Ant Colony Optimization for Job Scheduling in Grid with Alea Simulator

Chinmay Joshi
Department of Computer Engineering
A. D Patel Institute of Technology

Siddharth Shah
Department of Computer Engineering
A. D Patel Institute of Technology

Aniruddha K
Department of Computer Engineering
A. D Patel Institute of Technology

Prerak Thakkar
Department of Computer Engineering
A. D Patel Institute of Technology

Gopi Bhatt
Department of Computer Engineering
A. D Patel Institute of Technology

Abstract

Achieving high performance Grid scheduling in heterogeneous computing environment is critical. The Grid scheduling problem is an NP-complete problem. Because of its key importance on performance, the grid-scheduling problem in general has been extensively studied and various heuristics have been proposed. These heuristics are classified into a variety of categories such as job-Scheduling algorithms, Local-search algorithms, Duplication-based algorithms and Random based algorithms. Except for a few, these heuristics are mainly for system with fully connected homogeneous processors. The Ant Colony Algorithm has performed best compared to MET, MCT, OLB, MIN-MIN, MIN-MAX scheduling algorithms [8]. Problem with this algorithm is that it does not consider any resource failure and also does not consider CPU load at runtime. With the comparison of local search algorithms like First Come First Served (FCFS), (EDF) Earliest Deadline First, PBS (Priority Based Scheduling), EDF performs best in new extended gridsim toolkit called Alea[6]. We use an existing Ant Colony Optimization algorithm to perform job scheduling, in Alea Simulator and compare it with FCFS, EDF and PBS Scheduling algorithms.

Keywords- Grid Computing, Job Scheduling, Alea Simulator

I. INTRODUCTION

Grid is a system that provides parallel as well as distributed in the manner of sharing resources, co-ordinate resources and managing the resources [4]. Grid computing is the process of applying the resources to number of computers in a network for a single large-scale problem at the situations like usually to solve scientific or technical problem that requires a more number of computer processing cycles. It involves the use of software that can divide and provide pieces of a program in to thousand computers. It looks as distributed and cluster computing and as a form of network distributed parallel processing.

The basic algorithm "Ant Colony Optimization" has been built upon local search technique of scheduling in grid environment. These algorithms improves the performance like job finishing criteria, high throughput computing, good load balancing and also find feasible solution. This algorithm is performed best in increase the overall make span. The process of Ant Colony optimization is describe in Fig. 1[5]. The next few section discuss the related work and showing comparison results with different job scheduling algorithms.

A. Ant Colony Optimization Algorithm

procedure ACO
begin
Initialize the pheromone
while stopping criterion not satisfied do
Position each ant in a starting node
Repeat
for each ant do
Chose next node by applying the state transition rate
end for
until every ant has build a solution
Update the pheromone
end while
end

II. RELATED WORK

The Ant Colony Optimization algorithm may perform best when compared with other scheduling algorithms like FCFS, and EDF. However, it does not consider the CPU load at runtime.

In all the above mentioned algorithms, the task prioritizing phase considers only communication cost and average computation cost. And based on these, the ranks are assigned to the task in decreasing order. While none of them considers the CPU load at runtime.

It is expected that existing algorithm do these things. Ability to process a large amount of information in a distributed manner. Make decisions about how to allocate individuals for various tasks. Exhibits flexibility and robustness in response challenges.

Also it will expect to speed up the execution time by considering the load on the processor at runtime, i.e., if the number of resource in a level is more than the number of processor than the resource is not assigned.

A. Grid Architecture

The Architecture principles for a Global Grid have to be established on the following principles, which summarized below:
Employ a common network infrastructure
Transport any traffic type
Integration of various transport media
Adaptation to changes
Provide Quality of Service
The architecture of the Grid is often described in terms of "layers", each providing a specific function. In general, the higher layer is user-centric, whereas the lower layers are more hardware-centric (Figure 1).

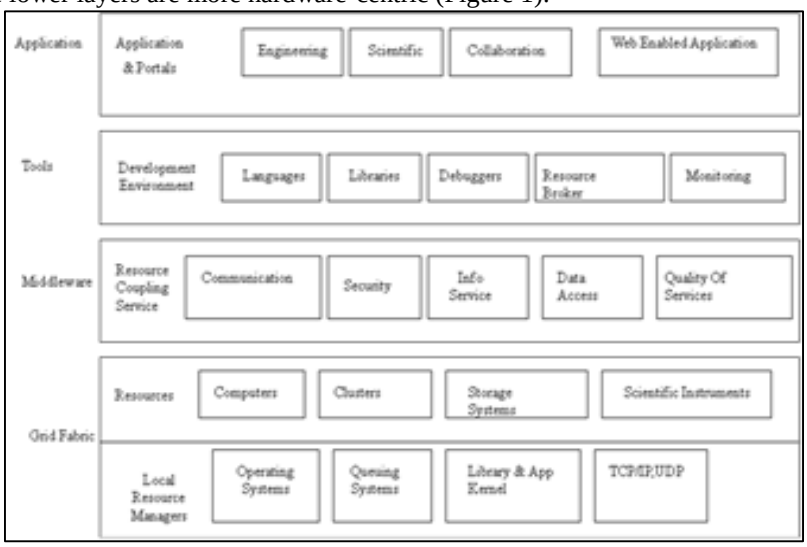

Fig. 1: Layers of Grid Architecture

1) Network Layer

At the bottom is the Network Layer, which assures the connectivity for the resources in the Grid. On top of it lies the Resource Layer, made up of the actual resources that are part of the Grid, such as computers, storage systems, electronic data catalogues, and even sensors such as telescopes or other instruments, which can be connected directly to the network.

2) Middleware Layer:

The Middleware Layer provides the tools that enable the various elements (servers, storage, networks, etc.) to participate in a unified Grid environment. The Middleware layer can be thought of as the intelligence that brings the various elements together.

3) Tool Layer

The Tool Layer is made up of tools that assure connectivity between Middleware and applications. It deals with several languages using a set of libraries to integrate the requests either from lower or higher layers.

