A Combined Replication-Adaptive Scheduling Model for Desktop Grid Environment

Shailaja Pandey
National Institute Of Science and Technology, Berhampur,

Ashu A
Assistant Professor
National Institute Of Science and Technology, Berhampur,

K.Hemant K Reddy
Assistant Professor
National Institute of Science and Technology, Berhampur,

ABSTRACT
Now a day's replication is an effective approach to improve the efficacy of distributed system, where large amount of data (terabytes or peta-bytes) is handled. An efficient replica technique is more effective than a shared distributed system (network attached storage, object based storage and storage area network) and common access point. In a distributed system, data access time depends on unreliable network bandwidth especially in desktop grid. The data transfer is a major bottleneck in data intensive distributed grid environment due to high latency and low and unreliable bandwidth. In such an environment, an effective scheduling and effective replica technique can reduce the amount of data transfer across the internet by dispatching a job to a node where the required data are present for its operation. As the computing scale and the amount of data involved in grid applications is increasing exponentially, which causes grid resources to wait for long time period for data transfer when the involved data is saved in the remote nodes. This degrades the overall system performance. Using the file sharing mechanism in a distributed file system with a replica technique or by using a nature inspired meta-heuristic optimization technique system performance can be improved. In case of file sharing mechanism with replication techniques data can be processed in parallel. In this paper we proposed a novel combined model for data replication and job scheduling for the desktop grid environment. A reliability based replica management technique is proposed for the distributed grid environment in such way that overall data transfer is minimized. An adaptive technique is proposed for job scheduling which considers the parameters like node efficiency value, past execution history from execution log and node locality value (is a weighted parameter, depending upon the availability of replica).

General Terms
Distributed System, Data Placement, Big Data, Grid.

Keywords
Heuristics, data replication, adaptive, desktop grid, distributed systems, reliability.

1. INTRODUCTION
In data grids [1, 2], most of distributed applications normally require access to a large amount of data (terabytes or peta-bytes). Managing the huge amount of data at central point is ineffective due to extensive access latency and load on the grid server. Hence, such huge dataset must be separated and stored in different physical locations. In a grid communication environment, the efficiency of accessing a huge amount of distributed data depends on the availability of bandwidth of the network. Slow data access inhibits the performance of data-intensive applications running on distributed systems, even more in desktop grids. Figure 1 presents the proposed simple hierarchical distributed model. The entire model is framed with the help of different programming labs, where all nodes of each lab are treated as a cluster of nodes with a cluster head. There are two kinds of communication between sites in a cluster grid Intra-communication and Intercommunication. Intra-communication is the communication between nodes belonging to the same cluster group. In Intercommunication the communication is between nodes across clusters. Within a cluster, network bandwidth between nodes will be better than across clusters. Thus, to reduce network latency and to avoid low bandwidth bottleneck in a cluster grid, it is imperative to reduce the number of intercommunications. To solve this problem, we consider two important aspects of intercommunication: effective job scheduling and replication mechanism. Consider a case where many authorized users to submit jobs to solve a data-intensive problem. For faster execution, scheduling of jobs to suitable nodes is necessary because data transfer between different nodes within the system is time consuming and other some factors. The factors that improve the efficiency of the distributed system need to be considered during the scheduling, these are available heap memory and available CPU load, location of data, network bandwidth and node reliability is the prime importance. If a job is submitted to a grid node and the required data is residing on the same node (local node), the job can process data without any delay for getting data from a remote node. Data replication is another novel technique for data-intensive systems by replicating data in geographically distributed nodes. When the user submitted jobs need to access huge amounts of data from remote nodes, the dynamic replica - optimizer in the site tries to store replicas on local storage for any possible future use. If data reside on the local node, the regularity of remote data access decreases. This diminishes the job execution time. Hence, in this way, Inter-cluster communications can also be avoided.

