Scheduling Methods in HPC System: Review

Lett Yi Kyaw
Cloud Computing Lab, UCSY
University of Computer Studies, Yangon
(UCSY)
Yangon, Myanmar
lettyikyaw@ucsy.edu.mm

Sabai Phyu
Cloud Computing Lab, UCSY
University of Computer Studies, Yangon
(UCSY)
Yangon, Myanmar
sabaiphyu@ucsy.edu.mm

Abstract

Parallel and distributed computing is a research area that is complex and rapidly evolving. Researchers are trying to get more and more methods and technologies in parallel and distributed computing. Most users who use Pay as you do go design, elasticity, and virtualization mobility and automation make cloud computing an attractive option to meet the development needs of some HPC users. Over the years, the research algorithm has also changed accordingly. The basic principles of parallel computing, however, remain the same, such as inter - process and inter - processor communication schemes, parallelism methods and performance model. In this paper, HPC system, scheduling methods and challenges are discussed and given some potential solutions.

Keywords**:** *parallel and distributed computing, high performance computing, methods, challenges*

I. INTRODUCTION

Scheduling technology is very important role in Information Era. Data input is processing in various ways and displaying. High Performance Computing (HPC) performs and demands many computers to work multiple tasks concurrently and efficiently. For a long time, the preparation of work for parallel computers has been subject to study. For a long time, the preparation of jobs since parallel computers has been subject to study. Job execution time is defined as the amount of processors allocated to it or requires users to provide estimated time for execution of jobs. The simulation software produces the estimation runtime attribute based on the specified range of estimation errors. The proposed method is to estimate and reduce execution time in a system.

Today, a super computer is and will always be a one-of - a-kind framework. Only a small number of users can use it. The operating mode can be compared to an experimental facility's exclusive use. A supercomputer does not usually have free resources. Typically the client has to wait, not the other way, to use a supercomputer program. During the past few years, supercomputers have been widely used in scientific research and industrial development. The architectures of such systems have varied during time. Today, to construct a real scientific grid, the main building blocks are mostly in place. The requisite communication quality is given by high speed wide area networks.

HPC system such as Grid computing [11], grid computing is a distributed network of large numbers of connected computers to solve a difficult problem. Servers and personal computers perform different functions in the grid computing model and are loosely connected to the internet and low-speed networks. Computers may connect through scheduling systems or directly. Grid computing involves dynamic digital organization (doctors, scientists and physicists), sharing of resources (reliable and unreliable collection of resources) and peer-to-peer computing. The grid also aims to provide access to computing power, scientific data resources and analytical facilities and is a grid computing challenge.

The key issue for high-performance computing is executing computational processes on a specific set of processors. Although the literature has also proposed a large number of schedules for heuristics, most of them target only homogeneous resources [19]. Future computing systems are most likely as the computational grid to be widespread and highly heterogeneous.

To achieve their performance goals, the development of research applications running on these systems has typically complicated new structural changes and new technology capabilities. Growing demand for new resources generation of HPC systems (computing nodes, memory, power, etc.) need to be supported by the ability to run more and larger applications with increased resource utilization as well as HPC system job turnaround time.

The traditional purpose of scheduling algorithms is as follows: find a mapping of processor tasks and order tasks to fulfill a task graph and a set of computer resources: (i) task precedence constraints are met; (ii) resource constraints are met; (iii) a minimum schedule is established. A divisible job is a job that randomly divides the cab into any number of processors in a linear fashion. It applies to a parallel job perfectly: every sub-task can be performed in parallel on any number of processors.

II. BACKGROUND

Parallel systems are valuable resources, such as supercomputers, which are widely shared by user groups.

A. Parallel Computing

The early standardization on a single machine device, the von Neumann computer, gave a lot to the rapid penetration of computer systems into commerce, research, and education. A von Neumann pc incorporates a principal processing unit (CPU) connected to a storage unit (memory). The CPU executes a saved application that specifies a chain of study and writes operations at the memory. This easy model has proved remarkably robust. Its endurance over more than forty years has allowed the look at of such important subjects as algorithms and programming languages to proceed to a big volume independently of developments in laptop architecture.

