CHAPTER 3
JOB SCHEDULING IN CLOUD COMPUTING

3.1 OVERVIEW

This Chapter highlights job scheduling concept and taxonomies that lie at the core of cloud computing environment. Detailed discussion covering the of purpose job scheduling with its taxonomies followed by job scheduling algorithms in cloud computing is carried out in this Chapter. The study and performance analysis of various existing scheduling algorithms in terms of energy efficiency in successfully achieved in this Chapter.

3.2 INTRODUCTION

There are wide varieties of job scheduling strategies or algorithms that are available to allocate the jobs to available resources. These algorithms were formulated based on some various methods. As a result, it seems difficult to admire all competencies of numerous jobs scheduling algorithms and asses their performance quantitatively. The scheduling is broadly classified as local and global scheduling. Algorithm which works with the job allocation and execution that resides on single resource is local scheduling while global scheduling policy make use of the information about multiple available resources as in cloud computing to assigns jobs to multiple resources optimize performance objective. Two Heuristics based algorithms i.e. Efficient MQS (Efficient Multi Queue Scheduling) and ACO (Ant Colony Optimization) are implemented and the results of both of them are compared on the parameters of energy consumption and time consumed. The results obtained have been validated through simulation of different scenarios. The Hierarchical taxonomy of scheduling algorithms is shown below in the Figure 3.1

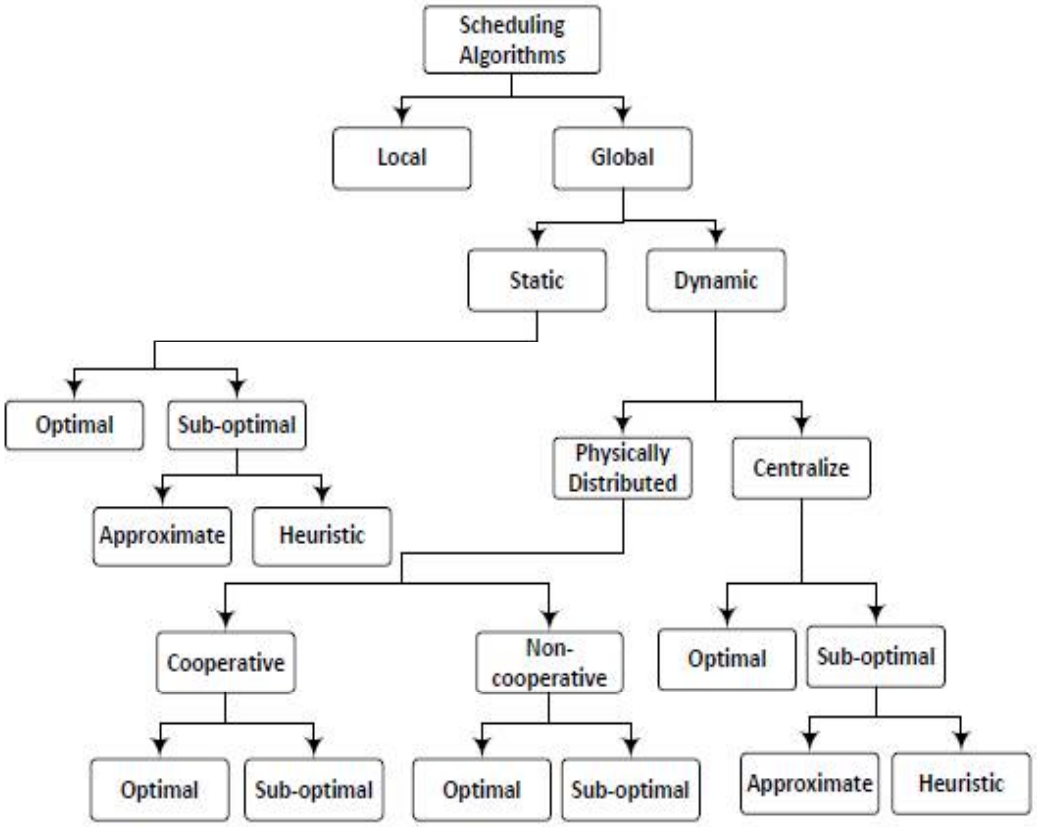

Figure 3.1 Hierarchical Taxonomy of Scheduling Algorithms [133]

3.3 NEED OF SCHEDULING IN CLOUD COMPUTING

One of the major advantages of switching to the clouds is the scalability ability of the applications. Contrary to the grids, the ability to scale the cloud resources allows their real-time provisioning so as to meet the application constraints. Cloud services would be accessible at a much lower cost. Typically, tasks are scheduled according to the user requirements. New scheduling policies are required to overcome several complications put forward by the network properties between the consumer and the resources. Present day scheduling policies work at combining conventional scheduling policies as well as network aware policies to offer an improved and much more effective job scheduling framework [52].

Earlier, scheduling algorithms were executed in the grids but due to decreased performance, there was a requirement to realized scheduling algorithms in the cloud itself. It would facilitate the workflow management system and solve the Quality of Service issue whereas the conventional approaches still need the advance resource

reservation. Cloud services are accessible at a lower cost, just like resources like bandwidth, and storage. It has to be noted that cloud applications usually require difficult and complicated execution environments. In the case of cloud computing, a user may need to handle a number of virtualized resources; hence, it's not possible to manually assign the jobs. Such environments are tough to produce on the grid resources. Scheduling is an important feature in the field of cloud computing and need to accomplish competent assigning of the resources [52] [53].

3.4 AIM OF JOB SCHEDULING IN CLOUD COMPUTING

A lot of factors like user needs, type of system etc have to be considered while designing a scheduling algorithm. Avoiding indefinite starvation i.e. a process must not wait indefinitely during the process service. Minimizing overhead as overhead causes wastage of the resources and if overhead is minimized, the overall performance of the system improves a lot.

Enforcing priorities i.e. if a system assigns any priorities to the process, the algorithm must process the process with highest priority first. Achieving balance between response and utilization so that all the resources of the system are busy. Depending on the type of system, a user may expect the following things from the scheduler [77]:

(a) **Quality of Service:** The main aim of the cloud is to offer computing and storage services to the user, and meet their resource demand. The performance of the resources supplied by the provider is judged through quality of service. When it is about task allocation, it's a must to guarantee the quality of service of the resource [77].

(b) **Load Balance:** Job scheduling and load balancing are very closely related in the cloud computing environment. Task scheduling is responsible for proper matching of the tasks and the resources. Due to this pertinence of the mentioned, load balancing becomes of great importance. Load balancing states the level two loads in the task scheduling - virtual machine load stage and resource layer load stage [50].

(c) **Best running time:** Tasks are divided into various categories according to the user requirements and then, a best running time is set according to different goals of each task. It indirectly improves the QoS [50].

(d) **Economic Principles:** Cloud computing resources are widely distributed among different organizations all around the world. Each organization has its own management policies which are greatly affected by the location of the resources as well. As a business mode, cloud computing offers relevant services using these resources according to different user requirements, making demand charges very reasonable. The market economy is the key driver for task scheduling as well as resource managed and we must ensure that both - consumer and provider must benefit from the cloud services so that the technology as a whole can move forward [50].

3.5 THE SCHEDULING MODEL IN THE CLOUD DATA CENTERS

The scheduler selects from among the submitted cloud user jobs from the queues that are ready to execute and maps them to the virtual machines for processing .At a same time cloud scheduler can allocate multiple user jobs to multiple virtual machines in an optimal manner. Job scheduling can be performed based on distinct parameters in different ways. User jobs can be statically allocated to various resources at compile time or can be dynamically allocated at runtime [33].The scheduler working in cloud is shown below in Figure 3.2

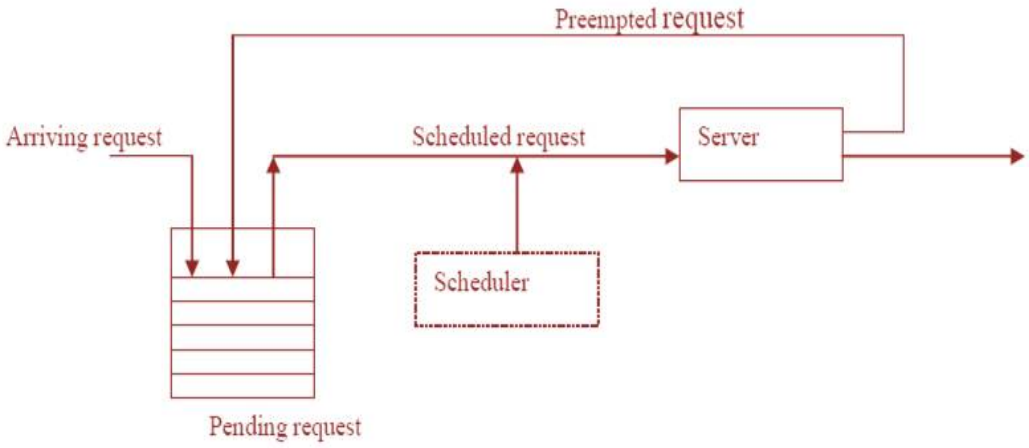

Figure 3.2 Scheduler Working in Cloud [121]

