali2018 and Google trace respectively
3: Calculate the CPU usage of top $N$ sorted jobs ($TopCPU_{Ali}$ and $TopCPU_{Google}$) in Ali2018 and Google respectively
4: $RatioCPU_{Ali} = TopCPU_{Ali}/TotalCPU_{Ali}$ and $RatioCPU_{Google} = TopCPU_{Google}/TotalCPU_{Google}$
5: return $RatioCPU$ of Ali2018 and Google trace
```

2) Resource requests: In Google trace, the CPU and memory usage status is recorded in the table *task usage* [18]. We use the following method to calculate the resource usage of the jobs in Google trace.

Suppose a job contains the $task\ num$ tasks, T is a two-dimensional vector representing statistical time slices, and T_i represents the time period of each time slice of $task_i$. A is also a two-dimensional vector representing the resource usage of each time slice, and A_i represents the resource usage of each time slice of $task_i$. When calculating the resource usage of $task_i$, r_i is $A_i \cdot T_i$. For example, as shown in the Fig.5, the total resource usage of the task is the sum of the areas of the graphs. The total resource usage of the job in Google trace is the sum of the resource usage of all the tasks it contains.

In Ali2018, the jobs are also divided into multiple tasks to execute, and each task will have multiple execution instances. Suppose a job in Ali2018 contains n tasks, each task has j_i instance, and the CPU resource request of each instance is r_k, then the total CPU resource request R of the job is shown in Eq. 1. This paper calculates the resource occupation of each job in Google trace and Ali2018, and draws the cumulative

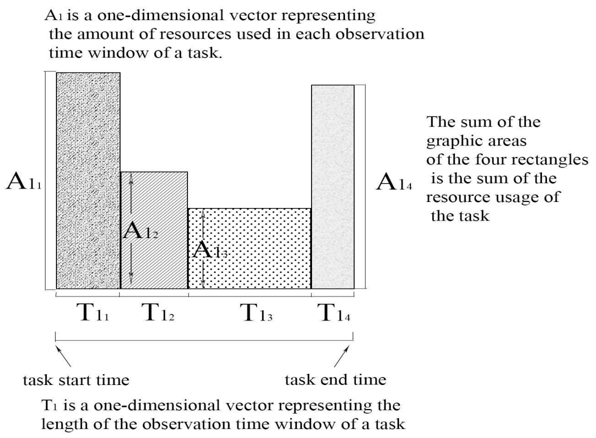

A₁ is a one-dimensional vector representing the amount of resources used in each observation time window of a task.
A1₁
A1₂
A1₃
A1₄
The sum of the graphic areas of the four rectangles is the sum of the resource usage of the task
T1₁
T1₂
T1₃
T1₄
task start time
task end time
T₁ is a one-dimensional vector representing the length of the observation time window of a task

Fig. 5: Schematic diagram of Google trace resource calculation method

resource usage distribution of the top 90,000 jobs in Google trace and the top 500,000 jobs in Ali2018.

