Pipelining and Replica Optimization for Infrastructure as Utility Grid

N.Mathiarasi
PG student
Department of Computer Science and Engineering
Nandha College of Technology
Erode

K.Abinaya
PG Student
Department of Computer Science and Engineering
Nandha College of Technology
Erode

K.Kiruthikadevi
Assistant Professor
Department of Computer Science and Engineering
Nandha College of Technology
Erode

ABSTRACT
Utility Grid is the collection of computer resources in a heterogeneous distributed environment which encourages researchers to investigate the benefits and drawbacks on executing workflows. While processing a large amount of workflows in utility grid environment, one of the most challenging problems is scheduling of workflows without replications/repetitions. Already, a two-phase approach called partial critical path approach to schedule the workflows in grid environment which aims to minimize the cost of workflow execution under user defined deadline. However when someone applies pipelining mechanism with such algorithm, the results obtained is not more optimized. In this paper, PCP algorithm is adopted with Replication Optimization phase (R-PCP) along with streaming pipelining mechanism. Applications such as scientific simulations, sensor network analysis generate huge amounts of data, which must be streamed efficiently. The streaming services must meet an application's quality of service (QoS) and this mechanism should guarantee that no data are lost during processing. The proposed mechanism operates on polynomial time complexity, which is suitable for optimizing large number of workflows. The main objective of this paper is to: implement pipeline mechanism in a large grid environment and making grid environment to work in an optimized way. The simulation results are shown on the paper gives promising results.

General Terms
Grid computing, scheduling in grid, Pipelining, Replication scheduling.

Keywords
Grid workflow scheduling, Replica optimization, Pipelining of utility grid workflows, utility grids, QoS-based scheduling.

1. INTRODUCTION
Utility grid in comparison with other traditional community grids, they provide best effort service. There are many research works that address the problem of scheduling without replication in Utility Grids. Utility Grids guarantee the required QoS of users via Service Level Agreements (SLAs) [1]. An SLA is a contract between the provider and the consumer of services describing the qualities of service offered to the consumer. The price has a key role in this contract: it encourages providers to advertise their services to the market, and also encourages user to meet their requirement satisfaction [3]. Here adapt a mechanism called Replica Optimization is used to avoid replication of workflows. In order to maximize the performance and efficiency pipelining mechanism is used on each site. The objective of Replica Optimization mechanism is to remove the replication of jobs scheduled to a grid. This paper has another mechanism called pipelining which paves way to make computation of large scientific workflows in an efficient way. When RPCP is handled, every site is checked for the replications and if any replications occur, they will be removed with the help of a tool called Optor. Initially, the algorithm checks the overall deadline for the process to be completed and checks the configuration which is needed to complete the job execution. Whenever a job is given to a grid environment, initially jobs are scheduled to individual site, and then they are processed. Each site is checked for the configuration, whether it can process a specific task or not. According to which the jobs are allocated. Pipelining mechanism is done in every site (within computation element).

Utility Grid model consists of one (or more) workflow management system [7], pipelining mechanism, and several Grid Service Providers (GSPs), each of which offers services. Replica Optimization process is done with the help of Resource Broker.

2. SCHEDULING SYSTEM
The proposed scheduling and replica optimization system model consists of an application model called workflow management model, a utility Grid model with Grid Service Providers (GSPs), and a performance criterion for pipelining, scheduling and replica optimization. An application is given in a form called directed acyclic graph *G (T, E)* where T is a set of n tasks $\{t_1, t2\ldots., tn\}$, and E is a set of arcs. Every arc $e_{i,j}=(t_i,t_j)$ represents a constraint with precedence that indicates that the task t_i should complete its execution before the task t_j starts. In a given task in the form of graph, a task without any parent is referred as an entry task, and a task without any child is referred as an exit task. The algorithm requires only single entry and single exit task. One should always add two duplicate tasks called t_{entry} and t_{exit} to the beginning and the end of the workflow, respectively. These duplicate tasks have zero execution time and they are connected with zero-weight dependencies to the actual entry and exit tasks.

