CHAPTER 6
SMARTER MULTI QUEUE JOB SCHEDULING (SMQS): PSEUDO CODE AND IMPLEMENTATION

6.1 OVERVIEW

In this chapter the proposed smarter multi queue job scheduling (SMQS) policy is implemented successfully considering the tradeoffs between performance improvement and energy saving. The experimental results of proposed smarter multi queue job scheduling (SMQS) are compared and discussed with the existing efficient multi queue job scheduling (EMQS) algorithm in terms of energy consumption and execution time.

6.2 INTRODUCTION TO SMARTER MQS

As discussed in previous chapter about the Smarter MQS job scheduling model which effectively separate user jobs into two job queues then give equal importance in executing each submitted jobs from both the queues. The results obtained are tested on different number of cloud user jobs. The proposed smarter MQS algorithm will empower us to reduce energy consumption while naturally to some degree will reduce job completion time and the overall cost. The Chapter also describes simulation environment and simulation specific details.

6.3 PSEUDO CODE OF SMARTER MQS

```
1. Let the input Jobs={T1,T2,T3,T4……Tn}.

2. Let the network Nodes={Sys1, Sys2,Sys3…Sysn}

3. Create Jobs matrix based on various processing Factors

4. For each Job.Processing do

5. Create the threshold Point for prioritized    Group of Jobs.

   Multi_queues=Break.Prioritized(Based on energy(threshold));
```

```
6.  End Job_Processing
7.  Initialize System network with MPI network message;
8.  MPI.response.table(acknowledgement.Nodes.add);
9.  If Job. ready(Exists in execution matrix's)
        Load network configurations
10. Distribute Jobs through divide and merge policy.
11. Eliminate executed Jobs from network processing
12. end_if
13. End Repeat till all jobs will not get executed
14. Evaluate system performance.
15. Stop
```

6.4 IMPLEMENTATION

6.4.1 Implementation Tool: .Net Framework Using Microsoft Windows Azure Platform

Microsoft Windows Azure platform also support applications that are built on .Net framework or other ordinary computing languages that are supported by Windows systems. For example, C#, C++, Visual Basic etc. Software that is required to develop a cloud computing application on .NET framework [96]:

A. Microsoft Visual Studio 2010

B. Microsoft Windows Azure Platform

6.4.2 Microsoft Visual Studio 2010

The visual studio consists of cloud development tools which the developers can to code cloud centric apps and services. Developing such applications has become much easier to code and then test as it can be tested conveniently using the simulation environment of the Windows Azure. Even Microsoft visual studio has tons of features that allow easy development of both desktop and web applications and azure software helps in easy debugging, testing and deployment. In this section, exposure about how to develop applications using this Microsoft software is carried out. According to MSDN, both of this software enable easy building/development, editing, configuration, debugging and deployment the applications that can run on windows azure.

Microsoft [97] also offers direct support for developing Azure apps using Visual Studio. Before you add any azure tools, it's crucial to install IIS. It's used for the local development simulation of cloud and the easiest way to do it is using a web platform installer that's available at microsoft.com/web. We have to select the platform option and then click to include all the recommended products in the web server. We can also need to pick the Cloud Service project template which will open up a new cloud service Project Dialog, that in turn, allows adding various roles to the cloud services.

The roles that can be added are of two types –

(a) Web role which is basically a web app that's running on the IIS and is accessible via http or https.

(b) Worker role which is a background processing app that runs a .Net code and has the ability to expose internal as well as internet-facing endpoints.

6.4.3 Microsoft Windows Azure Platform

Windows Azure helps to build the web applications that run and store their data in the Microsoft data centers. It can be used to store data only with the applications that use the data that is outside the public cloud. It can also be used to connect the on-premise applications [92]. The platform also offers a runtime execution environment for the managed codes so as to host and run highly scalable solutions. Every Windows Azure computer instance is also a virtual machine (VM) instance that's created by the platform. The team hosting the application can configure the number of instances. Each VM instance, then, runs an azure agent so as to connect as well as communicate with the Windows Azure fabric. It also has a local file system which can be utilized by the web or worker role during their life-time but once this virtual machine instance is shut, virtual machine and local storage goes away [94].The Microsoft Window Azure Architecture is shown below in Figure 6.1