2. RELATED WORK
In data intensive distributed system, scheduling performance depends on effective computation and efficient data management technique. The replication of data sets is a technique which has been prevalent since very long time and is now adopted in data-intensive grid computing to make the grid job execution more effective. It is not a new technique, but taking decision on replica position is crucial. Several replication policies and strategies which make decisions on positioning of replicas have already been proposed. Most of the proposed strategies relies on certain assumptions along with different guarantees to clients. Solutions of these replication policies and strategies can be classified into on-line and batch modes [1].

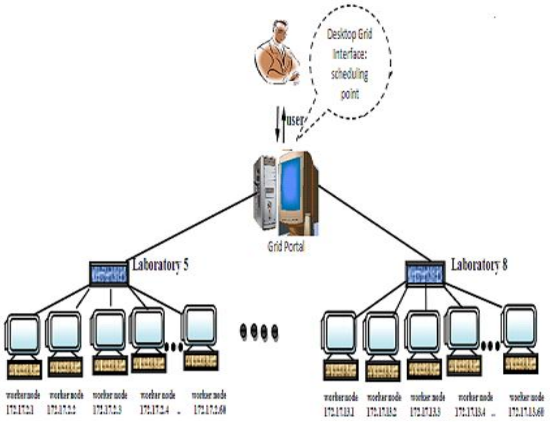

Figure 1 Hierarchical Cluster Grid Structure

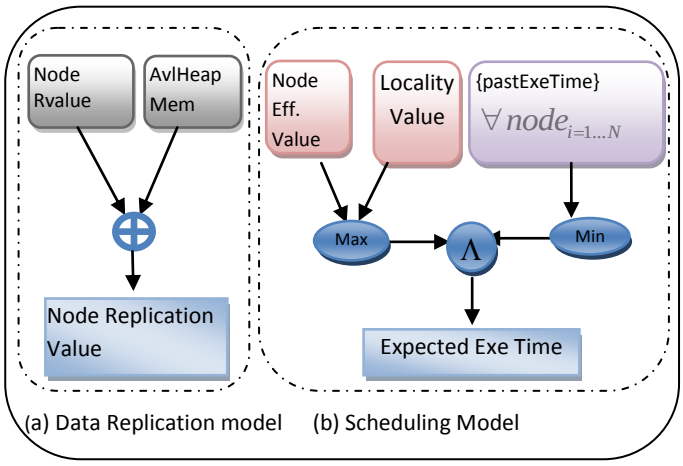

Figure 2 Logical Designed cluster Structure for Replication

In the online mode, the jobs are assumed to arrive one-by-one, generally a predetermined distribution is considered and it is the responsibility of the grid schedulers to dispatch these jobs as soon as it receives them. In the batch mode (also known as batch-of-jobs or bulk), the jobs are assumed to get selected in a bulk and the grid scheduler assigns this set of jobs to the nodes at the same time. Results in [1] show that the online mode is a fair representation of small grids and massive systems always process their jobs in the batch module. Most of the approaches usually use either on-line mode or batch mode, but both (on-line & batch) modes are used in very few approaches. Achieving optimality in performance scheduling is a major bottleneck in distributed systems. Effective data communication and data management are major issues involved in data intensive grid systems.

Workload management is a major task of grid job scheduling mostly in case of batch mode processing. A resource broker is created in European Data Grid (EDG) project for the workload management. The EDG project is an extended version of the Condor project [2]. In the latest version of gLite from EGEE project, shared sandbox approach was proposed to solve the batch mode grid job scheduling problem [3]. In deadline based scheduling strategy, data transfer time is taken into consideration in case of data intensive applications [4]. These proposed strategies consider either of the priority or policy control mechanisms instead of considering the whole co-allocation and co-scheduling issues present in data intensive grid job scheduling.

