J Inf Process Syst, Vol.16, No.1, pp.132~144, February 2020
https://doi.org/10.3745/JIPS.01.0050

ISSN 1976-913X (Print)
ISSN 2092-805X (Electronic)

JOURNAL OF INFORMATION PROCESSING SYSTEMS JIPS

A Load-Balancing Approach Using an Improved Simulated Annealing Algorithm

Mohamed Hanine* and El-Habib Benlahmar*

Abstract

Cloud computing is an emerging technology based on the concept of enabling data access from anywhere, at any time, from any platform. The exponential growth of cloud users has resulted in the emergence of multiple issues, such as the workload imbalance between the virtual machines (VMs) of data centers in a cloud environment greatly impacting its overall performance. Our axis of research is the load balancing of a data center's VMs. It aims at reducing the degree of a load's imbalance between those VMs so that a better resource utilization will be provided, thus ensuring a greater quality of service. Our article focuses on two phases to balance the workload between the VMs. The first step will be the determination of the threshold of each VM before it can be considered overloaded. The second step will be a task allocation to the VMs by relying on an improved and faster version of the meta-heuristic "simulated annealing (SA)". We mainly focused on the acceptance probability of the SA, as, by modifying the content of the acceptance probability, we could ensure that the SA was able to offer a smart task distribution between the VMs in fewer loops than a classical usage of the SA.

Keywords

Cloud Computing, Load Balancing, Quality of Service, Simulated Annealing, Virtual Machine, Workload

1. Introduction

Cloud computing is an emerging technology that, soon after its introduction, has become a pillar of the Internet of Everything (IoE) due to the fast advancement of communication technology. It provides a flexible and cost-effective solution for many services through the Internet. Cloud computing offers multiple services to the user [1-8] and resulted in an exponential user growth that exposed a weak cloud resources management. These resources are in the virtual form, which is the most important characteristic of the cloud system.

Subsisting to the user's needs is very complex and can impact the overall performance of the cloud. Therefore, load balancing was given more attention by multiple researchers. Some load-balancing technics were proposed, but the fast growth of the cloud made them obsolete and no longer able to offer a good load-balancing solution.

This article suggests an improved usage of the meta-heuristic "simulated annealing (SA)" to ensure a better load balancing between a data center's virtual machines (VMs) in order to ensure a greater quality

Manuscript received July 5, 2017; first revision July 24, 2018; second revision October 2, 2018; third revision November 27, 2018; accepted December 2, 2018.

Corresponding Author: Mohamed Hanine (mohamedhanine24@gmail.com)

* Laboratory of Information Technology and Modeling, Faculty of Sciences Ben M'sik, Hassan II University Mohammedia, Casablanca, Morocco ({mohamedhanine24, h.benlahmer}@gmail.com)

of service (QoS) and to maximize the use of the provided resources.

We structured our article as follows. In Section 2, we will briefly discuss the previous load-balancing approaches. We will then present our approach in Section 3. Afterward, we will compare the results of our approach, after being implemented into the CloudSim simulator, with other load balancers in Section 4. Finally, in Section 5, we will conclude our findings.

2. State of the Art

Multiple load-balancing approaches were proposed in [9,10]. We will briefly present some of them while exposing their flaws which rendered them unable to balance the workload between VMs. Then, we will proceed by presenting some meta-heuristic approaches in a way that explains our meta-heuristic choice.

2.1 Load Balancers

Load balancers are used to improve task distribution between the VMs. In this section, we will briefly present some load-balancing approaches that were proposed in [10].

2.1.1 General algorithm-based category

Also known as the classical category, this category contains all the classical algorithms, such as round robin (RR) [11], weighted round robin [10], least-connection [12], and weighted least-connection [13]. It also includes load-balancing algorithms that do not need the cloud's architecture to operate. Some of these algorithms include trust management [14], cloud-friendly load balancing [15], two-phase load balancing [16], and stochastic hill climbing [17].

We will now proceed by explaining the algorithms stated above.

Round robin: RR is a process-scheduling algorithm based on the first come first served (FCFS) algorithm [18] that cyclically execute a process from the queue in a defined time limit. Overall, it is a good algorithm, but it does not have control over the workload distribution.

Weighted round robin: Weighted RR is a scheduling algorithm that adds weight to nodes depending on their CPU capacity. It helps the algorithm by allocating more requests to the server with high capacity. Similar to RR, weighted RR assigns requests to the node cyclically. The only difference is that nodes with higher specifications are given more requests.