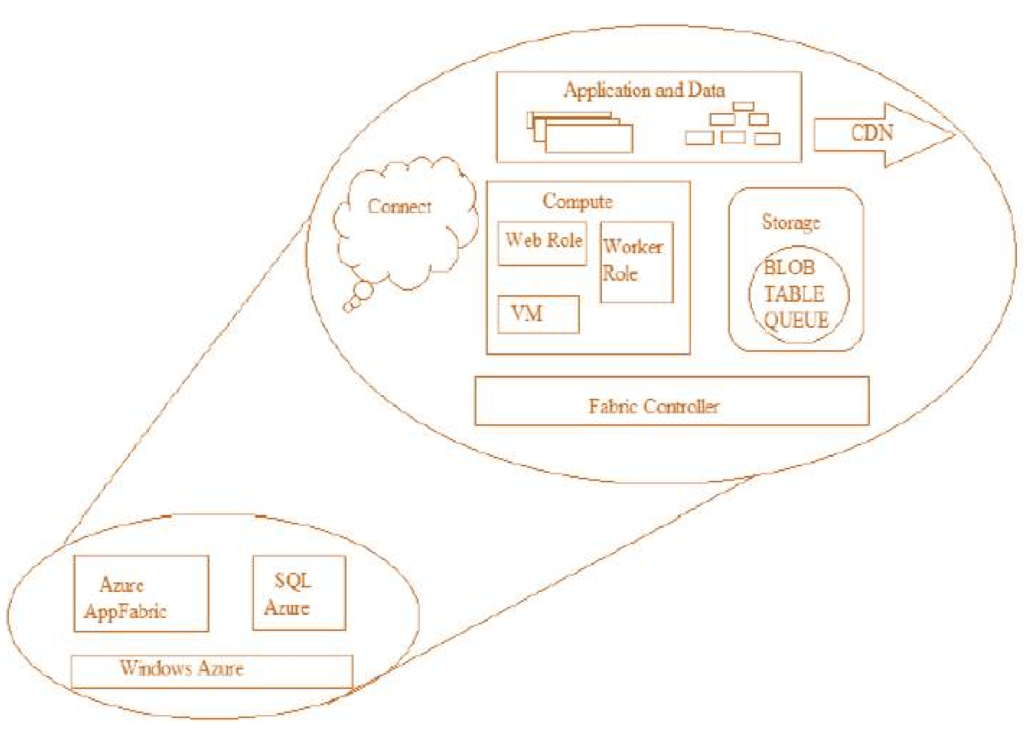

Figure 6.1 Microsoft Windows Azure Architecture [93]

6.4.3.1 Microsoft Windows Azure Components: The two main Microsoft Window Azure Components are defined as Cloud Application and Data Management. They are elaborated as under:

(a) **Cloud Applications:** There are two roles offered by Windows Azure [92]:

Worker Role: Worker Role is a general role that is designed to run a variety of codes. For example, an application that has to process data in parallel.

Web Roles: Web role is a role that is designed for code that directly talks with web browsers or http clients. It depends on the Microsoft web server - IIS. For example PHP applications or ASP.NET applications.

(b) **Data Management:** Each windows azure application runs in one or more virtual machine. Each virtual machine has local storage that an application can use freely. The Windows Azure offers the following data management services such as SQL azure, blob, tables, queues, and Azure file services [92]:

SQL Azure: It consists of SQL Azure database and SQL Azure Data Sync. The SQL Azure database offers a cloud-based DBMS. It helps the cloud and on-premise applications, both, to store the relational data in the servers in the Microsoft data centers. It is built on Microsoft SQL server and an organization has to pay to use this technology [93].

Blob (Binary Large Object): BLOB is one of the simplest ways to store data. A storage account contains one or more containers and each container has one or more blobs that store unstructured data in large amount. They can be as large as 1 TB. Blobs are further classified into blocks so as to help transfer the large blobs [93].