4) Application Layer

The highest layer of the structure is the Application Layer, which includes all different user applications (science, engineering, and business, financial), portals and development toolkits supporting the applications. In this layer of the Grid, grid user will "see" and most of the time interacts through their browser. This layered structure can be defined in other ways. For example, the term fabric is often used for all the physical infrastructure of the Grid, including computers and the communication network. Within the

Middleware layer, distinctions can be made between a layer of resource and connectivity protocols, and a higher layer of collective services. However, in all schemes, the Applications Layer remains the topmost layer [10].

B. Simulation in Grid Environment

The Alea simulator is modular, composed of independent entities which correspond to the real world. It consists of the centralized scheduler, the job sub-mission system, and the Grid resources. These entities communicate together by message passing. Currently Grid users are not directly simulated but a job generator attached to the job submission system is used to simulate job arrivals.

The job submission system stores jobs before and after they are executed and communicates with the scheduler to get a scheduling strategy, which it further uses to select a resource to execute a job.

The job generator is attached to the job submission system and it is used to simulate job arrivals. It generates new synthetic jobs that appear during simulation run. The job arrival times correspond to the selected statistical distribution. Currently we support uniform and normal distribution, but it is easy to add new distributions or real workload data.

The scheduler is responsible for schedule generation and further optimization. In the dynamic situation this schedule may change in time as some jobs are already finished while new ones appear. Since it is a standalone entity it has to be able to communicate with other entities, mainly with the job submission system. To keep the scheduler extensible it was designed as a modular entity composed of three main parts. The first part is responsible for communication with the job submission system. The second part stores dynamic information about each Grid resource such as jobs currently being executed or prepared schedule. It also implements functions that approximate make span, tardiness, and other values important for the scheduling process. These information are used by the third part of the scheduler, i.e., by the scheduling algorithms. Each Grid resource is responsible for the job execution. The resource is selected by the scheduler and job is then submitted by the job submission system. Completed jobs are returned to the job submission system [6][7].

III. SCHEDULING ALGORITHMS

A. FCFS (First Come First Served)

FCFS always schedules the first job in the queue, checking the availability of the resources required by such job. If all the resources required by the first job in the queue are available, it is immediately scheduled for the execution, otherwise FCFS waits until all required resources become available. While the first job is waiting for the execution none of the remaining jobs can be scheduled, even if the required resources are available [8].

B. EDF (Earliest Deadline First)

EDF is a dynamic scheduling algorithm used in real-time operating systems. It places processes in a priority queue. Whenever a scheduling event occurs (task finishes, new task released, etc.) the queue will be searched for the process closest to its deadline. This process is the next to be scheduled for execution. EDF is an optimal scheduling algorithm on preemptive uniprocessors, in the following sense: if a collection of independent jobs, each characterized by an arrival time, an execution requirement, and a deadline, can be scheduled (by any algorithm) such that all the jobs complete by their deadlines, the EDF will schedule this collection of jobs such that they all complete by their deadlines [8].

C. ACO (Ant Colony Optimization)

The existing algorithm is work like these. The description of each phase follows:

1) Job Submission: A user submits a job to its local resource node. The jobs each user submits are independent of each other.
2) Ant Invocation: An ant is created and invoked in response to the user's request. The ant is initialized with the job supposed to be scheduled on the Grid.
3) Ant Search: The ant starts searching the Grid to deliver the job to the best suitable node (lightest loaded node) by taking one step a time. Each step consists of leaving one resource and moving to another resource in the Grid. The number of steps each ant takes can either be fixed for all the ants or different for each one.
4) Pheromone Laying: The ants carry the load information of the visited nodes along with themselves. As the ant is moving from one node in the network to another it builds up statistical information about the load of the nodes it has visited in resources.
5) Decision Making: The ants decide which resource to choose for their next step either by looking at the load table information of nodes or they choose a node randomly by the probability of a mutation factor.
6) Job Execution: Finally, the ant delivers the job to a resource and dies. Once the job is completed the answer will be sent back to the original resource [9]. Now here is the algorithm steps need to apply in simulator.

```
Step <-1
initialize ()
while (step < MaxSteps)
currentLoad  <- getResourceLoadInformation()
AntHistory.add(currentLoad)
LocalLoadTable.update()
if random() < MutRate
```

then nextNode = RandomlyChosenStep()
else nextNode = chooseNextStep()
MutRate <- MutRate - DecayRate
step <- step + 1
moveTo(nextResource)
deliverJobToResource()
Now second step is how to choose next step to deliver job to resource
bestResource<- currentResource
bestLoad <- currentLoad
for entry <- 1 to n
if entry.load<bestLoad
then best Resource <- entry.Resource
else if entry.load = bestLoad
if random.next < probability
then best Resource <- entry.Resource

IV. IMPLEMENTATION ENVIRONMENT

The Alea simulator is modular, composed of independent entities which correspond to the real world. It consists of the centralized scheduler, the job sub-mission system, and the Grid resources. These entities communicate together by message passing. Currently Grid users are not directly simulated but a job generator attached to the job submission system is used to simulate job arrivals[6][7]. The detailed work of Alea simulator is describe in Fig 2.

