Cluster based Performance Evaluation of Run-length Image Compression

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

Amit Chhabra
M.Tech (IT)
Guru nanak Dev University Asr.
Asst. Prof. at GNDU Cam. Asr

Harwinder Singh Sohal
M.Tech(CSE)
LLRIET , Moga
Asst. Prof at LLRIET, Moga

ABSTRACT
Modern data processing tasks involving high computation with huge data intensive work are not providing any usual response as they run over a conventional computing architecture, where the synergism capabilities of such machines are limited to single central processing unit. Improvement over such single processing architecture is not the big issue as many earlier efforts in this era has been performed, which involves overlapped pipelined architectures. Later the technology extends to involve multiple processing elements under the control of a common clock. A current trend involves multiple central processing units. Despite of such efforts, another way of achieving parallel effect is to make effective utilization of multi-computer hardware in the form of massively parallel clustering over a local area network. Further the experiment lead to the analysis of Run-length image compression over a network cluster-involving client – server model of computation consisting software modules implemented via TCP/IP sockets for the requirement of increased speedup as well as throughput. Finally, the conclusion containing comparisons over clustered environment will be discussed.

General Terms
Multi-Computer Cluster, Client Server TCP/IP Sockets, Compression.

Keywords
Parallel Clustering, Multi-Computers, Run-length Image Compression.

1. INTRODUCTION
Executing jobs with exhaustive computation and huge data processing over a single processor machines sometimes not accepted by real applications or scenarios where computation speedup as well as throughput will be considered as high priority attributes. Earlier research articles are related with simulated analysis having space sharing scheduling policies [2]. The Recent trend involves cluster based performance measurements where many people are devoting their efforts in cluster-based operation because of the easy availability of multi-computers and network infrastructures. In this research, the parallel cluster behaves like client-server architectural model, where one front-end machine act as a load distributor or a controller and rest of the nodes act as a processing units, also known as Asymmetric Multi-computer communication. Other benefits involves topological flexibility, Scheduling flexibility, fault tolerance, architectural-scaling flexibility etc [4]. provides transparencies in modern experimental approaches and frameworks. Cluster based parallel environment provides increased speedup and response time as compared to conventional machine architecture by running computation intensive jobs incorporating parallel behavior with workload partitioning and distribution [1]. The rest of the paper is organized under various sections. Section II describes the literature review. Section III indicates the topological software structure employed over the LAN. Section IV describes the socket semantics and connection establishment. Section V describes the programming paradigm used to implement parallel cluster. Section VI and VII describes the server side module interface and performance measurements based upon the collection of sampling results from the cluster environment respectively. Section VIII covers the performance metrics used in this experiment. Finally, section IX specifies conclusions and future directions.

2. LITERATURE REVIEW
Previous Literatures shows simulated multiprocessor scheduling environments having no. of virtual processors and job list. The environments are modeled as time-sharing or space-sharing policies where scheduler either distributes multiple jobs to multiple processors or allocates multiple processors to single job. The environment assumes virtual job lists arrived via poison distribution theory. Each job has its own total CPU burst time and a demand specifies the no. of processor required for achieving parallelism. Other experimentations involve clustered based operation over matrix multiplication. A large sized matrix typically 128X128, 256X256, 512X512 are considered for experimentation studies. Analysis results show that allocating extra computational resources without increasing job size leads to the performance degradation. Job Size must be increased to take benefits from such computational resources leading towards scaled speedup. Our work is similar but analyzes run length encoding scheme for high resolution (Twips Unit) image. In general the run length-encoding scheme is a lossyless compression technique covering each and every aspects of an image.

3. TOPOLOGICAL STRUCTURE

The cluster interconnection involves client-server model of communication containing nine independent machines with underlying homogeneous architecture equipped with Pentium 3.4 GHZ processors, 1 GB of RAM and windows XP SP2 operating system. One machine acts as a server containing server side software module having entire distribution logic, shared memory and controlling power. The rest of the eight independent nodes act as a client processing units cooperated to perform intended processing assigned by the server machine [1]. Each of the clients consists of client side software module cloning implemented to establish single instruction multiple data model (Flynn's taxonomy) [7] over different parts of the image. This parallel cluster having client-server architecture, where one front-end machine act as a load distributor and rest of the nodes act as a processing units is known as asymmetric structure of multi-computers. Other types of parallel computer architectures may be implemented via network cluster like overlapped pipelining, binary tree, hyper cube architectures, 2-D mesh etc [5]. Logical Image Data comprises of scan lines ranges for which a particular client performs their compression algorithm.

