ISSN (Print) : 0974-6846
ISSN (Online) : 0974-5645
Indian Journal of Science and Technology, *Vol 9(39), DOI: 10.17485/ijst/2016/v9i39/102089, October 2016*

Parallel Job Processing using Collaborative Time-Cost Scheduling for Amazon EC2

S. Nagadevi*

Department of Computer Science and Engineering, SRM University, Kattankulathur - 603203, Chennai, Tamil Nadu, India; nagadevi.s@ktr.srmuniv.ac.in

Abstract

Objective: Cloud Computing is based on the pay per usage model. Amazon EC2 is the public cloud which provides IaaS using this model. Amazon EC2 provides virtual machines to the users. Cost for the use of virtual machines is based on the time for which it is being used. Amazon EC2 charges for partial instance hours even if the instances are idle. To reduce the cost of usage for customers, number of instances and the execution time must be reduced. **Methods**: In this paper we proposed a collaborative time-cost scheduling for parallel job processing. Our method aims to reduce the number of running instances to reduce the cost. As time is proportional to cost, jobs are processed in parallel. We designed a collaborative time-cost scheduling algorithm that selects the most suitable machine to run the job. **Application**: We developed a cloud data storage portal that enables users to upload, download, delete and compress large chunks of data on the fly without the need to download it to a local system and compress it offline. **Findings**: The status of the scheduling job is available to the user in addition to the status of the machine. Our algorithm uses minimum number of instances with no place for instance being idle. The time is reduced due to parallel job processing and cost is also reduced compared to sequential scheduling..

Keywords: Amazon EC2, Collaborative Scheduling, Parallel Processing, Time-Cost, VM Instances

1. Introduction

Amazon provides IaaS to the cloud users on pay per use model. Resources are provided to the users as virtual machines known as instances. Amazon provides Amazon Machine Image (AMI) to instantiate the virtual machine. Each of the instances is having its own configuration for Memory, RAM and Bandwidth etc. Based on the configuration of the VM instances, these are categorized as Small, Large, and Extra-Large as in Amazon EC2. Xen virtualization is adopted in Amazon EC2. Based on the process of provisioning the instances are classified into On-demand, reserved and spot instances[1].

On-Demand Instances are provided for a short time i.e., hours. These instances are not provided for the long time. Any partial hour usages of resources are considered as full hours. In reserved instances the resource capacity is provided for rent with single time payment for the instances. Spot instances are based on the bidding amount of the user.

2. Background

Amazon EC2 allows the users to acquire resources in hourly basis. These instances can be installed and run at any time to process user data[2]. AMI is used to launch an instance. AMI consists of the details about the machine configuration and application. VM instance may undergo running, rebooted, stopped and re-started states after it has been launched. The instances are in active state until

*Author for correspondence

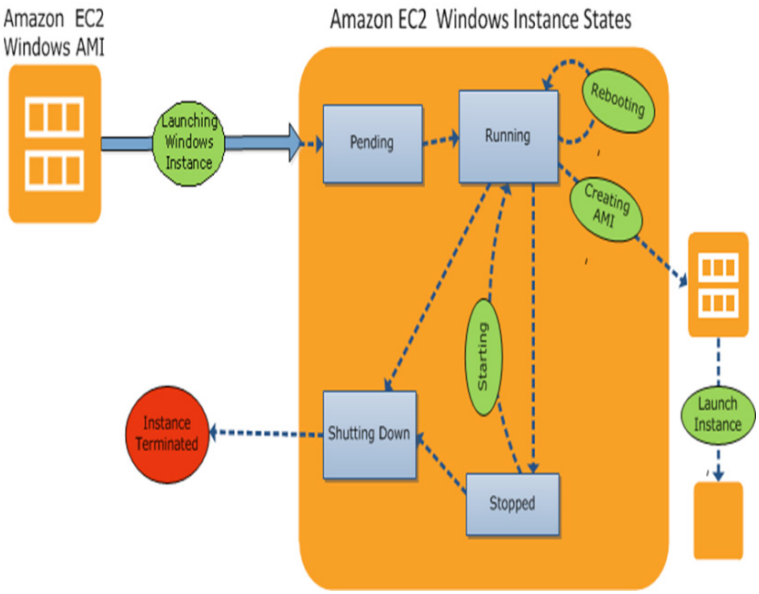

Figure 1. Functional flow of Amazon EC2 instance.

it is shutdown. The shutdown process terminates the instance. Customers are not charged when an instance is not in the running state.

