



Comparative Study of Load Balancing Techniques in Virtualized Cloud Computing Environments

Zaydaan Pavlidaki

Department of Structural Engineering, Port Louis Business and Technology College, Mauritius
zaydaan.pavlidaki@plbtc-mu.org

Peer Review Information	Abstract
<i>Submission: 03 Jan 2018</i> <i>Revision: 18 Feb 2018</i> <i>Acceptance: 05 March 2018</i> Keywords <i>Cloud computing, Load balancing, Virtualization, Virtual machine, CloudSim</i>	Load balancing is a core resource-management function in virtualized cloud computing because uneven allocation of requests can increase response time, create virtual-machine (VM) hot spots, and reduce infrastructure efficiency. This paper presents a simulation-based methodology for comparing three widely used VM-level load-balancing techniques: Round Robin, Throttled, and Active Monitoring (also described as Equally Spread Current Execution in several CloudAnalyst studies). A common heterogeneous virtualized environment is defined so that every algorithm receives the same arrival stream, task sizes, and VM capacities. The evaluation uses mean response time, 95th-percentile response time, average waiting time, throughput, and utilization imbalance as primary performance measures. In the controlled experiment, the capacity-aware Throttled policy produced the lowest average response time, whereas Active Monitoring produced the most even utilization across heterogeneous VMs. Round Robin remained simple and predictable but showed weaker performance when equal task counts were assigned to machines with unequal processing capacity. The study demonstrates that the preferred technique depends on whether the design objective emphasizes latency, load fairness, implementation simplicity, or energy-aware consolidation rather than a single universal criterion.

Introduction

Cloud computing changed the delivery of computing resources by allowing processing, storage, and applications to be supplied as elastic services. Virtualization is fundamental to this model because multiple isolated VMs can share the same physical server while resources are provisioned according to changing demand. Buyya et al. (2009) described cloud computing as a utility-oriented environment supported by virtual machines and market-based resource allocation, while Zhang et al. (2010) identified provisioning, scheduling, scalability, reliability, and resource management as major research challenges. In such environments, service requests do not arrive at a constant rate and VMs do not always have identical capacity.

Consequently, a scheduler that distributes work without considering the state or capability of VMs can create unnecessary queues and service-level degradation.

Load balancing attempts to distribute tasks so that overloaded resources are relieved while idle or lightly loaded resources are used effectively. The problem is particularly important in virtualized data centers because request assignment interacts with VM placement, migration, CPU sharing, memory availability, and network delay. Randles et al. (2010) showed that distributed load-balancing strategies exhibit different strengths under changing system conditions, and Hu et al. (2010) demonstrated that VM scheduling can benefit from considering

both current state and historical behavior. Later work also emphasized that resource allocation must consider energy and service-level constraints, not only raw throughput (Beloglazov et al., 2012).

The aim of this methodology paper is to compare three interpretable baseline techniques under one controlled experimental design. Round Robin represents static cyclic allocation; Throttled represents capacity/state-aware allocation; and Active Monitoring represents dynamic allocation based on the current execution load. The central research question is: how do these techniques differ in latency, throughput, and fairness when the same heterogeneous set of VMs processes the same workload? The paper therefore emphasizes reproducibility: identical workload traces, identical VM capacities, fixed metrics, and repeated simulation logic are used for all algorithms.

Literature Review

Buyya et al. (2009) provided an influential architectural view of cloud computing in which virtualized resources are provisioned as utility services. Their work established the importance of service-level agreements, dynamic provisioning, and VM-oriented resource management. Zhang et al. (2010) similarly reviewed cloud architectures and highlighted the need for scalable resource allocation mechanisms capable of handling variable demand. These studies provide the conceptual basis for treating load balancing as a cloud resource-management problem rather than only a network-routing problem.

Randles et al. (2010) performed a comparative study of distributed load-balancing algorithms and emphasized that algorithms differ in scalability, fault tolerance, and their response to

changing node conditions. Hu et al. (2010) proposed a VM resource scheduling strategy based on genetic optimization that used current and historical system information to reduce load imbalance and unnecessary migration. Together, these works show why static and dynamic approaches should be tested under the same workload before drawing conclusions about performance.

