The Concept of Workload Delay as a Quality-of-Service Metric for Consolidated Cloud Environments with Deadline Requirements

Evangelos Tasoulas, Hugo Lewi Hammer, Hårek Haugerud, Anis Yazidi, Alfred Bratterud and Boning Feng
Oslo and Akershus University College of Applied Sciences, Norway

***Abstract*—Virtual Machine (VM) consolidation in the cloud has received significant research interest. A large body of approaches for VM consolidation in data centers resort to variants of the bin packing problem which tries to minimize the number of deployed physical machines while meeting the Service-Level-Agreement (SLA) constraints. In this paper we introduce the concept of workload *delay* as a Quality-of-Service (QoS) metric that captures directly the resulting degradation that a cloud user would experience in the case where the SLA is violated. Our results, that are based on real-life trace-based simulations, show that consolidating VMs based on the level of utilization results in little control over the resulting delay, a particularly significant drawback when running jobs with deadline requirements, while we are able to control the delay much better if we take into account our suggested metric of the *delay*.**

I. Introduction

The use of cloud computing is increasing at a tremendous speed as a result of the use of virtual machines (VMs) having exploded in recent years. More and more companies are centralizing their resources to data centers to ensure uninterrupted power, better security, greater opportunities and availability. The number of data centers increased by 56 per cent worldwide from 2005 to 2010 [1], and more hardware is being installed to handle the rapidly increasing demand. Cloud computing now consumes more electricity every day than India [2], and Google alone consumes the same amount of energy as Norway's capital city [3].

The need to make cloud environments more energy-efficient is a major driver for data centers and cloud providers. The idea is simple in principle: always align the number of running physical machines (PMs) with the current demand. A lot of effort has been devoted to designing optimal solutions on how to consolidate VMs on PMs, e.g. see [4] for a review. The majority of the works consider the problem as a resource allocation problem and typically as a bin packing problem [5]. Perhaps the most common resource allocation strategy is to allocate VMs on PMs to achieve some level of utilization of the PMs [6], [7]. However it is not clear how the level of utilization is related to the performance of the VMs or the user experience for users using the VMs. E.g. if the VMs are packed with the aim of a high utilization of the PMs, the demand of the VMs may in some time periods go above the capacity of the PMs resulting in a degradation of the performance of the VMs and reduced user experience. In other words, the processing of tasks gets *delayed* compared to if the VMs were run on PMs with sufficient capacity. Such a degradation of the performance, or delay, is a direct measure of the user experience for users using the VMs. When consolidating VMs this measure therefore should be included as part of the methodology.

978-1-5386-1465-5/17/$31.00 ©2017 IEEE

II. Related Work

Lots of research has been revolved around mapping the problem of VM consolidation to the bin packing problem where PMs are assimilated to bins and VMs are assimilated to items. Generally, the aim is to reduce the number of PMs needed to pack different VMs across various resources such as CPU, memory, network I/O etc. Since bin packing is known to be an NP hard problem, several heuristics have been applied when it comes to VM consolidation, including the well established First-Fit-Decreasing (FFD), Best-Fit-Decreasing (BFD) or variants of these algorithms [8], [9], [10]. Other researchers have resorted to novel bio-inspired solutions, and perform the bin packing with algorithms such as the Ant Colony Optimization [7], [11].

The complexity of the consolidation increases exponentially as more dimensions are taken into account, hence, some studies reduce the problem to one dimensional bin packing by only considering the bottleneck resource. Wood et al. [12] devised *Sandpiper*, a consolidation system that uses a simple formula to combine dimensions associated to different resources (CPU, memory and network) in one metric, called *volume*, in order to quantify the load of VMs and PMs and deploy a classic one dimensional FFD bin packing algorithm.

An example of another dynamic consolidation approach is presented in [13], Beloglazov et al. propose to detect host overload using Markov chain modeling. The authors show that a necessary condition to improve the quality of VM consolidation is to maximize the mean time between migrations, and provide heuristics that trigger VM migrations while respecting this condition.

