Performance Analysis of Adaptive Resource Clustering in Grid

Ashish Chandak
Department of Computer Science and Engineering, National Institute of Technology, Rourkela, India

Bibhudatta Sahoo
Department of Computer Science and Engineering, National Institute of Technology, Rourkela, India

Ashok Kumar Turuk
Department of Computer Science and Engineering, National Institute of Technology, Rourkela, India

ABSTRACT
A grid provides abundant resources to the grid users. Task scheduling is a fundamental issue in grid computing. The objective of task scheduling is to allocate required resources to user request. Grid application that requires fast task execution does not perform well since tasks are assigned according to node availability not according to node computing capability. In this paper, we discussed adaptive resource clustering architecture that virtually grouping same computing capability nodes based on the number and resource requirement of tasks so that task execution becomes faster. In this paper, we evaluate the performance of adaptive clustering with static and without clustering of resources and task are schedule by Max-Min, Min-Min, FCFS heuristics and simulation results shows that our architecture outperforms in makespan and success execution rate of tasks.

Keywords
Task Scheduling, Adaptive Resource Clustering, Task Clustering, Success Execution Rate.

1. INTRODUCTION
The demand for computational resources are on the rise for applications such as in i) medical science viz. drug design modeling, brain activity analysis, cellular micro physiology, ii) scientific applications viz. weather forecasting, aerospace modeling and, iii) commercial application such as stock market portfolio management. The ever growing demand for computational requirements of these applications purports the need of distributed computing which can provide huge computational infrastructure as well as highly available resources. Grid has enormous computational capability and abundant resource availability to support these type of applications. Grid computing is considered to be a wide area distributed computing [1, 2] which provides sharing, selection and aggregation of distributed resources that spans not only locations but also various organizations, machine architectures and software boundaries to provide unlimited power, collaboration and information access to everyone connected to a grid and makes them use for their computational purpose. One of the important aspects of a grid is task scheduling. Since there exists high heterogeneity of resources such as PCs, Workstations, Clusters in grid which are not only distributed geographically but also have different time zone, fabric management policies, scheduling policies, application requirements and design patterns.

A major issue is how to distribute tasks among nodes. In traditional scheduling, tasks are assigned to any of the available nodes. Scheduling in multiprocessors has been also studied in [18, 17, 16, 13]. Grid application that requires fast task execution does not perform well since tasks are assigned according to node availability not according to node computing capability. Resource clustering along with task clustering is considered to be of great significance with regard to performance issue. Resource clustering is defined as coalition of same type of resources while task clustering is defined as the coalition of several fine grained tasks.

Task clustering mechanism is employed in [13, 14] while [5] address adaptive resource provisioning with a focus primarily on resource sharing and container level resource management. [6] were one of several groups to explore dynamic resource provisioning within a data center. Bresnahan et al. [12] describe a multi-level scheduling architecture specialized for the dynamic allocation of compute cluster bandwidth.

In summary, what distinguishes our work from others is use of task clustering in combination with adaptive resource clustering. Our architecture first clusters same resource requirement tasks and based upon number of tasks it cluster identical computing capability resources. If there are n resources then our architecture divides resources into three types viz. I/O, computational, data and tasks are mapped according to resource requirement. We evaluated the performance of adaptive clustering with static and without clustering of resources in terms of makespan and success execution rate of tasks and this combination of technique allow us to achieve lower makespan, high success execution rate of tasks.

The rest of the paper is organized as follows. We presented problem description in Section 2. Proposed solution for adaptive resource clustering which contains architecture is presented in Section 3. Performance evaluation of adaptive resource clustering is presented in Section 4. Finally, some conclusions are drawn in Section 5.

