ISSN: 0975-8585

Research Journal of Pharmaceutical, Biological and Chemical Sciences

Efficient and dynamic Parallel Job Scheduling for bioinformatics Data Management in Data Grid Environment.

K Ashok Kumar[1]*, and C Chandrasekar[2].

[1]Department of Computer Science Engineering, Sathyabama University, Tamil Nadu, India.
[2]Department of Computer Science, Periyar University, Tamil Nadu, India.

ABSTRACT

Recent years, the data storage and management are one of the foremost difficult issues within the grid computing infrastructures ans its environment because of the vast quantity of data are transmitted among the grid applications and also the operations are endlessly executed. Grid computing normally deals with infinite amount of data and the traditional computing contains dedicated servers for data duplication and data storage. To solve this problems raise in large data management and security threats, in this work suggest a new method for distributed data storage and parallel processing of data in grid computing environment. In our proposed study devising the concept of Central Grid Computing (CGC) is employed to manage both data storage and extraction. CGC divides all grid data server into multiple geographically disseminated domains to facilitate the locality and simplify the intra-domain data storage and management. Data stored in various localities are searched in parallel mode and securely accessed and allow to share the data between various grids by using cryptographic techniques. This research also proposes a database indexing algorithm for identifying and locating and extract the essential data from servers which in turn improves the latency involved in retrieving data from assorted geographical locations. From the overall finding of this research study also proved that our proposed frame work may perform better in the handling of large data storage and access than other existing frame work used in grid computing environments.

Keywords: Grid computing, Data management, Parallel Processing, S-IDEA algorithm, NB-Tree.

Corresponding author

INTRODUCTION

Grid computing is a one of the most efficient technique used in emerging types of networks, servers, storage systems, etc. and these networks, CPU cycles can be well used by combining pools of servers, number of storage systems and networks are formed into single huge virtual system for sharing the data dynamically at run time. These kinds of systems can be distributed to the globe and the grid is said to be heterogeneous, which contains some persona, supercomputers and mainframes computers. The grid consists of huge number of files, non-interactive tasks and in turn it can be viewed as distributed system. Grid computing distinguished from conventional systems of high computing performance is as it collectively computes and nodes are loosely joined together, geographically dispersed and heterogeneous [1]. A grid is utilized for a various purposes, only one grid can be committed to a specific application and usually grids are frequently developed with the purpose of middleware software libraries. Grid computing is an imminent technology and it is being used for various type of business can gain more by utilize by information technology resources. The grid computing is a flexible, safe, synchronized resource sharing between various collections of resources, individuals and institutions. Grid computing provides the virtualization of scattered computing resources such as storage capacity, network bandwidth, processing, and to build a single system image, enabling applications and users to uninterrupted access to huge IT resources. The seamless access to a network of worldwide distributed resources such as CPU cycles, input and output devices, storage capacity, whole applications, services and more conceptual elements like certificates and licenses is the major advantage of Grid computing [2].

Computationally intensive problem can be resolved by various techniques and it can be categorized into many tasks and it circulated over remote and local machines and finally results are grouped together from individual systems using grid approach [Foster et al 2010]. All individual systems are usually various software setup and few of the destination nodes may have high performance server. So we need to find the appropriate way of fetch the large set of data such as genomic information, protein information, etc of living species. For this technique requires an efficient data transfer from server to client nodes with seamless way. However the size of data and accessibility population are keep on increasing and it looks hard to design a method or storage system to handle the large-size data. This research chiefly focus on designing a system for large size data handling in grid computing approach to resolve the various problems in sharing, storage and effective retrieval of data. In addition to this, parallel processing and indexing the database were also proposed in this study.

RELATED WORK

The issues of parallel grid computing have been focused by many researchers of grid computing domain. Charlotte proposed a new replication scheduler, a java oriented grid system for distributed and large scale resources in [4]. The scheduler can assign the different tasks to the resources in a random way and it also has the choice of continuously assigning tasks to the resources till any of its replicas is completed by grid resources. [4]. In addition to the parallel applications consists of independent tasks, a shared memory programming model is proposed in this work. It is fully depend on a memory management method called two stage idempotent executions planning to ensure that containing numerous replicas for a job doesn't breach the semantics of single task implementation[43].