Scheduling process in the cloud computing environment has several basic components like computing entity, job scheduler, job waiting queue and job arriving process. The Figure 3.3 shows all these entities involved

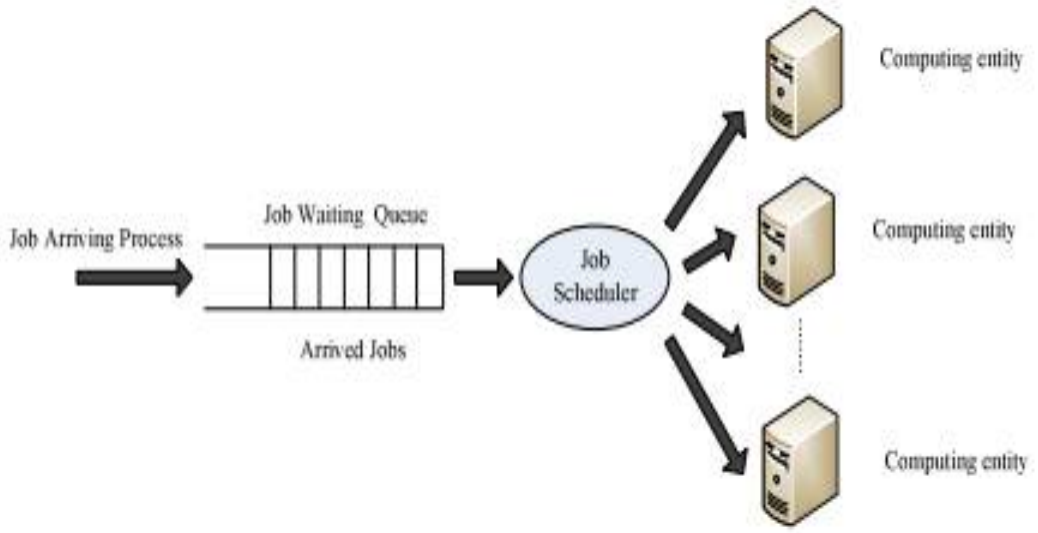

Figure 3.3 Essential Components of Job Scheduling in Cloud [33]

The scheduling process in the cloud computing scenario involves the following [33]:

(a) **Job Arriving**: It's the process in which the jobs enter or reach the scheduling system [33].

(b) **Computing Entity**: Provided through the implementation of virtualization in the cloud computing system, a number of VMs exist in the cloud system to provide all the computing resources and process the submitted jobs. Every computing entity is characterized by the computing capacity, which is determined by the number of tasks the entity can handle per second [33].

(c) **Job Waiting Queue**: Job Waiting Queue is the queue in which jobs wait until they get assigned to a particular machine for the execution [33].

(d) **Job Scheduler**: Job scheduler is a very important component of the scheduling process as it determines the order of execution of various jobs that are waiting in the job waiting queue [33].

3.6 FEATURES OF JOB SCHEDULING

In cloud computing system, job scheduling gets more complex due to the transparency and the dynamic flexibility. The strategies that focus just on efficiency will lead to higher costs and increase throughput and quality of service at the same time. Job scheduling has following characteristics in the cloud environment [49] [50]:

(a) **Catering to a Unified Resources Platform:** Due to the use of virtualized technology, we can abstract the underlying physical resources such as hosts, workstations and computers etc. into a unified resource pool and supply the use of resources in the form of a data center.

(b) Global Centralization: The technology of virtualization as well as the mirroring services has enabled task scheduling in cloud computing system to be globally centralized scheduling. It is because, cloud computing is already built around a model that supply a centralized resource by mirroring service to more than one distributed apps. This mirroring deployment makes the execution of the mirroring procedure much easier than before, which is what lead to global centralized scheduling.

(c) Independent Nodes in the Cloud: In the cloud computing system, each need is independent because the internal scheduling of each node is autonomous and schedulers do not interfere with the scheduling system of these nodes.

(d) Scalability: The scale of resources is usually limited in the initial stages. With the addition of more computing resources, the size of abstract virtual resources may come out to be large and the demand of application keeps increasing. Task scheduling has to meet the scalability features to not let the throughput get decreased

(e) **Dynamically Self-Adaptive:** Job scheduling has to be self-adaptive as expansion and shrinking of applications can be a necessary requirement at times. Even the virtual computing resources can expand and shrink at times. All the resources are changing constantly; some fail and some join up in the clouds.

3.7 TYPES OF JOB SCHEDULING

To enhance the cloud performance, effective job scheduling is required. There are various scheduling algorithms which can be classified as follows:

(a) **Dynamic & Static Scheduling:** Static job scheduling involves pre-scheduled jobs. It is assumed that the information about all the tasks is already available at the time of scheduling of the application as well as resources are also available with no resource failures. Dynamic job scheduling involves jobs that are

dynamically available to the scheduler. Scheduler doesn't need to determine the run-time in advance. It also considers resource failures, unlike static scheduling, and job priority. If a resource fails, the execution terminates. In such a case, the scheduler migrates the jobs to some other resource. If a higher-priority job centers in the system, the resource assignment is done immediately by the scheduler. In case the required resource isn't free, it terminates the job under execution and assigns the now freed up resource to the higher priority task. Thus, workload on the resources can vary significantly over time [34].

(b) **Centralized, Distributed & Hierarchical Scheduling:** There is a difference in the control of the resources as well as the knowledge of the overall system in the centralized and decentralized scheduling. The former has more knowledge of the system and has more control over the resources. It also monitors the state of the resources continuously. The main advantage of centralized scheduling is the ease of implementation, efficiency, proper monitoring of the resources and higher control over resources. The disadvantages to be mentioned are lack of efficient performance, scalability and fault tolerance [34].

(c) **Pre-Emptive and Non-Preemptive Scheduling**: In the pre-emptive scheduling, a job can be interrupted in the middle of its execution and it can be to move to some other resources; while the earlier resources in use are left unused and available to take up other jobs. The priority of the job is taken take into consideration, making it more helpful whereas non-preemptive scheduling does not do anything such thing and keeps the resource until the execution of the current job is finished [34].

(d) **Independent & Workflow Scheduling:** As evident from the name, tasks are handled independently in the independent scheduling whereas in workflow scheduling, there is dependency between various tasks. Dependency refers to the fact that there's a precedence order that exists between the tasks. A child task has to wait for the completion of execution of the parent task before it starts its execution. An overview of workflow scheduling system is schematically illustrated in Figure 3.4 as shown below

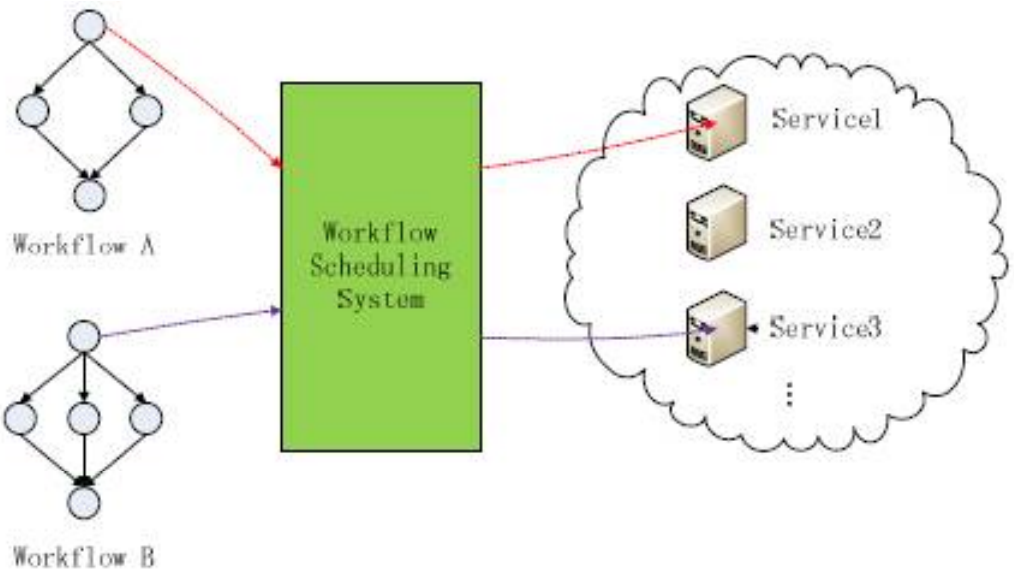

Figure 3.4 Workflow Scheduling System [129]

Users first submit workflow with their QoS requirements. The system then allocates appropriate services for processing the workflow tasks and schedules the tasks on the services according to the QoS requirements. Workflow can be represented using DAG (Directed Acyclic Graph) notation. Each task can be started only after the preceding task in the workflow represented in Directed Acyclic Graph (DAG) is finished [34]. .

