Task Scheduling Algorithms: Sufferage and Max-min Modifications

Sana Rashid
College of Computing and Sofware Engineering
Kennesaw State University
Marietta, USA
srashid7@students.kennesaw.edu

Yong Shi
College of Computing and Software Engineering
Kennesaw State University
Marietta, USA
yshi@kennesaw.edu

Abstract— **The use of cloud computing has increased over the past few years as it provides many benefits in terms of flexibility, scalability, elasticity, and even security compared to traditional computing methodologies. One integral part of cloud computing is task scheduling and this process allows the cloud to allocate resources in an efficient way to their respective tasks. Among the many task scheduling algorithms there is Max-min, Min-min, and Sufferage. This paper goes into detail how the Sufferage and Max-min algorithms can be modified to produce different results and possibly different interpretations from the original algorithms as well.**

Keywords— Task Scheduling, Cloud Computing, Sufferage, Max-min, Min-min, Ready Time, Virtual Machine

I. Introduction

Cloud computing is the delivery of computing services such as databases, networking, software analytics, servers and much more through a virtual hosting platform instead of physical hardware as is the case in traditional computing. There are numerous benefits in using cloud computing compared to traditional methods as it faster, flexible, scalable, and economical. “Developers [using cloud computing] are unburdened with the purchasing, configuring, provisioning, and maintaining of computing resources so that development time can be utilized efficiently” [4]. Additionally, cloud computing is much more reliable, so if one server fails, the data is not completely lost because it is stored in the cloud. Another benefit is that it is easily scalable since this computing is an on-demand service and developers can easily request more or less services whenever the need arises. Many companies like Amazon Web Services, Microsoft Azure, Google Cloud Platform, Oracle, and many more have implemented the cloud computing methodology.

The cloud provides many different resources and tasks, and to use those resources efficiently there needs to be a method of scheduling. Scheduling is the method by which threads, processes or data flows are given access to system resources, and the need for a scheduling algorithm arises from the requirement for most modern systems to perform multitasking (execute more than one process at a time) and multiplexing (transmit multiple flows simultaneously) [5]. In cloud computing, task scheduling is crucial for any cloud provider to reduce costs and maximize efficiency, flexibility, and scalability [3]. This, in turn, allows the user to have the best service. The most common task scheduling algorithms are First-Come, First Served (FCFS), Shortest Job First (SJF), Round-Robin, Max-min, Min-min, and Sufferage. This report will focus on the Max-min, Min-min, and Sufferage algorithms and their modifications.

II. Related Works

There has been some research with regards to task scheduling and how it is important in cloud computing. Some previous research papers have discussed how to make the Min-min, Max-min, and Sufferage algorithms better by improving runtime or changing some parts in the algorithm to increase efficiency. However, research has not been done on how to modify the algorithms so that the output is changed and can possibly give a new interpretation of the algorithm and this idea will be discussed and implemented later in this paper.

As discussed in the previous section, there are many different types of task scheduling algorithms and the most extensive ones are Min-min, Max-min, and Sufferage. In [1], the authors describe how these three algorithms run on O(N2) time and how to improve the run time complexity to O (N log N) time. Since only improvement was made on the run time of the algorithms, the output of the new algorithms is still the same and no new interpretation is made.

Similarly, research has been done to address potential problems with the Sufferage algorithm such as what to do if a task is not assigned to its first or second option virtual machines [4]. This allows for optimization of the Sufferage algorithm and keeps possible issues at bay, however, the output remains mostly the same.

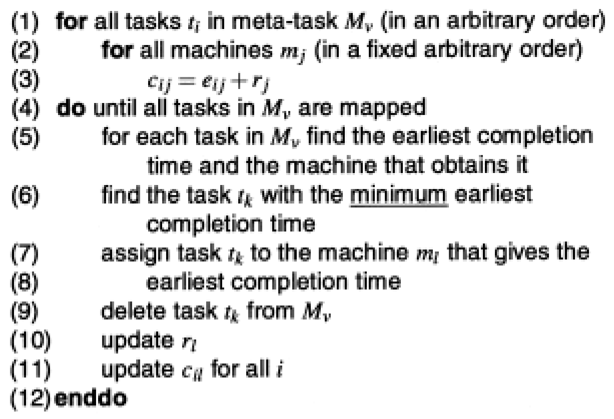
Most of the papers that have been made deal with how to optimize the current Min-min, Max-min, or Sufferage algorithms, but now how to change them to produce different results. Another example is in [2] the authors explain the Max-min, Min-min, and Sufferage algorithms in detail and how to implement them. This paper was heavily used in my research as it describes the three algorithms extensively. Moreover, the paper outlines a basic explanation but does not go into much information on how these algorithms can be optimized or changed to give distinct results as will be explored in my research.

