Research Journal of Applied Sciences, Engineering and Technology 8(23): 2334-2342, 2014
DOI:10.19026/rjaset.8.1237
ISSN: 2040-7459; e-ISSN: 2040-7467
© 2014 Maxwell Scientific Publication Corp.
Submitted: September 22, 2014 Accepted: October 24, 2014 Published: December 20, 2014

Research Article

Data and Job Aware Community Scheduler Framework for Grid Scheduling Problems

[1]G. Kalpana and [2]D.I. George Amalarethinam
[1]Department of CSE, SRM University, Kattankulathur, Chennai,
[2]Department of Computer Science, Jamal Mohamed College (Autonomous), Tiruchirappalli, India

Abstract: Grid technologies have brought the promise of flawless combination of distributed heterogeneous resources. Scheduling in Grid computing is the hot topic of research and challenging to manage task, job in efficient manner. For instance, tasks are assumed to include all data needed for its computation or tasks are just the processes and data is assumed to be available in Grid nodes. All the existing works doesn't focus on the data aware scheduling framework and it is almost impossible to make an optimal or approximate optimal scheduling for the end-to-end workflow with considering the intermediate data movement in grid computing environment. In order to solve this problem in this study a novel Community Scheduler Framework (CSF) approach is proposed for solving the job and data aware scheduling problem together and it can be integrated to grid host environment. The system is able to find data-affinity hosts for user requested jobs and to adjust the data replicas dynamically according to the job load. The proposed work reviews the policies for scheduling of grid jobs in the context of data and task aware-intensive applications. Proposed data and task aware CSF Meta scheduler framework makes it possible for job requests to set data needs not only as absolute requirements but also as functions for resource ranking. As the experimental results show that, this makes it more flexible than currently used resource brokers to implement different data-aware scheduling algorithms. The experimentation of the proposed Meta scheduler work is implemented with the help of grid simulation toolkit.

Keywords: Community Scheduler Framework (CSF), data aware scheduling, grid computing, grid simulation toolkit, Job aware scheduling, scheduling

INTRODUCTION

Grid Computing is a type of parallel and distributed system (Gandotra *et al*., 2011) that involves the integrated and collaborative use of computers, networks and scientific instruments depending on their ease of use, capability, cost and user requirements. These days, heterogeneous computing networks can be linked to each other through grid technology as it seems to be fully integrated machine. Then, very complex application programs can be implemented which needs high processing power and large amount of data (Foster *et al*., 2008). For instance, with the grid technology, several PC computers can be connected with each other. Grid computing has no restraint owing to its geographical area and the type of furtively resources. In certain cases, a grid network can be measured as a series of numerous big branches, different kinds of microprocessors, thousands of PC computers and workstations in all over the world. Grid network makes connection between those heterogeneous systems which have no steadiness among them. It also recognizes different types of resources and maintains to remote reach to them and lastly makes possible huge and complicated processing to be implemented with huge amount of resources as distributed in a powerful context (Broberg *et al*., 2008).

Scheduling in Grid computing is the hot topic of research and challenging due to heterogeneity and dynamism of resources in grid. In the simplest of cases, scheduling of jobs can be done in a blind way by simply assigning the incoming tasks to the available compatible resources. Yet, it is more profitable to use more advanced and complicated schedulers.

The scheduling system is able to provide parallel computational environments link to perform easier owing to the same features of programs and resources. For a moment, it can be said that grid computing is a software frame which collects resource status data, prepare suitable resources, anticipates the efficiency of each resource and determines the best resource (Banino *et al*., 2004). It can be said that, grid resource management system is responsible for controlling grid resources and its goal is to encourage efficiency (Heymann *et al*., 2000).

Most of the present Grid approaches are task-oriented approaches. For example, tasks are considered to include all required data for its tasks are just the processes and data is assumed to be available in Grid nodes. However, with the ever-increasing complexity of

Corresponding Author: G. Kalpana, Department of CSE, SRM University, Kattankulathur, Chennai, India

large-scale problems in which both tasks and data are to be scheduled, an integrated scheduling approach that would optimize allocation of both the task and the data is required. Data processing in scientific workflows slowly attains more attention owed to large amounts of data generated by complex scientific workflows will considerably increase the turn time of the whole workflow. It is approximately not possible to make an approximate optimal scheduling for the end-to-end workflow without considering the intermediate data movement.

In order to reduce the difficulty of the grid scheduling problem, several researchers are constrained by various unrealistic assumptions, which result in non-optimal scheduling in practice. A constraint forced by the majority of researchers in their algorithms results in their computation site can only start the execution of other tasks after it has completed the execution of the present task and delivered the data generated by this task. To deal with the data-aware workflow-scheduling problem, a sophisticated algorithm is used to pay attention on the dependencies of these tasks. In order to solve this data and task scheduling problem in the grid computing a Community Scheduler Framework (CSF) is proposed in this research. The proposed CSF integrates both data and job scheduling process.CSF schedulers would generally be expected to react to the dynamics of Grid system, typically by evaluating the present load of the resources and notifying when new resources join or drop from the system based on co-scheduling across multiple resource managers and economic scheduling.

