International Journal of Computer Theory and Engineering, Vol. 2, No. 2 April, 2010
1793-8201

A Cluster Based Parallel Computing Framework (CBPCF) for Performance Evaluation of Parallel Applications

Amit Chhabra, Gurvinder Singh

Abstract—**Quick and incremental growth in the processor power of desktop personal computers and network bandwidth due to recent extraordinary technological advances, have shifted the trend of parallel processing from conventional costly massively parallel supercomputers to the comparatively inexpensive cluster of networked desktop PCs for solving data and computation intensive sequential as well as parallel applications. For such parallel applications, cluster of LAN based networked PCs environment has become the boon in developing countries because of easy availability of relatively inexpensive computational resources. This paper presents a parallel computing framework based on cluster of networked desktop PCs that intends to optimally exploit the pooled computational strength of networked desktop PCs available in the intranet of university campus. This Cluster Based Parallel Computing framework (CBPCF) is based on the Master-Slave computing paradigm and it emulates the parallel computing environment. Performance statistics of such a cluster based framework is evaluated using experimental setup by running applications like parallel Matrix multiplication and Pi(∏)value approximation. Interpretation of results has shown that high bandwidth requirements in problems like matrix multiplication ,is a major hindrance to get good performance as major percentage of the turnaround time is consumed as communication time. In Contrary to matrix multiplication application, Pi approximation problem has shown good amount of speedup as well as efficiency due to more computation work involved than communication in the problem.**

Index Terms—Cluster, CBPCF, High Performance Computing, Master-Slave Computing paradigm.

I. Overview

Over the years different parallel computing structures have been proposed ranging from SIMD based array processors, MIMD based distributed memory multiprocessors with message passing and shared memory structures by Flynn's taxonomy[1] to Gordon bell's taxonomy[2] of MIMD based scalable distributed multi-computers or clusters. These scalable distributed multi-computers can be realized with the help of network based desktop PCs connected with the help of fast LAN.

In developing countries like India, LAN based networked desktop PCs environment has emerged as Poor man's Supercomputer for running computation intensive applications due to easy availability of the requisite resources. Customary million dollar High performance parallel computing platforms like supercomputers are being replaced by a few hundreds dollar fast LAN connected desktop PCs i.e. clusters. This alternative setup is popularly known as Network of Workstations (NOW)[3], Commercial off-the shelf computer clusters (COTS)[4], cluster farms, Multi-computer systems or distributed parallel computing systems. This can be said that trend of high performance computing is now tilted towards the cost-effective distributed memory MIMD multi-computer structure with message passing loosely coupled paradigm.

The rest of paper is organized as follows: In Section 2 design of proposed cluster based parallel computing framework is discussed in terms of computing paradigm and topology used. Section 3 discusses programming environment and socket connection establishment used in master PC program and slave PC program. In section 4 test bed for conducting the experiments is designed and results analysis is done based on the data collected from test bed. Performance of the CBPCF is measured by using metrics like speedup and efficiency. Section 5 deals with conclusion and suggested future directions.

II. Design Of Cluster Based Parallel Computing Framework

Formally a cluster[5] is defined as type of parallel or distributed systems consisting of set of independent desktop PCs interconnected by fast LAN cooperatively working together as a single integrated computing resource so as to provide higher availability, reliability and scalability

This paper presents the parallel computing framework based on cluster of Network desktop PCs. This Cluster Based Parallel Computing Framework (CBPCF) has been realized by the use of desktop PCs available in the intranet of university campus. The idea of using university campus intranet for the above said purpose derived from the fact that most of machines in the university campus and other similar type of organizations are lying idle during off days or non-working hours. Even during the working hours most of the users are only utilizing only about 5-10% of the available CPU power as they are running only low computation intensive applications like web browsing, text editing applications etc. Such an idle computational power can be effectively utilized by connecting available desktop PCs with the help of fast LAN, hence forming a Cluster based

Manuscript received on September 1, 2009.
Amit Chhabra is with department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, India.
Email: chhabra_amit78@yahoo.com.
Gurvinder Singh is with department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, India.
Email: gsbawa71@yahoo.com.