3.8 JOB SCHEDULING CRITERIA

There are several scheduling algorithms with each having different properties. The choice of algorithm decides which processes may be favored and which can be spared. To select an algorithm for a particular situation, properties of each of the algorithms available must be considered. Following constitute the scheduling criteria [77]:

(a) **Context Switch:** Context switch is the process of storage and restore of the state of a preempted process. It is done so that the execution of that process can be resumed from the same point at which it was preempted. But context switch usually leads to waste of time and memory, computationally intensive and puts a lot of overhead on the scheduler. Hence the design of a scheduler has to optimize these switches for better performance [78].

(b) **Throughput:** Throughput is defined as the number of process that is completed in one unit of time i.e. number of tasks completed per second. Throughput and the context switching are inversely proportional to each other [78] [79].

(c) **CPU Utilization:** CPU utilization refers to how busy the CPU is at any given point of time. The goal is usually to maximize CPU utilization. It can range from 0 to 100% [79].

(d) **Turnaround Time:** Turnaround time is the total time spent on completing a process i.e. time it takes to complete a process. Total around time is the sum total of time spend while waiting in the ready queue, time spent while waiting for processing and the execution as well as Input/ Output time[77][79].

(e) **Waiting Time:** Waiting time is the total time taken by a process while waiting in the ready queue. It is noteworthy that a scheduling algorithm does not affect the time during which a process is executed but it affects only the time that a process spends while waiting in the ready queue [77].

(f) **Response Time:** Response time is considered because turnaround time may not always be the best measure especially in an interactive system as in such a system, an early output maybe produced and while the earlier output is still being produced, some new results can also come up. Thus the response time is the time between a request submission and the first response. It must be low for optimum scheduling [77].

(g) **Fairness:** Fairness means that a scheduler must allocate a fair amount of resources to all the tasks in the system in the absence of any user or provider criteria [79].

3.9 JOB SCHEDULING PARAMETERS

The main scheduling parameters that are considered in the above-mentioned methods are listed as follows [33]:

(a) **Deadline**: Deadline the period of time between the submission of a task and by when it must be completed. A good scheduling algorithm always aims to finish a task before the deadline.

(b) **Make span:** Make span is the total of completion of all the tasks in the queue. A good algorithm always tries to have minimal make span [33].

(c) **Execution Time:** Execution Time is the time taken to execute a task. A good scheduling algorithm aims to minimize the execution time. In cloud environment scenario, it can be considered as the time taken to execute cloud user's request and provide services [133].

(d) **Completion Time:** Completion time is the total time taken to finish a job. It's the sum total of execution time and any delay caused in the cloud system. Again, a good scheduling algorithm aims to minimize completion time [33].

(e) **Performance:** Performance indicates the overall efficiency of a scheduling algorithm. A good scheduling algorithm considers performance at both the user end as well as at the cloud service provider end [33].

(f) **Energy Consumption:** Energy Consumption is one of the biggest issues faced by cloud service providers. Energy efficient data centers are also known by the name green data centers are the main center of attractions for most researchers in present day scenario Algorithms are expected to minimize power consumption and improve the performance [33] [133].

(g) **Quality of Service:** Quality of service includes many user input constraints like deadline, execution time, make span, cost etc. All of these are defined in the service level agreements which are documents that act like a contract between the consumer and the cloud service provider.

(h) **Load Balancing:** Load Balancing is the method in which the load is distributed across the cloud across nodes and links such that at any given time, no node or link gets overloaded or remain free. It targets to optimize usage of resources, increase throughput, reduce response time and avoid overload of any one of the resources. Good scheduling algorithms aim at keeping the load well balanced so as to increase the efficiency of the system [133].

3.10 JOB SCHEDULING ALGORITHMS

There has been immense development of various algorithms that can be used for allocation, scheduling and scaling of the resources in the cloud in an efficient manner. The primary objective of such a job scheduling algorithm is to enhance the performance and improve the quality of service while maintaining the efficiency and the fairness among various jobs and also reduce the cost of execution.

Conventional scheduling algorithms such as First Come First Serve, Shortest Job First, Round Robin, Priority Scheduling, Max-Min and Min-Min Scheduling aren't capable enough for such objectives, while some meta-heuristic algorithm include ACO (Ant Colony Algorithm), Efficient Multi Queue Job Scheduling (EMQS), GA (Genetic Algorithm) etc provide support for such objectives [37]. These all algorithms are elaborated below:

3.10.1 First Come First Serve (FCFS) Scheduling Algorithm

First Come First Serve is also known as "First in First Out" and as evident from the name, the job execution is done according to order of the arrival time of the job. FCFS algorithm may cause a convoy affect which can happen if there's a job requiring heavy amount of workload under execution and all other jobs are stuck in the job queue due to this [36]. The advantages and disadvantages for the first come first serve scheduling algorithm can be presented as [134]:

Advantages:

(a) No complex logic is involved. It involves the simple method of putting job requests in a queue and executing them one by one.

(b) It is pretty simple and easiest to implement.

(c) No starvation happens because eventually, every process gets a chance to run.

The Figure 3.5 explaining the working process of FCFS Algorithm is shown below

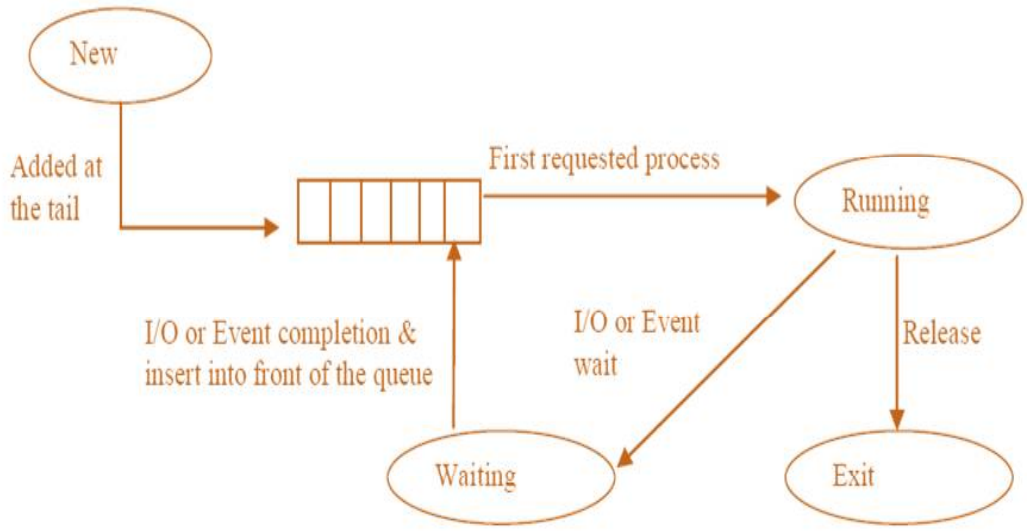

Figure 3.5 FCFS Scheduling Algorithm [121]

Disadvantages:

(a) Pre-emption cannot happen in this algorithm. If a process has been started, CPU will execute it until it is completely finished.

(b) Due to nonexistent pre-emption, if a process has a started and it takes very long, all the other jobs in the queue will have to wait for a very long time until that process under execution is completed.

3.10.2 Shortest Job First (SJF) Scheduling Algorithm

Shortest Job First is also known as Shortest Job Next, the algorithm picks the tasks with the minimum execution time first i.e. highest priority is given to the tasks that have the least execution time while the lowest priority is given to the tasks with the highest execution time. The scheduling can be either pre-emptive or non-preemptive. The former one will pre-empt the current process in execution whereas the latter will let the current process run so as to finish its CPU burst [36] [37]. The Figure 3.6 explaining the working process of SJF Algorithm is shown below

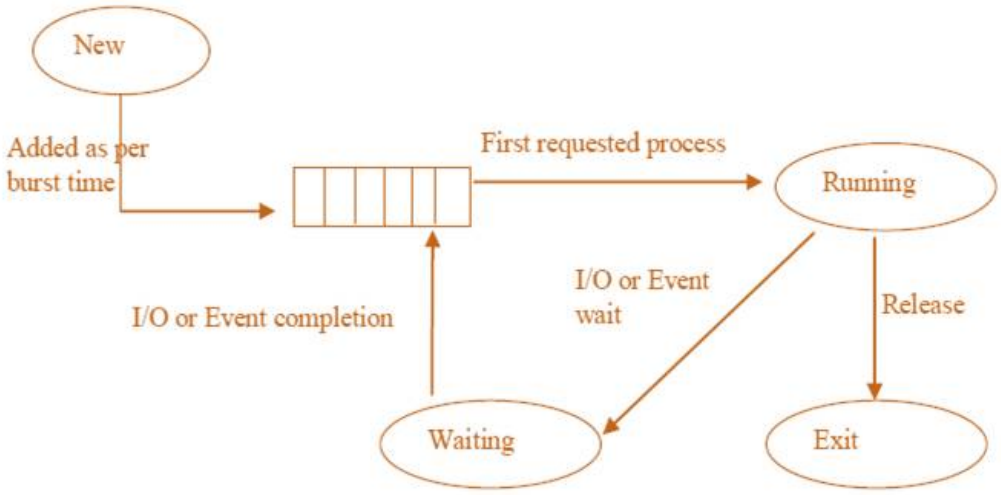

Figure 3.6 Shortest Job First Scheduling Algorithm [121]

The advantages and disadvantages for the shortest job first scheduling algorithm can be presented as [134]:

Advantages:

(a) Short processes get executed first, as evident from the definition.