In [14], the authors present iPOEM, a consolidation system that tries to reduce energy consumption without violating SLA constraints. iPOEM tries to optimize two key parameters in a data center, the max CPU usage (CPU$_{high}$) a PM should tolerate before triggering a VM migration and the min CPU usage (CPU$_{low}$) for turning off a machine. The authors provide two theoretical results that are the core of the iPOEM algorithm: Service Level Agreement (SLA) violations can be reduced by reducing CPU$_{high}$, while the energy consumption can be decreased if CPU$_{low}$ is increased. Based on these two intuitive observations, iPOEM performs a guided binary search to achieve two conflicting objectives, namely, reducing power consumption while operating within the SLA constraints.

III. Working Dataset

All the experiments and results that are following in this work are based on a real-workload dataset that we obtained from the *University of Oslo (UiO)*. The dataset contains performance traces from more than 1500 all-purpose Linux-based physical and virtual machines (VMs), collected every one minute over a period of 13 days during June 2016. The collected data include CPU and memory utilization measurements per machine. Each measurement represents the average CPU utilization for the previous minute, and the current memory utilization.

In this work we focused only on the VMs and CPU data. After cleaning up the data and filtering out hosts with missing or corrupted data points, we ended up dealing with 405 VMs totaling 1030 virtual CPU cores (vCores). The majority of the VMs has only one or two vCores, while a few VMs have four, eight or twelve vCores.

IV. The Concept of Workload Delay

As described in the introduction, it is natural to take into account degradation of user experience, or delay, when allocating resources to virtual machines (VMs). In this section we present a framework for delay that can be applied to real life resource allocation problems.

Suppose that we have M virtual machines, $VM_1, VM_2, \ldots, VM_M$, running on a physical machine (PM) equipped with C CPU cores. We divide a day into T equidistant time intervals of length Δt and let $t \in 1, 2, \ldots, T$ refer to time interval t. The VMs initiate tasks that require CPU processing time of the PM. Let $S_m(t)$ denote the tasks initiated by VM_m during time interval t and let $d_m(t)$ denote the total processing time needed to process these tasks. Below we refer to $d_m(t)$ as the *demand* of VM_m at time interval t. We define $d(t)$ as the total processing time to process all the tasks initiated in time interval t from all the VMs, i.e. $d(t) = \sum_{m=1}^{M} d_m(t)$. If $d(t)$ is above the total processing capacity of the PM, i.e. $d(t) > C\Delta t$, the processing of the tasks of the VMs will run slower compared to if the VMs were run on a PM with a capacity above the demand $d(t)$. We say that the VMs get *delayed*. More precisely, let $q(t)$ be the total processing time of the tasks not being processed in time interval t, and therefore are *queued* to the next time step (gets delayed). The total processing time of the tasks in queue can be computed recursively by the relation $q(t+1) = \max\{q(t) + d(t+1) - C\Delta t, 0\}$.

In Fig. 1 we present an example of how the delay will evolve over time in a given system (PM) with 12 CPU cores, when a workload with a given *Resource demand* in percentage must be served. The system in Fig. 1. has a max capacity of $C = 1200\%$ since each CPU core can serve 100% worth of workload, meaning that the *Resource utilization*, $d(t)$, cannot exceed the *Max capacity* at any time. Thus, whenever the *Resource demand* exceeds the 1200% threshold ($d(t) > C\Delta t$), workload requests are being queued and consequently delayed as the tail of the queue grows (depicted as *Delay added* in Fig. 1) while the system operates at max capacity. When the *Resource demand* is below *Max capacity* and if we have queued requests from before, the delay will be reduced (depicted as *Delay consumed* in Fig. 1) and as a consequence the *Resource utilization* will be higher than the *Resource demand* until all of the queued requests have been served. Whenever delay is added or delay is consumed, the *Current delay* that is experienced by the newly appended workload will be affected, and the sum of all *Current delay* values gives us the *Total delay* for the given system in the given period of time.