2. PROBLEM STATEMENT
In traditional scheduling, grid scheduler randomly selects a site with enough available resources to allocate the needed resources. This strategy of dispatching task to any of available site has two disadvantages i) Since each node has specific computing capability task execution might be slow if task is not assigned to particular node. ii) If task are not assigned according to its resource requirement then system performance decreases and it is not full utilization of resources. To overcome this

situation, we use static clustering in which same computing capability nodes are cluster from beginning regardless number of tasks as shown in Fig 1. But this approach has also disadvantages i) since resources are cluster regardless of number of task, if more number of task come of same type then system performance decreases. To overcome both situations we use adaptive clustering concept in which resources are cluster according to task number. Our model first cluster same resource requirement task and based upon number of task it cluster identical computing capability resources as shown in Fig 2. In general, scheduling algorithms have to deal with resource assignment to tasks and resource assignment refers to the selection of resources. In this paper, we focus on the resource assignment part. Regarding resource assignment, there are immediate mode and batch mode scheduling algorithms. Immediate mode algorithms schedule jobs as soon as they arrive in the system [8]. On the other hand, batch mode algorithms allocate a batch of jobs which are in the queue of the scheduler [9]. Batch mode method is taken into account in this paper.

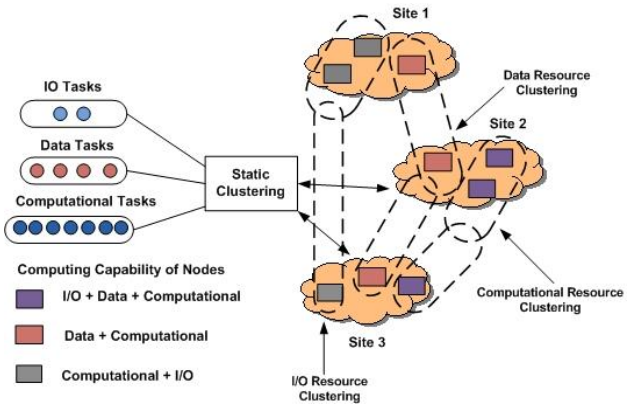

Figure 1. Static Resource Clustering

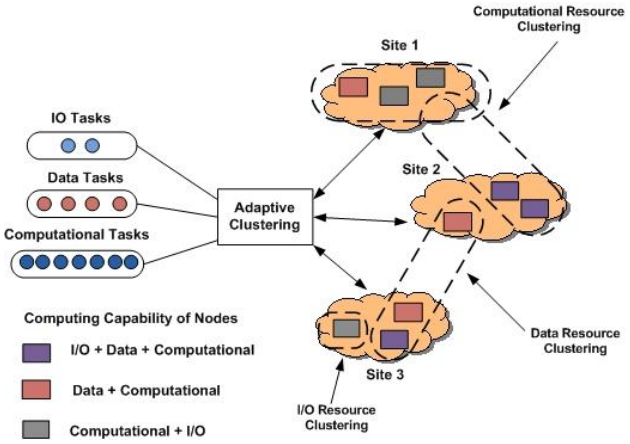

Figure 2. Adaptive Resource Clustering

3. PROPOSED SOLUTION

3.1 Architecture for Adaptive Resource Clustering

We consider following assumptions about task.

a) All tasks are independent.

b) Tasks are non preemptive: their execution on a node cannot be suspended until completion..

c) All nodes have the different computing capability.

d) Tasks are clairvoyant as their service demand are known to schedulers.

Components of adaptive resource clustering architecture:-

a) Site Information System (SIS)}:- It gathers the following information about each site S_i.

1) Identity of site S_i.

2) Status of the site S_i.

3) The total load at the site S_i.

4) Computation capability of each node of every site S_i.

b) Global Task Database (GTD):- This stores information about each task. The following attribute about a task is maintained at GTD:-

1) User Request Identity (Owner of the task).

2) Arrival time of the task.

3) Expected execution time required by the task.