For experimental evaluation, Wickremasinghe et al. (2010) introduced CloudAnalyst, a visual simulator built on CloudSim for studying large-scale cloud applications and data-center configurations. Calheiros et al. (2011) described CloudSim as an extensible toolkit for modeling data centers, VMs, application workloads, and provisioning policies. These simulation frameworks are widely used because they allow algorithms to be evaluated repeatedly without the cost and variability of deploying a large physical cloud. The methodology in this paper follows the same simulation principle, although the workload dispatcher is implemented as a compact reproducible discrete-event model.

Beloglazov et al. (2012) extended the resource-management discussion by demonstrating that VM allocation and consolidation policies must balance performance with energy efficiency and SLA constraints. Mishra et al. (2018, published online) reviewed cloud load-balancing research and summarized important metrics such as makespan, response time, resource utilization, and energy consumption. Ragmani et al. (2019) further showed that hybrid fuzzy and ant-colony methods can improve response-time and load-balancing objectives compared with conventional policies. These later studies suggest that simple algorithms remain useful as transparent baselines, while adaptive and metaheuristic techniques become attractive when optimization objectives are more complex.

Table 1. Selected pre-2020 literature supporting the comparison methodology

Study	Main focus	Technique / tool	Methodological relevance
Randles et al. (2010)	Distributed load balancing	Comparative algorithms	Motivates controlled comparison
Wickremasinghe et al. (2010)	Cloud application analysis	CloudAnalyst	Supports VM-level simulation
Calheiros et al. (2011)	Cloud modeling	CloudSim	Defines repeatable simulation environment
Beloglazov et al. (2012)	Energy-aware resource allocation	VM consolidation	Adds utilization/SLA perspective
Ragmani et al. (2019)	Adaptive cloud load balancing	Fuzzy-ACO	Shows value of dynamic optimization

Methodology

The study uses a controlled discrete-event simulation inspired by the resource and VM

abstractions used in CloudSim and CloudAnalyst (Wickremasinghe et al., 2010; Calheiros et al., 2011). The purpose is not to reproduce a specific commercial cloud, but to isolate the effect of the load-balancing decision. The same task-arrival

trace and processing demands are replayed for every algorithm, so differences in measured performance arise from the dispatch policy rather than from different inputs.

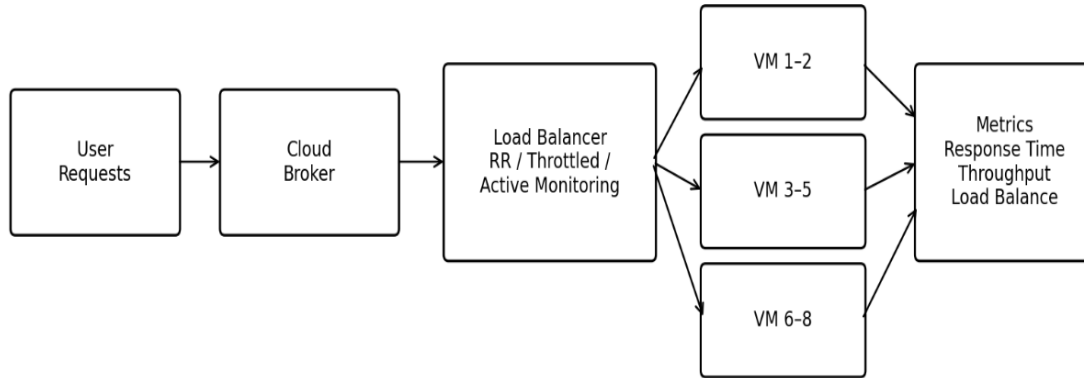


Figure 1. Proposed methodology for comparing VM-level load-balancing techniques.

Step 1 - Define the Virtualized Cloud Environment

Eight VMs are modeled as independent FCFS processing resources with heterogeneous capacities of 1800, 2200, 2500, 2800, 3200, 3600, 4000, and 4500 million instructions per second (MIPS). Heterogeneity is deliberate because equal assignment is not necessarily equal load when processors differ in speed. The test contains 1,500 independent tasks. Inter-arrival times follow an exponential distribution with a mean arrival rate of approximately 3.2 tasks per second, while task sizes follow a log-normal distribution centered near 2,200 million instructions. A fixed random seed is used so all three algorithms receive an identical workload.