These works focused on huge BoT applications consists of identical and independent tasks. They usually considered computation costs and adopted multi-port stable communication models and stable communication. In particularly Legrand et al. [20] and [19] developed centralized and decentralized schedulers. Offline and online solutions for scheduling simultaneous BoT applications are presented by Benoit et al [21]. As the aforementioned tasks are given theoretically huge solutions that depend on the impractical considerations, Casanova et al. [22] proposed a more practical application model and calculated non-clairvoyant heuristics based on abstract foundations. The above mentioned works builds a model to monitor the multiple practical dynamic Grid surroundings [37].

PROPOSED SYSTEM

The European Grid Infrastructure (EGI) [25], European Middleware Initiative (EMI) [26] and World Community Grid (WCG) [27] are real-world examples of grid computing environments. The main tasks of this Grid infrastructures is resource and job management, a word here utilized to jointly represent a set of procedures and problems related to the execution of tasks on Grid computing tasks. In order to manage their resources and jobs, the proposed system suggested a new framework called Central Grid Computing (CGC) that is split into two phases such as 1) Data Storage in Grid 2) Data Retrieval from Grid. The earlier is related to the administrator who owns/maintains large datasets such as protein database, nucleotide database and bio-medical datasets [44]. The later is related to the grid users who accesses the information stored in grid servers like Search of protein database, Search a nucleotide database. The Fig.1 shows the working paradigm of the proposed system. The administrator stores the details about users in grid storage servers via the central grid computing server. Consequently the users query the grid system to acquire their personal information from those grid servers.

Efficient method: To storage / locate the database

- Set id length of 13 bit id or more for database
- Set every two digits for locations and to identify the data base
- Use unique method (solution of problem identified) for find the required data base.
- When searching , use specific searching algorithm to make quick and search more easier (Indexing, Creating an NB-Tree).
- Here apply the parallel computation algorithm for process faster.

This can be run parallel in the entire existing database to retrieve the massive information"our system will retrieve information parallel from data sets and display the result quickly" [33] [34]

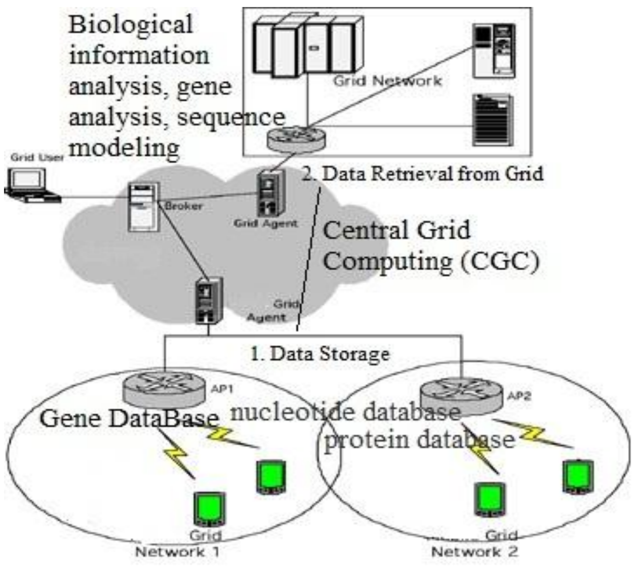


Figure 1: Working Paradigm of the Proposed System

Data Storage in Grid

Grid computing services provide well support for data storage based on various categories to be identified. Each and every service will be provided by only one system, or may constitute a collective set of tasks executed by multiple systems, by many vendors. Here we consider that our grid servers are linked through LAN and Internet technology.

Database Indexing

The database indexing algorithms used for high dimensional data points usually contains hard to implement algorithms and complex. This complexity may degrade the performance of the system. In this paper, a simple, but effective indexing method called NB-Tree (Norm B-Tree) is proposed for high dimensional data points of different dimension, using dimension reduction and improves the performance.