(b) It gives better throughput as more processes get executed in a particular amount of time.

Disadvantages:

(a) Processor must know how much time a particular process will take to complete execution. In most of the cases it is not possible.

(b) Long processes will often face starvation.

3.10.3 Round Robin (RR) Scheduling Algorithm

Round Robin is one of the most conventional, simplest and the popularly used algorithm. It works best for timesharing systems as it equally distributes the load to all the resources and works in the cloud computing system in a very similar manner as that of in a process scheduling system. The Figure 3.7 explaining the working process of Round Robin Algorithm is shown below.

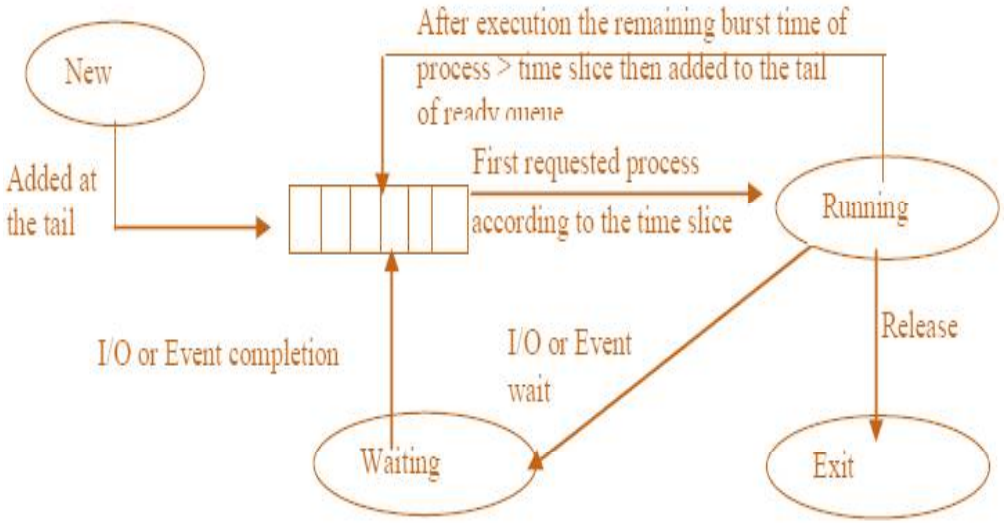

Figure 3.7 Round Robin Scheduling Algorithm [121]

The working of this algorithm includes a fixed time unit (quantum) and a circular queue. Each job's execution can happen only in this time quantum. The process works as follows: The CPU scheduler picks the first process in the queue and sets a time to interrupt that process after one time quantum so that it gets dispatched. If the job execution is not complete in the allotted time quantum, it is returned back to the queue and it waits for another round.

The biggest benefit of Round Robin algorithm is that jobs are executed in a turn wise manner and due to this sequence, the jobs don't have to wait for some other job to finish its execution. Hence, no starvation issue happens with this. But the problem is that if the workload is very heavy, the queue can get highly loaded and it takes a lot of time to finish all the jobs. Moreover, it's very tough to decide a perfectly suitable time quantum [38]. The advantages and disadvantage for the Round Robin scheduling algorithm can be presented as [134]:

Advantages:

(a) All process gets same priority as every process is served according to a fixed time quantum.

(b) No starvation happens as every process gets execution time.

Disadvantages:

(a) The throughput is highly dependent on the length of the time quantum. If time quantum is wrongly decided or is longer than required, the algorithm will exhibit same behavior as the FCFS.

(b) If time quantum is shorter than required, the CPU switches way too many times between different processes. This can reduce the CPU efficiency.

3.10.4 Min-Min Scheduling Algorithm

Min-Min algorithm attempts to map each task to the required resources in such a manner that they can finish the task in the minimum possible time. The algorithm estimates the execution and the completion time of each job on a particular resource available. The algorithm works in two phases - in the first phase, it estimates the least execution time for all the tasks. In the second phase, it picks the task with the least execution time. Then, it assigns the task to the resource that can complete it in minimum time. The procedure is repeated until all the tasks get scheduled [39].

3.10.5 Max-Min Scheduling Algorithm

Max-Min algorithm works in a very similar manner to the Min-Min one. The only differentiating feature is that in this algorithm, the task with maximum and earliest completion time is allocated first i.e. larger tasks are given priority in this algorithm [39].

3.10.6 Priority Scheduling Algorithm

In Priority scheduling algorithm scheduler allocate a fixed priority rank to each user process and the processes run on the priority basis. The jobs with the highest priority run first and the ones with lower priorities have to wait. The jobs that have equal priority are in the First Come First Serve manner. The biggest drawback of this algorithm is the starvation of some processes which may have lower priority [40]. A priority can be defined in two manners - internally as well as externally. The definitions are as follows [81]:

Internally defined priorities: They use some measurable quantity to calculate the priority of a process. For e.g., the number of open files, time limit, ratio of average I/O burst to the average CPU burst and memory requirements are often used to define the priorities [81].

Externally defined priorities: These are set by criteria outside the Operation system. For e.g., type & amount of funds paid for the computer usage, importance of process, the department sponsoring the process or some other political factors [81].

The Figure 3.8 explaining the working process of Priority Scheduling Algorithm is shown below

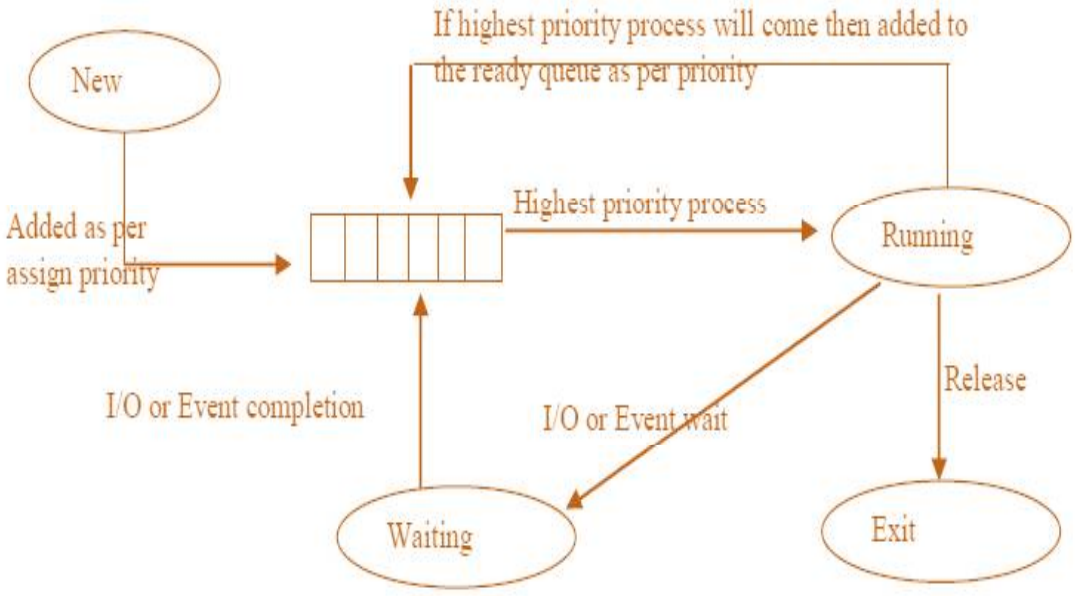

Figure 3.8 Priority Scheduling Algorithm [121]

The advantages and disadvantages for the Priority scheduling algorithm can be presented as [134]:

Advantages:

(a) The priority of a process can be defined on the basis of things like user preference or time & memory requirements. For e.g., in case of a high end game, it's important to have good graphics. So, the processes that update the screen in that game will have higher priority so that they can lead to better graphics performance.

Disadvantages:

(a) There is a requirement for a second scheduling algorithm to deal with processes with equal priorities. In case of pre-emptive priority scheduling, the higher priority process will get preference and if that happens, lower priority process will keep waiting for the end of all higher priority process and thus, face starvation.

