A Trace-Driven Evaluation of Cloud Computing Schedulers for IaaS

Zhihao Yao
Computer and Information Technology
Purdue University
West Lafayette, IN, USA
yao86@purdue.edu

Ioannis Papapanagiotou
Platform Engineering
Netflix, Inc.
Los Gatos, CA, USA
ipapapa@ncsu.edu

***Abstract*—Prior work in the Cloud domain has mainly focused on evaluating systems using synthetic workloads. Nonetheless, synthetic workloads are not always an accurate depiction of the real-world as they have been generated based on assumptions and statistical distributions that are not representative of the actual traffic that flows from the applications and microservices to the lower layers of the Cloud infrastructure. Moreover, the selection of diverse assumptions and distributions create additional complications on a fair comparison between platforms and reduces the ability to reproduce the experiment. In this paper, we compare synthetic workloads used in prior research with the Eucalyptus cloud traces. We find that the distributions employed in the synthetic workloads are not always in-line with the real usage pattern. Therefore, to deliver realistic and reproducible research outcomes, we propose a trace-driven research methodology and showcase an experimental design which employs a workload trace and simulation method. The experiment provides practical suggestions from a realistic input and environmental setting. In addition, it is straightforward to replicate the experiment so that the accepting process of research outcome is shorted by a practical implementation.**

I. Introduction

Cloud computing is becoming the standard of IT infrastructure and replacing the traditional on-promise deployment paradigm. Three levels of service in cloud computing are abstracted as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (SaaS) [1]. At the IaaS layer, the infrastructure resources such as computing, network, storage, etc. are provided in a flexible and on-demand manner to the cloud tenant. Thus the end user can provision virtual machines (VMs), Random Access Memory (RAM), virtual switches, or storage volumes on an as-needed and a when-needed basis. Similar operations are also performed in the container domain, where a microservice requests a Linux container (LXC) with specific resources. This can remove the hurdles of maintaining on-premise hardware infrastructure, or the VMs themselves in the container world.

The IaaS cloud providers usually maintain several data centers for resource isolation. Hence, many challenges can potentially emerge, such as how to improve the performance and provide competitive Service Level Agreements (SLAs) in a multi-tenant environment [2], or how to maximize resource utilization on limited hardware resources [3], [4]. Solving these challenges can help the cloud provider to increase their market share and business profit. Several techniques have been proposed to address these issues [5]. Among them, a number of studies have focused on the infrastructure resource management problem [2], [3], [6]–[8]. To showcase the effectiveness of each proposed technique, researchers developed an experimental or simulation platform and conducted a quantitative comparison.

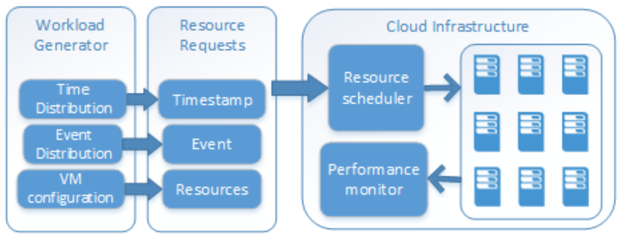

Fig. 1. Common Workflow of Cloud Resource Management Experiment

In many cases, we had to rely on synthetic workloads generators [9] and simulation frameworks to perform these experiments. It is not always feasible to have an ideal experimental environment where a large number of cloud production-like microservices operate on the testbed. The common workflow of IaaS resource management experiment is shown in Fig. 1. The synthetic workload generator produces every field in a resource request based on assumptions and statistical distributions. Then the synthetic requests are submitted to the resource scheduler and make placement decision in a simulated or real infrastructure cluster.

