Analytical Parallel Approach to Evaluate Cluster based Strassen's Matrix Multiplication

Nidhi Pasricha
M.Tech(CSE)
LLRIET, Moga.
Asst. Prof, LLRIET, Moga

Ankit Arora
M.Tech (IT)
Guru Nanak Dev University Asr.
Asst. Prof, LLRIET, Moga.

Rajbir Singh Cheema
M.Tech(CSE)
Guru Nanak Dev Eng. College Ldh.
Associate Prof, LLRIET, Moga.

ABSTRACT
Today current era of scientific computing and computational theory involves high exhaustive data computation, shifted the trend of data processing from conventional processing towards parallel processing by incorporating multiple processing hardware. Parallel hardware design can employ array processors, pipelined system which can be further extended to scalar and super scalar pipelined systems. Other hardware designs proposed, is based upon multiprocessors or they may be designed as distributed parallel cluster systems. In this paper, multi-computers are the basic hardware for cluster design over the local area network covering analysis of matrix multiplication with strassen's algorithm. The estimated results are then compared with traditional matrix multiplication algorithm. Strassen's multiplication approach reduces one multiplication out of eight by computing arithmetic additions/subtractions for each 2×2 matrix. High performance can be achieved as the idea is extended over to multi-computer cluster for large sized matrices. This work covers analysis of Strassen's ability of divide and conquers[5] to run in parallel by decomposing matrix size over cluster machines covering data parallel aspects with SIMD based computational model [4], where each cluster machine performs its own recursive divide and conquer approach as defined by strassen's methodology[9][10] to obtain partitioned matrix multiplication. Finally, the detailed distributed experiment along with connectivity interface and implementation will be discussed.

General Terms
Distributed Clustering, Strassen's algorithm, Divide and Conquer, Workload Partitioning and Distribution.

Keywords
Client-Server, TCP/IP Sockets, Matrix Multiplication, Data Parallel aspects, Space Sharing Policies.

1. INTRODUCTION
Today, there is a great trend of multi-processor architecture but most of the experimentation studies incorporate massive cluster computing as they provides flexibility over logical design of distributed machine. Distributed parallel Cluster consisting of set of independent clients modeled as Local Area Network cooperatively working together as a single integrated computational resource. The Idea behind this to provide higher accessibility, considering a industry scenario where large quantity of personal computers are discarded not because of hardware failure but because of their less performance as compare to modern scientific ability. Such machines can be organized as clustered groupware to achieve parallel effects. Reliability is the another major factor behind the design of cluster framework as load balancing issues can be resolved if one having extra load then other may share its load, such machines generally follows the rules of SMA (shared memory access). Scalability is the another major demand which generally not fulfilled by real multi-processor systems, theses machine designed to adapt fixed no. of processor chips, even such machines if modified requires extensive amount of hardwired circuitry to be changed. As compared to network clustering the nodes can be increased or decreased as per the requirements of the applications. Distributed processing provides a way of getting topological benefits by incorporating any of the cluster design, heterogeneity can be adapted if applications having different parallelism loads. So applications having no. of parallel threads each of which having less complex computational logic will be scheduled to low speed clusters, where as applications having large no. of parallel threads having complex computational and data intensive work will be scheduled to high speed cluster nodes. Scheduling transparencies can be achieved, only the controller knows which node having which sub-task and computation logic if the cluster is designed to compute multiple algorithmic logics for different class of problems [12]. In this experiment cluster behaves like client-server architectural model where server distribute the job to their clients via TCP/IP sockets [13][14]. Each client follows Strassen's algorithm [9][10] which is based on divide & conquer paradigm. The paradigm follows the space sharing policy [3], where no. of processors are allocated to different parts of single active job. Job is partitioned among various sub-tasks and each task is assigned to individual cluster node, after then cluster computes its designated computation and sends result back to the controller where the final consolidated result will be obtained as shown in the fig 1 where 5 steps methodology will be discussed. In the further article the research towards Stassen's algorithm [9][10]which divides the square matrices using n/2 fractions and divides till size n>2 and starts performing 7 multiplication and 18 additions for each pair of 2×2 matrix. Thus helps in reducing complexity time from traditional multiplication $O(n^3)$ to $O(n^{2.81})$ when run recursively over uni-processor architecture, as it reduces 8 multiplications to 7 and total of 18 arithmetic additions and subtraction. The detailed working of algorithm will be discussed in further algorithmic approach section. Time bounds above discussed is based upon sequential execution of both traditional approach and strassen's divide and conquer approach [5]. This work is extended to compute strassen's time complexity analysis over clustered operations parallely.

