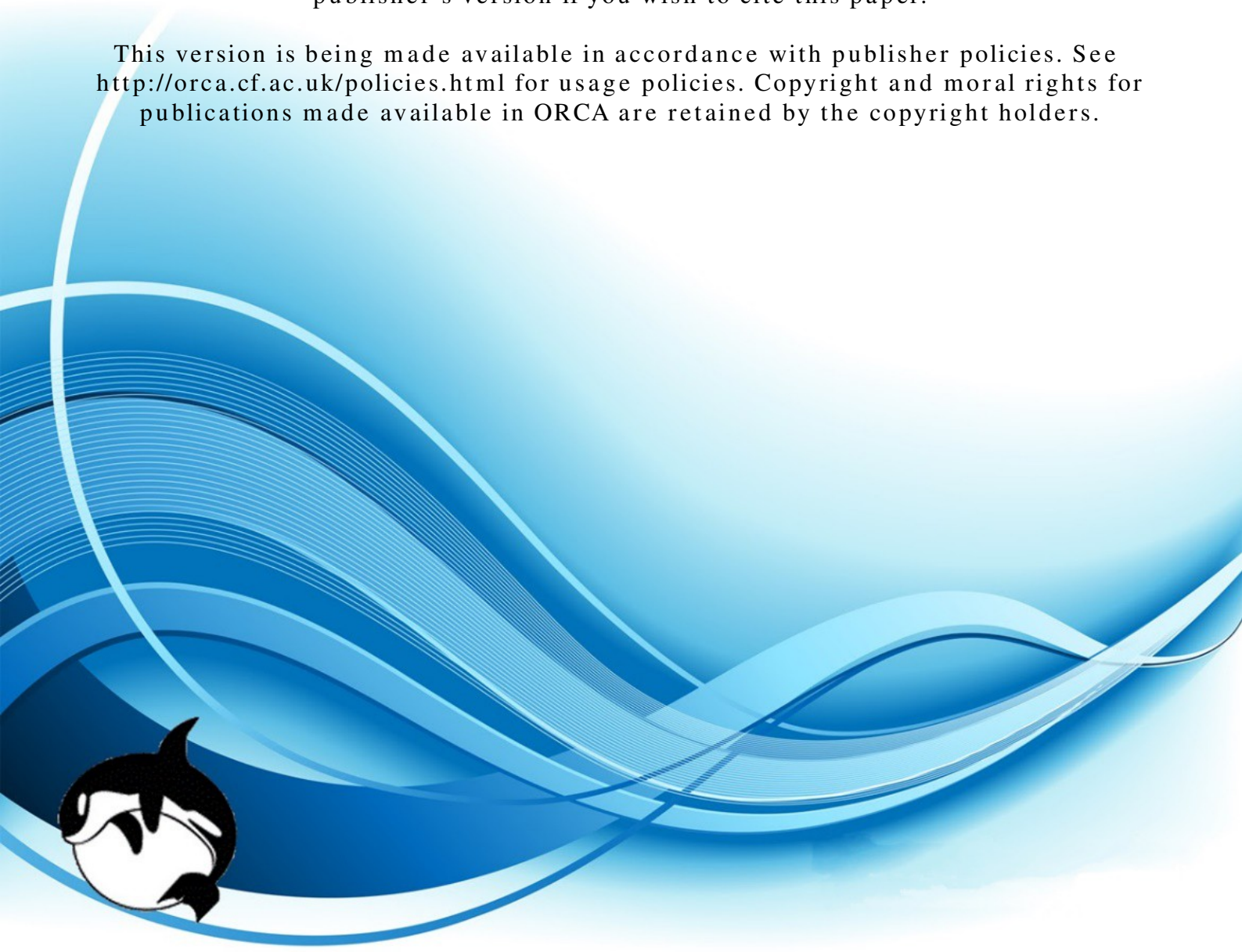

RESEARCH ARTICLE

Predicting Machine Behaviour from Google Cluster Workload Traces

Adnan Umer[1] | **Adnan Noor Mian**[2] | **Omer Rana**[3]

[1]Department of ML Engineering, Strong Analytics, Chicago, IL, 60611, USA

[2]Department of Computer Science, Information Technology University, Lahore, Punjab, 54000, Pakistan

[3]School of Computer Science, Cardiff Univeristy, Cardiff, CF24 4AG, UK

Correspondence
Adnan Noor Mian, Department of Computer Science, Information Technology University, Lahore, Punjab, 54000, Pakistan
Email: adnan.noor@itu.edu.pk

Funding information

Data centers today host a number of computational resources to support increasing demand for computation and storage. Understanding how these physical and virtual machines transition between different states of operation (referred to as machine lifecycle) enables more efficient job/task scheduling decisions to be carried out for user applications. Furthermore, it helps data center operators define policies on how new computational resources can be added or existing infrastructure decommissioned. Using Google cluster trace data set version3 collected from approximately 96k machines, we analyze machine failure and changes in machine lifecycle over time. We observed that there is a 13% chance of another machine failure under the same network switch within one minute of the previous machine failure. A Markov chain-based model is proposed, that can predict machine states at any given time. Using the model and estimated probabilities, we predicted machine state over a multi-day period with a high probability. Using predicted machine state, we reconstructed the active machines trend, comparing this with the trend reported in the data set – observing an error of 1.76%.

KEYWORDS
machine lifecycle, google cluster, machine state transition

1 | INTRODUCTION

The rising demand for storage and computing resources has compelled cloud service providers to upgrade and adjust their resource pool on a regular basis. Increasing requirement of accessing remote services has led to cloud providers updating/ decommissioning their computational infrastructure over time. As the data centers have to deal with larger and heterogeneous workloads, one of the major issues is how to efficiently manage the available resources while minimising operational cost. This aspect involves a number of considerations, such as: resource utilization optimization, optimal maintenance scheduling, etc – requiring a deeper understanding of machine dynamics and use. A better understanding of machine dynamics will also lead to more effective data center management policies. However, challenges in finding publicly available data sets have limited the potential studies on data center operations [1,2].

Google is committed to accelerating studies on data centers by regularly providing workload trace data sets. Google initially published this data set in 2010 [3], followed by an updated in 2011 [4,5]. In 2020, Google published v3 of this data set consisting of traces collected during May 2019 from approximately 96k machines belonging to eight different cells. Inside the cluster, machines can differ in hardware architecture and configuration. Machines of one particular hardware architecture and configuration are grouped and referred to belonging to the same *platform*. Machine behavior is somewhat dependent on the platform it belongs to – some requiring more maintenance than others.