Least-connection: This algorithm classifies the nodes by taking into consideration the number of connections between each node. The node with fewer connections will be given new requests. This method is the default method because it provides the best performance.

Weighted least-connection: Like least-connection, this algorithm calculates the connections of each node. Then, it adds the weight parameter to each node depending on its hardware. Weighted least-connection attributes new workloads to servers based on both the node's connections and its weight.

Trust management: The main idea behind this approach is to propose a VM Manager with adequate characteristics, which will improve resource utilization and decrease the system's response time. This approach supports both the load balancing between VMs and VMs' migration. It also offers a better

QoS for cloud computing. In this research, however, only a theoretical approach was proposed.

Cloud-friendly load balancing: This approach aims at decreasing the execution time of requests by migrating an object from a virtualized environment with interfering jobs and will also reduce the energy consumption to a lower value. However, this approach causes bottlenecking of the system due to the lack of fault tolerance.

Two-phase load balancing: This approach uses both the opportunistic load balancing and load balancing min-min scheduling algorithms in order to maintain the load balancing of the system while minimizing the execution time. No implementation of the proposed approach was provided.

Stochastic hill-climbing: This approach is based on the hill-climbing approach but with a random uphill move instead of the steepest. It offers better load-distribution management in cloud computing. This approach offers better performance in comparison with other algorithms. The only drawback of this approach is the lack of resource utilization, which is illustrated by the fact that this approach uses only one node.

2.1.2 Architectural-based category

Algorithms that require an architectural environment in order to operate, such as cloud partition load balancing [19], VM-based two-dimensional load management [20], DAIRS [21], and THOPSIS [22] are grouped in this category.

A brief presentation of each algorithm stated above will be provided in the following:

Cloud partition load balancing*:* A cloud partition is a model used to manage a large cloud. Basically, a cloud partition is a cloud infrastructure divided based on geographical locations. With the help of the main controller, the requests can be assigned to the nodes that are underloaded, thus improving the cloud's performances. This approach is still in the theoretical state.

VM-based two-dimensional load management*:* This approach uses restraints on both requests and nodes based on their metrics while scheduling in order to reduce the requests' migration. It provides a good task allocation. This algorithm lacks resources utilization.

DAIRS*:* In order to provide better task scheduling, this algorithm considers all of the nodes' metrics and the different queues in the system. This approach outperforms similar approaches, yet it is not suitable for deployment due to the fact that it is a centralized approach.

THOPSIS*:* This approach achieves better load balancing in a data center while reducing the VM migration rate by selecting which VM should migrate to a new physical machine and which physical machine should host it. By doing so, the physical machines are kept from being under- and over-loaded while using all of their parameters. Unfortunately, this approach provides a limited QoS to the SLA criteria.

2.1.3 Artificial intelligence-based category

This category contains all load balancers that use an artificial intelligence (AI) approach in order to provide a better load balancing. Some of these algorithms can be classified into the architecture-based category. We will proceed by briefly presenting some of the well-known AI-based algorithm, such as Bee-MMT [23] and ant colony optimization [24].

Bee-MMT: Bee-MMT is a variant of the bee colony algorithm that improves migration time while decreasing the power consumption by using VM migration. The only drawback of this algorithm is its

complexity, which makes it more difficult to implement in other systems.

Ant colony optimization: Developed after multiple studies on the behavior of ants while looking for food, this algorithm's main purpose is to find the shortest path. In other words, it is able to accurately determinate the state of each node. By doing so, it prevents having overloaded and underloaded nodes. The only drawback of this algorithm is its low throughput.

2.2 Meta-Heuristics Algorithms

Meta-heuristic algorithms are robust and known for being able to solve NP-hard and complex problems. Many searchers reached the conclusion that using a meta-heuristic approach instead of a classical optimization one helps to provide a better solution to the load balancing in cloud computing [25]. In the following text, we will briefly present some known meta-heuristics while also presenting and explaining the meta-heuristic used in our approach:

2.2.1 Tabu search

Tabu search is a metaheuristic approach that operates as follows. Initially, the algorithm will choose a random solution to the problem. Then, after a local scan to neighboring solutions, it chooses the best one as the optimal solution at a given time. Using a short-term memory formed by the previously visited solutions, it will avoid making loops on the same solutions by constantly checking the memory [26,27].

2.2.2 Genetic algorithm

The genetic algorithm is a meta-heuristic based on the evolution principle. It aims at finding the global solution of an optimization problem. Initially, the algorithm randomizes a solution that converges to the global minima after some evaluations with the objective function [28].