Six replica strategies: No replication strategy, Best Client strategy, Cascading Replication strategy, Plain Caching strategy, Cascading plus Cascading Replication and Fast Spread were proposed by Ranganathan and Foster [5,6] . These strategies are evaluated with three different data patterns strategies. They are (i) Random access (no locality in the patterns of access), (ii) Recently accessed file are most likely to be accessed again (temporal locality) and (iii) files recently accessed by a site are likely to be accessed by a nearby site (geographical and temporal locality). The simulation result of these strategies indicates that each data access pattern needs a different replica strategy. It was observed that of all, Cascading and Fast Spread strategies performed the best in the simulations as compared to traditional strategies. Ranganathan and Foster [7] proposed Job scheduling algorithms such as JobLeastLoaded, JobLocally, JobDataPresent and JobRandom to assign jobs in grid environment. These job scheduling algorithms are combined with three different replication strategies: DataLeastLoad, DataRandom and DataDoNothing to replicate data across nodes in data-intensive grid. In [9], a cost effective model is proposed to examine whether it is wrathful to create replicas or not. The efficacy of this model is in reducing average job execution time with replication than the normal case without replication. Close to files (CF), a job scheduling algorithm was proposed, which looks for the processors with least load near a node where data is present [10]. In this algorithm, it is assumed that the a single input data file is the requirement of the job. The simulation results show that the efficacy with respect to performance of Close-to-Files (CF) job scheduling algorithm is better than other job placement algorithm which places jobs on the nodes which has the largest number of idle processors.

Data intensive jobs are automatically queued, scheduled, monitored and managed among the nodes of the grid in Stork project [4]. In this project data intensive jobs give equal importance as computational intensive jobs. The motivation for this project is to efficiently utilize the computational resources with input data required for execution. The classes [4] method of Stork project is used to indicate a job as well as data requirements. This method is combined with a task scheduler method of the Condor project like DAGMan (Directed Acyclic Graph Manager) [16] that deals with the Integrated Replication and Scheduling Strategy (IRS) which integrates the scheduling and replication strategies. It separates jobs scheduling from data scheduling. This approach of combining Condor and Stork project covers the objectives of the computationally intensive and data-intensive jobs [13], but does not handle the batch mode processing. The Class Ad framework was used to express the relationships among different stakeholders in I/O communities where computational jobs and data files are linked together by binding Compute Nodes (CNs) and Storage Nodes (SNs) [4]. This framework does not consider policy issues in its optimization procedure, so it cannot deal with batch mode processing of grid jobs. The execution of this framework is similar to Stork project, where CPUs and data resources in the grid are linked to access a specific datasets and batch mode processing of grid job scheduling is not covered.

One of the complete scheduling approaches for simultaneous job and data file scheduling with the help of replication is Data Intensive and Network Aware (DIANA) [11, 12] scheduling is based on real GILDA [13] and CMS [14]. In this scheduling technique, jobs are classified based on their execution. In case of applications which are data intensive, jobs are moved to the best available CN with minimum download time of the required data files. In case of computationally intensive jobs, data files are replicated to the best available SN with a minimum upload time of the given dependent jobs. The decision is made in both cases based on: storage capacity of SN, the processing speed along with total CPUs available in the CN and network links connecting SNs and CNs. An alternative mechanism to DIANA is Best Map [15] which uses two mechanisms to iteratively minimize scheduling of jobs, and minimize delivery time of all data files through replication. Among BestMap and DIANA, the BestMap technique does not differentiate between jobs for scheduling or replication, rather always tries to to determine the best CN or SN to schedule/replicate a job or its dependent data files at each stage when compared to DIANA. The researchers in [17] applied machine learning approaches into their decision making process to detect data types (physical, biology, chemistry, etc.) and group them for better replications. An advanced metric is used which takes into consideration the number of times a data file is requested as well as its size, to identify hot data files. A notation of positive or negative distance between SNs to distribute data files was also invented in this approach. This distance calculated can be used to replicate data files and to achieve maximum distance among data files of different types and minimum distance of data files of the similar type. In the framework proposed by researchers in [20] a heterogeneous data file is assumed with only five distinct job types. In this framework the jobs are scheduled to reduce the amount of data transferred.