High performance computing investigates the parallel algorithm and strengthens the parallel computing architecture. Parallel computation could be a kind of calculation in which at one time different enlightenment is performed. The goal is to extend computation speed to possibly understand complex assignments of computing. There are two different loads and loads of parallelism refers to pipeline, whereas space parallelism involves numerous synchronous computing processors.

B. Scheduler

Scheduling is defined as a method through which a task, specified by some means, is assigned to resources required to carry out the specified task. The research can be virtual computing components such as threads, processes or data streams, which in turn are built on hardware resources such as processors, network connections or expansion cards. The scheduling in CPU is difficult to control others. Multiple processors have to be scheduled in parallel computing and to manage the resources for all processors any overlapping of the resources to produce any conflicting results are needed. So the scheduling in multiprocessors is more complex than scheduling in a single processor unit. In HPC system, parallel computing is using to get high performance. In scheduling of multiple processors it should be ensured that any processor should not be overloaded and any processor should not be under loaded. There will be multiple processors; there will be multiple queues, so there is need of scheduling multiple queues simultaneously.

TABLE I. TYPES OF HPC SCHEDULER

Scheduler Name	Description
Slurm[9]	One of the HPC scheduler, a highly scalable, fault-tolerant resource cluster manager and massive computer systems job scheduler. A commonly used plugin-based job scheduler that provides several optimization options, either by adjusting the configuration parameters or by implementing new plugins and new scheduling and policy collection.
LSF[3]	It is a tool for task management, a job scheduler for high-performance distributed computing. It also allows users to access vast amounts of computing resources distributed around the network in large, heterogeneous distributed systems and can provide major performance upgrades to applications.
Loadlever	It schedules jobs and offers functions for the faster and more efficient production, submission and processing the jobs in a dynamic environment [21].
Moab [22]	A method of task planning and management for use on clusters, supercomputers, and grids. It can support a wide range of policy planning and fairness dynamic priorities, and substantial reservations.
Torque	It is an open source resource manager [23] that enables batch job control and distributed compute nodes.

The primary purpose of the job scheduler is to assign resources to work for users and ensure that jobs are operating at their highest performance. It prevents the overloading of a given compute node and puts jobs on hold before resources are available.

The scheduler's secondary purpose is to monitor usage to ensure fair distribution of HPC resources over time. To run multiple applications concurrently, HPC schedulers order to execution of batch jobs to achieve high utilization while controlling their turnaround times. In HPC schedulers, the balance between utilization and turnaround time is controlled by the scheduler prioritization system and the scheduling algorithms.

Turnaround Time: *Time from submission to completion of process*

Production HPC system uses different workload manager that combine scheduling and resource management. The following table 1 is most popular schedulers.

Modern HPC tools usually consist of a cluster of computing nodes that provide the user with the ability to coordinate tasks and substantially reduce the time it takes for complex operations to be performed. Usually, a node is defined as a discrete unit of a computer system running its own operating system case. Modern nodes have several processors, also referred to as Central Processing Units or CPUs, each of which contains multiple cores that can process a separate instruction stream.

All things considered, in spite of which strategy is used, the idea of a high-performance system is stable. The management of a high-performance system (referred to as part of a single gadget or multi-computer cluster by multiple processors) is treated as a single computing commodity, placing demands on unmistakable hubs. The HPC framework could be a partitioned unit created and actualized unequivocally as an effective computing device.

C. Time-sharing and Space-sharing

Time sharing refers to any scheduling approach that allows others to preempt and restart threads later during execution. The amount of jobs that each processor can perform at the equal time is defining as the level of multi-programming. Future HPC systems will be much noisier, with much greater competition and heterogeneity, requiring the use of new, asynchronous programming models [5].

Space-sharing procedures only offer a string that uses a processor until its execution is complete or the most extreme period of time has been reached and the string is done. Space sharing techniques [14] control time by putting that work in a line and at the same time running all its strings when discharged from that line.

D. HPC Scheduling

Parallel computing has become most important issue right this time but because of the high cost of computer it is not accessible for everyone. Cluster is only technique that provides parallel computing, scalability and high availability at low cost, in fig 1. Cluster collection provides high-performance computing, while individual computers work to solve a problem at the same time. Clusters are designed as they provide computation and availability of high performance over a single computer. A cluster is a group of connected devices, such as computers and switches that function as a single system together. Each and every node of a cluster is associated with a single system. Each and every cluster node is either connected by wire (Ethernet) or wireless that transfers data between nodes. A strong cluster provides distributed computing consisting of standard desktop computers linked through a network like Ethernet. Linux operating system can be used to control the cluster. In this way, we can build high performance computing (HPC) at low-cost price.