2.2.3 Bat algorithm

Based on a bat's sense of echolocation, the Bat algorithm aims at reducing the iterations while looking for the best solution of a given problem and trying to provide the advantages of other algorithms [29].

2.2.4 Simulated annealing

Based on the physical annealing process where a material is processed in order to achieve a uniform structure, Simulated annealing solves complex optimization problems. The main advantage that this algorithm has over other meta-heuristics is its large error margin. In other words, SA is known for finding the global solution of a given NP-hard problem [30]. This margin error is defined as the acceptance probability, P, which allows SA to avoid local solutions. The acceptance probability is defined as follows:

$$P = e^{-\Delta E/T}, \tag{1}$$

where ΔE defines the energy variation of the material at different time-lapses and T is the temperature. Initially, T has a high value and will slowly decrease in every iteration. SA accepts poor solutions in order to find the global solution [31]. Below, the pseudo-code of SA will be presented [32]:

Initialize the system configuration;
Initialize s as the initial state;

Initialize T with a large value;
Initialize N the number of iterations;
Initialize t the temperature change counter with 0;
Repeat
Initialize *n* the repetition counter with 0;
Repeat:
Generate s' = s + Δs a neighbor state of s;
Evaluate ΔE = E(s) – E(s'):
Generate $P = e^{-\Delta E/T}$ the acceptance probability of s;
Generate R = random (0,1) the acceptance probability of s';
if ΔE(s) < 0
  keep s as the actual state;
  else if R > P
s' is the new state;
Increment n by 1 until reaching N;
Increment t by 1 until reaching T;
Stop;

As presented above, the main criteria that made us choose SA in order to develop our approach are its ability to find global optima. It is also easily implemented, flexible, and is able to solve nonlinear problems efficiently.

The only issue with the classical SA approach is the huge amount of iterations required to reach the best solution. In the following section, we will explain the part that we added to the SA to make it faster while keeping its capacity to reach the global solution.

3. Contribution

Multiple load balancer studies lead us to the following conclusion: QoS management is still lacking due to the fact that the metrics of tasks and VMs are not fully used while managing the workload. Different approaches were proposed to provide better load balancing while taking into consideration both the state of a VM and the task's parameters [33-36]. Following these approaches, we propose our load balancer based on several parameters which influence the normal unfolding of the burden-sharing and the allowance of the tasks between the VMs of a data center.

We chose to collaborate with the research carried out by [37]. This choice has as the main aim of the inclusion of the VMs' characteristics as input parameters—in addition to the tasks'—to provide a better task allocation to those VMs.

This approach can be described as follows:

Each VM possesses a metric dependent on the number of cores it has, called MIPS [37], which can be defined as "million instructions per second".

In order for a task to be executed, some instructions need to be computed that are also known as the task's length (TL) [37].

We define the strip length, S, defined in (2) below, of each VM as the optimal number of tasks it can support before being classified as overloaded. It is illustrated by the following equation:

$$S = MIPS_i * (\text{number of tasks in the queue}) / \sum iMIPS \quad (2)$$

Even if we estimated the strip length of each VM at a given time, the tasks' allocation is still not optimized [36].

Thanks to the strip length, we are now one step closer from providing a satisfactory load balancer.

In the next step, we will introduce the meta-heuristic SA to provide better tasks' allocation. The modifications that we are proposing are as follows:

- Initialize E(i) from (1) as $E(i)= MIPS_i-TL_j$, where $MIPS_i$ is the MIPS value of the VM_i and TL_j is the length of the task j
- When E(i) >0, the task j will be given to VM_i
- When E(i) <0, we will let P, the acceptance probability of the VM_i, determine whether the task will be given to the VM_i

We will now proceed by detailing the implementation of the acceptance probability P to the approach proposed in [34].

When $MIPS_i-TL_j<0$, we will proceed by calculating $MIPS_i-TL_{j+1}$. That way we will have the following:

$$\Delta E = TL(j+1) - TL(j) \quad (3)$$

Knowing that the only drawback of the SA is the great number of iterations that has to be set initially, we will consider T_s, the sum of the tasks in the queue, which will be the number of iterations required. That way, our approach will be faster than the normal SA approach, while staying accurate by taking into consideration the strip length of each VM.

The modified acceptance probability is as follows:

$$P = \exp[-\Delta E/Ts] \quad (4)$$

Afterward, we will compare P to R, a random value that will reset its value at each iteration:

If P>R → VM_i will take the task j.

If P<R → VM_i will take the task j+1.

The following flowchart (Fig. 1) and algorithm summarize how our approach works.

