Study of Resource Allocation with Optimize Backfilling using Arbitrary Search in Grid Computing

Sejal Yaduwanshi[1], Prof. Avinash Sharma[2]

[1]*PG Scholar*, [2]*Head and Associate Professor*

[1,2]*Dept. of CSE, MITS, Bhopal*

Abstract- Grid computing, one of the most inventive phrase used in IT, is emerging vastly distributed computational paradigm. A computational grid provides a collaborative environment of the hefty number of resources capable to do high computing performance to reach the common goal. Grid computing can be called as super virtual computer, it ensemble large scale geographically distributed heterogeneous resources. Resource allocation is a key element in the grid computing and grid resource may leave at anytime from grid environment. Despite a number of benefits in grid computing, still resource allocation is a challenging task in the grid. This work investigates to maximize the profits by analyzing how the tasks are allocated to grid resources effectively according to quality of service parameter and gratifying user requisition. A method of schedule based backfilling load balancing with gap search optimization algorithm has introduced to answer the above raised question about the resource allocation problem based on grid user requisition. The result of proposed method of SBBFP-GS algorithm ameliorates the grid resource allocation.

Index Terms: Grid Computing, Resource Allocation, Heterogeneous, Backfilling Load Balancing, Optimization.

I. INTRODUCTION

A computational grid is a large scale, heterogeneous collection of autonomous systems, geographically distributed and interconnected by heterogeneous networks. A grid can be defined as a large-scale geographically distributed hardware and software infra-structure composed of heterogeneous networked resources owned and shared by multiple administrative organizations which are coordinated to provide transparent, dependable, pervasive and consistent computing support to a wide range of applications. These applications can perform distributed computing, high throughput computing, on-demand computing, data-intensive computing, collaborative computing or multimedia computing.

Grid computing is a collection of computer resources from different geographical location to achieve a common goal. Computation grid uses network and combines computational resources from different geographical locations for distributed jobs. Job sharing (computational burden) is one of the major difficult tasks in a computational grid environment. Grid resource manager provides the functionality for discovery and publishing of resources as well as scheduling, submission and monitoring of jobs. However, computing resources are geographically distributed under different ownerships each having their own access policy, cost and various constraints.

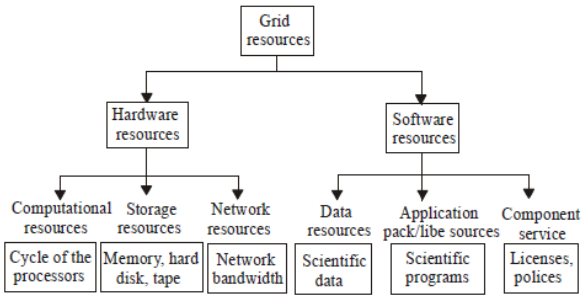

Figure 1: Type of resources in a grid computing

II. LITERATURE WORK

B.Priya et. al, 2019, Grid Computing developed as a wide scale circulated framework to offer powerful planned assets sharing and elite figuring. Network arranges the assets that are not liable to incorporated control. It utilizes standard open, broadly useful conventions and its interfaces. Matrix conveys non-trifling characteristics of administration, for example, the reaction time, throughput, accessibility and security. Network Computing is being received in different regions from scholarly, industry examination to government use. Lattices are turning out to be stages for superior and conveyed figuring. Lattice registering is the cutting edge IT framework that vows to change the manner in which associations and people Compute, impart and work together. The objective of Grid figuring is to make the dream of a straightforward yet enormous and amazing self-overseeing virtual PC out of a huge assortment of associated heterogeneous frameworks sharing different blends of assets. Booking is a bunch of decides and arrangements that control the request by which different positions are executed in a framework.

Massimiliano Caramia et. al, 2018, Grid planning, that is, the designation of circulated computational assets to client applications, is one of the most testing and complex errand in Grid registering. The issue of distributing assets in Grid planning requires the meaning of a model that permits neighborhood and outer schedulers to impart so as to accomplish a productive administration of the assets themselves. To this point, some monetary/market-based models have been presented in the writing, where clients, outside schedulers, and neighborhood schedulers haggle to improve their targets. In this paper, we propose a delicate/contract-net model for Grid asset portion, demonstrating the connections among the included entertainers. The exhibition of the proposed market-based methodology is tentatively contrasted and a cooperative allotment convention.