The workflow *ti* is processed by $m_{i\ \text{services}}$ $S_i= \{s_{i1}, si2\ldots si,mi\}$ from different service providers with different QoS attributes. According to the model, a service is defined as a

computational resource, and also the software required for executing a task. There are many QoS attributes for services, like execution period and its complexity, price, reliability, security, availability, and so on. Here, the experiment mainly concentrates on execution time and the cost. The cost of a service mainly depends on the execution time of a task, i.e., shorter execution times are more expensive because they need more configurations to run the job at shorter time. Assume that, $ET(t_{i,s})$ and $EC(t_{i,s})$ represents the estimated execution time and execution cost for processing task t_i respectively. Time complexity for executing a task is an important issue in Grid scheduling [13]. In addition to that, replica optimization needs additional execution time, which is resolved with the help of pipelining mechanism. Transferring data between tasks leads to more consumption of time and money. Assume that, $TT(e_{i,j,r,s})$ and $TC(e_{i,j,r,s})$ represents the estimated transfer time and transfer cost of sending the required data along $e_{i,j}$ from service r (processing task t_i) to service s(processing task t_j), respectively. Estimating the transfer time can be done using the amount of data to be processed and transmitted, and the bandwidth and latency information between services.

3. PARTIAL CRITICAL PATH ALGORITHM WITH REPLICA OPTIMIZATION AND PIPELINING

3.1 Central Idea

Initially workflows (task) are given as input. The proposed algorithm checks the configuration and overall deadline of the workflow is distributed over individual tasks. Each site consists of its own storage element and computation element. Storage element can store 100GB of data and computational element in every site consists of 50GB of data. These values can be changed according to the usage of data. Workflows are then scheduled to individual sites according to their configuration. This process is done with the help of a component called Resource Broker. Then, checking for the occurrence of replica is done with the help of a component called Optor, which checks every site for the replication of job. If replications occur, then the workflow with less priority is removed from the respective site. New workflow is scheduled for the site, after removing the replicated task from it. Pipelining mechanism is an efficient way to improve the rate of instruction execution. Pipelining mechanism is implemented in computation element of individual site. Here used streaming pipeline mechanism, which streams the workflows. By using the above mechanisms, the acquired results are in polynomial time, which is more efficient in nature.

4. REPLICA OPTIMIZATION

A Replica Optimization Service controls the occurrence of replicas [20]. Replica optimization gives information about when should a job requests a file, when and where replicas should be created and deleted. The main objective is to reduce job execution times through minimizing the use of network resources caused by transferring files between sites. For this simulator uses a replica optimization strategy, and some potential strategies. Replication/repetition is the main strategy of a distributed Data Grid. However, replication must be performed in a controlled fashion, as uncontrolled file copying all over the Grid would quickly saturate the network and the Grid would grind to a halt. The Replica Optimization Service (or Optimizer) on each site decides which replicas (from those stored throughout the Grid) jobs running on the site should access and which replicas to store on its own site. These decisions are controlled by a replica optimization strategy, and some of these strategies are explained in this section. The Optimizer (Optor) optimizes its local site, but any replication it undertakes should be for the benefit of the whole Grid. In other words an Optimizer performs local replica optimization but the aim is to achieve global optimization as the emergent result of local optimization. Thus every Optimizer has two goals:

- To minimize a single job's execution cost.
- To maximize the usefulness of locally stored files.
- To minimize latency.

Whenever an Optimizer is considering a file request, it executes the following tasks:

4.1 Replica Decision

If a requested file is not present on a site's Storage Element (SE), this process decides whether local replication of this file should take place. If the Optimizer decides not to replicate a file then the job must remotely access that file.

4.2 Replica Selection

When considering remote replicas, this process selects the best of those available. In general, the selection criterion depends on the chosen evaluation measure determined by the optimization strategy.

4.3 File Replacement

When a replica on remote site is selected for replication to the site's SE, the SE might not have sufficient alternative capacity to act as a spare. In this case, one or more local replicas must be deleted. The selection criteria for deciding which locally stored replicas to delete depend on some estimate of future value of each replica. A specific combination of algorithms for each stage defines a replica optimization strategy.