MIPS-based SA

```
Initialize the system configuration;
Initialize the initial state of the VMs;
Initialize T the number of tasks in the queue;
For (v:0→the list of the VMs)
        Get the MIPS of VM(v);
        Calculate S= MIPS(v)*T/∑v MIPS
        Set I=0;

        For (c:0→the list of the Tasks)
                Get the length of the Task(c)
```

If (MIPS(v)>=TL(c))
VM(v)←Task(c)
T--;
I++;
Else
For (b:c+1→the list of the Tasks)
Calculate P= exp(-(TL(c)-TL(b))/T)
Generate R a random value
If (P>R)
VM(v)←Task(c)
T--;
I++;
Repeat until T=0 and I=S

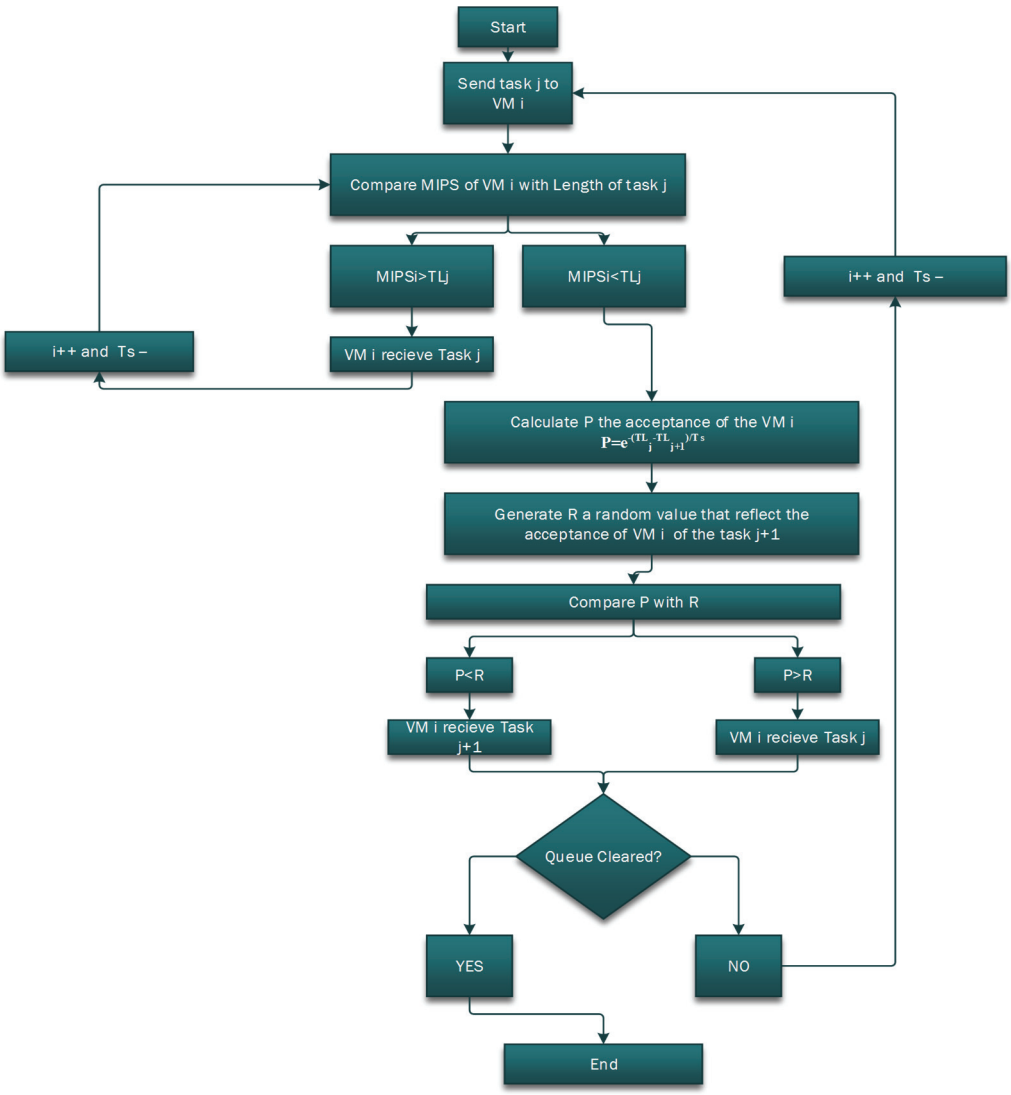


Fig. 1. Flowchart of our approach.