framework

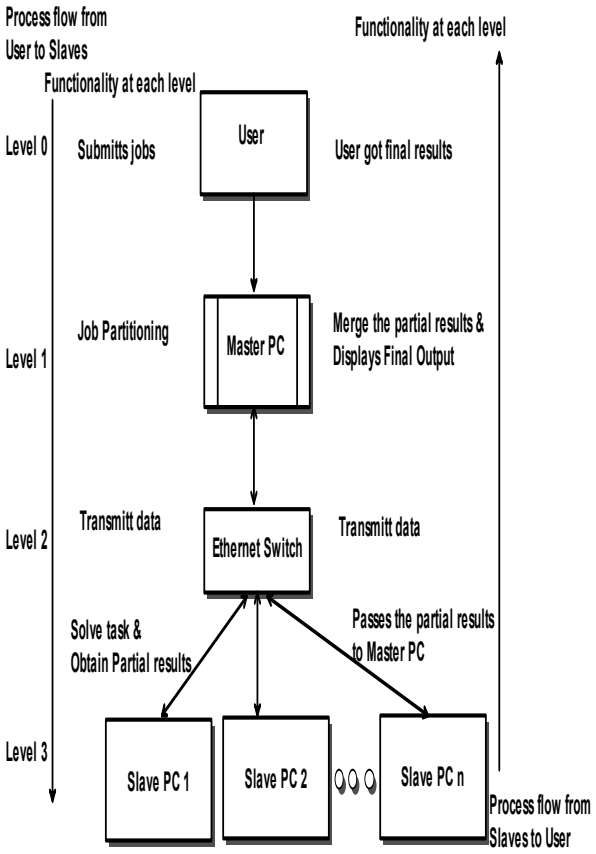

Figure 1: Showing computing paradigm, Topology and Functionality of the cluster based framework

This CBPCF will be beneficial for all the young researchers in the financially constrained small universities or other technical organizations where research in the area of high performance parallel computing is at a standstill due to lack of availability of requisite massively parallel supercomputer. It will also set the stage for them to pursue their research by providing them cost-effective solution for high performance computing.

In general a cluster[5] based setup has the following advantages in comparison with traditional multiprocessor systems;

1) Relatively inexpensive as compared to supercomputers due to easily availability of cheaper desktop machines connected through high speed LAN.
2) Easy scalability; one can easily add the number of contributing machines to cluster.
3) It is very easy to keep the cluster updated by simply replacing outdated desktop PCs with new latest ones. So there is no need to discard the whole cluster.
4) Better price/performance ratio as compared to multiprocessors systems.

The purposed CBPCF is going to use the cluster that inherits the above mentioned benefits of clusters.

A. Underlying Master-Slave Computing Paradigm, Topology and Functionality at each level

CBPCF is going to use the popular Master-Slave or Master-Worker computing paradigm[5][6] as shown in figure 1 and it will be using star network topology to connect Master PC with all the slaves using 10/100 Mbps Ethernet Switch and Cat-5 cable.

Key physical and logical components of this cluster based framework are

1) User-It submits the jobs to the Master PC.
2) Master PC –It partitions the jobs on the basis of data parallelism (domain decomposition) or control parallelism (functional decomposition) and passes these sub-jobs to the available slave PCs. It also gathers the partial results sent by Slave PCs and merges them to given final result.
3) Slave PC-It performs the computation work on the task given by Master and sends the result back to Master PC.

Role of these three main components in terms of their functionality is divided among two categories

Category I. When process flow takes place from user at Level 0 to slave PCs at Level 3 roles played by the three prime physical components are

- User-It submits the jobs to the Master PC.
- Master PC –It partitions the jobs on the basis of data parallelism or control parallelism and passes these sub-jobs to the available slave PCs.
- Slave PC-It computes the task given by Master and sends the result back to Master PC.

Category II. When process flow takes place from Slave PCs sitting at Level 3 to user at Level 0 roles played by three prime components are

- Slave PC- After performing the task given by master PC, it sends the partial results back to the Master.
- Master PC- It collects the partial results from all the slave PCs involved in solving a particular job and merges the partial results to obtain final output.
- User- user gets the final output from Master PC.

In this cluster based framework Master PC is acting as Administrator or Control manager of the cluster as it is managing and controlling the major activities (busy right from the arrival of job, partitioning the job, passing the partitioned sub jobs to slaves, and later on merging the partial results and displaying the output). Master PC gives the illusion to the user as if it is working on a massively parallel supercomputer by providing it a single integrated computing resource image.