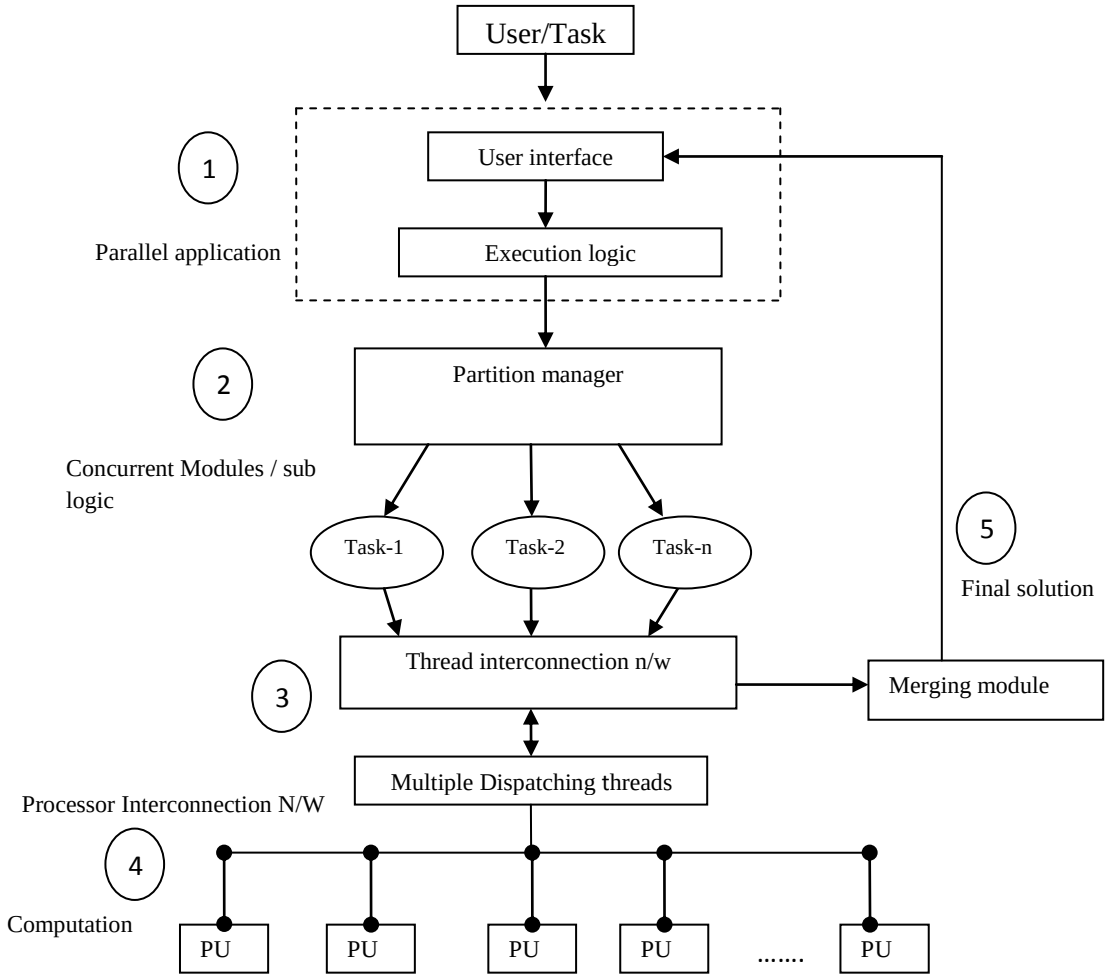

Fig 1: Space sharing policy

1. Parallel Application having capacity to run in parallel fashion.
2. Partition manager segmented the application logic to obtain concurrent modules/sub-tasks.
3. Thread Interconnection network required to share information about sub-task completion as well as thread synchronization.
4. Dispatching Thread Allocates a particular sub-task to a particular available/ free processor.
5. Merging module required to combine all of the sub-task solutions.

