



An Efficient Load Balancing Technique for Distributed Cloud Computing Environments

Taneesha El-Masry

Department of Computer Science and Engineering, Kelana Technical and Management College, Malaysia
taneesha.el.masry@ktmc-my.net

Peer Review Information

Submission: 16 Jan 2018

Revision: 12 Feb 2018

Acceptance: 19 March 2018

Keywords

Cloud Computing, Load Balancing, Virtual Machines, Task Scheduling, Resource Utilization

Abstract

Distributed cloud computing depends on efficient assignment of dynamically arriving tasks to heterogeneous virtual machines (VMs). Static policies such as Round Robin are simple, but they do not react to queue growth, processing-speed differences, memory pressure, or network demand. This paper develops a methodology for an Adaptive Weighted Resource Load Balancing (AWRLB) technique that selects a VM using a composite score derived from projected response time, CPU demand, memory fit, and bandwidth fit. The design is motivated by cloud load-balancing literature published up to 2018, which identifies response time, makespan, throughput, resource utilization, scalability, and overload avoidance as major performance criteria. A reproducible discrete-event simulation with eight heterogeneous VMs and 1,000 dynamically arriving tasks is used to compare AWRLB with Round Robin and a simple Least-Queue policy over ten random seeds. The proposed method reduces mean response time from 24.86 s under Round Robin to 17.11 s, decreases makespan from 216.11 s to 109.51 s, and raises throughput from 4.66 to 9.13 tasks/s. The coefficient of variation of VM utilization falls from 0.366 to 0.013, indicating a substantially more even use of resources. The methodology therefore provides a practical framework for testing resource-aware scheduling before implementation in a production cloud or a full CloudSim-based environment.

Introduction

Cloud computing extends distributed computing by delivering pooled processing, storage, and software resources as elastic services. Early cloud research emphasized on-demand provisioning, virtualization, scalability, and service quality as defining characteristics of the model (Zhang, Cheng, & Boutaba, 2010). Simulation environments such as CloudSim subsequently enabled repeatable evaluation of provisioning and scheduling policies without the cost of deploying large physical data centers (Calheiros et al., 2011). As workloads became larger and more variable, the problem shifted from merely

provisioning resources to deciding how incoming work should be distributed among available VMs. Load balancing attempts to prevent simultaneous overload and underutilization. A cloud may contain VMs with different processing rates, memory sizes, queue lengths, and network capacities; consequently, an allocation that is suitable at one instant can become inefficient after only a few task arrivals. Al Nuaimi et al. (2012) identified this dynamic behavior as a central challenge and distinguished approaches according to how they obtain and use system-state information. Shaw and Singh (2014) similarly argued that scheduling and load

balancing are tightly coupled because the selected server must both accept a request and map it to a suitable VM. Static algorithms have low overhead, but dynamic algorithms are better suited to heterogeneous and changing environments.

The performance objective is multi-dimensional. A policy that minimizes the number of tasks on a VM may still send a CPU-intensive job to a slow processor; a policy based only on CPU speed may ignore a long queue, high memory occupancy, or network transfer time. Energy-aware research further shows that unbalanced allocation can also increase operational cost by keeping more hosts active than necessary (Beloglazov, Abawajy, & Buyya, 2012). Reviews published before 2018 consistently list response time, makespan, throughput, resource utilization, migration overhead, scalability, and fault tolerance among the main evaluation metrics (Milani & Jafari Navimipour, 2016; Jafarnejad Ghomi, Rahmani, & Qader, 2017).

This paper addresses these issues through a resource-aware methodology rather than a single-resource rule. The proposed AWRLB technique combines the current backlog of each VM with the estimated execution cost of the arriving task and its memory and bandwidth demand. The research question is: can a low-complexity dynamic score improve both task latency and utilization balance in a heterogeneous distributed cloud compared with static Round Robin and a queue-count baseline? The methodology is designed to be reproducible and to remain compatible with standard cloud-simulation concepts.