When an instance is stopped, it stops running. But the ID of an instance, EBS volume and the data persists. Any store volumes of an instance is lost. Instance also loses its private and public addresses. But, when an instance is launched through VPC, it maintains the same IP address while stopping and starting an instance. Figure 1 depicts the functional specification of an Amazon EC2 Windows instance launched from an Amazon EC2 Windows AMI. It shows the instance's functional flow through the 4 states of pending, running, stopped, shutting down before getting terminated.

3. System Analysis

3.1 Problem Definition

In cloud, over provisioning and under provisioning of resources may result if the resources are not properly scheduled[3]. In both the cases the users are charged for the resource that they have not used or they may require additional resources to complete the job which leads to more cost. So, new scheduling algorithms are required for the optimal resource allocation. Related study[4,5,6] shows how the cost per user varies with the demand imposed to the application and how the choice for EC2 instance types and configurations reflects on the costs paid by the cloud customer. Our work aims at scheduling the resources on the basis of cost with time aware strategies, as cost is proportional to time in the case of Amazon EC2 instances as they are charged on the basis of usage by hour.

3.2 Existing System

The priority factor of Amazon EC2 instances is its billing cost. As far as the existing system is considered there is no minimum cost, even the instance consumed for partial hours are billed as full hours[7]. Billing stops only when the instance is terminated forever. But by doing so, all the installations done on the instance is lost forever. Stopping the instance (if 8GB EBS) will cost about $0.80 per month for storage (EBS backed data) even if it is never running. The concept of scheduling is not implemented for Amazon EC2 instances on processing data in parallel which results in chances of instances being idle, eventually ending up in resource wastage[8].

Cloud data storages normally used does not facilitate compression of data online. In order to compress it, the user need to download the data to a local system and compress it offline, then the compressed data is again uploaded to the cloud storage rather than compressing data on fly.

3.3 Disadvantages of Existing System

- Lack of scheduling on parallel data processing.
- Partial instance-hours are charged as full hours.
- No minimum cost for the instances.
- Additional costs if instances are not terminated forever.
- Stopping the instance will cost for EBS data storage even if it is never running.
- As the number of instances increases, network issues related to instances also increases.
- Chances of processing nodes or instances being idle.
- Status of the job executed on each instance is unknown to the use.

3.4 Proposed System

A Cost based scheduling based on time aware strategy known as "Collaborative Time-Cost Scheduling" is proposed for Amazon Elastic Compute Cloud instances on processing job in parallel which reduces the instance running cost per hour. Unwanted instances are shut down, with no place for instance being idle. With minimum number of instances, the user submitted job is carried out efficiently keeping the billing cost per hour as the major focus. With reduction in the number of instance utilized, network issues and congestions related to the instance are minimized making it easy for people who depend on quick database calls. With cost based scheduling, keeping billing rates of EC2 instance on close watch, minimum cost for the consumption of instances by hour can be billed overcoming the drawback of existing system in which partial consumption of instance by hour are billed as full hours.

A cloud data storage portal is also implemented in the proposed system as the User Interface (UI) which comprises of data upload, download, deletion and compression. The portal, unlike other existing cloud storage portal allows online compression on the fly with no need to download large chunks of file to compress it offline and then upload it again. It also allows its registered users to maintain an account in the portal where the user's files are stored and are accessed globally from anywhere.

3.5 Advantages of Proposed System

- Cost based scheduling of instances on parallel data processing.
- Partial consumption of instance by hour is billed as partial hour usage only eliminating full hour charges.
- Minimum number of instances is utilized thereby resolving network issues integrating the possibility of reuse of instances.
- No place for instance being idle as with scheduling the compute resources are efficiently utilized.
- A cloud data storage portal with files compressed on fly and accessed globally from anywhere.
- Status of the job executed on each instance is made available to the user.