5. LFU

Cache management involves most commonly used algorithm called LFU (Least Frequently Used). Often memory access takes a fraction of the time that disk access takes but the amount of potential data to read from the disk is much more than the memory cache can hold. LFU replacement is used when most frequently used blocks of data is present in cache i.e. the least frequently used blocks are deleted to minimize the number of disk accesses required. In a Grid implementation the LFU-based strategy will always replicate files to storage local to the job's Computing Element when a job running on the Computing Element requests them. If the local storage is full, the file that has been accessed least frequently in the previous time is deleted, creating space for the new replicas to be stored.

6. AN ECONOMIC APPROACH TO REPLICA OPTIMIZATION

An economic model for data access and replication has been developed to deliver a more stable and adaptable strategy for replica optimization. In the model, data files are purchased by Computing Elements (CEs) for running jobs and by Storage Elements (SEs) to make an investment that will improve their expected future needs. CEs try to minimize the file purchase cost, while SEs attempts to maximize their profits. CEs and SEs interact with intelligent optimization agents which

perform the reasoning required in the model. The adoption of an economic approach has two main motivations. The first reason is to be able to make replication optimization decisions in a distributed manner. By using an economic model, the dynamism of the market can be exploited to make informed decisions at job execution time. Economic approach to the replica optimization involves maximizing the performance of the processing and to minimize the cost of execution.

6.1 Internals of Economic Model

The abstract architecture for Replica Optimization (RO) agents, shown in Fig.1, includes the following components:

6.1.1 Access Mediator (AM)

This component processes file requests from jobs running on a CE. For each requested file, it selects the site with cheapest execution cost and schedules the task according to it (see Section 4.2.2). The AM gathers bids for the file from local and remote Storage Brokers (SB), selects the winner of the auction and performs file payments.

6.1.2 Peer to Peer Mediator (P2P)

These are responsible for establishing and maintaining a peer to peer communications infrastructure between Grid sites. They exchange information between access mediator and storage broker. Resource Broker is also associated with this process. Hence faster results can be achieved.

6.1.3 Storage Broker

This component is responsible for listening for file request messages from the local P2P and Access Mediator. If the requested file is stored in the corresponding SE, it responds immediately to the P2P Mediator or else to Access Mediator. If the file is not stored in the corresponding SE, it starts to search in the remote site or any other local SE, in order to obtain a local replica of the file and be able to reply to the parent auction.

The storage broker manages the utility values and smaller computational values; hence they can be easily retrieved. Those files which are accessed frequently are stored in cache memory and they can be accessed easily. If the cache memory gets full then, use LFU algorithm or any other economic models to replace the cache memory with the new data.

7. WORKFLOW SCHEDULING PROCEDURE

1: procedure ScheduleWorkflow(G(T , E), D)

2: determine computation services which are available

3: add t_{entry}, t_{exit} and their corresponding dependencies to G

4: compute EST (t_i), *EFT* (t_i) and *LFT* (t_i) for each task in G

5: *AST* (t_{entry}) ← 0, *AST* (t_{exit}) ← *D*

6: mark t_{entry} and t_{exit} as assigned

7: call AssignParents(t_{exit})

8: end procedure

6: schedule P on $s_{i,j}$ and set *SS*(t_i), *AST* (t_i) for each $t_i \in P$

7: set all tasks of P as assigned

8: end procedure

8. REPLICA OPTIMIZATION PROCEDURE

1: Procedure Replicate *(path)*

2: *best* ← null

3: *t* ←first task on the path

4: while (*t* is not null) do

5: *s* next slower service € *St*

6: If (assigning *t* to *s* is not admissible)

