An Optimized Flexi Max-Min Scheduling Algorithm for Efficient Load Balancing on a Cloud

Fale Mantim Innocent, Sitlong Nengak Iliya, Ramson Emmanuel Nannim, Datti Emmanuel Useni, Jakawa Jimmy Nerat

Abstract— Cloud Computing is a popular and cost-effective computing platform for hosting applications and delivering services over the internet. Its robustness assumes many forms which include: Infrastructure as a Service (IaaS); Platform as a Service (PaaS); Software as a Service (SaaS); Database as a Service (DaaS), Network as a Service (NaaS); and so on. Load Balancing is required to distribute tasks across Virtual Machines on one hand; and on the other hand, efficiently place Virtual Machines on physical servers such that resource and energy consumption is minimized. This paper proposes a new task scheduling algorithm: 'Optimized Flexi Max-Min Scheduling Algorithm'. This algorithm maintains a data structure which is modeled after a Binary Search Tree for estimations, enhanced searching, task allocation, and migration of tasks. CloudSim was used to model and simulate the cloud computing environment in order to obtain simulated data. Microsoft Visual Studio 2017's C#.NET was used to implement the Round Robin, traditional Max-Min, and the proposed task scheduling algorithms. The result of this experiment shows that the proposed algorithm outperforms Round Robin and the traditional Max-Min task scheduling algorithms.

Index Terms— Algorithms, Cloud Computing, CloudSim, Dynamic Scheduling, Load Balancing, Max-Min, Task Scheduling

1 Introduction

THE Cloud is a collection of resources shared with dynamism over a network [1] [2]. Numerous technologies make cloud computing possible. Of them all, virtualization is the most important. Virtualization [3] [4] provides a favorable approach through which resources on one or more physical servers can be shared through incomplete or complete machine solution. This entails hardware and software partitioning into multiple execution environments in which each partition can act as a complete system. Through virtualization, numerous applications can run on different performance-isolated platforms called Virtual Machines (VMs) that are all placed on one physical machine (PM) [5] [6]. Virtualization helps cloud providers to ensure that Quality of Service (QoS) [7] [8] is conveyed to users while the achievement of optimum server utilization and minimized energy consumption is possible. Virtualization provides redundancy via real-time data synchronization [9], [10] between datacenters in order to prevent data loss in case of system crash or other unplanned issues.

Cloud providers intend to maximize profit by minimizing operational cost. Power consumption [11] [12] primarily dominates the operational cost of cloud datacenters. The other issues that can be attributed to power management is carbon dioxide emission and system reliability. Power management is a load balancing issue even though other approaches, like green computing, exist for addressing this problem. Load balancing [13] [14] is a method used to distribute workload on numerous computers or a computer constellation via network links to achieve optimum resource utilization which maximizes throughput and curtails overall response time. This technique minimizes total waiting time as well as avoids too much overload on resources by distributing traffic among PMs so that data can be sent and received with minimum delay.

This research proposes an algorithm which is modeled after the traditional Max-Min task scheduling algorithm but optimizes key features which are identified as not being efficient. This claim shall be elaborated in SECTION 2.

Section 2 of this paper discusses 'RELATED WORK' while section 3 discusses 'SYSTEM MODEL'. In section 4, the proposed 'OPTIMIZED FLEXI MAX-MIN SCHEDULING ALGORITHM' is discussed. Section 5 presents 'EXPERIMENTS AND EVALUATION' while section 6 covers 'CONCLUSION'. Section 7 is 'REFERENCES'.

