International Journal of Engineering Research-Online
A Peer Reviewed International Journal
Articles available online http://www.ijoer.in; editorijoer@gmail.com

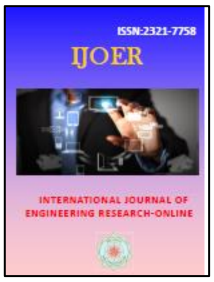
Vol.5., Issue.2, 2017
March-April

RESEARCH ARTICLE

ISSN: 2321-7758

DATA RETRIEVAL FROM DATA GRID WITH BLOOM FILTER IMPLEMENTATION

HARITHA S[1], NANDHINI R [2], NANDHINI D [3], SARANYA K[4]
[1,2,3]UG Student, Department of Computer Science and Engineering, Alpha College of Engineering, Chennai, T.N, India
[4]Assistant Professor, Department of Computer Science and Engineering, Alpha College of Engineering, Chennai, T.N, India
[1]harithashyam07@gmail.com, [2] nandhinirsnandhu@gmail.com, [3] nantinideena33@gmail.com
[4] sara.kar4@gmail.com



ABSTRACT
In the existing system, however significantly limits the usability of outsourced data due to the difficulty of searching over the data. It is a time consuming process. In the proposed system, every cluster comprises a number of nodes. Moreover, there is a master site that has all the files in the data grid. The storage of each cluster node is small therefore cannot accommodate all the files in the data grid. So files need to be brought from other nodes. The requested node checks if the closest node does not have the file, it searches the next closest node. Based on requested file popularity, master site replicate the file to cluster node otherwise clear files from the closest node based on Popular File Replicate Strategy (PFRF). In the modification part of the paper, we are adding Bloom Filter algorithm in order to capture the data in a short forms. This system will avoid the whole data storage in all the nodes same time every cluster node can maintain the Bloom filter index of the entire data stored. This process is very useful in order to fetch the data quickly.
Key Words— Data Grid , Bloom Filter , Bloom Filter Index, Replication , IPFRF.

INTRODUCTION

In the Data Grid, the users are distributed across a large geographical area. These users need access to a large volume of data which might be in a far node. This access consumes a large amount of time of bandwidth. Hence, data replication is needed to make more than one copy of the same file at different nodes which helps the user to fetch the file from its own storage or from the storage of a close node. As a result, both the consumed time and bandwidth will be reduced. Dividing the time into rounds usually leads to a better decision on which files to replicate because this decision is made after a large number of file requests and therefore the users will determine more accurately which files need to be kept in their storages. In this paper, a data replication strategy named Improved PFRF (IPFRF) is proposed. This strategy is based on PFRF but overcomes its drawbacks. The drawbacks are as follows:

First, this strategy does not determine to which cluster node the file is replicated. Therefore, a number of factors have to be used in that determination such as number of requests, free storage space, and node centrality. Second, this strategy only considers the number of requests to determine the file popularity (importance). However, there are other important factors to determine the popularity of a file such as how many times it was requested in the last round and the file size. Third, in PFRF, the average popularity for a file is defined as the sum of the popularities of the file only in the clusters having it divided by how many clusters having this file. However, some clusters might not have the file but have a high request rate for it. Therefore, the average popularity for a file is better to be defined as the sum of the popularities of the file in all clusters divided by the number of clusters in the Data Grid. Hence, also we are implementing the Bloom Filter index which will be present along with the Distributed Hash Table. The contents are retrieved and also using the stemming algorithm the words are at their base level.

BLOOM FILTER

A bloom filter is a data structure used to support membership queries.

```
Data: x is the object key to insert into the
bloom filter
Function: insert(x)
 for j : 1 . . . k do
 /* Loop all hash functions k */
i ← hj (x); if Bi == 0 then
 /* Bloom filter had zero bit at position i */
Bi ← 1;
end
end
```

Pseudo code for Bloom filter insertion

Properties:

- The amount of space needed to store the bloom filter is small compared to the amount of data belonging to the set being tested.
- The time needed to check whether an element is a member of a given set is independent of the number of elements contained in the set.
- False positives are possible, but their frequency can be controlled. In practice, it is a tradeoff between space/time efficiency and the false positive frequency.

A bloom filter is based on an array of m bits ($b_1, b_2, \ldots b_m$)that are initially set to 0.

0	0	0	0	0	0	0	0
b1	b2	b3	b4	b5	b6	b7	b8

There exists k independent hash functions each returning a value between 1and m are used. In order to “store” a given element into the bit array, each hash function must be applied to it and, based on the return value r of each function (r1, r2, ..., rk), the bit with the offset r is set to 1. Since there are k hash functions, upto k bits in the bit array are set to 1.

