AN ADAPTIVE META-SCHEDULER FOR DATA-INTENSIVE APPLICATIONS IN GRID ENVIRONMENT

Vimala.S[1], Sasikala.T[2]

[1]Anna University, Chennai.
[2]Principal, SRR Engineering College, Chennai.
Email: [1]vimala_aishwarya@yahoo.com, [2]sasi_madhu2k2@yahoo.co.in

Abstract

Grid computing becomes the de facto platform for scientific computations that incorporates geographically and organizationally dispersed, heterogeneous resources. These scientific and data intensive computations require large and multiple datasets to be transferred to the data storage and compute nodes. As there is rapid growth in communication through internet, the data transfer becomes the major bottleneck for the end-to-end performance for these scientific applications. A most practical way of increasing the throughput is using multiple parallel streams. Currently GridFTP protocol is designed for point-to-single point parallel data transfer. However the issue of simultaneous -multiple files to multiple locations is not studied so far. In this paper, we design an optimized Meta-scheduler by which multiple files can be transferred simultaneously to the destined compute nodes. A LBLC scheduling algorithm is designed to transfer multiple files to multiple locations simultaneously. A greedy method is followed at every stage. The Optimized proposed model gives better results compared to the non-optimized data transfer.

Keywords Grid computing, GridFTP, Optimization, Parallel TCP streams, Prediction, Data-dictionary.

I. INTRODUCTION

Grid computing, most simply stated is distributed computing taken to the next evolutionary level. Scientific experiments involve geographically distributed heterogeneous resources such as computational resources, scientific instruments, databases and applications. These experiments require large amount of data in terms of tera bytes to be transferred over wide area network for their computations. Though network initiatives, such as Atlas project and Tera grid provide high speed network connectivity to their users, yet the promised speed is not achieved due to processor limitations, insufficient network bandwidth, TCP tuning and disk performance bottle neck [11].

Grid Computing environment becomes a reality, which provides a demand drivers, reliable, inexpensive power for these users. Globus Grid Forum is working on protocols and standards to realize the importance of extreme high performance related issues. The GGF recommends to have high speed parallel data transfer and to store data near to the compute nodes.

Currently, GridFTP is an accepted data transport protocol of Grid community. GridFTP has the following features of solving the problems of older TCP communication. First, Multiple TCP connections can be established in parallel. Second, TCP socket buffer size can be negotiated between GridFTP server and client. However GridFTP is designed for point-to-point reliable data transfer [3][8].

Scientific and data intensive applications require multiple files to be transferred to multiple locations simultaneously for their valuable computations. The current GridFTP, though it supports parallel TCP connections, still higher speed is required to meet the demands of transferring large and multiple datasets [1]. Therefore, A Adaptive Meta-Scheduler is designed which receives multiple files from the users simultaneously and which does optimized parallel data transfer to the correct and intended compute nodes.

The architecture contains a Scheduler/Optimizer and GridFTP servers. The Meta-Scheduler opens multiple sockets to receive files simultaneously from the users. It then uses Location Based Least Cost (LBLC) scheduling algorithm which distributes files to GridFTP servers with fewer active connections relative to their destination IP [9][11]. A mathematical model is used to predict the number of parallel streams (N) the data to be striped at each GridFTP server. The GridFTP server establishes multiple (N) TCP connections in parallel. Since 'N' TCP connections are established, the throughput is N times larger than single TCP connection [3]. Here, the user is intended to give only the name of the executable, source address and

number of files to be transferred. This optimization technique is believed to be unique in terms of multiple-parallel file transfer.

In section II, we impart the related work regarding parallel data transfer, scheduling and GridFTP. In section III we discuss the design of our system. Section IV, shows the experimental results. Finally, in section V, we discuss the conclusion of our study.

II. RELATED WORK

There is limited number of studies that try to find out optimal number of TCP streams that are required for data transfer. Hacker[3] discuss how multistream TCP connections can improves aggregate TCP bandwidth. He also address the question how to select maximum number of sockets needed to maximize TCP throughput and demonstrates how packet loss rate and bandwidth affects Maximum Segment Size(MSS). In paper [4] PSockets were designed to exploit network stripping. The data is partitioned across several open sockets. Experiments were done using PSockets over Abilene network. C++ library were developed to incorporate network stripping. H.Sivakumar uses 12 TCP connections to improve the performance from 10Mb/sec to 75Mb/sec. Another study [5] on improving performance in High speed networks, uses sender side congestion control algorithm.