4. Experiments and Results

4.1 Experiments

In order to obtain concrete results for our proposed algorithm, we need a platform that simulates a cloud-based environment. We chose the CloudSim simulator due to the fact that it gives us full control of our datacenter's and tasks' metrics.

In the initialization stade, we used 1 physical machine, 2 VMs, and 10 tasks. Table 1 contains the details of this configuration, while Table 2 contains the tasks' lengths.

The MIPS value of each VM is randomly chosen from one to nine. In this scenario, VM 0 has six MIPS, and VM 1 has three MIPS.

The results of this simulation are explained in the following section.

Table 1. Cloud setup configuration details

Entry	Value
Data center	1
Number of HOSTS	1
CPU's cores number	10
CPU processing capacity	10 MIPS
Physical RAM	2048 MB
Number of VMs	2
CPU cores of each VMs	6-3
VM's MIPS	6-3 MIPS
Virtual RAM	512 MB
Manager	Xen

Table 2. Tasks' lengths

Task	TL
T0	5
T1	7
T2	4
T3	6
T4	2
T5	3
T6	1
T7	7
T8	2
T9	2

4.2 Results

As we implemented our algorithm on the CloudSim simulator, some classical algorithms were also implemented in order to provide a concrete comparison in terms of performance. The same scenario of having 10 tasks, 1 server, and 2 VMs was imposed on all the other algorithms.

At first sight, our approach provided a better task distribution than the other approaches (Fig. 2). The allocation can be proven by calculating the strip length of each VM:

- First VM:

  S(VM0) =6*10/9≈7

- Second VM:

  S(VM1) =3*10/9≈3

As demonstrated, VM 0 will process seven tasks while VM 1 will process three tasks.

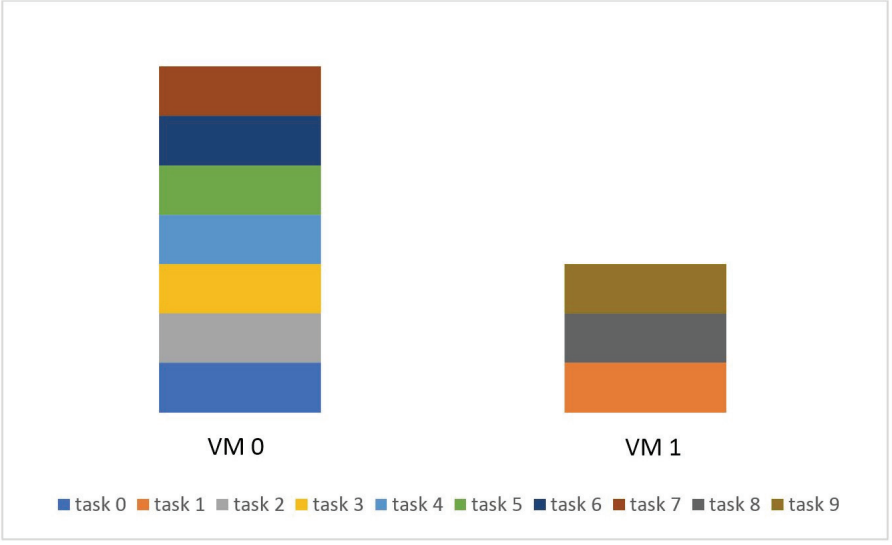


Fig. 2. Task allocation.

The second comparison performed between all approaches (Fig. 3, Table 3) is the execution time of each task for different algorithms. We chose FCFS, RR, and the classical implementation of the SA algorithm to compare with our approach. As shown in Fig. 3, our approach provides a faster process time which is greatly dependent on the tasks' allocation. While proceeding in the comparison, we noticed that some tasks have a higher length than the MIPS of the VMs processing them (Table 2). This allocation is supported due to the acceptance probability P and by comparing it with R, we can explain why the task was given to the VMs:

Task 1 allocation to VM 1:

P=exp [-(6-3)/9] = 0.72

R= 0.895

P<R, which means that VM1 process Task 1

The same thing is noticed for Task 7 and VM0:

P=exp [-(6-3)/3] = 0.37

R=0.318

P>R, this explains why VM0 added Task 7 to its workload

From the results of the previous experiments (Figs. 2, 3, and Table 3), we can declare that our approach offers an improved task distribution. This allows us to avoid having overloaded and underloaded VMs.

Furthermore, our approach provides faster load balancing, which can be seen in the overall process time of our approach being shorter than the processing time of the other load balancers (Table 3).