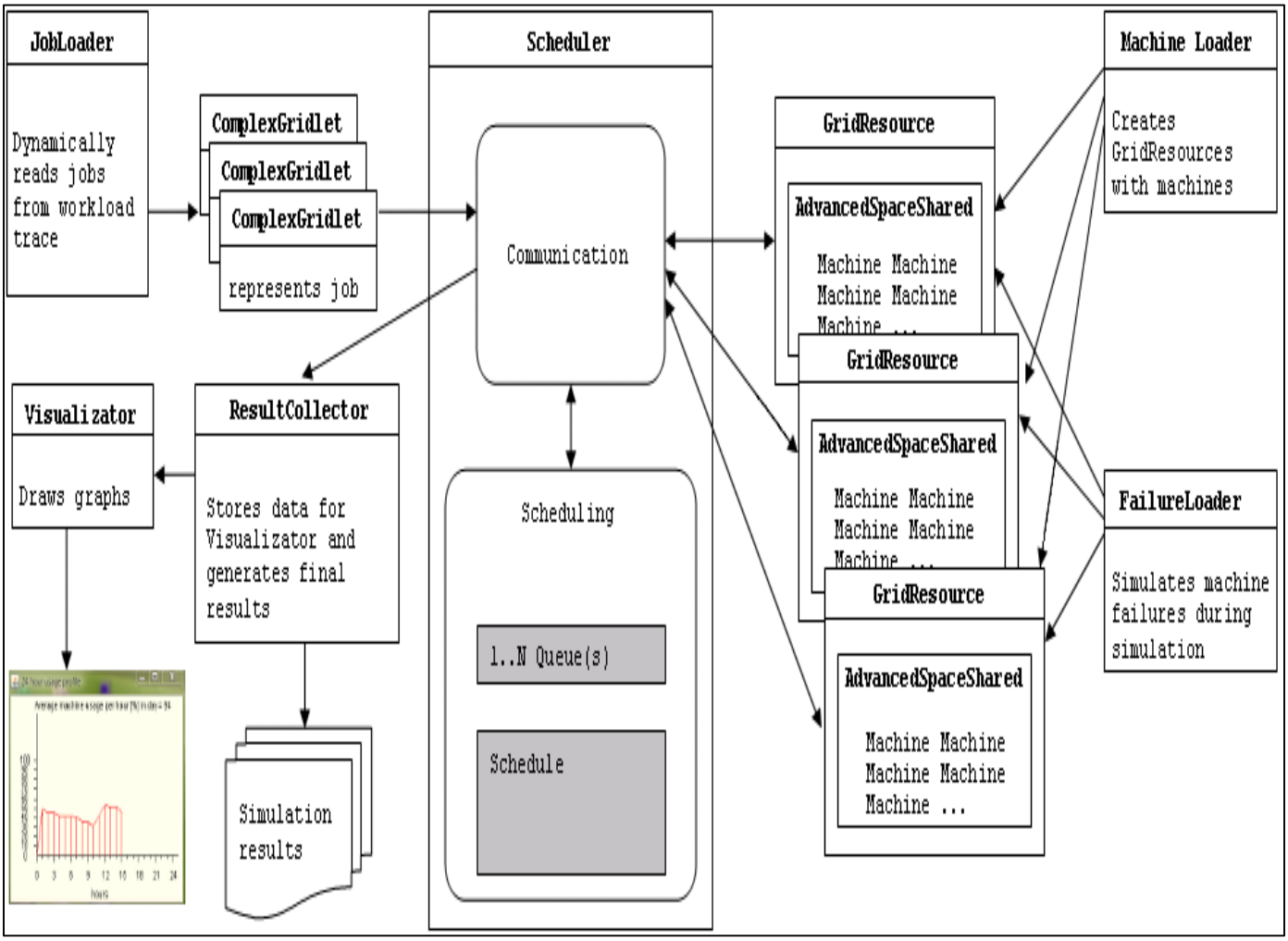

Fig. 2: Working of Alea Simulator

V. PERFORMANCE EVALUATION

This Chapter presents results of the Performance Evaluation of FCFS, EDF and ACO based on various parameters like Job Priority, Computational Length, Tardiness and Release Date. The aim is to compare the results of these algorithms and shows that ACO performed better than FCFS, EDF and PBS.

A. Job Priority

Job priorities determine which competing jobs can access limited resources (such as drives and media). When a job is started, the Job Manager assigns the job a priority number. The lower the job priority number, the higher the priority. The job with the highest priority gets the resources first. The priority of a job is based on the:
Job Priority Number of the operation type, client computer performing the operation, and the type of agent from which the operation originated

Priority Precedence (the weighing of the priority of the client computer relative to the priority of the agent).

B. Description

Max Job descriptions (103656 jobs)
Max Clusters (14 clusters)
Max CPUs (806 CPUs)

1) Example 1

No of Clusters: 6
No of Jobs: 50
Parameters: Job Priority

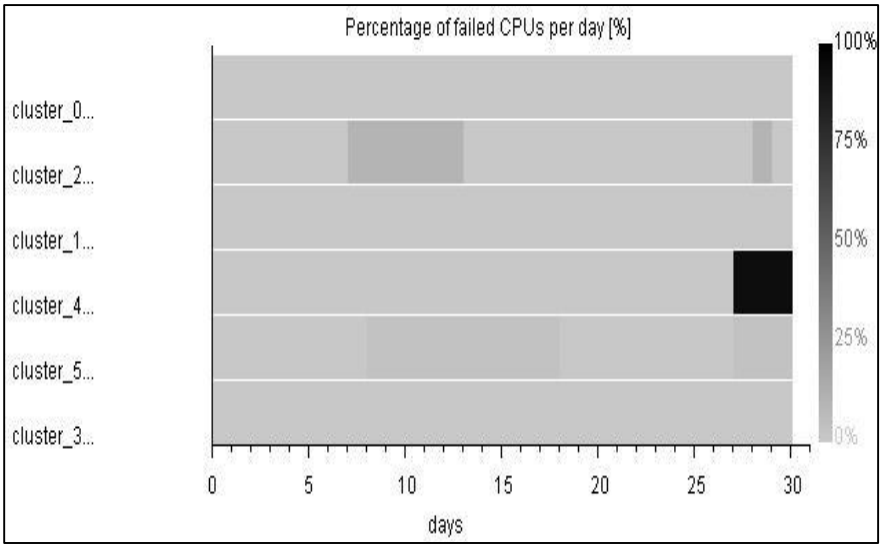

Fig. 3.1: Comparison of ACO, FCFS, and EDF & PBS with respect to Percentage of failed CPUs per day

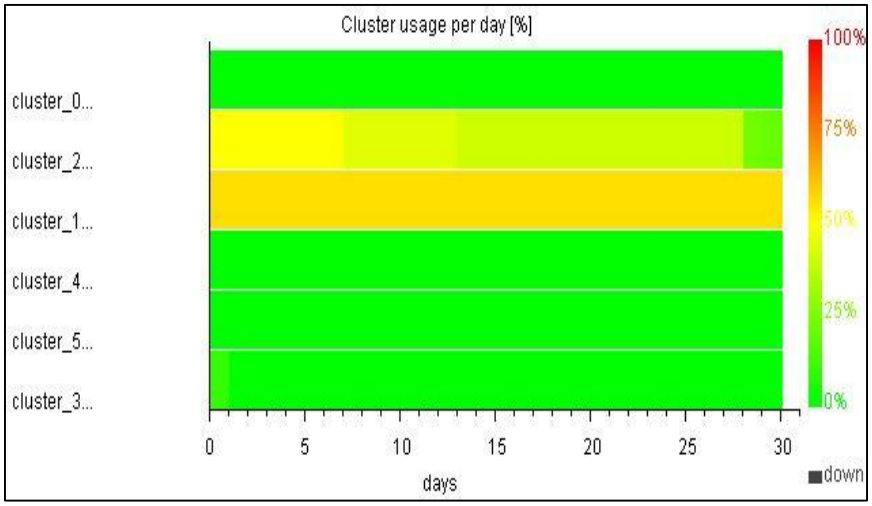

Fig. 3.2: Comparison of ACO, FCFS, EDF & PBS with respect to Number of waiting/running jobs