Tables: Windows Azure offers tables to enable the applications and data to work together in a more fine-grained way. However, it is to be noted that these aren't relational tables. The data stored in these is actually kept as a set of entities with certain properties. It has no defined schema. Instead, the properties of the entities can have types like int, bool, date time or string. An application doesn't have to SQL to access data from these tables. Windows Azure storage can easily partition it and keep it across various servers to boost performance [93].

Queues: These are slightly different than other management services like table and Blobs as they allow the communication between web and worker role instance. For example, if a user requests to perform a task via webpage that has been implemented by Windows azure web role, the web role instance receives that request and writes the message on the queue and then, the worker role reads that message and actually, performs the task [93].

Azure File Service: Windows Azure fabric offers high scalable solutions that can support large volumes of users accessing various applications at the same time; a key feature of Windows Azure. This is possible due to a feature known as scale-out that helps to manage the sudden rise of users. The fabric controller can manage and control the Wi

indicates the requirements for deployment like storage, no. of worker and web roles etc. If any machine fails, fabric controller gets the notification and as a response, it configures a new VM with the same configuration as that of the failed machine and then adds it to the fabric. This is done very quickly so that user never faces any issue or gets awareness about such a machine failure happening [94].

(c) **Networking:** At present, azure runs in thousands of data centers that are spread worldwide. When an application is run, any one or more of these data centers gets selected for use. It's also possible to connect to any of these data centers in the following ways:

Virtual Network: It is used with other Windows Azure services like VMs. For example, a virtual network can be used to offer connectivity to the VMs. The approach is ideal under certain conditions such as if we want to run Microsoft Share Point farm in the windows Azure. Apart from VMs, it's also possible to use a virtual network with cloud services. All the virtual machines that run in the cloud can already communicated with one another and there's no need to create a separate virtual network for this purpose [95].

Traffic Manager: It offers a choice between three load-balancing methods - round-robin, performance and failover [95]. Round-robin always distributes the traffic across all the services in an equal amount. The Performance is based on the network latency, it directs the traffic to the closest service while Failover acts in case of primary service fails and directs the traffic to a backup service.

(d) **Identity and Access Control**: Knowing about the user helps the application to understand or determine how to interact with that user. Azure helps to track user identities and integrates it with identity stores like active directory or multi-factor authentication.

6.5 RESULT ANALYSIS AND DISCUSSION

The performance evaluation of said proposed smarter multi-queue job scheduling model (SMQS) is done and it is compared with the Efficient Multi-Queue Job Scheduling Model. The simulation is carried out in the cloud informant to find the best schedule to process the user jobs on the available virtual machines. The overall performance of the said proposed model is measured by running it under randomly generated cloud configurations.

6.5.1 Performance Metrics

Cloud computing provides on-demand computing with high performance and high scalability. However, the increasing energy consumption by executing user's jobs on virtualized recourses in cloud network has become a major problem. Keeping this factor in mind energy aware job scheduling framework for cloud environment has been introduced. Applying energy-aware resource scheduling is an effective way to save energy, so the parameters which are considered are mentioned below:

(a) **Energy Consumption:** The ultimate goal of a proposed algorithm is to achieve minimizing energy consumption. The scheduling framework of proposed algorithm mainly focuses on the energy efficiency in cloud environment along with the minimum completion time. The calculation of energy consumed for submitted cloud user jobs is shown in equation 1:

$$\boxed{\text{Energy Consumption}_p = \sum_{p}^{s} (\text{Energy Consumption}_{pi} + (\text{Waiting Time}_{pi} * \text{Ideal Energy}_{pi}))_s} \quad \text{.... (eq.1)}$$

(b) **Execution Time:** The execution time for cloud user's submitted jobs is defined as the time taken by the scheduler in executing the submitted cloud user jobs. The calculation of energy consumed for submitted cloud user jobs is shown in equation 2:

$$\boxed{\text{Time Consumption}_p = \sum_{p}^{s} (\text{Time Consumption}_{pi} + \text{Waiting Time}_{pi})_s} \quad \text{...... (eq. 2)}$$

Both the algorithms i.e. Efficient MQS and Smarter MQS algorithms was executed and evaluated on the same set of cloud user jobs. The experiments were repeated using different values of submitted cloud user jobs. The simulation is carried on real time environment and for instance, N different user jobs (5, 10, 15 and 20) conducted in .Net Framework using Microsoft's Windows Azure Platform. In this real time simulation environment the user's job creation and execution order may vary on each iterations. The result graphs obtained on the best iteration are shown below on different cloud user jobs which show that the proposed Smarter MQS Algorithm has a big impact in the reducing the energy consumption and execution time.