4) Task resource requirement.

c) Task Clustering Agent (TCA): - It clusters same type of tasks viz. I/O, Data and/or Computational.

d) Task Distribution Manager (TDM):- It dispatches tasks to resource clusters.

e) Task Matchmaker (TMM):- Matching of resource cluster to a particular task cluster is done by TMM.

f) Adaptive Resource Management (ARM):- It cluster resources of same type {\it viz} I/O, data and/or computational based upon number of tasks. It decides number of resources required based on number of tasks. ARM will decide when to acquire resource and length of time for which resource should be required. ARM query SIS for resource information. ARM also decides about resource release policy. Resource acquisition and release policy are independent events.

The architecture for adaptive resource clustering is shown in Fig. 3 and sequence diagram for architecture is shown in Fig. 4. We represent the grid system as a M/M/s: N/FCFS queuing model where: M - represents exponential inter arrival times between tasks, M - represents exponential execution time of tasks, s - represents number of computing sites, N - represents capacity of system i.e maximum task allowed in the system (this includes executing task plus waiting task) , FCFS - represents

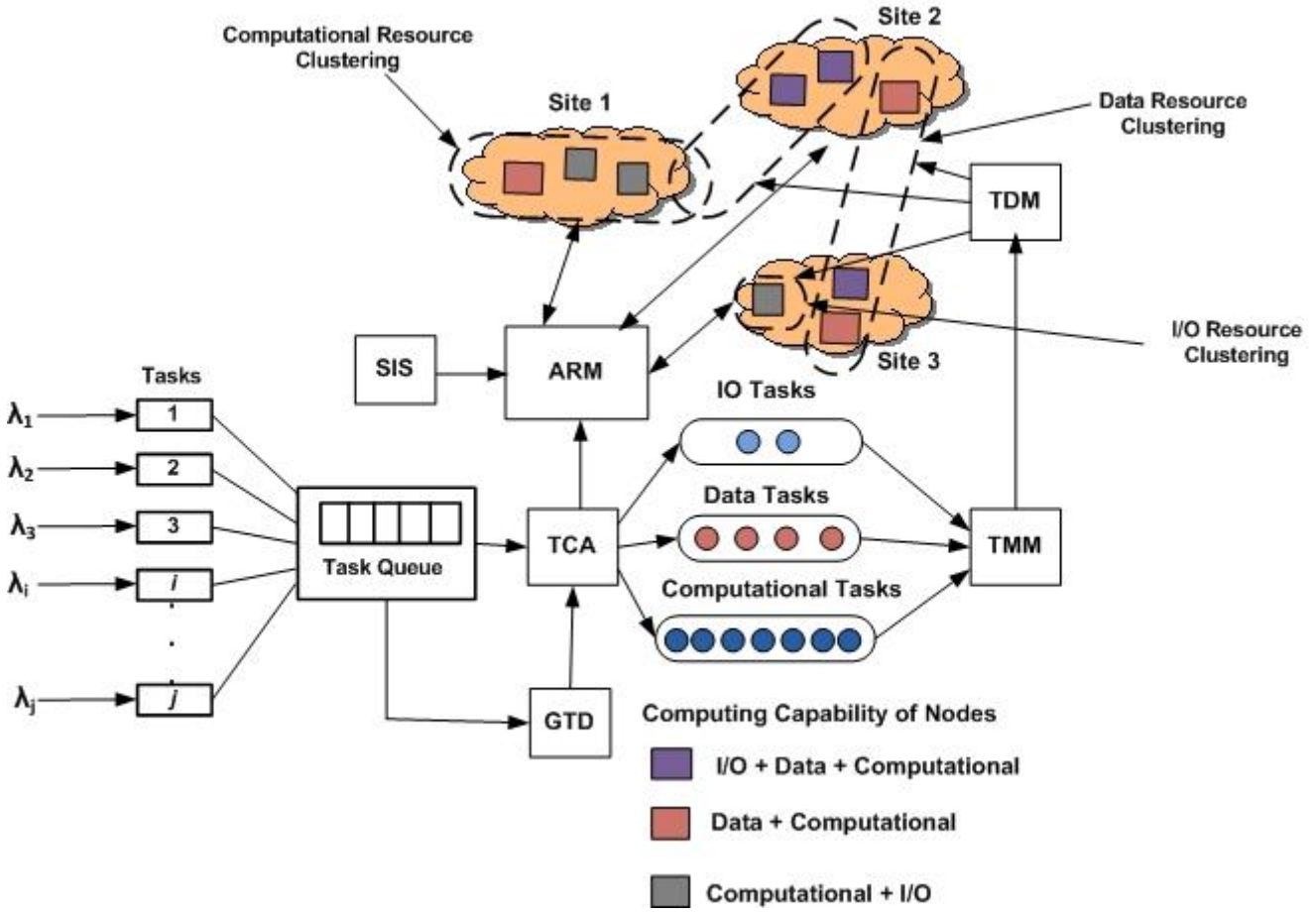

Figure 3. Architecture for Task Mapping on Adaptive Resource Cluster

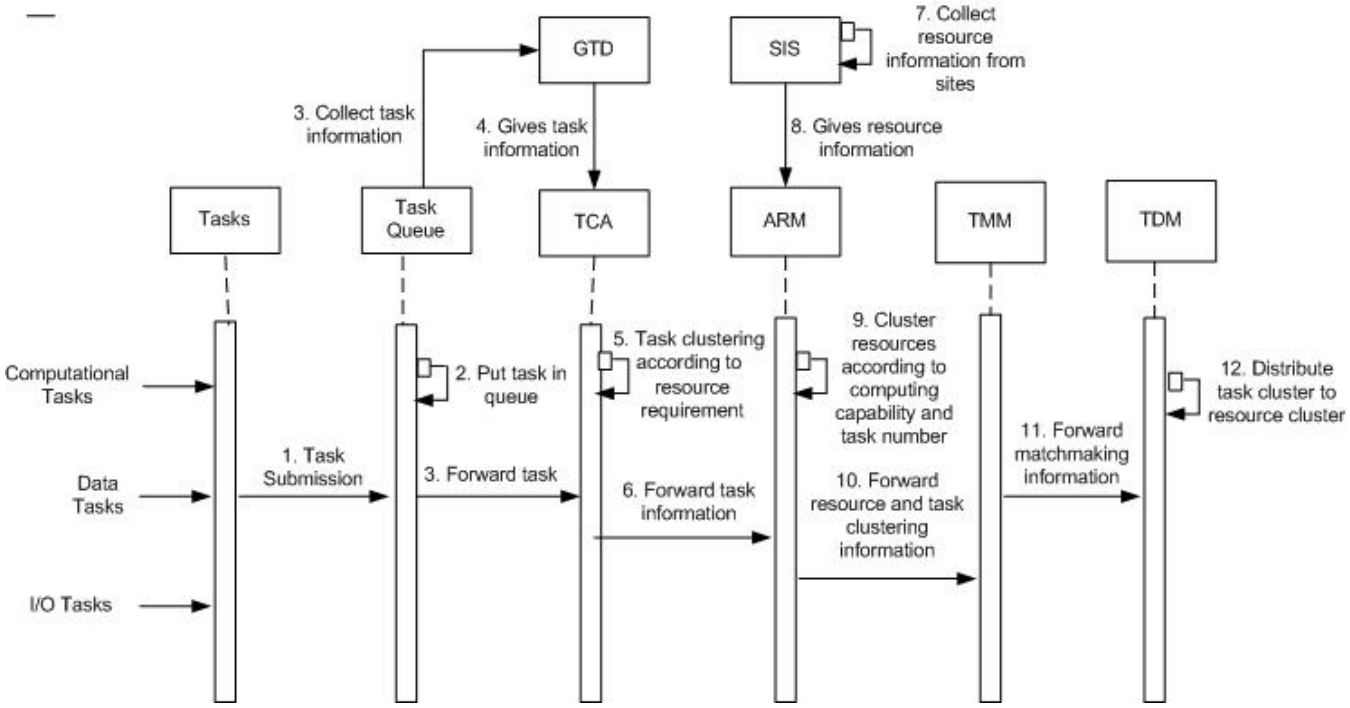

Figure 4 Sequence diagram for Adaptive Resource Clustering

First Come First Serve queue discipline.