7: Then *t* ← previous task on the path and continue

8: while loop

9: end if

10: If (*t* is the last task on the path) then

11: If (current schedule has a lower cost than

12: best) then

13: set this schedule as *best*

14: end if

15: *t* ← previous task on the *path*

16: else

17: *t* ← next task on the *path*

18: end if

19: end while

20: If (*best* is null) then

21: set sub-deadline *(t) =EST (t) +MET (t)* for all tasks

22: t on the path

23: else

24: set *EST* and sub-deadline according to *best* for

25: all tasks € *path*

26: end if

27: mark all tasks of the *path* as assigned

28: end procedure

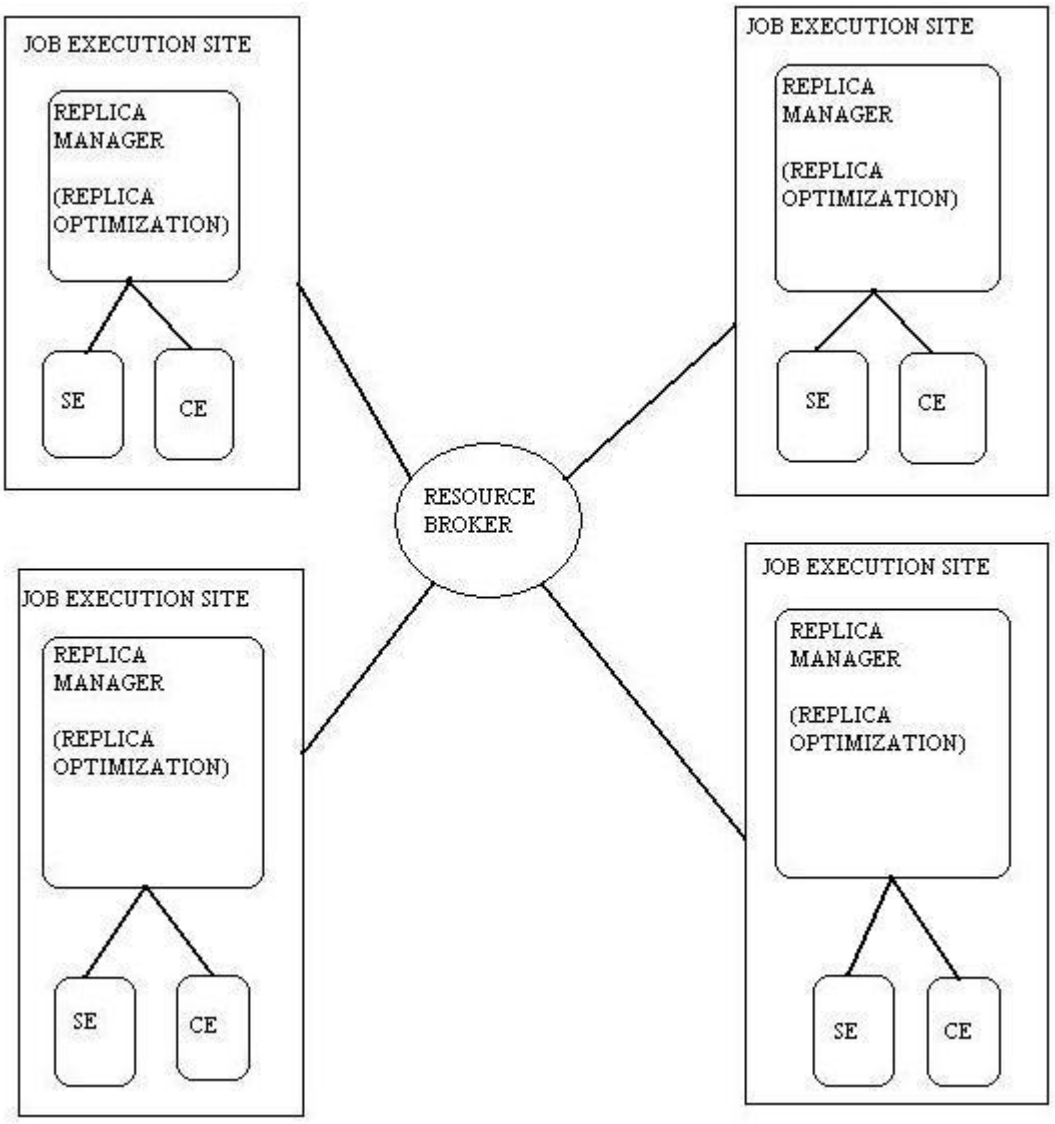


Fig 1: Components of the replica manager in economic model

9. PIPELINING OF WORKFLOWS

Pipelining is done within each processor. Here a technique called streaming pipeline which is shown below. Streaming pipelining to a task requires relaxing the basic assumptions of having tasks that support a simple request-response interaction, where a task reads its input as it starts and produces output once it is finished [25]. Given the goal of passing input data to a task as soon as it is produced by its predecessor (and avoid the blocking semantics), the workflow cannot wait for the task to finish before restarting it with the next element (like with the buffering semantics) or start another parallel instance (superscalar semantics).