The Table 6.1 shows the obtained result values for the two algorithms on number of user jobs taken as 5. Further the plot in Figure 6.2 which is based upon energy consumption and execution time compares the proposed Smarter MQS technique with Efficient MQS algorithm in cloud environment; clearly experimental results demonstrate the effectiveness of our Smarter MQS algorithm over Efficient MQS algorithm.

Table 6.1: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 5

	Efficient MQS		**Smarter MQS**	
No of Jobs	**Energy**	**Time**	**Energy**	**Time**
5	**336**	**31**	**127**	**29**

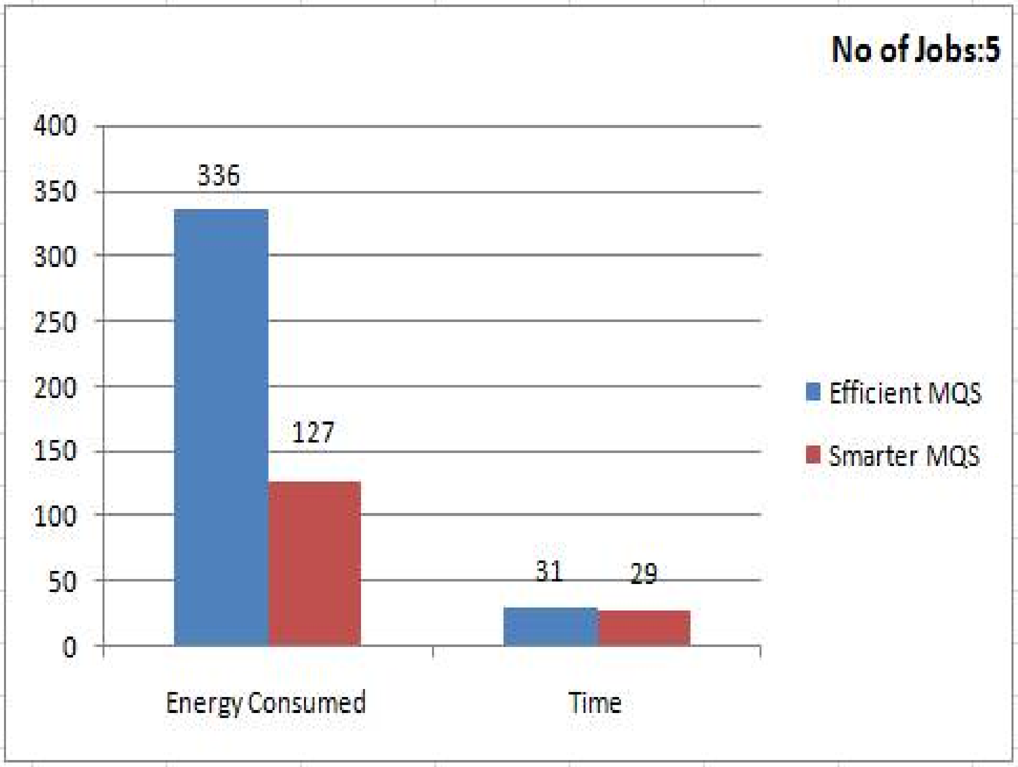

Figure 6.2 Comparison of (Efficient MQS and Smarter MQS) Algorithms on Number of Jobs: 5

The Table 6.2 shows the obtained result values for the two algorithms on number of user jobs taken as 10. Further the plot in Figure 6.3 which is based upon energy consumption and execution time compares the proposed Smarter MQS technique with Efficient MQS algorithm in cloud environment; clearly experimental results demonstrate the proposed Smarter MQS algorithm outperforms Efficient MQS algorithm.