However, several of the empirical assumptions for statistical distributions are made due to the lack of published characterization work of real IaaS cloud workload. A similar problem was identified in the Ethernet domain. Prior to the seminar paper [10], most prior studies used Markovian models which are amenable to accurate analysis and efficient control. Hence, this may result in an enormous diversity of results. Different decisions on workload assumptions and distributions also bring extra variables, making it hard for other researchers to replicate the experiments. Some work has been done on examining the reproducibility of simulation methodology [6], [11]. However, to our knowledge, the same question has not been raised on the synthetic workload used in IaaS cloud research.

Therefore we decided to look into the synthetic workload and explore the answer to the reality and reproducibility problem. First, we discuss and review the synthetic workloads used in prior IaaS resource management research and introduce a newly published real-world workload trace (Section II). By conducting an in-depth comparison (Section III), we find that there are some gaps between the statistical distributions used in synthetic workloads and real-world workloads. We showcase a trace-driven experiment example (Section IV) to illustrate how we can deliver realistic and reproducible results by taking advantage of real-world traces. We evaluate resource utilization performance of several resource scheduling algorithms that are based on bin packing heuristics and their variants. Our simulation experiments reveal that there is still a room for further improvement in the resource utilization of production cloud system. A realistic reproducible experiment may accelerate the adoption process from research outcome to practical system.

II. IaaS Cloud Workload

Normally a workload represents the input, i.e. a stream of operation events submitted by a client or an application to a system. In the IaaS cloud setting, the operation event is the infrastructure resource request. The cloud user is able to perform a resource operation at anytime given the on-demand nature of cloud computing. An IaaS cloud workload commonly consists of three basic fields:

1) *Start time*: It specifies the time when a request, or a batch of requests are submitted. In synthetic workload, the time interval between two continuous requests, i.e. inter-arrival time, is used to substitute the arrival time so that it can be generated from certain statistical distribution.
2) *Termination time*: It specifies the time point when a resource release request for previously provisioned resource is submitted. The termination time and start time decide the lifespan of an allocated resource. In synthetic workload, this field is normally replaced by lifetime or duration for the convenience of generating from a certain distribution.
3) *Requested resources*: The quantity of each requested IaaS resource, such as CPU core, memory, storage volume. The IaaS resource is generally provided in the form of virtual machine, storage volume, or an isolated resource like a container. In synthetic workload, a couple of VM configurations are defined for selection.

In the following subsections, we introduce two main categories of IaaS cloud workload: synthetic workload and real-world trace based workload.

A. Synthetic Workload

As mentioned before, a synthetic workload is the most popular workload in cloud scheduling research. It has been employed as the input of many experiments. Each field in the synthetic workload request is generated based on a pre-defined statistical distribution and assumptions. In this section, we provide a quick review of the distributions and assumptions used in prior synthetic workloads. We select synthetic workloads from three prior work that are published after 2010 as examples. They are:

- Two workloads used for evaluating a power optimizing resource scheduler [6].
- Two workloads used in a performance evaluation study for VM placement algorithms [12].
- One workload used in a backward speculative placement scheduler [7].

These synthetic workloads are all utilized as input in the evaluation experiments. Their distributions and VM type configurations are listed in Table I.

To generate the inter-arrival time and lifetime, the synthetic workload generator made different choices on selecting statistical distributions for the workloads in Table I. They chose an Exponential, Lognormal and Poisson distribution with different mean parameters for the synthetic inter-arrival time. Furthermore, Pareto distribution employed by the workload generator in [12] is added to the lifetime distribution list, also with different parameters. It's worth noting that all these workloads do not provide any source or reference to explain why a certain statistical distribution is chosen with that particular distribution parameter as the basis for generating a workload parameter. They simply made assumptions on data distribution. On VM resources generation, the requested resource in all synthetic requests in [6] are 1 core VM. The other workloads created more than one VM types for selection. However, two workloads in [12] chosen a VM type uniform randomly from all available types, the remaining workloads made choice based on pre-defined probability with no explanation. Another implicit assumption is all VM types follow the same inter-arrival time and VM lifetime distribution.

