Characterization and Comparison of Cloud versus Grid Workloads

Sheng Di[1], Derrick Kondo[1], Walfredo Cirne[2]
[1]INRIA, France, [2]Google Inc., USA
{sheng.di,derrick.kondo}@inria.fr, walfredo@google.com

***Abstract*—A new era of Cloud Computing has emerged, but the characteristics of Cloud load in data centers is not perfectly clear. Yet this characterization is critical for the design of novel Cloud job and resource management systems. In this paper, we comprehensively characterize the job/task load and host load in a real-world production data center at Google Inc. We use a detailed trace of over 25 million tasks across over 12,500 hosts. We study the differences between a Google data center and other Grid/HPC systems, from the perspective of both *work* load (w.r.t. jobs and tasks) and *host* load (w.r.t. machines). In particular, we study the job length, job submission frequency, and the resource utilization of jobs in the different systems, and also investigate valuable statistics of machine's maximum load, queue state and relative usage levels, with different job priorities and resource attributes. We find that the Google data center exhibits finer resource allocation with respect to CPU and memory than that of Grid/HPC systems. Google jobs are always submitted with much higher frequency and they are much shorter than Grid jobs. As such, Google host load exhibits higher variance and noise.**

I. Introduction

Miron Livny once said, "I've been doing research in *Clouds* before it was called *Grids*". This quote highlights fundamental questions: what is novel in Clouds, and what are the real differences between Clouds and Grids? Potential differences can arise in many areas, including but not limited to software configuration, resource and job management, and monetary charging models. High-level differences between Clouds and Grids are easy to identify. For instance, in Clouds, software configuration and management is eased via virtual machine images. Resource management is potentially more dynamic as virtual machines can be booted up and down over time. Charging models are pay-as-you-go.

While high-level differences may be easy to identify, the devil is in the details; substantiating such differences *quantitatively* and *statistically*, specifically for work load and host load, is critical for the design and implementation of Cloud systems. It is challenging because it requires detailed traces of production and often proprietary Clouds. Moreover, such traces are often enormous, and one requires statistical techniques for summarizing and understanding the data.

At a high-level, one could hypothesize fundamental differences in the time dynamics of load due to differences in the types of users and applications. In Grids and HPC systems, users often submit large scientific applications or experimental research applications. In Cloud data centers, users and applications are more commercial and interactive. Typical jobs come from web services (for meeting search queries, and online document editing and translations, for instance), and map-reduce applications (for building inverted indices of web documents or email, for instance).

Given those hypotheses, we characterize quantitatively and statistically host and work load in commercial data centers, and compare and contrast properties of Cloud versus Grid load. By *work* load, we refer to load due to incoming jobs and their corresponding tasks submitted by users. By *host* load, we refer to load on a particular machine due to executing tasks. We use the sole term *load* to refer generically to both *work* load and *host* load.

A detailed characterization of a system's load is critical for optimizing its performance. In the context of data centers, host and work load characterization is essential for job and resource management including but not limited to capacity planning, virtual machine consolidation, load balancing, and task scheduling. For instance, by characterizing common modes of host load within a data center, a job scheduler can use this information for task allocation and improve utilization. Alternatively, the resource management system can proactively shift and consolidate load via (VM) migration to improve host utilization, using fewer machines and shutting off unneeded hosts.

So, we investigate the following questions. First, what are the characteristics and statistical properties of work and host load in Cloud data centers? Second, what are the key similarities or differences of work and host load between Clouds and Grids, and the implications for job/resource management?

To answer those questions, we study a workload trace of a production data center at Google [1]. Google offers several Cloud services, such as web search, interactive documents, email, where users (indirectly) submit jobs without having to specify the details of their execution. As such, we believe this trace is representative of real-world Cloud workloads. The trace contains detailed measurements reported every 5 minutes of the states and events concerning users, jobs, tasks, and hosts. In total, the trace contains details about over 25 million tasks executed over 12,500 hosts during 1-month time period.

For comparison, we also characterize the work load and host load from different Grid systems, including AuverGrid [2], NorduGrid [3], sharcnet [4], as well as other HPC clusters, from Argonne National Laboratory (ANL) [5], RIKEN

Integrated Cluster of Clusters (RICC) [6], METACENTRUM cluster [7] and Lawrence Livermore National Laboratory (LLNL) [8]. Grid applications are mainly oriented for sicentific problems with heavy computation workloads.

We characterize the workload for Google's jobs, based on Google's trace data with over 670,000 jobs. By comparing to the real-world trace of many other Grid systems and HPC systems, we find that Google jobs usually have much shorter length are submitted at higher frequency. Each Google job takes relatively a small share of resources, leading to a much finer resource allocation granularity. Moreover, each Google job usually consists of only a single task; a typical user job, such as keyword search, is relatively compact and self-contained. In Grids, each job can consist of several processes running simultaneously over multiple cores.

With respect to host load, we give qualitative and quantitative descriptions of static metrics, such as machine's maximum host load. We also describe dynamic metrics, such as queue state, and relative usage levels compared to capacities. Interestingly, we find that host load in data centers has higher variance than in Grids. This is due to the fact that the Google workload consists of a large number of short tasks that complete within a few minutes.

For the remainder of the paper, we use the terms **data center and cloud server** interchangeably. In Section II, we describe our modeling approaches in processing the Google trace data. In Section III, we comprehensively analyze the characteristics of the workload for Google jobs and tasks as compared to those of Grid jobs. We characterize the host load of Google data center in Section IV, by taking into account the variance of priorities, resources, and time periods. Finally, we discuss related work in Section V and conclude with future work in Section VI.

II. System Model based on Google's Trace Data

In this section, we first briefly describe Google cluster's task scheduling mechanism, and data contained in the trace.