The NB-Tree structure uses a light mapping method for high dimensional data points of different dimension and its is computationally economical. The fundamental plan of NB-Tree is to use the Euclidean distance as the key value for high dimensional points. Thus resultant values obtained from dimension reduction can be well organized and later looked for results in one dimensional arrangement. Since B+ Tree uses Euclidean norm to cluster and sort the data points, it is the most efficient indexing technique which supports one dimensional structure and it is implemented by all the business a database. The mapping function maps data points from RD R1 (where RD is Relational Data and R1 is Relation function among the data) and ensures that the data points with near Euclidean values will be near in the plot. Hence, when processing a query and data, the system searches for the points (nodes) whose norm value is nearer to the norm of query point. In addition, in B+ Tree the leaves are connected as a linked list. Thus traversing elements in a B+-Tree is less time consuming operation.

Creating an NB-Tree

The creations of NB- Tree initially assume that the data space is standardized to the [0::1], a unit hypercube, and a random data point in the space is described as:

$$||P|| = \sqrt{P_O^2 + P_1^2 + \dots\dots\dots + P_{D-1}^2}$$

where P is the Pointer of space and D is Data point of data set or base.

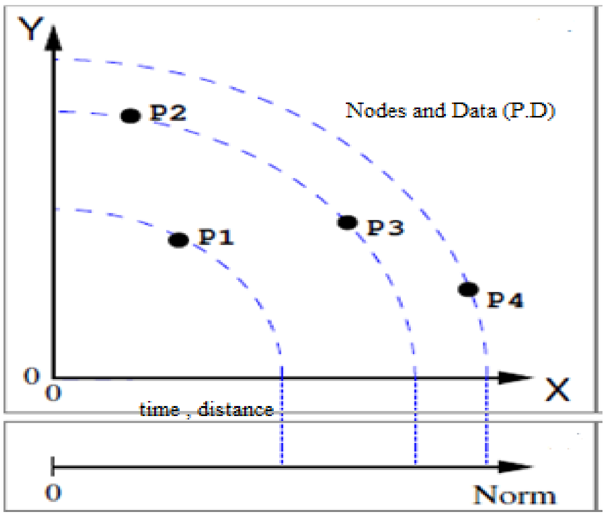


Figure 2: Dimension Reduction of node (p,d).

The norm is used as key when adding high dimensional data points in to a B+-Tree. After adding all the points, a set of one dimensional data sorted by norm value is obtained. Figure 2 illustrates a dimensional reduction for two-dimensional data points. Because this mapping function includes various points with the similar key value. Alternatively, it is said that neighbor data points have similar norms (keys).

Therefore, in a first step, two-dimensional points are charted to a one-dimensional line by calculating their Euclidean Norm. In a second step, to order these mapped data points through B+- Tree where all the consequent operations are performed. So the NB-Tree can be applied on current database without extra complication. Thus grid storage servers are well indexed utilizing the proposed method.

Searching

The process of searching in the NB-Tree initiates by calculating the query point norm. Now one Dimensional B+-tree can be searched against query points. The search procedures are based on the query type. Present indexing methods support three types of queries as follows: 1).Point Query. 2) Range Query. 3) k-NN Query (Here K-NN stand for k nearest neighbor). [28] [32].

This work uses Point Query since the administrator and user requires exactly matched records from the database. Range Query and k-NN Query are used in content-based retrieval.

```
1.leaf = searchB+Tree(dist(q))
2. Do
3. leaf = leaf.PreviousSearchToTheRight
4. upperLimit = upperLimit + delta
5. While(leaf.key <= upperLimit)
6. If(dist(leaf.point, q) < list.LastElement Or list.size < k)
7. list.Insert(leaf.point)
8. End If
9. leaf = leaf.right
10. End While
11. leaf = leaf.PreviousSearchToTheLeft
12. lowerLimit = lowerLimit + delta
13. While(leaf.key >= lowerLimit)
14. If(dist(leaf.point, q) < list.LastElement Or list.size < k)
15. list.Insert(leaf.point)
16. End If
17. leaf = leaf.left
18. End While
19. While(dist(list.LastElement(), q) > radius)
20. kNN = list[0, k-1]
```