The reason behind the current method of selecting a statistical distribution for a synthetic workload is the lack of published observations and characterization of real-world workloads. Therefore, researchers have to make decisions based on empirical observations from related fields [13]. Hence, this makes it hard to validate the synthetic workload and reproduce the experiment [11]. To the best of our knowledge, the assumption that the distributions of other computing workloads can be applied on IaaS cloud workload has not been verified. Moreover, it is difficult to make a fair estimation of a real performance improvement when the research outcome is implemented in a production system with a real-world workload.

B. Real-World Workload

Real-world workload traces are captured from various production systems and are publicly available for further investigation. People are able to take advantage of these traces to provide insights of system performance and usage pattern. For example, Abdul-Rahman et al. [14] analyzed user behavior and created system models based on Google cluster data set [15]. The statistical distributions concluded from Google cluster traces are referenced as the basis of many subsequent cluster management research. However, in the field of IaaS

TABLE I
SYNTHETIC WORKLOADS USED IN SELECTED PREVIOUS RESEARCH

	Workload	Inter-arrival Time	Lifetime	Requested Resource
Pucher et al. [6]	1	Exponential (mean=80sec)	Exponential (mean=500sec)	1 core VM
	2	Lognormal (μ=3.8, σ=1.0)	Lognormal (μ=4.5, σ=1.0)	
Mills et al. [12]	1	Exponential (mean=30min)	Pareto (mean=8 hours)	Random (uniformly) choose from 7 VM configurations
	2		Pareto (mean=4 hours)	
Calcavecchia et al. [7]	1	Poisson (mean=5 time unit)	Poisson (λ=110 time unit)	Random choose from 4 VM configurations based on pre-defined probability of [0.3, 0.25, 0.25, 0.2]

TABLE II
SUMMARY OF EUCALYPTUS WORKLOAD TRACE

Trace	Duration (days)	VM created	Host	Cores/Host
DS1	85	9173	13	24
DS2	283	900	7	12
DS3	279	2600	7	8
DS4	94	1436	12	8
DS5	33	8456	31	32
DS6	34	4157	31	32

cloud research, system log and workload trace are rarely publicly available. Commercial cloud providers usually treat such information confidential.

Eucalyptus cloud trace [16] was made available in 2014 and updated in 2015. Eucalyptus is an open source IaaS cloud computing framework [17], which is developed by Eucalyptus Systems Inc. The Eucalyptus cloud trace dataset consists of six workload traces which are captured from six production enterprise private cloud deployments. These cloud deployments host different types of workloads from software development and testing to product demonstration and sales. Each trace dataset represents the operation history of resource events happened in a single Eucalyptus availability zone. Each event records the timestamp, event type ($Start/Stop$ VM) and VM resource requested (only in $Start$ event). The VM types are configured based on different CPU cores. The traces also provide the VM instance id in every event so that we can match the $Start$ and $Stop$ event for a particular VM. Moreover, it contains the original VM scheduling decision made by production scheduler. The description of trace datasets is shown in Table II.

III. COMPARISON BETWEEN SYNTHETIC AND REAL-WORLD WORKLOAD

In this section, we present an in-depth multi-dimension comparison between the synthetic workloads and Eucalyptus workload traces. The goal is to figure out whether the synthetic workloads are able to reflect the workload pattern in the real-world.

A. Workload Comparison

We compare two types of workloads in terms of the three basic parameters in IaaS cloud workload: inter-arrival time, VM lifetime and VM type.

