MODELLING AND SIMULATION OF MIN-MIN AND PRIORITY ALGORITHM IN HPC ENVIRONMENT

Lett Yi Kyaw

Faculty of Computer Science, University of Computer Studies, Yangon (UCSY),
Yangon, State ZIP/Zone, Myanmar
lettyikyaw@ucsy.edu.mm

Kyar Nyo Aye

Faculty of Computer Science, University of Computer Studies, Yangon (UCSY),
Yangon, State ZIP/Zone, Myanmar
kyarnyoaye@ucsy.edu.mm

Abstract

High performance computing (HPC) scheduling is in charge of allocating available resources to execution jobs in a way that optimizes resource usage, reduces make-up, and boosts system performance. When researchers are not able to access the ready-to-use HPC testbed infrastructure, developing it is expensive and time-consuming. If the testbed is accessible, it is restricted to a small number of resources and domains, testing the scalability and adaptability of scheduling algorithms, and assessing scheduler performance for different resource situations and applications is more difficult to track down. From the user's perspective, the system must guarantee rapid response times, equitable resource distribution amongst jobs, and high system utilization and throughput. This paper described the significance and the need for such infrastructure for environments of modeling and simulation of proposed system.

Keywords: High Performance Computing (HPC); scheduling; performance; modeling; simulation.

1. Introduction

A growing focus on concurrent job processing has resulted in a fast improvement in the acceptance of heterogeneous domains and the availability of processor networks that enable the cost-effective utilization of underlying parallelism for applications like real-time and distributed database systems, weather forecasting modeling, and health care prototyping. A large amount of data must grow and the system must automatically calculate in order to investigate its simultaneous processing. Variability is brought into a system by the existence of processors with different features, such as speed, memory capacity, and unique processing capabilities.

HPC uses "supercomputers" and massively parallel processing techniques to solve complex computational problems through computer modeling, simulation, and data analysis. In order to solve complex problems quickly and efficiently, HPC combines a number of technologies under a single framework, including computer architecture, electronics and programs, algorithms, and application software.

Recently, cloud computing has come up as a cost - effective alternative to dedicated HPC application infrastructure. Long - term cloud application running avoids high capital costs and operating costs comrade with a specific HPC infrastructure. The proficiency to provide high on - demand elasticity HPC resource reduces the risks of under - supply and reduces resource consumption caused by over - supply. The cloud - based support for virtualization provides the HPC community with only an alternative way to support flexibility, customization, security, and control of resources. To solve these problems, current HPC scheduling systems are not developed.

The main issue of HPC is trying to perform computational operations on a particular set of processors. While several scheduling heuristics have been developed also represented in the literature, the most of them tend there is a single one at heterogeneous resources. Future computing systems are more similar to be distributed and extremely heterogeneous like the computational grid.

New architectural modifications and fresh technological technologies complex the structure of research operations typically operating on these systems in order for them to function goals. Increasing expect for new resource generation of HPC systems (computing nodes, memory, power, etc.) needs to be supported by the potential to run more and larger applications with better resource efficiency as well as response time of operation for HPC systems.

There is execution of multiple processes simultaneously, so there is increasing demand of more processing speed. The need of scheduling techniques to make them more efficient and get better results are searching and researching. The various parameters those needed to be considered while scheduling in parallel computing.

Scheduling is a key issue focused by many researchers in clusters of jobs. The schedule of jobs to be run at the right place at the right time is required for improving system efficiency. Simulation is a very powerful aid in pursuing research on scheduling policies and performance evaluation. Experimental work on real systems sometimes of heterogeneous resources. The components of a cluster may have dynamic behavior.

A collection of guidelines that establishes what needs to get done when. While many collection scheduling algorithms have been placed on in the literature, the design of those algorithms is complicated by factors such as implementation complexity, support for different service levels, and fairness requirements. The operating system's scheduling and resource allocation components are essential. The scheduling problem is determining which users will be impacted in a specific amount of time. The issue of distributing physical layer resources, such electricity and bandwidth, among these active users is referred to as resource allocation.

2. Related Work

In [1], an open-source scheduler simulation framework (ScSF) covering every stage of scheduling research through simulation has been offered by them. It is beneficial to comprehend how a model is built for scheduling an actual high-performance computing system and to apply a new method using the Slurm simulator. Although jobs are scheduled separately, workflow intermediate wait times are kept to a minimum since they have the same priority. Future work will take into account scheduling research aimed at designing a workflow scheduling algorithm and assessing it via the simulation of an extensive collection of experiments.