2 Related Work

Generally, load balancing or task scheduling algorithms are categorized into: static; and dynamic [15]. The static algorithms include: Round Robin (RR); and Opportunistic Load Balancing (OLB) algorithms. RR allocates tasks in turns while OLB takes into account a user's priority and its bandwidth demand. The dynamic ones include: Minimum Execution Time (MET); Minimum Completion Time (MCT); Min-Min; and Max-Min algorithms. The latter category aims at optimizing resource utilization in consideration of task execution time and expected completion time. Max-Min and Min-Min are specifically for tasks arriving in batches while MET and MCT are mainly for single task allocation [16]. Hybrid algorithms [17] [18] are possible as there are emerging load balancing algorithms. This category exploits the advantages of both static and dynamic algorithms. Load balancing algorithms tend to address two major issues: task scheduling across virtual ma-

- *Fale Mantim Innocent is with Federal College of Education, Pankshin, Plateau State, Nigeria. Email: thefmicorporation@gmail.com*
- *Sitlong Nengak Iliya is with Federal College of Education, Pankshin, Plateau State, Nigeria. Email: iliya_sitlong@yaoo.com*
- *Ramson Emmanuel Nannim is with Federal College of Education, Pankshin, Plateau State, Nigeria. Email: ramsonn78@gmail.com*
- *Datti Emmanuel Useni is with Federal College of Education, Pankshin, Plateau State, Nigeria. Email: duedatti@yahoo.com*
- *Jakawa Jimmy Nerat is a M.Sc. Computer Science student with Abubakar Tafawa Balewa University, Bauchi, Nigeria. Email: jimmycartty@gmail.com*

chines; and VM placement [19]. A handful of load balancing algorithms exist but only a few (emerging algorithms) shall be reviewed in the following paragraphs.

The following paragraphs will present a brief overview of some reputable algorithms proposed in their respective research articles. This is to further expose approaches that have been used for load balancing in the cloud environment.

In 'Analysis of variants in Round Robin Algorithms for load balancing in Cloud Computing', the Round Robin algorithm was discussed. It allocates tasks to virtual machines in turns [20] not minding the execution time of tasks thus leading to load imbalance in most cases. As tasks arrive, it simply assigns them to virtual machines in a circular fashion until the tasks are exhausted.

The 'Max-Min Task Scheduling Algorithm for Load Balance in Cloud Computing - Elastic Cloud Max-Min (ECMM)', a close variant of Max-Min, is a dynamic load balancing algorithm for jobs arriving in batches. This algorithm maintains a task status table to estimate the real-time load of virtual machines and the expected completion time of tasks, which can allocate the workload among nodes and realize the load balance [21]. A virtual machine status table, which comes from the task status statistics, is also maintained. This algorithm is based on the traditional Max-Min algorithm but varies slightly by the introduction of an update algorithm whose primary function is to remove completed tasks and update completion time of tasks that are still being executed. In fact, the introduction of the update algorithm is what gives it its elastic feature. A typical scenario for ECMM is a datacenter housing several virtual machines. Each time jobs arrive in a batch, they are sorted in descending order of their execution time. The task with the maximum execution time is then allocated to the virtual machine with the least (minimum) load. Each time a virtual machine completes the execution of a task, the estimated completion time for pending tasks is recomputed using the update algorithm.

The 'Enhanced Bee Colony for Efficient Load Balancing and Scheduling in Cloud' is a dynamic load balancing algorithm. Its typical scenario and assumption is that there are many datacenters housing several virtual machines. This algorithm is constituted by two sub-algorithms: the task scheduling algorithm; and the migration algorithm [22]. When tasks arrive, the load on every datacenter is computed by the Cloud Information System (CIS) and fed to the Load Balancer. A threshold value (between 0 and 1) is computed and it is used as a benchmark for categorizing datacenters into overloaded and under-loaded categories. If the threshold value of a datacenter is less than 0.5, such a datacenter is under-loaded otherwise, it is said to be overloaded. Tasks are then allocated to virtual machines in under-loaded datacenters using the task scheduling algorithm in real-time. After allocation of tasks, virtual machines in these datacenters execute these tasks. As they do so, some complete execution before others leaving them idle and the others overloaded. The migration algorithm is then used to balance the load on datacenters by moving some tasks from overloaded virtual machines to under-loaded ones. This also is done in real-time.

