Data-Driven Workload Generation Based on Google Data Center Measurements

Mert Yildiz and Andrea Baiocchi
DIET, Universita di Roma Sapienza, Rome, Italy
mert.yildiz@uniroma1.it, andrea.baiocchi@uniroma1.it

2024 IEEE 25th International Conference on High Performance Switching and Routing (HPSR) | 979-8-3503-6385-2/24/$31.00 ©2024 IEEE | DOI: 10.1109/HPSR62440.2024.10635925

***Abstract*—A large dataset of workload measurements has been released by Google. The wealth of disclosed data allows a deep dive into real workload patterns. With the aim of providing tools to generate realistic workloads in a simple way, we have extracted from Google's dataset job arrival times, number of tasks per job, required computation time, and memory of tasks. We define a statistical fitting of the relevant probability distribution, providing a simple tool to build artificial workload traces that mimic real traffic as represented by Google measurements. The workload generation algorithm is assessed by comparison of its mean response time on a test dispatching/scheduling system against the real traffic traces. In spite of being only a first-order generation model, it is shown that the proposed artificial generation can reproduce faithfully the performance of real workload in the case of large server clusters.**

***Index Terms*—Workload modeling, data centers, traffic measurements, data fitting, large server clusters**

I. Introduction

Assessing and comparing dispatching and scheduling policies on computational clusters in data centers requires the use of a realistic workload to feed the system. Theoretical models highlight the criticality of mixing large and smaller jobs in a multi-server environment, e.g., see [1], [2], [3]. This motivates devising ways to generate realistic workload to compare and optimize proposed dispatching and scheduling policies. Measured workload data provided by Google in 2011 [4], [5], [6] and 2019 [7], [8] represents a major contribution to the study and understanding of real workload patterns. In fact, little information is provided as to which specific applications run the captured jobs and tasks, yet it is stated that this is production traffic.

A wealth of features and detailed trace measurements are provided in Google's datasets. This huge repository does not easily lend itself to be employed directly into the simulations of dispatching/scheduling policies. With some simplifications, the dataset can be viewed as a collection of jobs, each of which comprises one or more tasks. To define the workload, we need to characterize: (i) job and task arrival times; (ii) the structure of jobs, i.e., how many tasks belong to a single job, what are the dependencies among tasks of the same job, if any; (iii) the amount of resources required by each task (CPU, memory).

Initial studies of 2011 trace data, documented in [5], analyzed task inter-arrival times, makespan, scheduling categories, and priorities, excluding resource demand. Borg, Google's scheduler [8], employs a complex job priority system. Priorities are strongly tied to factors like application nature, execution cost, and job significance, which are not documented in detail. Makespan measurement incorporates the effect of Borg's scheduling decisions. Workload characterization should instead be based on mere task resource demand.

In [9], the authors applied coarse-grained analysis and clustering, including k-means, for Google MapReduce workload modeling, focusing on task attribute selection and k-value determination for accurate classification. Lin et al. [10] proposed a GAN-based method to create cloud workloads, targeting precise machine/server workload replication. In [11], the authors utilize a CGAN-based technique for producing time-sensitive workloads, adeptly modeling the intricate interactions and dependencies among jobs, including their arrival patterns. [12] similarly generates workloads based on the hourly rate of virtual machine requests within cloud data centers. While GANs are powerful tools, their training is critical and their use requires intensive computation. Our emphasis in this work is instead on exploring the potential of simple statistical tools.

In this work, we define a simple workload generation model able to produce artificial workload traces that possess the same first-order characteristics of real workload, as appearing from Google datasets. Two problems have been addressed, posing the following research questions: (i) how can we compress the wealth of information provided by Google, extracting the most relevant features for workload generation? (ii) What are "relevant" features, i.e., how do we assess that the generated traffic trace is consistent with real data?