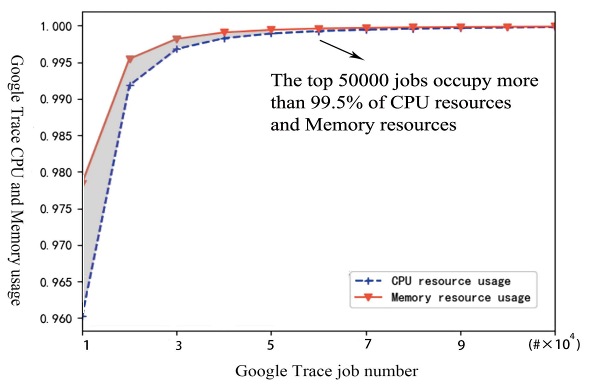
$$R = \sum_{i=1}^{n} \sum_{k=1}^{j_i} r_k \tag{1}$$

TABLE IV: Variable definition in Algorithm 1

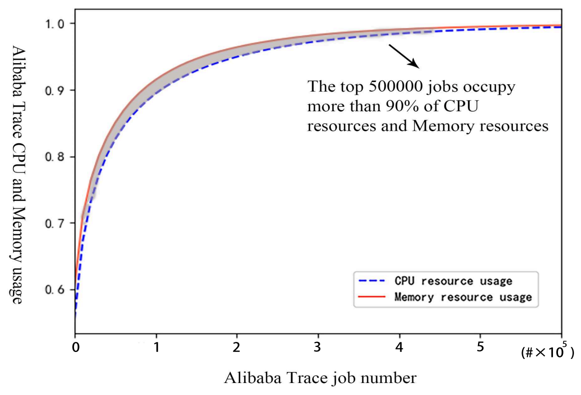
Ali2018	Variable interpretation
$TotalMem$	the memory usage of all the jobs in two traces
$TopMem$	the memory usage of top sorted jobs in two traces
$TotalCPU$	the CPU usage of all the jobs in in two traces
$TopCPU$	the CPU usage of top sorted jobs in two traces
$RatioMem$	top n jobs memory usage / all the jobs memory usage
$RatioCPU$	top n jobs CPU usage / all the jobs CPU usage

To be clearer, we first define some important notions as shown in the Table IV. In addition, we let N denotes the number of top jobs. Then, for any N that is less than the total number of jobs, we can calculate the ratio of the CPU resource usage of Top N jobs to the CPU usage of all jobs ($RatioCPU$) in the trace through Algorithm 1. $RatioMem$ can also use a similar calculation method. After we have calculated the $RatioMem$ and $RatioCPU$ corresponding to each N in Ali2018 and Google trace, it's simple to plot the cumulative distribution (Fig. 6) of CPU and memory resource usage.

As shown in Fig. 6, the top 50,000 jobs consume more than 99.5% of the resources, and the other jobs with more than 500,000 occupy less than 0.5% of the resources, that is, the jobs with less than 10% consume more than 99.5% of the resources in Google trace. In Ali2018, out of a total of 4.2 million jobs, the first 500,000 jobs occupy more than 90% of CPU resources and memory resources. It should be noted that in both types of trace, the memory resources are more job-intensive than the CPU resources. In other words, as we can see from Fig. 6(a) and Fig. 6(b), the red line representing memory usage is always above the blue line representing the

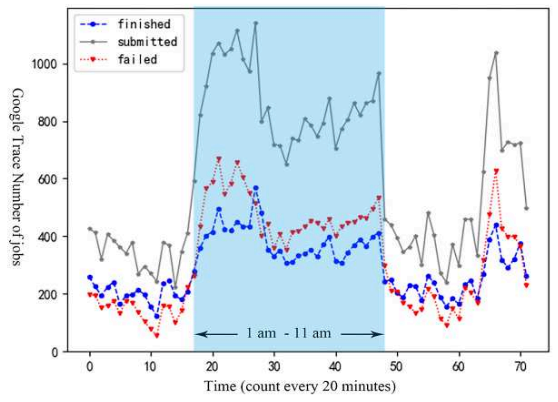


(a) CPU usage and memory usage in Google trace

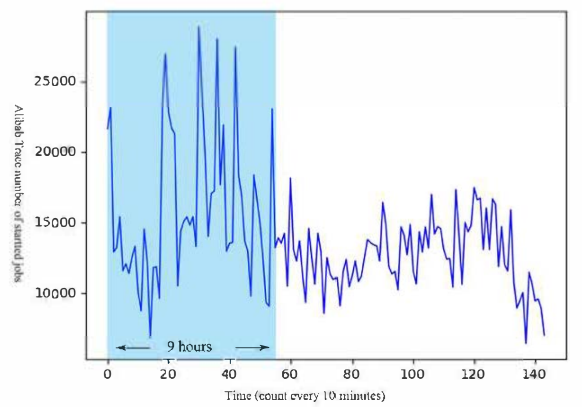


(b) Top jobs occupying most CPU and memory in Ali2018

Fig. 6: Top jobs occupying most CPU and memory



(a) Finished/Submitted/Failed Tasks Time Series in Google trace



(b) Finished/Submitted/Failed Tasks Time Series in Ali2018 trace

Fig. 7: Minutely job submission rates for a given day.

