Efficient and Collective Global, Local Memory Management For High Performance Cluster Computing

P. Sammulal† **and Dr.A. Vinaya Babu**††,

Osmania University, Hyderabad, India JNT University, Hyderabad, India

Summary
In Cluster Computing Environment the data latency time has significant impact on the performance when the data is accessed across clusters. Especially when executing data pertaining to satellite images for remote sensing or defense purposes, scientific or engineering applications. Designating a particular cluster for executing an application leaving behind the bandwidth aspects of accessing data across clusters can pose performance degradation. To overwhelm the stated problem we have proposed a new technique that buffers the data in Global Memory and Local Memory and regulate the need for communicating across clusters for data access. This Global Memory constitutes a permanent storage part and a temporary storage part, which are refreshed dynamically and at regular intervals based on the data access patterns. Local Memory regulates the Intra-Cluster communication for data access. Experimental results show performance improvement to considerable levels with the implementation of the concept, specifically when the cost of data access from other clusters is higher and is proportionate to the amount of data.

Keywords:
High Performance Cluster Computing, Job Scheduling, Memory Management,

1. Introduction

The first inspiration for cluster computing was developed in the 1960s by IBM as an alternative of linking large mainframes to provide a more cost effective form of commercial parallelism [1]. However, cluster computing did not gain momentum until the convergence of three important trends in the 1980s: high-performance microprocessors, high-speed networks, and standard tools for high performance distributed computing. A possible fourth trend is the increasing need of computing power for computational science. The recent advances in these technologies and their availability as cheap and commodity components are making clusters or networks of computers such as Personal Computers (PCs), workstations, and Symmetric Multiple-Processors (SMPs) an appealing solution for cost-effective parallel computing.

Cluster computing can be described as a fusion of the fields of parallel, high-performance, distributed, and high availability computing. It has become a popular topic of research among the academic and industrial communities, including system designers, network developers, algorithm developers, as well as faculty and graduate researchers. The recent developments in high-speed networking, middleware and resource management technologies have pushed clusters into the mainstream as general purpose computing system. This is clearly evident from the use of clusters as a computing platform for solving problems in number of disciplines.

In some scientific application areas such as high energy physics, bioinformatics, and remote sensing, we encounter huge amounts of data. People expect the size of data to be terabyte or even petabyte scale in some applications [2]. Managing such huge amounts of data in a centralized manner is almost impossible due to extensively increased data access time. To illustrate the scenario where a scientific application is executed in a cluster computing environment the data requirement of the application would be enormous, the required data may be scattered across several clusters. In this case, streamlining data access through the usage of the proposed memory management technique will improve the performance of the entire operation.

Memory management becomes a prerequisite when handling applications that require immense volume of data for e.g. satellite images used for remote sensing, defense purposes and scientific applications. Here even if the other factors perform to the maximum possible levels and if memory management is not properly handled the performance will have a proportional degradation. Hence it is critical to have a fine memory management technique deployed to handle the stated scenarios.

Scheduling is a challenging task in this context. The data-intensive nature of individual jobs means it can be important to take data location into account when determining job placement. Despite the other factors which contribute performance in a cluster computing environment, optimizing memory management can improve, the overall performance of the same. To address this problem, we have defined a combined memory management technique. The proposed technique focuses on optimizing memory usage, assuming the other factors which contribute to performance are performing to the optimum level.

Manuscript received April 5, 2008
Manuscript revised April 20, 2008

The rest of the paper is organized as follows. Section 2 presents some of the existing works in job scheduling and memory management. Section 3 describes the proposed combined memory management technique. Section 4 concludes the paper.

2. Related Work

Ann Chervenak et al. [3] review the principles that they are following in developing a design for data grid architecture. Then, they describe two basic services that they believe are fundamental to the design of a data grid, namely, storage systems and metadata management.

William H. Bell et al. [4] find the design of the simulator OptorSim and Various replication algorithms. After setting the simulation configuration they dedicated to a description of simulation results.

Kavitha Ranganathan and Ian Foster [5] describe a simulation framework that they have developed to enable comparative studies of alternative dynamic replication strategies. They present preliminary results obtained with this simulator, in which they evaluate the performance of five different replication strategies for three different kinds of access patterns.

Kavitha Ranganathan and Ian Foster [6] develop a family of job scheduling and data movement (replication) algorithms and use simulation studies to evaluate various combinations and they describe a scheduling framework that addresses the problems.