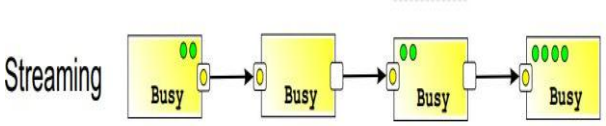


Fig 2: Streaming pipeline mechanism for large workflows

Thus, only if tasks allow a more flexible interaction based on multiple requests and multiple responses, the streaming semantics can be fully supported by a workflow language. Typical examples of data streaming for scientific applications include scientific simulations, sensor network analysis, etc. These applications generate huge amounts of data, which must be streamed efficiently and effectively: the streaming services (Fig. 2) must meet an application's quality of service (QoS) and should guarantee that no data are lost.

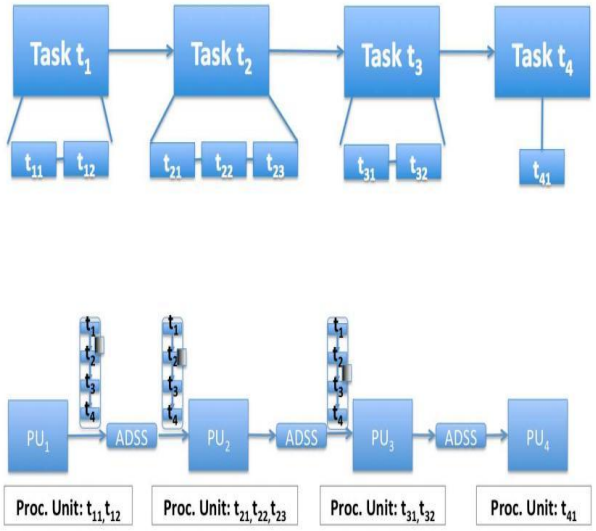


Fig 3: Autonomic Data Streaming Service (ADSS) unit

Autonomic data streaming service (ADSS) (Fig. 3) with in-transit processing into a workflow specification is explained here. Runtime resource allocation mainly depends on environmental conditions and it changes for different conditions of the same workflow. In the proposed work, the number of constraints is limited to the resource allocation. One can also implement a streaming service for utilizing a timed Reference net simulation for predicting future states of the streaming service. There are two main advantages of using streamed services: the Reference net which implements the ADSS and the timed model are coincident, and the second one is that the token distribution obtained from the Petri net implementation can be utilized to better understand demand for particular types of resources in the system.

10. EXPERIMENTAL RESULTS

There are two ways to choose these sample workflows (Fig. 4):

- Use a random DAG generator to create a variety of workflows with different characteristics such as number of tasks, depth, width, length etc.
- Use some scientific instruments or materials to produce workflows.

Workflows from other diverse scientific applications, which are: Montage: astronomy, CyberShake: earthquake science, Epigenomics: biology, LIGO: gravitational physics, SIPHT: biology.

Here the input from CMS which is Compact Muon Solenoid. The goal of CMS experiment is to investigate a wide range of physics, including the search for the extra dimensions and particles that could make up dark matter.

Fig. 4 shows the sample workflow and its coast assigned to reach the final destination. The goal is to minimize the cost of workflow execution here.

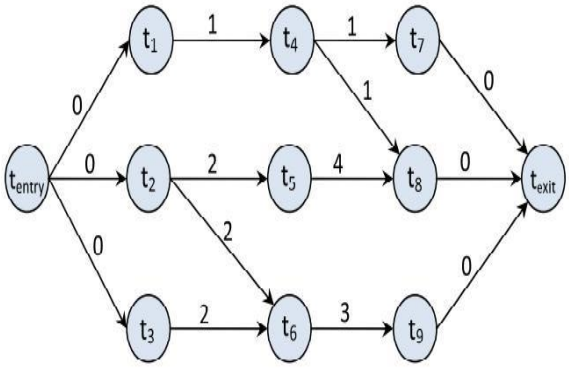


Fig 4: A sample workflow

The results of the simulation are shown in the output as table. Grid simulator called Optor sim is used. Optor sim supports various types of replication and optimization mechanisms which prevent the limitations of bandwidth, QoS etc.., Parameters like number of jobs, access method, and number of optimizers to be used entire details in the grid parameter setup.