LITERATURE REVIEW

Different types of scheduling are found in Grid systems as could have different scheduling requirement task independent or dependent; in contrast, the Grid environment features themselves impose restrictions such as dynamics, exploit of local schedulers, centralized or decentralized approach, etc. Grid workflows resolve several complex problems in Grids necessitate the combination and orchestration of several processes. This class of applications is known as Grid workflows. Such applications can take advantage of the power of Grid computing; still, the uniqueness of the Grid environment make the coordination of its execution very difficult (Cao *et al*., 2003; Yu and Buyya, 2006).

In static scheduling, it is considered that data associated to both grid resources and tasks of an applied program while scheduling is existing (Casavant and Kuhl, 1988). In dynamic scheduling, the main thought is that a task allocated is implemented while the applied program is running. This method is functional as it is not possible to decide running time, branching orientation and indefinite number of loop repetition and real time of tasks. Grid computing can be performed in two types of static and dynamic (Braun *et al*., 2000). Decision making is assigned to central scheduling about global scheduling or shared among several distributed scheduling (Christodoulopoulos *et al*., 2009).

There are several algorithms like genetic algorithms and heuristic approaches which are presented to schedule data. Radha and Sumathy (2009) made a comparison for those algorithms to find their performance in the data scheduling to select the best optimal algorithm. Even though those algorithms are functioned well, they have certain drawbacks which were corrected by using the techniques like Stork (Kosar and Balman, 2008) which is well suited in data placement and data movement. The Stork also contains disadvantages like congestion in the network through which the data is transmitted to the destination node (Kosar and Livny, 2004).

All the models presented here have either poor accuracy or they need a lot of information to be collected. Unfortunately, users do not want to present this information or have no idea about what to transfer to a data transfer tool. They need to make a projection of their data transfer throughput and must gather the information to optimize their transfer without caring about the characteristics of an environment and the transfer at hand. For individual data transfers, instead of relying on historical information, the transfers should be optimized based on instant feedback.

PROPOSED DATA AND JOB AWARE SCHEDULING CSF FRAMEWORK

Grid scheduling problem is really a challenging task. Job scheduling problem has become one of the challenging issue with the advance of distributed computing research. As new computation paradigms emerge such as many-task computing and cloud computing, the scheduling problem has become a major bottleneck in improving the performance of the proposed system. Dealing with many limitation and optimization condition in a dynamic environment it is very complex and computationally hard. In order to solve a job and data scheduling problem in grid computing environment, CSF is proposed which integrates both the information about the data and the job service for specific grid environment. The several number of tasks which is grouped into the job, then the data aware scheduling is performed. CSF is an open-source implementation of a number of Grid services, which together perform the functions of a grid meta scheduler or community scheduler.

Made some of the modification in the CSF framework by adding the task completion time for each one of the job in the job service stage and reservation service stage some of the additional requirements and rank expressions to account for the presence of data also defined to perform job and data aware scheduling task. The CSF classes can be extensive to provide more domain specific community schedulers and support many different kinds of grid operation models. By make use of the open source CSF, grid scheduling execution make sure that they interrelate with resource