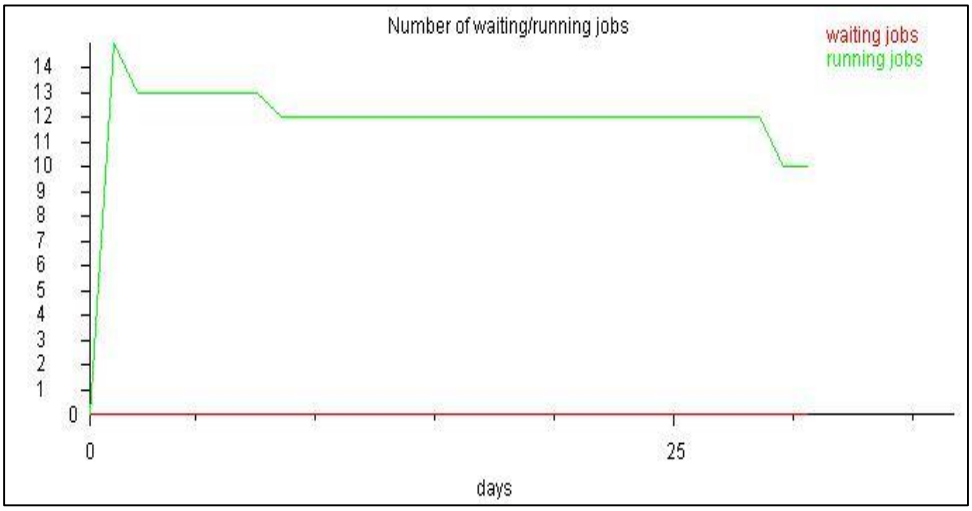

Fig. 3.3: Comparison of ACO, FCFS, EDF & PBS with respect to Number of waiting/running jobs

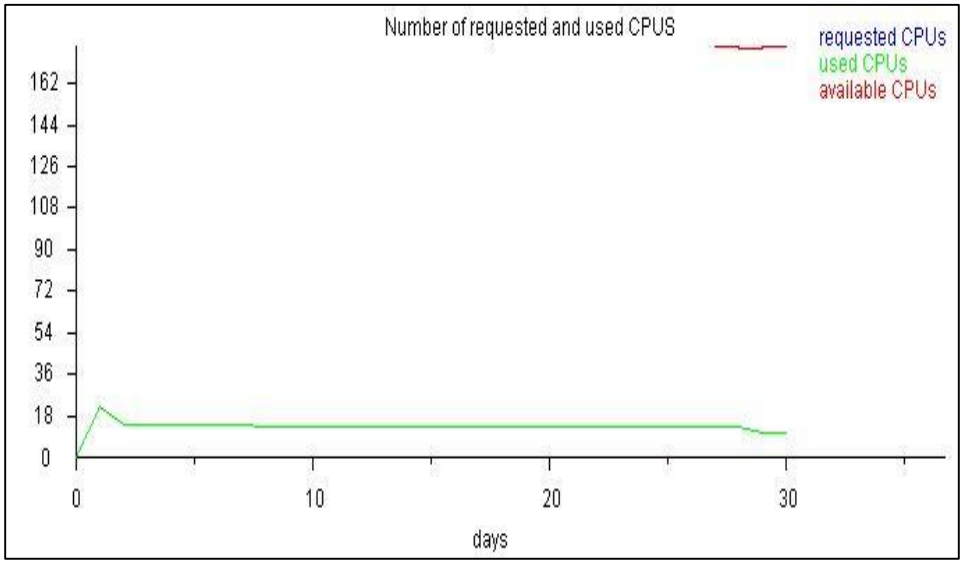

Fig. 3.4: Comparison of ACO, FCFS, EDF & PBS with respect to Number of requested and used CPUs

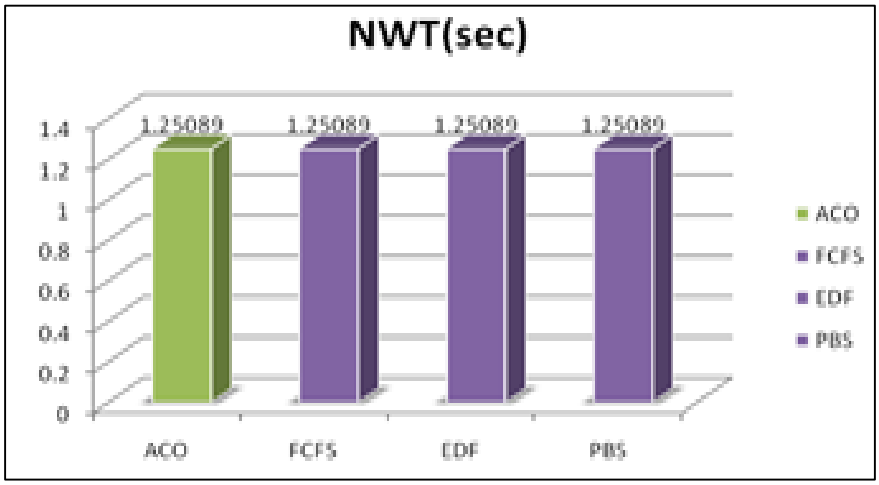

Fig. 3.5: Comparison of ACO, FCFS, EDF & PBS with respect to Net waiting time

C. Summary

For 50 jobs with 6 cluster and with both parameters the result will be same After simulation the result is from result collector, For job scheduling the NWT (net waiting time) of all algorithms = 1.25089sec The total days for job scheduling are 30 which is same.