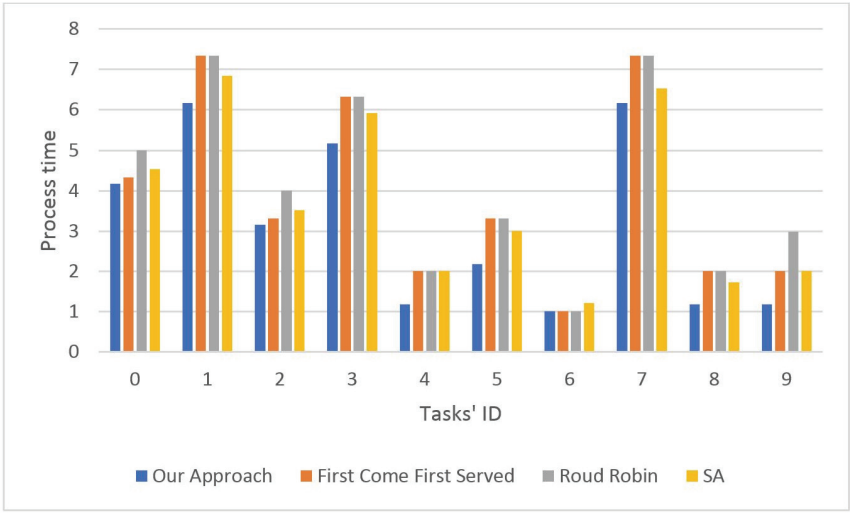


Fig. 3. Process time of all tasks.

Table 3. Load balancers process time comparison

Task	Our approach	FCFS	Classical SA	RR
0	4.67	4.33	4.53	5
1	6.67	7.33	6.84	7.33
2	3.67	3.33	3.52	4
3	5.67	6.33	5.92	6.33
4	2	2	2	2
5	2.67	3.33	3.01	3.33
6	1.17	1	1.21	1
7	6.33	7.33	6.53	7.33
8	1.17	2	1.72	2
9	2	2	2	3

5. Conclusions and Future Works

Cloud computing is a recent and rapidly evolving environments, which leads to many flaws and issues regarding QoS management in cloud computing. While aiming at providing a better solution, we proposed an improved version of the simulated annealing meta-heuristic in order to provide the appropriate task distribution to the VMs. Our approach, firstly, estimated the optimal workload of each VM provides a better task allocation. Then, by taking into consideration the tasks' parameters and the VM's strip length, the tasks are distributed to the VMs that can process them. This prevents VMs from being overloaded or underloaded. Additionally, if the length of the task is greater than the VM's MIPS, our approach introduces the acceptance probability of the SA, which has high fault tolerance. This will allow for better task allocation.

As for future works, we plan on (1) ameliorating this approach by adding more metrics and conditions to further improve the load's distribution and (2) expanding the experiments so that it can reflect realistic cloud environment usage.

References

[1] G. Gaspard, R. Jachniewicz, J. Lacava, and V. Meslard, "Equilibrage de Charge et Haute Disponibilite," 2009; http://generation-linux.elob.fr/dl/haute_dispo_rapport.pdf.

[2] S. Nepal, S. Chen, J. Yao, and D. Thilakanathan, "DIaaS: data integrity as a service in the cloud," in *Proceedings of 2011 IEEE 4th International Conference on Cloud Computing*, Washington, DC, 2011, pp. 308-315.

[3] C. Curino, E. P. Jones, R. A. Popa, N. Malviya, E. Wu, S. Madden, H. Balakrishnan, and N. Zeldovich, "Relational cloud: a database-as-a-service for the cloud," in *Proceedings of the 5th Biennial Conference on Innovative Data Systems Research*, Asilomar, CA, 2011.

[4] S. Frenot and J. Ponge, "LogOS: an automatic logging framework for service-oriented architectures," in *2012 38th Euromicro Conference on Software Engineering and Advanced Applications*, Cesme, Turkey, 2012, pp. 224-227.

[5] R. Hammad and C. S. Wu, "Provenance as a service: a data-centric approach for real-time monitoring," in *Proceedings of 2014 IEEE International Congress on Big Data*, Anchorage, AK, 2014, pp. 258-265.

[6] H. Al-Aqrabi, L. Liu, J. Xu, R. Hill, N. Antonopoulos, and Y. Zhan, "Investigation of IT security and compliance challenges in security-as-a-service for cloud computing," in *Proceedings of 2012 IEEE 15th International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing Workshops*, Shenzhen, China, 2012, pp. 124-129.