Scheduling method is mainly divided into two types: static and dynamic. With static scheduling, the consideration concerning processor placement and task assignment is made at the onset of the job execution. Static policies have low run-time overhead and the scheduling costs are paid only once, during the compilation time, for instance. As a consequence, static scheduling policies can afford to be based on sophisticated heuristics that lead to efficient allocations. Parallelism can be well exploited by spreading different tasks over different processors statically. Static scheduling needs to know in advance detailed information about the job's run-time profile which is relied on the input data, static scheduling carries some degree of unsure. The consequence may be an unbalanced load resulting in longer parallel execution time and low utilization of the network.

Dynamic scheduling, during execution, the number of processors assigned to a job can vary. The allocated processors are also assigned tasks during the execution of a job. In both their advantages and disadvantages, dynamic scheduling policies complement static policies. Typically, complex policies generate high overhead run-time, which can lead to performance degradation. But there are dynamic scheduling processes and adaptive behavior, which results in a high degree of load balancing.

HPC scheduling research [7] relates to how such research should be performed, because such research requires information, methodology, and resources that are not always available. Some of three basic methods for job scheduling research:

- Theoretical analysis
- Real system experiments
- Simulation

Theoretical Analysis

By defining boundary cases reflecting the best and worst-case results, algorithm behavior is analyzed. This approach may provide insight into the actions of the algorithm, but may not reflect the performance anticipated when performing a real workload. In HPC method, percentage of execution time spent in inter-process communication, congestion of memory bandwidth and performance of FLOPs are evaluating various ways to determine the degree of performance degradation when running practical HPC workloads.

Real System Experiments

Simple measurements of the behavior of the algorithm with a real system and workload still involve several different experiments [3] to be performed, which can be difficult for several reasons. However, a single experiment only produces evidence for one workload and process state, which is usually not enough to rationalize the general case or test a general hypothesis. Eventually, the conditions of workload are difficult to control, so evaluating specific scenarios can be challenging.

Simulation

A process is emulated, the emulated system runs a batch scheduler, and the scheduler receives a synthetic workload. This method allows the development of various experimental scenarios and the execution of large-scale experiments to produce information that enable general conclusions to be induced. Nevertheless, their findings are only true if they are indicative of the recreated workloads, processes and scheduling behaviors.

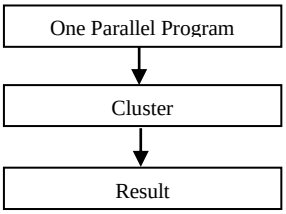

Figure 1. A cluster used to run a parallel program

No interaction with the administrator is needed through the usage of HPC program, jobs are completed sequentially without delays created by human interaction, which saves processing time this is generally wasted with human interaction. An application usually uses a parallel algorithm to run on a high performance cluster. A big task can be divided into several sub-duties, running on cluster-based exclusive nodes. The knowledge collected, resulting from the sub-duties, is translated into the special challenge's quit end result.

E. Need of Job Scheduling

Through a planning system responsible for defining available resources, a task is allocated to tools: matching job requirements with resources, making work ordering decisions and goals. Usually, the use of HPC resources is strong. In this manner, assets are not quickly accessible and employments are lined for future execution time until execution is regularly very long (numerous generation frameworks have and normal delay until execution of >1h), employments may run for a long time (a few hours, days or weeks).

Job Steps

A user job enters a job queue, the scheduler (its technique) chooses on begin time and asset properties of the work. The occupations can come at the shape of batch or interactive type.

- A batch job is a script is used to submit the job. A shell script may be a given list of shell commands to be executed in a given arrange.
- Today's shells are so modern simply can make scripts that are genuine programs with a few factors, control structures, circles.
- An interactive job is typically an assignment of one or more nodes where one of the cluster nodes receives a shell from the user.