4. Implementation

4.1 User Interface

User Interface is designed for cloud data backup to a remote instance. The cloud data storage portal can be used for data uploading, downloading, deletion and compression allowing users to create an account in it for storage of chunks of data, which can be globally accessed from anywhere in the world irrespective of geographic locations using the login id and password. Moreover the data stored in the portal can be compressed on fly without the need for downloading it offline and then compressing it. The requests for data compression can be carried out in parallel and is sent to the back end as the user submitted job to the job manager.

4.2 Scheduling Strategy

Once the VM is started, the Job Manager will start its execution. Job Manager receives the job from the user.

Then it communicates with Cloud Controller to control the VM. Cloud Controller allocates and de-allocates VM to the jobs[9].

Job Manager submits the tasks to the Task Manager. Task Manager executes the tasks and reports to Job Manger about the errors. The Job Manager decides on how many instances are required to complete the job and what kinds of instances are required to complete the job. For this purpose, JM utilizes a job scheduler. The responsibility of scheduling the job and deciding in which instance the job has to be run is imposed on the job scheduler. It starts the stopped instances whenever the need arises by instantiating a thread with the help of job manager and allocating the job submitted to the instances.

In Amazon EC2, the whole working of instances is purely based on billing rates, the scheduling has to be carried out with cost as the major focus. Hence a cost based scheduling is carried out to find the most appropriate instance for job allocation. Thereby reducing the cost involved with the usage of instances. Cost based scheduling facilitates on-demand running of the instances with efficient resource utilization.

4.3 Collaborative Time-Cost Scheduling

An efficient cost based scheduling using time aware strategies known as "Collaborative Time-Cost Scheduling " as shown in Table 1 is carried out to find the most eligible instance on to which the job has to be allocated in such a way that the cost charged per usage by hour is minimal for an instance. As time is proportional to cost, the availability time has to be set to maximum. Then the status of the instance is checked. For an instance in stopped state, the average start-up time of the instance is computed. If it is lower or equal to the availability time, then the availability time has to set to average start-up time and get the instance as the most eligible instance. For an instance in stopping state (which has not yet stopped), the time spent on waiting to get stopped has to be also considered before following the scenario for a stopped instance.

For a pending instance, which is not yet started; the remaining start-up time is computed by subtracting the start initiate time from the current time. As there won't be any current running job for a pending instance, the job run time is set to zero .Assuming if there are 2 tasks waiting in the queue maintained for the instance which is now in the pending state, time is computed for how long it would take for the instance to be free i.e., its availability time (the instance would be free only after the completion of the existing jobs waiting in the queue maintained for the instance). Availability time is calculated as the sum of remaining start time and remaining run time. Remaining run time is time taken to complete the tasks waiting for it in the queue. For example, if there are 2 jobs waiting in the queue, each of which takes 5s to start (remaining start time) and 20s for its execution (remaining run time), then the availability time for that instance is (20*2) +5 i.e., 45s. If the remaining run time is less than or equal to the availability time then the availability time is set as the remaining run time and the instance is selected as the eligible instance.

For a running instance, there is no start-up time as it has already started and if there are 2 tasks waiting for the instance in the queue, the remaining run time is the total run time of the tasks for it to be executed. For example, 2 tasks t1and t2 in its queue takes 20s each then the total run time has been computed as 2*20 i.e., 40s. Then the availability time of the instance is computed by considering the fact that if the instance is in running state then a task is also being executed in the instance. If it has been 5s since the task has started its execution, then the availability time becomes 35s. Thus the availability time of a running instance is computed from the current time subtracted from the sum of initiate time and total run time. If the remaining run time is less than or equal to the availability time then the availability time is set as the remaining run time and the instance is selected as the eligible instance. The cost and time comparison for file compression with and without scheduling is shown in Table 2.

With Cost based Scheduling, instances are billed only from the time instances are allocated even though the application is on run. Whereas in without scheduling, the instances are charged on the very time the application is started, as there is no minimum cost for the instances.

4.4 Queue Manager

Once the jobs are scheduled to be run on an instance, a Queue Manager (QM) maintains the scheduled jobs waiting for each instance to get ready or be free in a queue.