B. A typical Cluster Architecture

Typical traditional cluster architecture[5][6] is being followed in the proposed cluster based parallel computing framework as shown below in figure 2 with few changes as message passing mechanism using socket programming in Microsoft Visual Basic 6.0 is developed and used instead of standard message passing scheme using MPI/OMP or LAM-MPI. The features used in CBPCF at every level are indicated by dark blue thick line. This middleware message passing mechanism using sockets is in fact clubbed with the

application running at application layer.

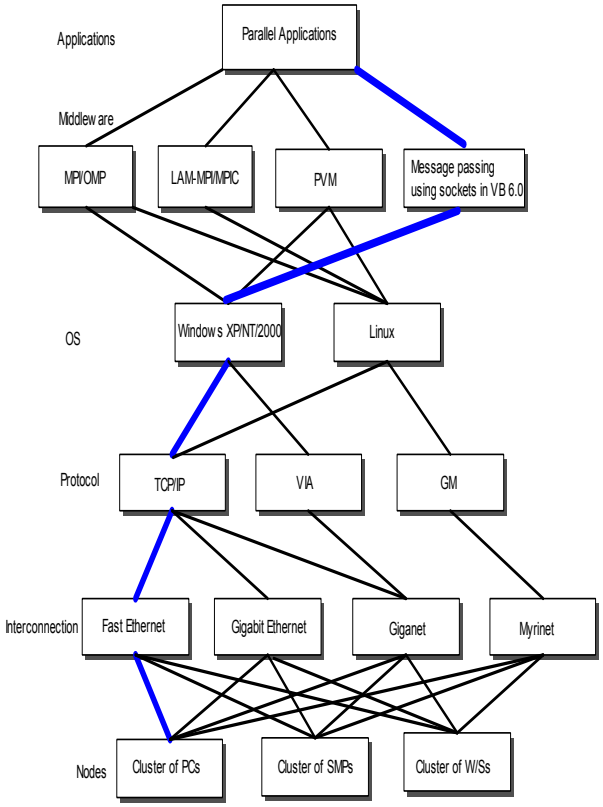

Figure 2: Physical architecture of the proposed cluster based parallel computing framework(CBPCF)

III. Programming Environment And Developer Interface Of The Cluster Based Framework

A. Programming environment for Master and Slave

The reason for adopting Microsoft visual basic 6.0 as a programming language for the proposed cluster based framework is that it provides a strong GUI and in built interactive IDE environment which is most suited to programmers for creating graphical objects as compared to command based programming languages like c or c++ where one has to do lot of effort to build graphics based applications. Another prime reason for its usage here is that it provides an easy way to use Windows socket APIs directly by incorporating Winsock control available in the language. Time consuming effort for creating sockets using long lines of code has been cut down to few lines of code in this language.

B. Socket connection establishment between Master and Slave

Network communications with the Ethernet devices is accomplished using a TCP/IP "socket" connection. A "socket" is a logical communication stream specifying the two end points of the connection. Each end point Ethernet device has a unique IP address and unreserved port number. A port is a special memory location that exists when two computers are in communication via TCP/IP. Applications use a port number as an identifier to other computers. Both the sending and receiving computers use this port to exchange data.

Socket communication[7] is the basis for the transfer of data and control messages between Master and Slaves. Data transmission or Communication between master and slaves is established and maintained by creating TCP/IP socket connection between Master PC and Slave PCs using Microsoft Window's Winsock (MSWINSCK.OCX) control available in Microsoft visual basic 6.0 language. This Winsock control is going to use .Dll file of Microsoft windows XP. Microsoft has wrapped up the Winsock and NETAPI API calls into a neat package that one can easily incorporate into their Visual Basic applications.

Winsock control enables you to create clients (slave) and servers (master) using the same control. This dual functionality enables you to specify through property setting the type of application you will be building. The Winsock control provides properties, methods and events that allow a connection to be established and maintained throughout for network communications with the end point device. When creating an application that uses the TCP protocol, one must first decide if your application will be a master (server) or a slave (client). Each of the slaves is provided IP address and specified unreserved port number.