Job scheduling is a mapping mechanism from the tasks of users to the proper selection and execution of resources [1].A scheduler may aim at one or more goals, for example: minimizing throughput, minimizing wait time, minimizing latency or response time or maximizing fairness [2].The following job scheduling algorithms are: First come first serve (FCFS), Shortest first job (SJF), Round - robin (RR), Priority scheduling, Min - min, Max - min, Genetic algorithm (GA) and so on which are using for job management.

A good scheduling algorithm allocates suitable resources to workflow tasks in order to complete the application and at the same time satisfy user requirements.

Job Scheduling Algorithms

Genetic Algorithm: A method for solving constrained and unconstrained problems of optimization foundation on natural selection, the mechanism that evolved biological process. A population of individual solutions is repeatedly updated by the genetic algorithm. The genetic algorithm selects individuals from the current population at random to be parents at each stage and uses them to produce the babies for the next generation. The genetic algorithm can be used to solve a variety of optimization issues that are not suitable for conventional optimization algorithms where objective function is discontinuous, non-differentiable, stochastic, or highly nonlinear. By using GA, each iteration generates a population of points and selects an optimal solution for the best point in the population. And then, by calculation, select the next population using random number generators.

GA is used in a number of different fields of usage. Hence, by providing arrays of bits or characters to represent the chromosomes, most researchers can introduce this genetic model of computation. Simple operations of bit manipulation allow crossover, mutation and other operations to be performed.

First come first serve (FCFS): FCFS is similar to the data structure of the FIFO (First in First out) queue, where the first data element added to the queue is the first element to exit the queue. In Batch Systems, this is used. Using a queue data structure, where a new process works through the queue's tail, it is easy to programmatically understand and execute, and the scheduler selects the process from the queue end. Purchasing tickets at the ticket counter is a perfect example of FCFS planning in real life. But there is an issue with FCFS, system priority is not taken into account. Parallel use of resources is not feasible and use of bad resources (CPU, I / O, etc.).

Shortest Job First (SJF): A scheduling strategy that chooses the process of waiting to execute next with the smallest execution time. Second, the system is completed, which takes the shortest time to complete the execution. Using ordered FIFO queue, this rule can be enforced. All processes in a queue are sorted on the basis of their CPU bursts. When the CPU is available, a system will be selected to run from the first position in a queue [16].

Round - robin (RR): When the process can be executed, procedures are given an equal slice of time. The

execution is carried out one after another on a circular order. So every work has a quantity when it can be done. If this quantity is not sufficient to complete the execution of the process, it will be stopped and the next process will be performed. Upon completing a full round, its turn will come again and so on. If a process is completed, it will go off the list and if another arrives, it will be put at the end of the list waiting for its turn.

There is no hunger in this algorithm, but it can often be too long. This is a typical and conventional load balancing algorithm, but the challenge in round robin is setting the time quantity. To calculate the dynamic quantity of time, various optimization algorithms can be combined with RR [4].

Priority scheduling: It gives a well-defined priority to each system. This way, each process has its own priority, depending on whether it will be run or wait. The first to run is going to be the highest priority operation, while others are going to wait for their turn. First will be handled high-priority activities [20]. Therefore, the low priority tasks would have to wait a long time in the queue. If the system crashes, all non-performing low-priority tasks will be lost.

Min – Min: This algorithm is based on the resource assignment principle that has the shortest completion time (fastest resource), a minimum completion time (MCT) functioned [8]. It is a sample algorithm but it gives the quick result when the size of the task in the task group that deals with other tasks is small compared to the large size task, on the other hand, when large size tasks are performed, it gives poor use of the resource and large maximum completion time of the task since larger tasks have to wait for smaller tasks to be completed. For all tasks to be performed, the Min - Min algorithm first finds the minimum time. Then, it selects the task with the least execution time among all tasks. The algorithm proceeds by assigning the task to the resource generating the minimum completion time. Min - Min repeats the same procedure until all tasks are planned. Min - Min algorithm's limitation is that it first selects smaller tasks that make use of high computational power resource. As a result, when the number of smaller tasks exceeds the large ones, Min-Min's schedule is not optimal.