3.10.7 Ant Colony Optimization (ACO) Scheduling Algorithm

Ant Colony Optimization has literally been inspired from ants. In fact, ants have inspired several researchers till date and this algorithm is an optimization technique inspired by the social behavior insects. The basic approach of ACO is to copy the behavior of ants. When ants have to search for food, they communicate using a chemical called pheromone. They begin their search randomly until one ant succeeds in finding a food source. As the ant leaves pheromones on its path, it is traceable by other ants. Based upon the intensity of pheromone deposit on multiple paths, the ants determine the shortest distance from their nest to the destination desired food. Over a period of time, ants choose the shortest path to food where huge amount of pheromones have been accumulated because shorter path selected with have high pheromone value as compared to other paths which posses low pheromone value [41] [42]. This phenomenon was first seen in the very famous double bridge experiment [43]. It ascertains that when given a choice between long and short path to food, ants consistently found the shortest path. And to prevent a suboptimal path, the chemical - pheromone evaporates over time [44]. However, the pheromones on the shortest path remain high, as in this case, deposit speed of the pheromones exceeds the evaporation speed. It is unique as compared to other algorithms as in this approach, every ant (job) assembles its own individual result set and is then, later incorporated into a complete arrangement. The elaborate characteristics of the ACO algorithm, a lot of algorithms based on ACO meta-heuristic have been developed and applied to many optimization problems. The Computational Flowchart of working of ACO Algorithm is depicted below in Figure 3.9

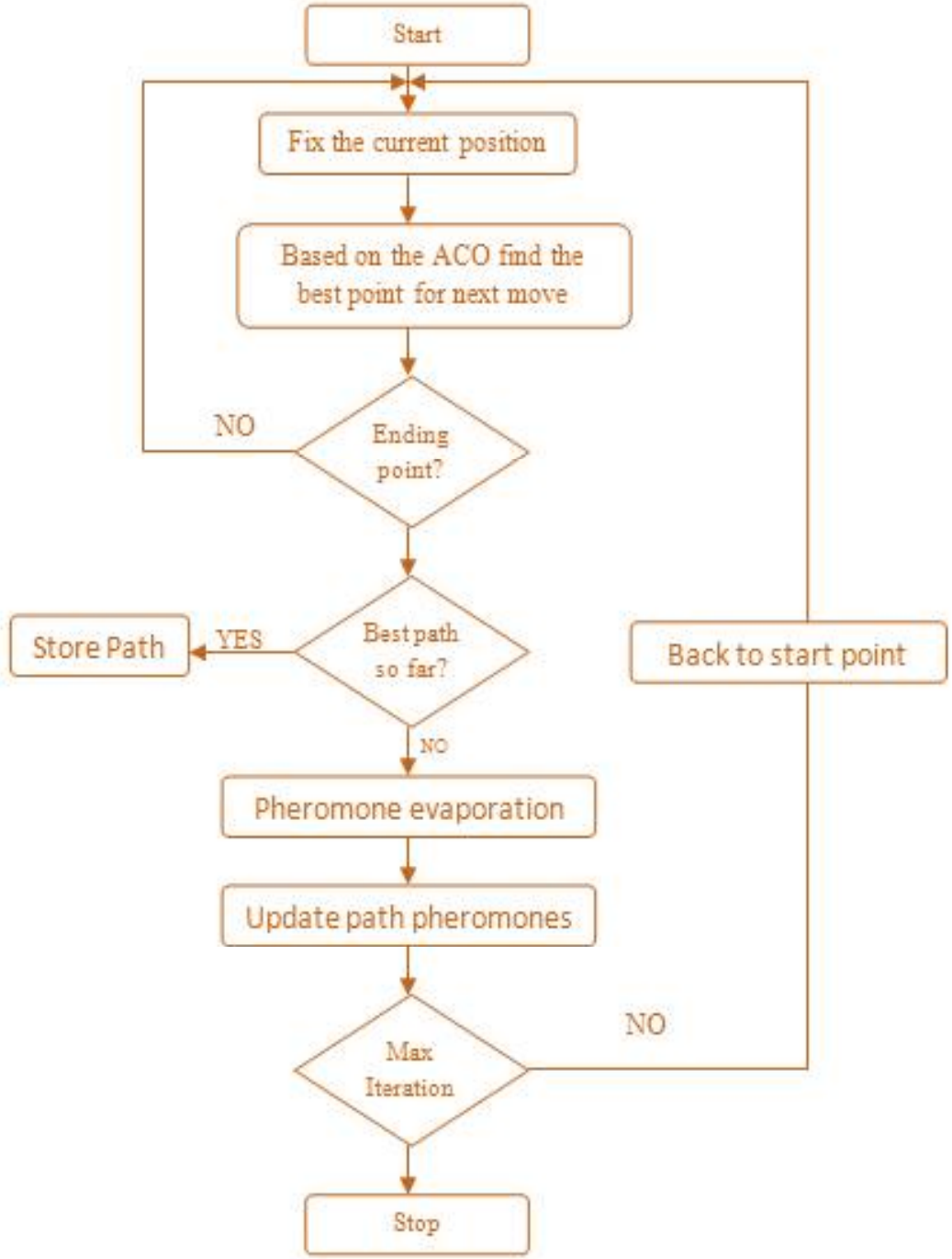

Figure 3.9 Flowchart of ACO Algorithm [44]

It is evident than an ACO algorithm can work on any combinational problems as long as the following can be defined [45]:

(a) Problem representation that can allow the nets to build or modify solutions incrementally.

(b) Heuristic desirability η of edges.

(c) A constraint satisfaction method that can force the construction of possible solutions.

(d) A rule to update pheromones that can specify how to update pheromone trail *n* on the graph's edges.

(e) A transition rule of the pheromone trail and the heuristic desirability.

The use of ACO algorithm in the scientific communities is growing. The main reason to pick this algorithm is that it will produce a high quality solution. Because of its success, the ACO has also been applied in other domains like computational intelligence, industries and real world applications. The application of Ant Colony Optimization Algorithm includes:

(a) Telecommunication Networks: It is considered a very effective algorithm for the routing technique in this field. The algorithm was applied to the routing issues with both packet switched and circuit switched networks. Circuit switching involves telephone network whereas packet switching involves local area network. The ant-inspired algorithm suggested through the concept of worked well with wired networks and hence, work really well with both adhoc and wired networks [46] [47].

(b) NP-Hard Problem: Heuristic techniques can be applied to a number of problems and performance can be compared with other techniques. ACO has already been tested with hundreds of different NP-hard problems. Some of the problems that have been tackled are: assignment problems, routing problem that happens during distribution of goods, assignment of set of objects to give number of locations under specific constraints, scheduling problems and so on. ACO has also been applied in the field of machine learning and the bio-informatics [48].

(c) Industrial Applications: The major use of this algorithm is in the Euro Bios where they applied the algorithm to a number of scheduling problems in the

industries like flow shop for example in the aluminum casting centre. The issue that arises with its use is the challenge to maintain real world capabilities, management and audit. Some authors had applied this algorithm to assemble the balancing problem as well as constraints between various tasks [48].

3.10.8 Efficient Multi Queue Job Scheduling (MQS)

An Efficient Multi Queue Job Scheduling Algorithm (EMQS) is presented earlier in [65] which based upon the concept of Multi Queue. In this algorithm each of the client submitted jobs is processed with equal importance. It sorts all the jobs in an ascending order depending on their burst time. The resources are provided by the queue manager which is a part of cloud computing that manages the utilization of resources present in one cluster.

The queue manager tracks the systems that run the jobs at the current time. Once the scheduler schedules the output of the queue, it collects again by the queue manager. Thus, it helps to improve the user satisfaction as user requirement can vary according to their present needs. It also helps to reduce starvation as it does dynamical allocation of jobs so that it can select the best suitable jobs from the job pools

The Three queues which are formed hold the jobs as:

(a) Small Queue: First 40% jobs get sorted in this.

(b) Medium Queue: Next 40% get sorted in this.

(c) Long Queue: The remaining 20% get sorted in this.