2. LITERATURE REVIEW

Previous research discusses the use of cluster based parallel computing framework (CBPCF) [3] for high-performance computing as a cost effective and attractive alternative for multiprocessors to analyze the matrix multiplication with traditional n^3 algorithm over a client-server architecture, where server distribute the job to their clients and each clients execute the job sequentially. This approach provides greater utilization of parallelism, as job executed on cluster nodes. Other research literature covering cluster operations implements image compression using run-length encoding techniques [1]. The performance of matrix multiplication can also be further optimized. This research paper involves improving further performance of matrix multiplication by strassen's algorithm [9][10]in cluster base system.

3. ALGORITHMIC APPROACH

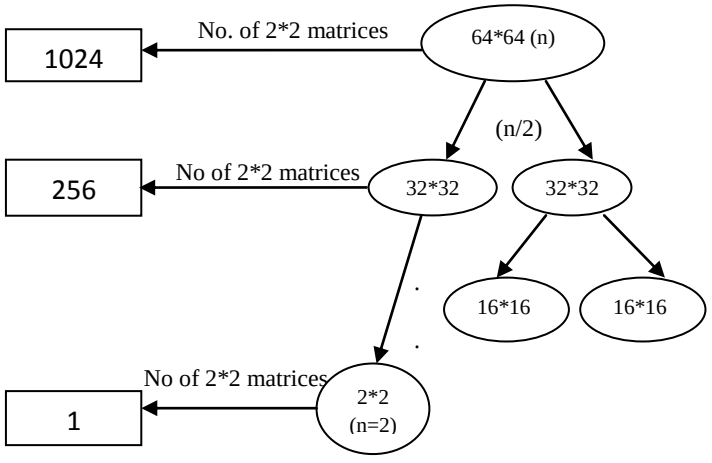

Fig 2: Showing algorithmic approach

Strassen's algorithm [9][10] is faster and involves less multiplications than traditional matrix multiplication algorithm. As described in earlier section, algorithm divides the square matrices using n/2 fractions and divides till size n>2. At reaching n=2, the partitioning stops and starts performing 7 multiplication and 18 additions for each pair of 2×2 matrix. Each cluster node receives a portion of the matrix then it firstly divides the matrix into 2's fractions and divides till it reaches into 2×2 matrices. So 64×64 matrices is divided into 32×32 matrices and each of them then divides into 16×16 and so on as shown in fig-2. Upon reaching 2×2 it follows equations described by the strasen's algorithm as-.

$$P = (A_{11} + A_{22)*(} B_{11+}B_{22)}$$

$$Q = (A_{21} + A_{22)*} B_{11}$$

$$R = A_{11} * (B_{12-} B_{22)}$$

$$S = A_{22} * (B_{21-} B_{11)}$$

$$T = (A_{11} + A_{12)*}B_{22}$$

$$U = (A_{21} - A_{11)*(} B_{11+}B_{12)}$$

$$V = (A_{12} - A_{22)*(} B_{21+}B_{22)}$$

$$C_{11} = P + S - T + V$$

$$C_{12} = R + T$$

$$C_{21} = Q + S$$

$$C_{22} = P + R - Q + U$$

4. MODELING PARADIGM

Modeling paradigm consists of two execution layers one is Hardware and another one is Software layer. Hardware execution layer contain processor and memory unit for processing and storage purpose. Software layer contain logical programmed module consisting master core program controlling logical program execution via partitioning and merging modules. Partition and distribution logic generates the sub-tasks and distributes over to the cluster nodes. Merging modules waits and consolidates final cluster outcomes after finishing cluster computation (see Figure 3). Further the Hardware configuration for the cluster framework is, Intel Pentium Dual-Core CPU with 3.2 GHz processor and 1 GB Ram, Network Ethernet switch configured with star topology and distributed software implementation with Microsoft Visual Basic 6.0 [7], Socket communication connections are implemented via Microsoft Winsock Control (Mswinsck.ocx). Following is the general code of Winsock listener [8].

```
Winsock1.Protocol = sckTCPProtocol
Winsock1.LocalPort = 1101
Winsock1.Listen
Winsock1.Accept
Winsock1.SendData Message
```