Max – Min: This algorithm is bottomed on the concept of assigning to the asset, which has the maximum completion time (fastest resource), a task with maximum completion time (MCT). It is a test algorithm but it gives the fast result when the size of the task in the metatask is big compared to the small size task. On the other hand, if small size tasks are performed, it gives poor use of the resource and a large maximum completion time of the task as smaller tasks have to wait for larger tasks to be completed. The purpose of this algorithm is to prioritize tasks with optimum completion time by executing them first before assigning other tasks with minimum completion time [13].

F. Components of HPC Scheduling

High-performance computing has five components: CPUs, memory, nodes, internodes in the network, and storage (disks, tape). Single-core CPU (processors) is no longer being used today. To date, all CPUs (processors) consist of the configuration used on the motherboard (multiple' cores' on a single' chip'). For a number of reasons, the trend of even more ' core ' per unit will rise. The node plays a major role in linking CPUs, memory, interfaces, computers and other nodes in a physical way. For a high-performance computing system, shared memory is often necessary.

There are five different node types: user node, control node, node of management, node of processing and node of computation, in Fig 2. The client node is the only portal to reach the cluster network for outsiders. Users typically have to log in to compile and run the tasks from the node. Fault-tolerant architecture is accomplished with hardware redundancy to be built in the system to ensure the client node is highly accessible. Control node is mainly responsible for delivering computer node to basic network services such as DHCP (Dynamic Host Control Protocol), DNS (Domain Name Service), NFS (Network File Service) and the distribution of computer node tasks.

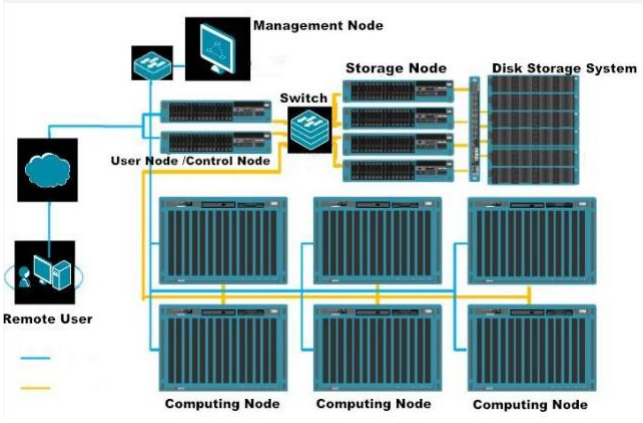

Figure 2. A High Performance Computing Cluster

III. CHALLENGES

In recent study, as the chosen platform for execution as the optimal medium for a wide range of workloads to be performed, there are four major challenges for heterogeneous computing clusters to address.

First, the majority of business applications currently in use were not designed to run on such large, open and heterogeneous computing clusters. To move these applications to heterogeneous computer clusters, particularly with significant performance improvements or

improvements in energy efficiency, is a quest and attempt to solve problem. Second, it is also overwhelming to generate new, ground-up enterprise applications to run on the different, heterogeneous computing framework. It is extremely challenging to write high-performance, energy-efficient programs [12]for these architectures to die on an unprecedented scale of parallelism, as well as difficulty in scheduling, interconnecting and sorting. Third, cost savings for utilization and distribution of IT services from the new shared network architecture are only feasible if multiple business applications can share resources in the heterogeneous computing cluster. Nonetheless, allowing multi-tenancy without adversely affecting that application's stringent performance service metrics needs complex scalability and virtualization of a wide variety of different computing, processing and interconnect devices, and this is yet another unresolved issue. Fourth, enterprise systems with unusually heavy load spikes experience greatly variable user loads. Meeting service metrics quality across multiple loads includes the use of an application's elastic computing resources in response to increasing user demand. There are no good solutions to this problem at the moment. There are no good solutions to this obstacle at the moment. The resource management dynamics in HPC are evolving. New application innovations and technical transitions introduce the scheduling models and process architecture with new concepts and specifications.

Another problem is that multicore chips are relatively simple processor core and will be underused if user programs are unable to provide enough parallel thread speed. It is the responsible of the programmer to write parallel high-performance software to make maximum use of the processor core. The new parallel multicore technology should have two characteristics to achieve high performance:

Fine grain thread: a high degree of parallelism is needed to keep each core processor busy. Another factor is that a core often has to operate on a small-size cache or scratch buffer that allows developers to break down a task into smaller tasks.