1) Example 2

No of Clusters: 6
No of Jobs: 1000
Parameters: Job Priority, Release date

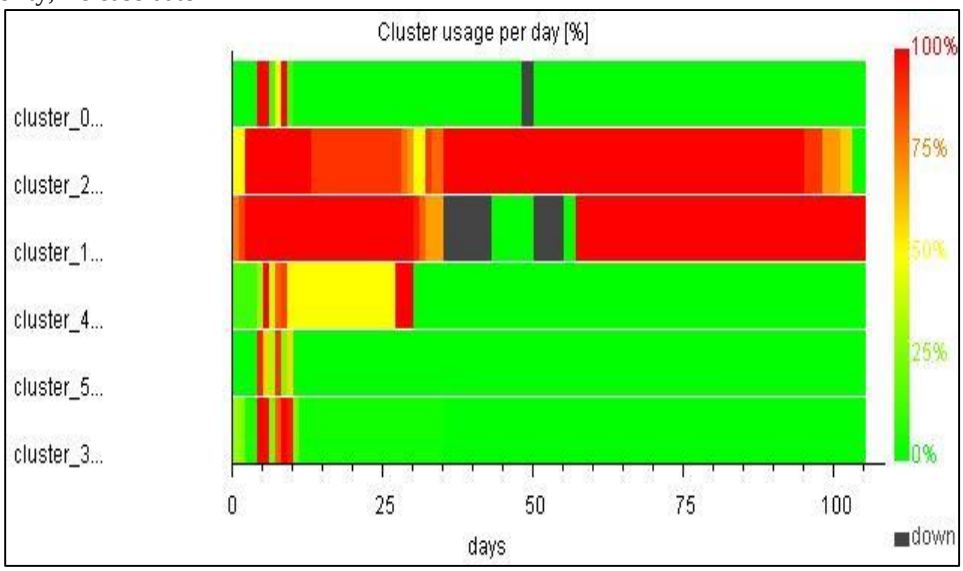

Fig. 3.6 (a): ACO

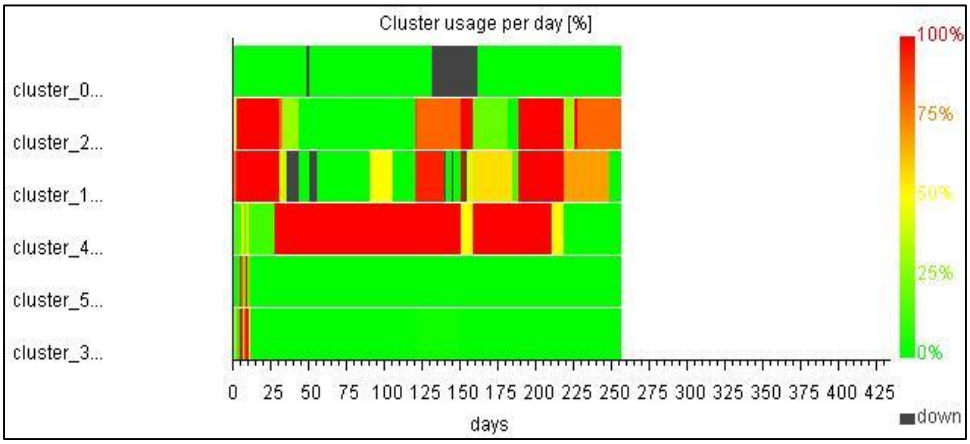

Fig. 3.6 (b): EDF & PBS

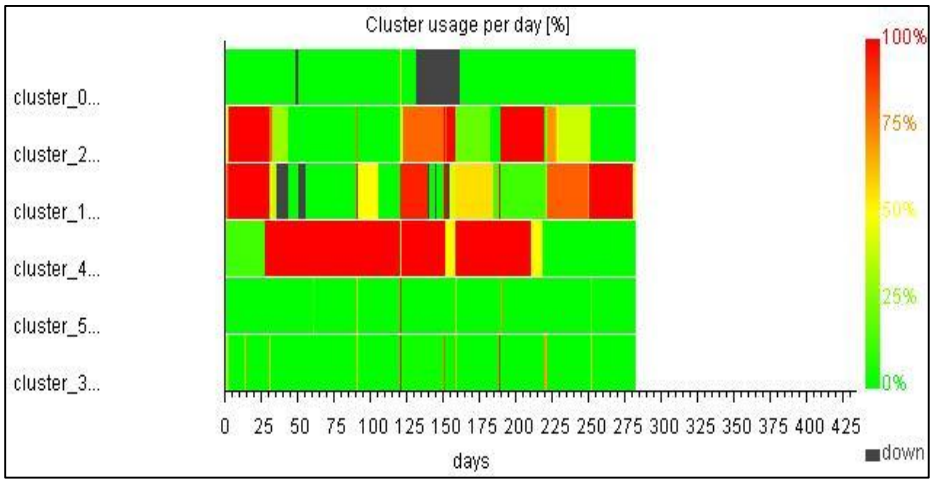

Fig. 3.6 (c): FCFS

Fig. 3.6: Comparison of (a) ACO, (b) EDF & PBS, (c) FCFS. with respect to Cluster usage per day

