Towards Optimizing Task Scheduling Process in Cloud Environment

Yong Shi, Kun Suo, Jameson Hodge, Divya Pramasani Mohandoss and Steven Kemp
College of Computing and Software Engineering
Kennesaw State University
Marietta, USA
Contact: yshi5@ kennesaw.edu

2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC) | 978-1-6654-1490-6/21/$31.00 ©2021 IEEE | DOI: 10.1109/CCWC51732.2021.9376146

***Abstract*—The cloud computing paradigm offers many advantages over traditional self-hosting computing solutions by abstracting computations from infrastructure. An essential task for any cloud provider is task scheduling, wherein tasks are assigned to computing resources within the cloud system by a broker. Numerous cloud task scheduling algorithms exist including Min-Min, Max-Min, and Sufferage—though each is defined by performance trade-offs. BSufferage is proposed as a novel cloud task scheduling algorithm that augments the performance of the classical Sufferage algorithm. Modeling the algorithm using the open source CloudSim package and comparing it to its precursor yields performance results for BSufferage—demonstrating decreased completion times, increased throughput, and improved load balancing of resources.**

Keywords— Task Scheduling, Cloud Computing, Sufferage Algorithm, Cloud Simulation

I. INTRODUCTION

Cloud computing is advantageous to traditional computing solution because the barrier of cost is reduced; users are only charged for their allocated resources. By abstracting from the computing resources, developers are unburdened with the purchasing, configuring, provisioning, and maintaining of computing resources so that development time can be utilized efficiently. Other advantages of cloud computing include integrations with other cloud services, high composability, and scalability benefits, as most providers offer integrated customization of computing power, storage, bandwidth, etc. The SPI model partitions the types of cloud providers into three categories, with each succeeding category becoming more abstract than its predecessor—on the bottom is Infrastructure as a Service (IaaS), on the middle is Platform as a Service (PaaS), and on top is Platform as a Service (PaaS). Many institutions and corporations utilize cloud computing to some degree, including but not limited to Google, Microsoft, Amazon, IBM, etc. Active research areas related to cloud computing include network topology, task scheduling, and distributed/parallel computing.

On both single-threaded processors and distributed or networked computing systems, tasks must be assigned to computing resources. An objective of any cloud provider is to distribute tasks among resources to maximize efficiency and reduce cost in a way that is scalable and flexible. Cloud providers execute tasks with both multitasking (running multiple tasks synchronously) and multiplexing (arranging multiple flows synchronously). The most effective task scheduling algorithms maximize throughput while minimizing completion time, though other factors that must be considered include load balancing and overall quality of service. Over the last few decades, researchers have designed many task scheduling algorithms including First-Come, First-Served (FCFS), Round Robin, Min-min, Max-min and Sufferage [1].

Each task scheduling algorithm is a policy that aims to minimize certain performance metrics while making trade-offs with other metrics. The most basic of these is the FCFS scheduling policy, where tasks are assigned to resources in the order they are received by the system. Round Robin (RR) is another policy in which each task is allocated an equal slice of time to share computing resources resulting in a large overhead and poor response time, though it is more commonly used for single-threaded systems. The Min-min policy allocates computing resources to the task with the potential of being completed the earliest, while Max-min does the same but for the task with the potential of being completed the latest. The Most Fit policy aims to maximize load balancing by assigning tasks to resources that best utilize them, though it suffers from the highest assignment failure ratio out of the listed algorithms. The Priority policy assigns every task a priority, with tasks having larger priorities being executed before others and tasks with the same priority being executed in order of arrival to the queue [2].

Load balancing is important to consider so that resource utilization is constant and that no resource is overloaded or under-utilized. Load balancing can be measured in CPU load, network load, memory capacity, storage, etc. There are many load balancing algorithms created by researchers [5, 6]. One such type is the family of static algorithms that requires the user having prior knowledge of the system. Other types are the dynamic algorithms for which prior knowledge is not required [3].

II. A CLOUD BASED TASK SCHEDULING APPROACH

As briefly described in the previous section Min-min, Max-min, and Sufferage are three common and closely related task scheduling algorithms [7, 8]. This section will detail the process of each policy and provide the basis for the proposed algorithm BSufferage.