The paper[6], the mathematical equation is developed BW_n=MSS/RTT_n n/P_n C_i 2b/3 where RTT is the round trip time and P_n Packet loss, C_i is a constant range (0,1). b is the number of packets that are acknowledged by a received message. The effect of cross traffic is considered as a challenging factor when trying to estimate the throughput. The paper [3] addresses how to determine the number of TCP connections needed to maximize throughput while avoiding network congestion. Packet loss is taken as major issue in this. Hacker [3] claims that even when the packet loss increases due to factors like lockout, congestion or convergence, the overall throughput is greater than single TCP connection.

In this paper[2], the optimal parameters in terms of number of TCP connections and TCP socket buffer size is investigated. Takeshi_Ito discuss about GridFTP protocol which Globus Grid Forum is trying to standardize for parallel data transfer. He also discuss how socket buffer size can be configured by client and server. This paper [7] presents Fast Parallel file replication. Raut_Izmailov discuss about point-to-multipoint data Replication at multiple sites. He designs various tree structures and demonstrates how user data can be replicated at multiple sites.

In this paper[8] an algorithm is designed for economy-based scheduling of distributed data-Intensive applications on data grid. The model takes into account the cost and times for transferring datasets required for a job from different data hosts to the compute resource on which the job will be executed and for its processing. The algorithm is designed to minimize the cost or time depending on the user's preferences.

Srikumar[9] proposes a Grid Broker that mediates access to distributed resources. The broker supports a declarative and dynamic parametric programming model for creating grid applications.

In this paper [1], Dengpon designed a service that predicts the optimal number of parallel TCP streams and a provision of estimated time of throughput for a specific data transfer. He used stork Data scheduler to improve the performance of data transfer jobs submitted to it. There are also other services and tools that try to give an estimate for number of TCP connections required for maximum throughput. In our approach, we propose to transfer multiple Files simultaneously from a single source to single or multiple destination based on destination IP and cost.

III. PROPOSED MODEL

In this section, a model is proposed which improves the data transfer time based on referred models. Fig 1. shows the model of a Grid environment that consists of computational resources, GridFTP servers, Optimizer and scheduler. The Grid users transfer large and multiple jobs to the computing resources for their effective computations. Discovering and allocating resources is of more crucial in Grid Environment. Schedulers play a major role in partitioning of jobs, parallel execution of jobs, allocating resources and for service-level agreements.

Consider a model for scheduling independent jobs on grid. Each job requires a set $Fj = \{ fj1, fj2, \ldots fjK \}$ of K datasets. The datasets can be files. The overall time taken to execute the jobs is the sum of the execution time and the time taken to transfer each of the K files from the respective user or

storage nodes to the compute nodes. The computation time is denoted by tck and the transfer time for the kth dataset fjk is denoted by ttjk, then the total time required for executing the job *j*, $tj = tck + ttjk$ where $ttjk =$ Response time(djk) + Size(fjk)/ BW(Linkd**jk**). BW(Linkd**jk**) is the available bandwidth for the network connection between the data host djk and the compute resource *r*. The economic cost for executing the job *j*, *ej* is given by $ej = ejr + efjkr$ where $efjkr =$ Access cost(djk) + Size(fjk) * Cost per unit size(Linkd**jk**r).[9] where ejr is the processing cost of the job *j* on the compute node r and cost of transferring the dataset *fj* by *efjkr*. Thus minimizing data transfer time, minimizes the total execution time thereby minimizing the total economic cost.

In this paper, the Meta-Scheduler schedules multiple files simultaneously to the compute nodes. This model minimizes the data-transfer time thereby minimizing the overall execution time. The Meta-Scheduler is designed using the network modeling techniques proposed in [1],[2],[8],[9].

The design presents three scenarios. Locating the compute node, Prediction of Number of TCP streams and Scheduling.

The Optimizer gathers information about the available compute resources through a resource information service. This information is stored in the data directory. The scheduler requests for name of the executable resource, source address and number of files to be transferred from the user. It then receives multiple files simultaneously by opening multiple sockets. The Scheduler, scans the data for the compute resources suitable for the job in the data dictionary and decides on where to submit the job based on the availability, cost of the compute resource, the location, access and transfer costs of the data required for the job. The resources nearest to the source with minimum execution time is selected as the destined compute node.