A *Google cluster* consists of many machines that are connected via a high-speed network. One or more schedulers receive and process a large number of user requests (a.k.a. jobs), each of which is comprised of one or more tasks. For instance, a map-reduce program will be handled as a job with multiple reducer tasks and mapper tasks. Each task (actually represented as a Linux program possibly consisting of multiple processes) is always generated with a set of user-customized requirements (such as the minimum CPU rate and memory size). Each job corresponds to one user, and it may contain one or more tasks, which are the smallest units of resource consumption. In addition, different tasks have different scheduling priorities, and there are currently 12 priorities in total.

According to Google's usage trace format [1], each task can only be one of the following four states, *unsubmitted*, *pending*, *running* and *dead*, as shown in Figure 1 (excerpted from [1]). Any newly submitted task will be first put in the pending queue, waiting for the resource allocation (the step (1) as shown in the figure). As long as some available resources meet the task's constraints, it will be scheduled onto the qualified resource for execution (step (2)). Users are allowed to tune their tasks' constraints at runtime, adapting to users' needs and the environment (step (3)). For example, a user can expand the upper bound of CPU rate and memory size for the task in the course of its execution. Any task may be evicted by system, killed or lost during its execution. After its execution, it will enter into the dead state (step (4) and (5)), and it can be resubmitted by users later (step (6)) to wait for further resource allocation.

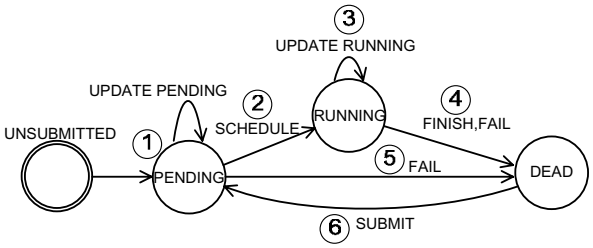


Figure 1. State Transitions of Google Tasks

Basically, the Google scheduler processes high-priority tasks before low-priority ones, and performs the first-come-first-serve (FCFS) policy on the ones with the same priority. The "best" resources will be used first, in order to optimally balance the resource demands across machines and minimize peak demands within the power distribution infrastructure, reaching approximately optimal resource utilization. Such a model will use the resources according to their abilities, leading to an approximate load balancing situation.

Based on the above task processing model, Google traced over 25 million tasks that were scheduled/executed across over 12500 heterogeneous machines within one month. All the tasks are submitted with a set of customized constraints and priorities, which will be discussed later. More than 10 metrics are collected during the one month of task-event monitoring, including CPU usage, assigned memory, observed real memory usage, page-cache memory usage, disk I/O time, and disk space.

When releasing the trace, Google normalized almost all floating-point values by its theoretical maximum value. These values were transformed in a linear manner. So, relative information about host load, for example, is preserved.

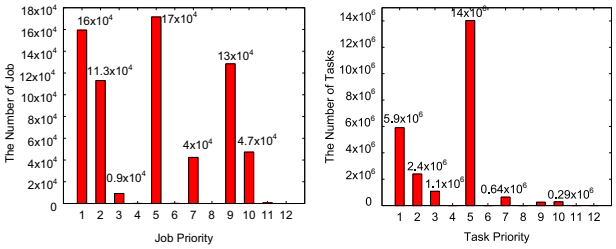
III. Analysis of Google Workload

In this section, we characterize the workload of jobs on a Google data center, by comparing it to that on other Grid or HPC systems. Based on our characterization, we find that Google jobs behave quite differently from the traditional Grid jobs, especially with respect to job length, frequency of submission, and resource utilization.

Based on Google's trace, we first group the all 25 million tasks in terms of their job IDs, and then compute the

statistics (such as mean CPU usage and memory usage) for each job. The Grid/HPC jobs to be compared are also from real-world trace data, available from the Grid Workload Archive (GWA) [9] and Parallel Workload Archive (PWA) [10] respectively. Their corresponding applications are usually for scientific reseach, which would probably cost a heavy workload on computation.

1) Job/Task Priority: In a Google data center, each task is submitted with a particular priority, which is selected from 12 levels. Any tasks with high priorities are able to preempt other tasks with lower priorities. All of the tasks that belong to the same job have the same priority. According to the histogram shown in Figure 2, there are three clusters of priorities, low priority (1∼4), middle priority(5∼8), and high priority (9∼12), for both jobs and tasks.



(a) The Number of Jobs (b) The Number of Tasks

Figure 2. Statistics based on Different Priorities

Also, we can observe that most of the jobs/tasks are assigned with relatively low priorities (from 1 to 5). So we study later the usage load level or idleness state of the system from the perspective of different task priorities. For example, if a machine's resource utilization is very full but over 90% of execution time is attributed to the tasks with low priorities, the machine can still be considered quite idle w.r.t the tasks that have relatively high priorities (e.g., the values are greater than 4).

2) Job/Task Length: The length of a Google job or a Grid job is defined as the duration between its submission time and its completion time. Its value could be affected by many factors, such as the work to be processed, the priority of the job, the system state (idle or busy) when it is submitted, and so on. We compare the cumulative distribution function (CDF) of the job length between Google and other Grid systems in Figure 3. We observe that Google jobs are quite shorter than Grid jobs: over 80% Google jobs' lengths are shorter than 1000 seconds, while most of Grid jobs are longer than 2000 seconds. This difference is mainly due to the fact that Grid jobs are usually based on complex scientific problems, while Google jobs, such as keyword search, are often real-time.

We also compare Google task lengths to AuverGrid's job lengths. We compare the mass-count disparity of the two systems in Figure 4. (We use the term task length and task execution time interchangeably.)