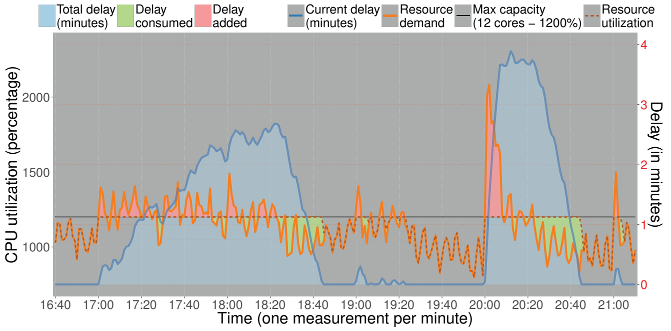

Fig. 1. How workload requests get delayed.

V. Applying the Concept of Delay for Consolidation

In this section we will demonstrate how the concept of delay can be engaged to directly control user experience when consolidating VMs. We use the dataset presented in section III to first bin-pack VMs based on the average utilization, and then based on the *delay*. We compare how the delay evolves against the total average utilization in a bin (a bin is the equivalent of a PM). First we apply the concept in the complete dataset in sections V-A and V-B, and then we perform the consolidation based on parts of our dataset, and cross-validate the delay on the missing parts of the dataset in section V-C.

A. Bin-pack based on average utilization

We apply the packing based on the one dimension Best-Fit-Decreasing (BFD) [15] bin packing algorithm, and each bin in our experiments equals to a hypervisor. The algorithm first sorts the VMs in a decreasing order based on the average utilization of each VM. Then the VMs are placed into bins of a given max capacity in order, and if a bin is overflowing, i.e. the total average utilization of the collocated VMs exceeds the max capacity, then the next available bin is tested. If all of the available bins overflow, the VM that caused the

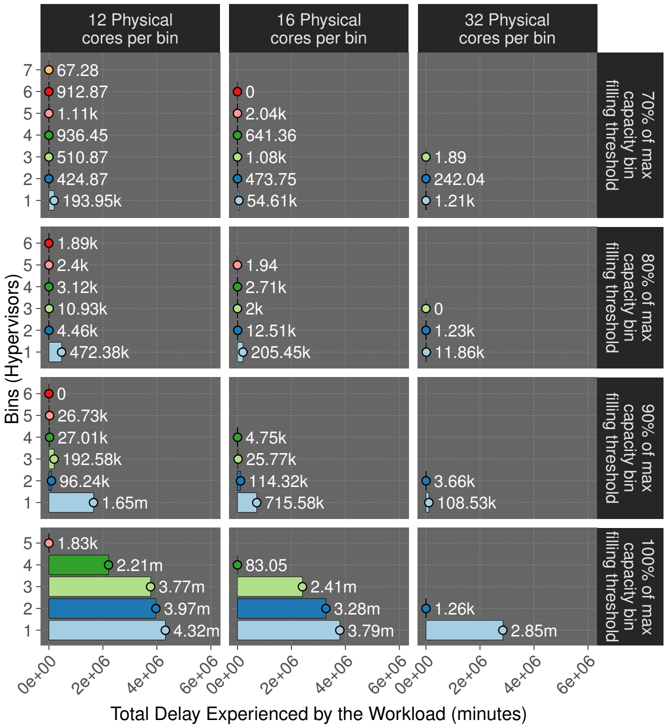

Fig. 2. Bin packing on the whole dataset based on average utilization.