Algorithm 1: Point Query (q,t_q)

PARALLEL JOB SCHEDULING AND DATA RETRIEVAL

Whenever the data are replicated, the copies of these data files are stored at various locations in the Data grid. The major theme is the efficient and quick access, which can be achieved by providing the data close to the user. A replication algorithm should have the solution for these questions, (1) which data files should be replicated; (2) where the replicas should be located in the system. (3) When and how many replicas should be produced.

When a user queries for files, all the servers should be searched parallel for getting fast response. Hence we purpose a novel Parallel Job Scheduling (PJS) that considers file locations, network characteristics, number of jobs waiting in queue, and disk read speed of storage drive at data sources. The central grid computing server takes the responsibility of implementing Parallel Job Scheduling strategy.

Parallel Job Scheduling (PJS)

For effective scheduling of jobs, the algorithm decides the suitable region, LAN and location correspondingly. The PJS reduce the effort of the user in probing for the suitable site by using the hierarchical tree to schedule a job.

Let $J_j = \{ f_1, f_2, . . . , f_m\}$ be the *m* number of files required files for a job *j*. For a Grid site *Sj*, based on the availability of *fi* in *Sj,* the replicas are classified into four subsets as follows.

• The first subset represents *On-site* set R^j_S in which every locally accessible replicas are present.
• The second subset is *LAN-site* set R^j_{LAN} in which the remaining replicas that can be seen in the local LAN are present.
• The third subset is *Region-site* set R^j_R in which the other replicas that should be contacted from local region are present.
• The fourth subset is Other-Region-site set R^j_{OR} in which the other replicas that should be contacted from other region are present.

Assume that for each f_i in R^{j}_{OR}, the bandwidth of Sj resides in B_{ij}. Now the time required to get f_i to S_j from $|f_i|/B_{ji}$, Here $|f_i|$ represents the replica size. Some cost terms are defined in following.

Inter-Region-Communication-Cost (InterRccj x): If job is distributed to region x , then; the inter-communications value are often computed by victimization the subsequent equation

$$\text{InterRcc : if } j \rightarrow x,\ \sum_{i=1..m}(\text{InterRcc}^{j}_{x}) \qquad (1)$$

Inter-LAN-Communication-Cost (InterLcc^{j}_{x}): If a job is sent to LAN x , then the inter-communications cost can be computed by using the factor LAN-to-LAN bandwidth

$$\text{InterLcc : if } j \rightarrow x,\ \sum_{i=1..m}(\text{InterLcc}^{j}_{x}) \qquad (2)$$

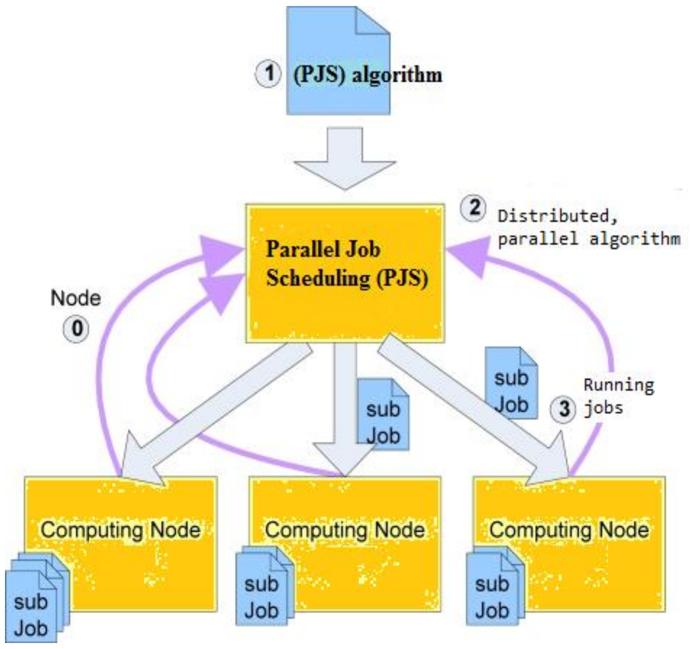


Figure 3: An example of hierarchical grid environment

The PJS consider the hierarchical Grid structure. Figure 3 is a simple example. The time required to retrieve a file from another site is drawn over each edge. A sample grid job needs six files (F1, F2, F3, F4, F5, F6) which are distributed in four different regions.

