IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 16, Issue 5, Ver. III (Sep – Oct. 2014), PP 69-72
www.iosrjournals.org

Topology Based Data Replication Strategies in Grid

Joseph John[1], Abin Paul[2], Binu A[3], Ginumol Jose[4], Asish P Mathew[5]
Department of Information Technology, Rajagiri School of Engineering & Technology, India

***Abstract*:** *Over the course of time many data replication strategy has been used in grid. Replication of data increases the access to data by reducing the data latency, it also increases reliability by maintaining multiple copies of data. This paper introduces three strategies that can be used to replicate data in the grid using the concepts of graph centrality and betweenness.*
Keywords: *centrality; betweenness; replication; data grid*

I. Introduction

Grid computing is an essential part of scientific experiments which involves large amount of computations. An example is the European data grid in CERN. The data involved in these experiments are large and often critical to the computations. So its is important that the data is made available to stakeholders that are authorized to access the data in the data grid. The replication strategies in the current literature do not give due importance to the topology or the structure of the grid [1] [2]. This paper introduces three strategies that duplicates data in specialized nodes. These nodes are selected based on the concepts of degree centrality and betweenness.

The grid can be represented as a graph G=(V,E) where g is the set of all vertices and E is the set of all edges. The vertices represent the nodes in the grid and the edges represent the connection between these nodes. The strategies presented here is to be used to augment the performance of the current replication strategy and not to replace them.

II. Strategies based on degree centrality

Degree centrality is the measure of the nodes incident on the node [4]. Higher the degree centrality of a node, higher is the number of other nodes that are reachable from the node. So if the data is present in that node it is easy for many other node to access the data. In current method the data is replicated at a node if the request for that at that node exceeds a certain threshold. By duplicating the data at a single node the performance increase happens only till that node need the data. For example in the below graph, Fig 1, suppose the required data is in node 1. When the request for a data crosses the threshold in node 4 the data is replicated at that node 4. The data has to travel through two hops to reach node 4. Now if after that data transfer if node 6 qualifies for data replication then the data still has to travel two hopes. The earlier data transfer had no effect in reducing the data latency of the second data transfer. The following strategies make sure that this is not the case.

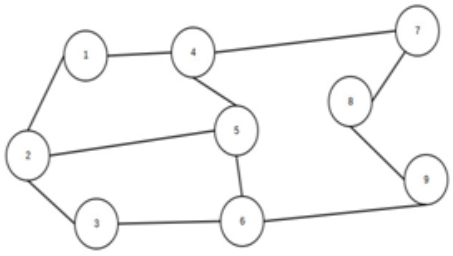

Fig 2.1: Degree centrality

A. Replicating data in nodes with maximum centrality

In this strategy when the data is replicated in a node it is also replicated in all the nodes with maximum centrality. For example in the above graph all the data that is replicated will also be replicated in nodes 2 and 5. Now if we compare it with the above scenario we can see that as a copy of the data exist in node 5, node 6 can access the data in a single hop. This reduces the data access latency in many cases.

The problem with the above strategy is that in some cases the node with maximum centrality may be far away from majority nodes. For example, Fig 2, in the below graph node 3 has maximum centrality, but its more than one hop distance away from majority of the nodes. In this case replicating data at node 3 may not improve the performance.

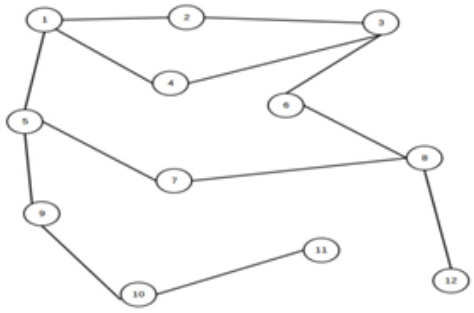

Fig 2.2: Issues with degree centrality, scenario 1

Another issue the above strategy will face is that in some cases the nodes with maximum centrality would be close to each other. When this happens the same data will be duplicated at neighboring nodes even when they or their neighbors will have no use of the data. Thus it leads to unnecessary data replication. For example in the graph below, Fig 3, the node with maximum centrality are node 6 and 7. So copying data in node 6 as well node 7 will increase the traffic overhead for the whole grid without much increase in the performance.