The authors [2] of in order to improve performance in decreasing delay time, this HPC system gathered data from the history or log files and then created a relational model between its characteristics and an objective function. In order to dynamically throttle DI applications, the authors of this paper offer an architecture for co-scheduling HPC and DI applications based on resource profile and usage estimates. In order to handle jobs with extremely dynamic changing behaviors, the IE model must retain previous data for a certain amount of time, which is why this paper will need more research in this field.

A task scheduler's configuration and setting parameters have an impact on the criterion. There is a growing demand for intriguing machine learning, AI, and big data analysis. A job scheduling simulator that enables administrators to examine the impact of mapping on job throughput was suggested by them [3]. In order to enable the simulation of targeting on large-scale computer clusters in the future, it will be necessary to plan for a reduction in the work scheduling simulator's processing time.

It is possible to extract the behaviors of large-scale programs running on an HPC system from log files or histories and then create a relational model that relates the program's attributes to its goal. Applying machine learning to find a scheduling function for work submission can solve this problem. The heterogeneity-resource in job scheduling with newly created schedulers that combine machine learning and workload trace features. Applications are started and run on a heterogeneous cluster. The scheduling module [5] aims to decrease delay time and enhance an objective measurement called average slowdown, which states that a job's waiting time ought to be proportional with its processing time.

One of the main study areas in HPC systems and data centers is job scheduling, which is essential to the efficient execution of workloads and the appropriate use of these resources. The existing HPC task scheduling system is not sufficiently intelligent to handle a variety of issues, as seen by the approaches taken by schedulers and the challenges faced by the field [6]. In order to monitor unusual activities and performance in an HPC system and to dynamically enhance system and job performance, they suggested an intelligent framework for HPC job scheduling.

This problem can be resolved by using a machine. Both cloud providers and users can access a multitude of resources as a service [7]. Based on potential ideal characteristics such as throughput, CPU utilization, fairness, turnaround time, etc., they have categorized several algorithms. There are two types of scheduling algorithms: preemptive and non-preemptive. Various scheduling methods are assessed according to factors in order to achieve high accuracy and compatibility for cloud-based HPC applications, since each method addresses different problems and solutions. Then, they used six work schedule traces from four scheduling policies and HPC (FCFS, WFP, SPF, SAF) to demonstrate a method. They also showed that splitting jobs into tiny and enormous classes is all that is needed to improve scheduling performance. Six mechanisms that rely on shrinking, expanding, check-pointing, and resuming ways to dynamically make room for time-sensitive on-demand application were proposed by them to handle the aforementioned difficulties. Within a single HPC system, they have identified and modelled the problem of scheduling hybrid workloads in HPC. A number of strategies have been put forth by them to balance the requirements of rigid, malleable, and on-demand applications. Job turnaround time, instant start rate for on-demand jobs, preemption ratio, and system usage are their evaluation criteria. Five different kinds of

workloads were used. Their intention was to divide work into minor and large categories and to give lesser activities priority over larger ones [3].

They used the Batsim simulator to implement every simulation [11]. They plan to find other work classes in the future that might profit from the batch scheduler giving them special attention. After analyzing the job scheduler's execution log for a predetermined amount of time, they propose an optimization strategy to cut down on busy time spent waiting on resources [9]. The conservative backfilling algorithm that they proposed can be used with the supercomputer. Applying the different scheduling algorithms to the current and analyzing the work execution is essential.

3. Background Theory

The core of cloud computing systems and HPC is job management. The main problems associating the efficiency of the entire computing system are job scheduling. Job scheduling is a mapping process, from user tasks to appropriate resource collection and execution. The idea that computer modeling and simulation constitute a new area of scientific inquiry in addition to theory and experimentation was first proposed. The process of allocating, managing, and maximizing tasks and workloads within a production process is known as scheduling. A computing system that can run large-scale, high-performance applications at the lowest possible cost and with the shortest execution time within a certain time scale is known as high performance computing, or HPC.

The funds request that the HPC job scheduler services be given one or more tasks. Task execution refers to carrying out a user request. How many times to run the command is defined in part by the job type property. It is impossible to do a task without a job. Before a job is finished, it must accomplish one or more tasks. You can specify dependencies between the jobs so that they execute in a particular order. Submission of a job requires the use of a job format. The cluster administrator creates job scripts for various work kinds.