Let λ_i be the rate of arrival of task from each grid user i at the grid scheduler. Assuming that there are j number of grid user, the total rate λ at which task arrive at the grid scheduler $\lambda_i = \sum_{i=1}^{j} \lambda_i$ Let μ_i be the rate at which a task is served at each site i. We assume that the service rate is independent and identically distributed. The combined service rate of all sites in a grid is $\mu = \sum_{i=1}^{n} \mu_i$. Our queuing model is characterized by following parameters :-

n - Number of tasks in the system.

λ - Arrival rate of tasks.

μ - Service rate of tasks.

ρ - The expected fraction of time the sites are busy i.e.

Utilization factor is given by $\lambda / s\mu$.

Then, steady state probability of having n tasks in the system as given by [3]

$P_n = (s\rho)^n P_0 / n!$ for $0 \le n \le s$

$= (s\rho)^n P_0 / s!$ for $s \le n \le N$

$= 0$ for $n > N$

Where P_0 is written as

$$P_0 = [\sum_{n=0}^{s} (s\rho)^n / n! + \sum_{n=s}^{N} (s\rho)^n / s! (s^{n-s})] ^{-1}$$

Expected task queue length i.e. expected number of tasks in the task queue given by [3]

$$L_q = \sum_{n=s}^{N} (n-s) P_n$$

$$= (s\rho)^s P_0 \rho / (s! (1- \rho)^2)$$

$$[(1- \rho^{N-s+1}) - (1- \rho) \quad (N-s+1) \rho^{N-s}]$$

We consider resource heterogeneity, which is node differs in processing capabilities. The resource heterogeneity is characterized by various computing capabilities. Nodes with identical processing capabilities are grouped into a type. Suppose there are n different types of nodes and N= {1, 2, 3, ……., n }. Each node P_i, i ε N is specified by three tuple (P_i, μ_i, θ_i) where μ_i, θ_i are processing speed and number of nodes of type P_i.

Total number of nodes is given by $\theta = \Sigma_{i=N} \theta_i$

Each node in the system is identified by a node id. The node ids of type N_1 are (P_1, P_2, P_3, …, P_n) and those of type N_2 ($P_{\theta 1+1}$, $P_{\theta 2+2}$, $P_{\theta 3+3}$ …… $P_{\theta n+n}$).The processing model is shown in Fig. 4. When tasks arrive to central queue then, they are categorized into three types I/O, Computational and Data type by task clustering agent. Adaptive resource manager does adaptive clustering of resources based on number of tasks and task distribution manager will assign task to different groups. Task system is partitioned such that

T_i= { T_i ε T| T_i =I/O}

T_j= { T_j ε T| T_j = Computational}

T_k= { T_k ε T| T_k = Data}

4. PERFORMANCE EVALUATION

4.1 Simulation Model

We developed a simulation application in matlab to carry out the experiments. Each simulation experiment ends when 1000 jobs executions are completed. Fig. 3 shows the simulation model which consist of nine nodes each having different computing capability. The arrival of tasks is modeled as Poisson random process. To evaluate performance we have considered following three types of tasks: a) I/O intensive tasks b) Data intensive tasks c) Computational intensive tasks. We evaluate performance of without clustering, static clustering and adaptive clustering of resources and tasks are schedule using simple heuristic viz. Max- Min, Min-Min and FCFS heuristics.

4.2 Performance Metrics

The purpose of performance comparison is to quantitatively evaluate the improvement that system would experience using adaptive resource clustering in comparison to the static clustering and without clustering. The parameter to be studied are as follows:

i) Makespan: - Makespan is calculated as maximum of completion time. [10].

MK=max (CT_{jobs}) Where, CT- completion time

ii) Successful execution rate: - $\sum_{0 \le i \le n} \psi_i / n$ where $\psi_i = 1$ if $T_c \ge T_d$

$\psi_i = 0$ if $T_c \le T_d$

Here, T_d and T_c denote deadline and completion time of Task J_i, respectively. [11].