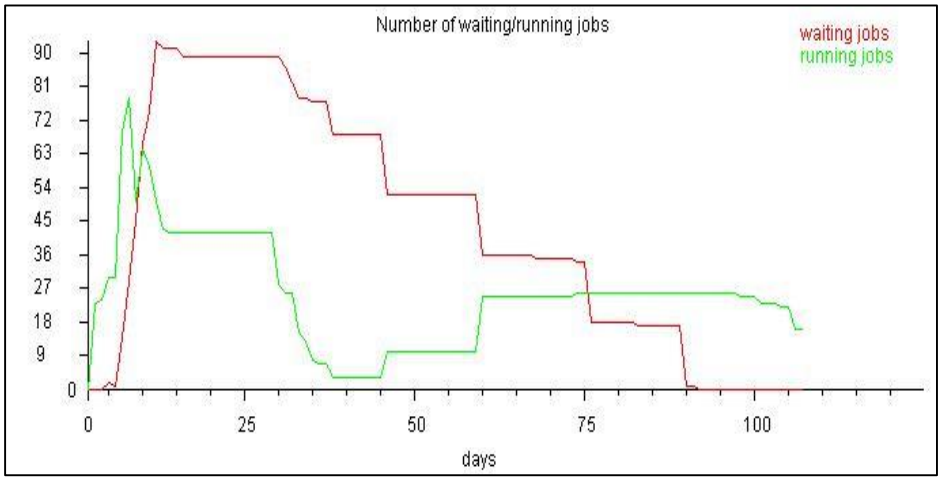

Fig. 3.7 (a): ACO

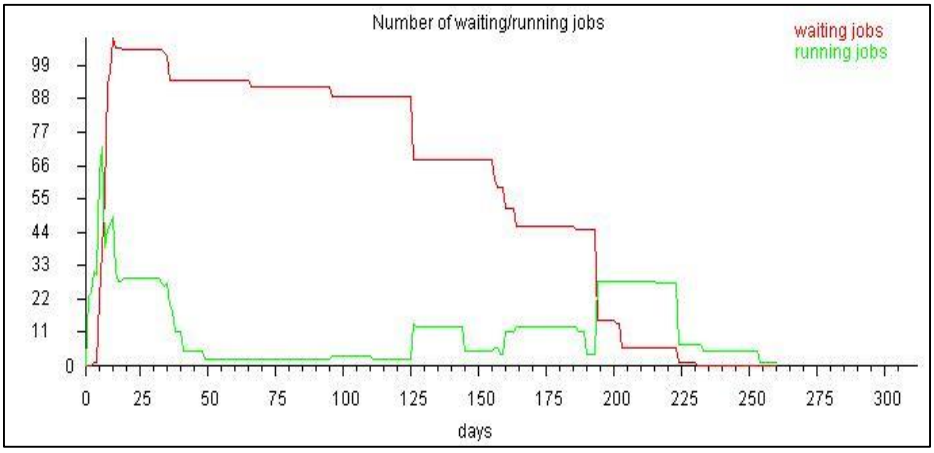

Fig. 3.7 (b): EDF & PBS

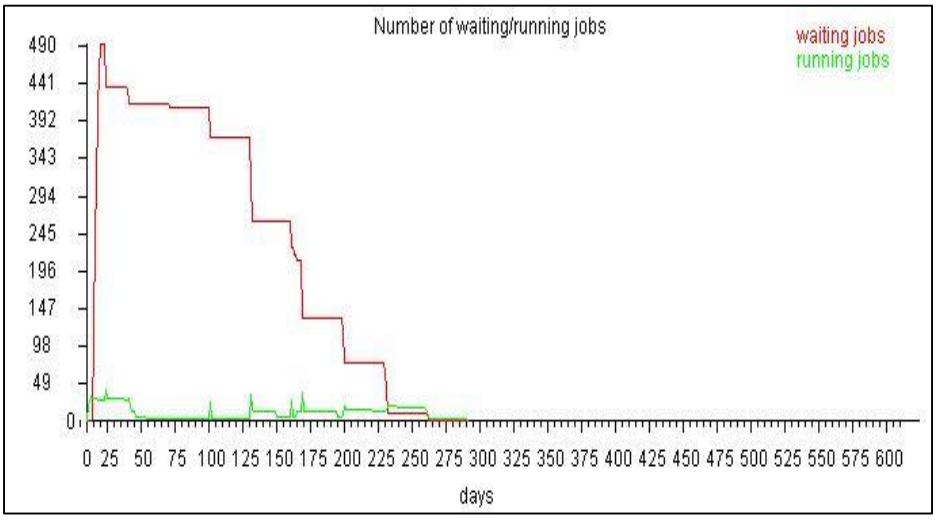

Fig. 3.7 (c): FCFS

Fig. 3.7: Comparison of (a) ACO, (b) EDF & PBS, (C) FCFS with respect to Number of waiting/running jobs