In high-performance computing, task scheduling techniques are primarily used to balance processor load, maximize resource utilization, and minimize makespan. A key element of the HPC-based cloud computing environment is scheduling. Numerous studies have been conducted on different scheduling algorithms that effectively schedule the computational operations being performed in a cloud environment.

Typically, HPC systems are constructed from numerous separate computers, each with a capability comparable to numerous private computers, connected via a network (commonly referred to as the interconnect). It's common to refer to every single computer as a node. Numerous different types of nodes with varying degrees of specialization are frequently found in HPC systems. To communicate with the HPC scheme, you connect to the head (also known as front-end or login) nodes. The real computation is carried out on compute nodes. The computing nodes that have direct access to these resources are managed by a scheduler or batch system. Depending on the HPC design, the computing nodes, even individually, may be significantly more powerful than a standard personal computer. They might have less common accelerators and frequently have different processors, each with a different number of cores (each with various cores) and may have less prevalent accelerators (such as GPUs) and other capacities on personal computers.

3.1. *Categories of Scheduler*

It is prevalent to allocate subsets of compute nodes to assignments (or employment) depending on user demands in order to share big systems among users. It can take a long time to finish these employments, so they come and go in time. HPC systems use a batch system or scheduler to handle the sharing of storage nodes between all the employment. Usually, the batch system has orders for job submission, status inquiry, and modification.

The HPC center decides the priorities of numerous employments on the compute nodes to execution, while ensuring that the compute nodes are not overloaded. First of all, the HPC method is transferring input data sets to the HPC scheme and creating a work submission script for user computing. The scheduler handles user computing after submitting user job submission script to the scheduler. Finally, the findings of user computation evaluate are coming out.

When we analyzed the scheduler in scheduling, there are mainly three types scheduler.

3.1.1. Operating Systems Process Schedulers

Tasks have to be assigned CPU times by an algorithm during the scheduling process. The most well-known process schedulers include Brain Fuck Scheduler (BFS), Completely Fair Scheduler (CFS), and so on.

3.1.2. Cluster Systems Jobs Scheduler

An algorithm has to assign nodes to jobs during event scheduling. Pay attention to high throughput and scalability. The most well-known task schedulers include Maui Cluster Scheduler (Maui), Simple Linux Utility for Resource Management (SLURM), and so on.

3.1.3. Big Data Scheduler

Nodes are assigned duties by an algorithm during event scheduling. Large and complicated data sets are processed and stored in jobs. Provide support for certain frameworks and a narrow range of issues. The most well-known process schedulers include Spark, Hadoop, and so on.

3.2. *Types of HPC jobs*

Five types of jobs can be distinguished from the global perspective of the job scheduler; rigid jobs, moldable jobs, malleable jobs, evolving jobs, adaptive jobs.

3.3. *Scheduling algorithms*

There are two primary categories of scheduling algorithms: time-sharing algorithms, which allocate certain jobs to discrete intervals, or slots, of processor time. In contrast, space-sharing algorithms allot the necessary resources to a specific task before it is executed completely. Usually, cluster schedulers operate in space-share mode.

3.3.1. Min-min and priority algorithms

Using a permanent work scheduling system that balances load is the simplest method to explain the Min-min algorithm [17] in grid computing. The Min-min method first determines which task has the quickest execution time overall. The job with the quickest execution time is then selected out of all of them. The work will keep getting assigned by the algorithm to a resource that gives an accurate duration of completion. The min-min method [16] offers a better makespan while being easy to use and quick. However, it prioritizes the shortest jobs, which prevents it from making effective use of the resources and results in an imbalance in the load. They presented enhance Min-min algorithm [8] is based on Min-min algorithm proposed.

1.For all submitted tasks in the set; T_i

2. For all resources; R_j

3. $Ct_{ij}=Et_{ij}+rt_j$; End For; End For;

4. Do while jobs set is not empty

5. Find tasks T_k that cost minimum execution time.

6. Assign T_k to the resource R_j which gives minimum expected complete time.

7. Remove T_k from the tasks set

8. Update ready time rt_j for select R_j

9. Update C_{ij} for all T_i

10. End Do

Fig. 1. Min-min algorithm

The incoming process is scheduled using a scheduling algorithm based on priority [12]. Priority scheduling runs the processes in order of priority if there are two tasks or processes that are in the ready state, or prepared for execution, and they have the same priority. Which one should be prioritized is determined by the user using the priority algorithm. A method of scheduling processes based on priority is called priority scheduling.