Compact Muon Solenoid is used for carryout large scientific experiments, which produce large number of workflows. Hence these workflows needs to be processed simultaneously without any interruption and the resources must be equally allocated.

The results are shown along with their respective graph such that the values are inbuilt within it. The outputs shown are Figure 5, 6, 7 are Grid parameter setup, Grid statistics, Site Statistics

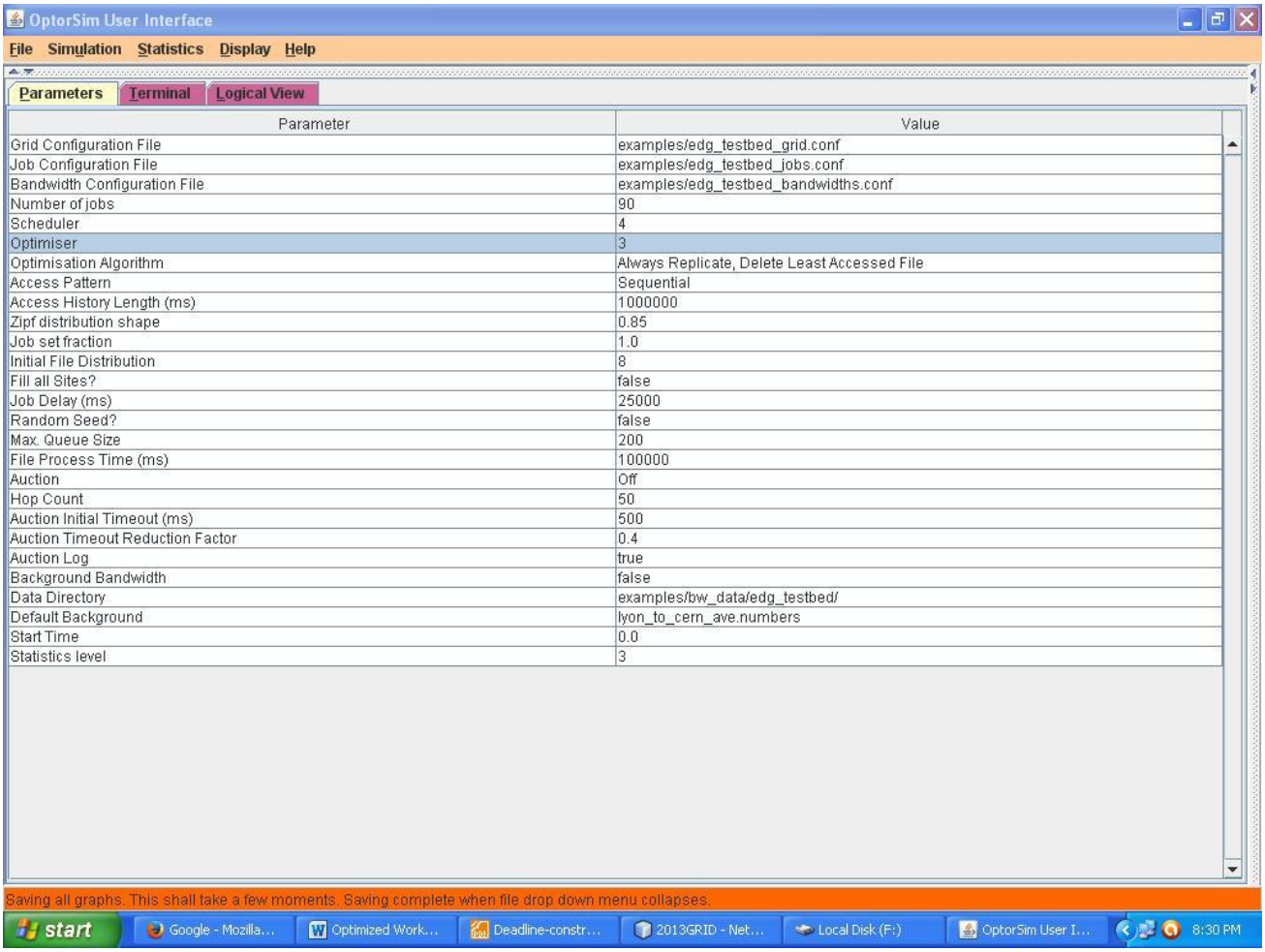
10.1 Complexity Analysis

The results from combined mechanism of scheduling, replica optimization and pipelining gives the polynomial time. The complexity for replica optimization is given as $O(l^m)$ time. For more efficiency the pipelining mechanism is considered to have $O(l)$ complexity.

Here the input is obtained from CMS, which produces physical job. Similarly, one can implement their proposed system for jobs from different fields such as biology, earthquake science and so on.

For different types of input, the configuration of the site is checked each time and the job is allocated according to the resource availability of the site.

Biological jobs encountered about $O\ (log\ n)$ time to execute the job. The time complexity mainly depends upon the size of the input workflow.



Parameter	Value
Grid Configuration File	examples/edg_testbed_grid.conf
Job Configuration File	examples/edg_testbed_jobs.conf
Bandwidth Configuration File	examples/edg_testbed_bandwidths.conf
Number of jobs	90
Scheduler	4
Optimiser	3
Optimisation Algorithm	Always Replicate, Delete Least Accessed File
Access Pattern	Sequential
Access History Length (ms)	1000000
Zipf distribution shape	0.85
Job set fraction	1.0
Initial File Distribution	8
Fill all Sites?	false
Job Delay (ms)	25000
Random Seed?	false
Max. Queue Size	200
File Process Time (ms)	100000
Auction	Off
Hop Count	50
Auction Initial Timeout (ms)	500
Auction Timeout Reduction Factor	0.4
Auction Log	true
Background Bandwidth	false
Data Directory	examples/bw_data/edg_testbed/
Default Background	lyon_to_cern_ave.numbers
Start Time	0.0
Statistics level	3