4. SOCKET SEMANTICS

Programming methodology consists of server side software program and client side software program. The server side software program consists of server socket containing listeners, initially running in listening mode for accepting client's connection requests. Each of the client machines consists of client socket containing request originators handled by a separate listener on the server side with unique port identification assigned to the both client and server application.

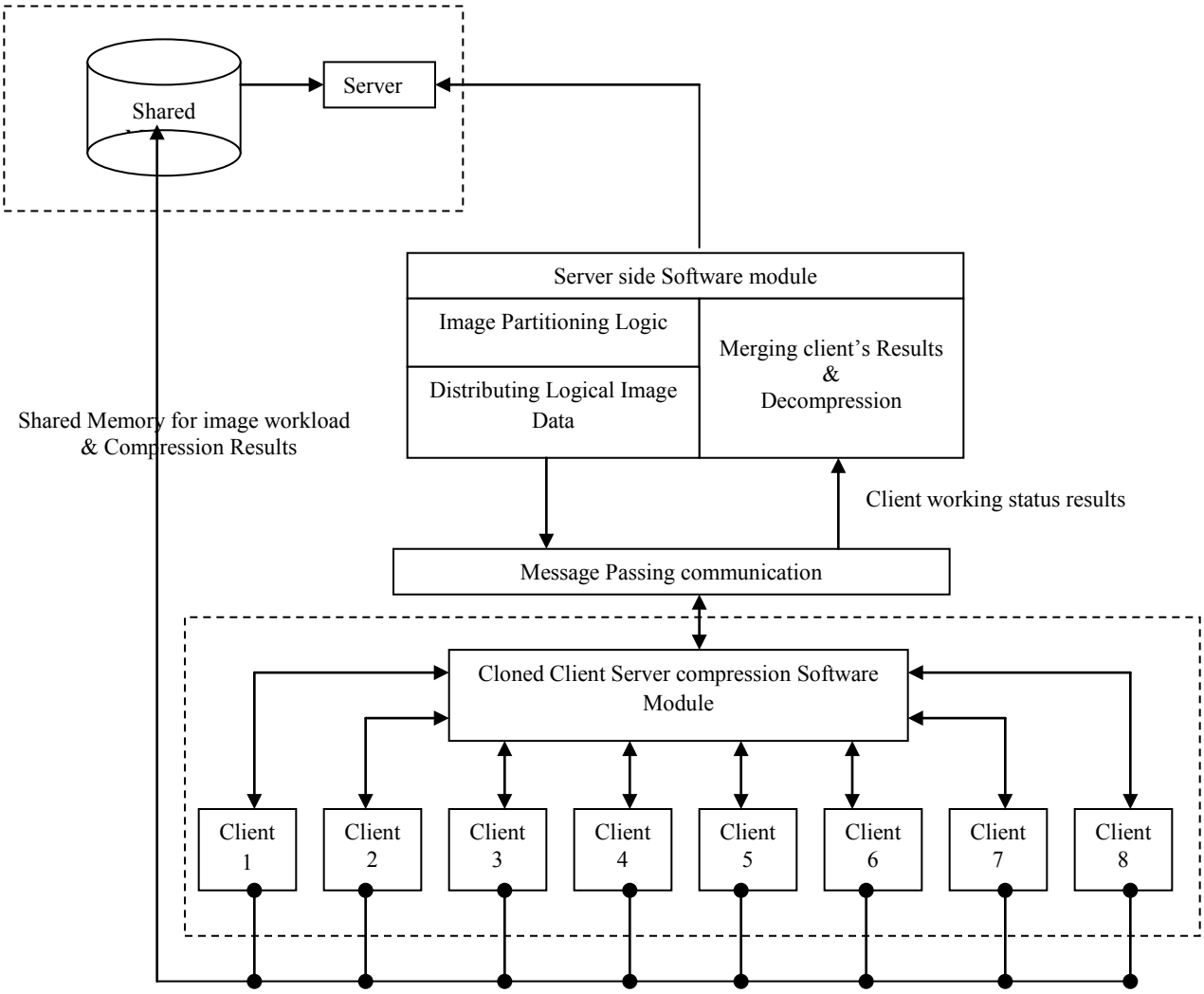

Fig 1: Describes Topological Structure

At server side corresponding listener's running in listening mode is waked up and establishes the connection. Unique port numbers will recognize server side listeners and client mapping [1][3]. All of the machines connected via network switch like a start topology and logically programmed to behave like a distributed parallel cluster.