Table 1. Collaborative time-cost scheduling algorithm depicting the association of time concept with the running cost of the instance before getting the eligible instance

Set availabilityTime as Max integer value;
Set eligible instance as null;
For each instance in the instanceList; check for its status
if instance status is Stopped;
calculate instance average startup time
if instance average startup time is less than or equal to the availability time
set availability time as the instance average startup time
set the current instance as the eligible instance
end if
end if
else if instance status is Pending
calculate remainingStartupTime
calculate current jobRunTime
calculate remainingRunTime as remainingStartTime + remaining runtime of instance queue - jobRuntime
if remainingRunTime is less than or equal to the availability time
set availability time as the remainingRunTime
set the current instance as the eligible instance
end if
end if
else
instance status is Running
calculate current jobRunTime
calculate remainingRunTime as remaining runtime of instance queue - jobRuntime
if remainingRunTime is less than or equal to the availability time
set availability time as the remainingRunTime
set the current instance as the eligible instance
end if
end else
end for
return eligible instance;

Table 2. Depicts the cost comparison for files on compressing with and without scheduling
*0.020 USD per hour for a t1.micro instance

File Size	Compressed File Size	Waiting Time	Turnaround Time	Instances
1979 kb	1170 kb	48s	11s	i-bfd911d0
1014 kb	927 kb	57s	9s	i-4ee4c527
10592 kb	790 kb	84s	22s	i-bfd911d0

Compressed File Size	Cost without Scheduling *(based on the start of the program)*
1170kb	133.250 cents
927kb	133.972 cents
790kb	134.058 cents

Compressed File Size	Cost with Scheduling
1170kb	(48+11)/(60*60)*0.020*100=0.0327 cents
927kb	(57 + 9)/ (60 * 60) * 0.020*100 = 0.0366 cents
790kb	(84 + 22)/ (60 * 60) * 0.020*100 = 0.0588 cents

Compressed File Size	Time without Parallel processing	Time with Parallel processing
1170 kb	59s	59s
927 kb	125s*(waiting for 1st job to finish)*	66s
790 kb	231s*(waiting for 1st and 2nd job to finish)*	106s

Table 2 Continued

Compressed File Size	Cost saved with Scheduling	Time saved with Parallel processing
1170kb	133.217 cents	0s
927kb	133.890 cents	59s
790kb	133.999 cents	125s

For each instance a separate queue has been maintained by QM. By default, the queue starts out with one worker thread. This is suitable for asynchronous processing of the items in the queue when the time to complete is as important as conserving resources, or when it's essential that items in the queue be processed in the same order as they were entered.

4.5 Task Manager Start-Up Listener

A Task Manager Start-up Listener checks whether the Task Manager (TM) is ready or not. Once the TM is started, a new operation listener has to be created and assigned to the instance.

4.6 Task Manager Operation Listener

A Task Manager Operation Listener checks whether there is any jobs scheduled for an instance is waiting in the queue, maintained by Queue Manager (QM). If there is any pending job for that instance, the job has to be submitted to the instance and waits till it gets executed. Once the pending jobs are completed, Job Manager (JM) is informed about the completion and this process is repeated till all the pending jobs are executed. If there is no pending job for an instance, instruction to shut down the unused instance and Operation Listener thread is given. Thus the wastage of unwanted compute resource such as instances which are nothing but VMs has been eliminated.

4.7 Task Manager Listener

Once the VMs are started, Task Manager Listener is also started automatically. For each task execution, a new thread is initiated for Task Manager Listener, which in turn informs the Task Manager Start-up Listener that has been running fulltime to create a new Task Manager Operation Listener for the instance and further details to Task Manager Listener. It is the responsibility of Task Manager Listener to establish a communication with Operation Listener upon job acceptance which is then submitted to Task Executor for its execution and the executed result is submitted back to Operation Listener, which is in communication with the Task Manager Listener checking whether there is any more jobs scheduled for an instance waiting in the queue, maintained by Queue Manager (QM) or not. If there is any pending job, it has to be submitted to the instance for execution, on completion the Job Manager (JM) is informed and this process is repeated till all the pending jobs are executed.

4.8 Task Executor

The execution of tasks is monitored by the Task Executor. Task Executor maps the tasks to its corresponding instances. Task parallelization is done by Task Executor. Concrete level maps the tasks to its instances. Abstract graph specifies the task level execution of a job.