TABLE III
DISTRIBUTION FIT RESULT

	Exponential	Lognormal	Poisson
Inter-arrival time	μ = 2611	μ = 5.27, σ = 2.57	λ = 2611
VM lifetime	μ = 44740	μ = 7.04, σ = 1.99	λ = 44740

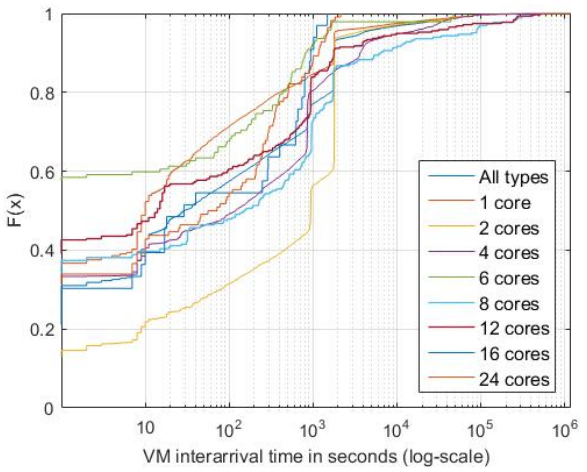

Fig. 2. CDF of VM inter-arrival time of Eucalyptus traces

We first show the Cumulative Distribution Function (CDF) of the VM inter-arrival time from Eucalyptus workload in Fig. 2. We aggregate the inter-arrival time from six traces and plot the CDF for each VM types. We also plot a CDF of all VMs. The distributions of different VM types are similar. We can only see slight difference of 8 and 12 cores VM. We also observe that approximately 40% VM provision requests are submitted at the same time, i.e. 0 inter-arrival time. Then we try to fit three statistical distributions that are listed in synthetic workloads of Table. I on aggregated data of all VMs. The fit result is shown in Table. III. By comparing the distribution parameters, we can easily notice that there is a significant distinction between synthetic workload distributions and real-world workload. The synthetic workloads generate new VM requests much faster than the real distribution.

Unlike the inter-arrival time, VM lifetime presents diverse CDF which means different VM types have different distributions when we look at the VM lifetime distributions of Eucalyptus workloads in Fig. 3. This observation invalidates

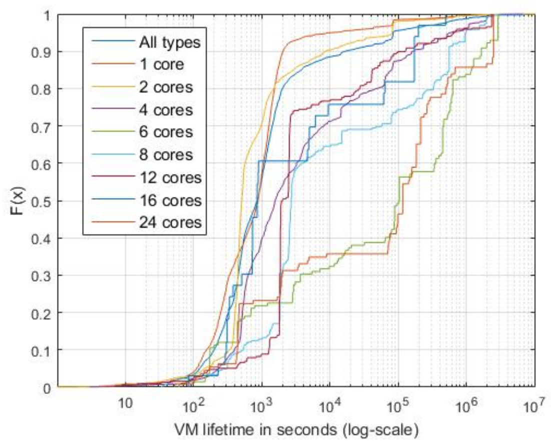

Fig. 3. CDF plot of VM life time in six traces

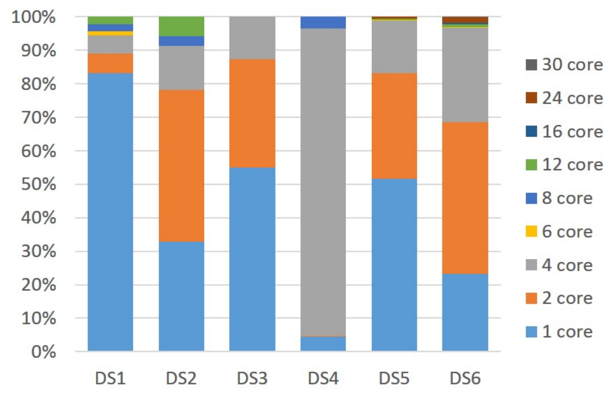

Fig. 4. The VM Type Distribution in Six Traces

the assumption made in synthetic workloads that all VM types follow the same distribution. Moreover, the distribution fit result, shown in Table. III, indicates the synthetic VM lifetime is much shorter than the real-world workload.

