Enhanced Hybrid Algorithm of Min-Min and Max-Min Task Scheduling Algorithms with Load Rebalancing

Andy Banh, Evan Mikesell
San Diego State University
avbanh@sdsu.edu, emikesell@sdsu.edu

I. Introduction

As the usage of cloud computing grows, the need for more efficient task scheduling and load balancing algorithms grows as well. The Hybrid Algorithm of Min-Min and Max-Min Task Scheduling, or HAMM, is an algorithm that combines the Min-Min and Max-Min scheduling algorithms in a way that helps mitigate the disadvantages of each algorithm in order to achieve a more efficient scheduling algorithm [2]. However, the HAMM algorithm leaves room for better load balancing. Therefore, we are proposing a new solution that improves HAMM to a version that is more capable of balancing tasks thereby improving its performance and efficiency in cloud computing. We will call this algorithm Enchanced Hybrid Algorithm of Min-Min and Max-Min, or EHAMM. EHAMM will add an additional step of balancing loads on machines when large imbalances are present after conducting the HAMM algorithm.

This proposal will outline: our motivation for tackling this problem, a summary of what our project will focus on, the challenges that we may run into while implementing EHAMM, what we hope to produce from this project, and finally our timeline of what our plan is for this problem.

II. Motivation

The need for a good task scheduling and load balancing algorithm existed long before cloud computing became prevalent, it is also a huge focus in the grid computing world, or when balancing the execution of processes on a CPU. As such, the problem of cloud task scheduling has been studied extensively over a long period of time and has many different solutions available for implementation. When it comes to cloud computing, scheduling tasks in a smart manner is important due to the increased costs involved when operating at a large scale. Optimal cloud task scheduling is particularly interesting because it is an NP-Complete problem with many different algorithms proposed for solving it [4]. These algorithms will range from simpler heuristic algorithms like Shortest Job First or Suffrage, to more complex meta-heuristic algorithms like the Genetic algorithm or the Ant Colony Optimization [1].

Two popular heuristic solutions are the Min-Min and Max-Min algorithms. The Min-Min algorithm focuses on assigning tasks with the shortest completion time to the machine that will give the quickest completion time, while the Max-Min algorithm focuses on assigning tasks with the longest completion time to the machine that gives the quickest completion time [1]. Both algorithms have potential disadvantages, Max-Min can lead to starvation of smaller tasks, while Min-Min can lead to starvation of larger tasks.

The HAMM algorithm is a selective-adaptive algorithm that uses both the Min-Min and Max-Min algorithms. It chooses whether to use the Min-Min or Max-Min method of scheduling each task based on the current state of the tasks that are not yet scheduled [2]. However, there is a lack of task rescheduling functionality that we believe could be included in the algorithm to better its performance. An example of a solution that implements task rescheduling would be the Rescheduling Enhanced Min-Min Algorithm which is a modified version of the Min-Min algorithm with task rescheduling implemented [3]. This algorithm reduces variance in load across machines after assigning the tasks. We hope that applying the same principles with the HAMM to create the EHAMM algorithm, we can achieve a similar kind of improvement in the load balancing.

III. Project Summary

The HAMM algorithm works by separating tasks into groups of large and small tasks and selecting either the Min-Min or Max-Min algorithm based on whether there are more large or small tasks remaining. This is an effective strategy for scheduling tasks, but can be improved. EHAMM will provide more effective load balancing than the HAMM algorithm. We will achieve this through rescheduling our tasks after initially placing them. This will result in a faster makespan, less variance in the load of each machine, and more efficient cloud computing system overall. We will be producing a report that outlines the performance details of EHAMM along with a comparison to its predecessor, the Min-Min algorithm, and the Max-Min algorithm.