Houda Lamehamedi et al. [7] 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 and their 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.

Sang-Min Park et al. [8] evaluate BHR strategy by implementing it in an OptorSim, a data grid simulator initially developed by European Data Grid Projects and their simulation results show that BHR strategy can outperform other optimization techniques in terms of data access time when hierarchy of bandwidth appears in Internet.

D. G. Cameron et al. [9] discussed an economy-based strategy as well as more traditional methods, with the economic models showing advantages for heavily loaded grids.

3. Combined Memory Management Technique

3.1 Global and Local Memory

Our memory management technique comprises global memory and local memory. Global memory (Mg) is common for all the clusters. Where as local memory is specific to the nodes of every individual cluster.

The global memory (Mg) constitutes a persistent storage and temporary storage. The data which are frequently accessed is stored in the persistent part and the less frequently accessed are stored in the temporary part. Intuitively the bandwidth between the global memory and the clusters will be significantly higher than the bandwidth across clusters. The local memory associated with every individual node of cluster hosts the data pertaining to the task assigned for that node. Simply the local memory consists of data that are required for the task deputed for the corresponding node.

3.2 The Scheduler and Memory Management

When a user makes a request, he specifies the required resources, the estimated execution time and the deadline. The request is forwarded to the scheduler. The scheduler consists of a resource management system which maintains details regarding the resource availability, resource under utilization. This information is updated periodically by the resource management system. Moreover the updations also happen after the completion of running requests.

The scheduler after the reception of a new request makes an analysis to identify a particular cluster to which the request can be forwarded. The scheduler primarily takes in to consideration the load of the processors of the nodes of the concerned clusters before the task is assigned. But this process of designating clusters for processing tasks would not yield optimum performance because bandwidth is also a major factor in determining the performance levels. So to overwhelm this problem we have proposed a new algorithm using global memory and local memory.

The conventional scheduling algorithm blindly fixes a particular cluster taking into account the availability of data the as the sole criterion. This method of designating a particular cluster for a request would lead to performance degradation. To illustrate the above scenario let us consider a particular request requires certain the cluster that is identified for the given request is based on the presence of major portion of required data and the cost for accessing remaining data is not considered and if it is significantly higher, then it has to be treated in a separately.

At the same time, if the task is designated to a cluster irrespective of the percentage of data present in that cluster and considering the cost of accessing the remaining data from the rest of the clusters through global memory the performance can be optimized further. We have proposed a new algorithm in 3.3 that also gives the needed importance to the cost of accessing data

3.3 Scheduling Algorithm

Assumptions

N_C	$\rightarrow$	Total Number of Clusters
C_{li}	$\rightarrow$	The Cluster handling the current job.
S_F	$\rightarrow$	Set of files requires for the file job (I)
SN_{WC}	$\rightarrow$	Set of nodes having the SFWC in the Me within CJi
SS_C	$\rightarrow$	Set of clusters having FMJ
SF_{WC}	$\rightarrow$	is a set of files available in CJi
SF_{Mg}	$\rightarrow$	is a set pf files to be transferred from SSC through Mg.

```
For Files within a Cluster
    for each files in $SF_{WC}$
        for each node in $SN_{WC}$
            t = Calculate time
        end
          $t_{\min}$ = min (t)
          Update $SQN_{WC}$
    end
$T_{WC} = \sum_{i=0}^{Sizeof\ (SF_{WC})} t_{\min}$

For Files between Clusters
    for each files in $SF_{Mg}$
        for each cluster in $SS_C$
            t = Calculate time to transfer file from $SS_{Ci}$
            through $M_g$
        end
    $t_{\min}$ = min (t)
    Update $S_{qc}$
end
```

$$T_{BC} = \sum_{i=0}^{sizeof\ (Sqc)} t_{\min}$$

$$T = T_{WC} + T_{BC}$$

Repeat the above steps for all the clusters

$$S_T = (T_0, T_1, T_2 ....T_{NC})$$

$$T_Q = \min(S_T).$$

Corresponding cluster is chosen to allot the job.

At regular intervals the data access patterns in Global Memory is analyzed. If the data stored in temporary portion has been accessed more frequently then it is shifted to the permanent storage part. Similarly the data present in the permanent storage part is also deleted to pave the way for new storage if it is not frequently referred.