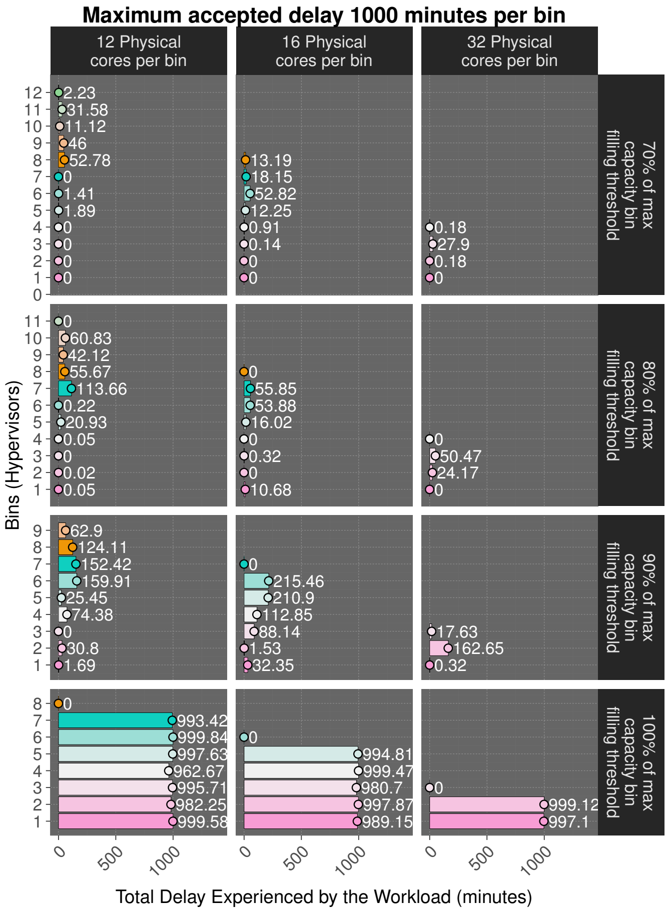

Fig. 3. Bin packing on the whole dataset based on the delay.

overflow is placed in a newly added bin. The capacity of each bin is defined by the number of physical cores that a hypervisor would have. We run experiments for bins with 12, 16 and 32 cores with a max capacity of 1200%, 1600% and 3200% respectively. Since we expect that when operating close to the max capacity of a hypervisor the total delay can be quite big depending on the workload, we also perform the bin packing as if the max capacity of each bin was $\{70\%, 80\%, 90\%, 100\%\} * max\_capacity$ (essentially underprovisioning the hypervisors), but still calculate the total delay based on the max capacity. The results of the bin packing based on the average utilization are illustrated in Fig. 2, and a few notes can be made:

- The average total delay per bin is less for the cases where *max capacity bin filling threshold* is less than 100%. In particular for the case where the bin filling threshold is 100% the total delay is "exploding".
- The last bin in all cases typically exhibits significantly lower total delay. That is due to the last bin always containing the *remaining* VMs that did not fit in previous bins, and most of the times the last bin is less utilized.
- For the same set of experiments in each facet of Fig. 2, the variance of the total delay between bins is significant, demonstrating the inability to control the delay efficiently.

B. Bin-pack based on delay

In section V-A we demonstrated the poor ability to control delay when the average utilization metric for bin packing was used. In Fig. 3 we perform a similar bin packing like the one explained in section V-A, but we now use the delay concept and try to control the delay by taking into account a *maximum accepted* total delay per bin during the bin packing process. For the case presented in Fig. 3 the maximum accepted delay per bin is set to 1000 minutes. That is 1000/13 minutes per day on average since our dataset contains data for 13 days (refer to section III). When we bin pack based on the delay, if the total delay when adding a VM in a bin exceeds the *max accepted delay per bin*, the VM will be placed in another bin. If no bin can meet the *max accepted delay per bin* requirement, then the VM will be placed in a new bin. The differences of bin packing based on the *total delay* (Fig. 3) when compared to bin packing based on the *average utilization* (Fig. 2) are mainly two:

- Fine control of the delay can be achieved. As shown in the case where *max capacity bin filling threshold* = 100%, the total delay for all bins is slightly below the *max accepted delay per bin* - 1000 minutes.
- More bins are used when packing based on the delay. This is an indication that the bins end up being less utilized. So the important aspect that we study in the next section (section V-C) is how much is the delay per bin with respect to the average utilization when bin-packing VMs

with the two different methods; based on delay or average utilization.