[7] Z. Zheng, J. Zhu, and M. R. Lyu, "Service-generated big data and big data-as-a-service: an overview," in *Proceedings of 2013 IEEE International Congress on Big Data*, Santa Clara, CA, 2013, pp. 403-410.

[8] B. Calder, J. Wang, A. Ogus, N. Nilakantan, A. Skjolsvold, S. McKelvie, et al., "Windows Azure Storage: a highly available cloud storage service with strong consistency," in *Proceedings of the 23rd ACM Symposium on Operating Systems Principles*, Cascais, Portugal, 2011, pp. 143-157.

[9] S. Sharma, S. Singh, and M. Sharma, "Performance analysis of load balancing algorithms," *World Academy of Science, Engineering and Technology*, vol. 38, no. 3, pp. 269-272, 2008.

[10] M. Mesbahi and A. M. Rahmani, "Load balancing in cloud computing: a state of the art survey," *International Journal of Modern Education and Computer Science*, vol. 8, no. 3, pp. 64-78, 2016.

[11] A. Aditya, U. Chatterjee, and S. Gupta, "A comparative study of different static and dynamic load balancing algorithm in cloud computing with special emphasis on time factor," *International Journal of Current Engineering and Technology*, vol. 5, no. 3, pp. 1898-1907, 2015.

[12] J. Vashistha and A. K. Jayswal, "Comparative study of load balancing algorithms," *IOSR Journal of Engineering*, vol. 3, no. 3, pp. 45-50, 2013.

[13] R. Lee and B. Jeng, "Load-balancing tactics in cloud," in *Proceedings of 2011 International Conference on Cyber-Enabled Distributed Computing and Knowledge Discovery*, Beijing, China, 2011, pp. 447-454.

[14] P. Gupta, M. K. Goyal, and P. Kumar, "Trust and reliability based load balancing algorithm for cloud IaaS," in *Proceedings of the 3rd IEEE International Advance Computing Conference (IACC)*, Ghaziabad, India, 2013, pp. 65-69.

[15] O. Sarood, A. Gupta, and L. V. Kale, "Cloud friendly load balancing for HPC applications: preliminary work," in *Proceedings of the 41st International Conference on Parallel Processing Workshops*, Pittsburgh, PA, 2012, pp. 200-205.

[16] S. C. Wang, K. Q. Yan, W. P. Liao, and S. S. Wang, "Towards a load balancing in a three-level cloud computing network," in *Proceedings of the 3rd International Conference on Computer Science and Information Technology*, Chengdu, China, 2010, pp. 108-113.

[17] B. Mondal, K. Dasgupta, and P. Dutta, "Load balancing in cloud computing using stochastic hill climbing-a soft computing approach," *Procedia Technology*, vol. 4, pp. 783-789, 2012.

[18] S. Stattelmann and F. Martin, "On the use of context information for precise measurement-based execution time estimation," in *Proceedings of the 10th International Workshop on Worst-Case Execution Time Analysis (WCET)*, Brussels, Belgium, 2010, pp. 64-76.

[19] G. Xu, J. Pang, and X. Fu, "A load balancing model based on cloud partitioning for the public cloud," *Tsinghua Science and Technology*, vol. 18, no. 1, pp. 34-39, 2013.

[20] R. Wang, W. Le, and X. Zhang, "Design and implementation of an efficient load-balancing method for virtual machine cluster based on cloud service," in *Proceedings of the 4th IET International Conference on Wireless, Mobile & Multimedia Networks (ICWMMN)*, Beijing, China, 2011, pp. 312-324.

[21] W. Tian, Y. Zhao, Y. Zhong, M. Xu, and C. Jing, "A dynamic and integrated load-balancing scheduling algorithm for Cloud datacenters," in *Proceedings of 2011 IEEE International Conference on Cloud Computing and Intelligence Systems*, Beijing, China, 2011, pp. 311-315.

[22] F. Ma, F. Liu, and Z. Liu, "Distributed load balancing allocation of virtual machine in cloud data center," in *Proceedings of 2012 IEEE International Conference on Computer Science and Automation Engineering*, Beijing, China, 2012, pp. 20-23.

[23] S. M. Ghafari, M. Fazeli, A. Patooghy, and L. Rikhtechi, "Bee-MMT: a load balancing method for power consumption management in cloud computing," in *Proceedings of the 6th International Conference on Contemporary Computing (IC3)*, Noida, India, 2013, pp. 76-80.