Mass-count [11] is a very important metric used to summarize the key features (such as heavy tails) for specific distributions. It is made up of the "count" distribution and the "mass" distribution. The "count" distribution simply refers to the cumulative distribution function (CDF) as it counts how many items are smaller than certain size. The "mass" distribution weights each item, specifying the probability that a unit of mass belongs to an item. Specifically, their values are calculated based on Formula (1) and (2), where $f(t)$ is referred to as the probability dense function.

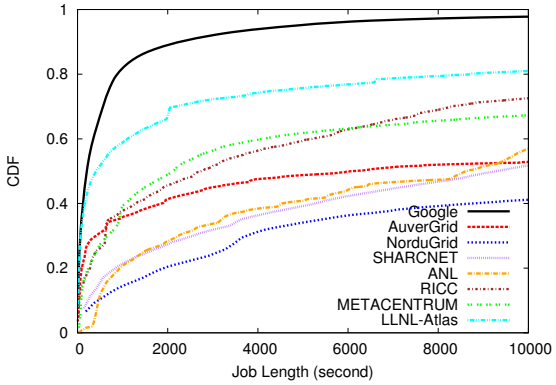
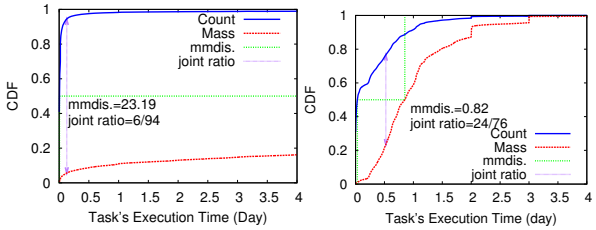


Figure 3. The CDF of Job Length of Google and Grid Systems



(a) Google's Task Length (b) AuverGrid's Task Length

Figure 4. Mass-Count Disparity of Task Lengths (Google v.s AuverGrid)

$$F_c(x) = \Pr(X < x) \tag{1}$$

$$F_m(x) = \frac{\int_0^x t \cdot f(t)\, dt}{\int_0^\infty t \cdot f(t)\, dt} \tag{2}$$

By comparing the two curves, we can determine whether the distribution follows Pareto principle [12], heavy tails, or other statistical features. In the analysis, *joint ratio* (a kind of Gini coefficient [11]) is a critical measure index, defined as X/Y, meaning that X% of the items account for Y% of the mass and Y% of the items account for X% of the mass. The *mm-distance* (abbreviated as mmdis.) shown in the figure is defined as the horizontal distance of the two points that are right in the middle of the CDF of the Count curve and Mass curve. Longer distance means a larger number of period lengths each with long duration.

Task lengths are 1.29 times longer on average in AuverGrid than in Google. However, AuverGrid's maximum task length is 1.61 times smaller than that in the Google cluster. Statistics shows that the average and maximum values of

task's execution time among AuverGrid's 340,000 submitted tasks are 7.2 hours and 18 days respectively, while those in Google cluster are 5.6 hours and 29 days respectively. Through our analysis of task length's mass-count disparity (see Figure 4), the distribution of Google's task lengths exhibits the Pareto principle much more than that of Grid's. Specifically, about 94% of tasks' execution times in Google's data center are less than 3 hours. In contrast, only 70% of tasks in AuverGrid are smaller than 12 hours. Such a difference in task length significantly impacts the fluctuation of host load, which we discuss later.

3) Job Submission Frequency: Job submission frequency is evaluated via the submission interval length between two consecutive job submissions. This reflects the interactivity between users/administrators and the systems. Figure 5 presents the CDF of the submission interval length. We can clearly observe that the submission interval length of Google jobs is much shorter than that of Grid jobs, which means that the frequency of Google job submission is much higher than that of Grid jobs. This can also be confirmed by Table I, which shows the minimum/mean/maximum number of jobs submitted per hour in Google and Grid/HPC systems. It is observed that the frequency of Google's job submission is much higher than that of other Grid systems.

We use the fairness index [13] to demonstrate the stability of the job submission frequency. The fairness index is defined in Formula (3), where x_i refers to the number of job submissions within one hour in different periods. The higher value of the fairness index, the higher stability of the job submissions. Table I shows that Google jobs are submitted with higher and more stable frequencies than Grid jobs, while Grid job submissions exhibit significantly low fairness because of their strong diurnal periodicity.

$$f(x) = \frac{\left(\sum_{i=1}^{n} x_i\right)^2}{n \sum_{i=1}^{n} x_i^2} \tag{3}$$

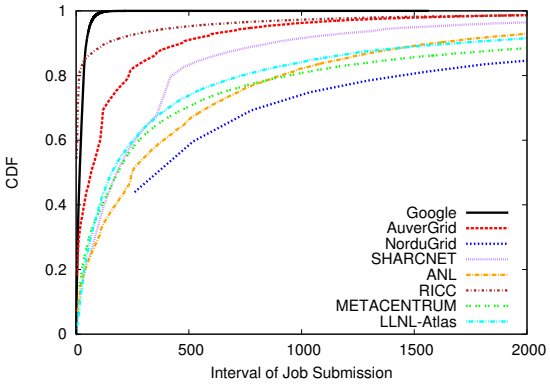


Figure 5. The CDF of Submission Interval of Google and Grid Systems

4) Job Resource Utilization: The last insight about the jobs is on their resource utilization, such as the CPU usage and the memory usage per job. We evaluate the CPU usage by the ratio of the cumulative execution time on one or more processors and its wall-clock time, as shown in Formula (4). For comparison, we use the mean memory size used by the job to evaluate its memory utilization.