Our work focuses on understanding machine operation/states belonging to different platforms in Google data centers through workload traces. In particular, we are interested in understanding how machine availability is impacted by machine failure and maintenance operations. This can be used to predict machine state at any given time. Knowledge of machine state can provide better insight on scheduling applications on these resources and offer potential mechanisms for managing user workloads (and potential impact these are likely to have on user-based Service Level Agreements). Using this data set, we extended the Markov chain-based model from [1] to predict machine state. We reconstructed the timeline from the events to address common queries such as: (a) if a machine is removed from the cluster, what is the likelihood of the machine rejoining the cluster and after what duration? (b) how often do machine failures happen, i.e. machines that leave the cluster and do not re-join; (c) how does a machine failure impact other machines under the same network switch? (d) do machine failures happen randomly, or are these more likely to happen at particular times of the day?

The rest of the paper is organized as follows. After presenting related work in section 2, we briefly describe the data set used and its properties in section 3. We detail our experimental setup and results under section 4. A Markov model to predict machine state is described in section 5, along with model evaluation and results. In the end section 6 discusses concluding remarks and future work.

2 | RELATED WORK

Scheduling and reliability remain important considerations for user applications deployed over cloud infrastructure. A number of publications addressing these concerns exist at different levels of granularity. Availability of real-world traces to better understand and characterize these problems remains limited however. To understand the dynamics of machine operation and availability, a large-scale deployment of machines is necessary to fully study their usage. Only a few data sets collected from large-scale deployments are available for public research. Google has such a large-scale deployment of machines and in 2010 for the first time a 7-hour sample of resource-usage information from a Google production cluster [3]. Google has subsequently been releasing anonymized Google Cluster Workload Traces data sets

periodically. In 2011 a second version of Google Cluster Workload Traces data set was released [4,5] covering 29 days, and including activities of jobs, tasks and machines in a data center with a total of 12583 single-cell machines. In 2020 Google made available another cluster data set (v3) [6], which includes machines, collections, comprising instances of 8 cells, with approximately 96.4k machines and 12.6k machines per cell. Google Cluster Workload Traces data set v3 has seen limited analysis in the research community compared to v2. Data center cluster can be divided into two categories:

1. analysis that relates to jobs and/or tasks – with some of the primary questions in this context being: how jobs are scheduled? what is their turnaround time? and how job priority is maintained?
2. analysis of machine behavior in a data center. The primary aspect is to analyze the state transition as machines move between different states (e.g., NEW, REMOVE, REJOIN, UPDATE, etc.), and potential events that lead to these transitions. This aspect is also useful to understand how the capacity of a data centre changes over time and the likely resource pool a user application can access and maintain access to over time.

Although significant work already exists focusing on job scheduling, e.g. analyzing job performance metrics in clusters and data centers [7,8,9,10,11,12], limited attention has been given to understand machine dynamics. In particular, [11,7] studied task events, and using a clustering algorithm classified tasks based on their characteristics. Mishra et al. [7] deem task resource consumption significant for both task scheduling and capacity planning. Their workload classification entails: (1) identifying workload dimensions; (2) constructing task classes using an off-the-shelf clustering algorithm, such as k-means; (3) determining the breakpoints for qualitative parameters (i.e. duration, core, and memory, etc.) for the workload; and (4) merging adjacent task classes to reduce the number of tasks to be scheduled. In [11], the authors identify a Pareto-similar distribution to approximate the number of applications and resource attributes. [9] took into account scheduling limitations imposed by tasks associated with a job, as well as how frequently jobs are killed/evicted and then rescheduled. [13] analyzed traces and present a comparing of v3 with v2 of the Google Cluster Workload Traces data set. [14] and [15] show that due to significant heterogeneity in cloud resources and task execution, popular simplifications techniques are unsuitable. Their analysis confirmed a strong need to build new cloud resource schedulers that are able to respond to the highly dynamic nature of cloud workloads.

As jobs generated from user applications are executed on physical machines, understanding machine dynamics is vital, as this directly impacts the execution of these jobs. Several studies focused on software and hardware failures [16 17 18 19] and general availability of machines [18 20 21 22 23 24 25 26 27 28]. All of these generally make use of self-collected data sets for analysis. In particular, [16] described a model for data centers to highlight dependencies among machine availability. Similarly, [17] built a Naive Bayes classifier that predicts job failure probability using a self-collected data set of failures from scientific workload running on Amazon Web Services (AWS). In [19], the authors proposed replacing the commonly used exponential and Pareto distributions with either Weibull or hyper exponential distributions to estimate the reliability of 105 grid computing systems. The authors in [18] looked at the the empirical and statistical properties of system errors and failures over the course of a year on a network of nearly 400 heterogeneous servers executing a variety of workloads. Their research demonstrated that the system error and failure patterns are made up of time-varying behavior with long periods of stagnation. These stagnant intervals reveal a variety of strong correlation structures and periodic patterns, which have an impact on performance but can also be used to solve problems. In [20], the authors used two DEC VAX clusters to identified correlated failures.

In [21], the authors suggest a heuristic strategy for extracting intermittent faults from error logs, which they call Dispersion Frame Technique(DFT). The DFT was tested using data gathered over 22 months from 13 file servers, proving its ability to remove intermittent mistakes and verify hypotheses with just a fourth of the number of entry points

required by traditional statistical analysis. Failure times are fitted using a Weibull distribution with shape parameters of 0.92 for permanent faults (those requiring the restoration of an operator) and 0.5-0.8 for intermittent failures. In[22], the authors investigated the dependability of a diverse group of approximately 1200 Internet hosts. The average time to failure and repair is 17-20 days and 2-4 days, respectively – with the authors suggesting that both failure and repair times are not exponentially distributed. Over 40 days, Krishnan et al.[23] collected data on failures observed by websites, followed by a statistical analysis on this data. The authors discovered that host failures take less time to repair than network failures, with a mean time to repair of less than 15 minutes in 70% of cases and a mean interval between failures of 4 days.

In[24], the authors examined the dependability of roughly five hundred Windows NT servers over four months. The authors discovered that software and hardware problems are the two most common causes of system downtime. Although scheduled maintenance and changes in hardware and software configurations take less time, they account for 31% of all reboots. Reboots of one or more machines in the same domain happen in bursts, implying that an issue frequently necessitates multiple reboots to resolve. The Weibull distribution is also a suitable fit for the mean time between failures in this scenario, with a shape parameter of 0.5 to 0.95 when solely considering software failures. In[25], the authors present a framework for developing highly accessible Internet services. The time-to-boot is best described as a Weibull distribution (rather than exponential or Pareto) with a shape parameter between 0.33 and 0.49 for all three clusters investigated.