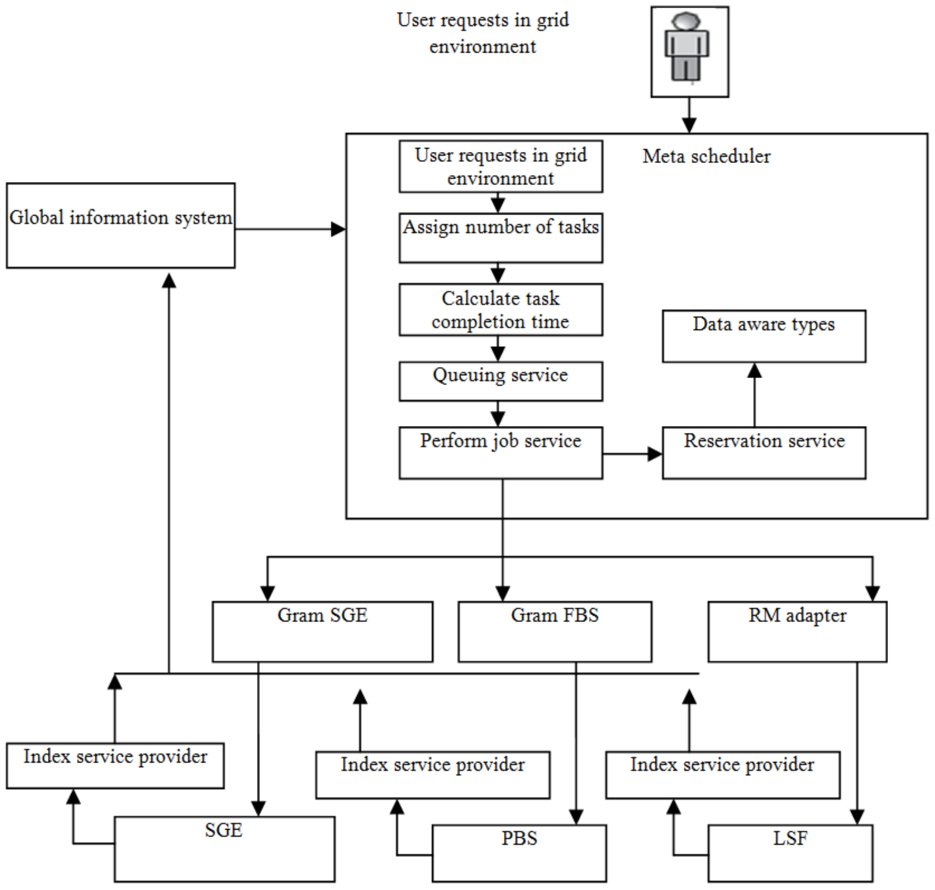


Fig. 1: Proposed CSF architecture for data and job aware scheduling

managers by means of standard interfaces underlying details of the specification (Platform Computing Co., 2004).

The purpose of the met scheduler is to permit end-users to act together with underlying resource managers in a system and efficiently solve job and data aware scheduling problem by defining the protocols that interact with resource managers, requirements of the each data, task completion time for each job. In addition, the Meta scheduler is a high-level abstraction for some of the concepts that are key to resource management including a "job", a "resource reservation" and a "scheduler". The scheduling framework of the proposed modified CSF scheduler for data aware and job scheduling problem in grid environment is shown in Fig. 1.

Consider a number of tasks which is requested by user. The main objective is to minimize the turnaround time of this group of tasks and efficiently handle data in scheduling process in different computation sites, in which order these tasks should be executed and how the data flow between these tasks should be scheduled. Let us consider N_{CG} be the number of computation sites in grid host environment. N_{dg} is the number of data sites in grid host environment for each number of the task. N_{tg} is the number of task that run in the grid host environment P_i is a set of computation sites p_i. $P = \{p_i\}, i = 1, \dots N_c$. D is the set of the data sites of $d_{ig} =, i = 1, , \dots N_{gd}$ in grid host environment. C is a set of computation links between computation sites $c_{p_i p_j}$, data sites $c_{d_k d_l}$ and computation-data sites $C = \{c_{p_l d_k}, c_{d_k d_l}, c_{p_i d_k}\}, i, j = 1, \dots N_{gc}, k, l = 1, \dots . N_{gd}$.

Each task consists of two phases, one is computation and the other is intermediate data transfer. Once a task starts, it cannot be stopped until the computation work is done. An intermediate data transfer is initiated; it cannot stop until this transfer is finished. Each task t_i needs a finite amount of time t_{ij} to finish if it can be scheduled to computation site p:

$$Ext(t_i, p_j) = \begin{cases} t_{ij} t_i \text{ is applicable on } p_j \\ \quad \infty \text{ otherwise} \end{cases} \tag{1}$$

Then the execution time is calculated for each task the priority level or queuing service level is calculated based on the weight value with waiting time:

$$J_{wt} = \left(\frac{ext(t_i, p_j)}{avgwt}\right)^{\alpha} \tag{2}$$

where, $\mathrm{ext}(\mathrm{t_i}, \mathrm{p_j})$ is the time that a job waits in the queue and avgwt is the average job waiting time in the

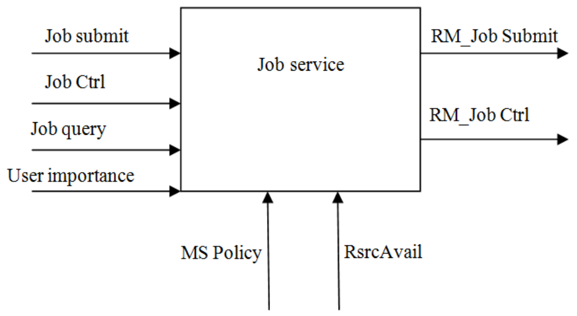


Fig. 2: Perform job service

system and α is the weighting factor. Job running time also can be a factor for a dynamic priority by favoring shorter running jobs over longer running jobs:

$$J_{Rt} = \left(\frac{r}{avgrt}\right)^{\beta} \tag{3}$$

where, r is the average runtime for the tasks of a job and avgrt is the average of the average runtime for the Map tasks for all jobs and β is the weight factor. Further, consider the number of remaining unscheduled tasks for a job as another as:

$$J_{JS} = \left(\frac{n}{avgnt}\right)^{\gamma} \tag{4}$$

where, n is the number of remaining tasks for the job and avgnt is the average number of remaining tasks per job over the entire queue and γ is the weighting factor. The combination of these above Eq. (2-4) becomes the final priority for the job in the system as:

$$priority = \left(\frac{ext(t_i, p_j)}{avgwt}\right)^{\alpha} * \left(\frac{r}{avgrt}\right)^{\beta} * \left(\frac{n}{avgnt}\right)^{\gamma} \tag{5}$$

The importance of each user in the queue may be accounted as:

$$U_q = u^{\delta} \tag{6}$$

After the queue service of the each user is calculated for number of the jobs then performs the job service task. A Job Service provides an interface for insertion jobs on a resource manager and interacting with the job formerly it has been dispatched to the resource manager. The Job Service gives essential matchmaking capabilities between the needs of the job and the underlying resource manager for processing the job. More highly developed Job Services levels are illustrated in Fig. 2. It considers the advanced job features such as interactive execution, parallel jobs across resource managers and jobs with requirements based on SLAs.