The Min-min scheduling policy is split into two phases: first is the calculation of the minimum completion time for every task in queue, and second is the allocation of the task with the minimum completion time to its corresponding virtual machine. Completion time is computed as the sum of the task's execution time (which depends on the size of the task and the

978-0-7381-4394-1/21/$31.00 ©2021 IEEE

specifications of the virtual machine) and the time until the virtual machine is available. Each virtual machine can only execute one task at a time, so tasks must wait in a queue until being assigned. Once all combinations are calculated, the task with the lowest potential completion time is assigned to the virtual machine with the earliest completion time. Consequently, tasks with small computing requirements are favored for brokering over tasks with large requirements.

The Sufferage scheduling policy is based around a quantifier called the "sufferage" value, which is calculated for each queued task with lower values representing tasks that makes the system "suffer" by yielding a suboptimal pairing. The sufferage heuristic is determined by first calculating the minimum and second minimum completion times for each queued task; it is defined as the difference between these two values. Like the Min-min algorithm, the completion time is computed as the sum of the task's execution time and the time until the virtual machine is available. Tasks dispatched to virtual machines have their sufferage values checked against the currently executing task and the task with the higher value gets priority over the resource. In contrast to Min-min, the Max-min algorithm arbitrates the task that will be theoretically completed last, rather than selecting a task which will be theoretically completed first. This favors tasks with larger requirements as to avoid long waits and possibly starvation in the execution queue.

Different from Min-min, the Max-min algorithm arbitrates the task that will be theoretically completed last, rather than selecting a task which will be theoretically completed first. This favors tasks with larger requirements as to avoid long waits and possibly starvation in the execution queue.

The three classical cloud task scheduling algorithms described have simple heuristics for determining how to broker tasks to resources—by calculating minimum earliest completion time, maximum earliest completion time, or maximum sufferage value. Consequentially, the cloud system might exhibit unexpected behavior (like task starvation where tasks never leave the queue) due to the simple brokering logic. Also, none of these algorithms consider the load balancing of resources, instead opting for basic heuristics based on completion times. Because tasks and data are more diverse in practice than in theory, more complex heuristics are needed to target multiple performance metrics. To overcome the shortcomings of these algorithms, a new approach is required.

In this paper, we propose an algorithm called BSufferage for task scheduling that augments the performance of algorithm of Sufferage.

A. BSufferage Algorithm

We design an algorithm BSufferage, which is an extension of the traditional Sufferage algorithm. In the first stage of the BSufferage algorithm, we calculate Exe_{ij} which is the time required for them to execute on virtual machines, record $Avail_j$ which is the availability time of each virtual machine in cloud, and compute $Complete_{ij}$ which is the expected computation time of a task t_i to be completed on a virtual machine V_j.

In the classical Sufferage task scheduling policy, the sufferage value is calculated as the difference between the earliest completion time and the second earliest completion time. Thus, the task with the largest sufferage value or difference in first and second completion times is assigned before all others. Though this considers more information than both Min-min and Max-min, the sufferage value merely considers the two quickest theoretical mappings. To illustrate a case in which more information might be required, if a task is not assigned to its first option virtual machine, nor is it assigned to its second option virtual machine, what would be the cost when it must choose the third option virtual machine? What other information from the calculated completion times can we use when choosing which task to broker?

Our BSufferage algorithm attempts to address these issues. BSufferage algorithm is performed in an iterative way. In each iteration, for each task t_i, we first sort the completion times on all the virtual machines, and find three minimum values *min*, *2nd_min* and *3rd_min*. We also calculate the standard deviation σ of the difference of $Complete_{ij}$ and *min*.

Next, we calculate a value

$$s_i = (2nd\_min - min) * (3rd\_min - min) * \sigma (Complete_{ij} - min) \quad (1)$$

With this augmentation on the original algorithm, not only are tasks with large differences between the earliest and second earliest completion times preferred, but also tasks with large differences between second and third earliest completion times. In addition, the standard deviation of all sufferage values between earliest completion time and all other completion times is recorded and serves as the quantifier that determines which tasks are brokered to which virtual machines.