P. Keerthika et. al, 2017, Grid Computing gives consistent and versatile admittance to wide-region dispersed assets. Since, computational network shares, chooses and totals wide assortment of topographically disseminated processing assets and presents them as a solitary asset for tackling enormous scope registering applications, there is a requirement for a planning calculation which considers the different necessities of matrix climate. Subsequently, this exploration proposes another booking calculation for computational lattices that considers load adjusting, adaptation to internal failure and client fulfillment dependent on the network design, asset heterogeneity, asset accessibility and occupation attributes, for example, client cutoff time. This calculation decreases the makespan of the timetable alongside client fulfillment and adjusted burden. A reproduction is led utilizing Grid Simulator Toolkit (GridSim). The reenactment results shows that the proposed calculation has better makespan, hit rate and asset usage.

III. PROBLEM IDENTIFICATION

Load balance is also an important issue in grid environment. The purpose of load balancing is to balance the load of each resource in order to enhance the resource utilization and increase the system throughput. The main objective of load balancing methods is to speed up the execution of applications on resources whose workload varies at run time in unpredictable way. Hence, it is significant to define metrics to measure the resource workload. Every dynamic load balancing method must estimate the timely workload information of each resource. This is key information in a load balancing system where responses are given to following questions:

1. How to measure resource workload?

2. What criteria are retaining to define this workload?

3. How to avoid the negative effects of resources dynamicity on the workload.

4. How to take into account the resources heterogeneity in order to obtain an instantaneous average workload representative of the system?

V. PROPOSED METHODOLOGY

On the basis of above mentioned activities happened we retrieve the current load information and the initial load information from the database. Initial Status collect the information about all connected nodes like resource entry and job entry. Computation is done after the initial status The basic algorithm of proposed work SBBFP-GS is as follows:

[M'] = SBBFP-GS (Q, M)

// Q is the queue for incoming jobs

// M is a map between jobs and resource nodes

// M' is the updated allocation map

Step 1: Initialize the status of all nodes (N).

Step 2: Initial status=.Previous

Step 3: While jobs=N and N>0 do

Step 4: if Current state is ready to change then

Step 5: Current = Get change state (); //Computation stage

Step 6: Threshold = generate threshold (upper bound, lower bound); //Load Balancing

Step 7: Now Random Search Optimization technique apply on Q.

7.1 Initialize random search parameter θ_0 on the basis of current state and threshold value, initial point $X_0 \subset Q$ and iteration index k=0.

7.2 Generate a collection of candidate points $V_{k+1} \subset Q$ according to a specific generator and associated sampling distribution.

7.3 Update X_{k+1} based on the candidate points V_{k+1}, previous iterates and algorithmic parameters. Also updates algorithm parameters θ_{k+1}.

7.4 If a stopping criterion is met then stop, otherwise increment k and return to step 7.2.

Step 8: Through Gap Search, find the optimized job set in queue Q.

Step 9: Get first job j from Q.

Step 10: while $j \neq null$

Step 11: $N_j \leftarrow$ the number of nodes required by j;

Step 12: $N_{idle} \leftarrow$ the number of idle nodes;

Step 13: **if** $N_j \leq N_{idle}$ then remove j from Q and dispatch it to any N_j idle nodes;

Updates M accordingly;

if j is not at the head of Q then

insert j into $Q_{backfill}$;

else

$N_{backfill}$<- the number of nodes running jobs arriving later than j;

if $N_j \leq (N_{backfill} + N_{idle})$ then

Suspend jobs in $Q_{backfill}$ that arrive later than j and move then back to Q

According to descending order of their arrival time until the number of

Idle node is greater than N_j;

Remove j from Q and dispatch it to N_j idle nodes;

Update M;

Step 14: j<- get the next job from Q and goto step 10.

Step 15: Go to step 3 with decrement list of nodes N.

Step 16: M' = M;

Step 17: Finally obtain the updated jobs and resource node allocation map with required load balancing.

VI. RESULTS AND ANALYSIS

We can compare different load balancing technique with our proposed technique. On the basis of load balancing with scheduling method, FCFS (First Come First Serve) and EASY Backfilling used as an existing technique and Schedule Based Backfilling with Gap Search (SBBFP-GS) used as a proposed method. There are four parameters are here for compare results of different approaches.