5. Performance Analysis

Performance of parallel job processing carried out with Collaborative Time-Cost Scheduling on Amazon EC2 is analyzed on the basis of time and cost. In Amazon EC2, instances are charged per usage by hour. Hence the time is proportional to cost as the duration of usage of instances increases cost also increases and so the whole working of instances is purely based on billing rates, thus scheduling

has to be carried out with time and cost as the major focus. Time and Cost comparison for three sample set of data are shown in Figure 2, Figure 3, Figure 4, Figure 5, Figure 6 and Figure 7. The running instances and jobs running on the instances are shown in Figure 8 and Figure 9.

A bar chart depicting fall and rise of time on the basis of parallel processing and without parallel processing has been plotted. On the basis of time consumed in seconds falls to null or 0s when there is no tasks running and becomes equal when only one task is executed. With parallel data processing (simultaneous running of multiple jobs), the time in seconds falls half of that of the one without parallel data processing (in which jobs are run one by

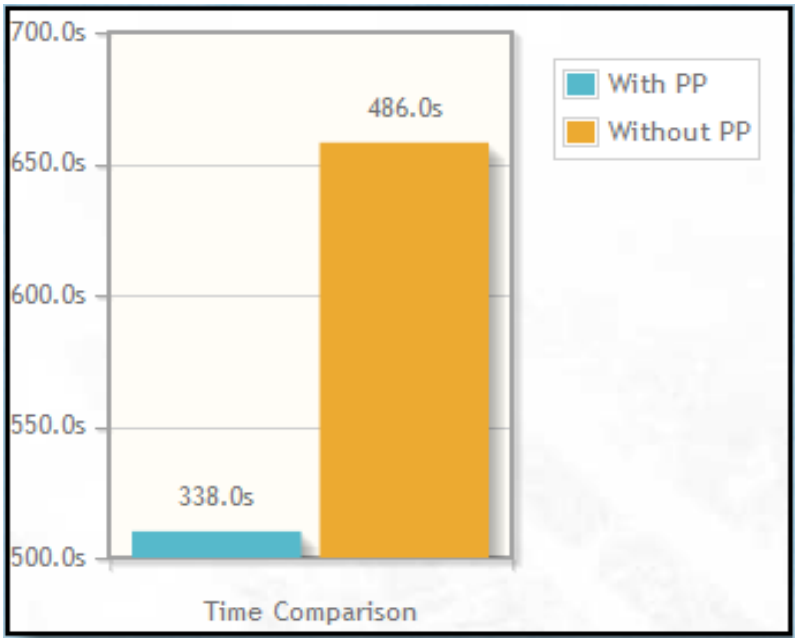

Figure 2. Time comparison with and without parallel processing for Sample 1.

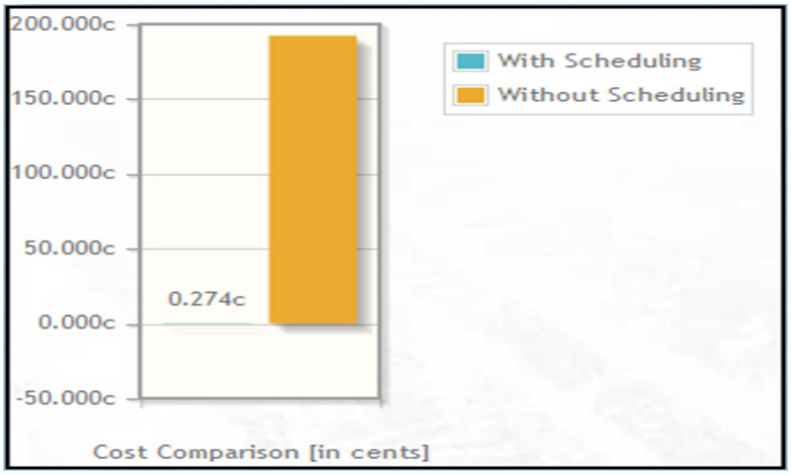

Figure 3. Cost comparison with and without scheduling for sample 1.

one) when there is more than 1 task i.e., 2 or more tasks running.