In each iteration, we choose a task t_q with the largest value of s_i and assign it to its first option virtual machine V_p that gives its earliest completion time and remove t_q from the task pool. At the end of this iteration, the available time of V_p will be updated because it accepted a new task t_q, and the completion times of all remaining tasks on V_p will be updated accordingly.

III. Experiments

The proposed BSufferage algorithm is implemented in the free and open-source CloudSim cloud computing library for Java on a computer with Intel Core i5-4430 @ 2.90 GHz and 8 GB of RAM. CloudSim was selected to test the algorithms because it allows for the simulation of multiple cloud components such as Datacenter, Host, Broker, Virtual Machine and Cloudlet [4]. This benefits the experiments by minimizing the number of unknown variables within the model and by providing a controlled, transparent environment that can be modified on-the-fly. CloudSim is highly extensible and offers many add-ons and features in addition to the components contained in the main package. In the example code and documentation are numerous examples of common cloud computing scenarios including network topology, the MapReduce programming paradigm, and cloud task scheduling.

The performance metrics being recorded and measured during the simulation to compare between BSufferage and Sufferage are turnaround time, load balancing, and throughput. Turnaround time refers to the time between submission of a task and its completion. Load balancing is measured through the

standard deviation of the virtual machine load as described in the previous section. Throughput is the number of tasks that the system completes in a given span of time. A program is implemented to generate synthetic data of the computational power of each virtual machine and size of each task. The number of virtual machines tested ranges from 9, 12, …, 30. The number of tasks (described by Cloudlets in CloudSim package) ranges from 10, 20, …, 100.

A. Experiments on BSufferage

Figure 2 shows the change of throughput (Y axis) for BSufferage on various Cloudlets (10 to 100) as the number of the virtual machines (VMs) changes from 9 to 30 (X axis). Figure 2 shows that when the number of VMs increases, throughput increases as well. Figure 3 shows the change of turnaround time (Y axis) for BSufferage on various Cloudlets (10 to 100) as the number of the virtual machines (VMs) changes from 9 to 30 (X axis). Figure 3 shows that when the number of VMs increases, the average turnaround time decreases.

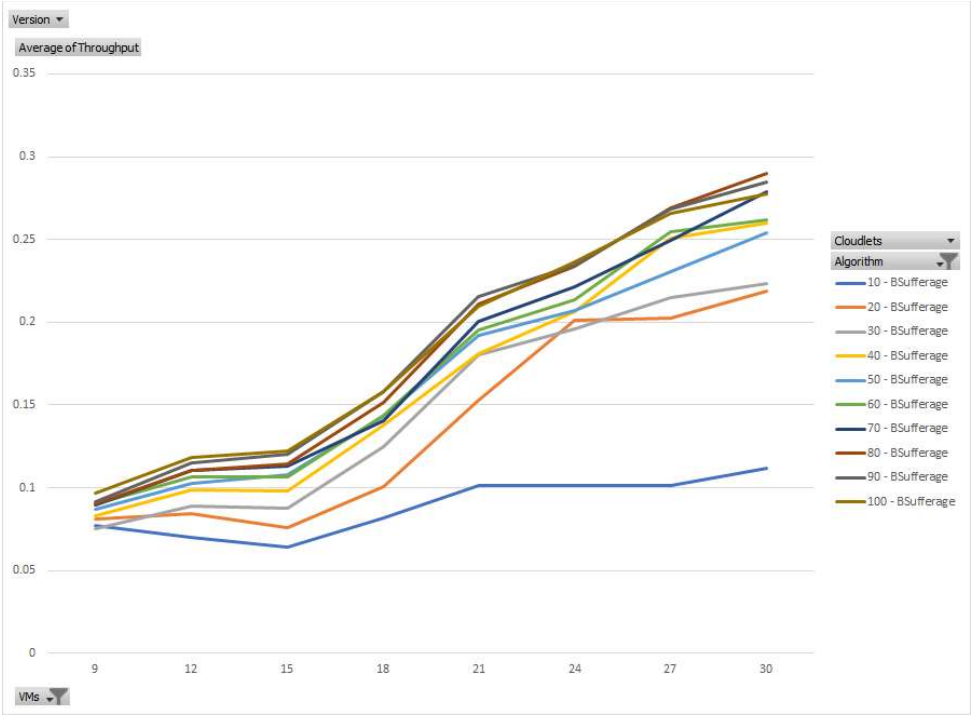

Fig. 1. Throughput for BSufferage on Cloudlets (10 to 100) and Virtual Machines (9 to 30)