Master PC is also identified by its Hostname or IP address. Creating a Master means that your application will "listen" on a designated port assigned to a particular slave. Whenever the slave makes a connection request attempting for connection with the Master Hostname, the master identifies that slave using its port number and then it can accept the request and thereby complete the connection. Once the connection is complete, the Master and slave are free to communicate with each other. Using Connect Method of Winsock (i.e.Winsock.connect), master and slaves get connected with each other. Master side socket has a request handler or a listener for every port assigned to each of the slaves and it listens to the requests originated by slaves for establishing the connection. This step by step socket connection establishment is shown in figure 3 from step 1 to step 5.

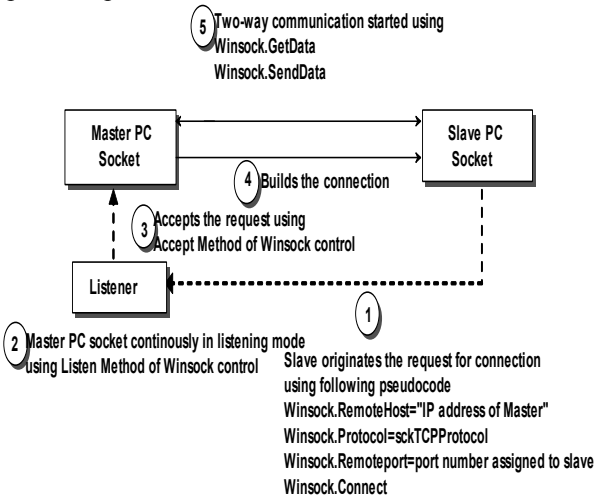

Figure 3: Socket connection establishment and data transfer procedure

C. GUI interface for Master

Master interface is basically a resource manager of the CBPCF as it provides the computing resources (slaves) to the jobs. Master also monitors the activities of all the connected slaves like checking the status of slaves whether

they are free or busy other than performing its usual tasks like partitioning the workload, sending the partitioned workload to slaves and collecting results back from slaves. Socket connection establishment and data transfer between master and slave is done using the procedure as discussed in Section B.GUI interface for the master program is shown in figure 4.

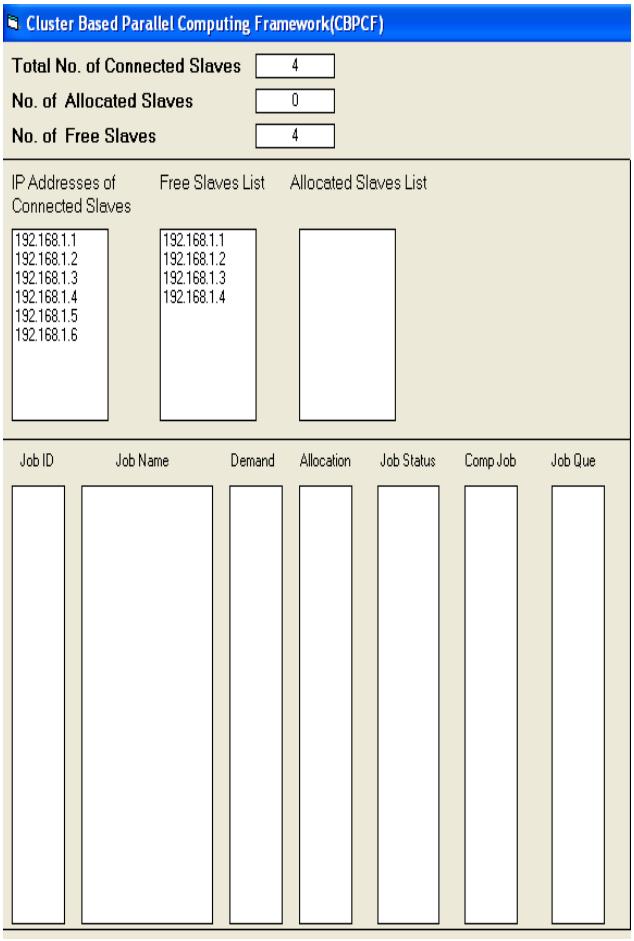

Figure 4: Developer interface for the master

Developer interface of the master displays the status of slaves as well as jobs.

IV. Experimental Setup And Performance Measurement

For the sake of running parallel applications on the cluster based parallel computing framework (CBPCF), a set of seven homogenous machines has been taken as a necessary hardware in the experimental setup. One out of them is acting as Master PC and others are designated as Slave PCs. In the proposed cluster based framework, master PC distributes the partitioned workload by copying it into slave PCs hard-disk. Other network communications i.e. control messages between master and slaves takes place using socket message passing mechanism.