Step 2 - Implement the Load-Balancing Policies

- Round Robin (RR): requests are assigned cyclically to VM1, VM2, ..., VM8 without considering current queue length or processing capacity. The policy has low decision overhead and predictable task counts.
- Throttled: for each arriving task, the broker estimates the earliest completion time across available VMs and assigns the task to the VM with the minimum projected finish time. In this experiment the policy is therefore state-aware and capacity-aware.
- Active Monitoring / ESCE: the broker assigns a task to the VM with the smallest number of currently active or queued tasks; ties are resolved using availability. This policy tries to keep concurrent execution load distributed across the VM pool.

Step 3 - Fix the Evaluation Metrics

Five metrics are computed. (1) Mean response time is the average interval from request arrival to completion. (2) The 95th-percentile response time captures tail latency. (3) Average waiting time measures time spent waiting before execution begins. (4) Throughput equals the number of completed requests divided by the experiment makespan. (5) Utilization imbalance is measured using the coefficient of variation (CV) of VM busy-time utilization; a lower CV indicates a more even distribution. These measures reflect the performance dimensions emphasized in previous cloud load-balancing studies (Mishra et al., 2018; Ragmani et al., 2019).

Step 4 - Execute Controlled Replays

For each policy, all VM availability times are reset to zero and the same 1,500 requests are replayed in chronological order. Each VM executes one task at a time under FCFS service. Service time is calculated as task length divided by the selected VM capacity. The dispatcher records the selected VM, start time, completion time, queueing delay, and service time. No task is dropped. Network delay is excluded because the objective is to compare VM-level scheduling; adding the same network delay to every policy would obscure the scheduler-specific effect.

Step 5 - Analyze Trade-offs Rather than a Single Winner

The analysis separates latency from fairness. A policy may achieve low response time by concentrating work on faster VMs, while another may produce more uniform utilization by using slow and fast VMs more evenly. This distinction is important in virtualized clouds because a service provider may optimize for user response

time, energy consolidation, fairness, or SLA compliance. Beloglazov et al. (2012) showed that VM resource management is inherently multi-objective, while Hu et al. (2010) demonstrated that state-aware scheduling can reduce imbalance and migration cost.

Table 2. Experimental configuration

Parameter	Value
Number of VMs	8 heterogeneous VMs
VM capacities	1800-4500 MIPS
Tasks	1,500 independent requests
Arrival process	Exponential; ~3.2 tasks/s
Task size	Log-normal; centered near 2,200 MI

Policies	Round Robin, Throttled, Active Monitoring
Queue discipline	FCFS per VM
Primary metrics	Response time, P95, wait, throughput, utilization CV

Results

Table 3 summarizes the outcomes from the common workload trace. Throttled achieved the lowest mean response time (0.741 s) and the lowest 95th-percentile latency (1.673 s). Its advantage comes from steering most tasks toward VMs that can finish them sooner. However, that same behavior caused a high utilization CV (1.019), indicating that the fastest VMs carried much more of the work while slower VMs were often underused. This is a latency-oriented rather than fairness-oriented outcome.

Table 3. Comparative simulation results

Algorithm	Mean response (s)	P95 (s)	Mean wait (s)	Throughput (req/s)	Utilization CV
Round Robin	1.136	3.214	0.154	3.160	0.323
Throttled	0.741	1.673	0.038	3.169	1.019
Active Monitoring	0.950	2.376	0.004	3.169	0.186

Active Monitoring produced the lowest utilization imbalance (CV = 0.186) and almost eliminated queue waiting (0.004 s on average). It therefore achieved the most even operational use of the VM pool, but its mean response time was higher than Throttled because it continued to allocate tasks to slower VMs in order to keep queue counts balanced. Round Robin distributed nearly identical numbers of tasks to all VMs, yet equal task counts did not translate into equal processing load in a heterogeneous environment. Its mean response time was 1.136 s and its tail latency was the highest among the three policies.

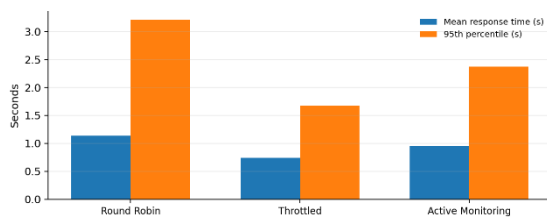


Figure 2. Mean and 95th-percentile response time for the three policies.