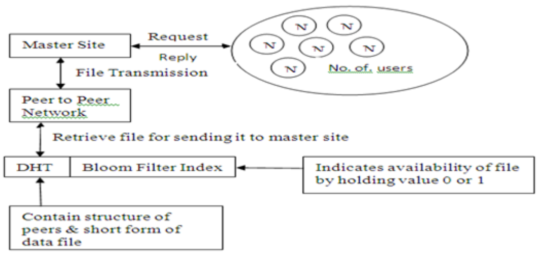

Master Site
Request
Reply
N
No. of. users
File Transmission
Peer to Peer Network
Retrieve file for sending it to master site
DHT
Bloom Filter Index
Indicates availability of file by holding value 0 or 1
Contain structure of peers & short form of data file

Fig. 1: Architecture Diagram

Related work

A. Data Replication and the Storage Capacity of Data Grids: Storage is undoubtedly one of the main resources in data grids, and planning the capacity of storage nodes is an important step in any data-grid design. This paper focuses on storage-capacity planning for data grids. We have developed a tool to calculate, for a specific scenario, the minimum capacity required for each storage node in a grid, and we have used this tool to show that different strategies used for data replication may lead to different storage requirements, affecting the storage-capacity planning.

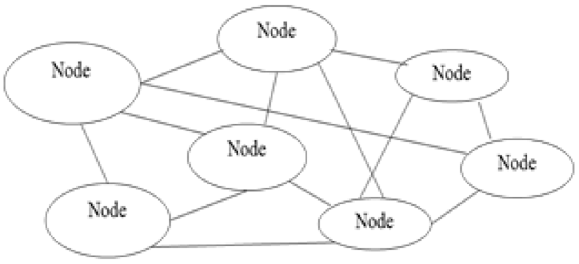
B. *Analysis of Scheduling and Replica Optimisation Strategies for Data Grids using OptorSim*: Many current international scientific projects are based on large scale applications that are both computationally complex and require the management of large amounts of distributed data. Grid computing is fast emerging as the solution to the problems posed by these applications. To evaluate the impact of resource optimisation algorithms, simulation of the Grid environment can be used to achieve important performance results before any algorithm is deployed on the Grid. In this paper, we study the effects of various job scheduling and data replication strategies and compare them in a variety of Grid scenarios using several performance metrics. We use the Grid simulator OptorSim, and base our simulations on a world-wide Grid testbed for data intensive high energy physics

experiments. Our results show that scheduling algorithms which take into account both the file access cost of jobs and the workload of computing resources are the most effective at optimising computing and storage resources as well as improving the job throughput. The results also show that, in most cases, the economy-based
replication strategies which we have develope improve the Grid performance under changing network loads.

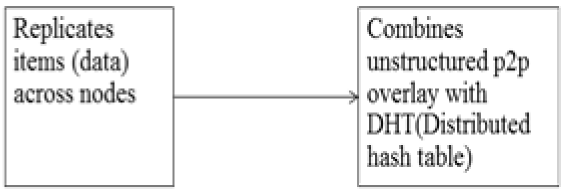
C. Data Replication Strategies in Grid Environments: Data Grids provide geographically distributed resources for large-scale data-intensive applications that generate large data sets. However, ensuring efficient and fast access to such huge and widely distributed data is hindered by the high latencies of the Internet. To address these problems we introduce a set of replication management services and protocols that offer high data availability, low bandwidth consumption, increased fault tolerance, and improved scalability of the overall system. The estimation of costs and gains is based on factors such as run-time accumulated read/write statistics, response time, bandwidth, and replica size. To address scalability, replicas are organized in a combination of hierarchical and flat topologies that represent propagation graphs that minimize inter replica communication costs. Our results prove that replication improves the performance of the data access on Data Grids, and that the gain increases with the size of the datasets used.

D. A Framework for Replication in Data Grid: Data replication is the creation and maintenance of multiple copies of the same data. Replication is used in Data Grid to enhance data availability and fault tolerance. In this paper, a framework for replication in the Data Grid is proposed. The framework is considered an environment that can be used to evaluate the performance of different replication strategies. An event-driven simulator written in Java is used to evaluate the performance of three replication strategies based on the new proposed framework. These strategies are No Replication, Plain Caching, and Fast Spread. The simulation results show that Plain Caching strategy achieved the best performance, while No Replication strategy achieved the worst performance in terms of number of requests served locally.

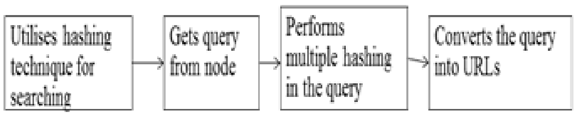
NETWORK CONSTRUCTION: Network has many numbers of node and their details. It maintains the connection details also. Nodes are interconnected and exchange data directly with each other nodes. Nodes are connecting with other nodes in the network. Network server maintains the node ip address, port details and status. Node give request to server and get the needed response from server.