Fig 5: Grid parameter initial setup

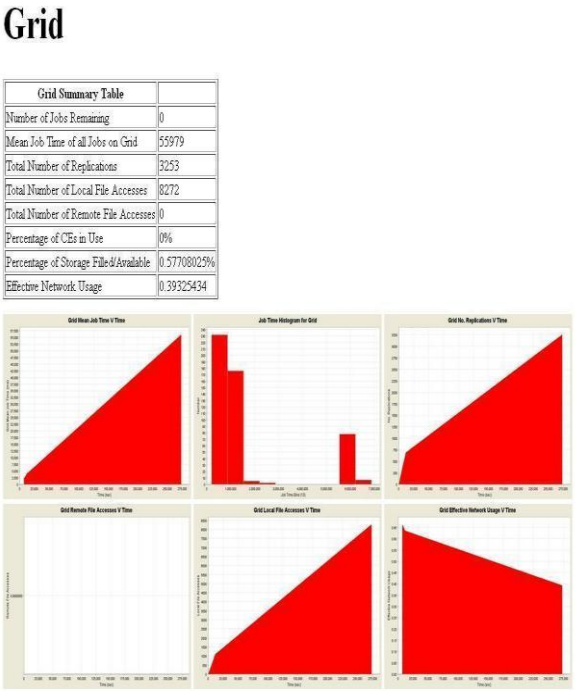


Fig 6: Grid statistics

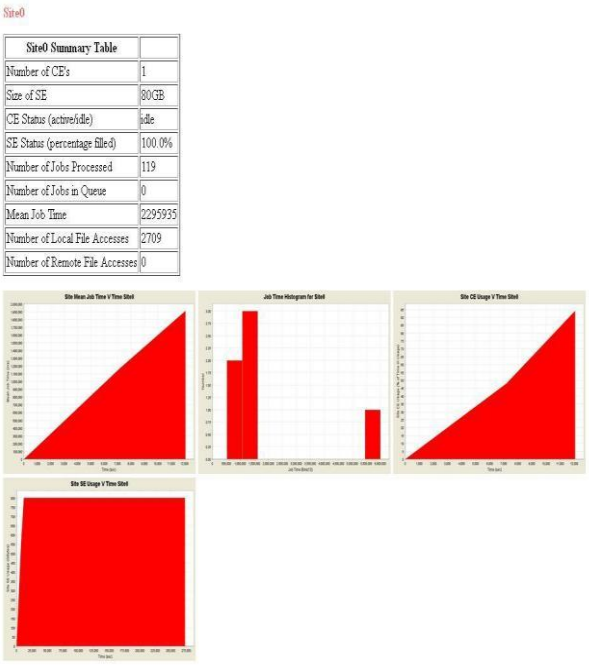


Fig 7: Site Statistics

11. CONCLUSION

Grid computing is one of the most promising and rapidly emerging trends in processing large number of workflows. While processing such large data, one has to be familiar with the problems that may arise during processing. A combined mechanism of scheduling, replica optimization and pipelining of workflows is used to minimize the cost of executing the workflows and to make the grid environment to work in an optimized way. Here the configuration of each site and then allocate the job according to it. Hence, conditions such as deadlock, critical path conditions can be avoided. The main objective of this paper includes achieving QoS, SLA satisfaction, deadline constraint service; replications are optimized to make the computation much more efficient, pipelining is done to reduce the time to compute a job, faster and optimized results, and economic way of scheduling.

In the future, this proposed model can be used in cloud computing along with pipelining mechanism to achieve better system performance while performing utility jobs,

12. REFERENCES

[1] European Strategies towards Next Generation Grids by D. Laforenza.

[2] Market-oriented Grids and Utility Computing by J. Broberg, S. Venugopal, and R. Buyya.

[3] Scheduling Scientific Workflow Applications Using Genetic Algorithms by J. Yu and R. Buyya.

[4] Pegasus: A Framework for Mapping Complex Scientific Workflows Onto Distributed System by E. Deelman et al.

[5] Scheduling of Scientific Workflows in the Askalon Grid Environment by M. Wieczorek, R. Prodan, and T. Fahringer.