Lastly, we compare the VM type distribution in the workloads. Figure 4 depicts the percentage of each VM type in six Eucalyptus workload traces. Again, we can see a distinct percentage distribution on each workload trace. For example, one core VM takes up more than 80% in $DS1$ trace whereas only up to approximately 50% in the remaining traces. Moreover, four cores VM accounts for over 90% VM in $DS4$ trace. The only common pattern we can conclude from Fig. 4 is that the small VM (cores $\leq$ 4) accounts for more than 90% of total VM created in all six traces. Although synthetic workloads also have a variety of VM types, they either select a VM type for a request based on uniform probability [12] or pre-defined probability [7]. In contrast, real-world trace suggests that the probability of each VM type varies in different deployments.

B. *Discussion*

Through the comparison between the parameter distribution of synthetic workloads and Eucalyptus workload traces, we observe major differences and assumptions on synthetic workloads that may not be valid in our traces. In fact, it is challenging to choose an accurate statistical distribution to represent cloud user activity patterns in all scenarios. An early investigation [18] on Eucalyptus workload trace characterization has pointed out that there is no distribution that can achieve high goodness of fit, i.e. pass the Kolmogorov-Smirnov test, for VM lifetime or inter-arrival time in $DS1 - DS3$ traces. Similarly, Abdul-Rahman et al. concluded that the formal statistical distribution fits for resource configuration is unclear based on Google cluster trace [14]. Our findings on six Eucalyptus workload traces also confirm this conclusion.

IV. Realistic and Reproducible Experiment methodology for IaaS cloud research

In this section, we showcase a realistic and reproducible simulation experiment on IaaS resource scheduling problem. We replay the Eucalyptus workload traces as the simulation input thus the experiment is reproducible and can be cross-validated. The validity and effectiveness of such trace driven simulation has been proved in many related computational fields [19], [20].

The experiment aims to evaluate the performance of several resource scheduling algorithms. Generally, the on-demand IaaS cloud resource scheduling problem can be formalized as the online bin packing problem [8]. The virtualization host is treated as bins with limited space. A set of balls with different sizes, i.e. VM with different resource demand, need to be allocated in the bins. The objective is to use as fewer bins as possible. Fewer bins or active hosts means energy and cost saving. The online property indicates that the placement scheduler has no knowledge of incoming balls which match the nature of cloud on-demand resource provision.

Many heuristics have been developed for solving the NP-hard bin packing problem [21], such as $Best\ Fit$ and $First\ Fit$. The $Best\ Fit$ algorithm always place the next incoming ball in the most utilized bin, of course, it should have enough empty space to place the ball. While $First\ Fit$ perform a linear scan on the bin set and select the first bin that has enough empty space as placement destination. We also consider dynamic resource scheduling approach which has been proved effective in improving infrastructure resource utilization [5]. When a VM is terminated, the dynamic scheduling is triggered to see whether it is possible to migrate all VMs in an under-utilized host to other active hosts. If possible, we can perform VM migration and turn sleep the source host to further reducing power consumption. When choosing a migration destination for a VM, the dynamic scheduler also utilizes the basic heuristics.

A. *Simulation Design*

We simulate an IaaS cloud test-bed consisting of a number of virtualization hosts for each Eucalyptus workload trace. We only configure the CPU resource of the virtualization host because it is the requested resource recorded in the workload trace. Based on the VM type definition in Eucalyptus cloud, other resources such as memory and disk capacity of a VM are determined by the VM core [17]. The VM types are distinguished according to their CPU core count. The simulated cluster and hosts for each workload trace are configured based on Table. II.

In this experiment, we compare the resource utilization performance of five scheduling algorithms: $Best\ Fit$ (BF), $First\ Fit$ (FF), their migration enabled variants (BFm and FFm) and the scheduling algorithm used in original traces. We measure the number of active virtualization hosts in the cluster over time and the average CPU utilization of active hosts. These two metrics are able to reflect the effectiveness on maximizing resource utilization and minimizing power consumption of a scheduling algorithm. In addition, we also measure the number of VM migrated during simulation. It is known that VM migration comes with performance degradation [22]. Although people are working on reducing the performance impact during migration, the negative effect still cannot be ignored for now. The smaller number of migrated VM means less performance degradation of running VM.