'A PSO-Based Algorithm for Load Balancing in Virtual Machines of Cloud Computing Environment' is a hybrid load balancing algorithm aimed at addressing NP-hard combinational optimization problem to establish the mappings between jobs submitted by user terminals and dynamical resources encapsulated in virtual machines [23]. The primary concern addressed by this algorithm is that if only execution time is taken into consideration when scheduling cloud resources, a serious load imbalance problem may occur between virtual machines (VMs) in the cloud computing environment. Hence, task execution time is optimized in view of both the task running time and the system resource utilization. This algorithm is an improvement on the standard Particle Swarm Optimization (PSO) algorithm. It introduced a simple mutation mechanism and a self-adapting inertia weight method by classifying fitness values. The typical scenario or assumption of this model is that independent and interrelated tasks can be submitted by terminal users in which case the interrelated jobs can be divided into small separate tasks that can run without interferences, so it just have to deal with how to balance the workload of VMs with independent tasks.

'An Optimized Flexi Max-Min Scheduling Algorithm for Efficient Load Balancing on a Cloud' is a dynamic load balancing algorithm modeled after the traditional Max-Min scheduling algorithm. It features major modifications in order to optimize and add adaptable features to make it flexible and efficient in varying scenarios. Even though most algorithms ignore recommendations for implementation such as data structures, problems with such algorithms are still detected or projected via simulation. The traditional Max-Min algorithm is for tasks arriving in the same batch [21] [24]. The following are problems identified with the traditional Max-Min algorithm:

- Load balancing is done during task allocation.
- It lacks mechanisms for dealing with uncompleted tasks as new tasks could arrive while some tasks are still being executed.
- It puts only execution time into consideration when scheduling tasks.
- The Max-Min algorithm maintains an unsorted list of tasks and virtual machines.
- Sorting is done in $O(n^2)$ time.
- Searching is done in O(n) time. It is primarily a blind search.

The aforementioned problems are handled by the Optimized Flexi Max-Min algorithm as follows:

- An update algorithm is introduced to handle uncompleted tasks.
- A task migration algorithm is also introduced to balance load across virtual machines in real-time.
- Jobs are scheduled either in batch or as single tasks.

- Other parameters like task running time and system resource utilization are put into consideration when scheduling tasks.
- A Binary Search Tree (BST) is used to maintain tasks and VMs. This BST is a self-balancing tree. The nodes of this tree represent VMs. Each node is a data structure capable of maintaining task entries for each VM. In order to achieve load balancing on VMs, tasks are migrated between nodes (VMs). Through this, the BST achieves self-balance.
- Due to the use of BSTs, no sorting is required as tasks are automatically allocated to the nodes on the tree (VMs) which are already sorted.
- Searching is informed/heuristic. Searching is done in O(*log*n) time.

3 System Model

Task Model

Definition 1 Each time a VM executes a task, the task does consume resources. This task resource consumption is in accordance with the availability of resources at its respective node (VM).

$$T_i = [Tcpu_i, Tmem_i, Tdisk_i, Tnet_i]; \quad (1)$$

Definition 2 Execution Time: For every task *i* executed on VM *j*, the time it will take to complete task *i* is referred to as its weight or execution time and it is represented as E_i^j.

Load Model

Definition 3 Node Load: The load at a given node is the summation of the weight of all tasks allocated to such a node. This can be represented as L_i^j.

$$L_i^j = \sum_{i=1}^{n} Ei^j; \quad (2)$$

Definition 4 Load Benchmark: This value is a computation of the average of Node Load L_i^j represented as benchmark(L)

$$\text{benchmark(L)} = \sum_{i=1,j=1}^{n,m} Li^j \ / \ \text{m}; \quad (3)$$

4 Optimized Flexi Max-Min Scheduling Algorithm

The Optimized Flexi Max-Min Scheduling algorithm maintains a BST-like data structure to organize VMs such that each node on the tree represent a VM. The nodes are capable of holding multiple task entries. Once tasks arrive, whether in batches or as singletons, the tasks are allocated to the nodes (VMs) of the tree by a task allocation algorithm. Obviously, some VMs will finish execution of tasks earlier than others leaving them idle or causing an imbalance on the VMs tree. An update algorithm is then used to estimate completion time for uncompleted tasks and delete completed tasks. A migration algorithm is used to possibly remove pending tasks from nodes (VMs) and reassign to others in order to balance the tree (load balancing). This process continues until all tasks on all nodes (VMs) have finished execution and deleted from nodes. This algorithm consists of three sub-algorithms: Task Allocation Algorithm; Update Algorithm; and Migration Algorithm.