(1) Average Machine Usage

(2) Job Cluster Uses/Day

(3) Number of Requested and Used CPU

(4) Number of Waiting and Running Jobs

DAYS	FCFS	EASY BACKFILLING	PROPOSED TECHNIQUE (SBBFP_GS)
0-5	13.2	12.8	13.2
5-10	40.6	40.4	42.4
10-15	2.2	1.8	2.2
15-20	45.5	41.6	45
20-25	45.6	44	50
25-30	52	52.4	54.8
30-35	61.4	61	65
35-40	32.4	33	34
40-45	28.6	30	30.8
45-50	57.4	58	62
50-55	60.4	64.2	65.4
55-60	34	34.2	35

Table 1: Average Machine Usage (in Percentage) with Different Load Balancing Strategies

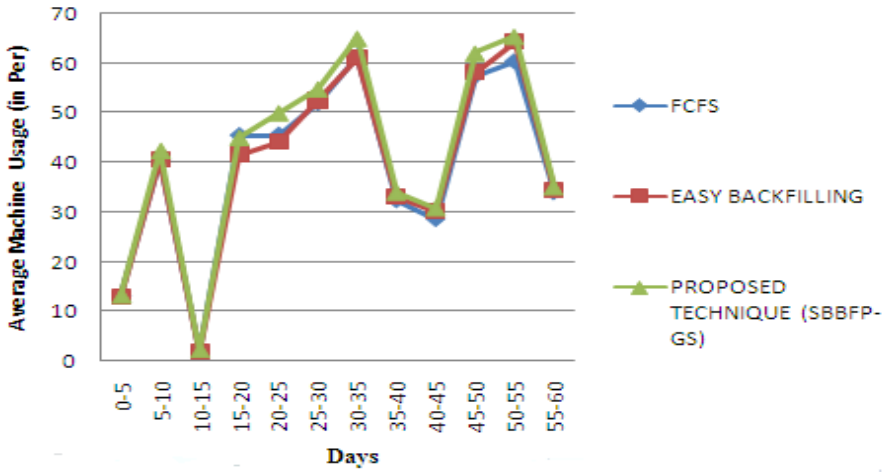

Figure 2: Average Machine Usage (in Percentage) with FCFS, EASY Backfilling and Schedule Based Backfilling Policy with Gap Search Strategies

As per above graph, proposed technique schedule based backfilling policy with gap search has more utilize machine resources like CPU and other equipments as compare than FCFS and EASY Backfilling method.

DAYS	FCFS	EASY BACKFILLING	PROPOSED METHOD(SBBFP_GS)
0-5	0.6	0.8	0.6
5-10	40	40	40
10-15	1.8	1.8	2
15-20	35.6	31.6	32
20-25	35.2	36.8	34.5
25-30	14.6	15	10.5
30-35	39.8	35.8	35
35-40	12.2	12.6	9.6
40-45	21.4	21.6	20.8
45-50	42.6	44.4	44
50-55	29.8	31	30
55-60	23.8	28.2	23.4

Table 2: Job Cluster Uses per Day (in Percentage) with Different Load Balancing Strategies