As for the first question, carried out an extensive study of the correlation structure of quantities such as task computation time and memory requirement [13]. Some results are reported in Section V. In all examined cases tasks belonging to the same job start all at the same time. There is also no evidence of dependencies among tasks of the same job. hence, we identify a four-dimensional descriptor of the workload: job arrival time, number of tasks making up a job, task computation time, and maximum memory requirement of a task. After giving an extensive account of distributions of such quantities across all eight Google traces (one for each observed data center), we propose a simple statistical fitting of real data.

Regarding the second question, the artificially created workload is designed to align with the real data in terms of first-order statistical properties. Since the target of the workload generation

This work was partially supported by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, in partnership with Telecommunications of the Future (PE0000001 - program "RESTART").

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on October 27,2026 at 15:08:33 UTC from IEEE Xplore. Restrictions apply.

model is to feed simulations and performance evaluation of scheduling policies, we assess its adequacy by comparing the mean response time resulting from the application of real workload traces and artificially generated one to a test multi-server system, as the system load and system size (number of servers) are varied. It turns out that the statistical first-order generation model matches the performance achieved by real data perfectly provided we consider large systems (in the order of thousands of servers). Those are also the system settings that provide the best results in terms of mean response time. Hence this is the most interesting range of parameters from a system engineering point of view. It is, therefore, an interesting outcome of this study that we apparently do not need a highly sophisticated machine learning model to match real workload, as long as we use artificial data traces to optimize the mean response time of dispatching and scheduling algorithms, which calls for using large scale server clusters to tame workload variability.

The rest of the paper is organized as follows. Section II gives a concise outline of the Google measurement environment and dataset structure. Section III gives an account of the statistical analysis of the Google workload features selected to represent the workload. The workload generation algorithm is presented in Section IV and its assessment against real data in Section V. A final discussion closes that Section.

II. Google Dataset

Google has published the cluster usage traces v3 dataset which provides information about 8 different Borg cells for the month of May 2019 [4], [7]. These 8 Borg cells are in different time zones such as New York, Helsinki, and Singapore [8]. Each Borg cell consists of 5 tables and comprises three main types of tables: Machine tables, Collection tables, and Instance tables. Specifically, Machines are detailed in two distinct tables: the *MachineEvents* table outlines the events related to machines, while the *MachineAttributes* table provides specifications on machine characteristics. Google calls jobs and alloc sets as collections, they refer to tasks and alloc instances as instances, and use things to refer to collections or instances. The life cycles of collections and instances are captured in the *CollectionEvents* table and *InstanceEvents* table, respectively.

When it comes to alloc instances, the reported resource usage encompasses all tasks operating within the instance for each recorded sampling period, as documented in the *InstanceUsage* table. In terms of scale, Machine tables contain thousands of entries. The *CollectionEvents* table expands significantly to millions of entries, detailing essential information about jobs/collections. Meanwhile, the Instance tables scale up even further to billions of records, offering detailed insights into tasks/instances.

A. Common Techniques and Fields

The 2019 traces focus on resource requests and usage and contain no information about end users, their data, or access patterns to storage systems and other services.

Each record has a timestamp which is in microseconds, relative to a specific point 600 seconds before the start of a trace period. The timestamps are stored as 64-bit integers. As a matter of example, an event occurring 20 seconds after the trace starts would have a timestamp of 620 seconds. There are also two special timestamp values for events outside the trace window: a timestamp of 0 for events before the trace period and a timestamp of $2^{63} - 1$ (MAXINT) for events after the trace period. Usage measurement times, capped at 300 seconds, follow the same offset rules to distinguish events relative to the trace period, even though these measurements are to the nearest second but reported in microseconds for consistency. The accuracy of these timestamps is generally reliable, though minor discrepancies may arise from clock drift among different machines in a cluster, not from scheduling or utilization issues.

Borg measures RAM in bytes and normalizes these values by dividing by the maximum machine memory size observed in the traces, setting the machine with the largest memory as a baseline of 1.0.

CPU resources are measured in "Google compute units" (GCUs), with one GCU equaling the computing power of one CPU core on a standard machine.