Table I shows the sample data of the data dictionary. The actual information about locations of compute nodes, the execution time and utilization cost are levied by the service provider. To reduce the complexity of mapping the jobs to the compute nodes, the data in the data dictionary are sorted in ascending order based on least cost and location.

The Optimizer predicts the number of streams the data to be striped at GridFTP server. The Optimizer transfers different sampling size of data for each file. A minimum of three sampling of data transfer is tried

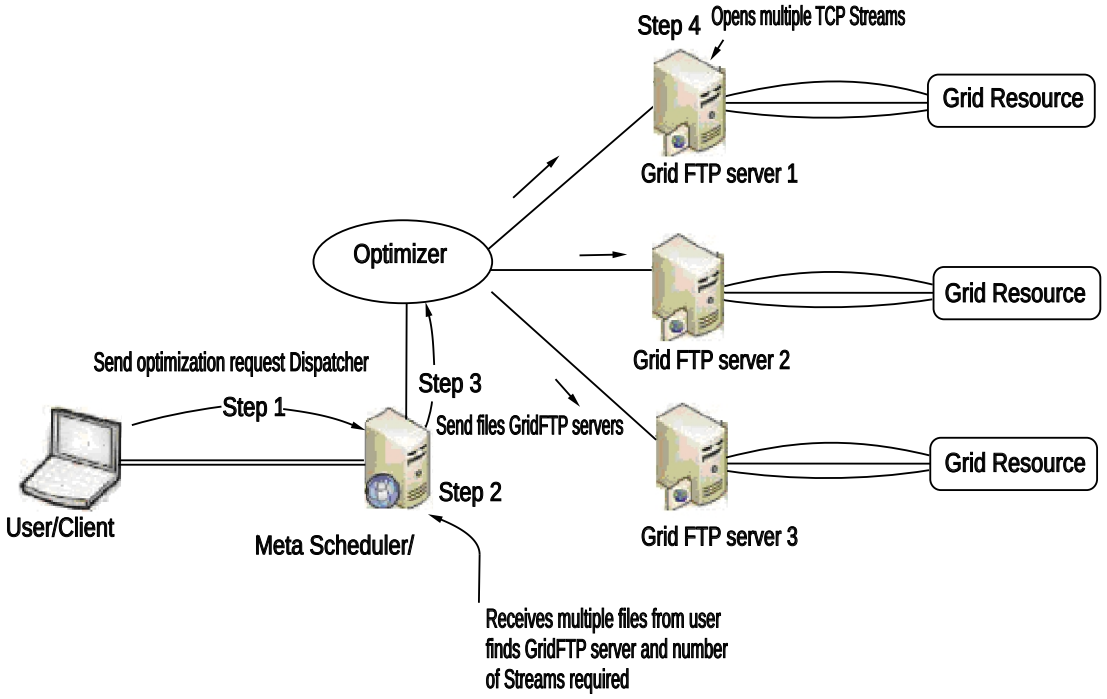

Fig. 1. Architecture of the Optimization Service.

for correct selection of parallelism levels. It measures the throughput for every data transfer. The mathematical equation

$$Th \leq (MSS/RTT)^{*}(c/\sqrt{p}) \quad ...(2)$$

TABLE I

Resource	Location in Terms of (ms)	Execution Time (ms)	Execution Cost ($)	GridFTP Server	Threshold (ms)
1	4100	80	31198	1	48
2	6800	54	76054	2	33
3	8100	120	500,000	3	72

is used for calculating the number of parallel streams. Where (Th) is the throughput. The MSS is Maximum transmission unit (MTU) size minus the TCP header size. The RTT is the time taken for the segment to reach the receiver and time taken by the acknowledgment packet to return to the sender. The packet loss rate (p) is the ratio of missing packets over total number of packets, c is a constant and n is the number of parallel streams.

The number of parallel streams for every iteration of sampling is doubled and corresponding throughput is observed. The sampling is stopped when the throughput results constant even if the number of streams increases. The number of streams at which throughput becomes constant is predicted as the required number of streams the data to be striped at GridFTP server. After locating the node and predicting the number of TCP streams, the job is dispatched to selected compute resource.

However, the compute node may get overloaded if all the files are submitted to the same optimized resource. Therefore, a threshold is fixed. If the waiting time for the job to be submitted is greater than the threshold the next compute node in the data dictionary is selected as the compute node.

The compute node thus uses the dataset received from the user and does its computational task. After completion of the tasks, the results are sent back to the scheduler host. The Location Based Least Cost (LBLC) scheduling Algorithm is shown in Algorithm I.