The Architecture of Efficient MQS is shown below in Figure 3.10

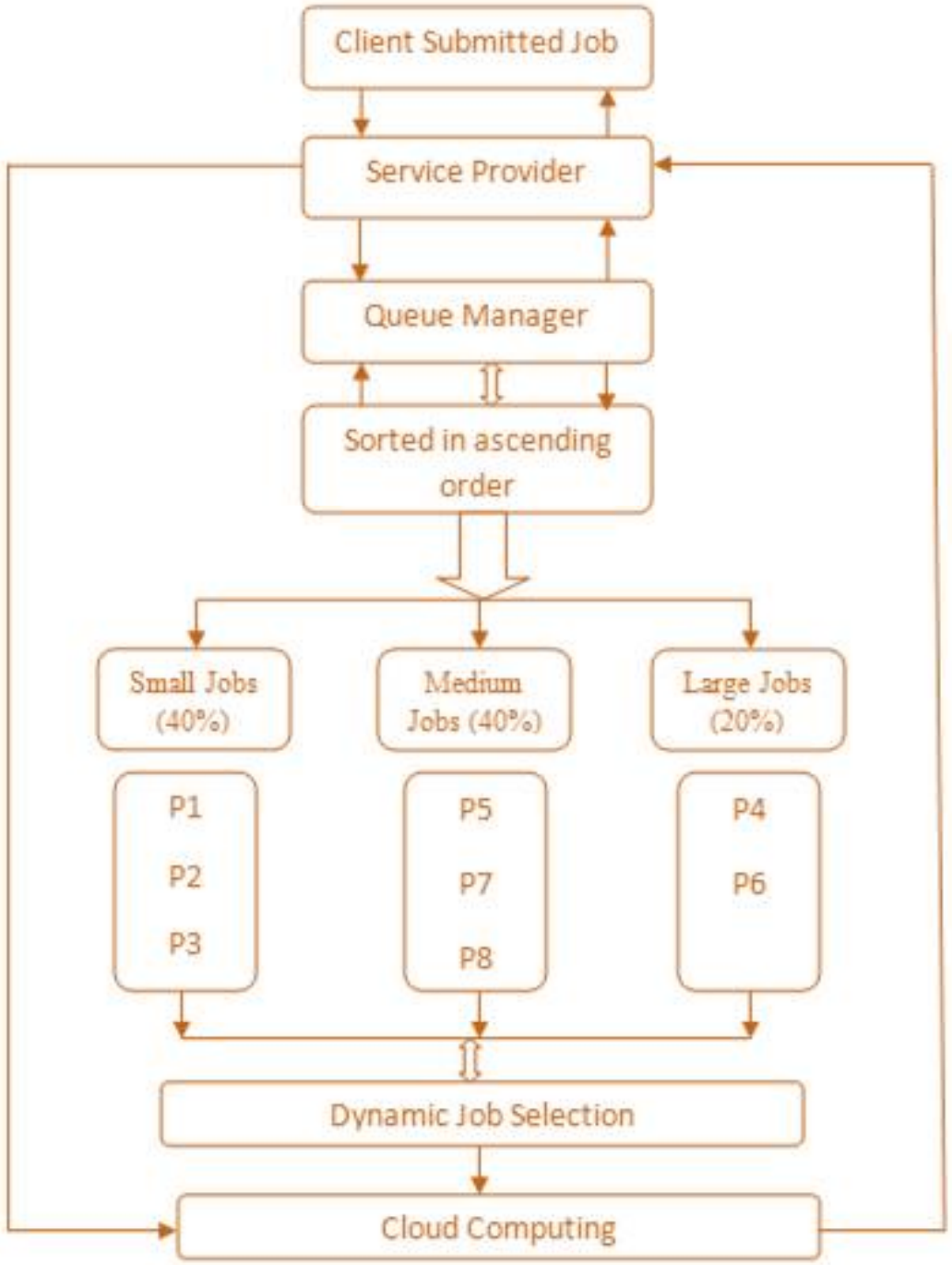

Figure 3.10 Architecture of Efficient MQS for Cloud Computing [65]

The optimal allocation decreases time and availability of space in a productive manner without compensating the quality of the system.

3.11 PERFORMANCE ANALYSIS IN TERMS OF ENERGY EFFICIENCY

3.11.1 Experimental Scenario and Performance Metrics

For scheduling the cloud user jobs scheduler have option to choose and implement from the vast number of job scheduling algorithm available, which has been illustrated in the above section. The performance evaluation of two job scheduling heuristic aalgorithms i.e. Efficient MQS (Multi-queue job scheduling) and ACO (Ant colony optimization) algorithm is carried out in cloud environment. In each scenario the user's job creation and execution order may vary; so results of best two iterations are shown .The strengths of both algorithms have been measured on the parameter of energy consumption and time consumption in cloud computing.

Scenario 1: With the 5 number of jobs, the efficient MQS breaks up the jobs in three types - small, medium and long queue while the ACO algorithm is based on random search. On the implementation Efficient MQS algorithm showed better accomplishment in terms of energy and time consumption as compared to the ACO. The results of best iteration conducted are shown below in Table 3.1 and corresponding comparative graph to values are shown below in Figure 3.11 and 3.12.:

Table 3.1: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 5

	Efficient MQS		**ACO**	
No of Jobs	Energy	Time	Energy	Time
5	327	26	605	72
	351	31	651	72

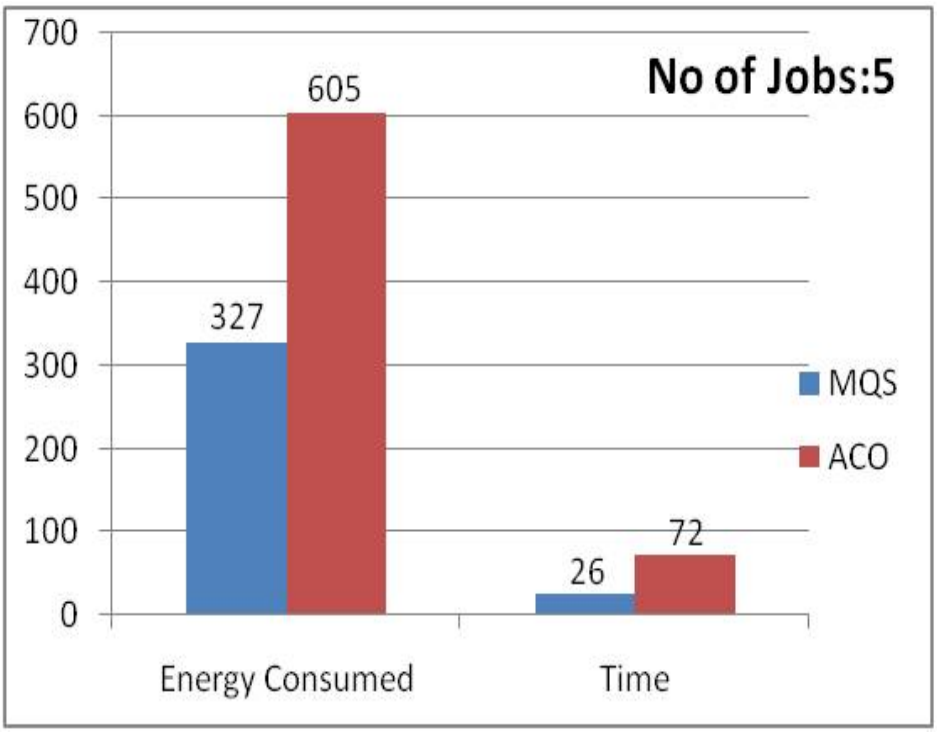

Figure 3.11 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 5 (Ist Iteration)

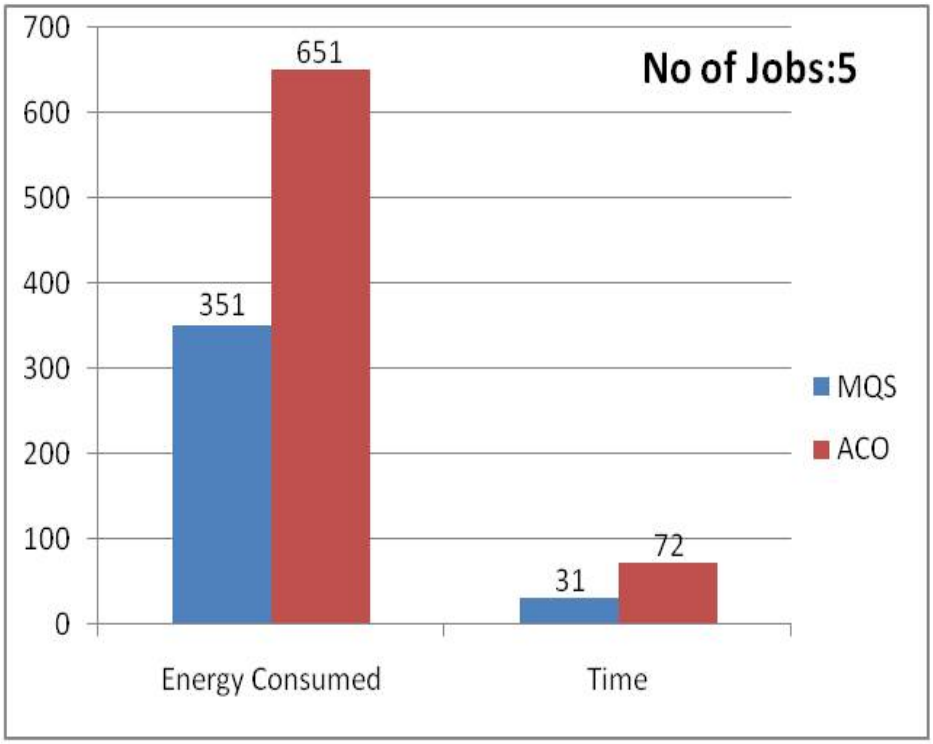

Figure 3.12 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 5(IInd Iteration)

Scenario 2: With cloud user jobs to be scheduled taken as 10, the MQS breaks the jobs different job queues while ACO again uses the random search phenomena schedule jobs. In each implementation, the scheduler arranges the number of jobs in different sequences and hence, the results obtained from the best iterations suggest that MQS consumed lesser energy and time as compared to ACO algorithm in each cycle. The values obtained represented in Table 3.2 and corresponding graph to values are shown below in Figure 3.13 and 3.14.