In[26], the authors looked at data center network reliability to see which components were the least unreliable and how failures affected overall system performance. Device failures are less common than link failures, according to the authors of[26]. Device failures occur in bursts daily, and large-scale maintenance causes times of high frequency (authors include in the failure definition events where devices are power cycled during planned maintenance). Although software problems outnumber hardware breakdowns, the latter causes the greatest downtime. In[28] failure statistics are used from the high-performance computing facility at Los Alamos National Laboratory over 9 years, containing 23K failures across 20+ distinct systems. The authors used this data set to identify root cause of failure, the average time between failures, and the average time to repair. They discovered that typical failure rates vary greatly between systems, ranging from 20 to 1000 per year, and that time between failures is effectively described by a Weibull distribution with decreasing hazard rate. The average repair time varies from less than an hour to more than a day depending on the system, and repair times are effectively represented by a Lognormal distribution.

Although the studies listed above carry out job analysis across cloud and high performance computing resources, the general limitation has been the lack of analysis of other related aspects – such as machine behavior. We posit that machine behaviour should be an important consideration in understand resource failure and subsequent re-integration of the machine into the resource pool. We therefore analyse Google traces to investigate the role that machine states have on the behavior of jobs executing on these machines.

3 | GOOGLE CLUSTER WORKLOAD TRACES DATA SET

Google Cluster Workload Traces v3[4] holds events of machines collected from 8 cells, with approximately 96.4k machines and 12.6k machines per cell. Machine events consist of three types of events – ADD, REMOVE and UPDATE. In a cluster, the machines can either be active or inactive. When a machine is in the active state, it is available to the cluster. Machines go to the inactive state for software/ hardware maintenance. Some of the machines never rejoin the cluster. ADD event lead to machines moving from inactive to the active state, REMOVE event does the opposite, and UPDATE event keeps the machine in available state. Machine state transitions are shown in Figure 1.

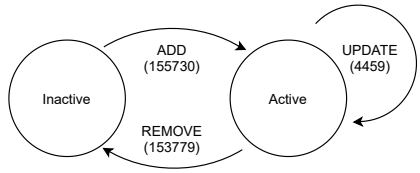

FIGURE 1 Machine events (as defined in data set) along its frequency and state transition

The Google data set only represents ADD, REMOVE and UPDATE events, providing a high-level overview of machine state transition. To get an in-depth overview of machine behavior and its state transition, we derived NEW ADD and REJOIN events from existing ADD events, discarding existing ADD events – as ADD events are the union of NEW ADD and REJOIN events. We know that Machine IDs usually remain the same across the whole machine life cycle[13], based on this we can split ADD events into NEW ADD and REJOIN as described in Algorithm 1.

Algorithm 1 Split ADD events into NEW ADD and REJOIN events

Require: A: set of all ordered ADD events
Ensure: $\widetilde{N}$: set of all NEW ADD events
Ensure: $\widetilde{R}$: set of all REJOIN events
  $\widetilde{N} \leftarrow \emptyset$; $\widetilde{R} \leftarrow \emptyset$; $M \leftarrow \emptyset$;
  for all $a \in A$ **do**
    $m \leftarrow$ Get-Machine-ID(a)
    if m in M **then**
      $\widetilde{R}.insert(a)$
    else
      $\widetilde{N}.insert(a)$
      $\widetilde{M}.insert(m)$
    end if
  end for

Similarly, REMOVE events represent both temporary removals for software/ hardware maintenance as well as permanent removal of machines. To provide a better distinction between these categories, we derived permanently REMOVE events from the more general REMOVE events as defined in Algorithm 2. Our analysis therefore assumes NEW ADD and REJOIN as two separate events and mention REMOVE (permanent) separately. Figure 2 describes state transition as a result of an event and frequency of each event across the entire data set.

Similar to v2[4] of this data set, v3 holds events of machines, collections and instances. Google Cluster Workload Traces v2 has one cell and 12.6k machines, whereas v3 consists of 8 cells, with approximately 96.4k machines and 12.6k machines per cell. A brief comparison of both data sets is given in Table 1. We see that on average, v3 has a greater number of events compared to v2, except for the UPDATE event. This may be either due to fewer changes to machine attributes or a change in the way attributes are assigned to machines. Repair probability is the probability that the machine will rejoin the cluster if a machine is REMOVEd. According to Table 1, repair probability in v3 is 98.24% and is approximately 0.677% lower than v2.

Algorithm 2 Extract REMOVE (permanent) Events

Require: E: set of all ordered events
Ensure: R: set of all REMOVE (permanent) events
  $R \leftarrow \emptyset$; $\widetilde{R} \leftarrow \emptyset$;
  for all $e \in E$ **do**
    $t \leftarrow$ Get-Event-Type(e); $m \leftarrow$ Get-Machine-ID(a)
    if t is REMOVE **then**
      $R.insert(e)$
      $\widetilde{R}.insert(m)$
    end if
    if t is ADD and m in $\widetilde{R}$ **then**
      $i \leftarrow$ Get-Index-Of$(m, \widetilde{R})$
      $R.removeAt(i)$
      $\widetilde{R}.removeAt(i)$
    end if
  end for

TABLE 1 Comparison between 2011 (v2) and 2019 (v3) traces data set

	Version 2 (cell)	**Version 3** (total)	**Version 3** (cell avg)
Total Events	37780	309833	38729
Initial Machines	12477	91902	11487
NEW ADD	106	4653	582
REMOVE	8957	153779	19222
UPDATE	7380	324	40
REJOIN	8860	151077	18884
REMOVE (Permanently)	97	2702	338
Repair Probability	0.98917	0.98243	-