IV. Project Details

A. Architecture and Environment

In order to test our algorithm's effectiveness, we will be using CLOUDSLab's CloudSim simulator. This is a simulator that can model a cloud computing environment. The simulator is written in Java and is widely used by those doing

research related to cloud computing. CloudSim allows for the simulation of data centers, virtual machines, and allows us to implement algorithms to schedule our tasks [5]. We will be implementing our EHAMM algorithm, the original HAMM algorithm, Min-Max algorithm, and Min-Min algorithm in order to compare their performances. All of our code implementing these algorithms will be written in Java and both the simulator and our algorithm will be run on the Eclipse IDE.

CloudSim will allow us to track a variety of performance metrics in our experiments, such as the makespan, which is the measure of the time it takes the whole set of tasks to complete. Other metrics that will be looked at include the load balancing and average waiting time of a task. We may consider more performance metrics and extending the functionality of CloudSim if time allows for it after we have looked at the core metrics for our research.

Our simulation will be running on a PC with eight cores, 32 GB of RAM, and a SSD. We hope that by using a powerful PC, it will reduce the chances of any possible variances or irregularities in the data being caused by the device the simulation is running on rather than the simulation itself. This will ensure that we get the cleanest data possible once we begin analyzing the performance for our final report.

Fig. 1 shows our rescheduling technique. We look to move tasks off the higher than average load machines. We define machines as being high load if they are a certain threshold percentage above the overall average load.

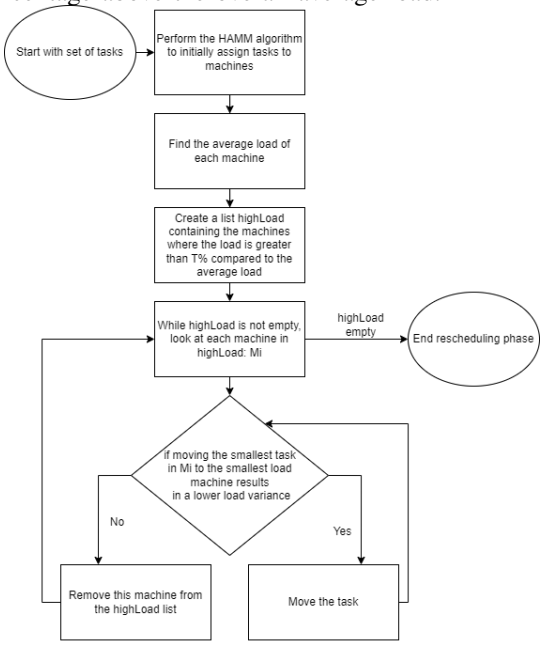

Figure 1: EHAMM Rescheduling

B. Implementation Issues and Challenges

There are a few different challenges we may run into with implementing EHAMM. To start, working with CloudSim could prove to be problematic. It is a complicated library with many different connected parts. We will have to gain a strong understanding of the simulator if we want to use it effectively. This will take some additional research and testing to get things right and is critical to being able to complete any other areas of our research.

Additionally, CloudSim is an ideal virtual representation of a cloud computing network and not a real-world cloud computing network. There are many scenarios that we cannot test in CloudSim that could occur in an actual network or are so specific that CloudSim just simply cannot reproduce. Because of this, our model's performance will be representative of a system that is highly controlled and ideal rather than one that can randomly fault due to unforeseen circumstances at any time. However, this does not mean that our results are invalid, only that the real performance of EHAMM may not reach identical performance levels of our results.

Finally, some performance metrics we are interested in are not able to be tracked easily through CloudSim. For example, tracking the power consumption of the different algorithms could be difficult. We will have to look into extending the functionality of CloudSim if we want to measure our energy efficiency. The task of extending the functionality of CloudSim itself will also be difficult and modification to it can end up creating producing inaccurate data if we are not careful with what we are modifying or if we aren't not implementing new functions exactly as they operate in the real world.