III. Existing Algorithms

Below is a brief discussion of the original Min-min, Max-min, and Sufferage algorithms. For all the algorithms the main goal is to assign each task to a virtual machine, so the iteration will continue until all tasks are assigned.

A. *Min-min*

This algorithm allocates the resources to tasks that will allow the fastest completion time. The first step in the Min-min algorithm is to divide each cloudlet's ready time by the virtual machine's ready time to create a two-dimensional matrix of completion times. Store this matrix in a temporary two-dimensional array as it will be used later on. Then, each column of the two-dimensional matrix should be added with the corresponding ready times which are stored in an array. After this, a while loop should be used to traverse each row of the matrix to find the minimum completion times. The minimum completion time is stored in another data structure with its respective virtual machine. Next, from the minimum completion time array, the least value is considered and its virtual machine and this value is updated in the ready time array for that virtual machine. Finally, the virtual machine with the least time is mapped to its corresponding task, and these steps continue until all the tasks are assigned. The algorithm below explains how Min-min can be implemented.



```
(1)  for all tasks $t_i$ in meta-task $M_v$ (in an arbitrary order)
(2)       for all machines $m_j$ (in a fixed arbitrary order)
(3)            $c_{ij} = e_{ij} + r_j$
(4)  do until all tasks in $M_v$ are mapped
(5)       for each task in $M_v$ find the earliest completion
               time and the machine that obtains it
(6)       find the task $t_k$ with the minimum earliest
               completion time
(7)       assign task $t_k$ to the machine $m_l$ that gives the
(8)            earliest completion time
(9)       delete task $t_k$ from $M_v$
(10)      update $r_l$
(11)      update $c_{il}$ for all $i$
(12) enddo
```

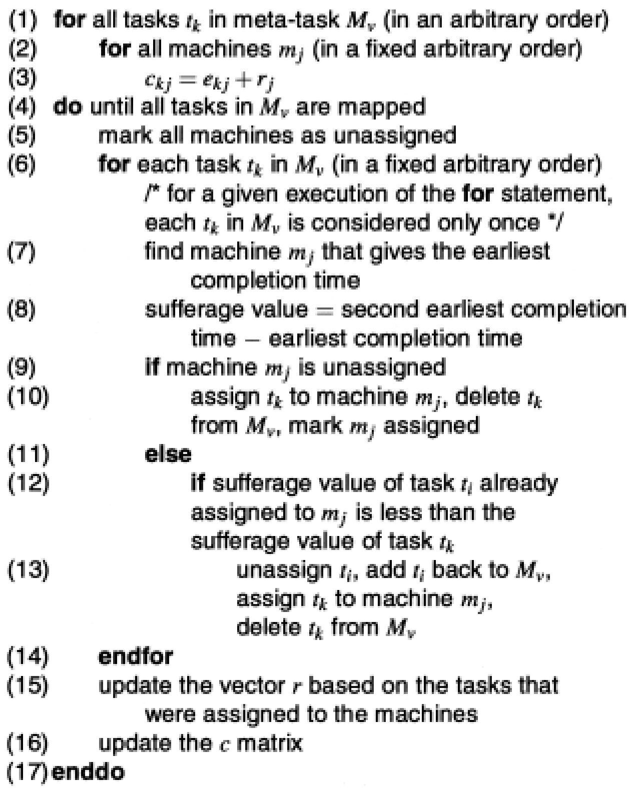
Fig. 1. Min-min Algorithm [1]

B. *Max-min*

In contrast to Min-min, this algorithm allocates the resources to tasks that will have the latest completion time. The Max-min algorithm is very similar to Min-min, however, the maximum completion time is taken into account when comparing the completion time values instead of the minimum. Then the maximum value is updated in the ready time array with the corresponding virtual machine and the task is mapped to that virtual machine.