The Final part of the paradigm is necessary and also is very computation intensive and takes more time if the large no. of cluster machines involved. Not only merging but also the partitioning time is more.

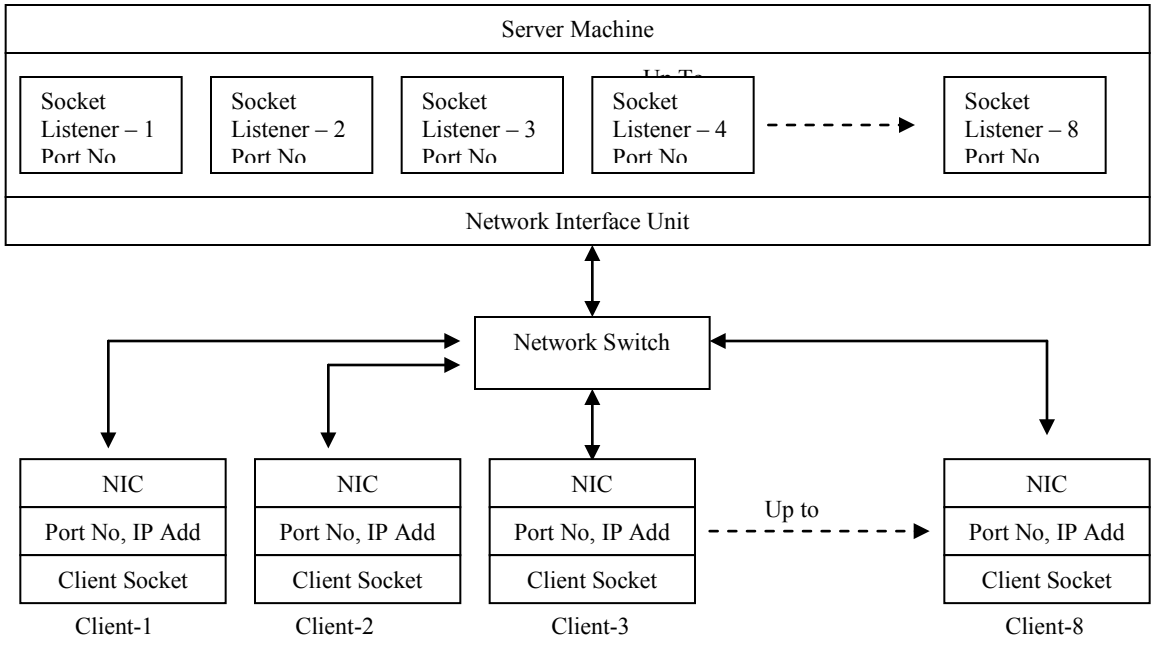

Fig 2: Describes Socket Semantics and Connection Establishment

5. PROGRAMMING PARADIGM

Run-length encoding implemented parallely by means of divide and conquer paradigm with coarse-grained workload partitioning and distribution. The Functionality of clients and server with respect to parallel divide and conquer paradigm [8] is as follows –

1. Server machine partitioned the image logically depending upon the number of specified clients using σ/P. Where σ stands for the total no. of scan lines available in the image and P specifies no. of clients involved in the computation. Server then distributes logical image partitioning information to designated clients via message passing Interface.

2. Each of the allocated clients then performs Run-length compression scheme over different parts of the image, sends compression result back to the shared memory structure and working status via message passing communication to server for ensuring the completion of the task.

3. Upon getting completion status, the server merges all of the individual clients results into one to make final solution.