Algorithm 1: LBLC Scheduling Algorithm

1 Get source address && name of executable &&number of files to be transferred
2 Create n sockets
3 Bind the sockets to empty ports
4 **While n>0 do**
5 process← fork()
6 Listen to the port
7 if file
8 Accept connection
9 receive file
10 CALL select node
11 CALL optimized parameter streams
12 Strip files by n
13 Transfer file
14 receive results
15 Close connection
16 **endwhile**
17 **Select node :**
18 **while nearest && least_cost || nearest && least_execution_time**
19 **if (waiting_time > threshold)**
20 goto selectnode
21 **else**
22 return selected node
23 **until**
24 **optimized parameter streams:**
25 threshold ← α
26 Stream No1 ←1
27 **Calculate throughput1**
28 repeat
29 StreamNo2 ← 2* StreamNo1
30 Calculate throughput2
31 StreamNo1 ← StreamNo2
32 throughput1 ← throughput2

33 until throughput2>= throughput1

34 n= StreamNo2

35 return n

IV. EXPERIMENTS AND RESULTS

The experiments were based on the architecture shown in Fig.1.The scheduler requests for name of the executable resource, source address and number of files to be transferred from the user. It then, opens multiple sockets equal to that of number of files. The scheduler then, scans the data dictionary and maps the file to the best suitable compute node based on Location and Least cost. The Location (distance to the compute node) is given in terms of millisec. Cost for execution in terms of dollars ($) and time for transferring data over network links between the compute resources and the user in millisec. Two constraints are considered during job scheduling. The nearest resource with the least cost or nearest resource with least execution time. Based on users need the resources are selected from the data directory.

The optimizer sends sample data to the selected computed node. A minimum of three sample size of data is sent and the corresponding throughput and time is measured. The number of TCP streams is doubled for each iteration of sample data. The RTT found to be stable from 100MB. Therefore the sampling is stopped. The transfer time and throughput is found to be better at sample size 100MB and TCP stream 4. The file is therefore stripped by four at GridFTP server and transferred to the selected node. The same procedure is followed for all files received.

Another criteria is, the compute node may get overloaded if all the files are submitted to the same

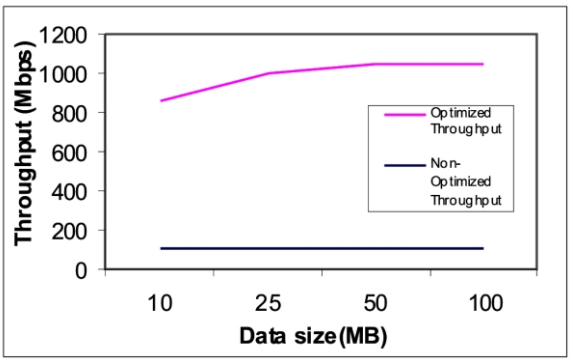

Fig. 2 Sample size vs Throughput

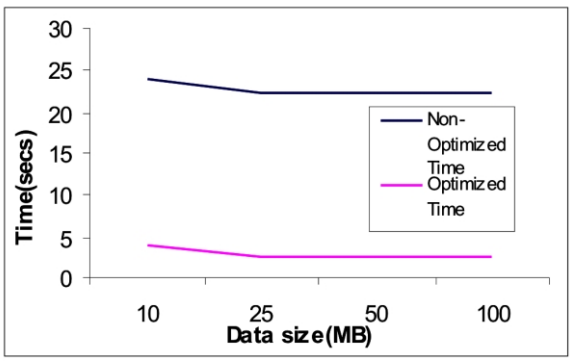

Fig. 3 Sample size vs Time

optimized resource. Therefore, a threshold is fixed. If the waiting time of the file to be submitted is greater than the threshold the next compute node in the data dictionary is selected as the compute node. Fig 2 shows the sample size versus Throughput.

Fig 3. shows sample size versus Time. The time for the corresponding sample size is measured. These graphical figures compares throughput and time for optimized and non-optimized transfer.