The interfaces provided by the Job Service in Fig. 2 include:

Job submit: User job submission requests from the grid host environment. A job is submitted with resource requirements in RSL (Resource Specification Language) syntax, user importance with the completion time of the each job, with a reservation id in order to bind the job to a specific resource reservation. Otherwise a job is submitted with the name of a specific resource manager on which to run the job.

Job ctrl: Controls a job after it has been instantiated from user in the grid host environment. This would include the ability to suspend/resume, checkpoint and kill a job.

Job query: Query the status of a job and queue status of the job in the queuing service. This includes information about the each one of the job such as execution time of the each job, status of a job and resource usage.

The Job Service make use of information such as policies, which are defined at the met scheduler level and resource information based on available resource managers, queues, host and job statuses as provided by the Global Information Service. The Job Service uses the Resource Manager Adapter (RM Adapter) to submit jobs to the fundamental resource manager to control running jobs. The Job Service also makes use of the Global Information Service to store its own state information of the each job that running on the grid host environment. The information stored includes job submission information, job status and a list of jobs that a particular Job Service instance can manage, which provides the ability for the Job Service to recover from any faults.

The Reservation service allows end-users or a Job Service to reserve resources under the control of a resource manager to guarantee their availability to run a job and their data are accessed by user based on the user request and their priority. This service allows reservations for any type of resource (e.g., hosts, software licenses, or network bandwidth). These types of the reservations service categories also additionally consist of the following information to perform the reservation process. Reservations can be is made, a Job Service sends a job to a resource manager that is associated to a provided reservation. Some of the policy decisions prepared by the Reservation Service for Platform CSF contain how many hosts a particular user or user group can reserve at a time, when reservations can be done and what types of hosts can be reserved. The Reservation service provides the following interfaces:

Add Rsv: Request a new reservation with a particular resource requirement, starting at a specific time for a given duration.

Delete Rsv: Remove a reservation.

Query Rsv: Retrieve the details of a particular reservation.

Job category information:
Transfer: This job type is for transferring a complete or partial file from one physical location to another one. This can include a get or put operation or a third party transfer.

Allocate: This type of job is employed for allocating storage space at the destination location, allocating network bandwidth, or found a light-path on the route from source to destination. On the whole, it deals with all essential resource allocations prerequired for the placement of the data.

Release: This job type is used for releasing the corresponding resource which is allocated before.

Remove: This job is used for physically removing a file from a remote or local storage server, tape or disk.

Locate: Logical file name is given; this job check with a Meta data catalog service:

- **Close Data** (logical file name): Requirements expression. For each Computing Element (CE), it is examined as True only if specified data is seized by a close Storage Element (SE) and as False otherwise:
- **Has close data (logical file name):** Rank expression. For each CE, it is evaluated as 1 if specified data is held by a close SE and as 0 otherwise.
- **Size close data (logical file name):** Rank expression. For each CE, evaluated as the size of the specified data if held by a close SE and as 0 otherwise.

Register: This type is used to register the file name to a Meta data catalog service.

Unregister: This job unregisters a file from a Meta data catalog service.

The Reservation Service uses the information about the existing resource managers and policies that are distinct at the Meta scheduler level based on the job completion time, user request service in the queue service and will build with the use of a logging service to log reservations. The Reservation Service uses the RM Adapter interface to create and delete existing reservations. The Reservation service implements the Agreement Factory Type as defined in the OGSI Agreement specification so that a client can make reservations based on agreement terms supplied. Reservations can also be specified using RSL (Fig. 3).

Global information service: The Global Information Service offers a warehouse for required information to function the Meta scheduler is dissipated in the Fig. 4 for precise data and job scheduling process. Service combined various lower-level Grid services with the purpose of replicate service data between Grid services and to create a dynamic data-generating and indexing node. The mechanism used in the Global Information Service consists of:

Registry components: Provides static information about installed services such as service name, location and service ACLs. Each Platform LSF-based cluster registers itself in the Cluster Registry, enabling Job and Reservation services to find available resource managers.

Service data provider manager: Manages the acquisition of dynamic data into the Global Information Service via external programs. Resource Information Provider Service (RIPS) is to collect job and resource information from each cluster in the system. This includes aggregate information about the individual resource managers (e.g., how many jobs in the system, average job turnaround time, what host types are available, etc.). Also, it collects more detailed information about individual jobs, queues, reservations and hosts.