[24] C. K. Teoh, A. Wibowo, and M. S. Ngadiman, "Review of state of the art for metaheuristic techniques in Academic Scheduling Problems," *Artificial Intelligence Review*, vol. 44, no. 1, pp. 1-21, 2015.

[25] K. Nishant, P. Sharma, V. Krishna, C. Gupta, K. P. Singh, and R. Rastogi, "Load balancing of nodes in cloud using ant colony optimization," in *Proceedings of 2012 UKSim 14th International Conference on Computer Modelling and Simulation*, Cambridge, UK, 2012, pp. 3-8.

[26] E. Ikonomovska, I. Chorbev, D. Gjorgjevik, and D. Mihajlov, "The adaptive tabu search and its application to the quadratic assignment problem," in *Proceedings of the 9th International Multi-conference Information Society (IS 2006)*, Ljubljana, Slovenia, 2006, pp. 26-29.

[27] G. A. E. N. Said, A. M. Mahmoud, and E. S. M. El-Horbaty, "A comparative study of meta-heuristic algorithms for solving quadratic assignment problem," *International Journal of Advanced Computer Science and Applications*, vol. 5, no. 1, 2014.

[28] F. Neumann and C. Witt, *Bioinspired Computation in Combinatorial Optimization: Algorithms and Their Computational Complexity*. Heidelberg: Springer, 2010.

[29] X. S. Yang, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization (NICSO 2010)*. Heidelberg: Springer, 2010, pp. 65-74.

[30] E. H. L. Aarts and P. J. M. van Laarhoven, "Simulated annealing: a pedestrian review of the theory and some applications," in *Pattern Recognition Theory and Applications*. Heidelberg: Springer, pp. 179-192.

[31] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671-680, 1983.

[32] K. L. Du and M. N. S. Swamy, "Simulated annealing," in *Search and Optimization by Metaheuristics*. Cham: Birkhauser, 2016. pp. 29-36

[33] Y. Fahim, E. B. Lahmar, E. H. Labriji, A. Eddaoui, and S. Ouahabi, "The load balancing improvement of a data center by a hybrid algorithm in cloud computing," in *Proceedings of the 3rd IEEE International Colloquium in Information Science and Technology (CIST)*, Tetouan, Morocco, 2014, pp. 141-144.

[34] Y. Fahim, E. B. Lahmar, E. H. Labriji, and A. Eddaoui, "The tasks allocation based on the pre-estimation of the processing time in the cloud environment," *Journal of Theoretical and Applied Information Technology*, vol. 75, no. 3, pp. 350-355, 2015.

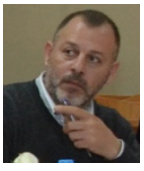
[35] Y. Fahim, E. B. Lahmar, E. H. Labriji, and A. Eddaoui, "The load balancing based on the estimated finish time of tasks in cloud computing," in *Proceedings of the 2nd World Conference on Complex Systems (WCCS)*, Agadir, Morocco, 2014, pp. 594-598.

[36] Y. Fahim, H. Rahhali, M. Hanine, E. H. Benlahmar, E. H. Labriji, M. Hanoune, and A. Eddaoui, "Load balancing in cloud computing using meta-heuristic algorithm," *Journal of Information Processing Systems*, vol. 14, no. 3, pp. 569-589, 2018.

[37] S. Roy, S. Banerjee, K. R. Chowdhury, and U. Biswas, "Development and analysis of a three phase cloudlet allocation algorithm," *Journal of King Saud University-Computer and Information Sciences*, vol. 29, no. 4, pp. 473-483, 2017.

Mohamed Hanine **https://orcid.org/0000-0001-5417-3503**

He received an engineering degree in Telecommunications and Networks at University Abdel Malek Essaidi, Tetouan, Morocco, in 2015. He is currently pursuing a Ph.D. degree in Information Modeling and Technologies at the University of HASSAN II, Casablanca, Morocco. His research interests include information modeling, cloud computing, load balancing, and beyond.



El-Habib Benlahmar **https://orcid.org/0000-0001-7098-4621**

He is a professor in the Department of Mathematics and Computer Science at Faculty of Science Ben M'Sik, Hassan II University, where he has been since 2008. Since 2016, he has been a coordinator of the Master Data Science & Big Data. From 2014 to 2018 he served as Team Leader: Semantic Web and Knowledge Extraction at the TIM laboratory. He received his habilitation to supervise research in 2014 and is currently an associate professor in Computer Science at Faculty of Science Ben M'Sik Casablanca. His research interests include metasearch, information retrieval, semantic web, automatic processing of natural language, cloud computing, and beyond.