4.1 Task Entry Structure And Virtual Machine Parameters

The Task Entry for every VM is a four tuple where *id* mean a unique task identification number, *vm_id* is a unique VM identification number, *exectime* refers to task execution time or weight, and *comptime* refers to estimated completion time for task.

$$T_{id} = \langle id, vm\_id, exectime, comptime \rangle; \quad (4)$$

Every VM also maintains certain parameters to help monitor its activities. The parameters are *vm_id* which means VM unique identification number, *load* refers to the total weight of tasks allocated to the VM, *finishtime* refers to the *timestamp* the VM will finish executing the tasks allocated to it, and *last* refers to the time the last task finished execution.

$$VM_{id} = \langle vm\_id, load, finishtime, last \rangle; \quad (5)$$

4.2 Task Allocation Algorithm

STEP 1: Insert all VMs on a BST. Each VM id serves as key to their respective nodes on the BST.
STEP 2: Identify VM with the least load on the BST and assign task to it. Repeat this process until all tasks have been allocated.
STEP 3: While VMs are computing tasks, do the following:

- Update tasks and VMs parameters.
- Migrate pending tasks from overloaded VMs to underloaded or idle ones to obtain load balance

STEP 4: Terminate algorithm when VMs have executed all tasks.

4.3 Update Algorithm

STEP 1: Delete all completed tasks from the BST nodes (VMs).
STEP 2: For all uncompleted tasks, traverse the BST in-order and do the following:

$$VM_{id}.load = \sum_{i=1}^{n} Ti^{id}.exectime; \quad (6)$$

(recall: $L_i^j = \sum_{i=1}^{n} Ei^j$)

$$VM_{id}.finishtime = timestamp + VM_{id}.load; \quad (7)$$

$$T_{id}.comptime = VM_{Tid.vm\_id}.last + T_{id}.exectime; \quad (8)$$

STEP 3: Terminate algorithm when VMs have executed all tasks.

4.4 Migration Algorithm

STEP 1: For all pending tasks (that is tasks awaiting execution), traverse the BST in-order and do the following:

- Compute average load on all nodes (VMs).
- Use the value as benchmark to determine overloaded and under-loaded nodes (VMs) on the BST (under-loaded < benchmark and overloaded >= benchmark).

  benchmark(L) = $\sum_{i=1,j=1}^{n,m} Li^j$ / m; (9)
- Remove pending tasks from overloaded nodes (VMs) and assign to under-loaded ones in order to obtain a balance.

STEP 2: Terminate algorithm when VMs have no pending tasks.

5 Experiments and Evaluation

5.1 Experimental Conditions for Run

CloudSim was used to model and simulate VMs, computing resources, and energy consumption in order to evaluate the efficiency of load balancing for the proposed Optimized Flexi Max-Min scheduling algorithm. This paper evaluates the performance of the algorithm using parameters such as Average Task Pending Time, Task Response Time Ratio, and VM resource utilization. This experiment is to verify the performance of the Optimized Flexi Max-Min Scheduling Algorithm against its counterparts. The control group in this experiment include Round Robin (RR), Max-Min (MM), and the proposed Optimized Flexi Max-Min (OFMM) algorithms in this respective order. In this experiment, 6 tasks were scheduled against 4 VMs (see *Fig. 1* and *Fig. 2*).

5.2 Discussion of Results

In terms of Average Task Pending Time, the Optimized Flexi Max-Min scheduling algorithm outperforms both Round Robin (3rd) and the traditional Max-Min (2nd) scheduling algorithms. For Average Task Response Time Ratio, the Optimized Flexi Max-Min scheduling algorithm also performed better than its counterparts with Round Robin (2nd) and the traditional Max-Min (3rd).