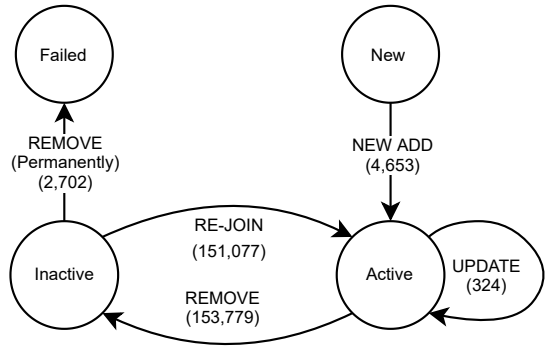

FIGURE 2 Events, state transitions and event frequency for machines

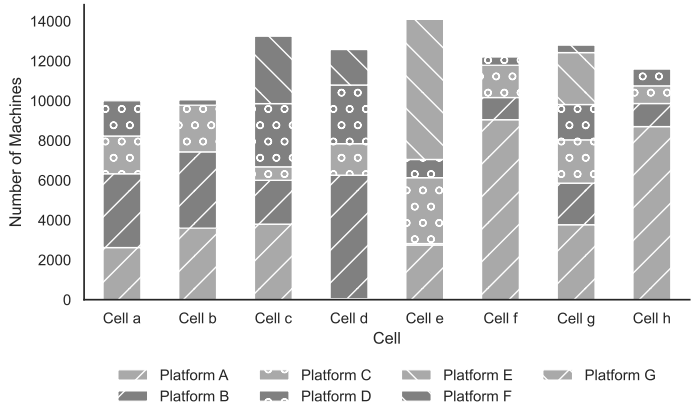

FIGURE 3 Distribution of Machines by Platform across cells

Based upon CPU vendor/architecture and memory module types, machines of similar configurations are grouped. One such configuration is known as a hardware platform. In the data set, each Platform is represented by its unique hash. For better readability, those hashes are replaced with letters, and letters are used throughout the paper. Initial analysis of the data set shows eight hardware platforms that mismatched the description provided by Google. After a thorough examination of machine events, we found that one hardware platform has only 4135 unique ADD events. Considering machine events from these 4135 machines, we observed that the CPU and memory capacity of these machines are missing. After ADD event, there is an UPDATE event lagging by a few seconds, and this event holds both CPU and memory capacity information, and the platform hash is different. We speculated that a specific platform hash is given to the machine during initialization. We removed all the 4135 unique ADD events and changed the incoming UPDATE event to be ADD events. Figure 3 shows the distribution of machines by platform across different cells. We see that across the whole cluster, some cells have some Platforms in abundance as compared to other Platforms.

Considering Figure 3 and Table 2, Platform E and F have lesser number of initial machines compared to other Platforms. We also see that Platform G initially has 27 machines and later all of them are removed, based upon that we can speculate that Platform G was retired.

TABLE 2 Machine Events and Repair Probability by Platform

	A	B	C	D	E	F	G	Overall
Initial Machines	32194	18716	14063	11958	9449	5495	27	91902
NEW ADD	2127	1686	437	141	208	54	0	4653
REMOVE	52447	31610	25610	17228	19314	7543	27	153779
UPDATE	133	21	74	47	40	9	0	324
REJOIN	51914	31208	25276	17066	18114	7499	0	151077
REMOVE (Permanently)	533	402	334	162	1200	44	27	2702
Total Events	106621	64525	51397	34482	37676	15105	54	309833
Repair Probability	0.989837	0.987283	0.986958	0.990597	0.937869	0.994167	0	0.982429

We observed that only 2.9872% of all ADD events represent the addition of a new machine. In contrast, all other events represent rejoining an existing machine removed, most likely for maintenance. According to Table 2 there exist 7 different types of platforms. Each Platform had some machines at the beginning of trace collection, and more machines are added afterward. A lot of machines rejoins after being removed, and a few machines failed to join back. Any machine that rejoins is mostly removed for scheduled maintenance.

Google Cluster Workload Traces Data set version 3 has a size of 2.7TB in compressed form. Google provided the data set in JSON format and as BigQuery tables. In the case of JSON, the data set size exceeds 7TB. That size of data set comes will its unique challenges. Using the Google BigQuery version of the data set resolves the majority of challenges associated with JSON format. However BigQuery is expensive, on the other hand using JSON format, we can use commodity hardware. For this study, the data set is downloaded in compressed JSON format from the Google Storage bucket. Uncompressed JSON requires a lot of memory and as an optimization step, compressed JSON is loaded in Spark Cluster and saved as Apache Parquet format. Only the required data from Apache Spark Parquet is read, transformed, and saved to a CSV file for further processing using Python.

4 | EXPERIMENTS AND RESULTS

We grouped machine events by Platform and plotted histograms for each event type, where time is expressed in hours and bin size is set to 24 hours. Figure 5 contains a histogram for different types of events across all platforms and also for each individual platform where y-axis on the left represents the actual frequency of events. It is worth noting that the frequency of events is well spread over the whole duration across all platforms. To compare event volume across all platforms, we normalized events by initial machines in respective platform (plotted on right y-axis) and observed no resembles across platforms. Each Platform is behaving differently from the other. For some platforms, new machines are introduced at a higher frequency than others. Similarly event volume differs for each platform.

We observed in Figure 5 that the REMOVE and REJOIN events are quite similar. This is because 98.24% of removed machines rejoined. We also observed several spikes in REMOVE/REJOIN events for all platforms. This might be due to critical software updates being rolled out. The machines' maintenance and software updates are frequent, and overall a maximum of 10% of the initial machine count is observed. It varies across platforms and Platform B has the highest observed REMOVE and REJOIN frequency within 24 hrs at 16%. We also observed across all platforms UPDATE event frequency is very low. This might be due to machine attributes not being changed as often as machine

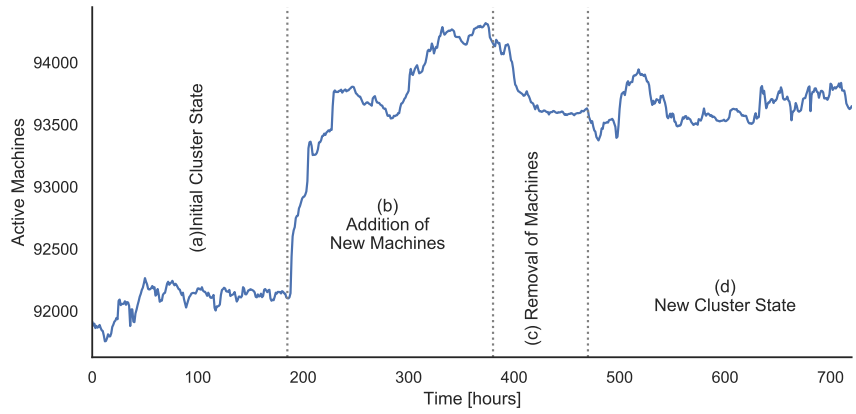

FIGURE 4 Active machines Timeline

maintenance is performed.

Furthermore, NEW ADD events are low in numbers and usually have spikes. For example, for Platform B, more than 300 machines are added within 24 hours. Similarly, for Platform A, up to 400 machines are added within 24 hours. Overall that counts to 1.6% and 1.2% of the number of initial machines in respective Platforms. Looking at REMOVE (Permanently) histograms of Figure 5 we can observe machine failures are spread out over the whole period, but there are spikes in failures, for instance, we see 400+ machines belonging to Platform E failed within a day which might be some scheduled degradation of hardware rather than actual machine failures. Moreover, across all platforms, new machine additions are higher than machine failures, which means that the number of the machines inside the cluster/cell increase with the time.