The data documentation introduces Normalized Compute Units (NCUs) for both CPU and memory, scaling them within the range of [0,1]. Tasks run in Linux containers for resource isolation, and Google provides usage data based on non-overlapping measurement windows, typically 5 minutes long. CPU and memory usage is sampled roughly once per second, with various metrics (average rate, maximum rate, and percentiles for CPU; average and maximum consumption for memory) normalized to NCUs.

Another vital aspect is job/task priority. In Google's system, entities are assigned priority levels, represented by small integers where 0 is the lowest. Higher-priority entities receive preferential resource allocation over lower-priority ones. Priorities are categorized into tiers: *Free-tier* (≤ 99) has minimal charges and weak or no Service Level Objectives (SLOs) for resource allocation. *Best-effort Batch* ($100 - 115$) is managed by the batch scheduler with low charges and no SLOs. *Mid-tier* ($116 - 119$) has intermediate SLOs between free and production levels. *Production-tier* ($120 - 359$) is the highest normal use priority, avoiding eviction of latency-sensitive tasks. *Monitoring-tier* (≥ 360) is for tasks monitoring lower-priority tasks' health.

The final aspect to consider is the life cycles and event types associated with jobs and tasks within the system. Initially, a job or task is submitted to the system. Subsequently, it may either be processed immediately or placed in a queue, depending on the presence of existing jobs or tasks within the queue. Upon completion, tasks are categorized based on their outcome. Google categorizes successfully completed tasks as "FINISH". Conversely, tasks that do not complete successfully are labeled as "KILL", "LOST", or "FAIL", depending on the specific nature of the unsuccessful completion.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on October 27,2026 at 15:08:33 UTC from IEEE Xplore. Restrictions apply.

III. Data Analysis

This section summarizes the insights stemming from an analysis of the dataset. The dataset records a trace of requests for computational resources by users or developers, tracking jobs that may consist of multiple tasks. It provides data on job arrival times, tasks' start and end times, and the CPU and RAM resources requested and utilized. A comprehensive analysis of these key features is detailed, with findings to be discussed in the following sections.

A. Preprocessing

The initial step in deriving meaningful workload profiles from the dataset involves data cleansing. However, it is advisable to first review the quantitative metrics of all data centers, considering the combined data volume of the eight data centers amounts to 2.4TB. In Table I, the first column identifies each data center, referred to as a Borg cell in Google's terminology. The second column lists the total number of jobs (NoJ), and the third column details the total number of tasks (NoT). The fourth column provides a pair of values indicating the average and the maximum number of tasks associated with each job. The final two columns denote the average CPU time consumed by a task (in seconds) and the average maximum memory utilization by the tasks, respectively.

Table I
Data Centers' Quantitative Information

	NoJ	NoT	NoTPJ (mean,max)	CPU (sec)	Mem
a	5,182,091	74,490,982	(14, 97,089)	3.96	0.00150
b	1,976,871	82,429,912	(41, 124,870)	31.07	0.00259
c	7,303,369	301,945,886	(41, 104,637)	6.62	0.00168
d	2,772,507	135,974,832	(49, 118,399)	35.30	0.00363
e	1,307,619	66,730,755	(51, 134,285)	193.26	0.00326
f	1,272,348	56,116,318	(44, 177,803)	29.93	0.00199
g	1,018,740	23,374,889	(22, 174,162)	148.76	0.00393
h	3,280,618	35,308,060	(10, 153,654)	407.20	0.00351

To ensure our analysis's accuracy and reliability, we conducted a preliminary review of the trace data to identify and eliminate anomalies. Specifically, to remove the border effects of the finite traffic trace, tasks that started and ended across the boundaries of the observed time window (the month of May 2019) were excluded. Additionally, tasks exhibiting zero CPU and/or zero memory usage were also omitted, as those anomalous values of consumed resources could not be explained from the available documentation. The tasks removed for the above reasons amount to less than 1% of the overall number of tasks.

In the preceding section, we noted that Google marks tasks that have successfully completed with the status "FINISH." Tasks labeled as "KILL," "LOST," or "FAIL" also have the record of duration, CPU, and memory usage. However, it is uncertain whether these values accurately reflect the task's actual requirements. Consequently, we have limited our analysis exclusively to tasks labeled as "FINISH".