3.4. *Performance Metrics of Scheduling Algorithms*

Software for managing jobs and resources is evaluated based on a number of factors. The goal of each job is certainly to do it correctly the first time, but performance must also be assessed systemically. The following are some of the main features that are taken into consideration:

Throughput: the quantity of tasks finished in a given amount of time

Turnaround time: the amount of time it takes to finish a process after it is submitted

Waiting time: Waiting time is the total amount of time a process spends waiting in line.

Response time: Response time is the interval of time that passes between issuing a command and the output's initial generation. The length of time it takes the system to respond to a user's request for the first time.

CPU Utilization: CPU usage is the proportion of time that the CPU is used and not idle.

Makespan: The whole time needed to complete a set of tasks, beginning to end. The makespan is the longest time that any task can take to finish. The minimal makespan of the scheduler is necessary for optimal system performance.

3.5. *Simulator*

When predicting and analyzing the performance of both present and future HPC systems, modeling and simulation are essential tools. The job scheduling algorithms are designed with certain performance and optimization objectives in view.

To study multi-site parallel work scheduling techniques within a multi-cluster computing Grid, the BeoSim has been put into practice. Real workloads or synthetic workloads can both power the BeoSim [4].

Batsim allows a production scheduler to be used in simulation. It is in charge of simulating the RJMS whereas the scheduling component of the real RJMS makes decisions [11].

While the SimGrid [10] , the Simbatch and the SimBONIC are all based on the C programming language, all following simulators are based on the popular Java language . While Java is usually efficient than C, it allows easy development and effortless portability. Some of researchers, Gridsim is extend SimGrid to construct simulator

Alea is built on the GridSim simulation framework. task submission, resource management, and task dispatching are driven by the preset workload format, resource categories, and job dispatchers. Java implementation that is cross-platform and open-source [13] [14] [15].

3.6. *Simulator Parallel Workloads Behavior*

The cumulative experience with the Parallel Workload Achieve, a library of job level consumption data from large-scale parallel supercomputers, clusters, and grids, has shown many of these challenges. Concurrent workload data issues include missing, inconsistent, or inaccurate data; changes in device configuration during the logging cycle; and user behavior that is not representative of the user base.

Scheduling parallel jobs will involve a specific number of processors in each job, and so on. The best scheduling algorithm can depend on the job size distribution, or the possible correlation between job size and run time.

Generally, we can define a few measures that can be used as a standard basis for any analysis of characterizing the workload;

1) Choice of the set of parameters capable of defining workload behavior.
2) Selection of the correct instrumentation, i.e., use of existing performance/monitoring tools or ad hoc after installation.
3) Collection of experimental measurements.
4) Review of data about the workload.
5) Construction of static/dynamic workload models.

3.6.1. Log Formats

Where jobs are submitted using a historical workload description file (log), and the log's original timestamps are used to determine when they should arrive. dynamic workload: in this [19], the scheduler's performance determines whether job arrival times remain constant or fluctuate during the simulation. Thus, the simulator offers a feedback loop that interacts with the workload reader and notifies it when a task is finished. The ability to parse logs is a requirement for log analysis. Certain systems have specified standard log formats, but there isn't one for parallel task schedulers; instead, each has a unique format with peculiarities of its own. Compute-intensive tasks are typically executed on HPC infrastructures.

The applications that operate on HPC platforms and the resource and job management systems that coordinate and oversee resource sharing are just two examples of the platforms' many layers of hardware, software, and services.

4. Proposed System

When user submit jobs to the system, in the proposed system workloads are read by workload reader. In this proposed system tested by using simulator because it is used to test different scheduling policies independent of the physical computer hardware. HPC supercomputers are expensive, costing hundreds of millions of dollars annually to operate so researchers are using simulators for experimental scheduler testing that could reduce to long job delays and even errors.

The proposed method, called the scheduler, chooses a resource machine by scheduling it based on procedure and configuration system. And result collector collect result from workload file after scheduling. Output will show the graph by using visualization or .csv format.