CPU usage. i.e., for a given N, we can find that $RatioMem \geq RatioMem$ for both of these two traces.

Finding 3: The usage for job resources in Google trace and Ali2018 is subject to pareto's law, that is, a small number of large jobs occupy the vast majority of resources.The memory resources are more job-intensive than the CPU resources in both Google trace and Ali2018.

D. Task execution analysis

In this section, we mainly study the dynamic characteristics hidden in the execution of the job. We investigate the workload status of Google trace and Ali2018 by analyzing job submission, completion, failure and other job event information. We mainly analyze the daily patterns of workloads varied according to different time and the resource utility.

1) Daily patterns of workloads: Fig.7 shows the number of tasks submitted by Google trace and Ali2018 within 24 hours. As can be seen from Fig. 7, the number of job submissions of Google trace was significantly higher from 1am to 11am than that of other time periods, when Google cluster was supposed to start processing batch tasks. The number of job submissions at 2am in Google trace is 5 times the number of job submissions at 2 pm. We can assume that the workload on Google trace is uneven at different times on the same day. Ali2018 doesn't give a specific start time, so we can't tell the real time of a timestamp. The number of job submissions of Ali2018 remained at a high level for 9 hours in 24 hours, while the number of job submissions in the rest of the time was less. We can conclude that the workload of Ali2018 is also uneven. Putting some time-insensitive tasks into space time to perform is an important way to improve system performance in both Google trace and Ali2018.

Finding 4: The workloads of Google trace and Alibaba Trace vary greatly at different times of the day. The number of task submissions in Google trace during busy hours is 5 times that of the idle time period. Ali2018 workload remained at a high level for 9 hours in 24 hours

2) Task failure rate and task submission trend: It can be seen from Fig. 8 that the number of job submissions, the

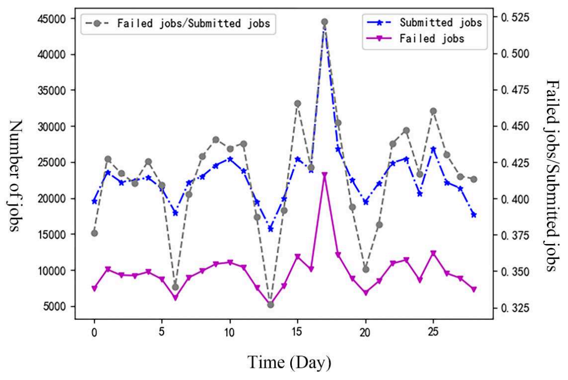


Fig. 8: Failed job ratio in Google trace

number of job failures and the change trend of the job failure rate curve are consistent. For example, from the sixth day to the tenth day and then to the thirteenth day, all three are descending and then rising, and finally falling. It is worth noting that the curve fluctuation range of the job failure rate is much larger than that of the job submission number curve, so we can speculate that the job failure rate is sensitive to the change of job submission number. If Google adjusts the machine in time to increase the available resources after the number of submitted jobs increases, the number of failed jobs will be greatly reduced.

Finding 5: the number of job failures and job submissions in Google trace are consistent, and the failure rate of jobs is similar to the trend of job submissions.

3) Waste of Google trace resources: In Google trace, there is a class of tasks that are always in a commit, schedule, kill loop, and that cannot be executed due to a lack of runtime conditions or other factors. For example, the task whose job id is 4121195473 has been executed for a whole 29 days (the time span of Google trace observation) and 114 machines have been changed for execution, but the task is still not successfully completed. Another example is the task execution status in job 515042969. These tasks (in job 515042969) have been killed more than 2.2 million times in total, but they still don't finish. Obviously, the scheduling algorithm of Google trace does not pay attention to these tasks, resulting in a serious waste of resources. Based on the statistics of *task events* and *task usage*, we find that the top 1000 jobs of being killed consume about 19% CPU resources and 11% memory resources, yet none of these jobs are successfully executed within 29 days. If the scheduling of Google trace can focus on this kind of tasks, wait for the conditions on which the task depends to be satisfied before starting execution, it will surely greatly improve the performance of the system.

Finding 6: The scheduling algorithm of Google trace has obvious resource waste.