Table 3.2: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 10

	Efficient MQS		ACO	
No of Jobs	**Energy**	**Time**	**Energy**	**Time**
10	885	58	2077	146
	935	59	2103	146

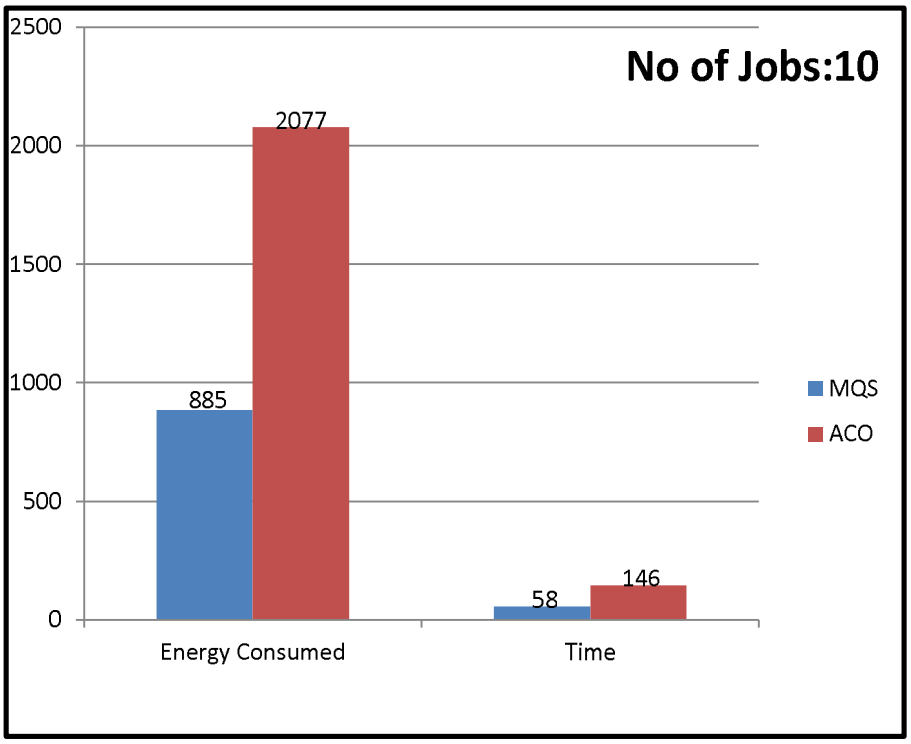

Figure 3.13 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 10 (Ist Iteration)

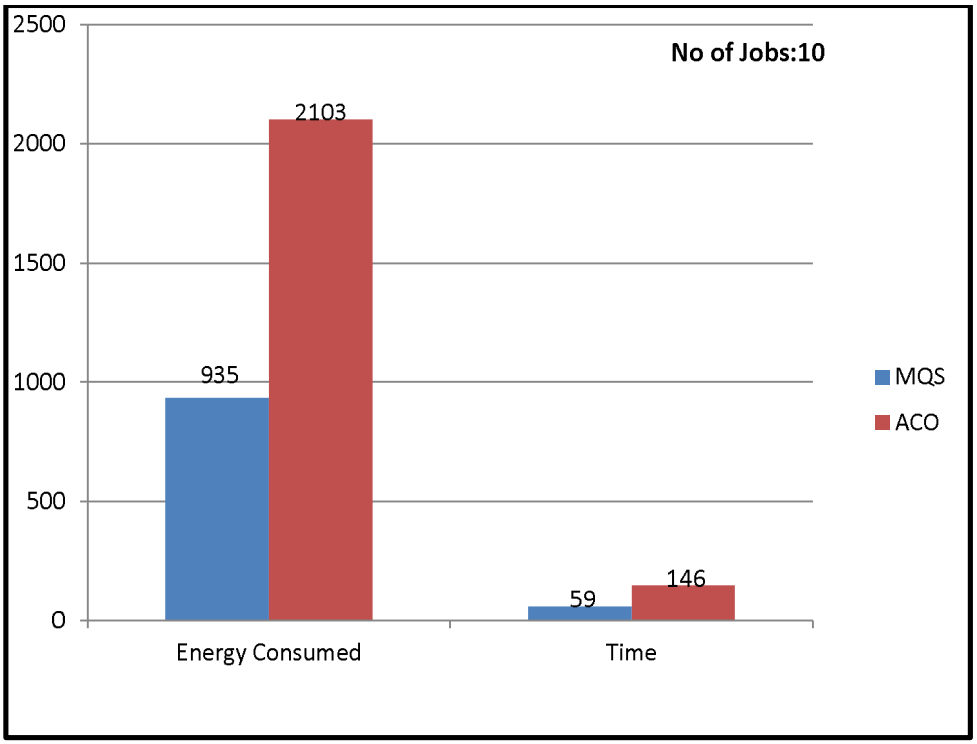

Figure 3.14 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 10 (IInd Iteration)

Scenario 3 : When no of cloud user jobs to be scheduled are taken as 15, the result obtained with best two iterations by both techniques is shown in Tabular form (Table 3.3) while figure 3.15 and 3.16 depict comparative analysis graph between two techniques. Clearly Efficient MQS Algorithm schedules the user jobs in better way when compared with ACO Algorithm in term of energy consumption and time.

Table 3.3: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 15

	Efficient MQS		**ACO**	
No of Jobs	**Energy**	**Time**	**Energy**	**Time**
15	1525	68	3935	211
	1548	71	4010	211

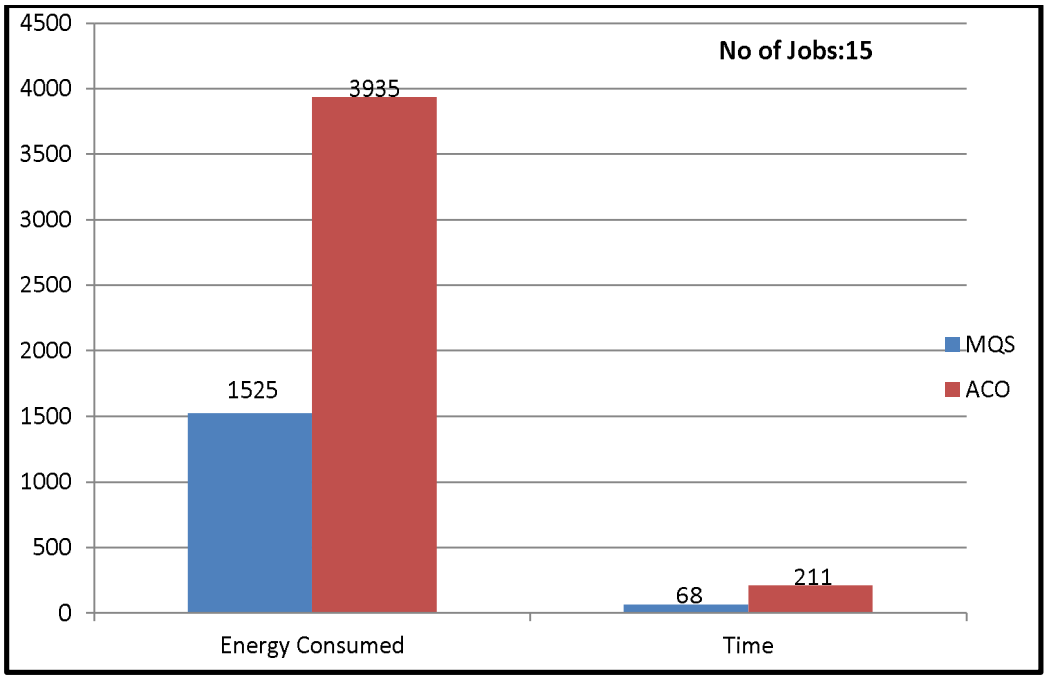

Figure 3.15 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 15 (Ist Iteration)