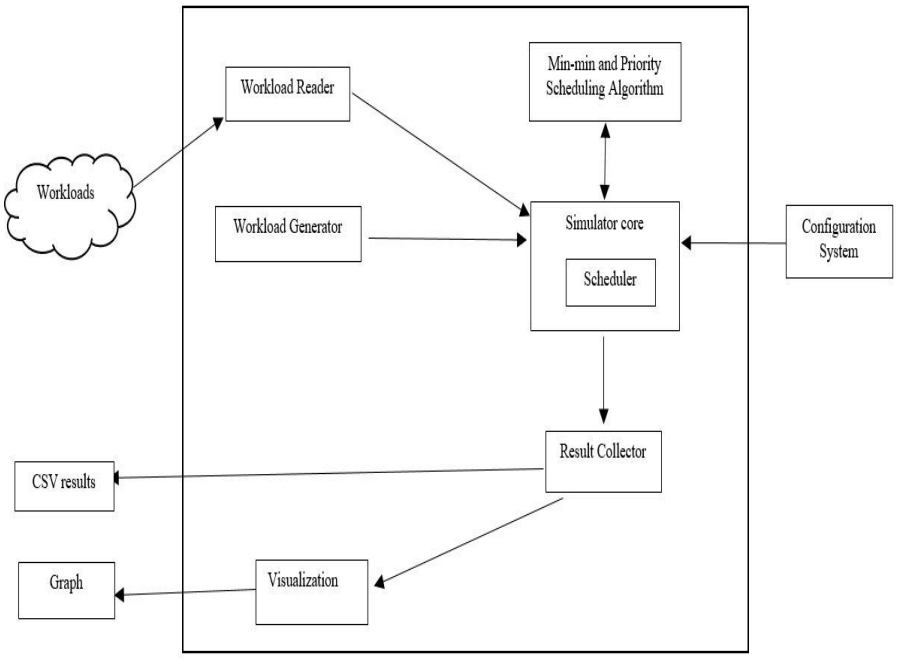

Fig. 2. Proposed System

One of the most used scheduling techniques in batch systems is priority scheduling. A priority is given to every process. The highest priority process should be carried out first, and then others. Priority processes are carried out in order of arrival to the processing queue. Any resource demand, including memory and time, can be used to determine priority [12]. If we have one short task, the Min-min could simultaneously perform several long tasks while performing one shorter. In this case the total makepan is calculated by quick task execution. In proposed system, the incoming tasks are classified their categories that may reduce the execution time of Min-min algorithm. In HPC system, machine clusters are assigned with appropriate task to process. The Min-min algorithm's main disadvantages are load imbalance, and job starvation with large service time. Since HPC system processing involves job scheduling and then investigating the variety of scheduling techniques that were implemented in the past became important.

The proposed system is based on Min-min algorithm and priority scheduling algorithm. Compute the job size and predicted execution time when the user submits a batch job (job size is equal to the number of cores the job requires). When computing the job size, select the task with the shortest anticipated execution time if the size of the job is less than or equal to the resources available. When a job's size is less than or equal to the threshold value, it is assigned to a small compute node; however, because the small compute node is busy during that period, it is assigned to a large compute node. If we have one short task, the Min-min could simultaneously perform several long tasks while performing one shorter. In this case the total makepan is calculated by quick task execution. In proposed system, the incoming tasks are classified their categories that may reduce the execution time of Min-min algorithm. In HPC system, machine clusters are assigned with appropriate task to process. The Min-min algorithm's main disadvantages are load imbalance, and job starvation with large service time. Since HPC system processing involves job scheduling and then investigating the variety of scheduling techniques that were implemented in the past became important. On the other hand, jobs given to large compute nodes have sizes that are neither smaller than nor equal to the threshold value. Wait till the resources are available if the size of the task does not exceed or equal the available resources. After allocating resources, update the job set and compute node list.

5. Conclusion

Some existing algorithm having to be considered the performance of the system is not satisfied. The issues are still and researchers are trying to solve them. Jobs can be categorized according to user requirements, and then the optimal running time can be determined based on the various objectives for every job. It will improve the quality of job scheduling indirectly in a cloud environment and HPC. Resource and Job Management for HPC is a continuously evolving domain. Algorithm is modified for scheduling using HPC. High performance can get than

other existing algorithm. Priority of small job is assigned to small node reduced waiting time than large node. On particular target HPC systems, job scheduling methods with acceptably thorough models for ability conflict, job interaction, and interference will be evaluated. Modelling and simulation is very important role for researcher, ongoing research are testing and simulation using simulator.

Acknowledgments

I would like to express my appreciation to my dedicated supervisor for their continuous guidance, my family for their constant support and encouragements.

Conflicts of Interest

The authors have no conflicts of interest to declare.