If a job is scheduled to the region on basis of highest hit percentage of required replicas like Region or node B, then the number of inter-region- communications and the execution time of the job would be reduced since the data can be accessed faster. It's possible that the number of replicas in one region can be greater than other but the total amount of replicas would be smaller. The job can't be scheduled based on the number of replicas. Moreover, hunting for the best site from a large amount of distributed resources may direct to extensive latency.

The PJS consists of three steps decision process. In first step, a best region for minimizing the inter-region communication cost is selected. For example in Region B, it has to retrieve File3 and File4. The Region C can be the best location for File3 and File4 which has minimum $\text{InterRCC}^{j}_{\text{RegionB}} = 8$ s. This indicates that the region with maximum number of matches of necessary replicas would not be the good solution. In second step, the best LAN among all the available LANs is selected to reduce the Inter-LAN-Communication-Cost (InterLCC^{j}_{x}). LANa is selected in previous example. [36]

Once the appropriate LAN is chosen, the third step is to choose the top site Sj from chosen LAN (in example LANa) on basis of two main parameters: queue length and estimated file transfer time. The overall completion time of the Data Grid jobs can be reduced by reducing file transfer time because the Data grid jobs require high file transfer time. The completion time can also be affected by the Queue length, i.e. the number of uncompleted jobs in the queue[38]. Using a parallel strategy the best site can be selected. The proposed parallel strategy is described in the following process. Some matrices in the scheduling are algorithm explained first: Let B_{ji} is the available bandwidth at the time from site S_j to the location that f_i exists in. $\text{PropagationDelay}_{ij}$ is network/ propagation latency /delay (in seconds) from location Sj to Si . DiskSpeed_i represents the data transfer rate of storage devices of systems in site i.. Then transfer time for file fi (TransferTimef_i) is calculated by

if ($B_{ji} < \text{DiskSpeed}_i$)

$$\text{TransferTime}_{fi}=\text{PropagationDelay}_{ij}+ (|f_i| * 8)/B_{ji} \qquad (3)$$

Else

$$\text{TransferTime}_{fi} = \text{PropagationDelay}_{ij} + |f_i|/\text{DiskSpeed}_i$$

Let $J_x = \{f_1,f_2, \ldots , f_m\}$ be the m number of required files for job x. Now estimated file data transfer time of job x when scheduled on location S_j ($\text{JobTime}_{x,j}$) is given by

$$\text{JobTime}_{x,j}= \textstyle\sum_{i=1}^{m} \text{Min}(\text{TransferTime}_i) \qquad (4)$$

When a number of sites store the replica of fi, then the one with minimum TransferTime is selected. Let k denotes the number of jobs waiting in queue of location Sj. The TotalTime_j for location S_j is computed by

$$\text{TotalTime}_j= \textstyle\sum_{x=1}^{k} \text{JobTime}_{x,j} \qquad (5)$$

At the end of job scheduling, the servers which contain the files required by the users are located and the connection is established. Now we have to search the database of the located server to get the required file data.

```
Input : $J_i$,…. $J_n$ represents n number of jobs and {R}, {L} denotes the set of regions and LANs respectively.
1. Find the availability of $J_i$ at given {R}
2. if available then
3.    calculate $\text{InterRcc}^j_x$ of job j for all the regions in {R}
4.    calculate $\text{InterLcc}^j_x$ of job j for all the LANs in {L}
5.    step 1: choose a region which has minimum $\text{InterRcc}^j_x$
6.    step 2: choose a LAN which has minimum $\text{InterLcc}^j_x$
7.    step 3: calculate $Q_L$ and $\text{TransferTime}_{fi}$ for each site $S_j$ in chosen LAN
8.    SELECT site with minimum $\text{TransferTime}_{fi}$
9.    return file
10.  repeat for all jobs
11.else
12.end
```