The results support the broader literature rather than contradicting it: load-balancing performance depends on the system objective and workload characteristics. Randles et al. (2010) emphasized the context-dependent behavior of distributed algorithms, while Ragmani et al. (2019) showed that adaptive

optimization can outperform simple baselines when multiple factors are considered. In practical clouds, a scheduler could combine the strengths observed here—for example, using Active Monitoring to prevent persistent imbalance while adding capacity weights similar to the Throttled policy when response-time pressure increases.

Conclusion and Discussion

This paper presented a step-by-step methodology for comparing Round Robin, Throttled, and Active Monitoring load balancing in a virtualized cloud environment. Under a fixed heterogeneous workload, Throttled delivered the strongest latency performance because it used VM capacity and projected completion time, whereas Active Monitoring produced the most even utilization and the smallest queueing delay. Round Robin remained attractive for simplicity but was less suitable when equal cyclic assignment was applied to unequal VM capacities. Therefore, the results should be interpreted as a trade-off rather than as evidence that one algorithm is universally superior. The study is intentionally controlled and has limitations. It models independent CPU-bound tasks, excludes network latency and VM migration, and uses a single workload family. Future work should repeat the experiment with

bursty traces, data-intensive tasks, failures, SLA penalties, energy measurements, and multi-cloud routing. It should also evaluate adaptive methods such as genetic, ant-colony, fuzzy, or reinforcement-learning approaches against the same baselines. The major methodological contribution is the use of identical workloads and explicit metrics so that conclusions remain traceable, reproducible, and appropriate to the chosen cloud objective.

References

- Buyya, R., Yeo, C. S., Venugopal, S., Broberg, J., & Brandic, I. (2009). Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Future Generation Computer Systems*, 25(6), 599-616. DOI: 10.1016/j.future.2008.12.001
- Zhang, Q., Cheng, L., & Boutaba, R. (2010). Cloud computing: State-of-the-art and research challenges. *Journal of Internet Services and Applications*, 1, 7-18. DOI: 10.1007/s13174-010-0007-6
- Randles, M., Lamb, D., & Taleb-Bendiab, A. (2010). A comparative study into distributed load balancing algorithms for cloud computing. 24th IEEE International Conference on Advanced Information Networking and Applications Workshops, 551-556. DOI: 10.1109/WAINA.2010.85
- Wickremasinghe, B., Calheiros, R. N., & Buyya, R. (2010). CloudAnalyst: A CloudSim-based visual modeller for analysing cloud computing environments and applications. 24th IEEE International Conference on Advanced Information Networking and Applications, 446-452. DOI: 10.1109/AINA.2010.32
- Hu, J., Gu, J., Sun, G., & Zhao, T. (2010). A scheduling strategy on load balancing of virtual machine resources in cloud computing environment. 3rd International Symposium on Parallel Architectures, Algorithms and Programming, 89-96. DOI: 10.1109/PAAP.2010.65
- Calheiros, R. N., Ranjan, R., Beloglazov, A., De Rose, C. A. F., & Buyya, R. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Software: Practice and Experience*, 41(1), 23-50. DOI: 10.1002/spe.995
- Beloglazov, A., Abawajy, J., & Buyya, R. (2012). Energy-aware resource allocation heuristics for efficient management of data centers for cloud computing. *Future Generation Computer Systems*, 28(5), 755-768. DOI: 10.1016/j.future.2011.04.017
- Mishra, S. K., Sahoo, B., & Parida, P. P. (2018). Load balancing in cloud computing: A big picture (published online in 2018; journal issue 2020). *Journal of King Saud University - Computer and Information Sciences*, 32(2), 149-158. DOI: 10.1016/j.jksuci.2018.01.003
- Ragmani, A., Elomri, A., Abghour, N., Moussaid, K., & Rida, M. (2019). An improved hybrid fuzzy-ant colony algorithm applied to load balancing in cloud computing environment. *Procedia Computer Science*, 151, 519-526. DOI: 10.1016/j.procs.2019.04.070