After filtering out these data, we proceeded to examine the fundamental features of the remaining tasks. All subsequent analyses and modeling have been conducted based on this filtered dataset. The resulting dataset extracted from Google public data consists of eight `csv` files, one for each data center of Google involved in the measurement campaign. Each `csv` file is a five-column table, whose columns are defined as follows[1].

1) J: integer number representing the identifier of a job.
2) I: integer number between 0 and n_j, representing local identifiers of tasks belonging to job $J = j$.
3) A: arrival time of task measured in seconds[2].
4) C: running time in seconds required to run the task on a Google Normalized Computing Unit (GNCU).
5) M: maximum amount of memory required to run the task, expressed in Google Normalized Memory Unit (GNMU).

Quantities relevant to task $I = i$ of job $J = j$ will be denoted with a double index, e.g., $A_i(j)$ is the arrival time of task i belonging to job j. The reasons why for this representation of the workload will be clarified in subsequent Sections.

B. Analysis of Data Features

Scanning task arrival times it is apparent that tasks belonging to the same job arrive at the same time[3]. Therefore, we refer to *job* arrival times as the (common) value of arrival time of any of the tasks of the considered job. Figures 1a and 1b show the empirical Probability Density Function (PDF) (solid line curve) of the Inter-Arrival Times (IATs) of jobs for two Google data centers named cell 'a' and 'b'.

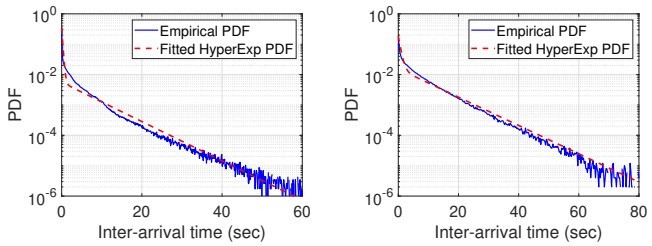


Figure 1. PDFs of inter-arrival times (a) of cell 'a'. (b) of cell 'b'.

IATs cover a wide range of values, spanning from microseconds (almost simultaneous arrivals, given the time resolution of the measurement) up to hundreds of seconds. After an initial steep decrease, an almost constant slope line emerges, giving evidence of exponential decay of the IAT PDF (note that the y-axis is logarithmic). The far right end of the PDF tail is affected by increasing estimation noise so that we cut the plot down to the probability level 10^{-6}.

Although there were visible variations across different cells, further examination demonstrates that a unified PDF can accurately fit the data for all cells.

[1]The dataset extracted for all eight Google data centers and organized according to the definitions above are published in https://github.com/MertYILDIZ19/Google_cluster_usage_traces_v3_all_cells.

[2]In the dataset published in GitHub arrival times are in milliseconds.

[3]Arrival times differ by less than 1 μ s, which is the time resolution of time measurements.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on October 27,2026 at 15:08:33 UTC from IEEE Xplore. Restrictions apply.

The PDF of the task count per job is plotted in Figure 2 for all Google data centers. The patterns across the data centers exhibit remarkable consistency, with the bulk of jobs comprising one or two tasks. A more detailed investigation into jobs with two tasks revealed that these tasks typically work independently of each other. It can be checked that in 99.86% of instances where a job has exactly 2 tasks, the second task starts before the first has concluded. This leads us to determine that assuming no dependency among tasks belonging to the same job is acceptable, also considering that about 96% of jobs have a single task. Hence, only a small minority of jobs have multiple tasks, even though a few exceptional outlier jobs may scale up to several thousand tasks.

Building on our prior discussions, the next focal point is the overall CPU usage and maximum memory requirements of tasks. The Google dataset includes key metrics such as *average_usage*, which represents the mean usage over a 300-second scheduling window; *maximum_usage*, indicating the recorded maximum usage during this window; and *random_sampled_usage*, detailing CPU usage observed during a randomly chosen 1-second sample within the window. For long-duration tasks, this dataset enables the creation of a highly accurate stochastic model for CPU usage [8].