A variety of meta-heuristics approaches have been suggested to improve the jobs scheduling in distributed environments such as grids. Dynamic and Static heuristics [18] are the two different types of categories. These categories are differentiated based on number of tasks considered at every scheduling step or based on the objective tasks set. The grid job scheduling can be characterized as an NP problem [19]. It has been proved that nature inspired meta-heuristics approaches such as Genetic Algorithms, Particle Swarm Optimization and Firefly Algorithms provide a better solution in case of NP problems [19]. The authors in [19] applied GA for the scheduling jobs in grid environment. It was also proved that GAs as one of the best options for the evolution of fuzzy knowledge, as it is the case of Pittsburgh and Michigan approaches. After making a close look at the aforementioned techniques, we have realized that most of these systems are usually designed either by minimizing the timespan of execution all jobs or by reducing the time to transfer the data files in a system and very few techniques consider both the approaches. Mostly, this algorithm does not have the flexibility to be extended to other systems and in most of the situations they lack the facility to generalize. We can also observe that the algorithms which are designed for the on-line mode cannot be extended to the batch mode and vice versa. Therefore, in our work, we tried to model a generic approach with the help of a heuristic technique (adaptive) with any real system which includes a combination of both data-intensive and computational intensive job oriented systems. We have taken into consideration the general case of job to data file dependency.

3. SYSTEM MODEL

As no single replication strategy is suitable for all types of grid model and all types of applications. Number of replication strategies and grid logical structures are discussed for different data intensive applications in section 2. In this paper we propose a combined grid model, where grid nodes are classified into clusters. These clustered nodes are physically and logically grouped into clusters. An adaptive heuristic algorithm is proposed for job scheduling and an efficient data replication technique is proposed for data replication. This adaptive scheduling algorithm is employed for scheduling the jobs to resources on node efficiency value, locality value along with a past execution history of the node. As replication decision is very crucial for performance of the data grid, we consider these parameters in proposed combined model. Figure 2 presents the logical model of desktop data intensive grid model, where nodes are grouped into clusters on various parameters.

4. COMBINED MODEL

The proposed combined model uses two major parameters like, node reliability value and localization value. Replica management is an important technique in data intensive grids and even more in case of desktop grids. In our proposed model; reliability value is calculated on the basis of availability of node in comparison to the nodes that available in the cluster group and localization value depending on the point of job and data submission and scheduling point where the job is assigned for execution.

A combined model is proposed for effective replica management and a heuristic technique for job scheduling. Effective replica management is to minimize the communication overhead during data access from local and remote nodes of different clusters. Adaptive scheduling framework [22] is proposed to minimize the overall execution time user submitted jobs. In case of replica management, we proposed a replication policy by considering the parameter that puts an impact on data intensive job scheduling. The parameters used to frame the replication policy are NodeRvalue: node reliability value and TotHeapvalue(Nodei): total heap memory of the node is considered for deciding the place of replica. In scheduling policy, current node status, previous execution history of the node and Locvalue: locality value is considered.

The proposed combined model considers the three major factors like node reliability value, resources current status, previous execution time and node locality value.

$$\text{CombinedModel=NodeReliablityValue+CurrentStatus+}$$

$$\text{PrevExeTime+LocalityValue}$$

$$CModel = NodeR_{value} + NodeEf_{value} + \text{Pr}evExeTime + Loc_{value} - - - (1)$$

Combined model is framed for replication policy and scheduling policy

$$CModel = \{\text{RepPolicy+SchPolicy}\}$$

$$\text{RepPolicy} = NodeR_{value} + TotHeap_{mem}(Node_i)$$

$$\text{RepVal}_{node_i} = \max\left\{NodeR_{value} + TotHeap_{mem}(Node_i)\right\}$$
$$\forall\, node_{i=1\ldots N} ---(2)$$

$$SchPolicy = NodeEf_{value} + \Pr evExeTime + Loc_{value}$$

$$ExpExeTime = \left\{\max\left(NodeEf_{value} + Loc_{value}\right) \Lambda \min\left(\Pr evExeTime\right)\right\}$$
$$\forall node_{i=1..N} --(3)$$

Adaptive scheduling is based on the expected execution time and is calculated in equation (3).