[6] New Grid Scheduling and Rescheduling Methods by F. Berman et al.

[7] A Taxonomy of Workflow Management Systems for Grid Computing by J. Yu and R. Buyya.

[8] Computers and Intractability: A Guide to the Theory of NP-Completeness.W.H. Freeman by M.R. Garey.

[9] Static Scheduling Algorithms for Allocating Directed Task Graphs to Multiprocessors by Y.K. Kwok and I. Ahmad.

[10] Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing by H. Topcuoglu, S. Hariri and M. Wu.

[11] Improving Scheduling of Tasks in a Heterogeneous Environment by R. Bajaj and D.P. Agrawal.

[12] A High Performance Algorithm for Static Task Scheduling in Heterogeneous Distributed Computing Systems by M.I. Daoud and N. Kharma.

[13] Towards a General Model of the Multi-Criteria Workflow Scheduling on the Grid by M. Wieczorek, A. Hoheisel and R. Prodan.

[14] Multi-Objective Planning for Workflow Execution on Grids by J. Yu, M. Kirley and R. Buyya.

[15] QoS Support for Time-Critical Grid Workflow Applications by Brandic, S. Benkner, G. Engelbrecht and R. Schmidt.

[16] Cost-Based Scheduling of Scientific Workflow Applications on Utility Grids by J. Yu, R. Buyya and C.K. Tham.

[17] Scheduling Workflows with Budget Constraints by R. Sakellariou, H. Zhao, E. Tsiakkouri.

[18] Bi-Criteria Scheduling of Scientific Grid Workflows by R. Prodan and M. Wieczore.

[19] Multiobjective Differential Evolution for Scheduling Workflow Applications on Global Grids by A.K.M.K.A. Talukder, R. Buyya and M. Kirley.

[20] An Ant Colony Optimization Approach to Grid Workflow Scheduling Problem with Various QoS Requirements by W.N. Chen and J. Zhang.

[21] Deadline Division-Based Heuristic for Cost Optimization in Workflow Scheduling by Y. Yuan, X. Li, Q. Wang and X. Zhu.