Physical description of cluster configuration in terms of hardware and software components used is shown in Table 1.

Parallel applications named Matrix multiplication and Calculation of value of Pi are used as benchmarks for the performance evaluation of proposed CBPCF.

Table 1: Cluster configuration

Component Name	No.	Configuration	OS Support	Language
Master PC	01	• CPU Speed 3.4 Ghz • RAM 1 GB • HDD 80 GB • LAN card Realtek • IP address 192.168.1.1	Windows XP SP2	Visual Basic 6.0
Slave PC_1 to Slave PC_6	06	• CPU Speed 3.4 Ghz • RAM 1 GB • HDD 80 GB • LAN card Realtek • IP address(s) 192.168.1.2-192.168.1.7	Windows XP SP2	Visual Basic 6.0
Ethernet Switch	01	• 24 Port 10/100 Mbps Ethernet switch (Model ARL-524F Make HCL)	-	-
Cat 5 UTP cable	-	• Cat 5 UTP cable supports transmission upto the rate of 100MB/s	-	-

Experiment 1: Case study of Matrix Multiplication application

In this experiment, two square matrices A and B of different sizes 128x128, 256x256 and 512x512 are chosen for multiplication. Elementary n^3 matrix multiplication sequential algorithm is implemented on any one slave to get sequential time t_s.

For parallel matrix multiplication a parallel program is developed using Visual Basic 6.0 in which a master PC program partitions the matrix A into equal-sized rows according to the number of available slaves. It then passes this partitioned equal workload i.e. sub-matrix A of certain rows and entire matrix B to the available slaves. Master PC takes care of load balancing by passing equal amount of workload to every slave. The slaves computes the matrix multiplication task at their ends and passes the computed results back to the master PC which merges all these partial results to obtain final matrix C. This parallel matrix multiplication implementation follows SPMD scheme with the same n^3 elementary sequential matrix multiplication algorithm running simultaneously on all the slaves but with different data sets.

Experiment 2: Case study of Pi(∏) value approximation application

In serial version of the problem, Pi is calculated by drawing a circle inside the square and then points are randomly generating in that square. The distinction is made between number of points that are inside the circle as well as inside the square. Ratio of points inside the circle to the total number of points in the square (*npoints*) is obtained. The value of Pi is obtained by multiplying this ratio by 4.This

requires a loop to be executed ranging from 1 to *npoints* and within this loop, points are randomly generated and points belonging to circle as well as square are calculated. As more and more *npoints* are generated, better approximation of Pi can be obtained.

In parallel strategy, this loop of *npoints* is broken into equal portions depending upon the number of available slaves. This is done by dividing *npoints* by the number of available slaves. Each of the slaves is now responsible for executing its own portion of the loop for a specified number of times and calculating the circle points. All the slaves will pass their count of calculated circle points to master and which in turn sum up all the circle points, divides this sum by total number of points in square(*npoints*) and later on multiplying this ratio by 4 to obtain final Pi approximation.

Performance Metrics

For the sake of evaluating performance of the proposed cluster based framework, two quantitative performance metrics speedup and efficiency have been chosen.

Speedup (S)-It is a measure of relative performance between a multiprocessor system and a uniprocessor system and is defined as

$$Speedup(S) = \frac{\text{Execution time using one processor}(t_s)}{\text{Execution time using p processors}(t_p)}$$

Time ts is measured by running the applications on any one processor from the cluster system.

Time t_p is measured by running the applications on p processors available in the cluster based framework. Time t_p includes the time taken is measured from the moment the job is being submitted to the master to the moment the master collects back results and merges them to form final result. This time t_p includes the time taken by master to partition the job into sub-jobs, time taken for sending the sub-jobs to slaves for computation work, time taken by slaves for computing sub-jobs, time taken by slaves to send the partial results back to master and finally time taken by master to merge the partial results to form the final result.

Efficiency (E)-It is the measure of utilization of the processors for sake of computation. In other words it gives the fraction of the time that the processors are being utilized in computation work. It is defined as

$$Efficiency(E) = \frac{\text{Execution time using one processor}(t_s)}{\text{Execution time using p processors}(t_p) \times \text{No.of processors}(p)} = \frac{S}{p}$$

Value of efficiency(E) will always lies between 0 and 1.