References

[1] Rodrigo, G. P.; Elmroth, E.; Ramakrishnan, L. (2018): ScSF: A Scheduling Simulation Framework, Springer, Springer International Publishing AG, part of Springer Nature 2018.
[2] Souza, A.; Rezaei, M.; Laure, E.; Tordsson, J. (2019): Hybrid Resource Management for HPC and Data Intensive Workloads,19th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGRID), 2019.
[3] Matsui, Y.; Watashiba, Y.; Date, S.; Yoshikawa, T.; Shimojo, S. (2020): Job Scheduling Simulator for Assisting the Mapping Configuration Between Queue and Computing Nodes, Springer nature Switzerland AG 2020, pp 1024-1033, 2020. https://doi.org/10.1007/978-3-030-15032-7_86 .
[4] Jones, W. M.; Ligon, W. B.; Pang, L. W.; Stanzione, D. (2005): Characterization of Bandwidth-Aware Meta-Scheduler for Co-Allocating Jobs Across Multiple Clusters, The Journal of Supercomputing 34(2):135-163, November 2005.
[5] Chung, M. T.; Pham, K. T.; Thoai, N.; Kranzlmueller, D. (2019): A New Approach for Scheduling Job with The Heterogeneity-aware Resource in HPC Systems, IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems, 2019.
[6] Fan, Y. (2021): Job Scheduling in High Performance Computing, 20 September,2021.
[7] Raqqaq, S.; Khan, F.; Wahid, A.; Sha, M. A.: Scheduling Algorithms for High-Performance Computing: An Application Perspective of Fog Computing, Springer Nature Switzerland AG 2019,2021.
[8] Amalarethinam, D.I. G.; Kavitha, S. (2016): Enhanced Min-Min Algorithm for Meta-task Scheduling in Cloud Computing, IJCTA, 9(27), pp 85-91, 2016.
[9] Yoon, J. W.; Hong, T. Y.;Park, C. Y.; Noh, S. Y.; Yu, H. C. (2020): Log Analysis-Based Resource and execution Time Improvement in HPC: A Case Study, Applied Science, 10 April,2020.
[10] Legrand; A.; Marchal, L., Casanova, H. (2003): Scheduling distributed applications: the SimGrid simulation framework, In Procedings of the third International Symposium on Cluster Computing and the Grid, pp 138-145, IEEE,2003.
[11] Dutot, P. F.; Mercier, M.; Poquet, M.; Richard, O. (2016): Batsim: a Realistic Language-Independent Resouces and Jobs Management Systems Simulator, In 20th Workshop on Job Scheduling Strategies for Parallel Processing, Chicago,United States, May 2016.
[12] https://www.guru99.com/priority-scheduling-program.html
[13] Buyya, R.; Murshed, M.; Abramson, D.; Venugopal, S. (2005): Scheduling parameter sweep applications on global Grids: a deadline and budget constrained cost-time optimization algorithm, International Journal of Software: Practice and Experience (SPE), 35(5):491-512,2005.
[14] https://github.com/aleasimulator/alea
[15] Dalibor, K.; Hana, R. (2010): Alea 2: Job Scheduling Simulator, ICST, ISBN 78-963-9799-87-5,2010.
[16] Chen, H. K.; Wang, F.; Helian, N.;Akanmu, G. (2013): Min-Min Scheduling Algorithm for Efficient Resource Distribution Using Cloud and Fog in Smart Buildings, Advances on Broadband and Wireless Computing, Communication and Applications (BWCCA 2018), 19 October 2018.
[17] http://www.e2matrix.com/blog/2018/01/22/job-scheduling-algorithms-in-cloud-computing/
[18] https://jsspp.org/workload/
[19] https://www.cs.huji.ac.il/labs/parallel/workload/logs.html

Authors Profile

Lett Yi Kyaw received the M.C.Sc. degree from the University of Computer Studies, Myeik, Myanmar in 2009. She is currently working as an Associate Professor in the University of Computer Studies, Taungoo, Myanmar. Her other research interests are Cloud Computing, High Performance Computing, Machine Learning and Deep Learning. She can be contacted at email: lettyikyaw@ucsy.edu.mm.

.

Kyar Nyo Aye received the Ph.D.(IT) degree from the University of Computer Studies, Yangon, Myanmar in 2013. She is currently working as an Associate Professor in the University of Computer Studies, Taunggyi, Myanmar. Her other research interests are Cloud Computing, High Performance Computing, Machine Learning and Deep Learning. She can be contacted at email: kyarnyoaye@ucsy.edu.mm.