Algorithm 2: Parallel Job Scheduling

Searching NB-Tree

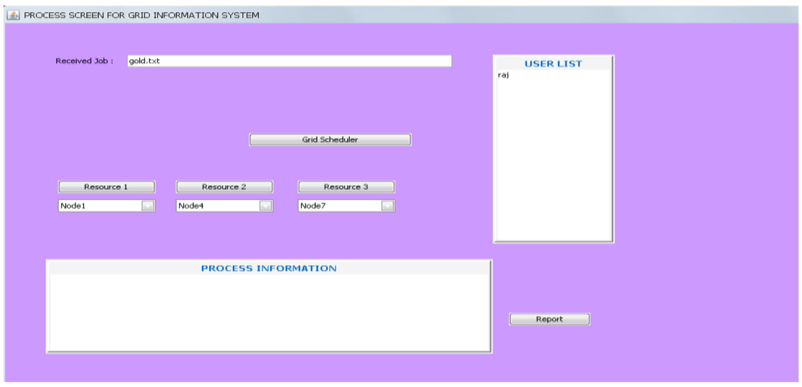
The resultant value obtained at the end of computing the query point Euclidean norm, is used as key to look the B+-Tree. The distance to the query point is computed for the point(s) returned by the B+-Tree (if any). If the resultant distance is zero, it denotes that the query point presents in the NB-Tree. The located server then sends the data to central grid server for further processing.

EXPERIMENTAL RESULTS AND DISCUSSION

In this section the experimental evaluation performed is described and performance of the proposed system is analyzed. All experiments were performed on a PC Pentium Dual Core @ 2.30GHz running Windows

XP with 384 MB of RAM and 15GB of disk. The proposed system is implemented using CoreJava, Globus Toolkit 4 and GridSim package.

The following snapshots show the major processes of proposed system such as secure grid storage, task scheduling and data retrieval. Figure 4 shows the information about grid infrastructure and input data. After getting the input data the submit button will be pressed which takes to the grid information system and to central grid server. Figure 5 shows the encryption process at central grid server. Figure 6 depicts the grid scheduling process where the grid scheduler allocates the task to different nodes parallel.

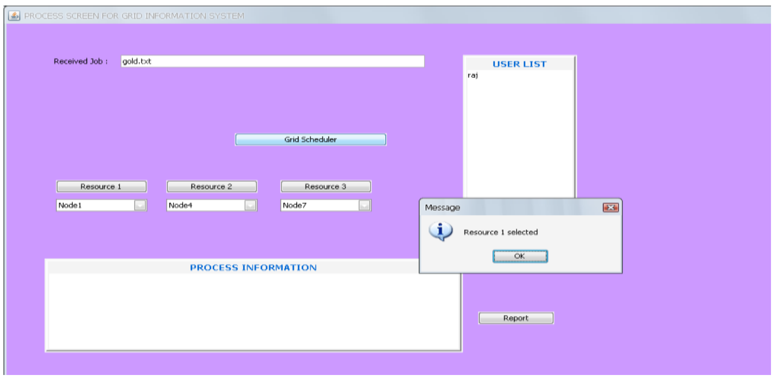

PROCESS SCREEN FOR GRID INFORMATION SYSTEM

Received Job : gold.txt

USER LIST: raj

Resource 1: Node1
Resource 2: Node4
Resource 3: Node7

Figure 4: Grid application with selected sample data.


PROCESS SCREEN FOR GRID INFORMATION SYSTEM

Received Job : gold.txt

USER LIST: raj

Resource 1: Node1
Resource 2: Node4
Resource 3: Node7

Message: Resource 1 selected

Figure 5: Parallel Task Scheduling

The output and benefits of the proposed system are examined in following. Number of Intercommunications and Computing Resource Usage are the considered parameters in this work.