Table 6.2: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 10

	Efficient MQS		**Smarter MQS**	
No of Jobs	**Energy**	**Time**	**Energy**	**Time**
10	**961**	**68**	**468**	**66**

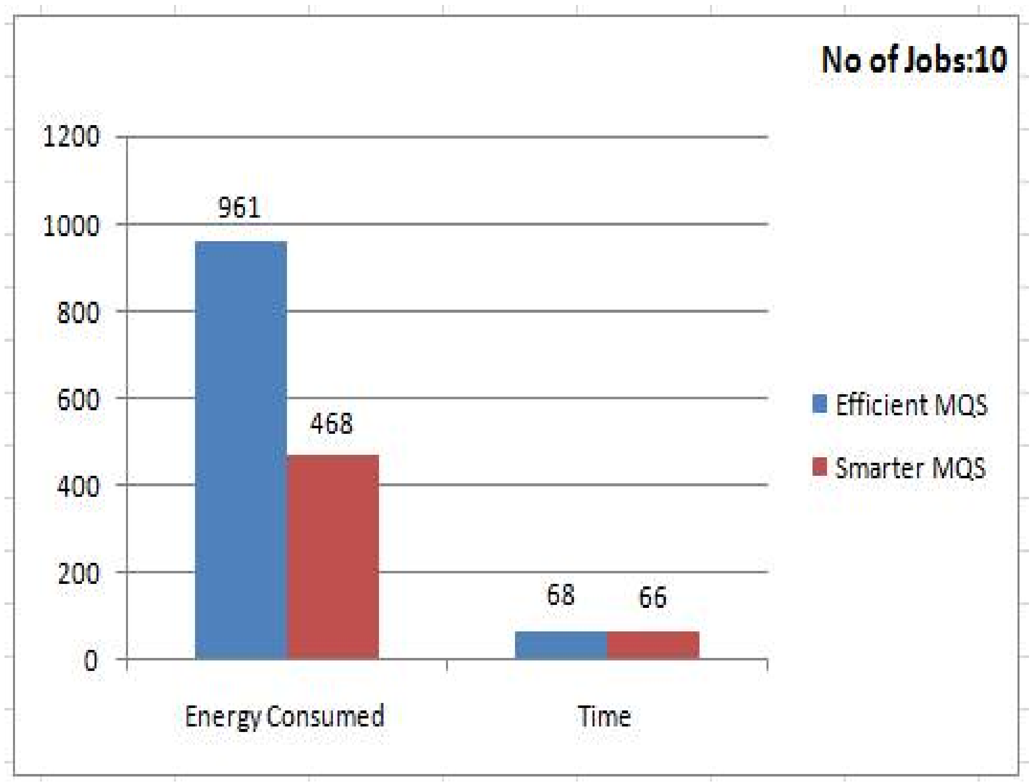

Figure 6.3 Comparison of (Efficient MQS and Smarter MQS) Algorithms on Number of Jobs: 10

The Table 6.3 shows the obtained result values for the two algorithms on number of cloud user jobs taken as 15. Further the plot in Figure 6.4 which is based upon energy consumption and execution time compares the proposed Smarter MQS technique with Efficient MQS algorithm in cloud environment; clearly experimental results show that the proposed Smarter MQS algorithm significantly reduces energy consumption and execution time than the Efficient MQS algorithm.

Table 6.3: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 15

	Efficient MQS		**Smarter MQS**	
No of Jobs	**Energy**	**Time**	**Energy**	**Time**
15	**961**	**68**	**468**	**66**