Asynchronous program execution: The existence of a synchronization point can seriously affect the performance of the program when there are many processor cores. And reducing unnecessary points of synchronization can subsequently increase the degree of parallelism.

IV. CONCLUSION

We present HPC framework, its scheduling and research challenges in this paper as express in table 2. Task scheduling on parallel machines is a research field that has been well explored and has led to widespread use and there are several methods, including automatic calculations of runtime, partial executions, and smarter allocation schemes for processors. Parallel processing builds on existing technologies implemented in new scenarios and serving diverse users and priorities. Max-min algorithms is the execution of tasks with maximum completion time may first increase the total response time of the system and thus make it inefficient and it has these drawbacks so that researchers may consider the different methods to implement this algorithm. Min-min algorithm is first considered to be the shortest jobs, so it fails to efficiently leverage the resources that lead to a load imbalance, so efficient methods are needed to solve this problem. FCFS which can be a long waiting time if the short request is waiting for the long process [15]. It is not ideal for a system of time sharing where it is critical that each user receives the CPU for the same time interval.

The through in round robin largely depends on the selection of the size of the time quantum. If time quantum is too large it behaves as FCFS. If time quantum is too short much of the time is spent in process switching and hence low throughput and one cannot assign priority to any process which can be a drawback. Priority scheduling will have to consider which parameters are set to high priority within system.

Genetic algorithm has different parameters (size of population, times, rates of mutation, etc.) and there are many problems not well defined (parameters are not well defined or not known).Researchers are need to decide which optimization method to use for research environment to solve problem. Longer processes in shortest job first algorithm have more waiting time, so starvation occurred and need to solve this issue. Some are used average waiting time to reduce long waiting time. We agree that several areas of HPC still need the attention of the researcher. We will have a detailed study of HPC problems and challenges in the future.

TABLE II. COMPARISON OF EACH METHOD

Algorithm	Comparison	Advantages	Disadvantages
Genetic Algorithm (GA)[18]	It is necessary to use as a tool to solve the problems of optimization and search. GA includes several operations to execute the algorithm with several techniques.	This combines good solutions and optimizes towards a goal over time.	Without through too much time for GA to check for optimum results, some new genetic values should be incorporated into the population.
First Come First Serve (FCFS)[15]	FCFS starts imply the first task to be done.	For long process, FCFS is suitable than others and one of the easiest methods.	Each and every small process should wait for its turn to utilize the CPU also, throughput is not emphasized.
Shortest Job First (SJF)[16]	Shortest job starts to be process.	It is optimized for average waiting time.	There is more number of short jobs in a system, the long jobs will be starvation.

Round-Robin (RR) [10]	Time is to be allocated to resources in a time slice manner in this scheduling algorithm.	In a general-purpose, time-sharing or transaction processing system, RR is successful. There is also a small overhead on the processor.	Care must be taken when determining the quantity value, and if the quantity time is too high, the result is also weak.
Priority scheduling	A well-defined priority to each system.	A handled high-priority activity, so user defines process is more suitable.	The low priority tasks would have to wait a long time in the queue of the system.
Min-min [6]	Chooses smaller tasks first to be done.	One resource can execute only one at a time and resources are known in prior.	This is caused load imbalance between large task and small task to execute in the system.
Max-min [6]	Chooses bigger tasks to be completed first.	Prioritize tasks with optimum completion time by executing them first before assigning other tasks with minimum completion time.	Execution of task with maximum completion time first might increase the total response time of the system thus making it inefficient.

REFERENCES

[1] W. Bradley, S. Ramaswamy, R.B.Lenin and D. Hoffman, "Modelling and Simulation of HPC Systems Through Job Scheduling Analysis", January 2011.

[2] M. A. Obaida, J. Liu, "Simulation of HPC Job Scheduling And Large-Scale Parallel Workloads", 978-1-5386-3428-8,IEEE,2017.

[3] IBM LSF & HPC User Group @SC18, LSF&HPC User Group/SC18/@2018 IBM Corporation .

[4] N. K.C Das, M. S. George and P. Jaya, "Incorporating weighed Round Robin in Honeybee Algorithm for Enhanced Load Balancing in Cloud Environment", 978-1-5090-3800-8, IEEE,2017.