6. SERVER SIDE MODULE INTERFACE

Cluster connected server side parallel interface is developed by using Microsoft Visual Basic 6.0 language [6]. The interface has three sections. The first section labeled parallel execution specifies the information regarding no. of child's/cluster computers, pending child's, time duration in milliseconds, status of the job and speed up over single processor machine. Second section represents the sequential Run-length compression algorithm containing information regarding sequential time duration. Third section represents the testing stage, where the image can be regenerated from compression results. The Image used for compression has the resolution (4095 X 1935) in twips units, which is both computations intensive as well as data intensive. Client side module interface is just the Run-length compression algorithm implemented to perform SIMD computation over different parts of the image. In the further sections computation results as well as performance metrics used in this experiment will be described. The sockets communication is performed via micro-soft Winsock activex control (Mswinsck.ocx) 6.0 version. The control has sendData method through which the message passing communication will be performed. Connect method is used to perform connection

establishment. The same work can also be performed via java socket classes and threads, where each thread acts as a listener for specific clients.

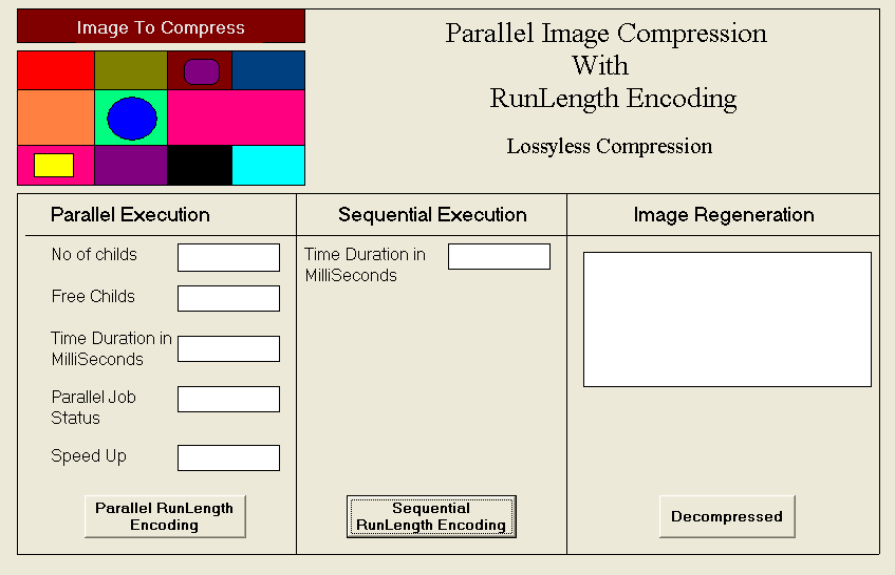

Fig 3: Server Side Module Interface

7. PERFORMANCE MEASUREMENTS

The experiment takes place by testing Run-length compression over the cluster of eight independent computing machines. Such variations are described in Milliseconds inclusive of both partitioning and merging time. So the time represented in Table 1 represents the total time from job partitioning to till its final completion after merging. This experiment, partition size are calculated by considering total image scan lines by no. of cluster machines as –

$$\text{Psize} = \frac{\sigma}{P}$$

.σ refers to the total no. of image scan lines and P refers to the no. of cluster machine used for computation

Table 1: Run-length Timing Variations

Cluster Clients	Time (Ms)	Time(Sec)
1	129723.44	130
2	66684.123	67
3	44013.556	44
4	35680.083	36
5	27830.64	28
6	25162.77	25
7	25944.34	26
8	28572.52	29

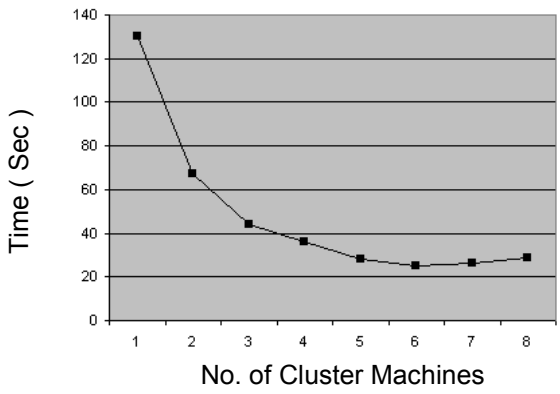

Fig 4: Run-length Compression Timing Variations

The above Timing variations of Run-length encoding over the parallel cluster is from the task submission to up to the time of solution merging i.e. the final outcome of all of the participating cluster machines. The image considered for this experiment is very high resolution and is not in pixel format but in twips format. In general parallel computing concepts are invented for high computation and data intensive work having large computational time over uni-processor architectures. Considering the following table 2 and fig 5 for speedup computations, further the efficiency and parallel overhead will be discussed.