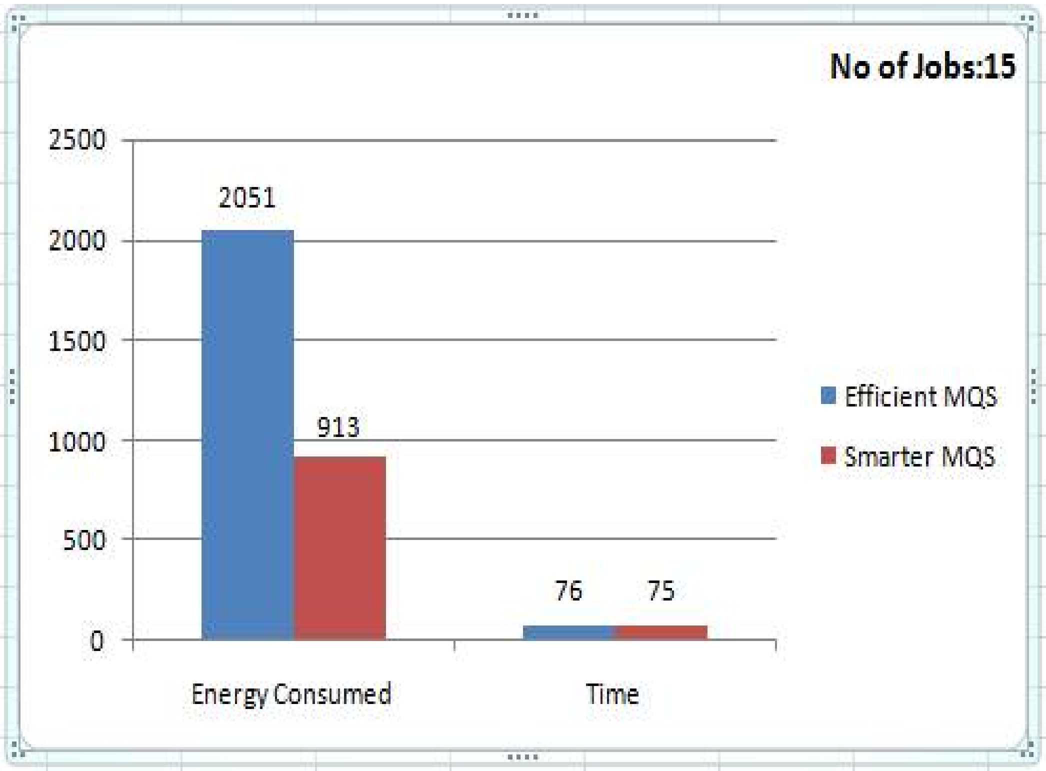

Figure 6.4 Comparison of (Efficient MQS and Smarter MQS) Algorithms on Number of Jobs: 15

The Table 6.4 shows the obtained result values for the two algorithms on number of cloud user jobs taken as 20. Further the plot in Figure 6.5 which is based upon energy consumption and execution time compares the proposed Smarter MQS technique with Efficient MQS algorithm in cloud environment; clearly experimental results shows that the proposed Smarter MQS algorithm works efficiently by reducing energy consumption and execution time than the Efficient MQS algorithm.

Table 6.4: Energy Consumption (in joules) and Time Consumption (in ms) on Number of Jobs – 20

	Efficient MQS		**Smarter MQS**	
No of Jobs	**Energy**	**Time**	**Energy**	**Time**
20	**2528**	**109**	**1486**	**92**