For memory usage, it may not be immediately clear which metrics to prioritize for generation. Therefore, we analyzed the average, maximum, and sum of the averages to identify correlations and their importance in the generation process. After thorough consideration, we selected the maximum of *average_usage* values for each task. This approach ensures task completion without failures by accounting for the maximum memory usage, as a baseline level of memory is essential.

In terms of CPU considerations, we have quantified computational consumption by calculating the lifetime CPU utilization. This was achieved by multiplying the average CPU usage by the makespan of the task, conceptualizing it as the service time since a task will consume that amount of CPU for its entire duration. We have examined the Cumulative Distribution Function (CDF) for both overall CPU utilization and maximum memory requirements across all cells. These CDFs are depicted in Figure 3 and Figure 4, respectively.

It is apparent from these figures that a single distribution can capture the empirical CDFs estimated from data of each Google data center. Also, the shape of the CDF curve shows that there is a phase transition behavior, i.e., a region where the CDF has a steep increase, from negligible values to values very close to 1. A third remark, emerging from the plots in Figures 3 and 4 is that CPU usage and maximum memory values span a vast range of values, covering about six orders of magnitude (note that x-axis has a log-scale). This is yet another manifestation of a phenomenon that arises in many traffic scenarios, namely the existence of "mice" and "elephants". That is to say, many jobs and their component tasks are relatively small, but a non-negligible fraction of tasks are huge, i.e., their size is hundreds or thousands of times the average task size.

IV. Workload Generation Algorithm

Machine learning has recently been proposed for computational workload generation [10]. Even though they are achieving good accuracy in mimicking the real data, there is room for applying classic statistical inference to generate real-life data. This approach is interesting since it is fully explainable, being traditional white-box modeling. We aim at an especially simple modeling, that captures first-order features of the workload, namely, inter-arrival times, number of tasks per job, CPU time required by tasks, and maximum memory required by tasks.

As illustrated by curves depicted in Figures 1a and 1b, a consistent distribution trend is noted in the inter-arrival times across different cells. A good trade-off between the simplicity and flexibility of the model to capture multiple decay rates

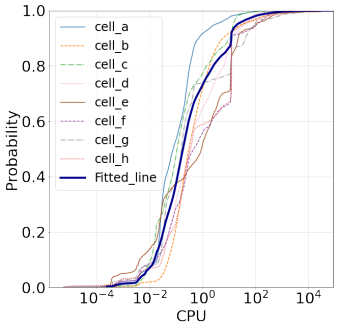


Figure 3. Fitted empirical CDF of CPU usage of all cells.

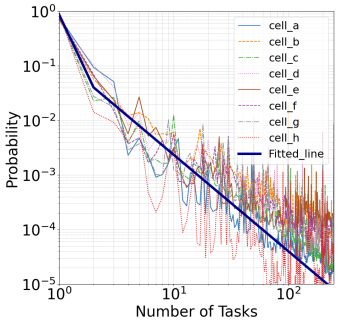


Figure 2. PDF of the number of tasks per job for all Google data centers.

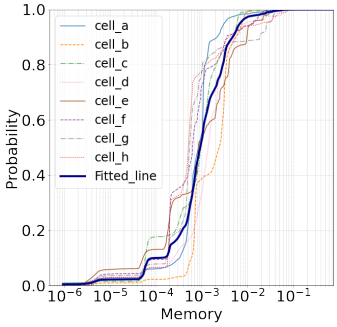


Figure 4. Fitted empirical CDF of memory usage of all cells.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on October 27,2026 at 15:08:33 UTC from IEEE Xplore. Restrictions apply.

as exhibited by the empirical PDF of inter-arrival times is the hyper-exponential distribution:

$$f(t) = \alpha\mu_1 e^{-\mu_1 t} + (1-\alpha)\mu_2 e^{-\mu_2 t} \quad (1)$$

A hyper-exponential PDF is fit to the sequence of IATs in each Google data center traffic trace. The fit is obtained by matching the following three features of the PDF:

- The mean $\alpha/\mu_1 + (1-\alpha)/\mu_2$.
- The decay rate of the right tail of the PDF with μ_2.
- The value of the density in a small interval (1/25 of the mean IAT) close to $t = 0$, where the highest value of the PDF is attained.