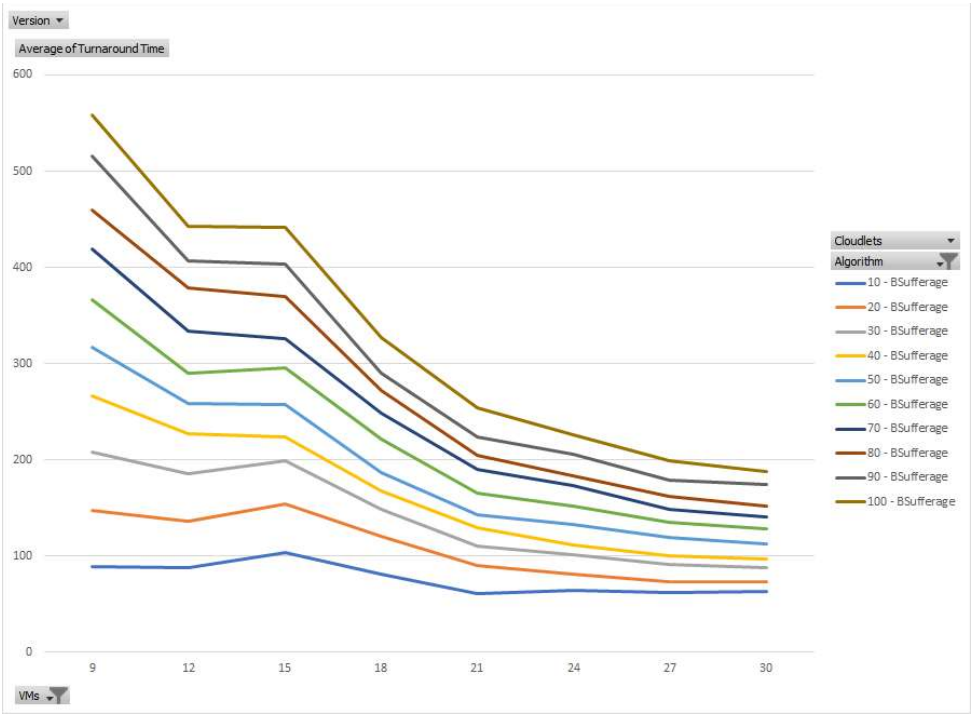

Fig. 2. Turnaround Time for BSufferage on Cloudlets (10 to 100) and Virtual Machines (9 to 30)

We compare the performance of BSufferage with the classic Sufferage algorithm. Figures 4, 5, and 6 show examples of the comparison of BSufferage and Sufferage on throughput when they are performed on various number of cloudlets; from these, we can see that by average BSufferage has higher throughput than Sufferage. Figure 7 shows an example of the comparison of BSufferage and Sufferage on the turnaround time; from this, we can see that BSufferage has lower turnaround time than Sufferage.