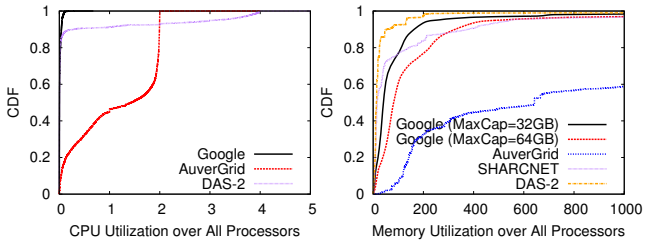
Table I
THE NUMBER OF JOBS SUBMITTED PER HOUR

	Google	AG	NG	SN	ANL	RICC	MT	LLNL
max. #	1421	818	2175	22334	132	4919	2315	240
avg. #	552	45	27	126	10	121	24	8.4
min. #	36	0	0	0	0	0	0	0
fairness	0.94	0.35	0.11	0.04	0.51	0.14	0.04	0.23

$$CPU\_Usage = \frac{\#\ of\ CPU \cdot ExeTimePerCPU}{WallClockTime} \tag{4}$$

We present the CPU usage and memory usage in Figure 6 (a) and (b) respectively. In Figure 6 (a), we can observe that the CPU used by Google jobs is always smaller than that of other Grid systems. Specifically, a large majority of Google jobs just need one processor per job at any time, though each job may contain multiple tasks submitted in a sequential order. In contrast, the jobs in AuverGrid and DAS-2 are likely parallel programs simultineously running on multiple execution nodes.

As for memory usage, Google trace does not expose the exact memory size used by jobs but their scaled values compared to the maximum memory capacity of each node. By assuming the maximum memory capacity to be equal to 32GB and 64GB respectively, we could also compare Google jobs and Grid jobs with respect to their memory utilization. In Figure 6 (b), it is observed that the memory size used by Google jobs is always relatively small as users' short-term, interactive jobs are dominant and not resource intensive.



(a) CPU Usage (b) Memory Usage

Figure 6. CPU & Memory Usage of Jobs

IV. ANALYSIS OF GOOGLE'S HOST LOAD

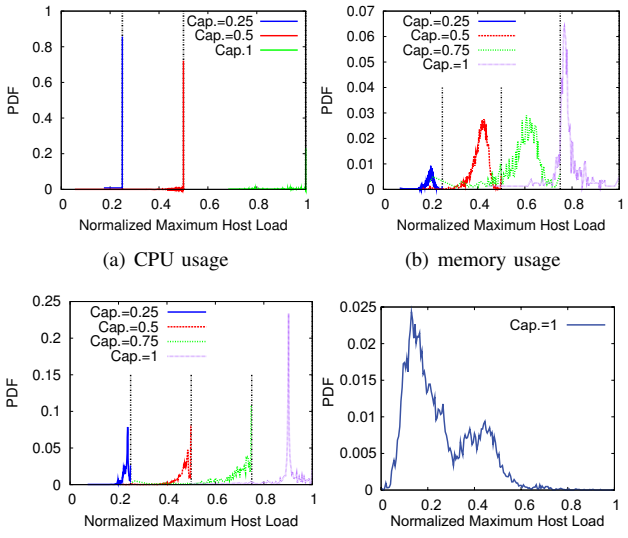
In this section, we first focus on the statistics of Google clusters' static metrics, such as machine's maximum host load on different attributes. After that, we will analyze two types of dynamic metrics (queuing state and usage levels), which change over time. Our analysis is based on the observation of the Googles' thousands of machines.

A. Analysis of Static Metrics

In the Google cluster, different machines have different capacities for various resource types (CPU, memory, page

cache). So it is necessary to characterize the heterogeneity as well as the maximum loads on different machines. In the trace, the normalized hosts' capacities are provided by dividing by the maximum value w.r.t. different attributes. According to statistics, we further infer the upper bound using the maximum resource usage value over the lifetime of the trace for each host. That is, from the task usage on a given machine, we calculate the machine resource usage, and use the maximum resource usage to indicate the machine's valid ability. According to our Google co-author, this is a valuable estimate, since the capacity of *usable* resource in user space is often less than the *full* capacity due to system overheads (from the kernel, for instance.)

According to the statistics on 12500 machines, Figure 7 shows the probability distribution of the maximum consumed host load during the whole lifetime of the trace, w.r.t. the four significant attributes, *CPU usage*, *memory consumed*, *memory assigned*, and *page cache used*. CPU usage is measured based on CPU core seconds per second, that is, the more cores a machine owns, the higher computation ability it would have. The consumed memory indicates the practical value of the memory size consumed by applications, while the assigned memory means the memory size allocated to the applications. The page cache metric records the total amount of Linux page cache (i.e., file-backed memory) including both mapped and unmapped pages. Recall that all the usage values in the Google trace are normalized based on the corresponding maximum node capacity, we can only show the relative heterogeneity instead of the exact amounts. The black dotted line in the figure indicates the normalized heterogeneous capacities, e.g., for all machines, CPU's capacities are 0.25, 0.5 and 1; memory's are 0.25, 0.5, 0.75 and 1.



(a) CPU usage (b) memory usage

(c) memory assigned (d) page cache memory

Figure 7. Distribution of Maximum Load

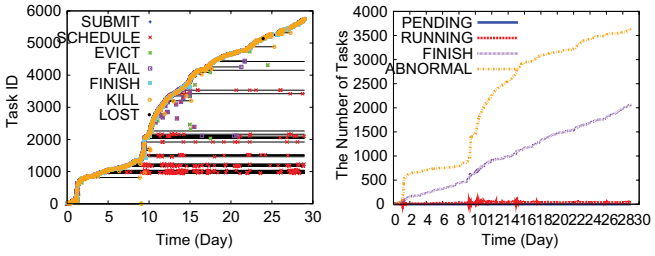
Figure 7 (a) shows that most of machines' maximum host loads are very close to their capacities. Specifically, for the machines with relatively low and middle CPU capacities, and over 80% and 70% of these hosts' maximum loads are equal to their capacities respectively. From Figure 7 (b) and (c), we clearly see that the memory capacity among all machines can be split into four groups, and the expected values of the maximum memory size consumed on machines are kept around 80% of their corresponding capacities, which implies that the memory overhead of each machine must be non-negligible. In comparison, the summed assigned memory size is around 90% of the capacity with high probability. The page cache capacities on machines are the same as each other, while the maximum consumed values shows a clear bimodal distribution.