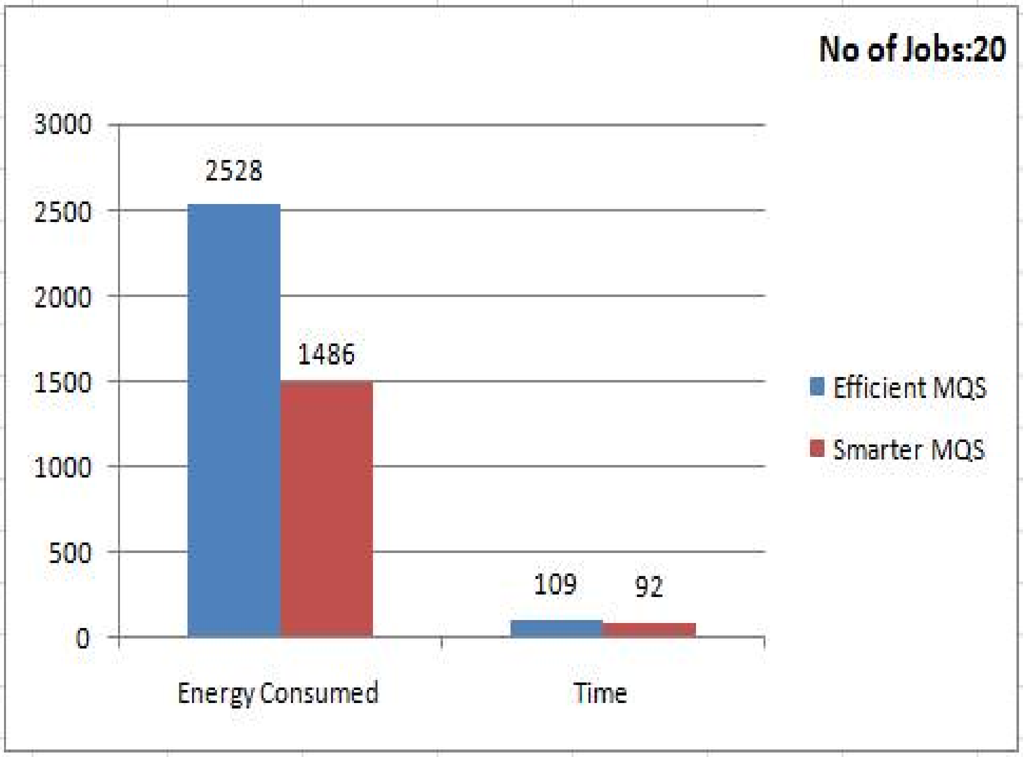

Figure 6.5 Comparison of (Efficient MQS and Smarter MQS) Algorithms on Number of Jobs: 20

The Table 6.4 shows the comparative result values for the two algorithms on number of cloud user jobs 5, 10, 15 and 20 on the basis of energy consumption. Further the plot in Figure 6.6 corresponds to Table 6.5 values illustrates that the proposed Smarter MQS algorithm efficiently scales down the energy consumption than the Efficient MQS algorithm and provide energy efficient framework for scheduling cloud user jobs.

Table 6.5: Energy Consumption (in joules) Comparison of Efficient MQS and Smarter MQS algorithms on Number of Jobs: 5,10,15,20

No of Jobs	Efficient MQS	Smarter MQS
5	336	127
10	961	468
15	2051	913
20	2528	1486

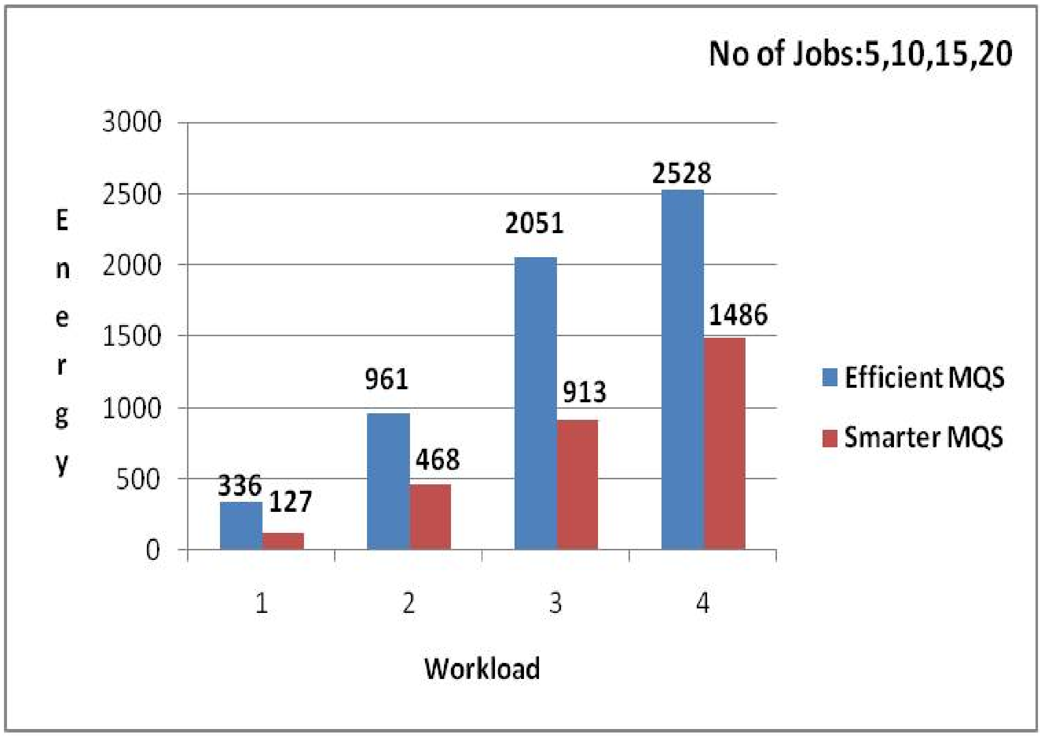

Figure 6.6 Energy Consumption Comparison of Both Efficient MQS and Smarter MQS Algorithms