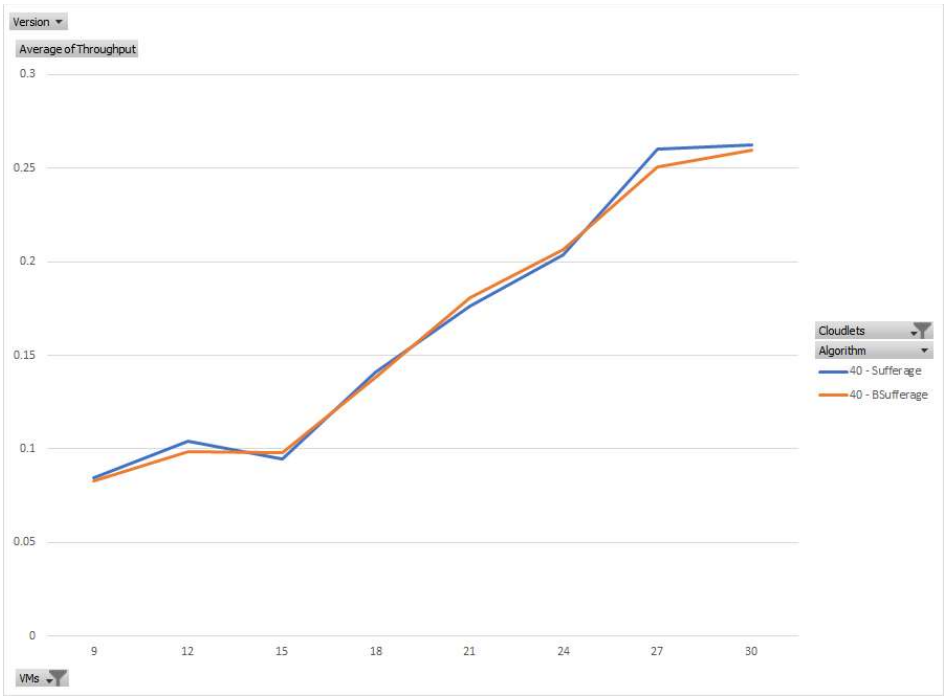

Fig. 3. Throughput of BSufferage vs Sufferage with 40 Cloudlets

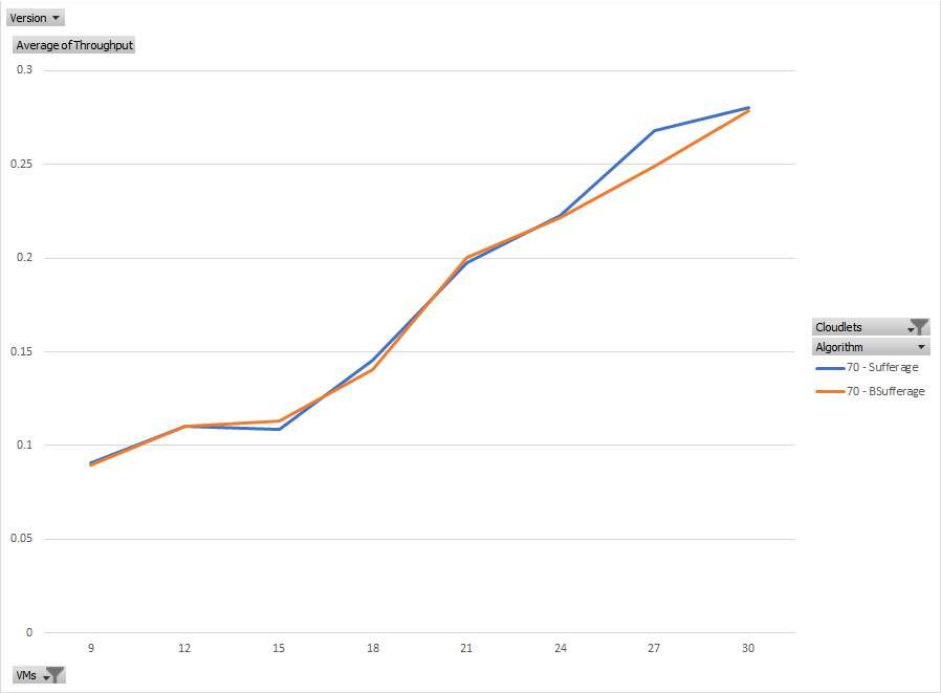

Fig. 4. Throughput of BSufferage vs Sufferage with 70 Cloudlets

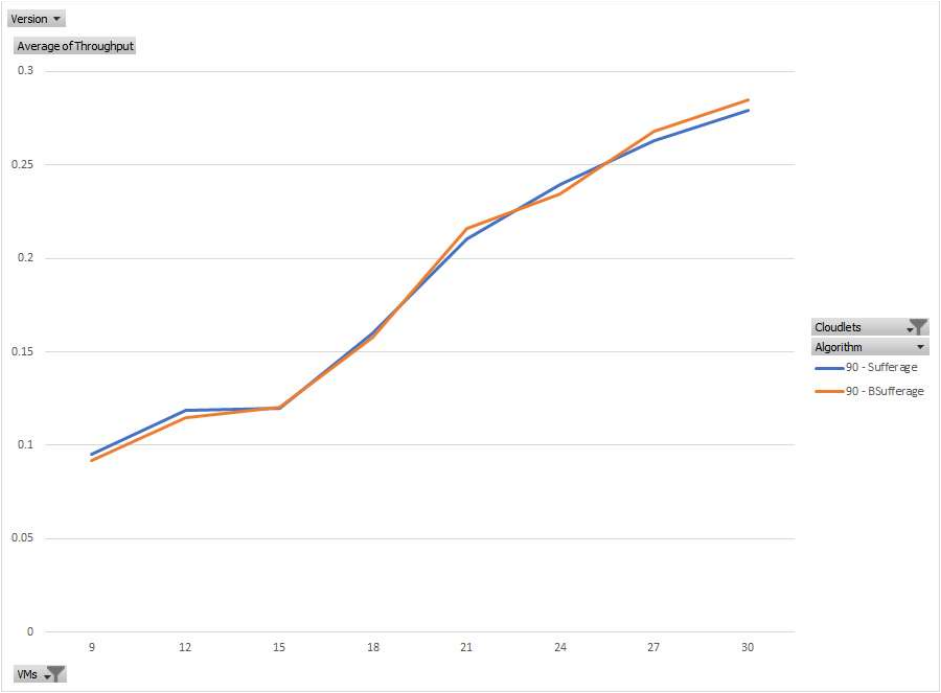

Fig. 5. Throughput of BSufferage vs Sufferage with 90 Cloudlets

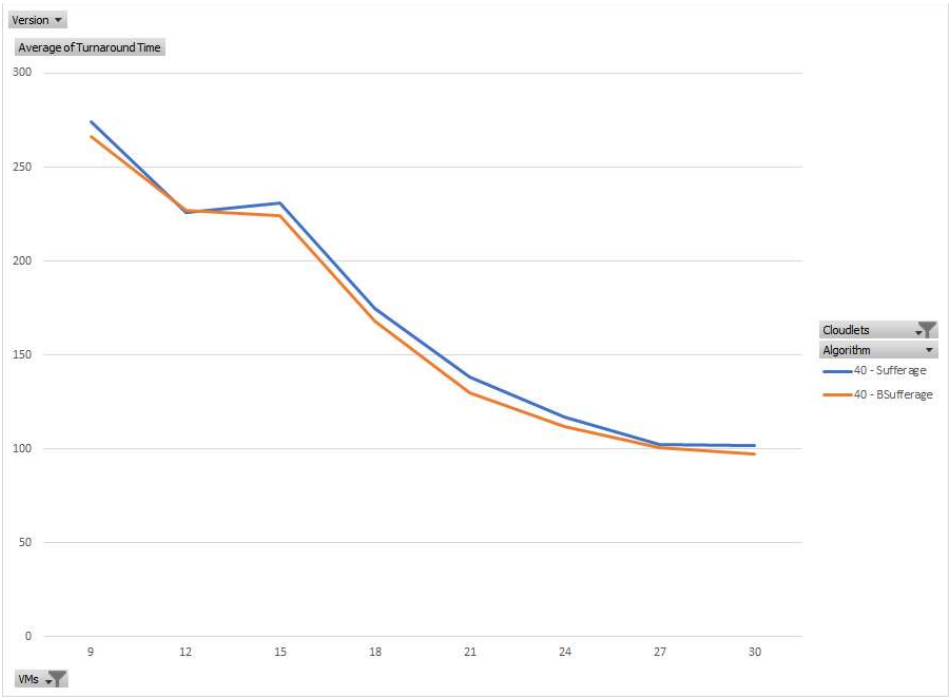

Fig. 6. Turnaround Time of BSufferage vs Sufferage with 40 Cloudlets

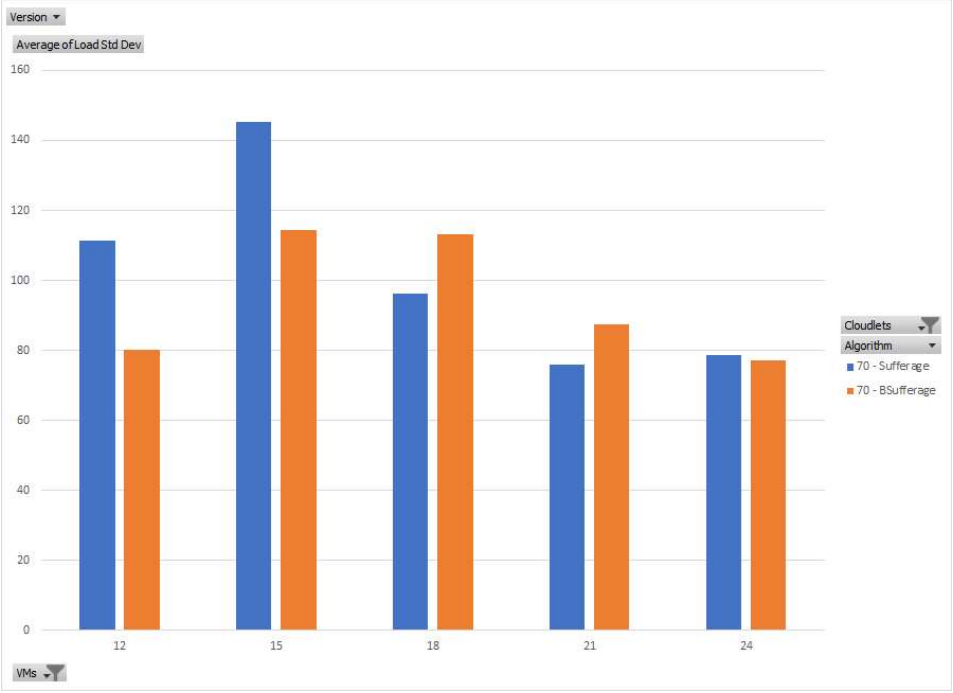

Fig. 7. VM load standard deviation of BSufferage vs Sufferage with 70 Cloudlets

Figures 8 shows an example of the comparison of BSufferage and Sufferage on the standard deviation of virtual machine workload when they are performed on various number of cloudlets. From figure 8 we can see that by average BSufferage has smaller standard deviation than Sufferage, which mean it achieves better load balancing than Sufferage.

IV. Conclusion