B. Analysis of Dynamic Metrics

We investigate the characteristics of the dynamic features of the resource metrics appearing in the Google cluster trace data. We also compare the load changes between Grid platform and Cloud platforms, with respect CPU usage and memory usage.

1) Machine's Queuing State: A machine's queuing states refer to the number of tasks that are kept in different states (pending, running, finish, or abnormal) respectively. The distribution of states changes over time as new tasks are submitted/scheduled or old tasks are finished normally/abnormally. Task's abnormal termination is due to one of the following four task events, *evicted* by higher priority tasks, *failed* because of task failure, *killed* by its user, and *lost* because of its missing source data.

Different task events occur with different frequencies, but different machines exhibit similar distributions on the different types of events. We present the task event trace of a particular machine, in Figure 8 (a). We use the black lines (between start time and end time) to indicate the running period for each task.



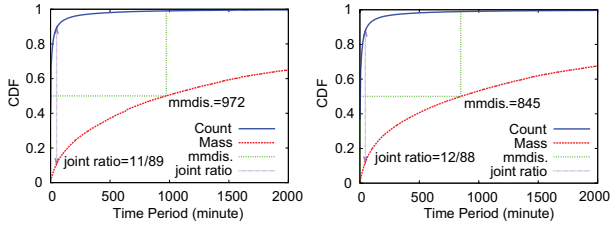
(a) Task Events (b) Queuing State

Figure 8. Task Events and Queuing State on a Particular Host

Based on the different states the tasks are treated over time, we can imagine them being kept in different queues (pending-queue, running-queue, dead-queue) on the machines. For example, we define the *running queue state* to be the number of running tasks at some time point. Figure 8 (b) also presents the queuing states over time. Specifically,

the running-queue state increases gradually from zero to 40 and then this number will be kept stable until the end of the trace. The pending-queue state (i.e., the number of pending tasks) is always 0 (except for the bootstrap period of the system), which means that each task is able to be immediately scheduled as it is submitted. Moreover, the number of finished tasks increases linearly and many of them belong to the abnormal-completion state. According to statistics, for the totally 44 million task-completion events , about 59.2% are abnormal ones, among which most of them belong to the *fail* state (50%) or the *kill* state (30.7%).

We show more statistics about running tasks on nodes, since the system's dynamic state is mainly attributed to the running tasks instead of the pending or finished ones. We split the running-queue state into 6 intervals based on different number of running tasks, [0,9], [10,19], [20,29], [30,39], [40,49], [50,$\cdots$]. Then, we analyze the mass-count disparity of the period lengths in which the running-queue state is unchanged. We observe the distribution of these lengths over all machines (see Figure 9) exhibits the Pareto principle especially from the perspective of the joint ratio and mm-distance metrics. That is, the majority of the unchanged-state lengths are quite short. In Figure 9, we just show four major intervals for simplicity. It is obvious that the first three intervals follows the 10/90 rule, while the last one ([40,49]) follows the 15/85 rule. The distribution is thus skewed as about 90% of tasks have very short continuous durations and contain 10% mass. Moreover, the first three intervals also show a similar mm-distance, while the last one presents a much smaller mm-distance. This implies that the [40,49] running-queue state changes much more frequently than the other ones.



(a) # of running tasks = [10,19] (b) # of running tasks = [20,29]

(c) # of running tasks = [30,39] (d) # of running tasks = [40,49]

Figure 9. Mass-Count of Duration in Unchanged Queuing State

By studying the number of running tasks, we realize that the duration in which the running-queue state is constant on each machine can be several hours on average. This means that the running-queue state is relatively stable.

The relative usage of a machine's resource is most relevant for load prediction. We define a machine's **resource usage** to be the ratio of the usage of some attribute to its capacity. The range of resource usage is thus in $[0, 1]$. We will also use **resource utilization**, **usage level**, and **load level** interchangeably. We will characterize in detail CPU and memory resource usage in the following sections.

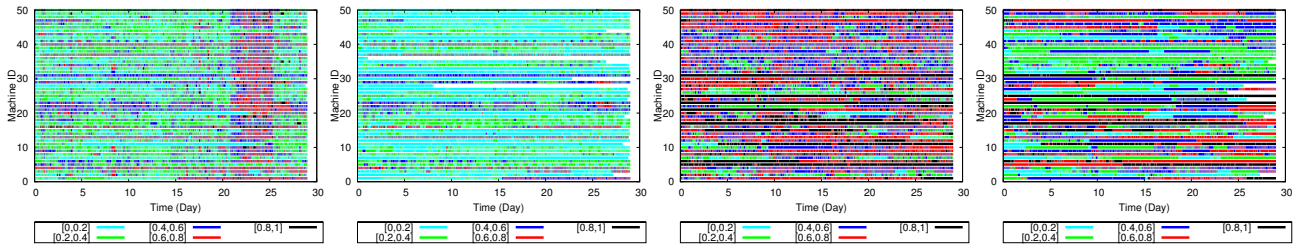
2) Machine's Usage Level: Based on the scaled CPU capacity information provided by the Google trace, we can compute the relative usage level (or load level) for the CPU and memory relative to their capacity on each machine. As follows, we present a snapshot of the relative CPU/memory usage on 100 randomly sampled machines, before discussing aggregate statistics and distributions.