Initial all nodes $NodeR_{value}$ is set to 1 (one), on execution of jobs on distributed system the node reliability value will change depending on the performance of the node and availability of node with respect to another node in the cluster. $NodeR_{value}$ will increase if the availability of the node is more than the other node in the cluster otherwise R_{value} will decrease as represent in equation (4).

$$NodeR_{value} = NodeR_{value} \pm \left[\frac{1}{\#\, AvlNodes(Cluster_{i=\{1\ldots no.of\ cluster\}}}\right] --(4)$$

As reliability of any node lies in between zero to one, min and max function are used to keep the $NodeR_{value}$ within the range as shown in equations (5) and (6).

$$NodeR_{value} = \min\left(1, NodeR_{value} + \left[\frac{1}{\#\, AvlNodes(Cluster_{i=\{1\ldots no.of\ cluster\}}}\right]\right) - (5)$$

Node reliability value $NodeR_{value}$ of a node is incremented based on the group in which it belongs and availability of the node with respect to the group members, as presented in Equation 5, similarly for node reliability value is decremented based on Equation 6.

$$NodeR_{value} = \max\left(0, NodeR_{value} - \left[\frac{1}{\#\, AvlNodes(Cluster_{i=\{1\ldots no.of\ cluster\}}}\right]\right) - (6)$$

Current computational and memory status of a node $NodeEF_{value}$ is calculated with set of parameters like $AvlHeap_{value}(Node_i)$, $AvlHeap_{value}(Node_i)$ and $No.of\ PJobs(Node_i)$ is formulated in equation (7).

$$NodeEf_{value} = AvlCPU_{load}(Node_i) * c1 +$$
$$AvlHeap_{mem}(Node_i) * c2 + \#\, pJobs(Node_i) --(7)$$

c1 and c2 are two constant weights associated with available CPU load and available heap memory with different weights in equation (7) for different types of jobs.

$$c1 = \left\{\begin{matrix}0.7,\ for\ Computational Jobs\\ 0.3,\ for\ Data\ Intensive\ Jobs\end{matrix}\right\}$$
$$c2 = \left\{\begin{matrix}0.3,\ for\ Computational Jobs\\ 0.7,\ for\ Data\ Intensive\ Jobs\end{matrix}\right\}$$

To minimize the communication overhead for data access from a node, locality value is considered as a major parameter in scheduling. It is a weighted parameter; weights are assigned based on the presence of data replica for the specified node and weights are presented in equation (8).

$$Loc_{value} = \left\{\begin{matrix}0.7, if\ \ job\ scheduled\ on\ same\ machine\\ 0.3, if\ \ job\ scheduled\ on\ different\ machine\end{matrix}\right\} --(8)$$

prevExeTime is calculated depending the job size and amount of data access involved from the past execution history.

4.1 Adaptive Algorithm

In this section a heuristic approach (adaptive algorithm) has been presented, in addition to dynamic information, this approach keeps track of previous job execution history to predict the completion time of future jobs in the aforementioned grid environment adaptively. The algorithm takes into account the processing capacity of the nodes, communication cost during the load balancing operation, heap memory requirement, pending jobs and locality value. The category of the problem we address here is a computational and data-intensive. The jobs are totally independent with no communication between them. Figure 3 describes the flow control of the proposed framework.