V. Conclusion

In this paper we conducted a comparative empirical Study on Google trace and Ali2018 from a multi-view perspective. To the best of our knowledge, we are the first to perform a comparative empirical study on Google trace and Ali2018. Our multifaceted analysis and new findings not only reveal insights that we believe are useful for system designers, IT practitioners and users but also promote more new researches on big trace data comparative analysis in large-scale clusters.

References

[1] e. a. Luiz Andra Barroso, Jimmy Clidas, *The Datacenter as a Computer*. Morgan and Claypool, 20009.

[2] G. R. G. G. A. G. George Amvrosiadis, Jun Woo Park, "On the diversity of cluster workloads and its impact on research results," in *2018 USENIX Annual Technical Conference (USENIXATC 18)*, 2018, pp. 533–546.

[3] "Facebook workloads repository," https://github.com/SWIMProjectUCB/SWIM/wiki/Workloads-repository, accessed March 24, 2019.

[4] Z. Ren, J. Wan, W. Shi, X. Xu, and M. Zhou, "Workload analysis, implications, and optimization on a production hadoop cluster: A case study on taobao," *IEEE Transactions on Services Computing*, vol. 7, no. 02, pp. 307–321, apr 2014.

[5] I. Cano, S. Aiyar, and A. Krishnamurthy, "Characterizing private clouds: A large-scale empirical analysis of enterprise clusters," in *Proceedings of the Seventh ACM Symposium on Cloud Computing*. ACM, 2016, pp. 29–41.

[6] "Google cluster-data," https://github.com/google/cluster-data, accessed March 24, 2019.

[7] "Alibaba cluster-data," https://github.com/alibaba/clusterdata/, accessed March 24, 2019.

[8] S. Gupta and D. A. Dinesh, "Resource usage prediction of cloud workloads using deep bidirectional long short term memory networks," in *2017 IEEE International Conference on Advanced Networks and Telecommunications Systems (ANTS)*. IEEE, 2017, pp. 1–6.

[9] W. Zhang, B. Li, D. Zhao, F. Gong, and Q. Lu, "Workload prediction for cloud cluster using a recurrent neural network," in *Identification, Information and Knowledge in the Internet of Things (IIKI), 2016 International Conference on*. IEEE, 2016, pp. 104–109.

[10] C. Peng, Y. Li, Y. Yu, Y. Zhou, and S. Du, "Multi-step-ahead host load prediction with gru based encoder-decoder in cloud computing," in *2018 10th International Conference on Knowledge and Smart Technology (KST)*. IEEE, 2018, pp. 186–191.

[11] W. Chen, K. Ye, Y. Wang, G. Xu, and C. Xu, "How does the workload look like in production cloud? analysis and clustering of workloads on alibaba cluster trace," in *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2018, pp. 102–109.

[12] 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.

[13] P. Minet, . Renault, I. Khoufi, and S. Boumerdassi, "Analyzing traces from a google data center," in *2018 14th International Wireless Communications Mobile Computing Conference (IWCMC)*, June 2018, pp. 1167–1172.

[14] M. Rasheduzzaman, M. A. Islam, T. Islam, T. Hossain, and R. M. Rahman, "Task shape classification and workload characterization of google cluster trace," in *2014 IEEE International Advance Computing Conference (IACC)*. IEEE, 2014, pp. 893–898.

[15] X. Chen, C.-D. Lu, and K. Pattabiraman, "Failure analysis of jobs in compute clouds: A google cluster case study," in *2014 IEEE 25th International Symposium on Software Reliability Engineering*. IEEE, 2014, pp. 167–177.

[16] M. Jassas and Q. H. Mahmoud, "Failure analysis and characterization of scheduling jobs in google cluster trace," in *IECON 2018-44th Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2018, pp. 3102–3107.

[17] C. Lu, K. Ye, G. Xu, C. Xu, and T. Bai, "Imbalance in the cloud: An analysis on alibaba cluster trace," in *2017 IEEE International Conference on Big Data (Big Data)*, Dec 2017, pp. 2884–2892.

[18] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., White Paper*, pp. 1–14, 2011.