Average runtime and average start-up time consumed for each instance is calculated on completion of each job executed on the instance in terms of parallel processing and without parallel processing are computed. For a 10 MB job consuming 40s for its completion, average runtime is calculated per MB. As the instances are always running, once the computing has started, the running

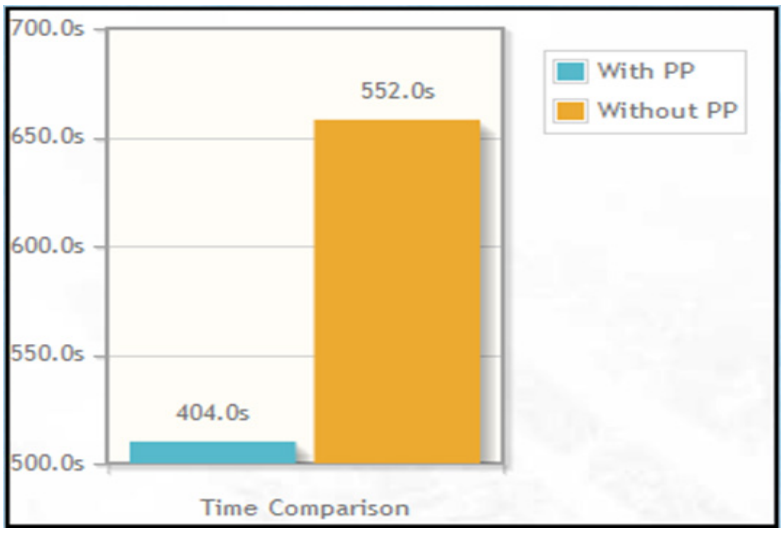

Figure 4. Time comparison with and without parallel processing for sample 2.

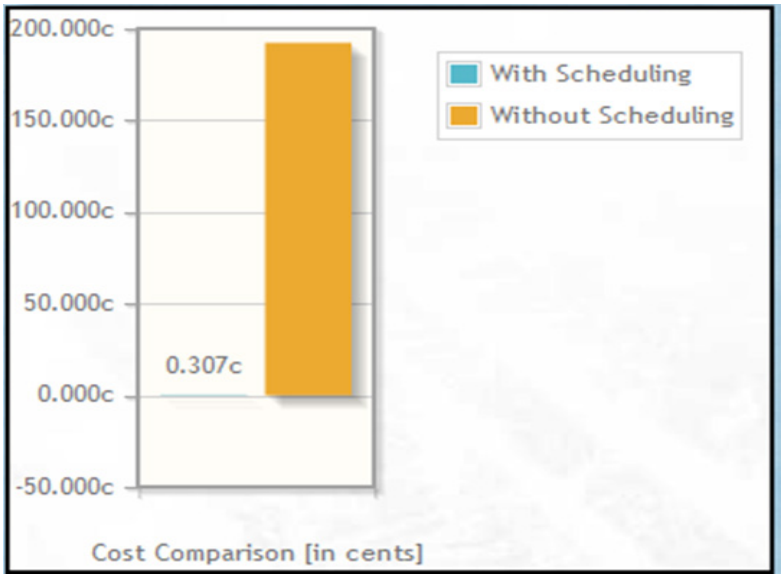

Figure 5. Cost comparison with and without scheduling for sample 2.

time of instance is calculated on the basis of hours of usage of instance along with the cost imposed on the instance.

With Collaborative Time-Cost Scheduling, the most eligible instance on to which the job has to be submitted in such a way that the cost charged for an instance per usage by hour is minimal. It facilitates on-demand start-up of instances. Thus only the run time of each instance has to be computed. If an instance runs for 1min and is shut down for 10 min, then again executed for another 2 min before stopping it, the total running time computed with scheduling is 3 min. But if it is computed without scheduling, the total running time comes around 13

