Asian Journal of Information Technology 5 (11): 1210-1212, 2006

Parallel Matrix Multiplication on Grid Enabled PC Cluster

[1]Karanjeet Singh Kahlon [2]O.P. Gupta and [2]Rakesh Jindal
[1]Department of Computer Science and Engineering, G.NDU, Amritsar, India
[2]Faculty of Computer Science, PAU, Ludhiana, 141004 India

Abstract: GPCC is a Grid enabled PC Cluster system developed for harnessing computational resources available over LAN for parallel computing. Parallel computing is essential for solving very large scientific and engineering problems. An effective parallel computing solution requires an appropriate parallel machine and a well-optimized parallel program. Most parallel matrix multiplication algorithms use matrix decomposition based on square root of the number of processors available. In this reserch, a grid enabled PC Cluster based on Windows OS integrating the DeskGrid API to communicate among the processors is u The tasks are allocated to the processors based on the log of the processor status and errors committed. sed. The decomposition of the matrix is independent of the no. of processors available for the computation. Dynamic allocation of the matrix pointer not only proved to be distributed memory efficient but also time efficient. This reserch is an attempt to highlight the ease and stable nature of the Grid enabled PC Cluster (GPCC) based on Windows OS.

Key words: Grid computing/cluster computing, parallel processing, parallel algorithm

INTRODUCTION

Ever since the invention of the computer, users have demanded more and more computational power to tackle increasingly complex problems. Parallel computing is essential for solving such very large scientific and engineering problems. An effective parallel computing solution requires an appropriate parallel machine and a well-optimized parallel program. Loosely coupled personal computers in a workgroup over the Intranet are very promising candidate for parallel machine in the scenario. Though, networked machines are having different types of processors with varying clock speed yet computing on the networked machines are becoming very popular to solve both data intensive and compute intensive scientific problems due to the demand for higher performance and lower cost.

In Distributed Parallel Computing, each processor has its own memory and can only access its local memory. The processors are connected with the other processors via a high-speed communication network. A common approach to programming multiprocessors is to use message-passing library routines in addition to conventional sequential program. Processors exchanges information with one another using *send* and *receive* operations. Such a design of distributed memory architecture using one way or two way communication between cooperative tasks within a parallel application is also called distributed parallel system.

In the past years, several research projects have been conducted to explore this area of exploiting idle machines available on a local area network to be used for distributed parallel computing, which constitute a Grid of PCs with 90% of the CPU power unused. In what follows, performance models, the sequential and parallel algorithms for matrices multiplication and the perform-ances and their analysis are presented.

PERFORMANCE MODELS

Speedup: Speedup[1] is used to quantify the performance gain from a parallel computation of a fixed size application on a single machine in the network. It is defined as follows:

Definition 1. The speedup for executing program P, denoted by S (P) in a network of PCs is given by

$$S(P) = T(P)_{seq} / T(P)_{par}$$

Where $T(P)_{seq}$ is the time taken by P on one single machine and $T(P)_{par}$ is the time on no. of PC in a network.

Efficiency: Efficiency or utilization is a measure of the time percentage for which a machine is usefully employed in parallel computing

Definition 2. The efficiency of parallel computing of application P in a network of PCs is defined as the ratio of the total effective computing time to the total available unused cycles in the network of PCs.

Correspoding Author: Karanjeet Singh Kahlon, Department of Computer Science and Engineering, G.NDU, Amritsar, India

Scalability: Scalability measure the ability of a parallel machine to improve performance as there are increases in the size of the application program and in the size of the system involved.

Definition 3. Scalability[2] exhibits performance linearly proportional to the computing power of the network i.e. number of the PCs used for computation.

The execution time of running program P on a Grid enabled PC Cluster can be divided into three distinct parts on each PC.

- T_{used} : it is used computing time at the PC
- T_{par}: it is effective time to compute subtask t, on networked PC, excluding the task creation time.
- T_{over}: it is the overhead of communication time, memory access on the networked PC.

MATRIX DECOMPOSITION FOR PARALLEL ALGORITHM

The matrix multiplication application is well suited for checking both type of granularity of parallel application i.e fine grain and coarse grain applications. The product C of the two matrices A and B is defined by

$$c_{ij} = \sum_{k=0}^{n-1} a_{ik} \times b_{kj}$$

where a_{ij}, b_{ij} and c_{ij} is the element in ith row and jth column of the matrix A, B and C respectively. In order for the matrix multiplication to be defined, the dimensions of the matrices to be multiply as AB = C must satisfy $(n\times m)(m\times p) = (n\times p)$. In this study, conformable square matrices will be used for simplicity. The matrices A, B and C are all $n\times n$ matrices. Sequential algorithm requires n^3 additions and multiplications; therefore its time complexity is $O(n^3)$.

To implement the matrix multiplication in parallel[2,3], the matrix A is decomposed into several multiple rows as shown in the Fig. 1 depending on the number of processor available for computing under the network.

The parallel algorithm for installed network of PCs has two main characteristics.