C. Evaluation of consolidation algorithms

In this section we cross validate the bin packing methods described in sections V-A and V-B. The goal is to evaluate in particular to what extent we are able to control the delay when bin packing based on delay is applied in a partially known dataset. We perform the cross validation by introducing a training period in our dataset, and the validation, i.e. the resulting bin packing, is applied to the rest of the dataset. The training period is using 12 days and validation is applied on the 13th test day. We run several experiments for all possible combinations of 12 (training) + 1 (test) days. The results are presented in Table I and Table II.

In Table I the bin packing is based on the average utilization and the experiments are executed for 12, 16 and 32 physical cores. The results are the average of these three cases after performing 8 experiments per case for *max capacity bin filling thresholds* of 30% to 100%. In Table II the bin packing is based on the delay and similarly to the case when we bin pack based on average utilization, the experiments are executed for 12, 16 and 32 physical cores. However, when we bin pack based on delay we have one more parameter, the *max accepted total delay*, thus, we execute a set of experiments for all the combinations of *max capacity bin filling thresholds* of 70%, 80%, 90% and 100% and *max accepted total delay* of 1, 10, 100 and 1000 minutes per day.

In an ideal case one would want all of the bins (hypervisors) to be perfectly utilized, while there is no delay. Unfortunately this is not the case in the real-world. As seen in Table I, when the bin packing is based on the average utilization of the training dataset, the resulting average bin filling is fairly close to the given bin filling threshold when applied on the test day, but the delay of the workload is very unpredictable and the max observed delay in certain bins can be devastating. When the bin packing algorithm tries to consolidate with any *bin filling threshold* value over 80%, the average delay per bin is measured in thousands, and certain bins exhibit total delays in the order of million minutes as seen in the *max total bin delay in bins* column. Moreover, a substantial amount of the total workload is experiencing delay: more than 23% on average for the case where *bin filling threshold* is 80% and more than 65% for the case where the bin filling is 100%.

On the other hand, if one looks at Table II an observation that can be made is that in all of the cases the bin packing algorithm does not manage to pack the bins close to the desired *bin filling threshold*. The average utilization is around 30% less than the desired threshold. Nonetheless, the average delay per bin is lying around the desired *max accepted total delay*. Most importantly, when test-cases with similar characteristics are compared between Table I and Table II, the bin packing based on delay in general performs better by providing higher average utilization and less deviation from the *max accepted delay* parameter. The *max accepted delay* parameter could be used as a threshold for an SLA between the cloud provider and its customers. For example, in the case where the filling threshold is 70% in Table I, the average bin delay is 783.49 minutes and the average bin filling is 63.9%. In this case the median bin delay is only 18.45, and there is a bin with a max observed total delay of 46706.97 minutes, while only 9.77% of the workload has been delayed on average with an average of 2.02 minutes. These numbers demonstrate huge performance deviations between bins. This case can be compared with the case from Table II where the *max accepted delay* is 1000 minutes and the filling threshold is 100%. In this case the bin packing based on delay provided virtually the same average delay per bin, 783.87 minutes instead of 783.49 minutes, but the median bin delay is much closer to the average, 306.67 minutes, and the max observed delay in a bin is ~4 times less, 11596.53 minutes, than in the case when the bin packing happened based on the average utilization. In other words, when we bin pack based on the delay, the performance is more predictable with less deviation from the targeted SLA.

VI. Closing Remarks

In this paper we introduced the *concept of delay* that can be used as a user-perceptible QoS metric to validate performance in computing systems. Further, we demonstrated how the concept of delay can be applied to a VM consolidation scenario in clouds with long-running jobs, that are typical in cluster and Big Data deployments. Our results showed that a simple bin packing based on the delay metric, can deliver more predictable performance when compared to a simple bin packing based on average utilization.

The current work is the groundwork to formulate and demonstrate the *concept of delay*, and we identify areas for consideration and future work:

- In our example bin-packing method that was used for demonstrating the concept of delay, we incorporated a simple BFD algorithm based on the average utilization of each VM. We expect that if more advanced workload characterization methods are to be used [16], [17], higher consolidation can be achieved for the same amount of *total delay* per hypervisor.
- The delay is a metric that can be affected non-linearly by several factors, and we identify as the two most important ones the type of the workload and the underlying hardware. The same workload may look different if running on different hardware, and particularly in virtualized environments the same workload may look different even if running on the same hardware but sharing over-provisioned resources [18], [19].