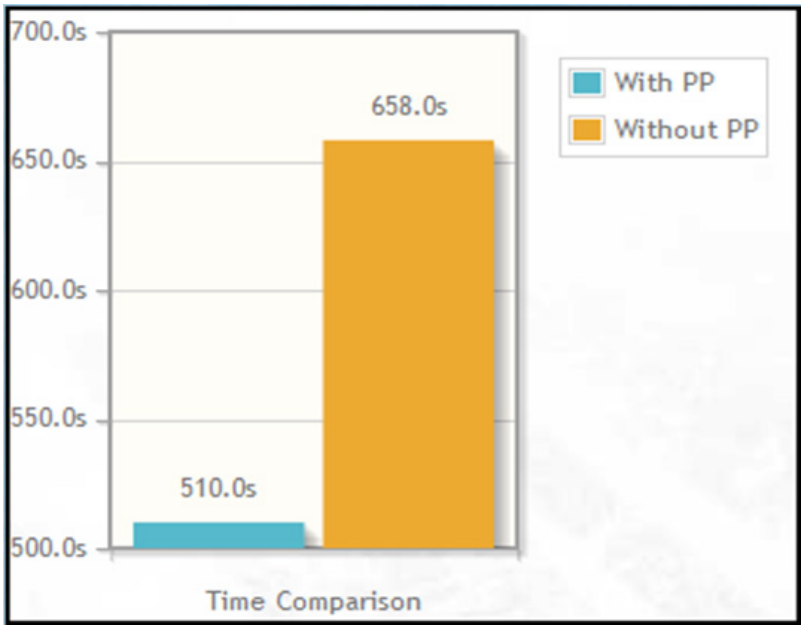

Figure 6. Time comparison with and without parallel processing for sample 3.

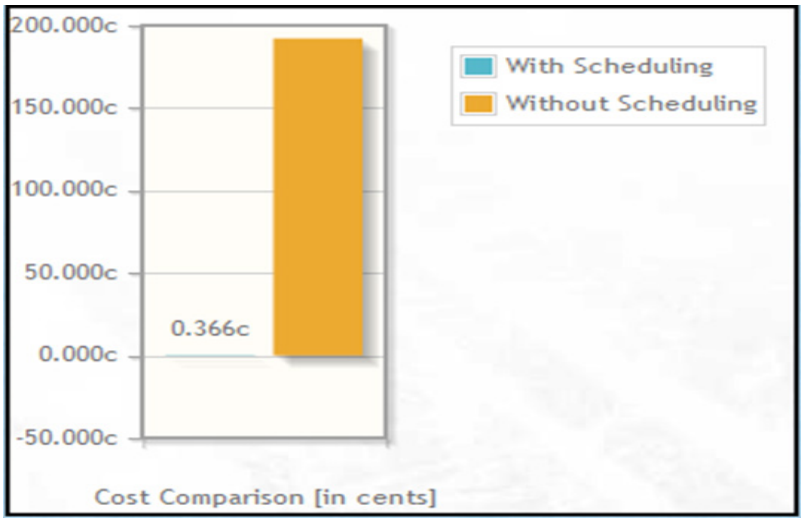

Figure 7. Cost comparison with and without scheduling for sample 3.

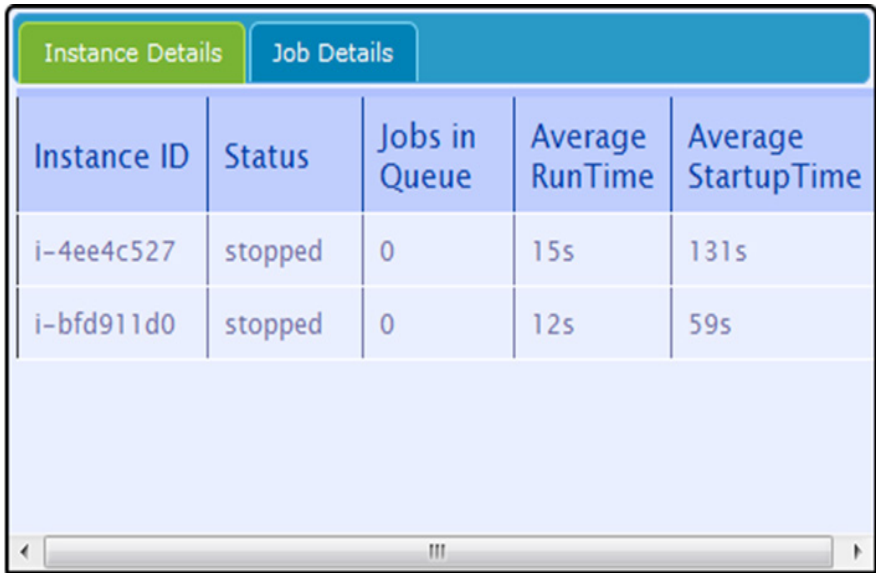

Figure 8. Instance details.