In terms of the distribution of the number of tasks with different priorities (as shown in Figure 2), we split all jobs/tasks into three categories, with low-priority (1-4), middle-priority (5-8) and high-priority (9-12) respectively. For a particular attribute with a specific priority category (e.g., CPU usage of the mid+high priority tasks), we further divide the whole usage range into five equal intervals, and plot a trace of dynamically changing load levels over time for 50 randomly sampled machines. This depicts the system's idleness level from the perspective of the tasks with different priorities.

From Figure 10 (a)$\sim$(d), we observe that CPU and memory usage differ. Specifically, Figure 10 (a) shows that most of the machines are relatively idle compared to their capacities in most of time (from the beginning to the 21th day and from 25th day till the end). Such an idle situation is attributed to the intention by reserving a certain portion of CPU resources to meet service level objectives (for instance, a threshold on web request latency) in case of unexpected load spikes. From the perspective of high-priority tasks, however, the percentage of host load is not that heavy, as shown in Figure 10 (b). This means that most of the CPU resources from the Google computer cluster are actually consumed by low-priority tasks, especially in the busy duration (from 21th day to 25th day). Hence, it is relatively easy to allocate idle CPU resources for the tasks with high priorities.

Figure 10 (c) and (d) present the snapshot of the load changes about memory usage. We can see that the majority of machines' memory usage is high compared to their capacities. Moreover, their fluctuations are quite unlike that of CPU usage. Specifically, some machines' memory usage are always relatively lightly-loaded (the green lines shown in the figure); some are often heavily loaded (the black and red lines); some memory usage always alternate between two load levels; and there also exist a few machines that demonstrate completely irregular changing memory usage.

We use Table II and Table III to show the statistics the durations where CPU or memory usage are constant. The



(a) CPU Usage with All Tasks (b) CPU Usage with Hi.Pr. Tasks (c) MEM Usage with All Tasks (d) MEM Usage with Hi.Pr. Tasks

Figure 10. Snapshot of Resource Usage Load

statistics are computed using all the tasks (including all priorities) running on all of machines. It is obvious that the continuous duration of CPU load approximately conforms to 30/70 rule, while memory's conforms to 20/80 rule. Moreover, the frequency of the CPU load changes is very high, in that the average duration is only about 6 minutes and the mm-distance is also small (18∼49 minutes).

Due to the limited space, we cannot show the complete statistics from the perspective of middle/high-priority tasks, but just present a few as follows: As for CPU usage, the uninterrupted load duration is 7∼8 minutes on average, and the joint ratio is always about 30/70, in the situation where only middle/high-priority tasks or high-priority tasks are taken into account; as for memory usage, the joint ratios are about 20/80 and 15/85 for the middle/high-priority tasks and high-priority tasks respectively. In particular for high-priority tasks, CPU's mm-distance mainly appears in 14∼40, except for [0,0.2] whose mm-distance is 371, while memory's is in [560,1700], which clearly indicates that the CPU usage changes much more frequently than that of memory usage over time.

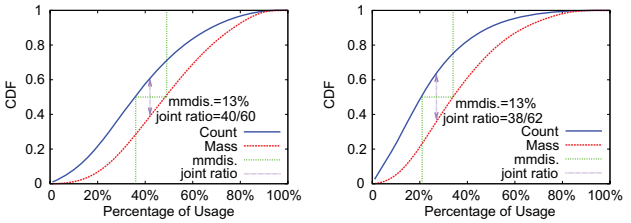
Table II
CONTINUOUS DURATION OF UNCHANGED CPU USAGE LEVEL

	[0,0.2]	[0.2,0.4]	[0.4,0.6]	[0.6,0.8]	[0.8,1]
avg value (minute)	6	6	6	6	5
max value (minute)	41269	40302	3590	1452	2575
joint ratio	26/74	28/72	30/70	30/70	27/73
mm-distance (minute)	49	25	18	19	24

Table III
CONTINUOUS DURATION OF UNCHANGED MEMORY USAGE LEVEL

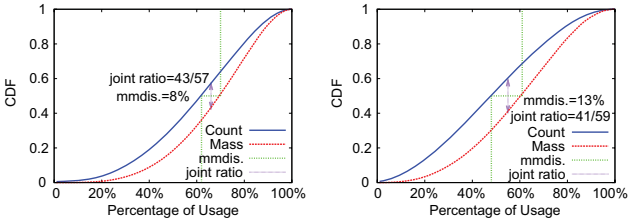
	[0,0.2]	[0.2,0.4]	[0.4,0.6]	[0.6,0.8]	[0.8,1]
avg value (minute)	6	9	10	10	10
max value (minute)	31556	40302	10996	16736	16826
joint ratio	20/80	23/77	26/74	23/77	18/82
mm-distance (minute)	119	83	63	95	351

Now, we focus on the distribution of the mass-count disparity of the relative resource utilization whose values range from 0% to 100%. Figure 11 and Figure 12 present such statistics about CPU usage and memory usage respectively. By comparing the two figures, we can further confirm that the CPU usage is much lower than memory usage, relatively. Specifically, the percentage load of CPU is about 35% w.r.t. all the tasks and about 20% for the high-priority tasks, while memory's are about 60% and 50% respectively. In addition, we can also know that the distribution of usage is relatively uniform because of the small value of mm-distance and large value of joint ratio.