C. *Sufferage*

The first step in the Sufferage algorithm is the same as the two previous task scheduling algorithms of creating the two-dimensional matrix with the ready times. The main difference of in the Sufferage algorithm is that it “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” [4]. To calculate the sufferage, the first minimum and second minimum of each task is subtracting and stored in a data structure such as an array, list, or Hash Table. Then each sufferage value is considered to its corresponding task and the machine with the minimum completion time. If the machines are similar for more than one task, then the smallest completion time is considered, and the process is repeated for the remaining tasks. The algorithm below explains how the Sufferage algorithm can be implemented:



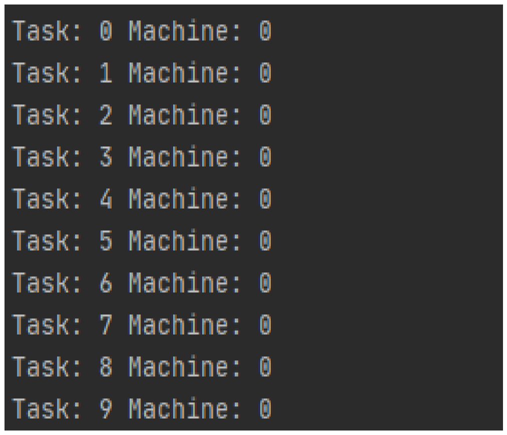
```
(1)  for all tasks $t_k$ in meta-task $M_v$ (in an arbitrary order)
(2)      for all machines $m_j$ (in a fixed arbitrary order)
(3)          $c_{kj} = e_{kj} + r_j$
(4)  do until all tasks in $M_v$ are mapped
(5)      mark all machines as unassigned
(6)      for each task $t_k$ in $M_v$ (in a fixed arbitrary order)
             /* for a given execution of the for statement,
             each $t_k$ in $M_v$ is considered only once */
(7)          find machine $m_j$ that gives the earliest
                 completion time
(8)          sufferage value = second earliest completion
                 time − earliest completion time
(9)          if machine $m_j$ is unassigned
(10)             assign $t_k$ to machine $m_j$, delete $t_k$
                 from $M_v$, mark $m_j$ assigned
(11)         else
(12)             if sufferage value of task $t_i$ already
                 assigned to $m_j$ is less than the
                 sufferage value of task $t_k$
(13)                 unassign $t_i$, add $t_i$ back to $M_v$,
                     assign $t_k$ to machine $m_j$,
                     delete $t_k$ from $M_v$
(14)     endfor
(15)     update the vector $r$ based on the tasks that
             were assigned to the machines
(16)     update the $c$ matrix
(17) enddo
```

Fig. 2. Sufferage Algorithm [1]

IV. Proposed Algorithms

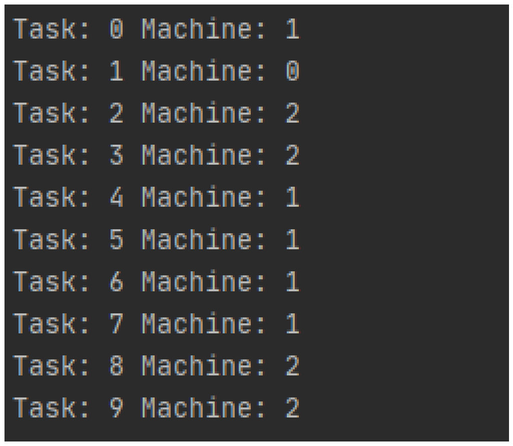
A. *Sufferage Modification*

Currently, the sufferage algorithm calculates the sufferage by subtracting the minimum and second minimum completion time of each task. To modify this algorithm, I decided to calculate the sufferage by subtracting the maximum and second maximum completion time of each task. This will allow us to see which machines will make the system "suffer" more as we are taking into account the maximum and not the minimum values. Below is the output of the original and updated Sufferage code respectively. The output demonstrates which virtual machine is assigned to each task.



```
Task: 0 Machine: 0
Task: 1 Machine: 0
Task: 2 Machine: 0
Task: 3 Machine: 0
Task: 4 Machine: 0
Task: 5 Machine: 0
Task: 6 Machine: 0
Task: 7 Machine: 0
Task: 8 Machine: 0
Task: 9 Machine: 0
```

