ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A SEMANTIC BASED SCHEDULING ALGORITHM FOR DATA INTENSIVE APPLICATIONS ON GLOBAL GRIDS

Mr. P.Kumar*, Dr. M.Moorthi
* Department of Computer Applications, Kongu Arts and Science College, Erode-638107.
Department of Computer Applications, Kongu Arts and Science College, Erode-638107.

ABSTRACT

Due to the increasing size of data intensive applications , the datasets are need to be stored in a distributed manner and multiple copies of data sets has to be replicated to provide easier access to the grid applications. In these kinds of systems, scheduling is a key challenge because of the location of data and heterogeneity of a node by holding multiple types of data sets in the same location. We propose a promising technique to schedule the grid applications, using semantic based classification. Our algorithm classifies the grid resources according to the semantic meanings of data sets it holds, using which it identifies the data location which are necessary for the grid applications to be executed successfully.

KEYWORDS: Network Algorithm, Distributed computing, Application and Grids,

INTRODUCTION

The computational grid is a promising platform that provides large resources for distributed algorithmic processing. Such platforms are much more cost-effective than traditional high performance computing systems. However, computational grid has different constraints and requirements to those of traditional high performance computing systems. To fully exploit such grid systems, resource management and scheduling are key grid services, where issues of task allocation and load balancing represent a common problem for most grid systems. The load balancing mechanism aims to equally spread the load on each computing node, maximizing their utilization and minimizing the total task execution time. In order to achieve these goals, the load balancing mechanism should be 'fair' in distributing the load across the computing nodes. This implies that the difference between the heaviest-loaded node and the lightest-loaded node should be minimized. In this paper, we will use a standard performance metric—the application make span, to evaluate our approaches and other approaches that have been proposed in the literature. The application make span is defined in this study as the amount of time taken from when the first input file is sent for computation (to a computing node), to when the last task is completed by a computing node.

BACKGROUND STUDY

There has been many methodologies been proposed earlier to schedule the data oriented applications. In this chapter we review few of the techniques:

Centralized Load Balancing Approach:
In the centralized approach, one node in the system acts as a scheduler and makes all the load balancing decisions. Information is sent from the other nodes to this node. This generates traffic in the grid networks and also increases the scheduling and execution time. Overall this approach increases the cost of a process.

Decentralized Approach:
In the decentralized approach, all nodes in the system are involved in the load balancing decisions. It is therefore very costly for each node to obtain and maintain the dynamic state information of the whole system. Most decentralized approaches let each node obtains and maintains only partial information locally to make sub-optimal decisions. Static load balancing algorithms assume all information governing load balancing decisions that can include the characteristics of the jobs, the computing nodes and the communication network are known in advance. Load balancing decisions are made deterministically or probabilistically at compile time and remain constant during runtime. The static algorithms have one major disadvantage—it assumes that the characteristics of the computing resources and communication network are all Known in advance and remain constant. Such an assumption may not apply to a grid environment, Refer [1].

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

DYNAMIC LOAD BALANCING

Dynamic load balancing algorithms attempt to use the runtime state information to make more informative load balancing decisions. Undoubtedly, the static approach is easier to implement and has minimal runtime overhead. However, dynamic approaches may result in better performance. One simple load balancing algorithm is the Best-fit algorithm. In this algorithm, tasks are assigned according to their order in the queue. Each task is then scheduled to available computing nodes based on the completion time offered by the nodes; the node that completes the task the fastest (taking into account its current load) is chosen.

Histogram Based Scheduling

This algorithm is proposed for global load balancing in structured P2P systems. Each node in these systems has two key components: 1) a histogram manager maintains a histogram that reflects a global view of the distribution of the load in the system, and 2) a load-balancing manager that redistributes the load whenever the node becomes overloaded or under loaded. They exploit the routing metadata to partition the P2P network into non overlapping regions corresponding to the histogram buckets.

SCP Based Scheduling

In this the application is scheduled according to the availability of data set to execute the process. It maintains the Meta data of grid systems and using which it checks for the availability and requirement of data sets to execute the process successfully. After that a grid which contains maximum data will be chosen to execute the process.

PROPOSED METHOD

We propose a novel methodology to schedule the data oriented grid applications in an efficient way to reduce the overall execution time of the applications.

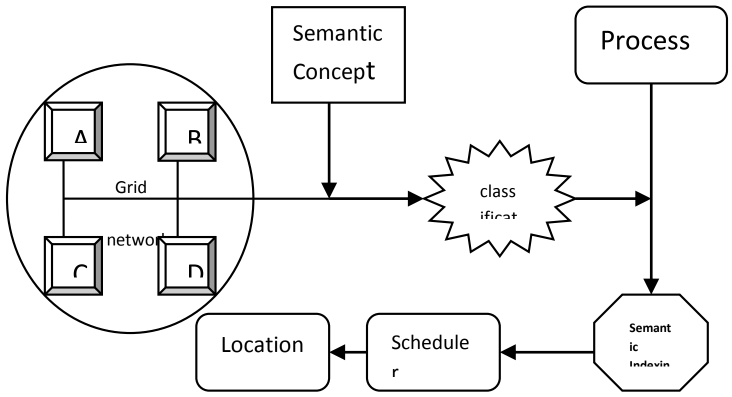


Fig1: Represents overall architecture of the proposed system.

Our proposed methodology consists of the following steps:

PRE-PROCESSING

In this step we read the semantic concepts and the Meta data about the data sets available in the data grids. we remove the noisy semantics and meta data and clean the meta data to ensure the quality of semantic indexing.