[5] S. Hofmeyr,C. Iancu, J. A. Colmenares, E. Roman and B. Austin," Time-Sharing Redux for Large-scale HPC systems", 978-1-5090-4297-5, IEEE,2016.

[6] N. Thakkar,.R. Nath,"Performance Analysis of Min-Min, Max-Min and Artificial Bee Colony Load Balancing Algorithm in Cloud Computing", IJACS,ISSN 2347-8616, Volume7, Issue 4, April 2018.

[7] G. P.R. Alvarez, P. Ostberg, E. Elmroth, K. Antypas, R. Gerber, and L.Ramakrishnan, "Towards Understanding Job Heterogeneity in HPC: A NERSC Case Study", In Proceeding of the 16th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid'16), IEEE/ACM 2016.

[8] G. Sharma, P. Bansal,"Min-Min Approach for Scheduling in Grid Environment", IJLTET, ISSN: 2278-621X, Vol.1, Issue 1, May 2012.

[9] A. Jokanovic,M. D'Amico,J. Corbalan,"Evaluating SLUM Simulator with real-machine SLUM and vice versa",https://www.researchgate.net/publication/328886964, November 2018.

[10] B. Mohamed, N. E.AL-Attar, W. Awad, F. A. Omara, "Dynamic Job Scheduling Algorithms Based on Round Robin for Cloud Environment", Research Journal of Applied Sciences, Engineering and Technology 14(3): 124-131, 2017.

[11] F. Xhafa, A. Abraham,"Computational models and heuristic methods for Grid scheduling problems", Future Generation Computer Systems 26 (2010) 608_621, doi:10.1016/j.future.2009.11.005, Elsevier, 2009.

[12] S. Wallace, X.Yang, V. Vishwanath, W. E. Allcock, S. Coghlan, M. E. Papka, Z. Lan,"A Data Driven Scheduling Approach for Power Management on HPC Systems", 978-1-4673-8815-3,IEEE,2016.

[13] C. Diaz, J. E. Pecero, P. Bouvry, "Scalable, low complexity, and fast greedy scheduling heuristics for highly heterogeneous distributed computing systems", Journal of Supercomputing 67(3), 837–853(2014),2014.

[14] T. Li, V. K.Narayana, T. EI-Ghazawi,"Exploring Graphics Processing Unit(GPU) Resource Sharing Efficiency for High Performance Computing", https://www.researchgate.net/publication/260422348,November,2013.

[15] A. P. U. Siahaan,"Comparison Analysis of CPU Scheduling: FCFS, SJF and Round Robin", IJEDR,Volume 4,Issue 3, ISSN: 2321-9939,2016.

[16] M. A.Alworafi, A. Dhari, A. A. Al-Hashmi, "An Improved SJF Scheduling Algorithm in Cloud Computing Environment", ,International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT), , 978-1-5090-4697-3,IEEE,2016.

[17] L. Kishor, D. Goyal, "Comparative Analysis of Various Scheduling Algorithms", International Journal of Advanced Research in Computer Engineering& Technology (IJARCET), Volume 2, Issue4,April 2013.

[18] S. P. Lim, H. Haron,"Performance Comparison of Genetic Algorithm, Differential Evolution and Particle Swarm Optimization Towards Benchmark Functions", Conference on Open Systems (ICOS), 978-1-4799-0285-9,IEEE,December2-3,2014.

[19] S. Razzaq, A. Wahid, F. Khan, N. u. Amin, M. A. Shah, A. Akhunzada, I. Ali ,"Scheduling Algorithms for High-Performance Computing: An Application Perspective of Fog Computing" , https://doi.org/10.1007/978-3-319-99966-1_10,Springer,2019.

[20] https://en.wikipedia.org/wiki/Scheduling_(computing)#Scheduling_disciplines .

[21] https://www.ibm.com/support/knowledgecenter/en/SSFJTW_5.1.0/com.ibm.cluster.loadl.v5r1.load100.doc/am2ug_ch1.htm

[22] https://computing.llnl.gov/tutorials/moab/, https://computing.llnl.gov/tutorials/dataheroes/moab/

[23] https://en.wikipedia.org/wiki/TORQUE,https://hpc-wiki.info/hpc/Torque