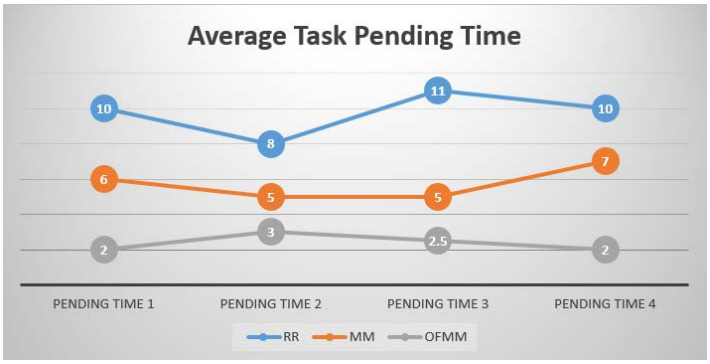

Fig. 1: Average Task Pending Time for RR, MM, and OFMM

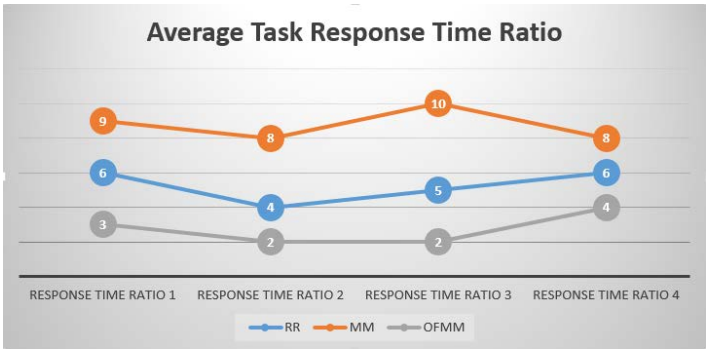

Fig. 2: Average Task Response Time Ratio for RR, MM, and OFMM

6 Conclusion

This paper introduced the Optimized Flexi Max-Min Scheduling algorithm. It shows how it varies from the traditional Max-Min task scheduling algorithm and the other algorithms reviewed in SECTION 2. This research also shows how a suitable data structure can be used to enhance a load balancing algorithm in a virtualized environment. This algorithm suits both tasks arriving in batches and single tasks. The results of the experiments show that the Optimized Flexi Max-Min Scheduling algorithm is competitive even though the experiments were conducted in a simulated cloud environment. In a real cloud environment, problems may occur due to bandwidth restrictions and task decomposition, hence, further research is recommended in this area.

References

[1] H. Alexa and C. James, "The Basics of Cloud Computing," Carnegie Mellon University, 2011.

[2] Torry Harris, "Cloud Computing - An Overview," [Online]. Available: http://www.thbs.com/downloads/Cloud-Computing-Overview.pdf. [Accessed 21 December 2017].

[3] G. Yongqiang, G. Haibing, Q. Zhengwei, H. Yang and L. Liang, "A multi-objective ant colony system algorithm for virtual machine placement in cloud computing," *Journal of Computer and System Sciences,* vol. 79, no. 8, pp. 1230-1242, 2013.

[4] K. C. N. M. Mosharaf and B. Raouf, "A survey of network virtualization," *Computer Networks,* vol. 54, no. 5, pp. 862-876, 2010.

[5] B. Paul, D. Boris, F. Keir, H. Steven, H. Tim, H. Alex, N. Rolf, P. Ian and W. Andrew, "Xen and the art of virtualization," in *SOSP '03 Proceedings of the nineteenth ACM symposium on Operating systems principles*, 2003.

[6] L. Flavio and D. P. Roberto, "Secure virtualization for cloud computing," *Journal of Network and Computer Applications,* vol. 34, no. 3, pp. 1113-1122, 2011.

[7] L. K. Ronald and D. V. Russel, Cloud Security: A Comprehensive Guide to Secure Cloud Computing, Wiley Publishing, 2010.