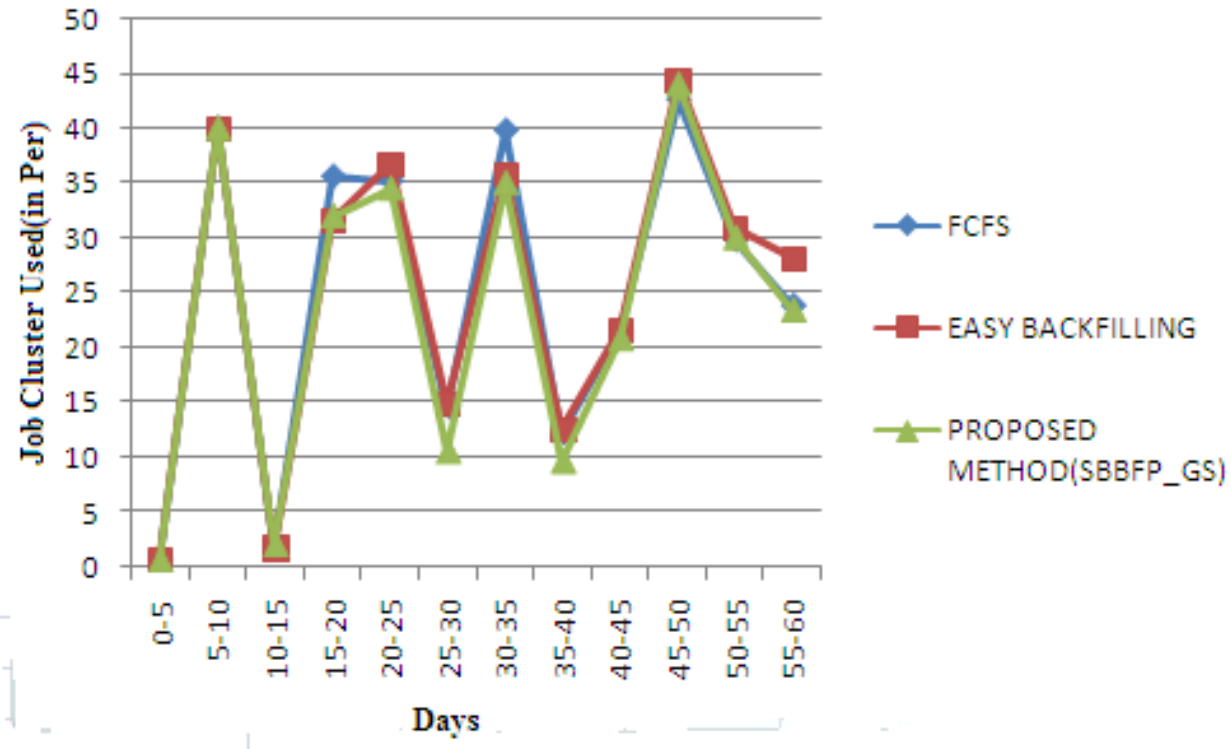

Figure 3: Job Cluster Uses per Day (in Percentage) with FCFS, EASY Backfilling and Schedule Based Backfilling Policy with Gap Search Strategies

As per above graph, proposed technique schedule based backfilling policy with gap search has more uses job clusters as compare than FCFS and EASY Backfilling method.

DAYS	FCFS		EASY BACKFILLING		PROPOSED METHOD (SBBFP-GS)	
	Requested	Used	Requested	Used	Requested	Used
0-5	0	10	0	20	0	25
5-10	310	480	300	431	551	484
10-15	0	20	0	20	0	24
15-20	0	430	0	439	0	448
20-25	333	454	264	426	0	475
25-30	0	209	0	149	0	244
30-35	370	522	363	504	432	544
35-40	0	165	0	175	0	192
40-45	0	270	198	276	0	294
45-50	280	501	280	507	492	545
50-55	330	461	375	451	372	484
55-60	1150	306	1023	340	0	390

Table 3: Number of Requested and Used CPU (in Percentage) with Different Load Balancing Strategies