Table 2: Run-length Speedup Variations

No. of Cluster Clients	Time (Sec)
1	0
2	1.94
3	2.96
4	3.61
5	4.64
6	5.20
7	5.00
8	4.48

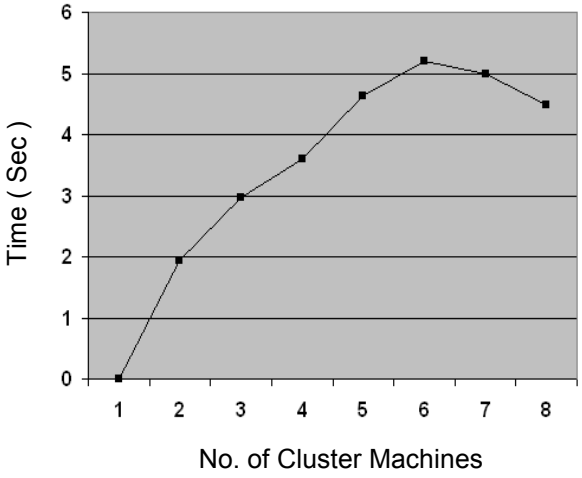

Fig 5: Run-length Speedup Variations

Following Table 3 describes the efficiency results of cluster computers when executing Run-length compression over the parallel cluster.

Table 3 Efficiency Per Cluster Machine

No. of Cluster Clients	Time (Sec)
1	0
2	0.97
3	0.99
4	0.9
5	0.93
6	0.87
7	0.71
8	0.56

Efficiency results shown as the cluster clients are increased to 6,7 or 8, the efficiency is decreased. This is because the size of the image is fixed and the computation resources are more than computation requirements, so partitioning and merging overhead is more than computation overhead.

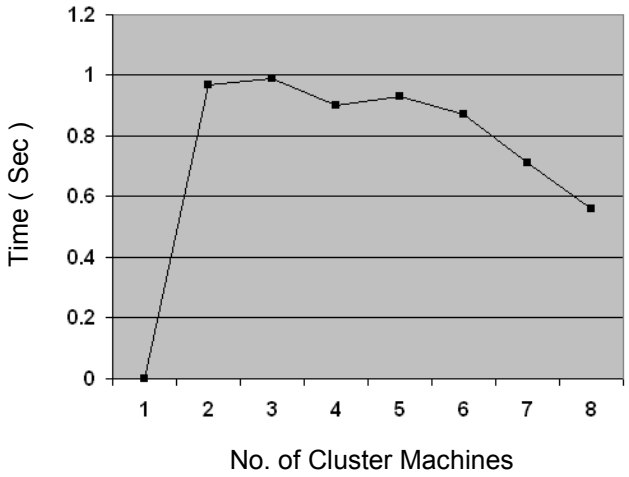

Fig 6: Run-Length Efficiency Variations

Consider other performance measurements generally described as parallel overhead. Parallel overhead is the overhead, which specifies the time spent in parallel computation managing the computation rather than computing results. Here β specifies the time consumed by parallel cluster having p machines and α refers to the time consumed by single machine for the same task. The overhead is calculated, as by multiplying no. of machines with the time consumed by no. of processor collectively and subtracting sequential time from it.

Table 4: Parallel Overhead (Sec)

No. of Cluster Clients	P * β	P * β – α
1	130	0
2	134	4
3	132	2
4	144	14
5	140	10
6	150	20
7	182	52
8	232	102

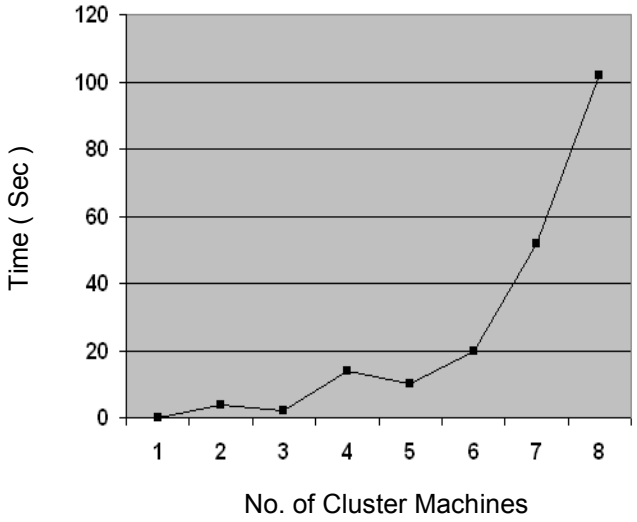

Fig 7: Parallel Overhead Per cluster Machine

8. PERFORMANCE METRICS USED

Following are the general-purpose performance metrics of parallel computation, which are used to evaluate Run-length image compression results.