4.2 Problem Formulation

In order to capture important characteristics of job scheduling in desktop grid systems, we considered the use of test-bed model. To formulate the problem under real test-bed model, an estimation of the required computational load of each job, the computing capacity of each resource, job migration cost due to unreliable network bandwidth, locality value and an estimation of the prior load of each one of the resources are considered to model the Expected Execution Time (EET).

In this paper, we consider the classical CompletionTime for scheduling problem. If 'M' machines are available for scheduling of 'N' submitted user jobs, where job *Jj* takes $Exe_{i,j}$ units of time if scheduled on machine Mi. The following sections describe the formulation of job scheduling for N jobs on M machines using an adaptive approach.

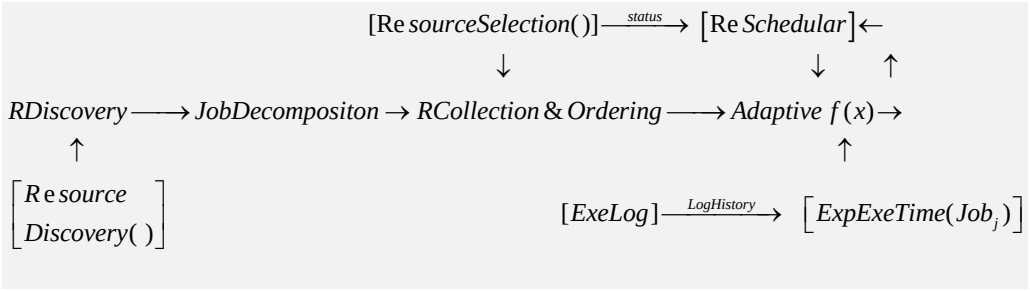

Figure 3 Framework Control Flow

Algorithm: Adaptive Scheduling

---

1. *Begin:*
2. *Grid g;*
3. *GridNodes=g.grid.totalLocalnodes()+g.grid.totalRemotenodes( );*
4. ***Resource collection:***
   for(i=0; i < GridNodes; i=i+1){NodeId[i]=g.getNodeId();
   AvlCpuLoad[i]=g.getAvailableCPUload();
   AvlHeapMemory[i]=g.getAvailableHeapMemory();
   pendJobs[i]=g.getPendingJobs();
   g.nextnode();
   *EfValue[i]= AvlCpuLoad[i]*c1+ AvlHeapMemory[i]*c2+ pendJobs[i];}*
5. ***Reliability value Calculation:***
   for(i=0; i < GridNodes; i=i+1) {
   NodeRvalue[i]= NodeRvalue[i]+(1/ GridNodes);
   RepValue[i]= NodeRvalue[i]+ g.getTotalHeapMemory();
   g.nextnode(); }
6. ***Job Decomposition:***
   TotTasks=Decompsiton(job);
   //-----discussed in reference [21]
7. ***LocVal calculation:***
   for(i=0; i < GridNodes; i=i+1){
   LocVal[i]=isReplicaPresent(nodeId);} //------is a function with return value of 0.7 or 0.3 according nodeId
8. ***Expected Execution Time calculation:***
   for(i=0; i < GridNodes; i=i+1)
   EET[i]= max(EfValue[i]+LocValue[i])+min(prevExeTimeFun(job,nodeId));
9. ***Resource Ordering and Allocation:***
   NodeId[i]=doOrdering(NodeId,EET);
   for(i=0; i <TotTask; i=i+1)
   g.execute(NodeId[i],task[i]);
10. ***End***

4.3 Fitness Function

An adaptive approach is used for calculating the Expected Execution Time (EET) for submitting jobs. The proposed model can be used to model the scheduling of jobs to resources with a multi-objective general formulation. The basic objective is that of minimizing the completion time that is, the time when the last job finishes its execution. To minimize the completion of tasks that submitted a locality value is used to prioritize the node in which replica of the requested data is present.