Results for the Matrix Multiplication application when run on different number of slaves are as shown below in Table2.

Table 2: Matrix Multiplication Completion time results

Time(in seconds)						
No. of Slaves	1	2	3	4	5	6
Matrix size 128x128	0.762	0.646	0.492	0.584	0.552	0.576
Matrix size 256x256	6.273	4.940	3.372	2.494	2.266	1.981
Matrix size 512x512	47.723	35.705	24.249	19.420	15.346	14.564

Speed and Efficiency results obtained from the above timings are shown below in Table 3.

Table 3: Matrix Multiplication Speedup and Efficiency results

No. of Slaves	S (128x128)	E (128x128)	S (256x256)	E (256x256)	S (512x512)	E (512x512)
1	-	-	-	-	-	-
2	1.18	0.54	1.27	0.63	1.34	0.67
3	1.55	0.52	1.86	0.62	1.96	0.65
4	1.31	0.33	2.52	0.63	2.46	0.62
5	1.38	0.27	2.77	0.55	3.11	0.62
6	1.32	0.22	3.16	0.52	3.27	0.54

Graphical representation of Matrix Multiplication speedup results is shown below in Figure 5.

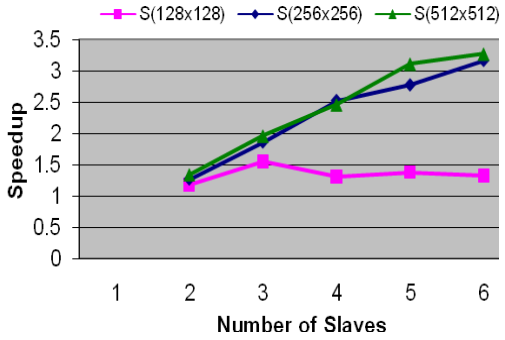

Figure 5: Matrix Multiplication Speedup

Graphical representation of Matrix Multiplication efficiency results is shown below in Figure 6.

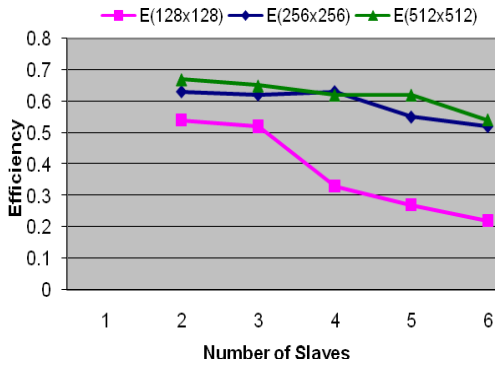

Figure 6: Matrix Multiplication Efficiency

Results for the Pi value Calculation application when run on different number of slaves are as shown below in Table 4.

Table 4: Pi calculation results

Time(in milliseconds)						
No. of Slaves	1	2	3	4	5	6
Calculating Pi value						
100,000 points	95.56	56.22	42.16	34.26	30.62	28.61
10,00,000 points	938.42	477.04	322.17	246.27	199.19	179.07
100,00,000 points	9310.79	4740.20	3152.77	2355.27	1899.21	1586.13

Table 5: Pi calculation Speedup and Efficiency results

No. of Slaves	S(100,000)	E(100,000)	S(10,00,000)	E(10,00,000)	S(100,00,000)	E(100,00,000)
1	-	-	-	-	-	-
2	1.69	0.84	1.96	0.98	1.96	0.98
3	2.26	0.75	2.91	0.97	2.95	0.98
4	2.78	0.69	3.81	0.95	3.95	0.98
5	3.12	0.62	4.71	0.94	4.90	0.98
6	3.34	0.55	5.24	0.87	5.87	0.97

Speed and Efficiency results obtained from the above timings are shown below in Table 5.

Graphical representation of Pi calculation speedup results is shown below in figure 7.

Figure 7: Pi calculation Speedup

Graphical representation of Pi calculation efficiency results is shown below in figure 8.