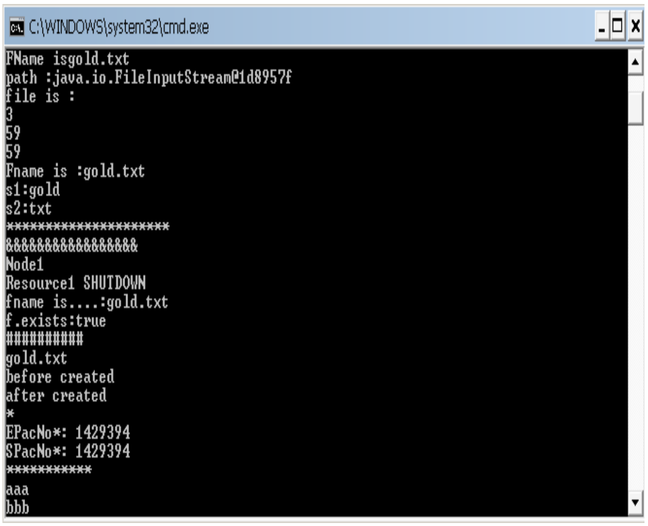
Table 1: Computer node details

Nodes (32)	CPU	Memory
10	Intel Pentium 4 @2.2Ghz	512mb
15	Core Duo @ 2.8 Ghz	2gb
7	Quad core @ 3 Ghz	2gb

Table 2: Resource Usage Table

Resources	Workload (tasks/minute)	CPU Usage (%)	Job Time (mins)	Method
12	365	75	30	Serial
28	1000	80	13.3	Static PS
32	2000	88	3.5	PJS(dynamic)

Table 1 and 2 shows the resource usage of grid servers simulated in the experiment. The CPU usage gets increased when the workload of grid environment is increased. It is observed that time required to complete the job depends on number of resources and CPU usage of grid servers.



```
C:\WINDOWS\system32\cmd.exe
FName isgold.txt
path :java.io.FileInputStream@1d8957f
file is :
3
59
59
Fname is :gold.txt
s1:gold
s2:txt
*********************
&&&&&&&&&&&&&&&&
Node1
Resource1 SHUTDOWN
fname is....:gold.txt
f.exists:true
##########
gold.txt
before created
after created
*
EPacNo*: 1429394
SPacNo*: 1429394
***********
aaa
bbb
```

Figure 6. Computing resource usage for various job scheduling and replication algorithms

The Figure 7 shows the computing resource usage. It represents the percentage of time in which Computing Elements are in live state. The PJS completes all jobs efficiently via proper computing resource usage thus CPUs would not be idle for longer time.

CONCLUSION

A grid computing environment that utilizes benefits of the Internet to give storage capability for the enormous data has become as an important research topic. To improve the overall performance of the grid environment, the job scheduling policy and indexed databases are very important factors because of the storage capacity of each Grid System is limited. Thus, this paper is proposed a CGC framework to store and maintains biomedical information, unique ID details of citizens in the grid environment. The central grid system proposed in this work play a major role as it receives tasks from administrators and users. To store and retrieve data from grid, this work proposed an efficient Parallel Job Scheduling (PJS) and NB Tree database indexing technique. The PJS uses the hierarchical scheduling to decrease the searching time for a suitable computing resource. From the experimental results it is found that the NB-Tree support point queries more. By using SIDEA, the data flow between central grid server and storage servers can be effectively protected. The proposed secure algorithm has two key points: improved degree of diffusion and increased key size.

The experimental results show that our proposed system achieves good performance in parallel job scheduling and searching the required data in grid servers. But due to the network bandwidth constraints and latency over the Internet, the true mathematical calculation over the Grid is still the limitations to this work. Future research direction involves in finding a task-level/coarse-grained distributed parallel programming interface for grid environment.