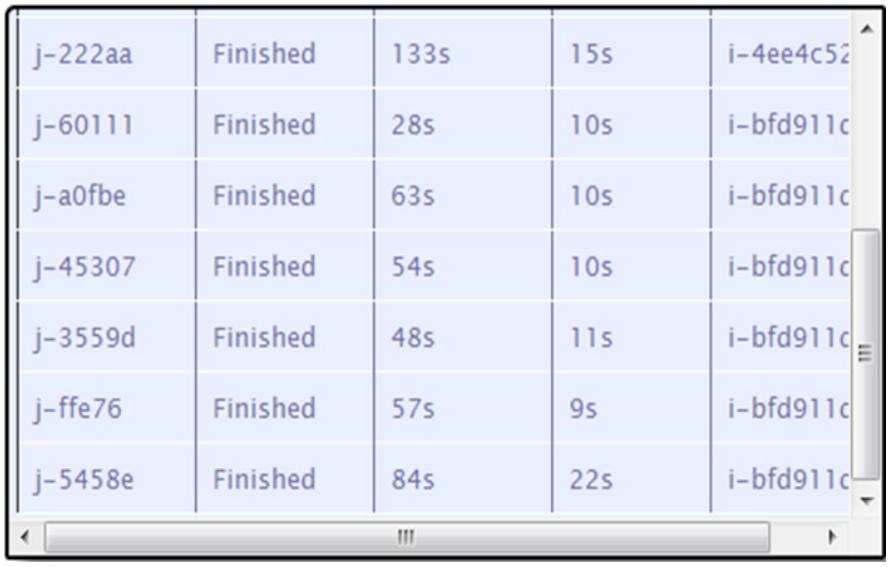

Figure 9. Details of the job running on instances.

min (1 min+10 min+2 min). Thus with scheduling, the instance will be charged only for 3 min, whereas the instance will be charged for total 13 min of running time, if it is considered without scheduling.

With parallel data processing in which jobs are run simultaneously, time taken for instance with longest run time is always on focus If 3 jobs scheduled to be run on different instances takes 2 min, 3 min,5 min each for their completion, the total execution time taken is 5 min. Wheras in the case of one by one execution of job without parallel data processing, the total run time is computed as the sum of run time of each of the 3 instances (2 min+3

min+5 min) i.e., 10 min. As the time increases, cost involved with the instance also increases hence the parallel processing of data ensures minimal cost.

6. Conclusion

It is identified that using the "Collaborative Time-Cost Scheduling" the cost of executing the jobs minimized. It also minimizes the time and number of instances required to complete the job in Amazon EC2.

7. References

1. Amazon EC2 – Virtual server hosting. Available from: https://aws.amazon.com/ec2
2. Amazon EC2 product Details. Available from: https://aws.amazon.com/ec2/details
3. Shyamala K, Rani TS. An analysis on efficient resource allocation mechanisms in cloud computing. Indian Journal of Science and Technology. 2015 May; 8(9):814–21.
4. Tayal S. Task scheduling optimization for the cloud computing systems. International Journal of Advanced Engineering Sciences and Technologies. 2011; 5(2):111–5.
5. Marshall P, Keahey K, Freeman T. Improving utilization of infrastructure clouds. IEEE /ACM International Symposium on Cluster, Cloud and Grid Computing; 2011.
6. Zhang Y, Huang G, Liu X, Mei H. Integrating resource consumption and allocation for infrastructure resources on-demand. IEEE 3rd International Conference on Cloud Computing; 2010.
7. Abirami SP, Ramanathan S. Linear scheduling strategy for resource allocation in cloud environment. IJCCSA. 2012 Feb; 2(1). DOI: 10.5121/ijccsa.2012.2102.
8. Zhong H, Tao K, Zhang X. An approach to optimized resource scheduling algorithm for open-source cloud systems. IEEE Computer Society. 2010.
9. Warneke D, Kao O. Exploiting dynamic resource for efficient parallel data processing in the cloud. IEEE Transactions on Parallel and Distributed System. 2011 Jun; 22.