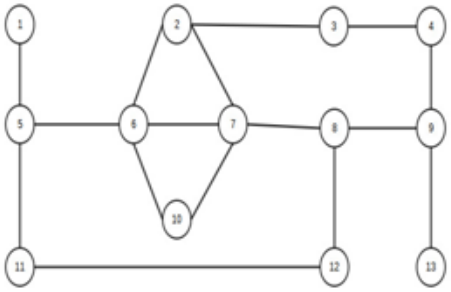

Fig 2.3: Issues with degree centrality, scenario 2

B. Replicating data in nodes with maximum centrality in a path

This strategy overcomes many shortcomings of the above strategy. In this case when the data is replicated on another node it is also replicated on another node in the path which the data travels. This node is selected on the basis of its degree centrality; the node selected will have maximum centrality in the path. For example in the graph given below, Fig4, if data from node 2 is replicated in node 7 it will also be replicated in node 4. This is because in the path 2-1-4-7 node 4 has the maximum centrality. Likewise if the data from node 4 is replicated at node 9 it will also be replicated in node in node 6.

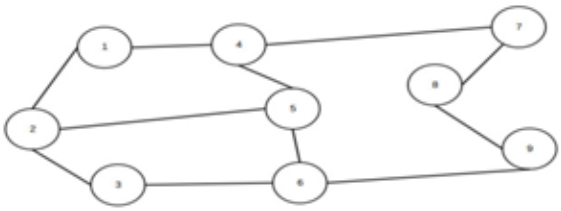

Fig 2.3: Second strategy based on degree centrality

The problem with this strategy is that in a path more than one node may have same centrality and they may be neighbors. This will lead to unnecessary copies which may not result in performance improvement. Also there may be multiple path from one node to another and while copies in one path may improve the performance the other may not.

III. Strategies based on betweenness of the nodes in a graph

Replicating data on nodes based on the betweenness of the node is another strategy this paper uses. The betweenness centrality of a node v is the total number of times that particular node appears in all pair shortest paths of a graph. let short(i,j) be shortest path between i and j. The informal algorithm is as given below. Here the assumption is that the shortest path for all pairs of nodes are already computed.

```
begin

    for i from 1 to n
          for j from 1 to i
                for  k from 1 to n
                  if k in short(i,j)
                    weight(k)++;
end
```

Node v is said to have maximum betweenness if weight(v) is the maximum in the graph. Finding the shortest path will take O(n3) if we use Floyd-Warshall [3] or quadratic version of dijkstras algorithm, so finding the betweenness also takes the same amount of time i.e O(n3). So the above algorithm for finding the betweenness also have a time complexity of O(n3). If the node with maximum or relative high betweenness is selected to store to store a replica, then the probability that it appears in any path is higher than other nodes. This makes it an efficient replica node.

When data is copied from node i to j,the probability that v, which is the node with maximum betweenness centrality, appear in the bath between i to j is high. So when the data is copied at node j the data is also copied at node v. If

weight(v)/no: of shortest path = 1

it means that v appears in all paths. In such a case there is no extra overhead to copy data in v. Even if that is not the case the overhead to copy data to node v is low. This I because node will be closer to the geometrical centre of the network, as a result it will be easy for other nodes to access the data from node v.

IV. Simulation and Results

This paper has proposed three replication strategies in grid. Two are based on degree centrality and one based on betweenness centrality. The performance of the strategies are evaluated against the replication strategies used now. That is storing copies in nodes based on the number of requests arrived on that particular node. The below given simulation confirms these strategies will have better performance. The simulation is done using ns2, the strategies based on degree centrality and betweenness centrality are compared with the existing replication mechanism. Simulation transfers a file of size 97.7 kb at a rate of 1Mbps.The packet size is 100 bytes with an .005 sec delay between packets.

In a network with 20 nodes,3 replicas are created using random conventional methods and the centrality based approach. Time to copy data from the replicas is shown in the fig 5.in fig 6 a single replica is created on a node having maximum betweenness and it is compared with a random selection of replica. The simulation shows that the time taken for file transfer for the proposed methods are lower than the existing methods.