Task counts per job are mostly concentrated on $k = 1$ task, with the rest following a power law. We calculated the weighted mean likelihood of a job having one task and applied power law fitting for higher counts. This approach, detailed in Equation (2), differentiates between single-task jobs and those with more, using counts from all cells to fit the probability of a job having only one task.

$$P(k) = \begin{cases} \frac{\sum_{i=1}^{8} N_{i1}}{\sum_{i=1}^{8} N_i}, & \text{if } k = 1 \\ ak^{-b}, & \text{if } 2 \le k \end{cases} \quad (2)$$

where N_{i1} is the number of jobs consisting of 1 task in cell i and N_i is the overall number of jobs, for $i = 1, \ldots, 8$. The parameters a and b were derived matching the power law fit with the empirical PDF shown in Figure 2.

We condensed the data from eight cells into a single CDF for CPU and maximum memory usage. We calculated each cell's CDF for CPU as $F_i(x) = P_i(C \le x)$ and memory as $F_i(x) = P_i(M \le x)$ and aggregated into a single CDF $F(x) = \frac{\sum_{i=1}^{8} N_i F_i(x)}{N}$ with N_i as the number of tasks from cell i, and N is the total number of tasks from all cells. To simulate workload using the CDF, we:

- Draw a random number u uniformly distributed in $[0, 1]$.
- Output the maximum y such that $F(y) \le u$.

As depicted in Figure 3, our CDF approximation exhibits a good match with respect to the empirical CDFs of CPU usage and memory usage across all eight cells.

V. Assessment of Workload Generation and Discussion

To evaluate the accuracy of the proposed first-order generation model against actual data distributions, the Pearson correlation coefficient from [14] was used. This method assesses the alignment between the proposed model approximations and real-world data. By dividing the trace data into subsets matching the size of our synthetic data and averaging the correlation coefficients, we inferred the linear correlation between the generated and original datasets. Pearson correlation coefficients for all 8 cells are listed in Table II. The coefficients of IAT, number of tasks per job, CPU usage, and memory usage are shown in each row and are generally close to 1.

The data in Table II suggest that generated data closely matches real data under feature independence. However, CPU and memory usage are expected to be correlated. To investigate this, we implemented k-means clustering [15], using Overall CPU Usage and Maximum Memory as metrics. The Elbow method [16] identified 4 distinct clusters, shown in Figure 5. For example, one class corresponds to tasks with low CPU usage (green dots) and a uniform spread of memory usage. Conversely, tasks with high CPU usage (red dots) show distinct memory usage values, indicating that CPU and memory requirements cannot be generated independently.

Table II
Correlation of features between the distribution of real data and generated data.

	IAT	NoT	CPU	Memory
ρ_1	0.976	0.997	0.914	0.865
ρ_2	0.985	0.998	0.923	0.928
ρ_3	0.991	0.993	0.894	0.872
ρ_4	0.983	0.993	0.968	0.894
ρ_5	0.990	0.998	0.999	0.913
ρ_6	0.986	0.998	0.989	0.941
ρ_7	0.985	0.999	0.956	0.887
ρ_8	0.979	0.999	0.990	0.892

To further explore the correlation among features in the original data set, that would escape a first-order synthetic generation, we conducted experiments using a multi-server queueing system to compare the mean response times of tasks of the artificial and of the real traffic traces as a function of the utilization coefficients of servers ρ, as shown in Figure 6a. Simulation were run for a $N = 64$ servers operating under Least Work Left (LWL) dispatching of tasks across servers and First Come First Served (FCFS) scheduling. The coefficient of utilization is tuned by introducing a scale factor μ that ties task service time X. We set $X = C/\mu$, where C is the amount of CPU required on a server having 1 Google Normalized Computing Unit (GNCU), as derived from Google's datasets. This amounts to assigning to each of the N servers of the multi-server system a processing capacity of μ GNCUs.