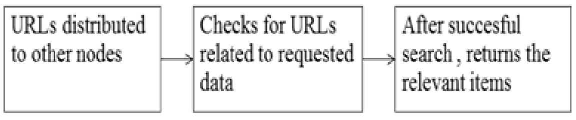
BLOOMCAST: In Unstructured P2P networks, Bloom Cast is an effective and efficient full text retrieval scheme. By leveraging a hybrid P2P protocol, Bloom Cast replicates the items uniformly at random across the P2P networks. Bloom Cast hybridizes a lightweight DHT with an unstructured P2P overlay to support random node sampling and network size estimation.


Replicates items (data) across nodes
Combines unstructured p2p overlay with DHT(Distributed hash table)

BLOOM FILTER: The bloom filter utilizes the hashing technique for the search of best document. The bloom filter gets the Query from the node, it performs multiple hashing in the query and as a result it converts the query into URLs.

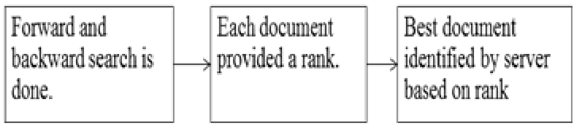

Utilises hashing technique for searching
Gets query from node
Performs multiple hashing in the query
Converts the query into URLs

DISTRIBUTION OF BLOOM FILTER AMONG THE NODES: Once the data are converted into the URL's, the url's are distributed to all other nodes. Once the node the request for the particular data in the network, the nodes will check the for the data or Urls related to the requested data. Once search has been finished, the best results will display to the user.


URLs distributed to other nodes
Checks for URLs related to requested data
After succesful search , returns the relevant items

RANKING AND RETRIEVAL OF DATA: Using the chord algorithm, the peer node will do forward and backward search and as a result each document is provided with the rank and hence according to the rank given, the best document is identified by the server and it is given to the user efficiently.

After ranking the documents, the user can choose the required data that they wanted. By using the Bloom Filter Concept, an effective and efficient data retrieval process is achieved in the Unstructured P2P Networks.


Forward and backward search is done.
Each document provided a rank.
Best document identified by server based on rank

CONCLUSION

In this paper, a round-based data replication strategy called IPFRF has been implemented. IPFRF is based on PFRF but overcomes the shortcomings of PFRF. IPFRF is superior to PFRF in terms of average file delay per request, average file bandwidth consumption per request, and percentage of files found. IPFRF strategy achieved a reduction in average file delay per request up to 19.38 and 60.74 percent in scenarios 1 and 2, respectively, while it achieved a reduction in average file bandwidth consumption per request up to 18.00 and 55.84 percent in the same scenarios. Additionally, IPFRF strategy achieved an improvement in percentage of files found up to 46.69 and 217.81 percent in scenarios 1 and 2, respectively.Bloom Filter also simplifies the retrieval process and the words retrieved are at the base level (using Stemming algorithm).

REFERENCES

[1] M. Bsoul, “A framework for replication in data grid,” in Proc. 8th IEEE Int. Conf. Netw. Sens. Control, Delft, The Netherlands, 2011, pp. 234–236.

[2] D. G. Cameron, A. P. Millar, C. Nicholson, R. Carvajal-Schiaffino, K. Stockinger, and F. Zini, “Analysis of scheduling and replica optimisation strategies for data grids using Optorsim,” J. Grid Comput., vol. 2, no. 1, pp. 57–69, 2004.

[3] R. Chang and H. Chang, “A dynamic data replication strategy using access-weights in data grids,” J. Supercomput., vol. 45, no. 3,

[4] S. Figueira and T. Trieu, Data Replication and the Storage Capacity of Data Grids. Berlin, Germany: Springer-Verlag, 2008, pp. 567–575.

[5] H. Lamehamedi, B. Szymanski, Z. Shentu, and E. Deelman, “Data replication strategies in grid environments,” in Proc. 5th Int. Conf. Algorithms Architectures Parallel Process., 2002, pp. 378–383.

[6] K. Ranganathan and I. Foster, “Identifying dynamic replication strategies for a high-performance data grid,” in Proc. GRID ’01:Proc. 2nd Int. Workshop Grid Comput.. London, United Kingdom:Springer-Verlag, 2001, pp. 75–86.

[7] M. Tang, B. Lee, C. Yeo, and X. Tang, “Dynamic replication algorithms for the multi-tier data grid,” Future Generation Comput. Syst., vol. 21, no. 5, pp. 775–790, 2005.

[8] S. Park, J. Kim, Y. Ko, and W. Yoon, “Dynamic data grid replication strategy based on internet hierarchy,” in Proc. 2nd Int. Workshop Grid Cooperative Comput., 2003, pp. 838–846.

[9] Q. Rasool, J. Li, G. Oreku, and E. Munir, “Fair-share replication in data grid,” Inform. Technol. J., vol. 7, no. 5, pp. 776–782, 2008.

[10] J. Wu, Y. Lin, and P. Liu, “Optimal replica placement in hierarchical data grids with locality assurance,” J. Parallel Distrib. Comput.,vol. 68, no. 12, pp. 1517–1538, 2008.