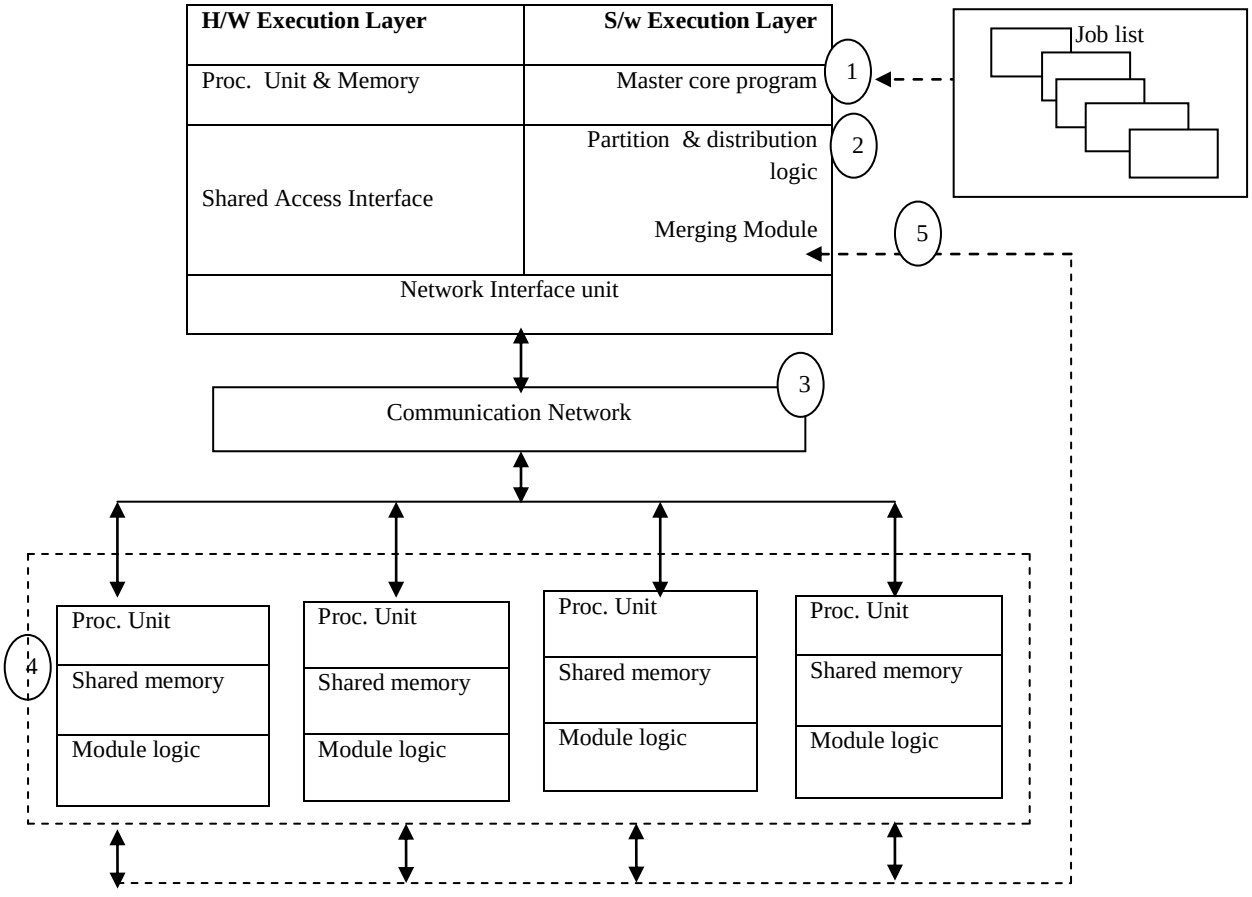

Fig 3: Modeling paradigm

5. CLUSTER DESIGN

Logical Distributed parallel design consisting master controller as a basic core part performs workload partitioning [6] and is also responsible for final result consolidation from individual cluster nodes. The master module also covers the job generation module, measurements of timing variations describing total time, partitioning and workload transmission time, merging and allocation time. This module during initialization establishes the interconnection via Microsoft Winsock control[8] to each of the intended cluster clients, then it partitions the workload, perform distribution via shared