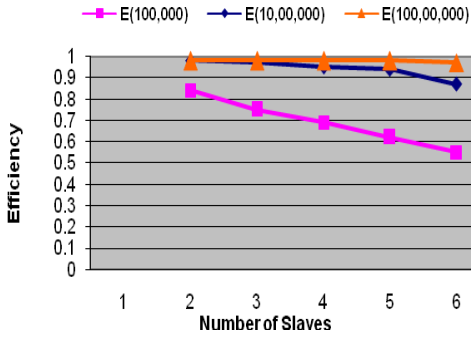

Figure 8: Pi calculation efficiency

A. Interpretation of results

Table 3 shows the speedup and efficiency results obtained from parallel matrix multiplication application. In the case of matrix sizes of 128x128, not much speedup is obtained by the use of multiple slaves because of the low computation requirement as the size of matrices used is not very large. As we move towards matrix sizes 256x256 and 512x512 speedup results shows the improvement. Efficiency in 128x128, 256x256 and 512x512 is not impressive due to low computation and high bandwidth requirement in all the three cases. Though it does improve in matrix sizes of 256x256 and 512x512.In terms of bandwidth, data to be transferred in all the three cases is of the order of few Mbits.

Network traffic will become performance bottleneck as more and more slaves are engaged in computation or when the problem size is scaled.

Table 5 shows the speedup and efficiency achieved while running parallel Pi calculation application with 100000(Case1), 1000000(Case2) and 10000000(Case3) number of points. Speedup increases in every case as the number of slaves goes up. In the case 1 with 100,000 points, efficiency decreases while increasing the number of slaves. This is due to low computation requirement in this case and also problem does not get scaled with the increase in the number of slaves. As soon as we scaled up the problem (in case 2 and case 3) by increasing the number of points to be calculated to have a Pi value, a drastic change in the efficiency results can be seen in the Table 5.This is due to the fact that in these cases problem i.e. workload is also scaled with the increase the number of slaves.

V. CONCLUSION & FUTURE DIRECTIONS

This paper discusses the use of cluster based parallel computing framework(CBPCF) for high-performance computing as a cost effective and attractive alternative to traditional multiprocessors. Such a cluster based framework is utilizing commodity based hardware and software to achieve higher performance, availability, scalability and reliability. This CBPCF inherits the advantages of typical clusters. CBPCF provides easy to use GUI computing environment. In this paper a homogeneous (all slaves are of same configuration) set of machines have been used in the

experimental setup. In future set of machines having different configurations will be chosen to observe the effect of heterogeneity on the performance of the proposed cluster based parallel computing framework.

Though cluster based framework has got many advantages over traditional multiprocessors systems, they do posses certain drawbacks. Network hardware i.e. LAN used in cluster computing is not specifically designed for parallel processing. Also there is very little software support available for treating cluster based framework as a single system. Present work can be extended in future to provide more generalized middleware software support.

REFERENCES

[1] M.J.Flynn, "Some computer organizations and their effectiveness," IEEE transactions on computers, 21(9):948-960, 1972.
[2] G.Bell, "Ultracomputer: A teraflop before its time," communications of ACM, 35(8):27-47, 1992.
[3] T. E. Anderson, D.E.Culler and D.A.Patterson "A case for NOW (Networks of Workstations)", IEEE Micro, Feb 1995.
[4] Andreas Boklund, Stefan Mankefors-Christiernin, Christian Jiresjö, Nima Namaki, "COTS-Cluster Evolution during the Last Decade and Extrapolation into the Next", Proceedings of the IASTED International Conference on Parallel and Distributed Computing and Networks, part of the 23rd Multi-Conference on Applied Informatics, Innsbruck, Austria, February 15-17, 2005, 614-619.
[5] R.Buyya, High Performance Cluster Computing: Systems and Architectures.Vol. 1,1/e, Prentice Hall PTR, NJ, May 1999.
[6] R.Buyya, High Performance Cluster Computing: Systems and Architectures.Vol. 2,1/e, Prentice Hall PTR, NJ, June 1999.
[7] http://msdn.microsoft.com/en-us/library/aa733709.aspx

Amit Chhabra has received B.Tech(Computer Science & Engineering) and M.Tech(IT) from Guru Nanak Dev University, Amritsar Author is working as a Lecturer in the Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar. His research areas include Distributed Systems and Parallel Computing.

Gurvinder Singh has received MCA and Ph.D from Guru Nanak Dev University, Amritsar. Author is presently working as an Associate Professor in the Department of Computer Science & Engineering, Guru Nanak Dev University, Amritsar. His research areas include Distributed Systems, Parallel Computing and Grid Computing.