Further, we reconstructed the timeline of active machines and plotted active machines over time, as shown in Figure 4. On the surface, it looks like the number of machines keeps on fluctuating. Over 1 month period, an average of 93,040 machines with a standard deviation of 820 were active all the time while a minimum of 91,649 and a maximum of 94,334 active machines are observed. Overall we observed an increasing trend in number of active machines. For better understanding and explanation we split timeline into four sections. In the beginning, as shown in Figure 4 section a, the number of active machines are fluctuating, but the trend is steady. However, on the 8th day of trace collection, as shown in Figure 4 section b, the number of active machines raised within 48hrs by about 1,500. Looking at Figure 5 we can see new machine additions to Platform A and Platform B caused the spike. New machine additions to Platform C and D keep on fluctuating for the rest of the period. As more and more machines were added, the number of active machines rose to a maximum of 94.3k. Soon after that lot of machines were removed from Platform E, but very few rejoined immediately, causing the number of active machines to fall to about 93.5k as shown in Figure 4 section c. Afterward, there exist few fluctuations, but the active number of machines trend remained steady as shown in Figure 4 section d.

We investigate the presence of temporal correlation between the failure of one machine leading to the failure of more machines under the same network switch using scatter plots. Figure 6 plots machine failures by network switch over time (only network switches having more than 20 machine failures are plotted to improve visuals). From Figure 6, we observe a relationship between the failure of one machine leading to the failure of more machines under the same network switch. Out of 2,702 total machine failures, 1,110 machine failures are plotted in Figure 6. However, it seems fewer machine failures happening at once or with very short delay under the same network switch. We first

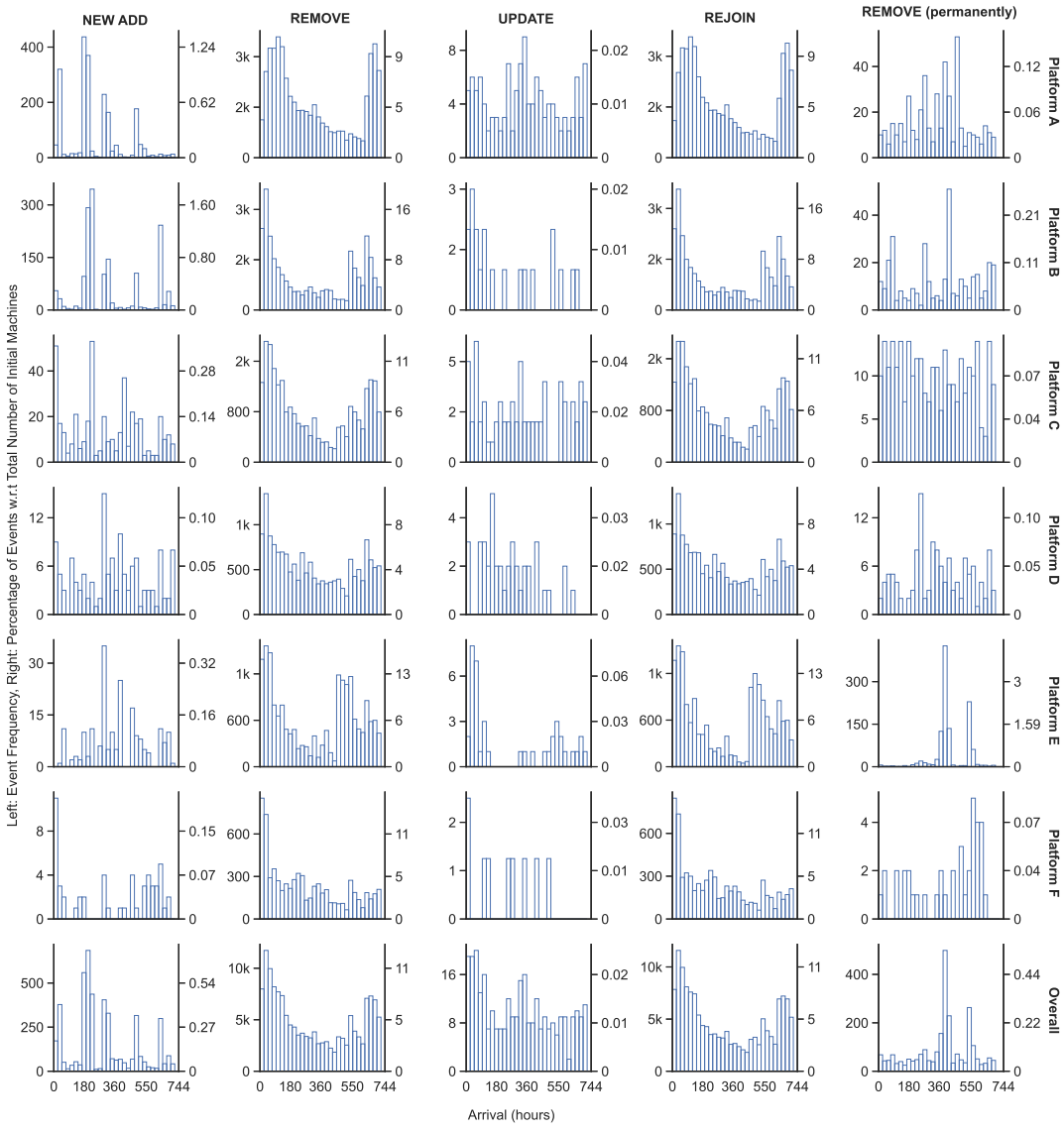

FIGURE 5 Histograms showing frequency of events (left y-axis) and their percentages w.r.t. the number of initial machines (right y-axis) on different hardware platforms

calculated the delay between the two consecutive machine failures under the same network switch to understand this better. We found that around 13% of machines failed within 1 minute of another machine failure under the same network switch. Figure 7 plots the distribution of delay between two machine failures under the same network switch (bin size is 1 minute). We see that most of the machines fail within 1 minute of the previous machine failure; afterward, subsequent machine failures within the network switch drops exponentially.

All the timestamps in the data set are relative to the trace collection start date. Assuming trace starts from Monday