The following equations are used to formulate the EET (Table-1) for test-bed, which is applied in formulating fitness value.

$$f(x) = \left\{ \max\left(NodeEf_{value} + Loc_{value} \right) \Lambda \min\left(\Pr evExeTime \right) \right\} \forall node_{i=1..N} \quad ---(9)$$

Equation (9) works for grid simulation, but in case of test-bed desktop grid difficult to predict the execution time of a job. In this paper, an adaptive predictive approach is used to predict the expected execution time from execution log. To achieve this, training test benchmark jobs are executed in desktop and results stored in a log.

This log information updated in a table (Table-1) using offline and table lookup approach is used to predict the expected execution time of online submitted jobs. Multi valued regression function is applied to predict the expected execution time for a job with respect to assigned available resource computational power. Table 2 gives the information of the previous execution grid details.

Table 1: GridExeTab: that keeps previous grid execution details

Job Id	No. of Tasks	Grid Size (nodes)	Tot Avg CPU Load	TotAvg Heap Memory	Completion Time	Comm Cost
J1	65	56	40.654	7.87653488E7	6544	323
J2	45	40	50.553	5.0143124E8	4243	321

Table 2: GridExeTab: keeps grid execution Log details

Task Id	Task Type	Task Size	Avg CPU Load	Avg Heap Mem	Exe Time	Comm Cost
T1		0.654	8.7645634E7	547	432	
T2		0.553	5.0143124E8	243	321	
Tnew	Xxx	Xxxx	xxxxx	?	?	

GridGain 4.0 middleware provides necessary information about the grid system namely, grid resource information and run time task execution details. The following grid run time features are extracted using GridGain, and can be used to schedule tasks. Such information from GridGain is used for performance tuning.

The basic objective is to minimize the completion time, i.e. the time when the last job finishes its execution. ExecutionTime of job_j in a Machine Mi is formulated in equation (9).

5. CONCLUSION

In this paper, we proposed a combined model which is based on scheduling policy and data replica policy. The nature of the jobs we address here is a computational and data-intensive

and these jobs are independent of communication overhead among them. The data replication policy is proposed by considering the parameter such as the node reliability value and total heap memory available for the node in the distributed grid environment. In addition to this data replication policy, an adaptive scheduling algorithm is proposed for efficient scheduling of job in the distributed grid environment. This adaptive scheduling algorithm keeps track of previous job execution history, locality value and the current status of the node. The proposed adaptive scheduling algorithm is used to predict the completion time of the new jobs that arrive in the future by considering above parameters.

6. FUTURE WORK

The combined model is proposed to implement in a desktop grid environment with GridGain4.0 middle ware. Initially we proposed the scheduling framework with adaptive scheduling technique, but same model can be framed within meta-heuristic approaches like nature inspired GA, PSO and Firefly algorithm. The performance of these nature inspired algorithm can be analyzed with respect to different grid and job size.

7. REFERENCES

[1] M. Tang, B.-S. Lee, X. Tang, C.-K. Yeo, The impact of data replication on job scheduling performance in the data grid, Future Generation Computer Systems 22 (2006) 254–268

[2] J. Frey, T. Tannenbaum, M. Livny, I. Foster, S. Tuecke, Condor-G: A computation Management agent for multi-institutional grids, Cluster Computing 5 (2002) 237–246.

[3] P. Andreetto, S. Borgia, A. Dorigo, A. Gianelle, M. Mordacchini, M. Sgaravatto, L. Zangr, S. Andreozzi, V. Ciaschini, C. Di Giusto, F. Giacomini, V. Medici, E. Ronchieri, V. Venturi, Practical approaches to grid workload & resource management in the EGEE project, in: Proceedings of the Conference on Computing in High Energy and Nuclear Physics, CHEP'04, 2004, pp. 899–902.

[4] H. Jin, X. Shi, W. Qiang, D. Zou, An adaptive meta-scheduler for data-intensive
applications, International Journal of Grid and Utility Computing 1 (2005) 32–37.