Fig. 3. Original Sufferage Output



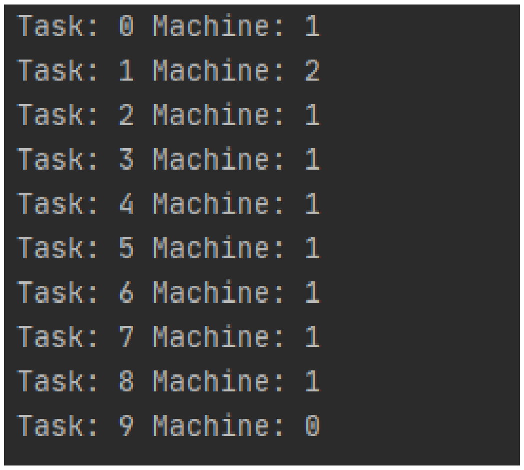
```
Task: 0 Machine: 1
Task: 1 Machine: 0
Task: 2 Machine: 2
Task: 3 Machine: 2
Task: 4 Machine: 1
Task: 5 Machine: 1
Task: 6 Machine: 1
Task: 7 Machine: 1
Task: 8 Machine: 2
Task: 9 Machine: 2
```

Fig. 4. Modified Suffereage Ouput

As can be seen, both outputs have different values and illustrate how one can assign the best machines to the tasks to make the system work more efficiently.

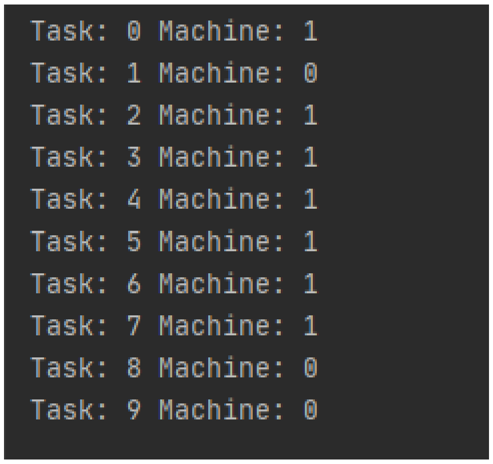
B. *Max-min plus Sufferage*

Currently, the Max-min algorithm only looks at the minimum completion time of each task and considers the tast with the largest completion time to assign to the respective machine, and there is no sufferage Involved. To modify this algorithm, I decided to combine the Max-min and sufferage algorithms to achieve different resutls. I calculated the sufferage of Max-min by subtracting the first m minimum and second minimum completion time of each task. Then I took the sufferage with the maximum and assigned the task to the corresponding machine with the maximum runtime. This is a little different that the original sufferage as I am considering one task at a time looking into if the machine has already been mapped to a certain task. The output of the original Max-min and Max-min plus sufferag is shown below:



```
Task: 0 Machine: 1
Task: 1 Machine: 2
Task: 2 Machine: 1
Task: 3 Machine: 1
Task: 4 Machine: 1
Task: 5 Machine: 1
Task: 6 Machine: 1
Task: 7 Machine: 1
Task: 8 Machine: 1
Task: 9 Machine: 0
```

Fig. 5. Orignial Max-min Output



```
Task: 0 Machine: 1
Task: 1 Machine: 0
Task: 2 Machine: 1
Task: 3 Machine: 1
Task: 4 Machine: 1
Task: 5 Machine: 1
Task: 6 Machine: 1
Task: 7 Machine: 1
Task: 8 Machine: 0
Task: 9 Machine: 0
```

Fig. 6. Max-min plus Sufferage Output

Acknowledgment

I would like to extend my special thanks to Professor Yong Shi for facilitating my first research project throughout the Spring 2022 semester. It was definitely a learning experience and I hope to expand my knowledge in the future.

References

[1] 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

[2] 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

[3] 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.

[4] Y. Shi, K. Suo, J. Hodge, D. P. Mohandoss and S. Kemp, "Towards Optimizing Task Scheduling Process in Cloud Environment," 2021 IEEE 11th Annual Computing and Communication Workshop and Conference (CCWC), 2021, pp. 0081-0087, doi: 10.1109/CCWC51732.2021.9376146.

[5] Shi, Yong. “What Is Task Scheduling?” CloudComputingExercises, https://sites.google.com/view/cloudcomputingexercises2/what-is-task-scheduling?authuser=0.