REFERENCES

[1] Ajay Kumar And Seema Bawa, Cornell University Lib,Jul, 2012.
[2] Menglan Hu And Bharadwaj Veeravalli, Ieee, October 2013
[3] Http://Www.Gridcafe.Org/What-Is-The-Grid.Html
[4] Foster, I, Kesselman, C, Nick, J.M., and Tuecke, S., Dec 2010.
[5] Diuf. H , May 18, 2009.
[6] C. Anglano, M. Canonico, Feb. 2005, 3470.
[7] Rosetta@Home, http://boinc.bakerlab.org/rosetta/, 2012.
[8] Einstein@Home, http://einstein.phys.uwm.edu/, 2012.
[9] D.P. Anderson and K. Reed, Int'l Conf. System Sciences, 2009.

[10] A. Grama, A. Gupta, G. Karypis, and V. Kumar, Addison Wesley, 2003.
[11] O.H. Ibarra and C.E. Kim, J. ACM, ,2007, vol. 24, no. 2, pp. 280-289.
[12] M. Maheswaran et al, 1999, pp. 30 44.
[13] H. Casanova, et al , May 2010, pp. 349-363.
[14] E. Santos-Neto et al , 2009, pp. 210-232.
[15] N. Fujimoto and K. Hagihara, 2008, pp. 391-398.
[16] W. Cirne, F. Brasileiro, D. Paranhos, L.F.W. Ges, andW. Voorsluys, J.Parallel Computing, Apr. 2006vol. 33, no. 3, pp. 213-234.
[17] Y.C. Lee and A.Y. Zomaya, IEEE Trans. Computers, June 2007vol. 56, no. 6, pp. 815-825,
[18] C. Anglano, J. Brevik, M. Canonico, D. Nurmi, and R. Wolski, Int'l Conf. Grid Computing, 2007.
[19] A. Legrand and C. Touati, Proc. IEEE INFOCOM, 2007.
[20] C. Anglano and M. Canonico, Proc. IEEE Int'l Symp. Parallel and Distributed Processing (IPDPS), 2008.
[21] A. Iosup, et al, (HPDC '08), 2008.
[22] O. Beaumont et al , Apr. 2008, vol. 19, no. 5, pp. 698-709.
[23] R. Bertin, A. Legrand, and C. Touati, 2008, pp. 118-125.
[24] A. Benoit et al , Feb. 2010, vol. 59, no. 2, pp. 202-217.
[25] http://www.egi.eu/
[26] http://www.eu-emi.eu/
[27] http://www.worldcommunitygrid.org/
[28] Erik Gast, Ard Oerlemans, Michael S. Lew, IJMIR, vol. 2, no. 4, 2013.
[29] Karol Estrada et al, Bioinformatics. Oct 15, 2009; 25(20): 2750–2752.
[30] K.Ashokkumar et al, Journal of Engineering Science and Technology Review 7 (4) (2014) 109- 113.
[31] Ming-Chi Tsai et al, 2007: 746–750.
[32] J.A. Jorge, (DASFAA 2003) Proceedings dasfaa-03, 2003.
[33] K.Ashokkumar, Dr.C.Chandrasekar, ICMS2014 Elsvier conference proceedings ,2014.
[34] K.Ashokkumar, Dr.C.Chandrasekar, proceedings of the National conference on man-machine interaction 2014.
[35] K.Ashok kumar, C.Chandra sekar, Proceedings of the INCOCCI- IEEE conference, 2010.
[36] Chang, R.S., Volume 23, Issue 7, August 2007, Pages 846–860
[37] Hu, Mengalan et al, IEEE Transactions on Computers, 2012.
[38] Sepahvand et al, Journal of Applied Sciences Reearch, 2011.
[39] Hsu, C.H, Future Generation Computer Systems, 2007 .05
[40] Dongbo Liu and Peng Xiao, Journal of Software (1796217X), 2013.
[41] Massimo Canonico, (WETICE 2007), 06/2007
[42] Dusit Niyato, IEEE/ACM International Symposium on Cluster Computing and the Grid, 05/2009
[43] www.creatis.insa-lyon.fr
[44] Andreas D. Baxevanis, ISBN 0-471-38390-2 (Cloth), ISBN 0-471-383910.