[5] I. Foster, K. Ranganathan, Design and evaluation of dynamic replication strategies for high performance data grids, in: Proceedings of International Conference on Computing in High Energy and Nuclear Physics, Beijing,China, September 2001.

[6] I. Foster, K. Ranganathan, Identifying dynamic replication strategies for high performance data grids, in: Proceedings of 3rd IEEE/ACM International Workshop on Grid Computing, in: Lecture Notes on Computer Science, vol. 2242, Denver, USA, 2002, pp. 75–86.

[7] I. Foster, K. Ranganathan, Decoupling computation and data scheduling in distributed data-intensive applications, in: Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing, HPDC-11, IEEE, CS Press, Edinburgh, UK, 2002, pp. 352–358.

[8] J. Basney, M. Livny, P. Mazzanti, Utilizing widely distributed computational resources efficiently with execution domains, Computer Physics Communications 140 (2001) 246–252.

[9] E. Deelman, H. Lamehamedi, B. Szymanski, S. Zujun, Data replication strategies in grid environments, in: Proceedings of 5th International Conference on Algorithms and Architecture for Parallel Processing, ICA3PP'2002, IEEE Computer Science Press, Bejing, China, 2002, pp. 378–383.

[10] H.H. Mohamed, D.H.J. Epema, An evaluation of the close-to-files processor and data co-allocation policy in multiclusters, in: 2004 IEEE International Conference on Cluster Computing, IEEE Society Press, San Diego, California, USA, 2004, pp. 287–298

[11] A. Anjum, R. McClatchey, A. Ali, I. Willers, Bulk scheduling with the DIANA scheduler, IEEE Transactions on Nuclear Science 53 (2006) 3818–3829.

[12] R. McClatchey, A. Anjum, H. Stockinger, A. Ali, I. Willers, M. Thomas, Data intensive and network aware (DIANA) grid scheduling, Journal of Grid Computing 5 (2007) 43–64.

[13] GILDA, Visited 2011. https://gilda.ct.infn.it/.

[14] CERN, Visited 2011, Compact Muon Solenoid (CMS). http://public.web.cern.ch/public/en/lhc/CMS-en.html.

[15] J. Taheri, Y.C. Lee, A.Y. Zomaya, Simultaneous job and data allocation in grid environments, The University of Sydney, Sydney, Australia, TR 6712011.

[16] E. Deelman, H. Lamehamedi, B. Szymanski, S. Zujun, Data replication strategies in grid environments, in: Proceedings of 5th International Conference on Algorithms and Architecture for Parallel Processing, ICA3PP'2002, IEEE Computer Science Press, Bejing, China, 2002, pp. 378–383.

[17] N.N. Dang, S.B. Lim, Combination of replication and scheduling in data grids, International Journal of Computer Science and Network Security (IJCSNS) 7 (2007) 304–308.

[18] L. Tseng, Y. Chin, and S. Wang, "The anatomy study of high performance task scheduling algorithm for grid computing system," Computer Standards and Interfaces, vol. 31, no. 4, pp. 713 – 722, 2009.

[19] O. Cordon, F. Herrera, and P. Villar, "Generating the knowledge base of a fuzzy rule-based system by the genetic learning of the data base," Fuzzy Systems, IEEE Transactions on, vol. 9, no. 4, pp. 667–674, Aug 2001.

[20] S. Abdi, S. Mohamadi, Two level job scheduling and data replication in data grid, International Journal of Grid Computing & Applications (IJGCA) 1 (2010) 23–37.

[21] Reddy, Hemant Kumar, Manas Patra, and Diptendu Sinha Roy. "Adaptive execution and performance tuning of parallel jobs in computational desktop grid using GridGain." Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference on. IEEE, 2012.

[22] Reddy, K. Hemant K., et al. "An Adaptive Scheduling Mechanism for Computational Desktop Grid Using GridGain." Procedia Technology 4 (2012): 573-578.