[8] J. Y and M. K, "Cloud computing - concepts, architecture and challenges," in *Computing, Electronics and2012 International Conference on Computing, Electronics and Electrical Technologies (ICCEET)*, Kumaracoil, India, 2012.

[9] A. Mohammad and H. Eui-Nam, "Fog Computing and Smart Gateway Based Communication for Cloud of Things," in *2014 International Conference on Future Internet of Things and Cloud*, Barcelona, 2014.

[10] G. Chunye, L. Jie and Z. Qiang, "The Characteristics of Cloud Computing," in *2010 39th International Conference on Parallel Processing Workshops*, San Diego, CA, USA, 2010.

[11] Z. Qi, C. Lu and B. Raouf, "Cloud computing: state-of-the-art and research challenges," *Journal of Internet Services and Applications,* vol. 1, no. 1, pp. 7-18, 2010.

[12] K. Dzmitry, B. Pascal and U. K. Samee, "GreenCloud: a packet-level simulator of energy-aware cloud computing data centers," *The Journal of Supercomputing,* vol. 62, no. 3, pp. 1263-1283, 2012.

[13] K. M. Nitin and M. Nishchol, "Load Balancing Techniques: Need, Objectives and Major Challenges in Cloud Computing - A Systematic Review," *International Journal of Computer Applications,* vol. 131, pp. 11-19, 2015.

[14] R. Martin, L. David and T.-B. A, "A Comparative Study into Distributed Load Balancing Algorithms for Cloud Computing," in *2010 IEEE 24th International Conference on Advanced Information Networking and Applications Workshops*, Perth, WA, Australia, 2010.

[15] W. Lee, J. S. Howard, P. R. Vwani and A. M. Anthony, "Task Matching and Scheduling in Heterogeneous Computing Environments Using a Genetic-Algorithm-Based Approach," *Journal of Parallel and Distributed Computing,* vol. 47, no. 1, pp. 8-22, 1997.

[16] S. Pinal, "A SURVEY OF VARIOUS SCHEDULING ALGORITHM IN CLOUD," *International Journal of Research in Engineering and Technology,* vol. 2, no. 2, pp. 131-135, 2013.

[17] M. M, M. N, K. Y, C. L. Y, G. T. E, Y. Z. A and T. D, "A parallel bi-objective hybrid metaheuristic for energy-aware scheduling for cloud computing systems," *Journal of Parallel and Distributed Computing,* vol. 71, no. 11, pp. 1497-1508, 2011.

[18] F. B. Luiz and R. M. M. Edmundo, "HCOC: a cost optimization algorithm for workflow scheduling in hybrid clouds," *Journal of Internet Services and Applications,* vol. 2, no. 3, pp. 207-227, 2011.

[19] T. M. Siva, S. R and Y. Lei, "Stochastic models of load balancing and scheduling in cloud computing clusters," in *2012 Proceedings IEEE INFOCOM*, Orlando, FL, USA, 2012.

[20] S. Pooja and M. Pramati, "Analysis of variants in Round Robin Algorithms," *International Journal of Computer Science and Information Technologies,* vol. 4, no. 3, pp. 416-419, 2013.

[21] M. Yinchi, C. Xi and L. Xiaofang, "Max-Min Task Scheduling Algorithm for Load Balance in Cloud Computing," in *Proceedings of International Conference on Computer Science and Information Technology, Advances in Intelligent Systems and Computing 225*, India, 2014.

[22] B. K. Remesh and P. Samuel, "Enhanced Bee Colony Algorithm for Efficient Load Balancing and Scheduling in Cloud," *Innovations in Bio-Inspired Computing and Applications,* vol. 424, pp. 67-78, 2016.

[23] L. Zhanghui and W. Xiaoli, "A PSO-Based Algorithm for Load Balancing in Virtual Machines of Cloud Computing Environment," in *International Conference in Swarm Intelligence*, Berlin, 2012.

[24] M. Gao and L. Hao, "An Improved Algorithm Based on Max-Min for Cloud Task Scheduling," *Recent Advances in Computer Science and Information Engineering,* vol. 125, pp. 217-223, 2012.

IJSER