The advantage of this approach over the same with a synthetic workload, is that the reproducibility and cross-validation. Moreover, there is no need to repeatedly perform the test and calculate statistical measurements such as confidence interval in a trace-driven experiment. The request streams recorded in the trace are deterministic and real. One can easily replicate the trace-driven experiment with the same scheduling approaches and simulated or real test-bed to reach the exact same conclusion.

B. Evaluation Result

Fig. 5 shows the changes on a number of active virtualization hosts over time. The lower the line indicates the better resource utilization under the same workload. Each subfigure represents the simulation result for a workload trace. We can easily observe that original trace utilize the largest number of active hosts, especially under heavy workloads ($DS1, DS5, DS6$). All hosts in the cluster are activated in most of the time. In the contrary, heuristic based algorithms are able to save nearly $\frac{1}{3}$ active hosts in $DS5, DS6$ workloads and even up to 60% at the beginning phase of $DS1$. In the result of other less heavy workload ($DS24$), the heuristic based algorithms perform better than the trace scheduling algorithm. When we compare the heuristic based algorithms, they do not present a huge distinction like before. The difference in the number of active hosts is limited in the range of one or two hosts. This is because both $Best\ Fit$ and $First\ Fit$ have the same upper bound, 1.7 times optimal number of used bins [21].

Then we present the CPU utilization results Table. IV. Not surprisingly, the utilization rate of trace scheduling result is the lowest in all six simulations. By applying the heuristic based algorithms, the average CPU utilization can be improved by up to 30%. Similarly to the active hosts' result, the difference between heuristic based algorithms is relatively small. However, we noticed that the migration enabled heuristics BFm and FFm outperforms basic heuristics in most cases, although the improvements are less than 5%. When we check the number of migrated VM in Table. V, comparing to the thousands of VM created in each trace, only a small portion of VM experienced migration for both algorithms. And the less VM migrated by BFm compare to FFm because BFm take into account the resource utilization factor when making initial placement and migration decisions.

TABLE IV
AVERAGE HOST CPU UTILIZATION RESULTS

	$DS1$	$DS2$	$DS3$	$DS4$	$DS5$	$DS6$
BF	82%	66%	47%	75%	78%	90%
FF	78%	64%	48%	74%	80%	86%
BFm	84%	67%	49%	75%	82%	90%
FFm	82%	65%	49%	74%	82%	87%
$Trace$	49%	55%	25%	63%	58%	66%

TABLE V
MIGRATED VM COMPARISION RESULT

	$DS1$	$DS2$	$DS3$	$DS4$	$DS5$	$DS6$
BFm	246	43	69	0	32	67
FFm	268	60	77	5	42	105

Based on the above observations, we can conclude that there is still a large potential of utilization improvements in production systems. The heuristic based algorithms are emerging techniques in academia but are rarely applied in real platforms. Even the basic heuristics are able to significantly increase the resource utilization without sacrificing the VM performance. One reason could be that the designer of a practical system are conservative on implementing new techniques unless they have solid evidence of improvement based on realistic workload. Reproducible trace-driven experiments are able to provide such evidence with real-world workload trace.

V. CONCLUSION

In this paper, we raise the question of how we can deliver realistic and reproducible outcome in IaaS cloud resource management research. By comparing the statistical distributions used in synthetic workloads with Eucalyptus workload trace, we identified a number of gaps such as time distributions with incorrect parameters and assumptions. Due to the fact that the real workload is hard to be modeled, we suggest that trace-driven experiment is preferable for generating a realistic and reproducible outcome. We showcase a trace-driven and simulation-based performance evaluation experiment for several resource scheduling algorithms. It is straightforward to replicate the experiment with the same trace input and same test bed configuration. Therefore, new techniques developed by research community can be easier to be implemented in practice.