Grid Classification

At this step with the output of pre-processing, we compute the similarity measure between the semantic concepts and the Meta data of the data sets. Based on the similarity values we group the grids into set, which will be further used to look up the grids according to the requirement of the processes. We compute Euclidean distance to compute the similarity measure. The classified results are indexed into the data base. We use semantic indexing to store the classification results. In semantic indexing, the Meta data are stored under a semantic concept based on the similarity value of Meta data with the semantic concept.

Algorithm for Grid Classification

Step1: Read semantic concepts
Step2: Read Meta data of data sets.
Step3: clean concepts and Meta data.
Step4: compute similarity measure between semantic concepts and Meta data.
Step5: Identify similar concepts and Meta data.
Step6: Group similar Meta data, under related concepts.

Scheduling

This step performs the scheduling of the process to help the process to be executed with in short time. The scheduler accepts the input job and retrieves the location of the data from the indexed data. Because of we use semantic indexing; it's very easier for the scheduler to retrieve the location of the data objects. This reduces the scheduling time of the processes and also execution time will be less.

Algorithm for Scheduling
Step1: Read Input Job
Step2: Identify set of data objects necessary to execute the job.
Step3: compute similarity measure of data objects with semantic concepts.
Step4: Identify the semantic concept with respect to similarity measure.
Step5: retrieve the location of datasets from the indexed results.
Step6: return the results.

At the end of the scheduling process the application will be returned with the location of the grid where the application has to be executed. The query processor will post the process to the returned location and will wait for the result and return the result to the application.

RESULTS AND DISCUSSION

The final results shows that our proposed scheduling algorithm reduces the overall execution time of the application by reducing the scheduling time and execution time. Our indexing scheme reduces the scheduling time. . The graphs in Figures shows the number of jobs completed versus time for the two scheduling strategies presented. Since the computation time was dominant, within cost minimization, the jobs were executed on the least economically expensive compute resource.

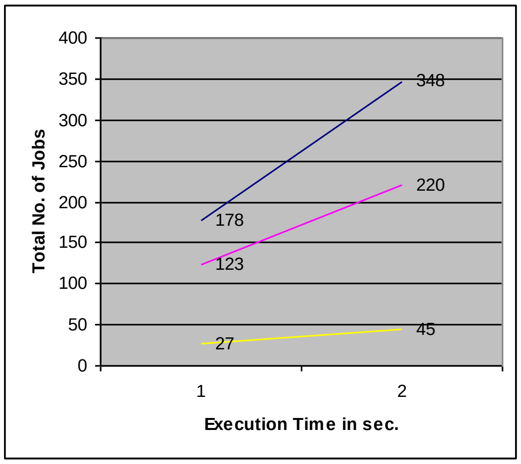
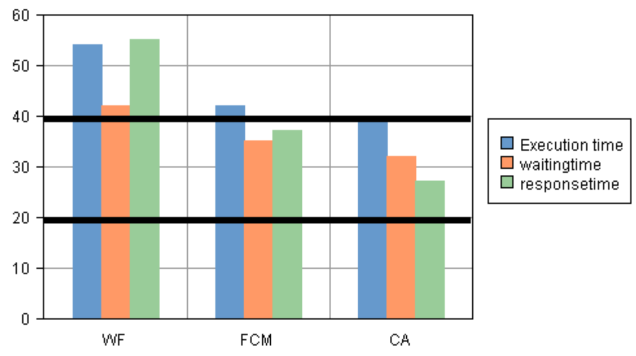


Fig: shows the analysis of different no. of process and time taken with different algorithm.
Blue line: Histogram based load balancing algorithm
Pink: Scp based scheduling
Yellow: Our algorithm.



The above figure shows the overall execution time is increased compare with the waiting time and response time

CONCLUSION

The proposed method achieves good results and reduces the overall execution time. We further analyse the query execution process to reduce the overall execution time. In this paper, we proposed a framework, semantic based global load balancing to enable global load balance for structured P2P systems. Each node maintains the load information of nodes in the systems using histograms. This enables the system to have a global view of the load distribution and hence facilitates global load balancing. We partition the system into non overlapping groups of nodes and maintain the average load of them in the histogram at a node. Even though the proposal is a general framework, it is possible to deploy different kinds of P2P systems on it. We demonstrated this by building

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 2.114

three well-known structured P2P systems: Skip Graph, BATON, and Chord on our proposal. Our performance evaluation shows that the proposed system is superior over other methods.

REFERENCES

1. S3: Scalable, Shareable and Secure P2P Based Data Management System, 2008.
2. V.R. Alvarez, E. Crespo, J.M. Tamarit, Assigning students to course sections using tabu search, Ann. Oper. Res. 96 (2000) 1–16.
3. I. Forster, C. Kesselman, The Grid: Blueprint for a New Computing Infrastructure, Morgan Kaufmann, 1998.
4. F. Glover, Future paths for integer programming and links to artificial intelligence, Comput. Oper. Res. 13 (1986) 533–549.
5. F. Glover, Tabu search. I, ORSA J. Computer. 1 (1989) 190–206.
6. F. Glover, Tabu search. II, ORSA J. Computer. 2 (1990) 4–32.
7. F. Glover, J.P. Kelly, M. Laguna, Genetic algorithms and tabu search: Hybrids for optimization, Computer. Oper. Res. 22 (1995) 111–134.
8. F. Glover, M. Laguna, Tabu search, in: D. Du, P.M. Pardalos (Eds.), in: Handbook of Combinatorial Optimization, vol. 3, Kluwer Academic
9. Publishers, Dordrecht, 1999, pp. 621–757.
10. G. Barbarosoglu, D. Ozgur, A tabu search algorithm for the vehicle routing problem, Comput. Oper. Res. 26 (1999) 255–270.
11. R.R. Brooks, S.S. Iyengar, J. Chen, Automatic correlation and calibration of noisy sensor readings using elite genetic algorithms, Artificial Intelligence 84 (1996) 339–354.