(a) low+mid+high priority tasks (b) high priority tasks

Figure 11. Mass-Count Disparity of CPU Usage



(a) low+mid+high priority tasks (b) high priority tasks

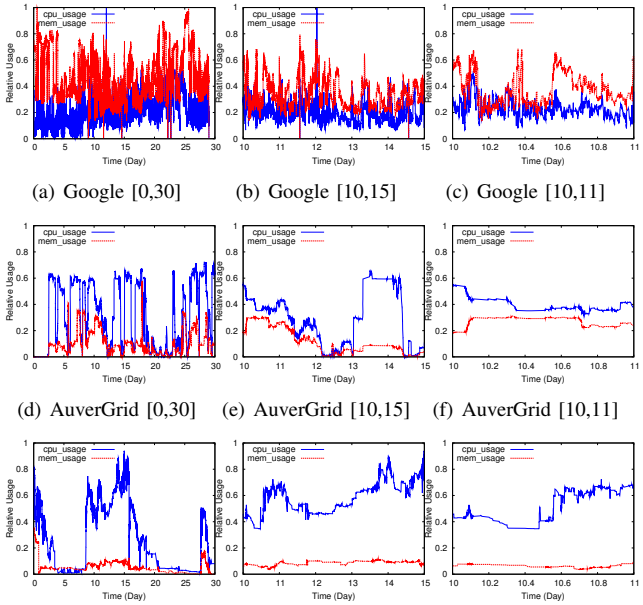
Figure 12. Mass-Count Disparity of Memory Usage

Finally, we explore the difference of load changes between Cloud and Grid, by comparing the resource usage of Google's compute clusters and those of AuverGrid [2] and SHARCNET [4], whose trace data can be downloaded from the Grid Workload Archive (GWA) [9].

In Figure 13, we present the CPU usage load of three machines, selected from the Google cluster, AuverGrid and SHARCNET, within one month, 5 days and 1 day respectively. What is the most interesting is that Grid's CPU usage is always higher than memory usage. This because most Grid jobs are computation-intensive. By contrast, in the Google cluster, a machine's CPU usage is usually lower than memory usage. This implies that Google tasks are not compute-intensive programs and they use other resources such as memory more intensively.

The second key difference observed is that Google cluster's CPU load has higher noise than the other two Grid systems. We measure noise by processing the trace with a mean filter [14], and then computing statistics on the

transformed trace. The minimum/mean/maximum noise of AuverGrid's CPU load are 0.00008, 0.0011, 0.0026 respectively, while those for the Google cluster's CPU load are 0.00024, 0.028, 0.081 respectively. It is clear that the noise of Google cluster's usage load is about 20 times as large as that of Grid's on average.



(a) Google [0,30] (b) Google [10,15] (c) Google [10,11]

(d) AuverGrid [0,30] (e) AuverGrid [10,15] (f) AuverGrid [10,11]

(g) sharcnet [0,30] (h) sharcnet [10,15] (i) sharcnet [10,11]

Figure 13. Host load Comparison between Google cluster & Grid Systems

Lastly, Figure 13 also shows that the host load of AuverGrid and HARCNET is more stable during relatively long periods (e.g., a few hours), while Google cluster's load changes rather frequently even in quite short periods, tens of minutes in length. Over all hosts, the mean autocorrelation of CPU usage load is about -8×10^{-6}, which is much smaller than AuverGrid's CPU autocorrelation (=1.006×10^{-8}). This is mainly due to the fact that Google cluster has to process millions of tasks, a large majority of which are smaller than in Grids, as shown in Figure 8 (a) and Figure 4. The fact that majority of Google's tasks are smaller will definitely lead to much finer resource usage and introduce more instability in terms of host load fluctuation. Moreover, as described in [15], Cloud tasks' placement constraints may also be tuned by users frequently over time, which may further impact the resource utilization significantly. All in all, it is more challenging to predict Google cluster's host load because of its higher noise and more unstable state.

V. Related Work

There is much related work on the characterization of workload and host load for Grid systems. H. Li [16] studied the workload dynamics on clusters and Grids, based on the real-world trace provided by Grid Workload Archive (GWA) [9]. His work revealed many features about Grid load, which can be leveraged to improve the system performance. Specifically, the Grid host load exhibits clear periodic or diurnal patterns, which can be used for host load prediction for the task scheduling [17]. E. Afgan et al. [18] exploited the performance characteristics of the NIBC BLAST application on a cluster environment. In our research, we comprehensively compare the workload between Cloud (Google) and other Grid systems, showing much shorter job length and much higher submission frequency in Cloud systems. D. Kondo [19] analyzed the resource availability of Desktop Grid systems based on the traces collected from different Grid softwares, including the Entropia DCGrid desktop grid [20], XtremWeb desktop grid [21] and so on. Our key contribution is to exploit the key insights about the workload and host load between Cloud and Grid systems, serving as a fundamental basis for the system performance improvement over Cloud systems.

Despite some works [15], [22], [23] also characterizing workload features for Cloud systems, they mainly focus on modeling running tasks such as tasks' placement constraints [15] or usage shapes [22]. Specifically, B. Sharma et al. [15] carefully studied the performance impact of task placement constraints based on the resource utilization from the view of tasks. Q. Zhang et al. [22] designed a model that can characterize task usage shapes in Google's compute clusters. In comparison to these two works, we intensively compare the load characteristics between Cloud and Grid systems, and summarize many new insights about Cloud load.

A. Khan et al. [24] designed a model to capture the CPU load auto-correlations in the Cloud data centers by leveraging the Hidden Markov Model (HMM). B.J. Barnes et al. [23] introduced a regression-based approach for predicting parallel application's workload, and the prediction errors can be limited between 6.2% and 17.3%. In comparison, by using Google cluster's large-scale one-month trace data, we comprehensively studied the host load changes about multiple resource attributes (including CPU, memory and page cache) in Cloud data centers. Specifically, we characterize the machine capacity, queuing size (the number of running tasks), the impact of task priority and task events, usage levels and so on.