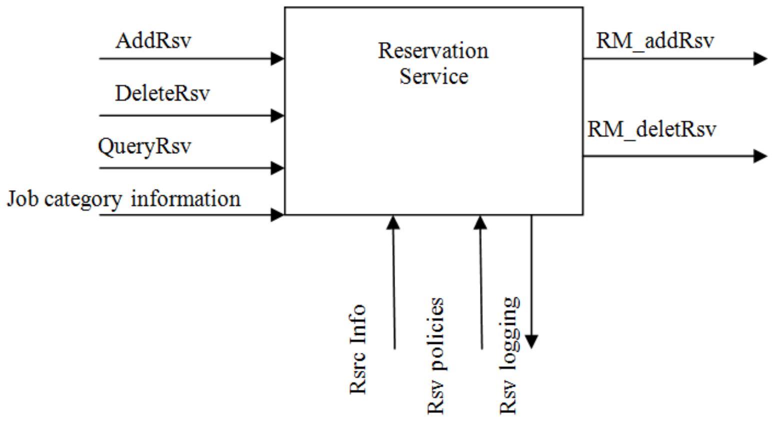


Fig. 3: Reservation service

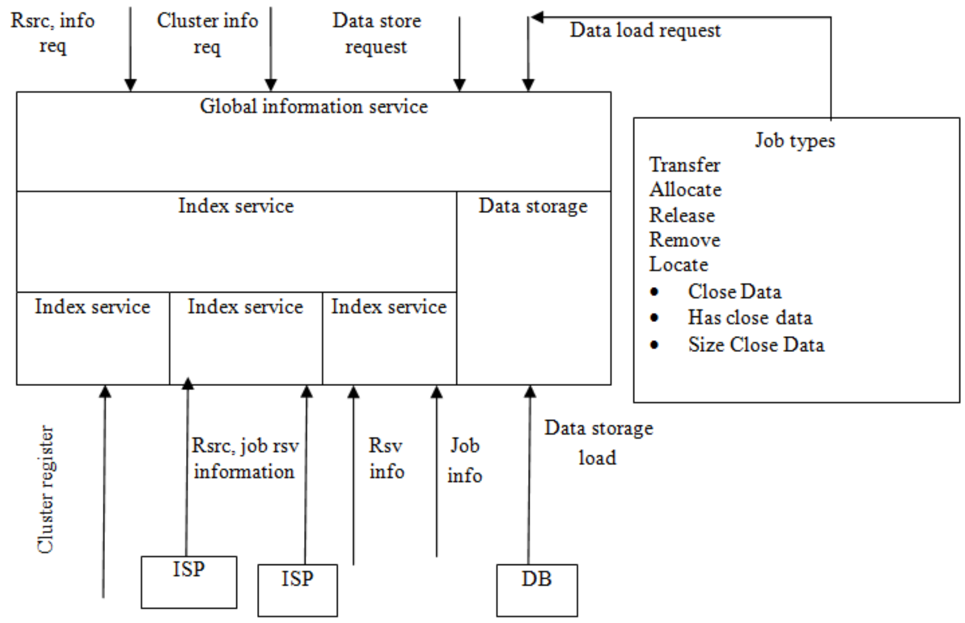


Fig. 4: Global information service

Service data aggregator: Here the service data of other services are collected and provides notification mechanisms. This is used in the Global Information Service to aggregate information about Job Service instances (e.g., submission, job status, execution, Job Service instance, etc.) and about Reservation Service instances (e.g., requests and individual reservation details).

Queuing service: The Queuing Service endow with scheduling capacity to the Meta scheduler. The CSF execution of the Queuing Service outfits a plug-in scheduling structure. Schedulers are java classes that execute the schedPlugin interface, which consist of four methods. The two most important methods are used at some point in the scheduling cycle that is sched Order and schedMatch. A series of jobs is listed one by one in the system that is agreed to schedOrder, which order the list based on which jobs have priority based on the implemented scheduling algorithm. The sched Match method goes with jobs among the resource managers (the RM Adapter resource managers or GRAM job factories) that are accessible to the scheduler. The job-to-RM mappings are refered as a “job decision”.

If there are multiple scheduling policies in the Queuing Service, the dissimilar schedulers will be call upon in certain order based on the job list and on the job decisions, allowing the effects of one scheduling connect (plugins) to be combined with the effects of another. For instance, two scheduling plugins are given as a part of the CSF Queuing service together with a FCFS queue and a job throttle. The FCFS scheduler orders the list of jobs based on their submission time to the Queuing service. During the matching phase, jobs are either mapped to an RM, based on a given cluster name or reservation id, or else the FCFS scheduler identifies an RM, based on a round-robin choice from the available RMs.

The job throttling scheduler not supposed to reorder the list of jobs from the FCFS scheduler. Its match phase makes sure that too many jobs are not sending to the same RM at the same time, so that:

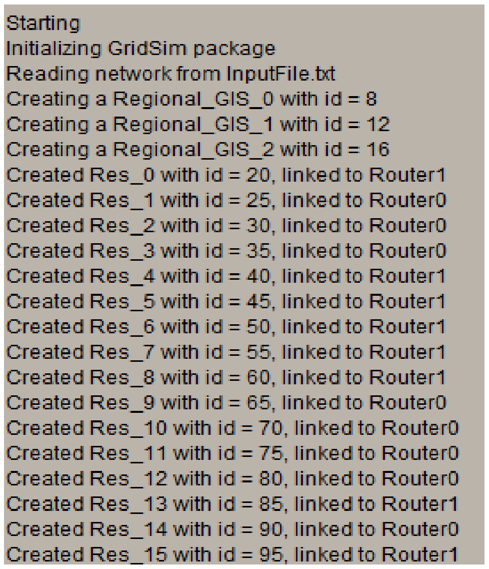
- An RM is not overwhelmed
- A job is not queued at an RM which may not run the job for awhile, when another RM may be available earlier to run the job

RM adapter: The RM Adapter is an interface for communicating with primary resource managers. The primary purpose of the RM Adapter is to provide a link between the Open Grid Services Infrastructure (OGSI)-compliant grid clients and legacy resource managers that do not have a native Grid service interface.