C. Timeline

- *CloudSim setup:* (March 20 - April 7) We will be conducting further research into the CloudSim simulator in order to implement our algorithms. We want to be able to run the simulator with a basic task scheduling method, and ensure that the results from this test are accurate with the expected results of the algorithm before we begin working on our algorithm.
- *Implementing our Improved HAMM algorithm:* (April 7 - April 12) We will begin implementing the improved HAMM algorithm. This process will take around a week to complete. Along the improved HAMM algorithm, we will also implement the original HAMM algorithm, Min-Min algorithm, Max-Min algorithm.
- *Testing the algorithms:* (April 12 - April 17) We will be conducting tests on our algorithms and comparing their performance based on different metrics. If we are ahead of time and finish with collecting data early, we will try to extend functionality of CloudSim and look at other metrics.
- *Data analysis and final report:* (April 17 - May 1) We will take our gathered data and begin analyzing it to create data sheets and visualizations for our final report. Along with this, we will also be preparing our presentation on our research.

Our timeline has a large amount of time allocated towards setting up CloudSim correctly and understanding how to use the simulator fully. We believe this will be the hardest part of our implementation. If it proves to be easier than expected, the timeline will move along much quicker, and we will have more time for other tasks such as implementing increased functionality to CloudSim for testing new metrics or writing the final research paper.

D. Deliverables

Our first and main product is to produce fully functioning virtual cloud computing network operating on our EHAMM algorithm. We hope to see that this version will be even more capable of handling task scheduling and load balancing compared to the HAMM algorithm and produce a greater performance such that it will be viable for use in cloud computing networks. If we still have the time after producing our report on our modified algorithm, we would like to take a look at the HAMM algorithm itself and see if there are any optimizations we can make to the algorithm in order to increase its performance even further.

After we've modified the algorithm and confirmed that it performs better than the original, we will also be producing a report on its performance in regards to other task scheduling algorithms. We will be comparing it to the original HAMM algorithm, the Min-Min algorithm, and the Max-Min algorithm. The report will analyse core metrics such as the makespan and variance in load balancing across virtual machines. If time allows, we'd like to also analyse performance metrics that are more difficult to analyse such as energy usage or critical scenarios such as a system in the cloud network failing while performing a task.

Another possible product of our research may be a CloudSim with extended functionality if we are able to complete our research on the core metrics of our selected task scheduling algorithms ahead of time. While we aren't exactly sure of the full capabilities of CloudSim at this time as we have not been able to work with it extensively yet, we can already see that it lacks a functionality for tracking power consumption.

V. Conclusion

Cloud computing has a growing importance for many businesses today, and with that the importance of efficient task scheduling algorithms grows as well. It is imperative that research is continued into finding higher performing algorithms due to increasing demand on the cloud. We will be improving an existing task scheduling algorithm, HAMM, which is a hybrid algorithm of the existing Min-Min and Max-Min strategies for task scheduling. Our EHAMM algorithm will be adding a step of rescheduling to the HAMM algorithm, in order to fix imbalanced loads on machines, and improve the performance of the algorithm overall.

References

[1] Venu, Gopika. (2020). "Task scheduling in cloud computing: A survey." International Journal for Research in Applied Science and Engineering Technology. 8. 2258-2266. 10.22214/ijraset.2020.5369.

[2] Syed, Ibrahim, "HAMM: A hybrid algorithm of min-min and max-min task scheduling algorithms in cloud computing (November 13, 2020). International Journal of Recent Technology and Engineering (IJRTE) , Volume-9 Issue-4, November 2020,page no:209-218, Available at SSRN: https://ssrn.com/abstract=3922243

[3] Amalarethinam, George & Kavitha, s. (2019). "Rescheduling enhanced min-min (REMM) algorithm for meta-task scheduling in cloud computing. 10.1007/978-3-030-03146-6_102.

[4] Murad, et al. (2022). "Optimized min-min task scheduling algorithm for scientific workflows in a cloud environment". Journal of Theoretical and Applied Information Technology. 100. 480-506.

[5] P. Humane and J. N. Varshapriya, "Simulation of cloud infrastructure using CloudSim simulator: A practical approach for researchers," 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), Avadi, India, 2015, pp. 207-211, doi: 10.1109/ICSTM.2015.7225415.