4.3 Makespan Results

Table 1, 2 and 3 shows the comparison of makespan with Max Min, Min Min and FCFS heuristics. Scheduling task using Max-Min heuristic in adaptive clustering gives 28$\%$ and 18$\%$ better makespan than without clustering and static clustering respectively. Scheduling task using Min-Min heuristic in adaptive clustering gives 31% and 28% better makespan than without clustering and static clustering respectively. By using FCFS heuristic for task scheduling in adaptive clustering gives 35% and 26% better makespan than without clustering and static clustering respectively. Overall adaptive clustering gives better makespan than static and without clustering. without clustering and static clustering respectively.

Table 1 Makespan Comparison in Max-Min

Tasks	Makespan in Seconds			Improvement	
	Without Clustering	Static Clustering	Adaptive Clustering	Over without clustering	Over static clustering
100	76	95	59	22	38
200	151	172	114	25	34
300	228	178	109	52	39
400	293	236	171	42	28
500	370	294	202	45	31
600	451	366	213	53	42
700	514	612	391	24	36
800	593	569	476	20	16
900	656	508	430	34	15
1000	721	591	423	41	28
Overall Improvement in %				36	30

Table 2 Makespan Comparison in Min-Min

Tasks	Makespan in Seconds			Improvement	
	Without Clustering	Static Clustering	Adaptive Clustering	Over without clustering	Over static clustering
100	76	89	56	26	59
200	146	122	92	37	24
300	221	168	125	43	24
400	295	233	186	37	25
500	366	288	225	39	28
600	440	340	302	31	13
700	490	481	389	21	24
800	585	677	434	26	56
900	670	508	452	33	12
1000	740	710	583	22	21
Overall Improvement in %				31	28

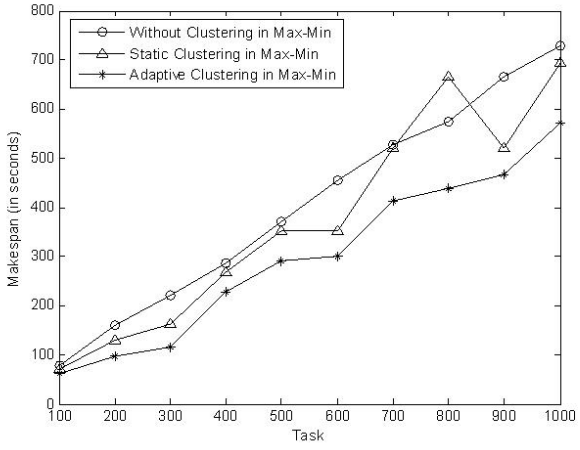

Figure 5 Makespan Comparison in Max Min

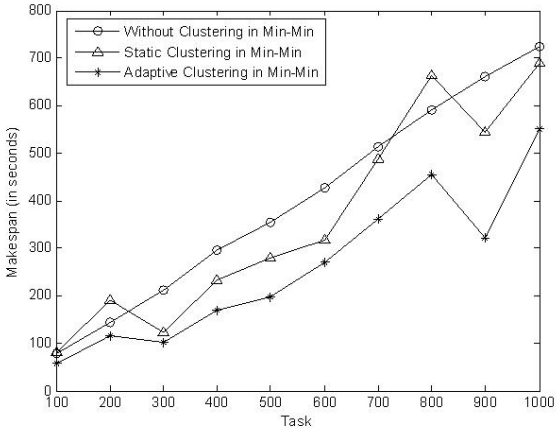

Figure 6 Makespan Comparison in Min Min

Table 3 Makespan Comparison in FCFS

Tasks	Makespan in Seconds			Improvement	
	Without Clustering	Static Clustering	Adaptive Clustering	Over without clustering	Over static clustering
100	75	72	53	29	26
200	156	187	127	19	32
300	220	153	125	43	18
400	311	241	188	40	22
500	368	536	310	16	42
600	450	349	221	51	37
700	516	406	292	43	28
800	588	460	339	42	26
900	669	514	468	30	9
1000	730	569	455	38	20
Overall Improvement in %				35	26

4.4 Success Execution Rate Results

Success execution rate are shown in Fig. 8, 9, 10. We can see from the results that adaptive clustering gives better success execution rate as compared to static and without clustering when task are schedule using Max-Min, Min-Min and FCFS heuristics..