EXPERIMENTAL RESULTS AND DISCUSSION

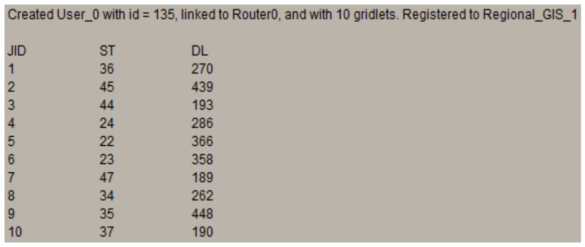
Experiments were conducted in order to evaluate the functionality and performance of the proposed approach implementation using resources of the files which are running under grid simulation tool. GridSim is a toolkit for modeling and simulation of resources and application scheduling in extensive parallel and distributed computing surrounding. The need for simulating complex systems is identified. The selected file project provides large-scale computing infrastructure for researchers conducting studies in data-intensive and computing-intensive applications from high energy physics, earth and life sciences.

It is employed to simulate application schedulers for single or multiple administrative areas distributed computing systems such as clusters and Grids.



```
Starting
Initializing GridSim package
Reading network from InputFile.txt
Creating a Regional_GIS_0 with id = 8
Creating a Regional_GIS_1 with id = 12
Creating a Regional_GIS_2 with id = 16
Created Res_0 with id = 20, linked to Router1
Created Res_1 with id = 25, linked to Router0
Created Res_2 with id = 30, linked to Router0
Created Res_3 with id = 35, linked to Router0
Created Res_4 with id = 40, linked to Router1
Created Res_5 with id = 45, linked to Router1
Created Res_6 with id = 50, linked to Router1
Created Res_7 with id = 55, linked to Router1
Created Res_8 with id = 60, linked to Router1
Created Res_9 with id = 65, linked to Router0
Created Res_10 with id = 70, linked to Router0
Created Res_11 with id = 75, linked to Router0
Created Res_12 with id = 80, linked to Router0
Created Res_13 with id = 85, linked to Router1
Created Res_14 with id = 90, linked to Router0
Created Res_15 with id = 95, linked to Router1
```

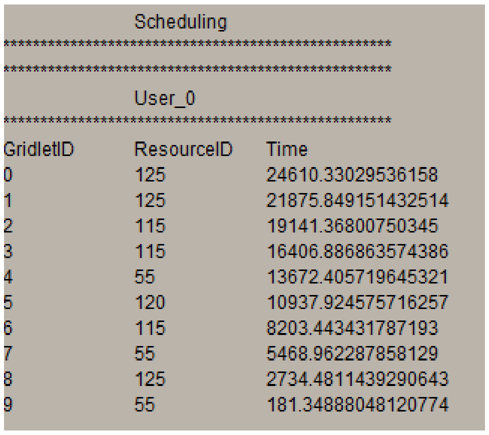
Fig. 5: Initialization of the resources



```
Created User_0 with id = 135, linked to Router0, and with 10 gridlets. Registered to Regional_GIS_1

JID        ST        DL
1          36        270
2          45        439
3          44        193
4          24        286
5          22        366
6          23        358
7          47        189
8          34        262
9          35        448
10         37        190
```

Fig. 6: Job creation for user 1



```
                 Scheduling
*****************************************************
*****************************************************
                 User_0
*****************************************************
GridletID        ResourceID        Time
0                125               24610.33029536158
1                125               21875.849151432514
2                115               19141.36800750345
3                115               16406.886863574386
4                55                13672.405719645321
5                120               10937.924575716257
6                115               8203.443431787193
7                55                5468.962287858129
8                125               2734.4811439290643
9                55                181.34888048120774
```

Fig. 7: Grid scheduling results of the user

Application schedulers in Grid environment is called as resource brokers, carry out resource discovery, selection and aggregation of a varied set of distributed resources for an individual user. It represents that, each user has their own private resource broker and therefore, it can be targeted to optimize for the needs and objectives of its owner. Where schedulers, maintaining the resources such as clusters in a single administrative domain, contains full control over the policy used for allocation of resources. This means, all the users required submitting their jobs to the central scheduler, which can be embattled to perform global optimization such as higher system usage and overall user satisfaction based on resource allocation policy or optimize for high priority users.