4. Conclusion

The proposed technique for data access across clusters shows substantial improvement reducing execution time. Providing due consideration to data access latency besides computational capability proves worthwhile. The proposed concept uses a combination of Local Memory and Global Memory that buffers data based on access patterns. The Local Memory takes care of moderating communication across nodes within the cluster. The Global Memory is bifurcated in to a Temporary Storage part and a Persistent Storage part. Data placed in these to parts are updated dynamically and at regular intervals based on the pattern of usage. The concept comes handy when the data to be accessed from another cluster is markedly costlier than if the same data is accessed through the Global Memory. Replenishing Global Memory dynamically and at regular intervals improves data availability further and eliminates frequent access to other clusters.

5. References

[1] R. Buyya (ed.), High Performance Cluster Computing: Architectures and Systems, vol. 1, Prentice Hall, 1999.

[2] Wolfgang Hosehek, Francisco Javier Jaen-Martinez, Asad Samar, Heinz Stockinger, and Kurt Stockinger. "Data Management in an International Data Grid Project," Proceedings of First IEEE/ACM International Workshop on Grid Computing (Grid'2000), Vol. 1971, pages 77-90 Bangalore, India, December 2000.

[3] A.Chervenak, I. Foster, C, Kesselman, C. Salisbury, and S. Tuecke, "The Data Grid: Towards an Architecture for Distributed Management and Analysis of Large Scientific Datasets", Journal of Network and Computer Applications, vol. 23, Pages 187-200, 2000.

[4] W. H. Bell, D.G. Cameron, L. Capozza, P. Millar, K. Stockinger, and F. Zini, “ Simulation of Dymanic Grid Replication Strategies in OptorSim”, Proceedings of the Third ACM/IEEE International Workshop on Grid Computing (Grid2002), Baltimore, USA, vol. 2536 of Lecture notes in Computer Science, Pages 46-57, November 2002.

[5] I. Foster, and K. Ranganathan, “ Identifying Dynamic Replication Strategies for High Performance Data Grids”, Proceedings of 3rd IEEE/ACM International Workshop on Grid Computing, vol. 2242 of Lecturer Notes on Computer Science, Pages 75-86, Denver, USA, November 2002.

[6] I. Foster, and K. Ranganathan, “ Decoupling Computation and Data Scheduling in Distributed Data-intensive Applications”, Proceedings of the 11th IEEE International Symposium on High Performance Distributed Computing (HPDC-11), IEEE CS Press, Pages 352-368, Edinburgh, U.K., July 2002.

[7] E. Deelman, H. Lamehaedi, B. Szymanski, and S. Zujun, “ Data Replication Strategies in Grid Environments”, Proceedings of 5th International Conference on Algorithms and Architecture for Parallel Processing (ICA3PP’2002), IEEE Computer Science Press, Pages 378-383, Bejing, China, October 2002.

[8] Sang-Min Park, Jae-Hoon Kim, Young-Bae Go, and Won-Sik Yoon, “ Dynamic Grid Replication Strategy based on Internet ?Hierarchy”, Lecturer Note in Computer Science, International Workshop on Grid and Cooperative Computing, vol. 1001, pp.1324-1331, Dec.2003.

[9] DG Cameron, AP Millar, and C. Nicholson, “OptorSim: a Simulation Tool for Scheduling and Replica Optimization in Data Grids”, Proceedings of Computing in High Energy Physics, CHEP 2004, Interlaken, Switzerland, September.

P Sammulal is Pursuing his PhD (Computer Science and Engineering) from Osmania University, Hyderabad, India. He has received the B.E degree from Osmania University in 2000 and MTech degree from JNT University in Computer science and engg., in 2002. He is in teaching and research since 2002. He has published 3 papers in International conferences/journals. His research is focused on memory management in cluster computing, face recognition in image processing He is working as an Assistant Professor in JNT University, Kakinada, INDIA.

Prof.Dr.A.VinayaBabu received PhD (Computer Science) from JNT University, Hyderabad, India. He has received BE and ME from Osmania University in Electronics and Communication Engineering, MTech from JNT University in Computer Science and Engineering. He is a professor in CSE and Director of School of Continuing and Distance Education in JNT University, Hyderabad, India. He is a life member of CSI, ISTE member of FIE, IEEE, and IETE. He has published about 35 research papers in International/National journals, International/National Conferences and Refereed International Conferences. His current research interests are distributed/parallel computing, Cluster computing, Grid computing, Network security.