Acknowledgements

We would like to acknowledge Jarle Bjørgeengen and the *University Center for Information Technology* (USIT) at UiO, UiO for providing us with performance data from production systems in order to carry out this work.

TABLE I
STATISTICS FOR BIN PACKING BASED ON AVERAGE UTILIZATION

Bin filling threshold (pct)	Avg total delay per bin (mins)	Med total delay per bin (mins)	Max total bin delay in bins (mins)	Avg bin filling (pct)	Med bin filling (pct)	Max bin filling (pct)	Avg workload delayed in bins (pct)	Avg delay experienced by workload when $delay > 0$ (mins)
30%	7.05	0.00	1923.19	29.20%	29.67%	65.50%	0.14%	0.06
40%	32.81	0.00	4332.07	38.19%	39.46%	70.59%	0.36%	0.24
50%	78.58	0.06	5770.64	46.10%	49.27%	80.21%	0.94%	0.48
60%	191.35	3.58	9044.37	58.41%	58.75%	88.90%	2.43%	1.01
70%	783.49	18.45	46706.97	63.90%	68.75%	99.80%	9.77%	2.02
80%	2265.91	140.32	118346.79	70.92%	77.72%	110.45%	23.08%	3.39
90%	8417.15	1046.43	178980.07	82.74%	87.99%	119.21%	48.37%	8.24
100%	33891.16	9798.13	293550.53	90.26%	95.97%	130.18%	65.77%	25.51

TABLE II
STATISTICS FOR BIN PACKING BASED ON DELAY

Accepted total delay (mins)	**Bin filling threshold (pct)**	Avg total delay per bin (mins)	Med total delay per bin (mins)	Max total bin delay in bins (mins)	Avg bin filling (pct)	Med bin filling (pct)	Max bin filling (pct)	Avg workload delayed in bins (pct)	Avg delay experienced by workload when $delay > 0$ (mins)
1	70%	0.06	0.00	21.91	27.52%	28.92%	54.50%	0.01%	0.01
	80%	0.15	0.00	21.35	31.64%	33.07%	63.45%	0.03%	0.02
	90%	1.14	0.00	290.55	36.98%	37.96%	66.86%	0.09%	0.06
	100%	8.34	0.10	776.45	43.46%	45.62%	81.41%	0.48%	0.16
10	70%	0.22	0.00	23.43	34.33%	37.02%	58.13%	0.06%	0.05
	80%	0.65	0.00	69.21	39.59%	42.73%	62.44%	0.12%	0.09
	90%	1.97	0.01	94.91	43.46%	49.19%	72.84%	0.34%	0.13
	100%	28.84	2.67	1242.55	54.93%	59.91%	83.69%	1.83%	0.31
100	70%	1.13	0.00	81.32	41.37%	45.38%	64.47%	0.20%	0.11
	80%	2.04	0.02	47.34	43.61%	50.77%	68.97%	0.48%	0.13
	90%	8.86	2.09	393.87	55.16%	59.09%	77.69%	1.73%	0.21
	100%	118.82	31.02	3103.94	58.94%	68.71%	87.80%	12.07%	0.51
1000	70%	5.14	0.03	248.11	47.28%	52.46%	67.47%	1.28%	0.14
	80%	19.89	1.02	942.61	54.23%	60.16%	73.98%	4.67%	0.21
	90%	88.04	6.37	2732.73	59.76%	69.50%	83.62%	9.74%	0.42
	100%	783.87	306.67	11596.53	69.40%	81.50%	98.31%	27.39%	1.69

REFERENCES

[1] J. G. Koomey, "Worldwide electricity used in data centers," *Environmental Research Letters*, vol. 3, no. 3, p. 034008, 2008. [Online]. Available: http://stacks.iop.org/1748-9326/3/i=3/a=034008

[2] Green peace, "Click clean: How companies are creating the green internet," http://www.greenpeace.org/usa/wp-content/uploads/legacy/Global/usa/planet3/PDFs/clickingclean.pdf, Apr 2014, [Online; accessed 20-Dec-2016].