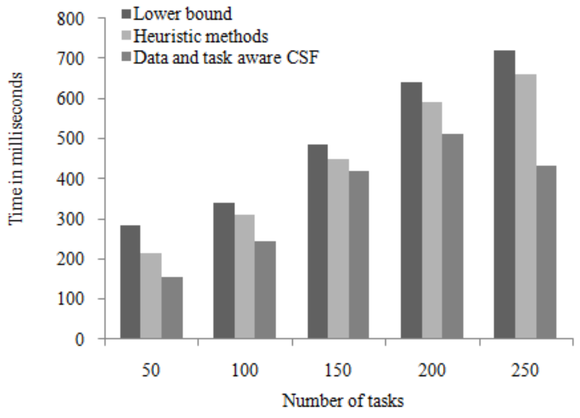


Fig. 8: Data and job aware scheduling for homogeneous distributed systems

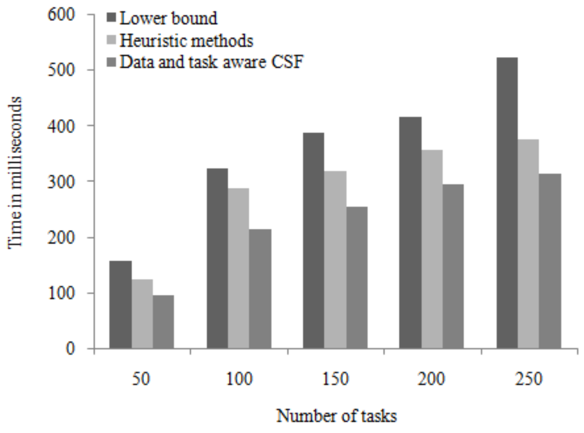


Fig. 9: Data and job aware scheduling for heterogeneous distributed systems

The number of the resource or task initialized using the grid simulation tool is shown in the Fig. 5 and 6.

In the Fig. 6 shows the job creation of the each user with data description length of the individual with their JID.

In the Fig. 7 shows the grid scheduling results of the individual user with the number of the resources allocated to single user with their gridletID, execution time comparison results of the each resource also mentioned in detail.

The transfer time for the data to be delivered to a computing site is the quotient of the summation of the data and the summation of the bandwidth from each storage site to the computing site. The transfer time for all these computing sites is the maximum among them for existing methods. Figure 8 and 9 shows the

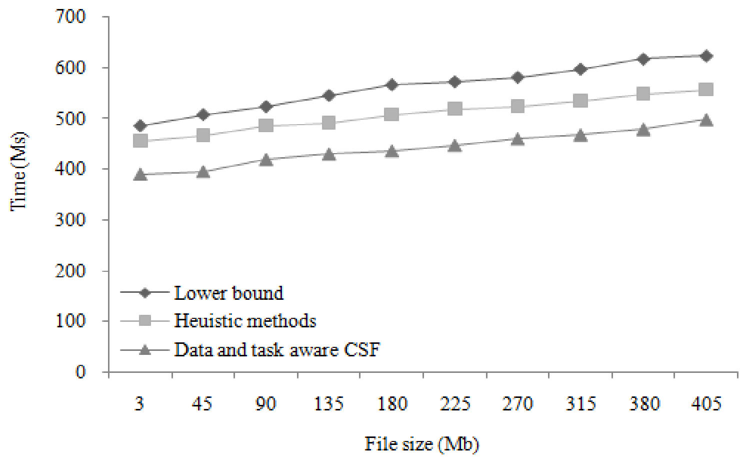


Fig. 10: Job completion time for methods

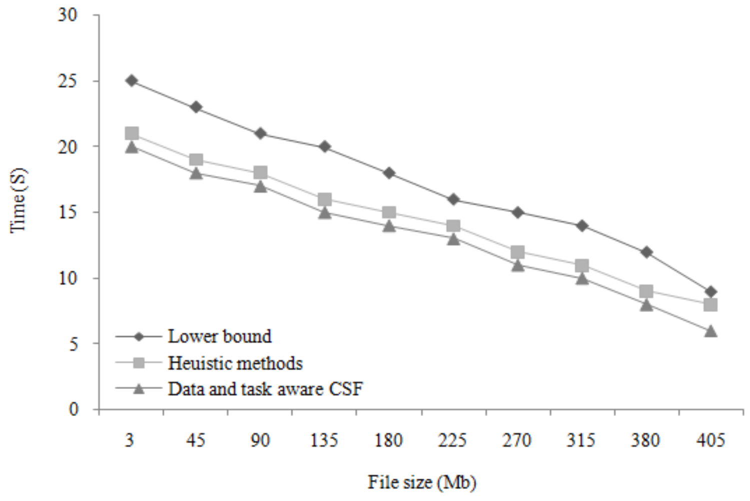


Fig. 11: Data transfer time for methods

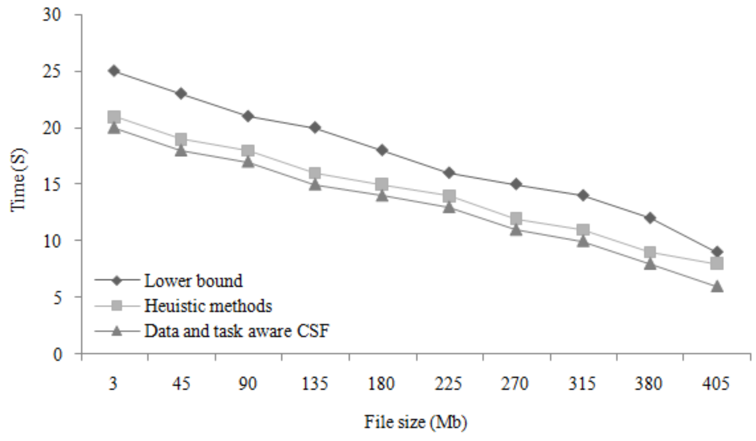


Fig. 12: Job turnaround time for methods

experimental results for different number of data files can be compared with different data aware scheduling methods and compare with existing random and the heuristic data aware scheduling methods, it shows that the proposed data and job aware scheduling CSF framework have taken less time to complete the task than the existing methods for both homogenous and heterogeneous distributed system.