Literature Review

The load-balancing literature evolved from broad taxonomies toward techniques that incorporate more detailed resource states. Al Nuaimi et al. (2012) surveyed cloud-specific algorithms and highlighted heterogeneity, scalability, point-of-failure risk, and performance measurement as

open challenges. Shaw and Singh (2014) reviewed scheduling and migration mechanisms, emphasizing that dynamic policies can react to changes after execution begins. Pattanaik, Roy, and Pattnaik (2015) compared dynamic approaches in CloudAnalyst and evaluated response time and data-center processing time, illustrating the usefulness of simulator-based comparison for cloud policies.

A systematic review by Milani and Jafari Navimipour (2016) classified methods as static and dynamic and concluded that no single technique optimizes all cloud metrics. Their analysis favored adaptive or hybrid strategies when workload conditions change. Jafarnejad Ghomi et al. (2017) expanded the taxonomy to include general, agent-based, natural-phenomena-based, application-oriented, network-aware, workflow-specific, and Hadoop/MapReduce approaches. They also observed that research increasingly considered energy efficiency in addition to response time and utilization.

Heterogeneous VM placement and service allocation became particularly important. Xu, Tian, and Buyya (2017) reviewed VM-placement load balancing and described the problem as NP-hard because the resource dimensions and placement choices grow rapidly with system size. Chen, Chen, and Kuo (2017) proposed a cloud load-balancing architecture that considers both processing capability and current computer load, demonstrating why capacity should not be treated as homogeneous. Dhari and Arif (2017) similarly used VM capacity, VM load state, and estimated task execution time in their Load Balancing Decision Algorithm. In 2018, Xu et al. applied ant-colony optimization to multidimensional VM allocation, while Ye and Zhang (2018) used an improved ACO formulation for task scheduling. These studies show the value of resource-aware decisions but also introduce metaheuristic search overhead that may be unnecessary for every online request.

Table 1. Selected pre-2019 literature supporting the proposed methodology

Study	Approach	Key metric / contribution	Methodological implication
Al Nuaimi et al. (2012)	Survey	Cloud load-balancing challenges and algorithms	Use dynamic state information in heterogeneous clouds.
Pattanaik et al. (2015)	Simulation comparison	Response and data-center processing time	Benchmark proposed policy against simple baselines.
Milani & Navimipour (2016)	Systematic review	Static/dynamic taxonomy; open issues	Evaluate more than one QoS/resource metric.
Xu et al. (2017)	VM placement survey	Heterogeneity and NP-hard allocation	Include processing capacity and VM state.
Chen et al. (2017)	Dynamic CLB	Processing power plus current load	Combine capacity and load in the decision rule.

Dhari & Arif (2017)	LBDA	Makespan and response time	Estimate task execution time before assignment.
Xu et al. (2018); Ye & Zhang (2018)	ACO-based methods	Multidimensional allocation / scheduling	Resource-aware optimization can improve balance, but online overhead should be controlled.

The gap addressed here is therefore not the absence of sophisticated optimization, but the need for a transparent online rule that uses multiple resource dimensions without iterative metaheuristic search. AWRLB adopts the multi-resource reasoning of later studies while retaining constant-time scoring per candidate VM, making it suitable for distributed service controllers where decisions must be made for every arriving request.

Methodology

1. Research Design and System Model

A quantitative simulation methodology is used. The cloud is represented by a set $V = \{v1, v2, \dots, vm\}$ of heterogeneous VMs and a stream $T = \{t1, t2, \dots, tn\}$ of independent tasks. Each VM v_j has a processing rate P_j in million instructions per second (MIPS), memory capacity M_j , and bandwidth B_j . Each task t_i has an arrival time a_i , CPU length L_i in million instructions, memory demand m_i , and data-transfer demand b_i . Tasks are non-preemptive and are processed first-come-first-served within the selected VM. This abstraction is consistent with the task/VM models commonly used in CloudSim-based studies (Calheiros et al., 2011) and load-balancing evaluations (Dhari & Arif, 2017).