The curve labeled "Shuffled real data" refers to experiments conducted with real data, where the ordering of job arrivals and the association between tasks and their respective service times were randomly shuffled, to destroy any structure in the data, while preserving exactly the first-order probability distributions of all features. It is apparent that the generated data yield

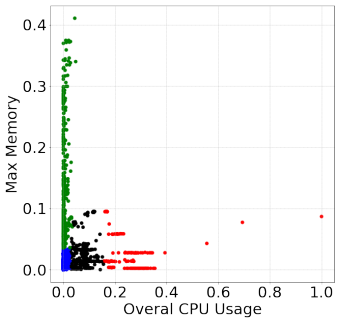


Figure 5. Clustered tasks by the overall CPU and maximum memory.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on October 27,2026 at 15:08:33 UTC from IEEE Xplore. Restrictions apply.

results that depart significantly from real data, although they match perfectly those of shuffled real data. This proves that our artificial generation captures first-order features with high accuracy, yet this may not be enough.

The results in Figure 6a point out that accounting only for a first-order probability distribution appears to be not enough for a good quality workload generation. However, Figure 6b shed more light on this issue, by showing mean job response time as a function of the number N of servers. Results obtained from real data and from artificial workload generation are compared. The curves are obtained by fixing the overall cluster sizes computation budget $N\mu$. This product is set so that the average coefficient of utilization of the system is 0.9. In other words, as the server cluster size N grows, the speed of each server decreases proportionally. We are thus exploring the trade-off between the degree of parallelism and individual server speed. It is apparent that, for relatively low values of N there is a big gap between the mean response time of the real and artificial workloads. This gap vanishes for large enough values of N. In other words, *parallelism appears to break correlations*, so that a simple first-order statistical model is perfectly able to reproduce the performance yielded by real workloads.

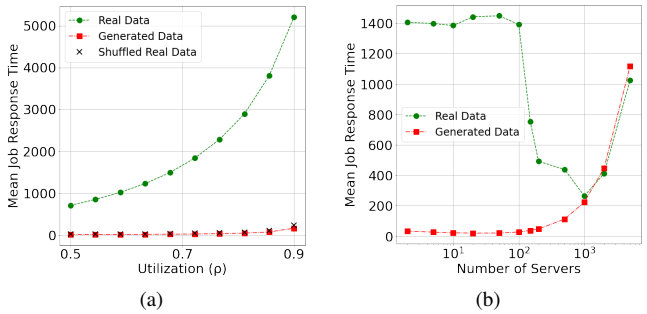
Most interestingly, the artificial workload generation starts gaining high accuracy exactly at the point where the mean response time of the real data has its minimum. In other words, if the system is being engineered to achieve the best mean response time performance, the considered range of system parameters is just that where artificial traffic matches the behavior of real workloads. These findings also point to interesting directions for future work.

VI. Conclusion

In this paper, we proposed a workload generation model by comparing synthetic data with real-world traces from Google data centers. While first-order statistical models can approximate metrics like inter-arrival times, the number of tasks per job, and resource usage, they fail to capture complex feature correlations. Leading to discrepancy in mean response time between real and synthetic data we conducted using a multi-server queuing system.

A key contribution of this work is providing a well-defined, simple-to-use model to generate a rich dataset of computational job workloads from real Google data center traffic. This dataset can aid in designing dispatching and scheduling algorithms and understanding the impact of real-life workloads on server clusters. Our findings indicate that as parallelism increases, resulting in reduced waiting times for arriving jobs, the discrepancy in mean response time diminishes. This allows simple models to more accurately replicate real-world performance, which is crucial for optimizing server clusters.

The model, dataset, and evaluation framework presented are valuable resources for future research. Future work could explore models that account for higher-order correlations and their impact on system design.