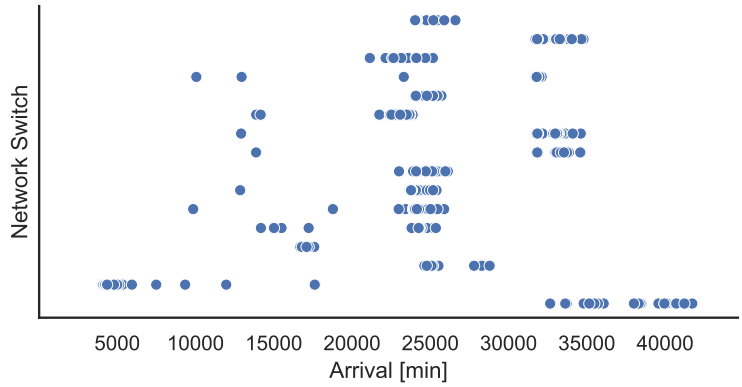

FIGURE 6 Machine failures by network switch

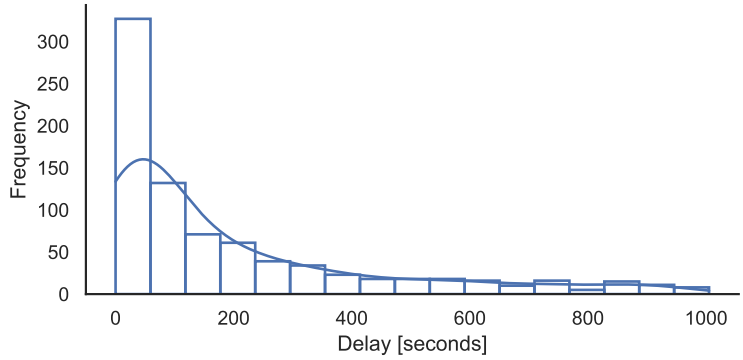

FIGURE 7 Delay between two machine failures using the same switch

at midnight, we formed a scatter plot with number of machine failure as bubble size, hour of the day on the x-axis and the day of the week on y-axis, as shown in Figure 8. We observed failures are spread out fairly over the whole duration. Increasing failures on Tuesday at midnight and Wednesday at noon are due to a spike in Platform F machine failures as can be see in the fifth column of Figure 5.

We calculated the time to recover from the timeline we constructed, i.e., the period between REMOVE and ADD events for a given machine that rejoined recently. We observed that the distribution of time to recover is heavily skewed. On average, it took 47minutes for a machine to rejoin, while the standard deviation is 4.5hours. It took a minimum of 1.2seconds and a maximum of 26.5days to recover. After two days, there are quite a few recoveries, and removing those gives us a normal distribution. Figure 9 plots distribution of time to recover where x-axis is limited to 2days. Most of the machines recover within 24hrs, as seen in Figure 9. Machines recovering within an hour are most likely to be those that have been taken offline for software upgrades.

Considering Figure 9, we can answer why REMOVE and REJOIN histograms look similar in Figure 5. As bin size in Figure 5 is 24hours, and on average machine recovers within 47 minutes, it leads to identical REMOVE and REJOIN histograms. This observation is also consistent with the observation provided by Google. Machines are

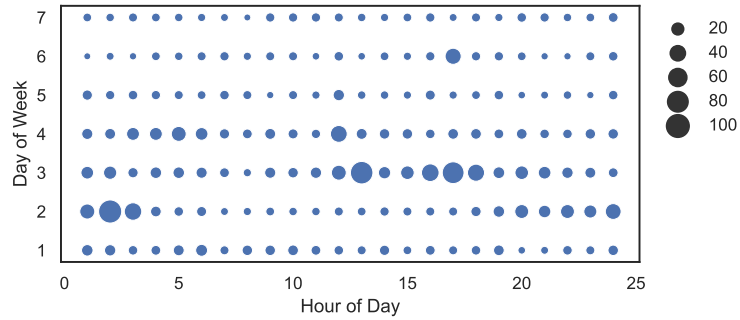

FIGURE 8 Machine failures at different hours

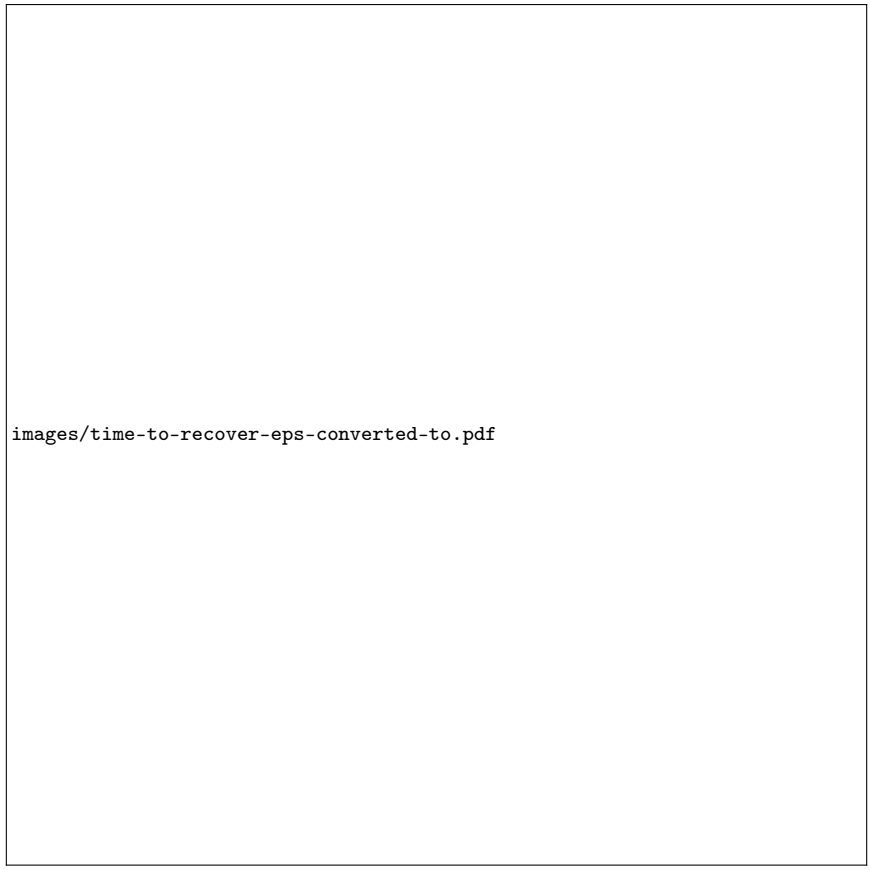

FIGURE 9 Time to recover

removed for software maintenance and other hardware/ software related upgrades. Most of the machines join back after software update or sometimes when the critical issues are resolved. We conclude that machine behaviour

differs across platforms – with no similarity in behaviour across platforms. To consider what makes a platform behave differently, we looked at other available machine information in the data set. Machine within a cluster are identical, although some can differ in hardware and operating – which can also change device drivers hosted on these machines. Hence platform behaviour differs due to different resource life-cycle and patches issued to OS and installed system software.

5 | MODEL

To predict machine state, we build a simple Markovian chain model. Markovian chain is a sequence of random variables $X_0, X_1, X_2, \ldots$ that satisfied the rule of conditional independence. For any positive integer n and possible states $i_0, i_1, \ldots, i_n$ of the random variables, Markovian property is defined as:

$$P(X_n = i_n \mid X_{n-1} = i_{n-1}) = P(X_n = i_n \mid X_0 = i_0, X_1 = i_1, \ldots, X_{n-1} = i_{n-1})$$

Thus, only previous state knowledge is necessary to determine the probability of the current state. For Markovian chain X at time t, probability of machine state transition is governed by transition matrix P_t. Given an ordering of a matrix's rows and columns by the state space S, the $(i,j)^{th}$ element of the matrix P_t is given by:

$$(P_t)_{i,j} = \mathbb{P}(X_{t+1} = j \mid X_t = i)$$

Transition matrices have the property that the product of subsequent ones describes a transition along the time interval spanned by the transition matrices. That is to say, $P_0 \cdot P_1$ has in its $(i,j)^{th}$ position the probability that $X_2 = j$ given that $X_0 = i$. And, in general, the $(i,j)^{th}$ position of $P_t \cdot P_{t+1} \cdot \cdots \cdot P_{t+k}$ is the probability of $\mathbb{P}(X_{t+k+1} = j \mid X_t = i)$.