Speedup is calculated by dividing single computer timing results with multi-computer timing results as –

$$\text{Speedup (S)} = \frac{\alpha}{\beta}$$

Where α stands for time consumed by single processor machine and β stands for the time consumed by multi-computer cluster of P machines. Table 2 and Fig 5 describes the timing results as speedup variations after executing Run-length compression over single machine as well cluster of P machines.

Efficiency is another major performance evaluation attribute related to the processor's throughput. Efficiency estimates that how well utilized the processors or machines in solving current problem. Efficiency also defined as average contribution per processor in computation speedup. Table 3 and Fig 6 describe the efficiency per machine.

$$\text{Efficiency (E)} = \frac{SpeedUp(S)}{P}$$

$$\text{Efficiency (E)} = \frac{\alpha}{\beta * P}$$

Where α and β refers to the time consumed by one cluster machine and time consumed by cluster of P machines respectively.

Parallel Overhead a measurement unit of performance evaluation, which specifies the time spent in parallel computation managing the computation rather than computing results. Table 4 and fig 7 describes the parallel overhead occurred after execution of the Run-length encoding –

$$\text{Parallel Overhead (POVR)} = P * \beta - \alpha$$

Consider Table 4 and fig 7 for understanding parallel overheads over clustered environment.

9. CONCLUSION & FUTURE DIRECTIONS

The experiment produces effective and efficient results incorporating cluster based Run-length compression covering cost effective utilization of multi-computers in the form of parallel cluster. As discussed, the experiment is based upon client-server computing, logically programmed to implement SIMD Flynn's based computation [7]. The ultimate conclusion describes that high computation as well as data intensive jobs can be prepared to run over a parallel cluster facilitating the needs of real world by achieving speedup benefits. Many organizations have lot of computing infrastructures discarded not because of their software/hardware errors due to aging but because of their low performance as compare to modern scientific activities, such industries can improve the efficiency of their low performance computing machines in the form of groupware as they collectively perform a single computational task. Despite of these, other benefits with respect to network-oriented environment includes flexibility for the programmer to implement any types of logical parallel structure. Programmer can schedule or reschedule any job to any network node as well as load sharable environment over the cluster nodes etc. Compression results computed after executing compression algorithm over a cluster oriented environment exhibits high quality speedup benefits as compare to single uni-processor computing structures, as shown in the Table 1, the time consumption calculatation is in the form of seconds including both partitioning and merging time, i.e. the time from job partitioning to the merging of final results. Note that the speedup benefits decreased in comparison to previous achievement as soon as the no. of nodes in the cluster increased, this is because the job size is fixed but there is an increase in processing units. So inference from this result is that allocating more and more processing entities to a fixed sized job will degrade the performance and wastage of computation resources as shown in the Figure-4 when cluster size is increased to 7 or 8. The Experiment says there must be the increase in input size if there is a increase in cluster size only then scaled speedup will be achieved. Future work will be the improvement over Run-length compression by combining some lossy compression schemes to make more advance compression benefits.

10. REFERENCES

[1] 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.

[2] 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

[3] Hemal V. Shah, Calton Pu, Rajesh S. M. 2006 Network-Based "Parallel Computing. Communication, Architecture, and Applications" Vol.-1602, Springer Berlin / Heidelberg, Dec 29.

[4] Chee Shin Yeo, Raj Kumar Buyya, Hossein Pourreza, Rasit Eskicioglu, Peter Graham, Frank Sommers 2005 Cluster "Computing: High-Performance, High-Availability, and High-Throughput Processing on a Network of Computers". ICCS- 5th International Conference, Springer Verlag Berlin Heidelberg.

[5] Daniel Schulze Zumkley Architectures of Parallel computers, Westfälische Wilhelm's Universitat Munster.

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

[7] M.J.Flynn, 1972 "Some computer organizations and their effectiveness," IEEE transactions on computers, 21(9):948-960.

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