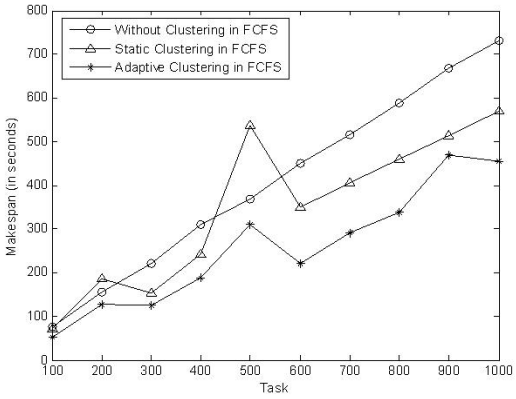

Figure 7 Makespan Comparison in FCFS

5. CONCLUSION

In this paper, we evaluated performance of adaptive resource clustering technique to enhance speed of task execution. In simulation experiments, we implement adaptive resource clustering and compare with static and without clustering of resource. Our strategy exploits effectiveness of adaptive resource clustering over static and without clustering. Table 1, 2 and 3 shows the comparison of makespan with Max-Min, Min-Min and FCFS heuristics. Task scheduling using Max-Min heuristic in adaptive resource clustering gives 28% and 18% better makespan than without clustering and static clustering of resources respectively while task scheduling using Min-Min heuristic in adaptive resource clustering gives 31% and 28% better makespan than without clustering and static clustering of resources respectively. Scheduling task by FCFS heuristic in adaptive clustering gives 35% and 26% better makespan than without clustering and static clustering respectively. From Fig. 8, 9, 10 we can conclude that success execution rate of tasks gets increased when tasks are scheduled in adaptive clustering environment. This confirms superiority of adaptive resource clustering over static and without clustering of resources.

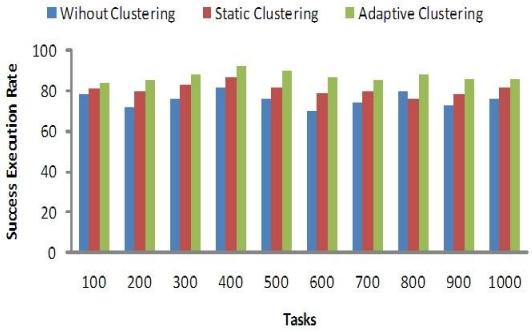

Figure 8 Success Execution Rate in Max Min

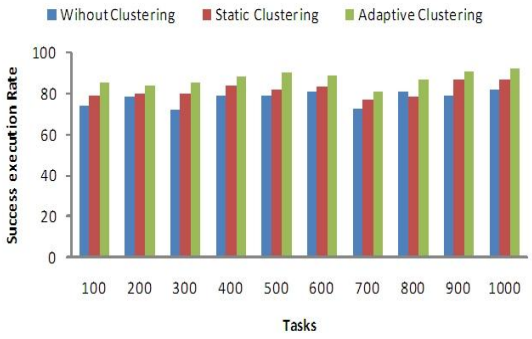

Figure 9 Success Execution Rate in Min Min

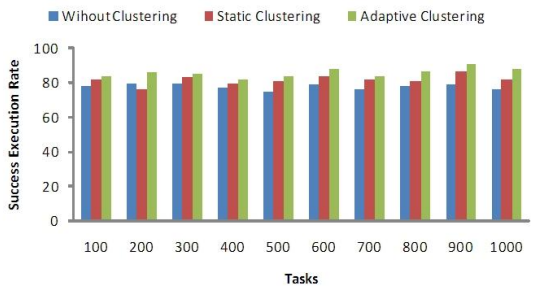

Figure 10 Success Execution Rate in FCFS

6. REFERENCES

[1] Ian Foster, Carl Kesselman, and Steven Tuecke, The anatomy of the grid: Enabling scalable virtual organizations, International Journal High Performane Computing Application 15, 200-222, 2001.