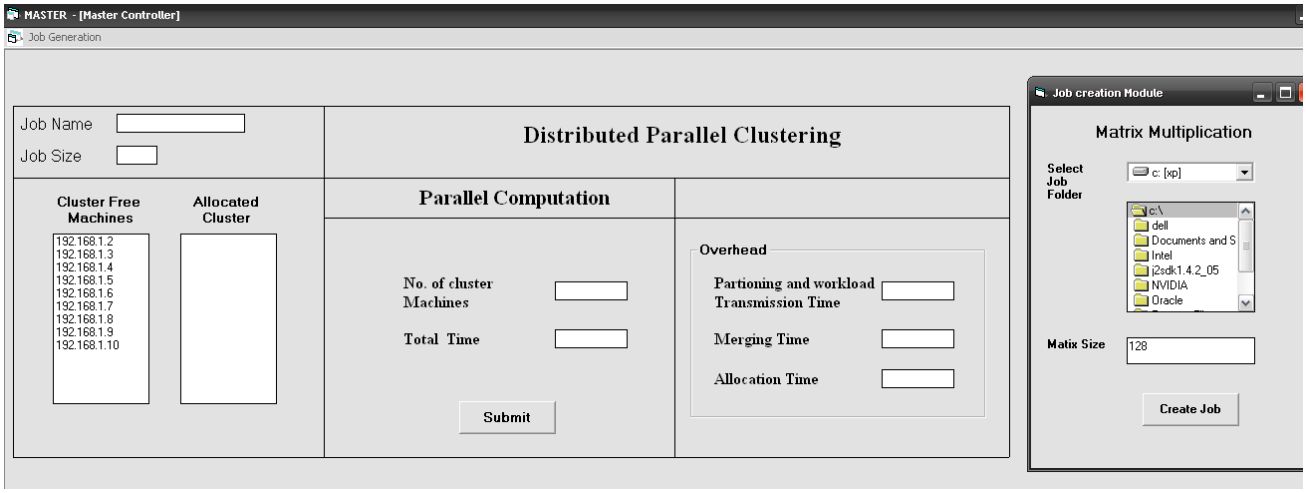

Fig 4: Cluster Design

Memory interconnection [11] and finally merging thread computes the final consolidation results. Message Passing communication is performed to send signals to the cluster nodes ensuring that their sub-task are now ready for computation. Job generation module handles the creation of matrices with desired sizes, the matrices are stored under text files which are partitioned and distributed over cluster nodes via TCP/IP file sharing service. Further the analysis results covers the comparisons considering total time consumed from job submission to till its final result covering each and every aspect of timing consumed regarding workload partitioning [6] and merging. Client side module is basically a task receiver performing desired computation and finally sends results back to master controller via share memory interface. Each cluster node uses message passing communication to send control message to the master regarding their sub-task completion

6. PERFORMANCE MEASUREMENTS

Various Timing parameters that help to measure the performance of cluster framework are: Total time (t), Partitioning and Input Work load Transmission time(pw), Merging time(m), Allocation time(a), Computation & Output File Transmission time(c). *Total Time (t)* is time consumed from job submissions to till its final completion. Allocation Time is the time elapsed while transmitting message to ensure each cluster node that their sub-tasks are ready for computation.

$$t = pw + m + a + c$$

Partitioning and workload transmission time(Pw) shows the time elapsed at master to partition the job and the time to transmit the work load over the network.

Allocation time(a) is the time which is elapsed in during transmission of control signal from master to cluster nodes about ensuring the job workload has been transferred to their shared memory and now ready to compute.