VI. Conclusion and Future Work

We characterize the work load and host load in a Google data center and compare them to those of Grid systems. We use a detailed workload trace of a Google data center with over 25 million tasks executed over 12,000 hosts in a 1-month time frame, and the trace data from many Grid/HPC systems. Our key findings are summarized below:

- *With respect to Work load of Job or Task:*
  - *Job/Task length*: Most jobs finish in tens of minutes and tasks finish quickly on the order of a few minutes. Statistics show that about 55% of tasks finish within 10 minutes and about 90% of tasks' lengths are shorter than 1 hour. A handful of tasks

last for several days or weeks and likely correspond to long-running services. Compared to Grid workloads, most Cloud task lengths are shorter, and at the same time the longest task lengths are longer. We believe this difference is due to the differences in Cloud users and applications, which include commercial applications such as web services. This indicates that load can be sporadic and highly variable.
  - *Job Priority*: We find that task priorities can be clustered into 3 groups, namely, low, medium, and high. As high priority task can preempt low priority task, load prediction thus can and should be tailored and evaluated for each of these groups.
  - *Job Submission Frequency*: Google jobs are submitted with much higher and more stable frequency than that of Grid jobs. The average number of jobs submitted per hour and its fairness index in Google are 552 and 0.94 respectively, compared to 8.4$\sim$126 and 0.04$\sim$0.51 in Grids.
  - *Job Resource Utilization*: Google jobs usually have lower resource demand for CPU and memory than Grid jobs, because Google jobs (such as keyword search) are more interactive and real-time in constrast to scientific batch jobs.
- *With respect to Host load*:
  - *Maximum load*: We find that Google host's maximum CPU load is often close to the CPU capacity, and the maximum memory usage is about 80% of the memory capacity. The maximum load is actually controlled in the Google system for guaranteeing the service level of requests in case of unexpected load spikes. In contrast, Grid resources can be highly utilized without having a high risk of losing users or customers.
  - *Machine usage level*: CPU and memory usage changes every 6 minutes, indicating again the volatility of load. CPUs are often idle, but memory usage is relatively high. CPU usage in Grids is higher and more stable. Noise of CPU load in the Google cluster is 20 times as high as that in Grids.

In the future, we will try to exploit the best-fit load prediction method based on our characterization work, and analyze and improve the job scheduling in Google data centers.

ACKNOWLEDGMENTS

We thank Google Inc, in particular Charles Reiss and John Wilkes, for making their invaluable trace data available. This work was made possible by a Google Research Award and by the ANR project Clouds@home (ANR-09-JCJC-0056-01).

REFERENCES

[1] Google cluster-usage traces: online at http://code.google.com/p/googleclusterdata.
[2] Auvergrid: online at http://www.auvergrid.fr/.
[3] Nordugrid: online at http://www.nordugrid.org/.
[4] Sharcnet grid project: online at https://www.sharcnet.ca.
[5] Anl cluster: online at http://www.anl.gov/.
[6] Ricc cluster: online at http://accc.riken.jp/ricc_e.html.
[7] Metacentrum cluster: online at http://www.metacentrum.cz/cs/.
[8] Llnl: online at https://www.llnl.gov/.
[9] Grid workloads archive (gwa): online at http://gwa.ewi.tudelft.nl/pmwiki/.
[10] Parallel workload archive (pwa): online at http://www.cs.huji.ac.il/labs/parallel/workload/.
[11] D. G. Feitelson, *Workload Modeling for Computer Systems Performance Evaluation*, 2011. [Online]. Available: http://www.cs.huji.ac.il/~feit/wlmod/
[12] R. Koch, *The 80/20 principle: the secret of achieving more with less*. Nicholas Brealey, 1997.
[13] R. K. Jain, *The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation and Modelling*. John Wiley & Sons, April 1991.
[14] P. S. R. Diniz, *Adaptive Filtering: Algorithms and Practical Implementation*, softcover reprint of hardcover 3rd ed. 2008 ed. Springer, Oct. 2010.
[15] 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 (SOCC'11)*. New York, USA: ACM, 2011, pp. 3:1–3:14.
[16] H. Li, "Workload dynamics on clusters and grids," *The Journal of Supercomputing*, vol. 47, no. 1, pp. 1–20, Jan. 2009.
[17] H. Li, R. Heusdens, M. Muskulus, and L. Wolters, "Analysis and synchesis of pseudo-periodic job arrivals in grids: A matching pursuit approach," in *7th IEEE International Symposium on Cluster Computing and the Grid (CCGrid07)*, 2007, pp. 183–196.
[18] E. Afgan and P. Bangalore, "Exploiting performance characterization of BLAST in the grid," *Cluster Computing*, Feb. 2010.
[19] D. Kondo, G. Fedak, F. Cappello, A. A. Chien, and H. Casanova, "Characterizing resource availability in enterprise desktop grids," *Future Gener. Comput. Syst.*, vol. 23(7), pp. 888–903, 2006.
[20] B. Calder, A. A. Chien, J. Wang, and D. Yang, "The entropia virtual machine for desktop grids," in *VEE*, 2005, pp. 186–196.
[21] C. Germain, V. Néri, G. Fedak, and F. Cappello, "Xtremweb: Building an experimental platform for global computing," in *1st ACM/IEEE International Conference on Grid Computing (Grid'00)*, 2000, pp. 91–101.
[22] Q. Zhang, , J. L. Hellerstein, and R. Boutaba, "Characterizing task usage shapes in google compute clusters," in *Large Scale Distributed Systems and Middleware Workshop (LADIS'11)*, 2011.
[23] B. J. Barnes, B. Rountree, D. K. Lowenthal, J. Reeves, B. de Supinski, and M. Schulz, "A regression-based approach to scalability prediction," in *Proceedings of the 22nd annual international conference on Supercomputing (ICS'08)*. New York, NY, USA: ACM, 2008, pp. 368–377.
[24] A. Khan, X. Yan, , S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *3rd IEEE/IFIP International Workshop on Cloud Management (Cloudman'12)*, 2012.