The Table 6.5 shows the comparative result values for the two algorithms on number of cloud user jobs 5, 10, 15 and 20 on the basis of time consumption. Further the plot in Figure 6.7 corresponds to Table 6.5 values illustrates that the proposed Smarter MQS algorithm efficiently improves execution time than the Efficient MQS algorithm.

Table 6.6: Time Consumption (in ms) comparison of Efficient MQS and Smarter MQS algorithms on Number of Jobs: 5,10,15,20

No of Jobs	Efficient MQS	Smarter MQS
5	**31**	**29**
10	**68**	**66**
15	**76**	**75**
20	**109**	**92**

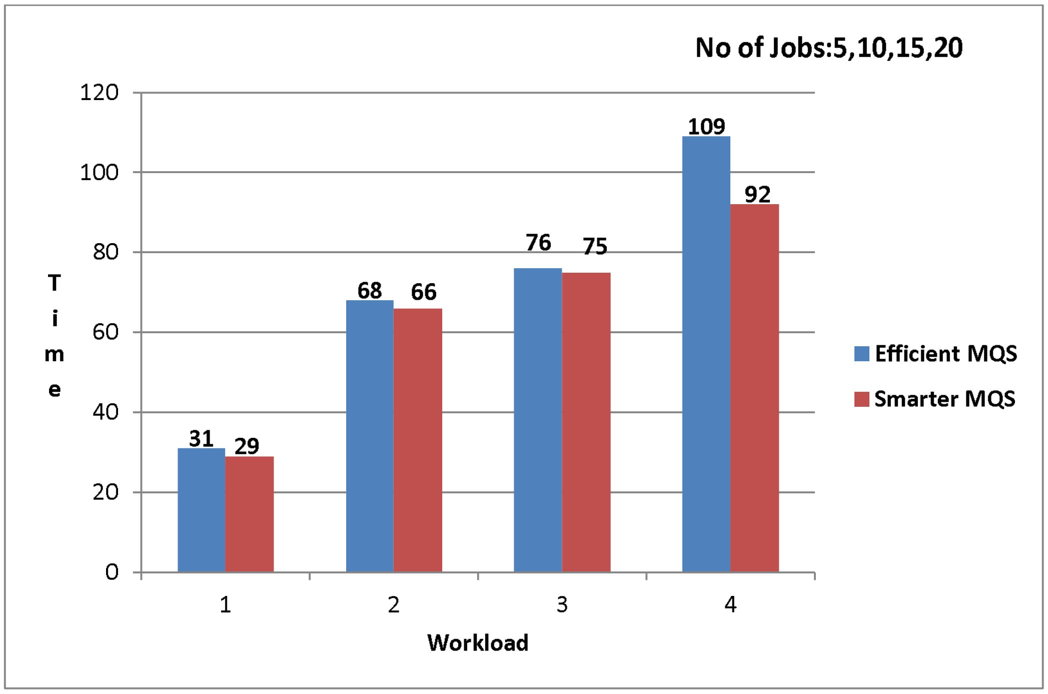

Figure 6.7 Time Consumption Comparison of Both Efficient MQS and Smarter MQS Algorithms

6.6 SUMMARY

This Chapter first discussed pseudo code of proposed algorithm followed by the detail description of the execution environment and at last implementation of proposed job scheduling algorithm (SMQS) for cloud computing environment is presented. The parameters used for the performance metrics are energy consumption and time consumption. Results for SMQS are compared with EMQS algorithm using above stated parameters. It is observed that SMQS provides better energy efficient scheduling than EMQS in each scenario on different number of cloud user jobs.