REFERENCES

[1] P. Mell and T. Grance, "The nist definition of cloud computing," *Communications of the ACM*, vol. 53, no. 6, p. 50, 2010.

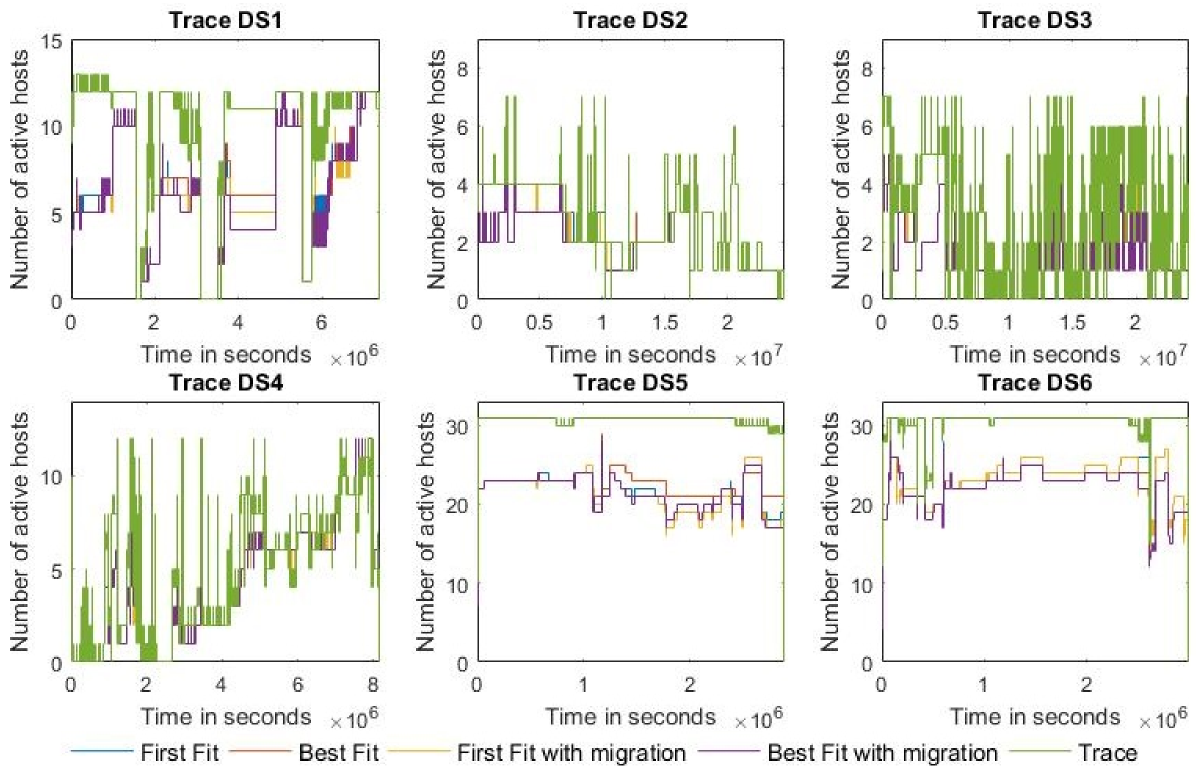

Fig. 5. The comparison on number of active hosts of different scheduling algorithms

[2] Z. Yao, I. Papapanagiotou, and R. D. Callaway, "Sla-aware resource scheduling for cloud storage," in *2014 IEEE 3rd International Conference on Cloud Networking*, Oct 2014, pp. 14–19.

[3] ——, "Multi-dimensional scheduling in cloud storage systems," in *2015 IEEE International Conference on Communications*, June 2015, pp. 395–400.