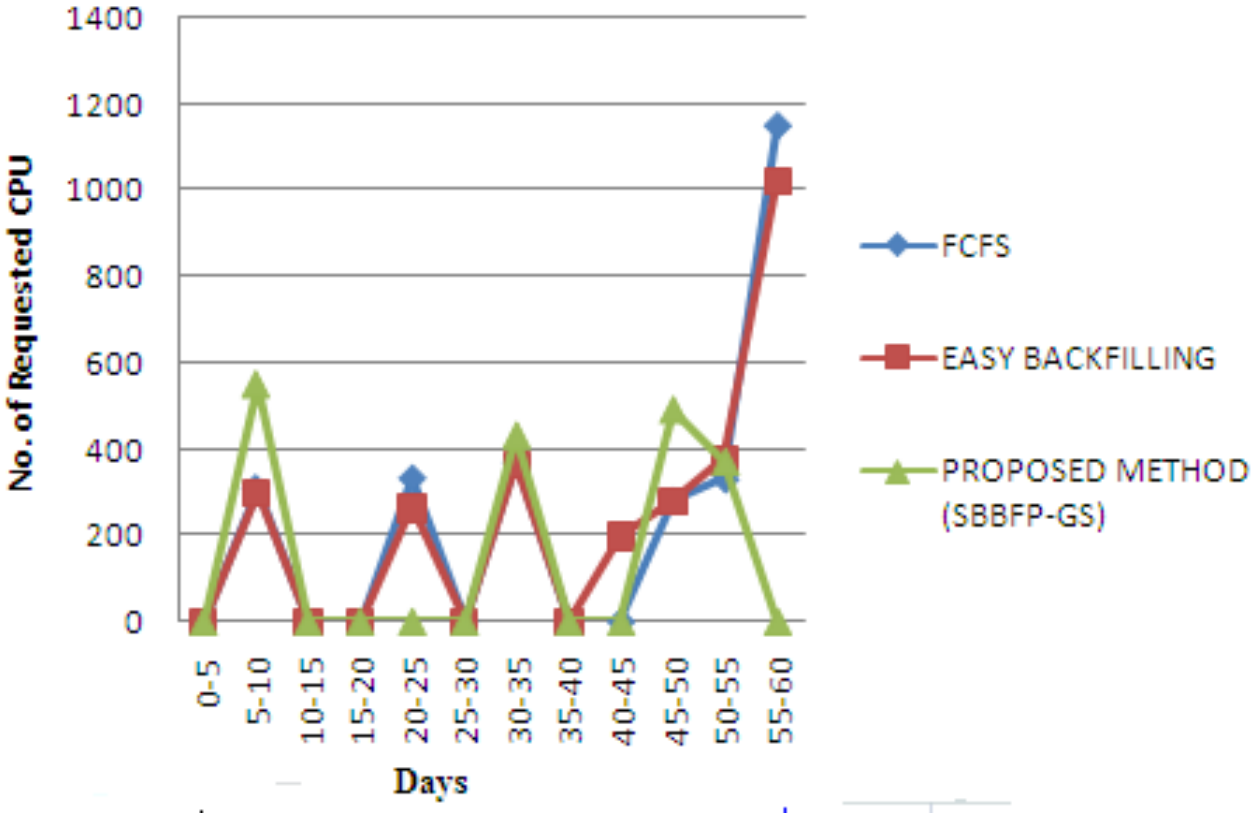

Figure 4: Number of Requested CPU with FCFS, EASY Backfilling and Schedule Based Backfilling Policy with Gap Search Strategies

As per above graph, proposed technique schedule based backfilling policy with gap search has more requested CPU as compare than FCFS and EASY Backfilling method.

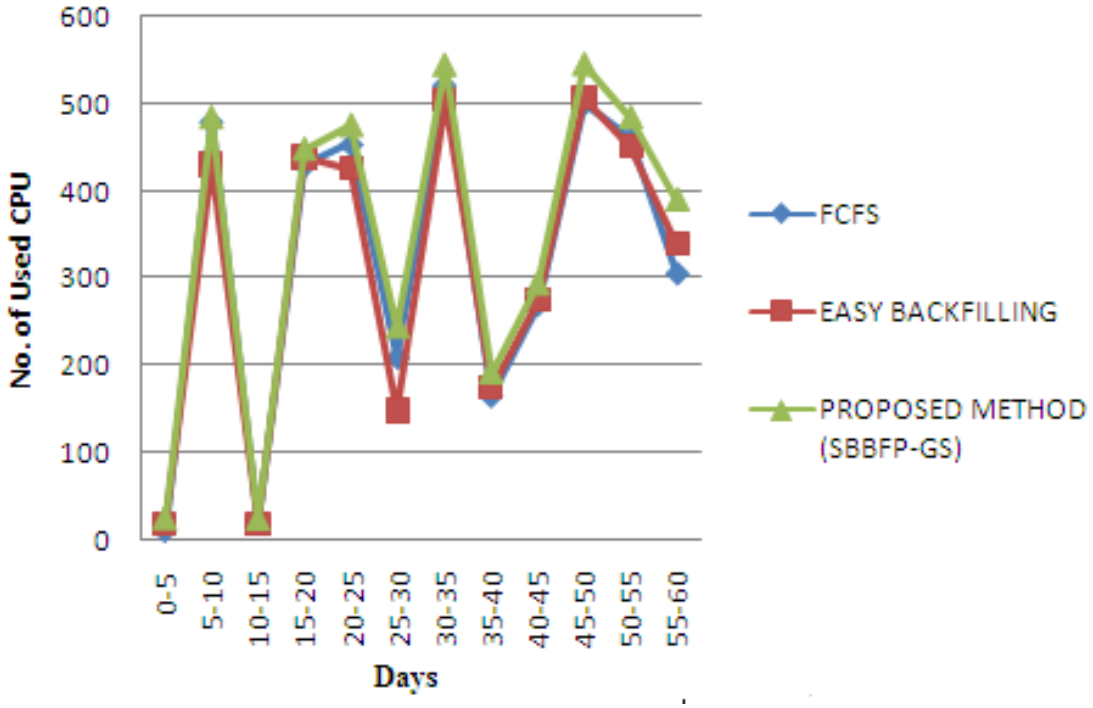

Figure 5: Number of Used CPU with FCFS, EASY Backfilling and Schedule Based Backfilling with Gap Search Strategies

As per above graph, proposed technique schedule based backfilling policy with gap search has more used CPU as compare than FCFS and EASY Backfilling method.