Augmenting the classical Sufferage cloud task scheduling algorithm yields the BSufferage policy with improved turnaround time, load balancing, and throughput. The performance of the algorithm is determined by an experiment run in the CloudSim framework and compared with its ancestors. From the results of the experimental data, our algorithm is measured to achieve a generally decreased completion time and improved load balancing of computing resources compared to Min-min and Sufferage.

References

[1] R. M. Singh, S. Paul and A. Kumar, "Task scheduling in cloud computing", International Journal of Computer Science and Information Technologies, Vol. 5 (6), 2014.

[2] P. Salot, "A survey of various scheduling algorithm in cloud computing environment", M.E, computer engineering, India,

[3] R. P. Padhy, P. G. P. Rao, "Load Balancing in Cloud Computing Systems", National Institute of Technology, Rourkela, May,2011.

[4] The Cloud Computing and Distributed Systems (CLOUDS) Laboratory, University of Melbourne. http://www.cloudbus.org/intro.html

[5] E. J. Ghomi, A. M. Rahmani, and & N. N. Qader, (2017). Load-balancing algorithms in cloud computing: a survey. Journal of Network and Computer Applications, 88, 50-71.

[6] Q. Xu, R. V. Arumugam, K. L. Yong, Y. Wen, Y. S. Ong, and W. Xi (2015). Adaptive and scalable load balancing for metadata server cluster in cloud-scale file systems. Frontiers of Computer Science, 9(6), 904-918.

[7] E. K. Tabak, B. B. Cambazoglu and C. Aykanat, "Improving the Performance of Independent Task Assignment Heuristics MinMin,MaxMin and Sufferage," in IEEE Transactions on Parallel and Distributed Systems, vol. 25, no. 5, pp. 1244-1256, May 2014. doi: 10.1109/TPDS.2013.107

[8] M. Maheswaran, S. Ali, H. J. Siegal, D. Hensgen and R. F. Freund, "Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems," Proceedings. Eighth Heterogeneous Computing Workshop (HCW'99), San Juan, Puerto Rico, USA, 1999, pp. 30-44. doi: 10.1109/HCW.1999.765094