2. Proposed Adaptive Weighted Resource Load Balancing (AWRLB)

For every arriving task, the controller first filters VMs that cannot satisfy the task memory requirement. For each eligible VM, it estimates the time at which the VM will become free and calculates the task service time. CPU execution time is approximated by L_i/P_j . A bandwidth component is added from the task transfer demand relative to B_j , and a modest penalty is applied when the task consumes more than half of the VM memory. The projected response term R_{ij} is the waiting time already queued on v_j plus the estimated service time of t_i .

The assignment score is defined as $S_{ij} = 0.55R_{ij} + 0.25C_{ij} + 0.12M_{ij} + 0.08B_{ij}$, where $C_{ij} = L_i/P_j$ is normalized CPU cost, $M_{ij} = m_i/M_j$ is memory pressure, and $B_{ij} = b_i/B_j$ is bandwidth pressure. The weights give highest priority to user-perceived completion time while retaining sensitivity to resource fit. The task is assigned to the VM with the minimum S_{ij} . After completion is scheduled, the VM availability and utilization counters are updated. The controller therefore reacts to both current queue state and heterogeneous capacity without an iterative global search.

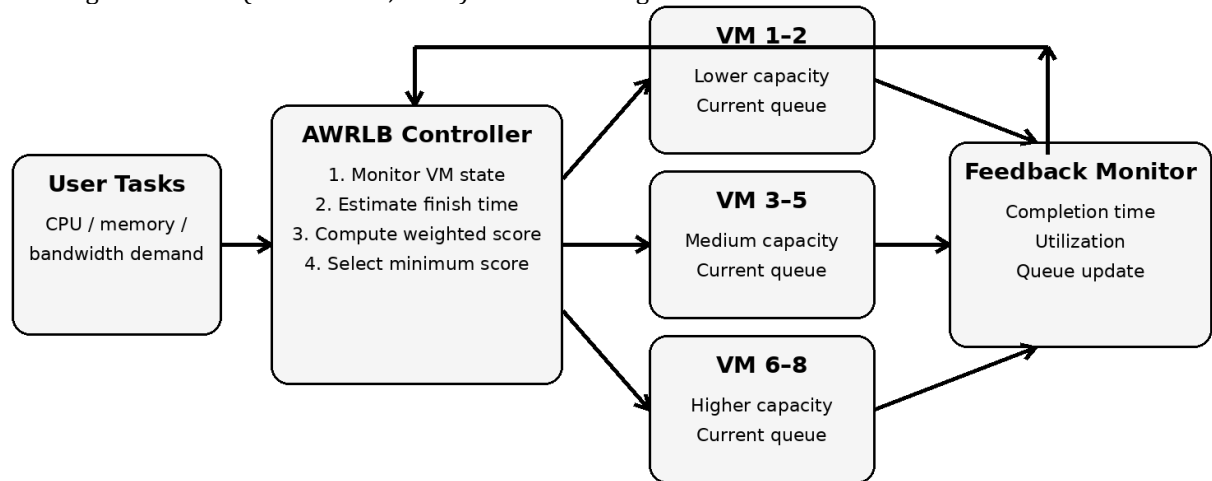


Figure 1. AWRLB task-allocation workflow with continuous VM-state feedback.

3. Step-by-Step Algorithm

- Receive a task and record its arrival time, CPU length, memory demand, and transfer demand.
- Read the latest VM availability time and static VM capacities (MIPS, memory, and bandwidth).
- Remove VMs whose memory capacity is below the task requirement.

- For each remaining VM, estimate waiting time, CPU execution time, bandwidth delay, and memory-pressure penalty.
- Compute the weighted AWRLB score S_{ij} and select the VM with the smallest value.
- Assign the task, update the VM's projected finish time, and record utilization statistics.
- Repeat for all task arrivals and calculate response time, makespan, throughput, and utilization imbalance.