(a) (b)

Figure 6. (a) Mean response time of tasks vs utilization for real data, shuffled real data, and generated data. (b) Mean response time of jobs vs number N of servers.

References

[1] M. Harchol-Balter, "Open problems in queueing theory inspired by datacenter computing," *Queueing Syst. Theory Appl.*, vol. 97, p. 3?37, feb 2021.

[2] D. Olliaro, M. Ajmone Marsan, S. Balsamo, and A. Marin, "The saturated multiserver job queuing model with two classes of jobs: Exact and approximate results," *Performance Evaluation*, vol. 162, p. 102370, 2023.

[3] X. Zhou, F. Wu, J. Tan, Y. Sun, and N. Shroff, "Designing low-complexity heavy-traffic delay-optimal load balancing schemes:theory to algorithms," *Proc. ACM Meas. Anal. Comput. Syst.*, vol. 1, dec 2017.

[4] 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*, EuroSys '15, (New York, NY, USA), Association for Computing Machinery, 2015.

[5] G. Da Costa, L. Grange, and I. de Courchelle, "Modeling, classifying and generating large-scale google-like workload," *Sustainable Computing: Informatics and Systems*, vol. 19, pp. 305–314, 2018.

[6] J. Wilkes, "cluster-data/ClusterData2011." https://github.com/google/cluster-data/tree/master, 2011.

[7] M. Tirmazi, A. Barker, N. Deng, M. E. Haque, Z. G. Qin, S. Hand, M. Harchol-Balter, and J. Wilkes, "Borg: the next generation," in *Proceedings of the Fifteenth European Conference on Computer Systems*, EuroSys '20, (New York, NY, USA), Association for Computing Machinery, 2020.

[8] J. Wilkes, "cluster-data/Clusterdata2019." https://github.com/google/cluster-data/tree/master, 2019.

[9] X. Lin, P. Lin, P. Huang, L. Chen, Z. Fan, and P. Huang, "Modeling the task of google mapreduce workload," in *Proceedings of the 15th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing*, CCGRID '15, p. 1229–1232, IEEE Press, 2015.

[10] W. Lin, K. Yao, L. Zeng, F. Liu, C. Shan, and X. Hong, "A GAN-based method for time-dependent cloud workload generation," *Journal of Parallel and Distributed Computing*, vol. 168, pp. 33–44, 2022.

[11] J. Xu, J. Liu, J. Yao, A. Ma, L. Xu, and X. Zhao, "Conditional generative adversarial network based workload generation for cloud cluster," in *Proceedings of the 2022 5th International Conference on Algorithms, Computing and Artificial Intelligence*, ACAI '22, (New York, NY, USA), Association for Computing Machinery, 2023.

[12] F. Koltuk and E. G. Schmidt, "A novel method for the synthetic generation of non-i.i.d workloads for cloud data centers," in *2020 IEEE Symposium on Computers and Communications (ISCC)*, pp. 1–6, 2020.

[13] M. Yildiz, *Job and Task Workload Generator Based on Google Data Center Measurements for Scheduling Optimization*. Rome, Italy: Master's thesis, University of Rome Sapienza, 2023.

[14] I. Sfakianakis, E. Kanellou, M. Marazakis, and A. Bilas, *Trace-Based Workload Generation and Execution*, pp. 37–54. 08 2021.

[15] M. Ahmed, R. Seraj, and S. M. S. Islam, "The k-means algorithm: A comprehensive survey and performance evaluation," *Electronics*, vol. 9, no. 8, p. 1295, 2020.

[16] E. Umargono, J. E. Suseno, and S. V. Gunawan, "K-means clustering optimization using the elbow method and early centroid determination based on mean and median formula," in *Proceedings of the 2nd International Seminar on Science and Technology (ISSTEC 2019)*, pp. 121–129, Atlantis Press, 2020.

Authorized licensed use limited to: Universita degli Studi di Roma La Sapienza. Downloaded on October 27,2026 at 15:08:33 UTC from IEEE Xplore. Restrictions apply.