.

Computation & Output File transmission time(c) is the time taken by cluster clients to execute the job as well as finally send output file over the network, it is the combined time of all allocated cluster nodes. *Pw* is the time used to generate sub tasks and their transmission over the network.

$$c = t - (pw + a + m)$$

Merging time(m) is the time taken by master to merge the workload results from the cluster clients to final outcome.

6.1 Performance Measurements

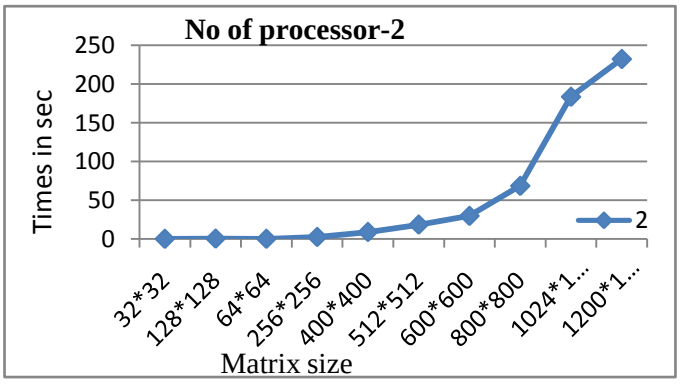

Fig 5: Cluster Timing with size-2

Table 1: Matrix Multiplication Timing Variations

Time in Sec						
No. of cluster Nodes	Size of matrix (n*n)	Total Time	Partitioning & Input workload Transmission Time	Merging Time	Alloc. Time	Comp. & Output File Transmission. Time
2	32*32	0.042	0.020	0.0007	0.0004	0.020
2	64*64	0.085	0.022	0.0013	0.0004	0.061
2	128*128	0.338	0.027	0.003	0.0004	0.0029
2	256*256	2.447	0.073	0.013	0.0004	2.359
2	400*400	8.878	0.15	0.037	0.0005	8.687
2	512*512	18.42	0.237	0.063	0.0004	18.124
2	600*600	29.66	0.417	0.090	0.0004	29.246
2	800*800	68.42	0.573	0.573	0.0004	67.634
2	1024*1024	183.45	1.110	0.600	0.0004	18173
2	1200*1200	232.20	1.239	0.677	0.0004	230.283
3	600*600	20.16	0.319	0.090	0..0004	19.657
3	900*900	25.96	2.180	0.287	0.0006	123.497
3	1200*1200	156.20	1.613	0.600	0.0004	153.956
4	32*32	0.059	0.039	0.0008	0.0004	0.018
4	64*64	0.090	0.044	0.0015	0.0004	0.044
4	128*128	0.232	0.049	0.003	0.0004	0.0017
4	256*256	1.369	0.123	0.139	0.0004	1.232
4	400*400	4.806	0.154	0.036	0.0005	4.517
4	512*512	9.818	0.237	0.063	0.0004	9.360
4	800*800	35.90	0.573	0.219	0.0004	28.542
4	1024*1024	74.11	1.452	0.389	0.0004	72.268
4	1200*1200	119.1	1.972	0.677	0.0004	116.531
6	600*600	10.90	0.708	0.090	0.0006	10.104
6	1200*1200	81.65	2.704	0.605	0.0005	78.345
8	32*32	0.099	0.077	0.0009	0.0005	0.020
8	64*64	0.78	0.087	0.0015	0.0005	0.691
8	256*256	1.57	0.223	0.014	0.0005	1.132
8	400*400	3.31	0.444	0.039	0.0005	2.833
8	512*512	5.930	0.683	0.061	0.0005	5.805
8	800*800	20.07	1.560	0.229	0.0005	18.287
8	1024*1024	40.47	2.502	0.389	0.0006	37.580
9	900*900	66.681	0.906	0.287	0.0004	65.486