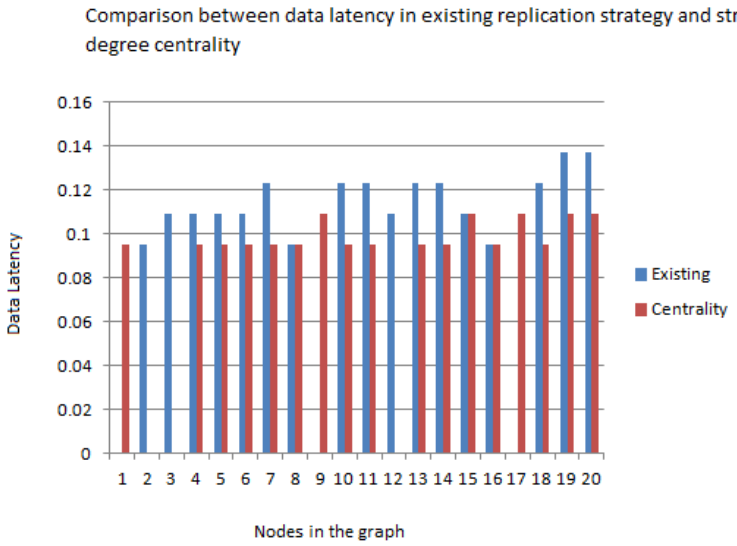

Fig 4.1: Degree centrality versus existing system

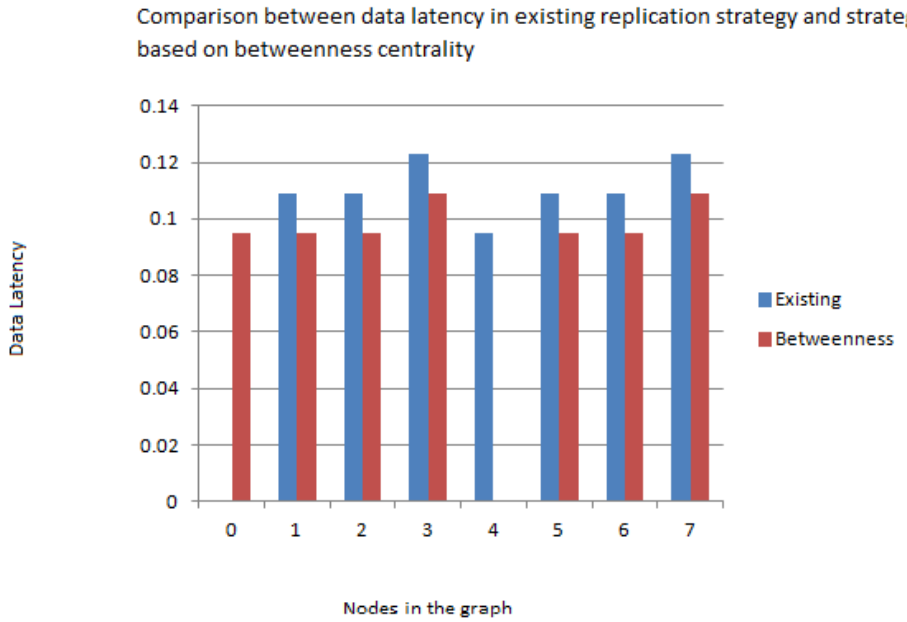

Fig 4.2: Betweenness centrality versus existing system

V. Conclusion

This paper prepossesses three strategies that for data replication in grid. The strategies are based on degree centrality and betweenness centrality. The simulation of the strategies vis-a-vis the current strategy shows that the method proposed here gives better result. Even with the improvements this paper proposes there are many areas of improvement possible. In the strategies proposed here hopcount is taken as the metrics for distance.. The problem with algorithms based on hopcount is that it does not take the real time issues like bandwidth availability or congestion in the communication medium. The improvement for this strategies should concentrate on tackling the issues with these real world issues.

References

[1]. Kavitha Ranganathan and Ian Foster , "Identifying Dynamic Replication Strategies for a High- Performance Data Grid ,"Grid Computing — GRID 2001 Lecture Notes in Computer Science Volume 2242, 2001, pp 75-86

[2]. Bill Allcock, Joe Bester, John Bresnahan, Ann L. Chervenak, Ian Foster, Carl Kesselman, Sam Meder, Veronika Nefedova, Darcy Quesnel, Steven Tuecke, "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing" in Mass Storage Systems and Technologies, IEEE,2001

[3]. Chai Deng-feng, Zhang Deng-rong"Algorithm and its application of N shortest paths problem "Info-tech and Info-net, 2001. Proceedings. ICII 2001 – Beijing IEEE

[4]. M.E.J Newman, "A measure of betweenness centrality based on random walks," oSocial Networks, Volume 27, Issue 1, Pages 39-53, Elsevier