4. Experimental Configuration and Baselines

A discrete-event simulator was implemented for this methodology so that the numerical results reported below are generated directly from the

stated assumptions rather than copied from earlier studies. Eight heterogeneous VMs are used, ranging from 1,200 to 4,000 MIPS, 2–16 GB memory, and 100–500 MB/s bandwidth. Each run contains 1,000 tasks with exponentially distributed inter-arrival times (mean 0.075 s). CPU lengths follow a lognormal distribution, while memory and transfer demands are sampled from discrete workload classes. Ten random seeds are used and results are averaged. The baselines are (i) Round Robin, which cycles through feasible VMs, and (ii) Least-Queue, which selects the VM with the lowest number of assigned tasks without correcting for VM capacity. These baselines intentionally represent low-overhead policies against which the benefit of resource awareness can be isolated.

Table 2. Simulation configuration

Parameter	Value
Virtual machines	8 heterogeneous VMs
Processing capacity	1,200–4,000 MIPS
Memory	2,048–16,384 MB
Bandwidth	100–500 MB/s
Tasks per run	1,000
Arrival process	Exponential, mean inter-arrival 0.075 s
CPU demand	Lognormal task length
Replications	10 random seeds
Baselines	Round Robin; Least-Queue
Metrics	Mean response, P95 response, makespan, throughput, utilization CV

5. Performance Measures

Response time is completion time minus arrival time; its mean and 95th percentile capture average and tail latency. Makespan is the interval from the first arrival to the final completion. Throughput is the number of completed tasks divided by makespan. Utilization for each VM is busy time divided by the experiment horizon. The coefficient of variation (CV) of utilization, defined as the standard deviation divided by the mean across VMs, is used as the imbalance indicator; a value nearer zero means the load is distributed more evenly. These measures are consistent with the response, makespan, throughput, and utilization criteria emphasized in earlier reviews and experiments (Pattanaik et al., 2015; Milani &

Jafari Navimipour, 2016; Jafarnejad Ghomi et al., 2017).

Results

Table 3 reports the average of ten simulation replications. AWRLB achieved a mean response time of 17.11 s compared with 24.86 s for Round Robin and 29.94 s for Least-Queue. Relative to Round Robin, this is a 31.18% reduction. The improvement is larger for tail latency: the average 95th-percentile response time fell from 90.97 s to 32.17 s, a 64.63% reduction. This occurs because the proposed score avoids repeatedly feeding slower VMs when their backlog is already high.

Table 3. Average performance over ten simulation replications

Algorithm	Mean response (s)	P95 response (s)	Makespan (s)	Throughput (tasks/s)	Utilization CV
Round Robin	24.86	90.97	216.11	4.66	0.366
Least-Queue	29.94	120.92	263.85	3.81	0.481
AWRLB	17.11	32.17	109.51	9.13	0.013

Makespan also improved from 216.11 s under Round Robin to 109.51 s under AWRLB, a 49.33% reduction, while throughput increased by approximately 96.06% from 4.66 to 9.13 tasks/s. The utilization CV decreased by 96.45%, from 0.366 to 0.013. The very low CV indicates that the VMs spend similar proportions of the experiment horizon doing useful work even though their capacities differ. Least-Queue performed worst because equal task counts do not mean equal computational work; a slow VM receiving the same number of tasks as a fast VM develops a longer backlog. This result reinforces the capacity-awareness arguments of Chen et al. (2017) and Xu et al. (2017).

The results should be interpreted as evidence for the methodology under the stated workload model, not as a claim that the same percentages will hold for every cloud. Real applications may contain dependencies, priorities, failures, VM migration, data locality constraints, and time-varying network conditions. Nevertheless, the experiment answers the research question positively: combining projected response with CPU, memory, and bandwidth fit can improve both latency and load balance without the search overhead of population-based metaheuristics. The design is therefore a suitable candidate for further validation in CloudSim, CloudAnalyst, or an experimental Kubernetes/OpenStack testbed.