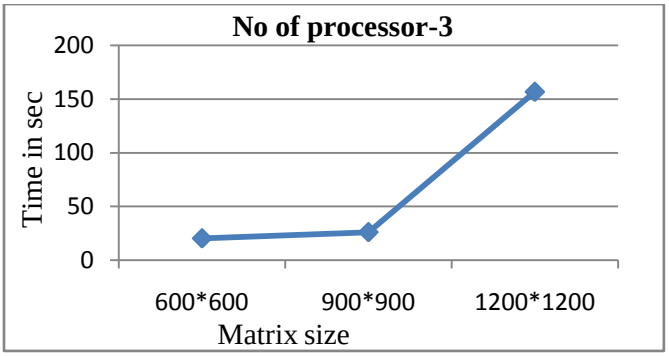

Fig 6: Cluster Timing with size -3

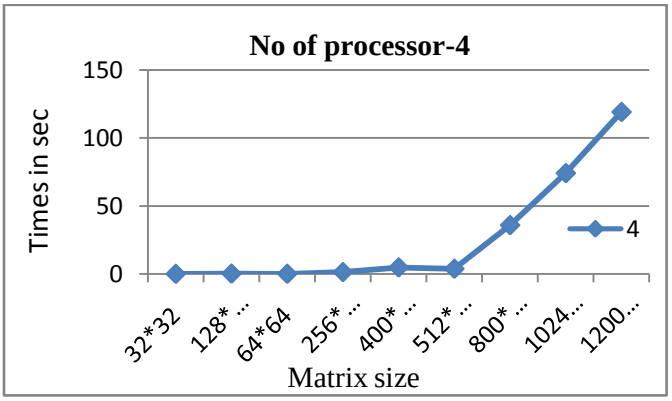

Fig 7: Cluster Timing with size -4

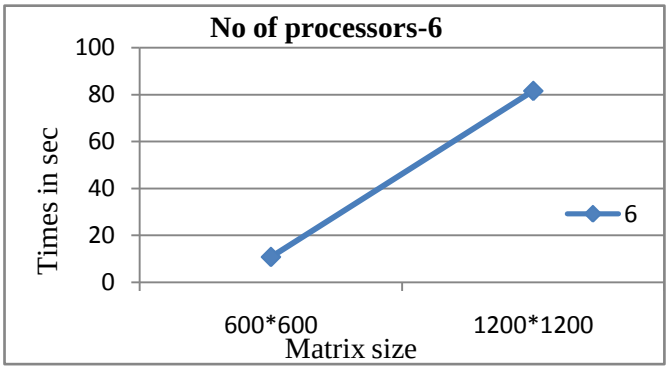

Fig 8: Cluster Timing with size -6

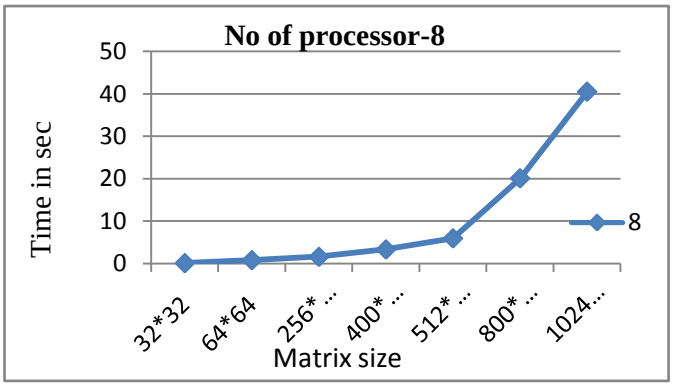

Fig 9: Cluster Timing with size -8

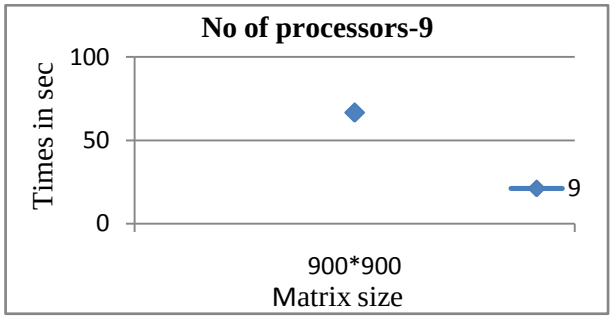

Fig 10: Cluster Timing with size -9

7. COMPARATIVE ANALYSIS

Result based upon the strassen's execution algorithm over cluster environment proved to be very effective than the previous implemented traditional matrix multiplication algorithm covered in research article "A Cluster Based Parallel Computing Framework for Performance Evaluation of Parallel Applications" published International Journal Computer Theory and Engineering, Vol.2, No.2 April, 2010. [2] As the analysis shows Total completion time for the computation of 128*128 Matrix with 2 processor is 0.338 sec where as with traditional algorithm the Total time is 0.646 sec. Similarly other matrices with their timing comparison results are shown in the following table