- It is based on Master-Slave paradigm
- Equal workload distribution

Algorithm

Step 1. Master (Head Node) reads data from user
Step 2. Master decompose the matrix A into multiple rows

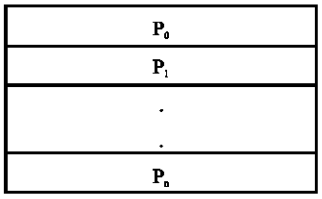

Fig. 1: Row decomposition

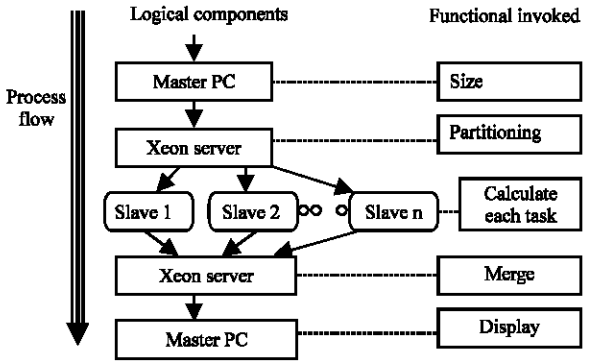

Fig. 2: Communication protocol

Step 3. Master broadcasts dynamic allocation of matrix B columns to slaves
Step 4. Master sends respective parts of first matrix to all other processes.
Step 5. Every process performs its local multiplication.
Step 6. All slave processes send back their result.
Step 7. Master (process 0) reads data and merges them.

To make this algorithm memory efficient, dynamic allocation of the rows and columns are being made to the nodes of grid enable cluster as shown in the Fig. 2. Head node or Master Computer is sending the pointer to the nodes. The algorithm is made time efficient by decomposing the matrix A in to sub tasks depending on the number of processors available and follows the following formula for assigning number of rows to each cluster node.

$$\text{No. of rows} = \frac{\text{Size of the matrix}}{\text{Number of processors}}$$

A 3-tier network design is used to implement the parallel matrix multiplication. The network design consists of three components.

- Master PC (Submitter)
- Server
- Cluster of nodes (Executors)

This environment is implemented using DeskGrid API and DLL files. DeskGrid is a full implementation of communication over TCP sockets Microsoft Win32 Platforms. The use of direct task-to-task TCP connections has been found to significantly outperform the older UDP-based daemons routed message passing implementation in PVM[3]. This feature opens up the possibility of utilizing resources commonly excluded from network parallel computing systems such as Macintosh and Windows based PCs. Tasks are managed by background daemon which is resident on each node of the grid enabled cluster. The daemon communicates with each other using TCP protocol. The message size is kept to the packet size of 4 k keeping in view the BDP (Bandwidth Delay Product) value calculated based on the Windows TCP Buffer[4] size of 64 K.

Moreover, we assume that at any given instant only one parallel program is in execution on the cluster and that the main memory of each desktop system is large enough to accommodate the working set of the parallel process it executes. Finally, we assume that the communication network carries only traffic generated by the desktop PC in the cluster (both by the parallel program and from jobs executed by other desktop PC).

The array sizes tested varied from 800×800 to 3200×3200. While each program was run its total execution time was recorded using the timing routine within the C compiler. The time for communicating the column matrix is also recorded within the program.

RESULTS AND DISCUSSION

To evaluate the performance of the above designed parallel machine and using the problem discussed in the earlier section as the parallel program, a set of experiments were conducted and data is recorded in tables as below. From the Table 1, the communication time shows that there is little variation, which is attributed due to the window TCP size was kept under BDP size.

Table 1: Computation vs communication time

Size	Process	MM Time (sec)	Speed up	Comm. Time (sec)	Effective speed up
800	1	28.72			
	2	24.56	1.169	0.295	1.183
	4	18.20	1.578	0.330	1.607
	8	12.45	2.306	1.580	2.642
	16	14.25	2.015	2.670	2.483
1600	1	42.80			
	2	34.30	1.247	0.980	1.285
	4	22.72	1.883	1.680	2.034
	8	16.50	2.593	2.506	3.058
	16	16.90	2.533	3.860	3.282
3200	1	58.20			
	2	40.36	1.442	1.90	1.513
	4	30.80	1.882	2.86	2.080
	8	25.70	2.264	3.80	2.654
	16	20.45	2.845	4.06	3.550

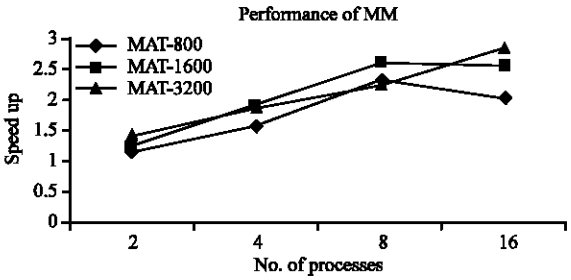

Fig. 3: Speed Vs Number of processor

It is seen that ratio of computation to communication time is very high and this design under study is well suited to coarse grain applications. Moreover, a speed up of 40% is achieved using dynamic allocation of the matrix pointers to cluster nodes.

CONCLUSION

In the present study, dynamic distribution of matrix pointers to the client nodes not only improved the execution time but also reduced the communication time. Such design of the Grid enable PC Cluster (GPCC) is well suited for the coarse grain applications and making the bandwidth applications scalable in nature as shown in the Fig. 3.

REFERENCES

1. Nanette, J.B., R.E. Felderman, A.E. Kulawik, C.L. Seitz, J.N. Seizovic and W.K. Su, 1995. Myrinet-A Gigabit per sec Local Area Network, In IEEE-Micro, 15: 29-36.
2. Typou, T. *et al*., 2004. Implementing Matrix Multiplication on an MPI Cluster of workstations, In 1st IC-SCCE.
3. Geist, A., A. Beguelin and V. Sunderam, 1994. PVM Parallel Virtual Machine-A User's Guide and Tutorial for Networked Parallel Computing, MIT Press, Cambridge, MA.
4. Dave, MacDonald and W. Barkley, Microsoft Windows 2000 TCP/IP Implementation Details.