[3] The New York Times, "Google details, and defends, its use of electricity," http://www.nytimes.com/2011/09/09/technology/google-details-and-defends-its-use-of-electricity.html, Sept 2011, [Online; accessed 20-Dec-2016].

[4] M. H. Ferdaus and M. Murshed, "Energy-aware virtual machine consolidation in iaas cloud computing," in *Cloud Computing*. Springer, 2014, pp. 179–208.

[5] D. S. Johnson, "Fast algorithms for bin packing," *Journal of Computer and System Sciences*, vol. 8, no. 3, pp. 272–314, 1974.

[6] Y. C. Lee and A. Y. Zomaya, "Energy efficient utilization of resources in cloud computing systems," *The Journal of Supercomputing*, vol. 60, no. 2, pp. 268–280, 2012.

[7] E. Feller, L. Rilling, and C. Morin, "Energy-aware ant colony based workload placement in clouds," in *Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing*. IEEE Computer Society, 2011, pp. 26–33.

[8] N. Bobroff, A. Kochut, and K. Beaty, "Dynamic placement of virtual machines for managing sla violations," in *2007 10th IFIP/IEEE International Symposium on Integrated Network Management*. IEEE, 2007, pp. 119–128.

[9] A. Corradi, M. Fanelli, and L. Foschini, "Vm consolidation: A real case based on openstack cloud," *Future Generation Computer Systems*, vol. 32, pp. 118–127, 2014.

[10] B. Cao, X. Gao, G. Chen, and Y. Jin, "NICE: Network-aware VM Consolidation scheme for Energy Conservation in Data Centers," in *20th IEEE International Conference on Parallel and Distributed Systems (ICPADS)*, Dec 2014, pp. 166–173.

[11] M. H. Ferdaus, M. Murshed, R. N. Calheiros, and R. Buyya, "Virtual machine consolidation in cloud data centers using aco metaheuristic," in *European Conference on Parallel Processing*. Springer, 2014, pp. 306–317.

[12] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif, "Sandpiper: Black-box and gray-box resource management for virtual machines," *Computer Networks*, vol. 53, no. 17, pp. 2923–2938, 2009.

[13] A. Beloglazov and R. Buyya, "Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints," *IEEE Transactions on Parallel and Distributed Systems*, vol. 24, no. 7, pp. 1366–1379, 2013.

[14] H. Zhang, K. Yoshihira, Y.-Y. Su, G. Jiang, M. Chen, and X. Wang, "ipoem: a gps tool for integrated management in virtualized data centers," in *Proceedings of the 8th ACM international conference on Autonomic computing*. ACM, 2011, pp. 41–50.

[15] D. Simchi-Levi, "New worst-case results for the bin-packing problem," *Naval Research Logistics*, vol. 41, no. 4, p. 579, 1994.

[16] B. Speitkamp and M. Bichler, "A mathematical programming approach for server consolidation problems in virtualized data centers," *IEEE Transactions on Services Computing*, vol. 3, no. 4, pp. 266–278, Oct 2010.

[17] A. Khan, X. Yan, S. Tao, and N. Anerousis, "Workload characterization and prediction in the cloud: A multiple time series approach," in *IEEE Network Operations and Management Symposium*, April 2012, pp. 1287–1294.

[18] M. A. El-Refaey and M. A. Rizkaa, "Virtual systems workload characterization: An overview," in *18th IEEE International Workshops on Enabling Technologies: Infrastructures for Collaborative Enterprises*, June 2009, pp. 72–77.

[19] F. Azmandian, M. Moffie, J. G. Dy, J. A. Aslam, and D. R. Kaeli, "Workload characterization at the virtualization layer," in *2011 IEEE 19th Annual International Symposium on Modelling, Analysis, and Simulation of Computer and Telecommunication Systems*, July 2011, pp. 63–72.