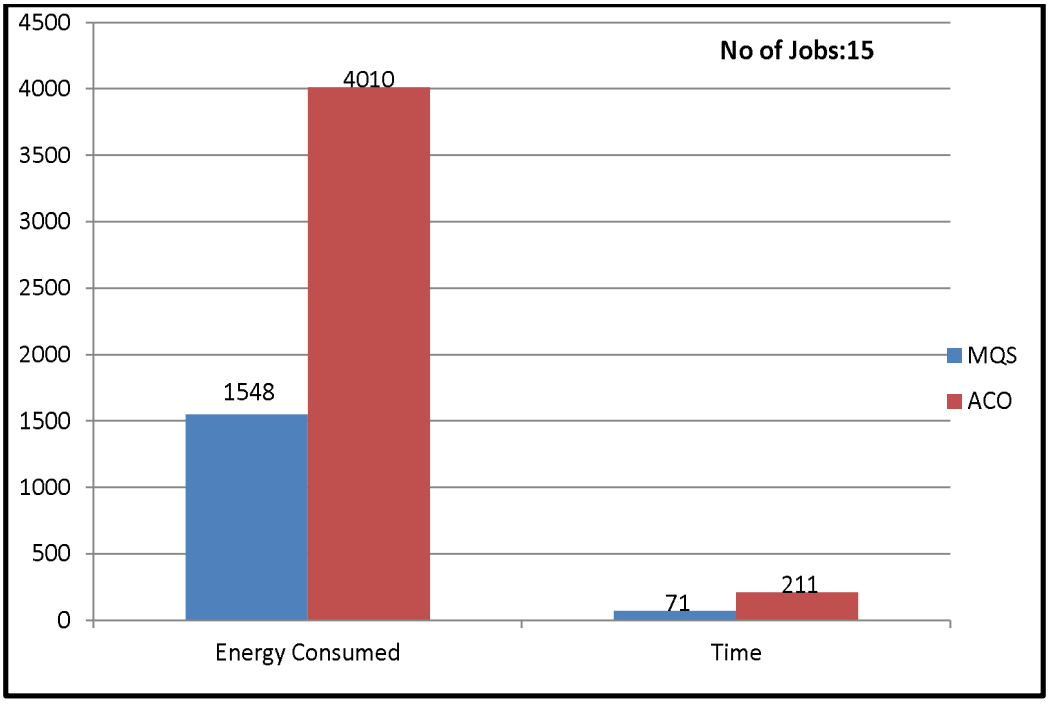

Figure 3.16 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 15 (IInd Iteration)

Scenario 4: When no of cloud user jobs to be scheduled are taken as 15, the scheduler schedule number of jobs in different order in each execution , so the result obtained with two best iterations by both techniques is shown in Tabular form (Table 3.4) while figure 3.17 and figure 3.18 depict comparative analysis graph between two techniques. Clearly Efficient MQS Algorithm shows best results in the terms of energy consumption and limits execution time as compared with ACO Algorithm.

Table 3.4: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 20

	Efficient MQS		ACO	
No of Jobs	**Energy**	**Time**	**Energy**	**Time**
20	2432	95	6921	281
	2527	103	7124	281

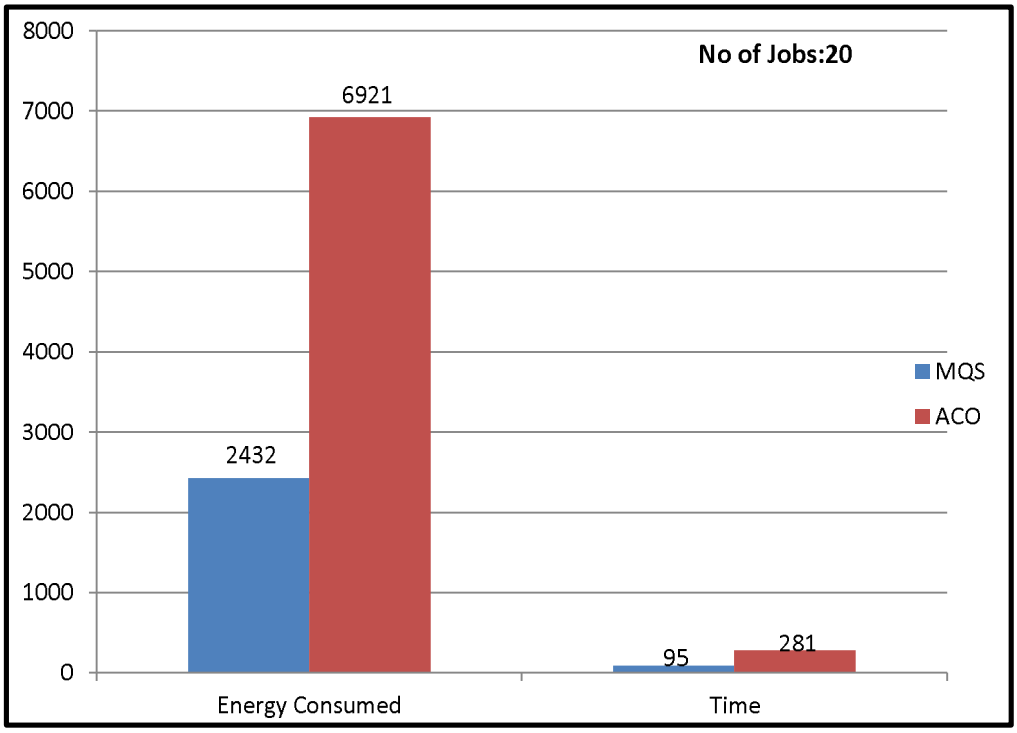

Figure 3.17 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 20 (Ist Iteration)

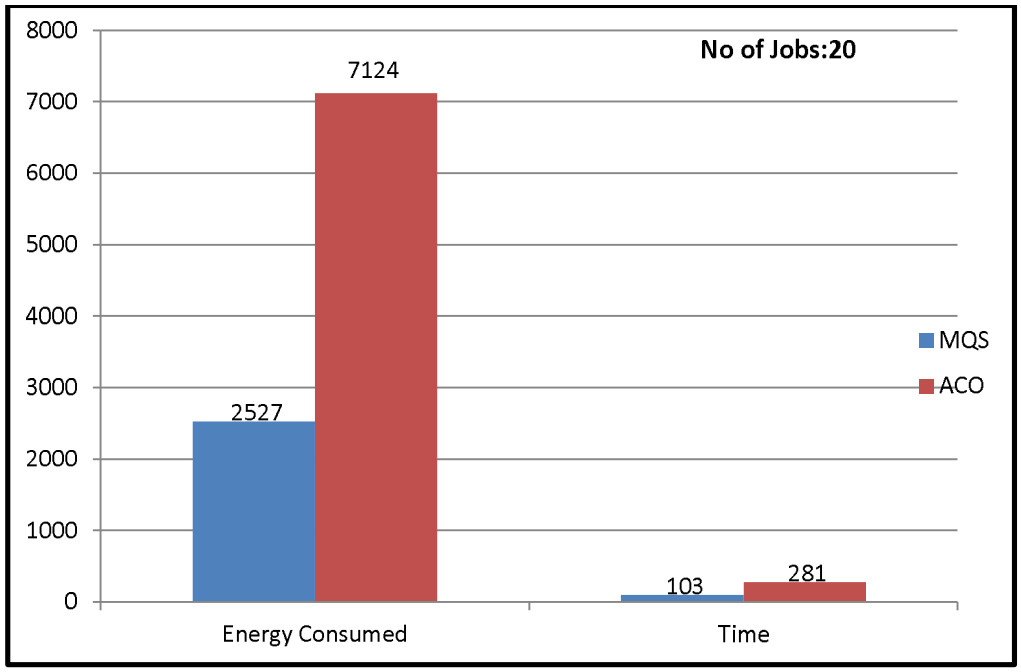

Figure 3.18 Comparison of (Efficient MQS and ACO) Algorithms on No of Jobs: 20 (IInd Iteration)

In this research work we analyzed the results by demonstrating rise in performance, sharp reduction in energy consumption, limiting execution time to some degree, minimizing cost and improvement of overall end user jobs experience. The Scheduler achieved its objective to reproduce the cloud service performance by handling the workload generated by cloud user jobs efficiently.

3.12 SUMMARY

This Chapter discusses various aspects of job scheduling used in the cloud computing and implemented two heuristic job scheduling algorithms. The ultimate goal for the cloud service provider in cloud environment is to how to make use of available cloud computing resources effectively so as to expand the profits with job scheduling system. We compared the performance of Efficient MQS Algorithm with the ACO Algorithm. Analysis and number results show that Efficient MQS algorithm for job scheduling performs much better than ACO algorithm in terms of minimizing energy consumption and execution time. All these analysis are used in proposing the new job scheduling algorithm SMQS (Smarter Multi Queue Job Scheduling) algorithm for cloud computing subsequently discussed in Chapter 4.