Conclusion and Discussion

This methodology paper developed AWRLB, a dynamic load-balancing technique for distributed cloud computing environments. The approach was derived from limitations documented in literature available by 2018: static assignment ignores runtime state, task-count balancing ignores heterogeneous processing capacity, and single-metric methods can create bottlenecks in memory or networking. AWRLB addresses these limitations with a compact weighted score that can be recalculated for each task using current VM availability and resource ratios. The simulation demonstrates that the proposed rule can reduce mean and tail response time, shorten makespan, increase throughput, and equalize utilization relative to two simple baselines. The result is consistent with the broader direction of pre-2019 research toward multi-resource and adaptive allocation (Dhari & Arif, 2017; Xu et al., 2018; Ye & Zhang, 2018). At the same time, AWRLB is intentionally simpler than ant-colony or other metaheuristic techniques, which may make it more practical for high-frequency online decisions. Future work should test sensitivity to the score weights, incorporate energy consumption and SLA penalties, support dependent workflows and

priority classes, and evaluate failures or VM migration. A further extension could distribute the controller itself to remove a single decision point. Most importantly, implementation should be repeated in a recognized cloud simulator or testbed with publicly documented workloads so that external researchers can reproduce the findings. Within the scope of this methodology, AWRLB provides a clear and computationally lightweight foundation for efficient resource-aware load balancing in heterogeneous distributed clouds.

References

- Al Nuaimi, K., Mohamed, N., Al Nuaimi, M., & Al-Jaroodi, J. (2012). A survey of load balancing in cloud computing: Challenges and algorithms. *Proceedings of the IEEE Second Symposium on Network Cloud Computing and Applications*, 137–142.
<https://doi.org/10.1109/NCCA.2012.29>
- 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.
<https://doi.org/10.1016/j.future.2011.04.017>
- 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.
<https://doi.org/10.1002/spe.995>
- Chen, S.-L., Chen, Y.-Y., & Kuo, S.-H. (2017). CLB: A novel load balancing architecture and algorithm for cloud services. *Computers & Electrical Engineering*, 58, 154–160.
<https://doi.org/10.1016/j.compeleceng.2016.01.029>
- Dhari, A., & Arif, K. I. (2017). An efficient load balancing scheme for cloud computing. *Indian Journal of Science and Technology*, 10(11), 1–8.
<https://doi.org/10.17485/ijst/2017/v10i11/110107>
- Jafarnejad Ghomi, E., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: A survey. *Journal of Network and Computer Applications*, 88, 50–71.
<https://doi.org/10.1016/j.jnca.2017.04.007>
- Milani, A. S., & Jafari Navimipour, N. (2016). Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and*

Computer Applications, 71, 86–98.
<https://doi.org/10.1016/j.jnca.2016.06.003>

Pattanaik, P. A., Roy, S., & Pattnaik, P. K. (2015). Performance study of some dynamic load balancing algorithms in cloud computing environment. 2015 2nd International Conference on Signal Processing and Integrated Networks (SPIN), 619–624.
<https://doi.org/10.1109/SPIN.2015.7095363>

Shaw, S. B., & Singh, A. K. (2014). A survey on scheduling and load balancing techniques in cloud computing environment. 2014 International Conference on Computer and Communication Technology (ICCCT), 87–95.
<https://doi.org/10.1109/ICCCT.2014.7001474>

Xu, M., Tian, W., & Buyya, R. (2017). A survey on load balancing algorithms for virtual machines placement in cloud computing. Concurrency and

Computation: Practice and Experience, 29(12), e4123. <https://doi.org/10.1002/cpe.4123>

Xu, P., He, G., Li, Z., & Zhang, Z. (2018). An efficient load balancing algorithm for virtual machine allocation based on ant colony optimization. International Journal of Distributed Sensor Networks, 14(12).
<https://doi.org/10.1177/1550147718793799>

Ye, J., & Zhang, L. (2018). The load balancing ant colony optimization based on cloud computing. Proceedings of the 2018 International Conference on Network, Communication, Computer Engineering, 953–958.
<https://doi.org/10.2991/ncce-18.2018.160>

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.
<https://doi.org/10.1007/s13174-010-0007-6>