The k-step transition matrix is $P_t^{(k)} = P_t \cdot P_{t+1} \cdots P_{t+k-1}$ and satisfies:

$$P_t^{(k)} = \begin{pmatrix} \mathbb{P}(X_{t+k} = 1 \mid X_t = 1) & \mathbb{P}(X_{t+k} = 2 \mid X_t = 1) & \cdots & \mathbb{P}(X_{t+k} = n \mid X_t = 1) \\ \mathbb{P}(X_{t+k} = 1 \mid X_t = 2) & \mathbb{P}(X_{t+k} = 2 \mid X_t = 2) & \cdots & \mathbb{P}(X_{t+k} = n \mid X_t = 2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbb{P}(X_{t+k} = 1 \mid X_t = n) & \mathbb{P}(X_{t+k} = 2 \mid X_t = n) & \cdots & \mathbb{P}(X_{t+k} = n \mid X_t = n) \end{pmatrix}$$

In our model as shown in Figure 10, machine can be in either of four pools, P_{new}, P_{active}, $P_{inactive}$ and P_{failed}. P_{new} holds machines which can be added to system. From P_{new} machine can jump into P_{active} which represents active machines pool represented by timed transition T_{add}. From P_{new} machine is either updated in-place represented by timed transition T_{update}, or removed from pool represented by time transition T_{remove} event and moved to $P_{inactive}$ pool. From $P_{inactive}$ pool, machine is either discarded represented by timed transition $T_{discard}$ and joined P_{failed} pool which holds discarded machines, or rejoined the P_{active} pool represented by time transition T_{repair}. State transition probabilities (T_{update}, T_{remove}, T_{repair} and $T_{discard}$) are calculated from data set, however T_{add} cannot be calculated as that depends on data center operators.

To evaluate the model, we first discarded NEW ADD machine events and reconstructed the timeline. To predict

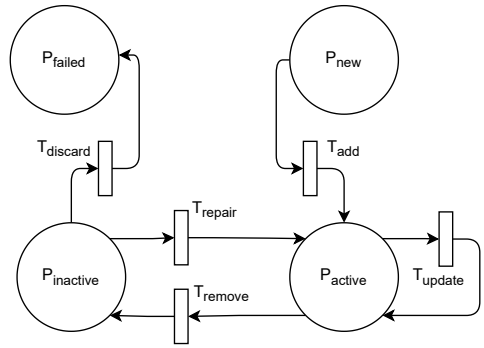

FIGURE 10 Machine State Transition: Markov Model

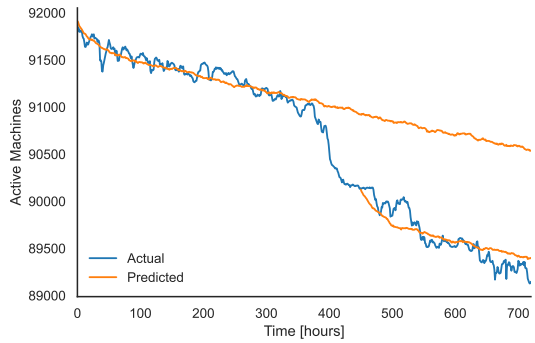

FIGURE 11 Timeline reconstruction using proposed Markovian Model excluding newly added machines

active machines timeline, we first predicted individual machine state for every hour in the trace collection period, using the initial number of machines as input. Using the estimated state transition probabilities, the model is used to predict the next state. Given a per hour machine state (from the data), we constructed a timeline of the active number of machines. Comparison of predicted and actual active number of machines timeline is shown in Figure 11. Initially both timelines run in parallel, but on the 18th day of trace collection, there is a significant drop in active machines and the model is not able to follow this significant drop. Restarting the model from the 19th day of trace collection, providing number of active machines at 19th day of trace collection as input, the model was able to follow the remaining trend. One significant deviation of predicted active machines from actual active machines is that predicted active machines doesn't have momentary spikes as we have observed in actual active machines trend.

As NEW ADD machine events depends on data centre operators, we picked NEW ADD events from actual data set and added NEW ADD events to predicted active machines timeline as shown in Figure 12. Trend is looking similar other than the the significant drop in active machine on 18th day of trace collection. We measured model performance by first reconstructing the active machine timeline from predicted machine state transitions and comparing that with the actual active machine timeline from the data set. We calculated the mean absolute percentage error of 1.76%. Using our model, we predicted the trend for active machines over the the next 15 days as shown in Figure 13. We observe a down ward trend which aligns with the actual data set. The active number of machines drops and data centre operators keep adding new machines to compensate and extend the resource pool.

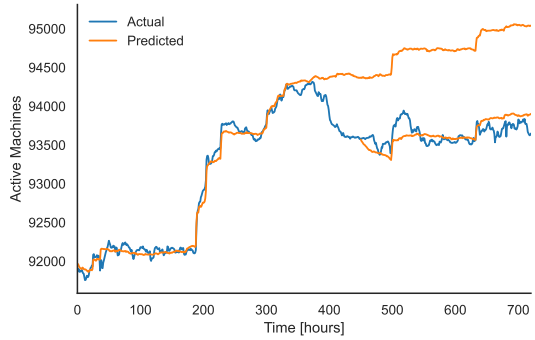

FIGURE 12 Timeline reconstruction using Markov model combined with NEW ADD events

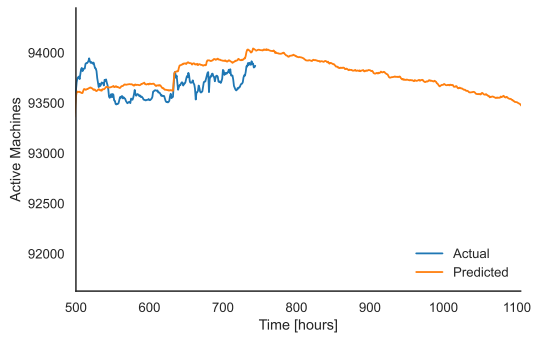

FIGURE 13 Predicted Active Machines trend for next 15 days using a Markov Model

6 | CONCLUSION

In this paper we analysed machine failures from the recently published Google data center data set. We observed that 98.24% of machines recover/rejoins on average in 47 minutes with a standard deviation of 4.5 hours. The active number of machines keeps on fluctuating during the collection period, but the overall trend remains steady if we ignore the new machine additions. Less that 2% of machines fail to join the cluster, and failures happen regularly. Machine maintenance/software updates are frequent, and we have observed REMOVE/REJOIN events up to 10% of the initial number of machines inside the cluster within a day. Furthermore, we find that the cluster/cell grows as more new machine added as compared with failed machines. We also found that failure in a machine sometimes leads to more machine failures. Specifically, if a machine fails, there are 13% chances of another machine failure under the same network switch within 1 minute. We extended a Markov chain based model that takes state transition probabilities and predict next state of machine. By predicting individual machine state, we can construct an active machine timeline. Comparing predicted with actual machine timeline we observed MAPE (Mean Absolute Percentage Error) of 1.76%, demonstrating that our model is able to predict machine state with high accuracy.