Figure 10 shows the job completion time of the proposed data and Job aware CSF scheduling is low and the existing random and the heuristic data aware scheduling methods.

Figure 11 transfer time of the proposed data and job aware CSF scheduling is low and the existing random and the heuristic data aware scheduling methods.

Figure 12 shows the job turnaround time of the proposed data and job aware CSF scheduling is lower than the existing random and the heuristic data aware scheduling methods.

CONCLUSION

In this study, a novel job and data aware scheduling method is proposed for solving Grid computing related to data and job aware scheduling problems and the integration of Grid schedulers into Grid architectures. After introducing some Grid job category information and their job completion time for each user in the grid host environment the CSF Meta scheduler perform job and data aware scheduling in the Grid computing domain.CSF provides basic capabilities for scheduling and can be used as a development toolkit for implementing community schedulers. This study revealed the complexity of the data and job aware scheduling problem in computational Grids when compared to scheduling in classical homogenous and heterogeneous distributed systems have less job completion time, data transfer time for the design of efficient Grid schedulers. The fact that different VOs may present heterogeneous needs and policies motivates us to argue in favor of a flexible system where different allocation algorithms that make use of data location information can be supported. This idea has led to the development of an enhanced Grid Meta scheduler for data and job aware scheduling problem. From a more general point of view, further research is needed in the area of coordinated planning of data placement and jobs allocation in grid environments.

REFERENCES

Banino, C., O. Beaumont, L. Carter, J. Ferrante, A. Legrand and Y. Robert, 2004. Scheduling strategies for master-slave tasking on heterogeneous processor platforms. IEEE T. Parall. Distr., 15: 319-330.

Braun, T.D., H.J. Siegel, N. Beck, L.L. Boloni, M. Maheswaran, A.I. Reuther, J.P. Robertson, M.D. Theys, B. Yao, D. Hensgen and R.F. Freund, 2000. A comparison study of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Technical Report TR-ECE-00-4, School of Electrical and Computer Engineering, Purdue University.

Broberg, J., S. Venugopal and R. Buyya, 2008. Market-oriented Grid and utility computing: The state-of-the-art and future directions. J. Grid Comput., 3(6): 255-276.

Cao, J., S.A. Jarvis, S. Saini and G.R. Nudd, 2003. GridFlow: Workflow management for grid computing. Proceeding of the 3rd International Symposium on Cluster Computing and the Grid (CCGrid'03), pp: 198-205.

Casavant, T.L. and J.G. Kuhl, 1988. A taxonomy of scheduling in general-purpose distributed computing systems. IEEE T. Software Eng., 14(2): 141-154.

Christodoulopoulos, K., V. Sourlas, I. Mpakolas and E. Varvarigos, 2009. A comparison of centralized and distributed meta-scheduling architectures for computation and communication tasks in Grid networks. Comput. Commun., 29: 1172-1184.

Foster, Y.Z., I. Raicu and S. Lu, 2008. Cloud computing and grid computing 360-degree compared. Proceeding of the IEEE Grid Computing Environments Workshop (GCE'2008). Austin, TX, pp: 1-10.

Gandotra, I., P. Abrol, P. Gupta, R. Uppa and S. Singh, 2011. Cloud computing over cluster, grid computing: A comparative analysis. J. Grid Distr. Comput., 1(1): 1-4.

Heymann, E., M.A. Senar, E. Luque and M. Livny, 2000. Adaptive scheduling for master-worker applications on the computational grid. Proceeding of the 1st IEEE/ACM International Workshop on Grid Computing (GRID 2000), Springer-Verlag, London, U.K., pp: 214-227.

Kosar, T. and M. Livny, 2004. Stork: Making data placement a first class citizen in the grid. Proceeding of the 24th International Conference on Distributed Computing Systems, pp: 342-349.

Kosar, T. and M. Balman, 2008. A new paradigm: Data-aware scheduling in grid computing. Future Gener. Comp. Sy., 25(4): 406-413.

Platform Computing Co., 2004. Open Source Metascheduling for Virtual Organizations with the Community Scheduler Framework (CSF)[WP]. Retrieved from: http://www.cs.virginia.edu/~grimshaw/CS851-2004/Platform /CSF_architecture.pdf.

Radha, B. and V. Sumathy, 2009. Comparison of ACO and PSO in grid job scheduling. CIIT Int. J. Network. Commun. Eng., ISSN 0974-9713 and Online: ISSN 0974-9616, DOI: NCE102009003.

Yu, J. and R. Buyya, 2006. A taxonomy of workflow management systems for grid computing. J. Grid Comput., 3(3): 171-200.