For non-optimized transfer , a single TCP stream is used for transferring multiple files. The various sample data as taken for optimized data transfer is transferred and the throughput and time is measured. Fig 2 shows the throughput comparison for optimized

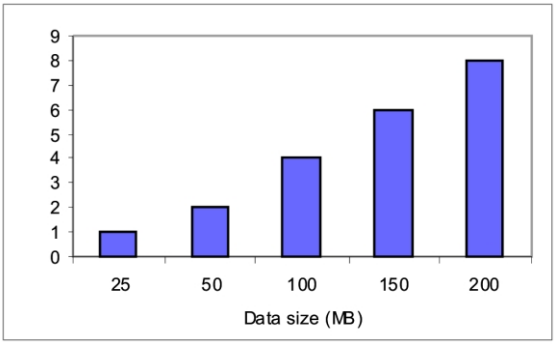

Fig. 4 Sample size vs Number of TCP streams

and non-optimized transfer.

Fig 4. shows the number of streams for the respective sample size. The experimental results shows that optimized throughput increases almost 9 times to that of non-optimized technique. From the experimental results the execution time and throughput of our

optimized technique is found to be better compared to non-optimized technique.

V. CONCLUSION

We design an Adaptive Meta-Scheduler. This scheduler does simultaneous multiple file transfer, selection of best compute resource in terms of execution time as-well-as utilization cost., stripping files into multiple streams. The scheduler uses LBLC algorithm for locating the compute node. Multiple files are received simultaneously from the user and based on the location, execution time, cost for execution, the files are assigned to the Grid FTP servers. A mathematical equation is used for predicting the number of parallel TCP streams the file to be striped at the Grid FTP server for achieving optimized throughput. A greedy strategy is followed at every stage. This Optimized Meta-Scheduler decreases total transfer time required for large number of data or file transfer submitted to it significantly compared to the non-optimized transfer. We plan to extent an optimized algorithm in Mobile-Grid Environment.

REFERENCES

[1] Dengpan Yin,Esma Yildirim, Sivakumar Kulasekaran, Brandon Ross and Tevfik kosar, "A Data throughput Prediction & Optimization service for widely Distributed Many-Task Computing". Proc. IEEE Transactions on Parallel and Distributed Systems, Vol 22. No 8, June 2011.

[2] Esma Yildirim, Dengpan Yin, Tevfik Kosar, "Prediction of Optimal Parallelism Level in Wide Area Data Transfers" IEEE Transactions on Parallel and Distributed Systems , VOL. XX, NO. XX, 2010

[3] Takeshi Ito,Hiroyuki Ohsaki, and Makoto Imase, "On Parameter Tuning of Data Transfer Protocol GridFTP for Wide-Area Networks". International Journal of Electrical and Electronics Engineering 2:8 2008.

[4] Thomas J.Hacker,Brain D.Athey, Brain Noble, "The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network, Proc. IEEE 2002

[5] H.Sivakumar,S.Bailey, R.L.Grossman, "PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks" Proc. IEEE 2000

[6] Tom Kelly, "Scalable TCP:Improving Performance in Highspeed Wide Area Networks" Laboratory for Communication Engineering, Cambridge University Engineering Department, Cambridge, United Kingdom. ACM SIGCOMM Computer Communication. Volume 33 Issue 2, April 2003

[7] Dong Lu, Yi Qiao, Peter A.Dinda, Fabian E.Bustamante, "Modeling and Taming Parallel TCP on the Wide Area Network" National Science Foundation. Proceedings of the 19th IEEE International Parallel and Distributed Processing, 2005.

[8] Rauf Izmailov,Samrat Ganguly,Nan Tu, "Fast Parallel File Replication in Data Grid" , NEC Laboratories America, Princeton, USA. 2004.

[9] Srikumar Venugopal and Rajkumar Buyya, "An Economy-based Algorithm for Scheduling Data-Intensive Applications on Global Grids", Department of Computer Science and Software Engineering, The University of Melbourne, Australia. 2004

[10] Srikumar Venugopal and Rajkumar Buyya, "A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids", Department of Computer Science and Software Engineering, The University of Melbourne, Australia. 2nd International Workshop on Middleware in Grid Computing, October 18,2004.

[11] Ioan Raicu, Ian T.Foster, Yong Zhao, "Many-Task Computing for Grids and Supercomputers", Proc .IEEE Workshop Many-Task Computing on Grids and Supercomputers(MTAGS),2008.

[12] Srikumar Venugopal and Rajkumar Buyya, "Cost-based Scheduling for Data-Intensive Applications on Global Grids",Grid Computing and Distributed Systems (GRIDS) Laboratory Department of Computer Science and Software Engineering. The University of Melbourne, Australia.. Sep 2011.