[2] Carl Kesselman Ian Foster. The Grid 2: Blueprint for a New Computing Infrastructure. ELSEVIER, Second edition.

[3] S D Sharma. Operation Research. Kedar Nath Ram Nath and Co, Fourteenth edition, 2001.

[4] Tracy D. Braun, Howard Jay Siegel, and Noah Beck, A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems, Journal of Parallel and Distributed Computing, 810-837, 2001.

[5] L. Ramakrishnan, L. Grit, A. Iamnitchi, D. Irwin, A. Yumerefendi, J. Chase. Toward a Doctrine of Containment: Grid Hosting with Adaptive Resource Control. IEEE/ACM SuperComputing.

[6] K. Appleby, S. Fakhouri, L. Fong, G. Goldszmidt, M. Kalantar, S. Krishnakumar, D. Pazel, J. Pershing, and B. Rochwerger, Oceano - SLA Based Management of a Computing Utility, In 7th IFIP/IEEE International Symposium on Integrated Network Management.

[7] Y. He, W.J. Hsu, C.E. Leiserson, Provably efficient online nonclairvoyant adaptive scheduling, IEEE Trans. Parallel Distrib. Syst. 19, 1263-1279, 2008.

[8] F. Xhafa, L. Barolli, A. Durresi, Immediate mode scheduling of independent jobs in computational grids, in: Proceedings of the 21st International Conference on Advanced Networking and Applications (AINA '07), IEEE, 970-977, 2007.

[9] F. Xhafa, L. Barolli, A. Durresi, Batch mode scheduling in grid systems, Int. J. Web Grid Serv. 3 19-37, 2007.

[10] Kobra Etminani, M. Naghibzadeh, A Min-Min Max-Min Selective Algorihtm for Grid Task Scheduling., ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia, 2007.

[11] Hesam Izakian and Ajith Abraham and Behrouz Tork Ladani, An Auction Method for Resource Allocation in Computational Grids, Future Generation Computer Systems, 26, 228 - 235, 2010.

[12] J. Bresnahan, I. Foster. An Architecture for Dynamic Allocation of Compute Cluster Bandwidth, MS Thesis, Department of Computer Science, University of Chicago, December 2006.

[13] H.D. Karatza, A simulation model of task cluster scheduling in distributed systems, in: Proceedings of the 7th IEEE Workshop on Future Trends of Distributed Computing Systems (FTDCS'99), December 20–22, 1999, Cape Town, IEEE Computer Society Press, 163–168, 1999.

[14] Kyriaki Gkoutioudi, Helen D. Karatza, Task cluster scheduling in a grid system, Simulation Modelling Practice and Theory, 18, 1242–1252, 2010.

[15] A. Gerasoulis, T. Yang, On the granularity and clustering of directed acyclic task graphs, IEEE Transactions on Parallel and Distributed Systems 4 (6), 686–701, 1993.

[16] S.P. Dandamudi, Performance implications of task routing and task scheduling strategies for multiprocessor systems, in: Proceedings of the IEEE Euromicro Conference on Massively Parallel Computing Systems, Ischia, Italy, 348–353, 1994.

[17] H.D. Karatza, A Comparative analysis of scheduling policies in a distributed system using simulation, International Journal of Simulation Systems, Science and Technology 1, 12–20, 2000.

[18] L.W. Dowdy, E. Rosti, G. Serazzi, E. Smirni, Scheduling Issues in high-performance computing, Performance Evaluation Review 26, 60–69, 1999.