VII. CONCLUSION AND Future WORK

The proposed SBBFP-GS Algorithm implements load balancing for scheduling the jobs. Experiments have been done for makespan that serves as a parameter for evaluating the efficiency of the algorithm and finally average resource utilization that serves as the evaluation parameter for proper load balancing. From the results and discussion section it is observed that load balance threshold achieves a better result than the existing FCFS and EASY Backfilling algorithms. The proposed SBBFP-GS algorithm considers the load of each resource at the time of scheduling which are very important in grid environment. This can be extended in future with factors for reducing the communication overhead of the grid system.

We showed the design and implementation of a protocol (SBBFP-GS) for load balancing with scheduling in a Computational Grid. The Grid is partitioned into a number of clusters and each cluster has a coordinator to perform local load balancing decisions and also to communicate with other cluster coordinators across the Grid to provide inter-cluster load transfers, if needed. Our results confirm that the load balancing method is scalable and has low message and time complexities. Our work is ongoing and we are looking into using the proposed model for real-time load balancing where scheduling of a process to a Grid node should be performed to meet its hard or soft deadline.

REFERENCES

[1] B.Priya and Dr.T.Gnanasekaran, “Grid Architecture for Scheduling and Load Balancing – An Assessment”, IEEE Conference ICICES2014, Dec-2019.

[2] Massimiliano Caramia and Stefano Giordani, “Resource Allocation In Grid Computing: An Economic Model”, Wseas Transactions On Computer Research, Page 19-27, Issue 1, Volume 3, January 2018.

[3] P. Keerthika And N. Kasthuri, “A Hybrid Scheduling Algorithm With Load Balancing For Computational Grid”, International Journal Of Advanced Science And Technology, Vol.58, Pp.13-28, 2017.

[4] K. Sathish and A. Rama Mohan Reddy, “Maximizing Computational Profit in Grid Resource Allocation using Dynamic Algorithm”, Global Journal of Computer Science and Technology Cloud and Distributed, Volume 13, Issue 2, 2016.

[5] Leyli Mohammad Khanli, Behnaz Didevar, “ A New Hybrid Load Balancing Algorithm in Grid Computing Systems”, International Journal of Computer Science Emerging Technology, Vol-2 No 5, Page 304-309, October, 2015.

[6] Resat Umit Payli, Kayhan Erciyes and Orhan Dagdeviren, “Cluster-Based Load Balancing Algorithms For Grids”, International Journal Of Computer Networks & Communications, Vol.3, No.5, Sep, Page 253-269, 2014.

[7] Belabbas Yagoubi and Yahya Slimani, “Dynamic Load Balancing Strategy for Grid Computing”, International Journal of Computer, Electrical, Automation, Control and Information Engineering Vol:2, No:7, 2013.

[8] Ralf Diekmann, Andreas Frommer and Burkhard Monien, “Efficient schemes for nearest neighbor load balancing”, www.elsevier.com/locate/parco, 2012.

[9] U. Karthick Kumar, “A Dynamic Load Balancing Algorithm in Computational Grid Using Fair Scheduling”, International Journal of Computer Science Issues, Vol. 8, Issue 5, No 1, September 2011

[10] Pawandeep Kaur and Harshpreet Singh, “Performance Analysis of Adaptive Dynamic Load Balancing in Grid Environment using GRIDSIM”, International Journal of Computer Science and Information Technologies, Vol. 3 (3), Page 4473-4479, Apr-2011.

[11] Kai Lu, Riky Subrata and Albert Y. Zomaya, “On the performance-driven load distribution for heterogeneous computational grids", www.elsevier.com/locate/jcss, Feb-2007.

[12] Priyanka Chauhan, Ritu Bansal, “Efficient Load Balancing and Resource Scheduling for Optimizing Cost and Execution Time Using ACO-A*Algorithm”, International Journal of Recent Research Aspects ISSN: 2349-7688, Vol. 1, Issue 2, pp. 189-196, September 2010.