Table 2: Matrix Comparison Result

Matrix size	Strassen's divide & conquer algorithm		Traditional matrix multiplication algorithm	
	2	4	2	4
128*128	0.338s	0.232s	0.646s	0.584s
256*256	2.447s	1.369s	4.940s	2.494s
512*512	18.42s	9.818s	35.705s	19.420s

8. CONCLUSION & FUTURE WORK

Cluster framework used in this research is configured via 10/100mbps Ethernet card with cat-5 twisted pair cabling. The Analysis predicted during running cluster execution describes that as the cluster nodes increases the timing consumed during workload transmission and final outcomes transmission from cluster nodes is growing exponentially. This is because some times the cluster clients simultaneously accesses master's shared memory, also during workload transmission from master to cluster clients takes more time because the architecture has single medium attached with master to Ethernet switch, so that medium will be overloaded and gives response delay. Further this extra time is incorporated inside the total time estimated above. The architecture can be improved if the master computer having multiple Ethernet card describing n-computing structure. Now the workload can be transmitted via parallel lines. In this case the nodes can be organized as a cluster group with one particular Ethernet card linked over the master, this will lead to the sub-cluster generation. Also the performance will be increased. Future work will covers the implementation analysis over this cluster architecture.

9. REFERENCES

[1] Ankit Arora, Amit Chhabra 2011 Cluster based Performance Evaluation of Run-length Image Compression Volume 33–No.5, international Journal of Computer Applications, Foundation of Computer Science New York.

[2] Amit chhabra, Gurwinder Singh 2010 Cluster Based Parallel Computing framework for Performance evaluation of Parallel Applications, Vol.2 April – 2, International Journal of Computer Theory and Engineering.

[3] Amit chhabra, Gurwinder Singh 2009 Simulated Performance Analysis of Multiprocessor Dynamic Space Sharing Scheduling Policy, Vol.9 Feb – 2, International Journal of Computer Theory and Engineering

[4] Michael Sung, 2000 SIMD Parallel Processing, 6.911: Architectures Anonymous

[5] Jonathan C. Hardwick 1997 "Practical Parallel Divide-and- Conquer Algorithms", CMU-CS-97-197

[6] Phyllis E.Crandall, Michael. j. Quinn 1993 Data Partitioning for Networked Parallel Processing, In Proceedings Data of Fifth Symposium on Parallel and Distributed Processing, Irving, TX.

[7] Visual Basic 6 Client/Server Programming Gold Book 1998, The Coriolis Group, ISBN: 1576102823.

[8] Bob Quinn Dec, 1995. Windows Socket Network Processing second edition Addison Wesley Professional, ISBN-10:0-201-633728.

[9] Ellis Horowitz, Sahni, 1978 Fundamentals of computer Algorithm Second edition, Computer science press.

[10] Thomas .H Coremen, Introduction to Algorithms, Second Edition Massachusetts institute of technology ISBN 0-262-03293-7.

[11] Omer Khan, MiesZko Lis, A Scalable Shared Memory Multicore Architecture. Massachusetts institute of technology Cambridge, MA, USA, june-2010, MIT-CSAIL-TR-2010-030.

[12] Steve J. Chapin, Syracuse University, Distributed and Multiprocessor Scheduling Volume 28 Issue 1, March 1996 ACM Digital Library New York, NY, USA.

[13] Kameswari Chebrolu, Socket Programming, Dept. of Electrical Engineering, IIT Kanpur.

[14] Rajinder Yadav, Client/Server Programming with TCP/IP Sockets, Sept 9, 2007.