[4] B. Ravandi, I. Papapanagiotou, and B. Yang, "A black-box self-learning scheduler for cloud block storage systems," in *2016 IEEE 9th International Conference on Cloud Computing (CLOUD)*, June 2016, pp. 820–825.

[5] Z.-H. Zhan, X.-F. Liu, Y.-J. Gong, J. Zhang, H. S.-H. Chung, and Y. Li, "Cloud computing resource scheduling and a survey of its evolutionary approaches," *ACM Computing Surveys*, vol. 47, no. 4, p. 63, 2015.

[6] A. Pucher, E. Gul, R. Wolski, and C. Krintz, "Using trustworthy simulation to engineer cloud schedulers," in *Cloud Engineering (IC2E), 2015 IEEE International Conference on*. IEEE, 2015, pp. 256–265.

[7] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti, "Vm placement strategies for cloud scenarios," in *Cloud Computing (CLOUD), 2012 IEEE 5th International Conference on*. IEEE, 2012, pp. 852–859.

[8] Y. Li, X. Tang, and W. Cai, "On dynamic bin packing for resource allocation in the cloud," in *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures*. ACM, 2014, pp. 2–11.

[9] J. Yin, X. Lu, X. Zhao, H. Chen, and X. Liu, "Burse: A bursty and self-similar workload generator for cloud computing," *IEEE Transactions on Parallel and Distributed Systems*, vol. 26, no. 3, pp. 668–680, 2015.

[10] W. Willinger, M. S. Taqqu, W. E. Leland, and D. V. Wilson, "Self-similarity in high-speed packet traffic: analysis and modeling of ethernet traffic measurements," *Statistical science*, pp. 67–85, 1995.

[11] A. Wolke, M. Bichler, F. Chirigati, and V. Steeves, "Reproducible experiments on dynamic resource allocation in cloud data centers," *Information Systems*, vol. 59, pp. 98–101, 2016.

[12] K. Mills, J. Filliben, and C. Dabrowski, "Comparing vm-placement algorithms for on-demand clouds," in *Cloud Computing Technology and Science (CloudCom), 2011 IEEE Third International Conference on*. IEEE, 2011, pp. 91–98.

[13] D. G. Feitelson, *Workload modeling for computer systems performance evaluation*. Cambridge University Press, 2015.

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

[15] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2012.03.20. Posted at http://code.google.com/p/googleclusterdata/wiki/TraceVersion2.

[16] R. Wolski, "Eucalyptus workload trace," https://www.cs.ucsb.edu/~rich/workload/, 2015.

[17] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, and D. Zagorodnov, "The eucalyptus open-source cloud-computing system," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*. IEEE Computer Society, 2009, pp. 124–131.

[18] R. Wolski and J. Brevik, "Using parametric models to represent private cloud workloads," *IEEE Transactions on Services Computing*, vol. 7, no. 4, pp. 714–725, 2014.

[19] E. J. Koldinger, S. J. Eggers, and H. M. Levy, "On the validity of trace-driven simulation for multiprocessors," in *ACM SIGARCH Computer Architecture News*, vol. 19, no. 3. ACM, 1991, pp. 244–253.

[20] B. Black, A. S. Huang, M. H. Lipasti, and J. P. Shen, "Can trace-driven simulators accurately predict superscalar performance?" in *Computer Design: VLSI in Computers and Processors, 1996. ICCD'96. Proceedings., 1996 IEEE International Conference on*. IEEE, 1996, pp. 478–485.

[21] D. S. Johnson, A. Demers, J. D. Ullman, M. R. Garey, and R. L. Graham, "Worst-case performance bounds for simple one-dimensional packing algorithms," *SIAM Journal on Computing*, vol. 3, no. 4, pp. 299–325, 1974.

[22] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of virtual machine live migration in clouds: A performance evaluation," in *IEEE International Conference on Cloud Computing*. Springer, 2009, pp. 254–265.