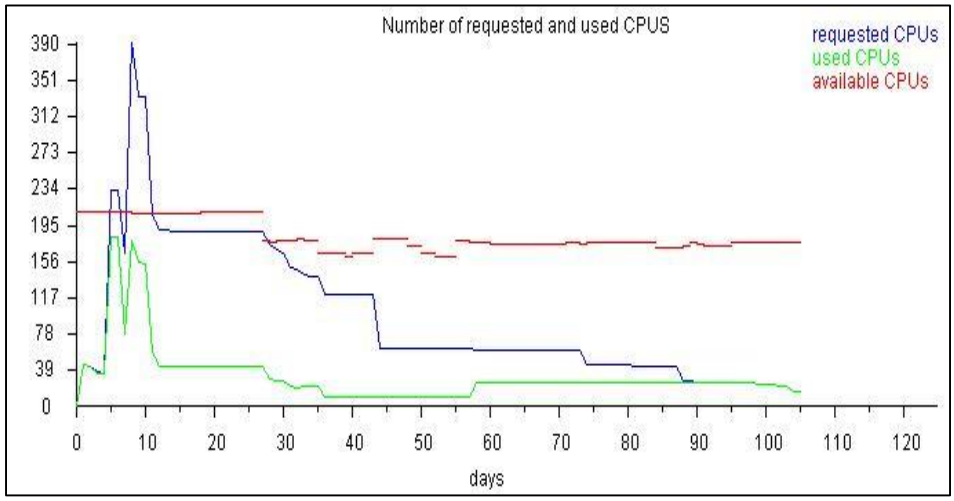

Fig. 3.8 (a): ACO

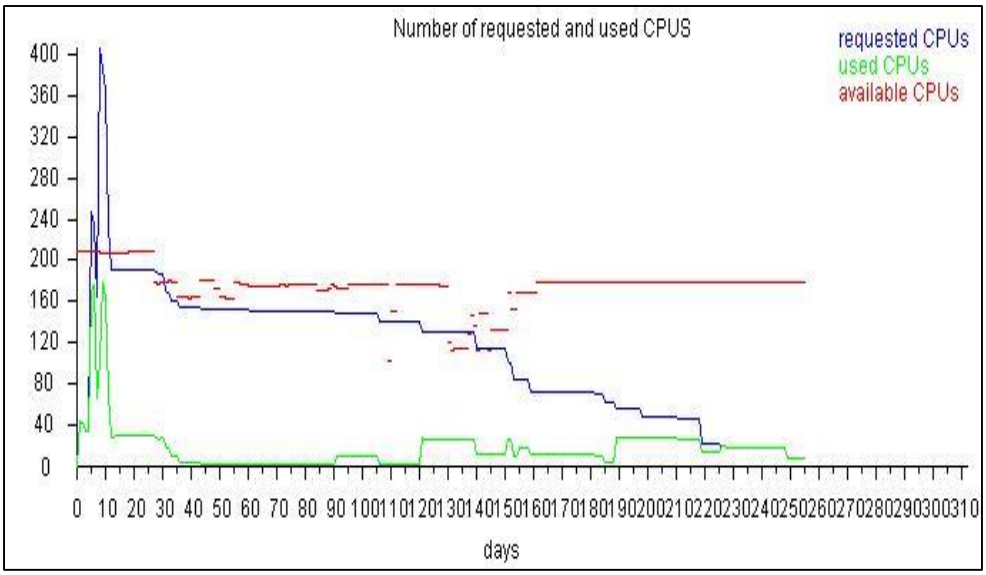

Fig. 3.8 (b): EDF & PBS

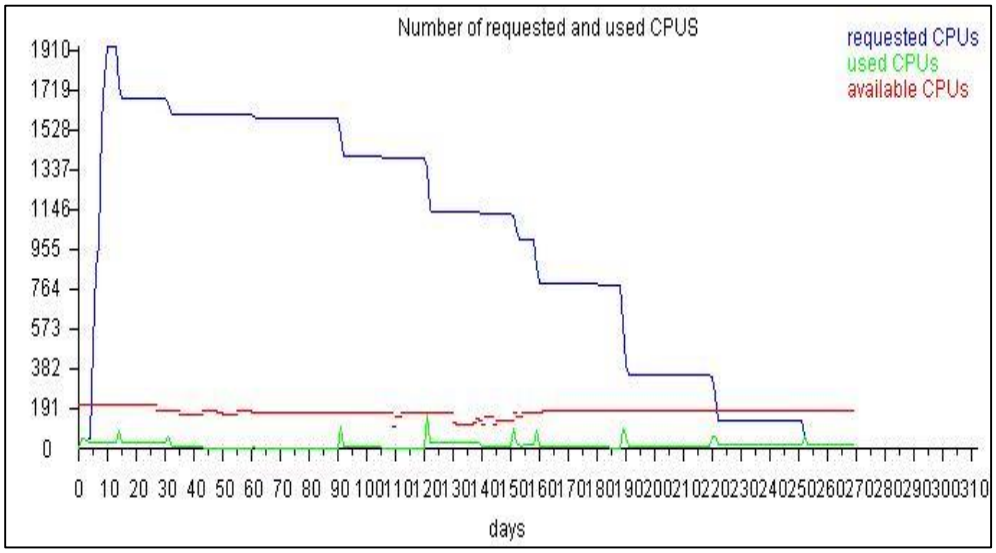

Fig. 3.8 (c): FCFS

Fig. 3.8: Comparison of (a) ACO, (b) EDF, PBS , (c) FCFS with respect to Number of requested and used CPUs

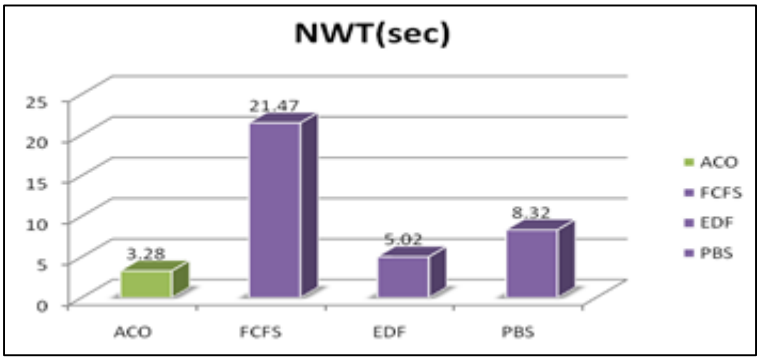

Fig. 3.9: Comparison of ACO, FCFS, EDF & PBS with respect to Net waiting time with job priority

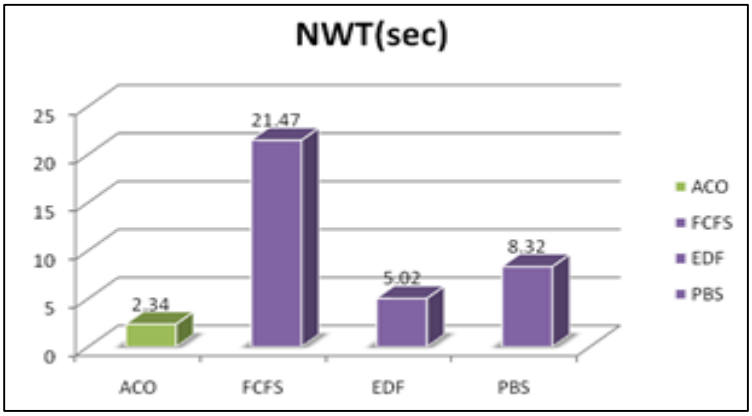

Fig. 3.10: Comparison of ACO, FCFS, EDF & PBS with respect to Net waiting time with release date

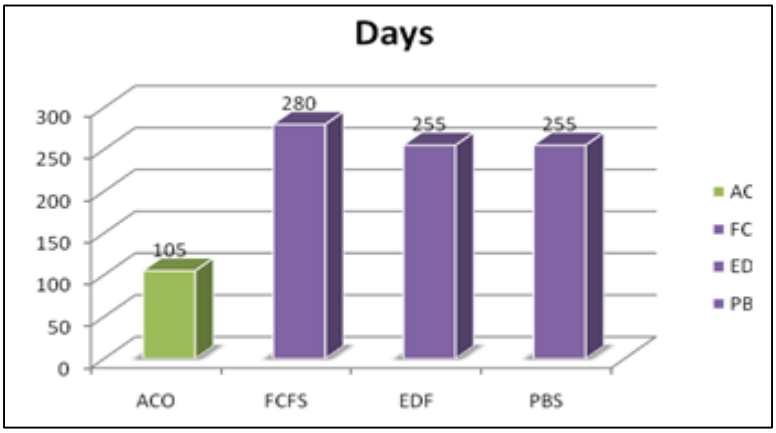

Fig. 3.11: Comparison of ACO, FCFS, EDF & PBS with respect to Number of Days

D. Summary

For 1000 jobs with 6 cluster and with Job both parameters the result will be different
After simulation the result is from result collector, for job scheduling the
NWT (net waiting time) for ACO = 2.34056sec
NWT (net waiting time) for FCFS = 21.47069sec
NWT (net waiting time) for EDF = 5.02877sec
NWT (net waiting time) for PBS = 8.32135sec
The no of days for job scheduling with ACO = 105 days
The no of days for job scheduling with FCFS = 280days
The no of days for job scheduling with EDF & PBS = 255days
So we can say that ACO is better performed then FCFS, EDF and PBS.