references

[1] Sebastio S, Trivedi KS, Alonso J. Characterizing machines lifecycle in Google data centers. *Performance Evaluation* 2018; 126: 39 – 63. http://dx.doi.org/https://doi.org/10.1016/j.peva.2018.08.001 doi: https://doi.org/10.1016/j.peva.2018.08.001

[2] Yamada D, Kuwata Y, Kasai R, Nakamura T. Automation of Message Handling in Cloud-based Managed Service. *Procedia Computer Science* 2013; 22: 643-652. 17th International Conference in Knowledge Based and Intelligent Information and Engineering Systems - KES2013http://dx.doi.org/https://doi.org/10.1016/j.procs.2013.09.145 doi: https://doi.org/10.1016/j.procs.2013.09.145

[3] Hellerstein JL. Google cluster data. Google research blog; 2010. Posted at `http://googleresearch.blogspot.com/2010/01/google-cluster-data.html`.

[4] Wilkes J. More Google cluster data. Google research blog; 2011. Posted at `http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html`.

[5] Reiss C, Wilkes J, Hellerstein JL. Google cluster-usage traces: format + schema. technical report, Google Inc.; Mountain View, CA, USA: 2011. Revised 2014-11-17 for version 2.1. Posted at `https://github.com/google/cluster-data`.

[6] Wilkes J. Google cluster-usage traces v3. technical report, Google Inc.; Mountain View, CA, USA: 2020. Posted at `https://github.com/google/cluster-data/blob/master/ClusterData2019.md`.

[7] Mishra AK, Hellerstein JL, Cirne W, Das CR. Towards characterizing cloud backend workloads: insights from Google compute clusters. *SIGMETRICS Perform. Eval. Rev.* 2010; 37(4): 34–41. http://dx.doi.org/10.1145/1773394.1773400 doi: 10.1145/1773394.1773400

[8] Di S, Kondo D, Cirne W. Characterization and comparison of cloud versus Grid workloads. In: ; 2012; Beijing, China: 230–238

[9] Liu Z, Cho S. Characterizing machines and workloads on a Google cluster. In: ; 2012; Pittsburgh, PA, USA.

[10] El-Sayed N, Zhu H, Schroeder B. Learning from Failure Across Multiple Clusters: A Trace-Driven Approach to Understanding, Predicting, and Mitigating Job Terminations. In: ; 2017: 1333–1344

[11] Di S, Kondo D, Franck C. Characterizing cloud applications on a Google data center. In: ; 2013; Lyon, France.

[12] Abdul-Rahman OA, Aida K. Towards understanding the usage behavior of Google cloud users: the mice and elephants phenomenon. In: ; 2014; Singapore: 272–277

[13] Tirmazi M, Barker A, Deng N, et al. Borg: the Next Generation. In: ACM; 2020; Heraklion, Greece

[14] Reiss C, Tumanov A, Ganger GR, Katz RH, Kozuch MA. Heterogeneity and dynamicity of clouds at scale: Google trace analysis. In: ; 2012; San Jose, CA, USA.

[15] Reiss C, Tumanov A, Ganger GR, Katz RH, Kozuch MA. Towards understanding heterogeneous clouds at scale: Google trace analysis. *Intel Science and Technology Center for Cloud Computing, Tech. Rep* 2012; 84.

[16] Menasce D. Performance and availability of Internet data centers. *IEEE Internet Computing* 2004; 8(3): 94-96. http://dx.doi.org/10.1109/MIC.2004.1297280 doi: 10.1109/MIC.2004.1297280

[17] Samak T, Gunter D, Goode M, et al. Failure analysis of distributed scientific workflows executing in the cloud. In: ; 2012: 46-54.

[18] Sahoo R, Squillante M, Sivasubramaniam A, Zhang Y. Failure data analysis of a large-scale heterogeneous server environment. In: ; 2004: 772-781

[19] Nurmi D, Brevik J, Wolski R. Modeling Machine Availability in Enterprise and Wide-Area Distributed Computing Environments. In: Cunha JC, Medeiros PD., eds. *Euro-Par 2005 Parallel Processing*Springer Berlin Heidelberg; 2005; Berlin, Heidelberg: 432–441.

[20] Tang D, Iyer R, Subramani S. Failure analysis and modeling of a VAXcluster system. In: ; 1990: 244-251

[21] Lin TT, Siewiorek D. Error log analysis: statistical modeling and heuristic trend analysis. *IEEE Transactions on Reliability* 1990; 39(4): 419-432. http://dx.doi.org/10.1109/24.58720 doi: 10.1109/24.58720

[22] Long D, Muir A, Golding R. A longitudinal survey of Internet host reliability. In: ; 1995: 2-9

[23] Kalyanakrishnan M, Iyer R, Patel J. Reliability of Internet hosts: a case study from the end user's perspective. *Computer Networks* 1999; 31(1): 47-57. http://dx.doi.org/https://doi.org/10.1016/S0169-7552(98)00229-3 doi: https://doi.org/10.1016/S0169-7552(98)00229-3

[24] Xu J, Kalbarczyk Z, Iyer R. Networked Windows NT system field failure data analysis. In: ; 1999: 178-185

[25] Heath T, Martin RP, Nguyen TD. Improving Cluster Availability Using Workstation Validation. *SIGMETRICS Perform. Eval. Rev.* 2002; 30(1): 217–227. http://dx.doi.org/10.1145/511399.511362 doi: 10.1145/511399.511362

[26] Gill P, Jain N, Nagappan N. Understanding Network Failures in Data Centers: Measurement, Analysis, and Implications. In: SIGCOMM '11. Association for Computing Machinery; 2011; New York, NY, USA: 350–361

[27] Schroeder B, Gibson G. A Large-scale Study of Failures in High-performance-computing Systems (CMU-PDL-05-112). 2018

[28] Schroeder B, Gibson G. A large-scale study of failures in high-performance computing systems. In: ; 2006: 249-258