1) Example 3

Comparison of all algorithms for Net Waiting Times and Days with 14 clusters:

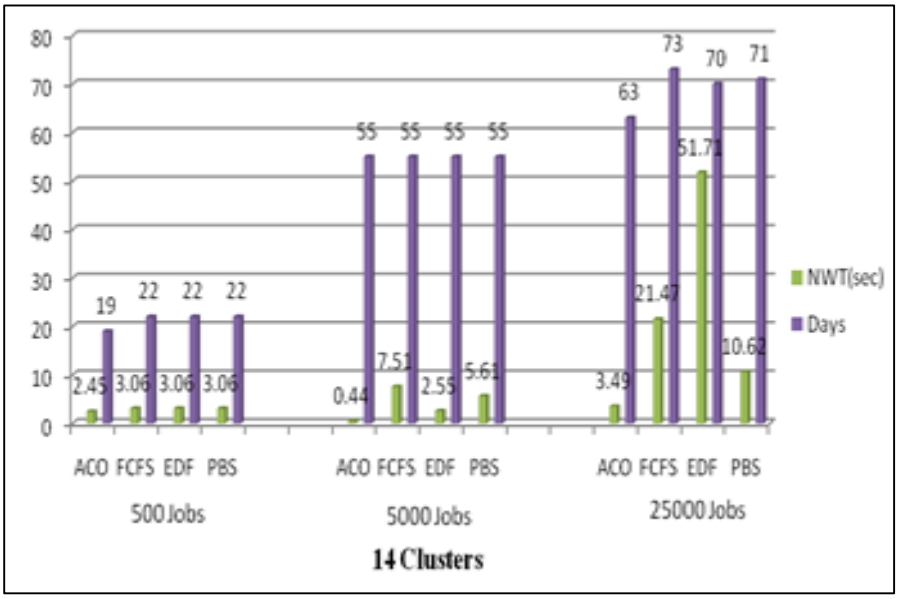

Fig. 3.12: Comparison of all algorithms for Net Waiting time and Days with 14 clusters

Summary: With 14 clusters and no of different jobs the result shows that if the number of jobs increase than simulation time and date is increase. From the final results we can say that ACO performes best when compare with FCFS, EDF & PBS

E. Final Results Summary with All Clusters

From the final results of NWT and Days with all clusters we can say that ACO, FCFS, EDF & PBS, if number of clusters are decreases than the result with ACo for NWT is low but number of days are increase and number of clusters are increases than the result for NWT is low and number of days are also decrease compare to FCFS, EDF & PBS. So ACO performed best in every case when compared to these algorithms.

VI. CONCLUSION

In this research we have investigated the use of Ant Colony technique in designing distributed grid job scheduling algorithm. The Ant Colony Optimization improve the overall local search performance, speed up the execution by decreasing Net Waiting Time and Days, and balancing the potential load on each processor when compared with existing algorithms like FCFS, EDF and PBS.
Final Results of NWT for ACO, FCFS, EDF and PBS with all clusters:
Average Net Waiting Time for EDF= 19.10sec
Average Net Waiting Time for FCFS = 10.66sec
Average Net Waiting Time for PBS = 6.43sec
Average Net Waiting Time for ACO = 2.12sec
Final Results of Number of Days for ACO, FCFS, EDF and PBS with all clusters:
Average Number of Days for EDF = 48days
Average Number of Days for FCFS = 52days
Average Number of Days for PBS = 49days
Average Number of Days for ACO = 45days

VII. FUTURE WORK

In future further improvements of job scheduling simulator. The Alea currently supports centralized scheduling approach. Its extension allow decentralized and multi-level scheduling, which is common for the large scale Grids. Some future extensions Implements other large scale scheduling algorithms. Performance evaluation of Alea simulator in Grid environment. Increase overall performance with different Gridsim parameters.

REFERENCES

[1] F. MAGOULES, J. PAN, K-A TAN AND A. KUMAR, Introduction to Grid Computing, CRC Press, 2009.
[2] GLOBUS ALLIANCE. Globus. Web Published, 2007. Available online at: http://www.globus.org/ (accessed August 3rd, 2010).
[3] Kousalya.K and Balasubramanie.P, Anna University, Kongu Engineering College, Tamilnadu, India, IJCSNS International Journal of Computer Science and Network Security, VOL.8 No.4, April 2008

[4] Penka Martincová, Michal Zábovský University of Zilina gement Science and Informatics, penka.martincova@fri.uniza.sk, in year 4/2007
[5] Stefka Fidanova and Mariya Durchova IPP – BAS, Acad. G. Bonchev, bl.25A, 1113 Sofia, Bulgaria stefka@parallel.bas.bg, mabs@parallel.bas.bg
[6] Dalibor Klusá˘cek Faculty of Informatics, Masaryk University Botanická 68a Brno, Czech Republic xklusac@fi.muni.cz
[7] Hana Rudová Faculty of Informatics, Masaryk University Botanická 68a Brno, Czech Republic hanka@fi.muni.cz
[8] R. Shakerian, S. H. Kamali, M. Hedayati, M. Alipour/ TJMCS Vol .2 No.3 (2011) 469-474
[9] Using Swarm Intelligence for Distributed Job Scheduling on the Grid A Thesis Submitted to the College of Graduate Studies and Research in Partial Ful_llment of the Requirements for the degree of Master of Science in the Department of Computer Science University of Saskatchewan Saskatoon By Azin Moallem c Azin Moallem, 03/2009.
[10] The Anatomy of the Grid Enabling Scalable Virtual Organizations * Ian Foster •¶ Carl Kesselman § Steven Tuecke • {foster, tuecke}@mcs.anl.gov, carl@isi.edu