Topological Structure for Parallelizing Multicomputer Cluster

Deepak Sharma
Computer Science & Engineering
Desh Bhagat Foundation Group of Institutions, Ferozepur Road
Moga, India
deepdeep2003@gmail.com

***Abstract*—Parallel computation, an extension to multi-programming architectures usually structured as tightly coupled organization of multiple CPU cores. Systems under such configurations require lot of effort to manage multiple tasks simultaneously. Operating systems for such hardware follows several real time constraints in order to enhance system performance. Normally, Operating system designates one processor from others as controller which acts as a load scheduler for the others and performs balancing when system performance degrades due to overloading of some of the CPU cores. Regardless of tightly coupled system, another cost effective organization to achieve parallelism is to interconnect multi-computers as a network of cluster. The advantage of this loosely coupled system is that system is under programmatic control. Low level sockets connections are created to make machine to machine communication possible. This work focuses on Multi-Ethernet Wired LAN Cluster (MEWC) and Broadcasting Wireless Access LAN cluster (BWAC) for executing multiple tasks like a grid. Further, the work analyzes both wired and wireless clusters along with some factors considered in communication network. Wireless network of multi-computers have the advantage of transmission speed. In wireless cluster, enhanced data transmission speed is achieved because of the effect of broadcasting links, where wired network survive on single communication link; although Multi-Ethernet distribution may be followed for the improvement. But still wireless LAN gives many advantages.**

Keywords—Parallel cluster; wireless communication network; broadcasting data; access point; multi-computers; Multiple Ethernet Interface Card

I. INTRODUCTION

Managing multi-computers requires stream connections among client and servers. Systems are programmed either through sockets or RMI (remote method invocation). Listeners are created under master controller waiting for requests generated from clients. Once the systems are authenticated, the master controller distributes tasks to their respective slaves via wireless communication. After task completion the slaves return to their controller for next task assignment. Loosely coupled interconnection is totally programmed by programmer to emulate parallel effect whereas tightly coupled system is under the control of operating system. Interference from the user side is totally abstracted. User can't manipulate the processor schedule by low level programming. Multi-computers provide similar level of parallel computation by communication network. Machines perform their work according to the logic assigned. So such systems are logically programmed for a given set of functionalities. Cluster computing provides industrial advantage to utilize low powered processors rather than discarding. This is because such systems are not fulfilling the requirements of HPC (high performance computing). In reality several high performance applications consists of huge data and computation intensive work which must be processed quickly and efficiently [1].

II. RELATED WORK

Work done so far implements wired framework for cluster computing. Each cluster machine is connected as star topology in the communication network. Data transmission and connection handshaking is implemented via sockets API. Listener and port numbers are assigned to the programs running at each cluster machine. Computation covers image compressions by decomposition of image into chunks and then performs distribution to various clients. Clients performs their assigned algorithmic work and return compressed result back to the controller, where controller waits for each client's task completion and finally consolidate the compression results. Similar work implemented for matrix multiplication Strassen and Winograd approaches. Matrix is divided into parts according to the algorithmic structure and then assigned to clients as the basic divide and conquer approach suggests. Protocols implemented for cluster programming is TCP/IP because of connection oriented smooth connections are required. But despite of this due to network errors their might be possible that data may be lost or received with delay. Further we will discuss the problems occurred in wired networks when implemented as clusters [2].

III. CLUSTER INTERCONNECTION

Clusters are configured to emulate a particular set of functionalities. Not all features are implemented in a single cluster, this is because as the functionalities are added more work has to be done on the controller side as well as layer of algorithms will be implemented under each cluster node from among an algorithm selection is made whenever node gets task ready for it. Complexity will be increased at software level whereas communication is dependent upon network speed, so data transmission may be delayed. So this will be improved via multi-cluster interconnection. Each cluster set is a group of controller and clients to a do a particular set of tasks. Each controller machine may further interconnect with other controllers either via start or mesh. This is required if

load balancing or stress estimation is the part of interconnection. Overloaded machines may be discarded from their intended cluster for getting next task assignment. Load may be scheduled to other cluster's machines if free or having capability to compute efficiently. Interconnecting wired clusters requires switch to switch level cable connection but in wireless interconnection wireless routers/access points are configured to communicate.

IV. Workload Characterization

Parallel workload consists of task data set to be computed by parallel hardware. Simulation systems are created for handling synthetic workload relatively a type of benchmarking. Workload under this scheme is totally a type of replica of original system under execution. Such workloads are created for testing simulation behavior acting as a foundation for future hardware development. Synthetic workload generated in random fashion covering task id along with other attributes like total execution time, arrival time, schedule time, completion time, waiting time, etc. In spite of synthetic workload real system constructed for parallel execution gets a complete task workload along with its algorithmic control logic as well as data sets. Here huge computation oriented work is given by user to the controller which further distributes tasks either by control parallel or data parallel approaches. Control parallel basically follows the rule of multi-tasking system whereas data parallel approaches follows the rule of divide and conquer methodologies. Divide and Conquer programs are controlled by a common clock usually at same time interval common algorithmic logic and different data sets are distributed. In control parallel systems different control units are needed to manage each algorithm and operational data simultaneously. This work focus on frame work for both wireless control parallel and data parallel systems. Control parallel approaches are usually known as TPS (task parallel system) Tasks are arrived and get schedule when machines free. Data parallel approaches are usually known as PDS (problem Decomposition and Distribution System) where data domain is divided in sub units of and then distributed among several clients for faster response [3], [4].

V. Layered Framwork for Parallel Cluster

Layered framework provides an underlying model to develop programmatic environment for clusters as described in Fig. 1. Each layer provides well-defined structural elements required to utilize cluster hardware. Cluster controller specifies a master node having parallel API routines called whenever parallelization is needed. Each cluster client node having algorithmic logic created to manage workload computation. Monitoring system maintains each cluster status to handle performance degradations if exists. The layered framework is similar to every cluster model whether a wireless or wired interconnection. Further, the research describes both single ethernet and multi-ethernet wired cluster architectures. Each interconnection consists of ethernet card along with cath5e twisted pair cables.

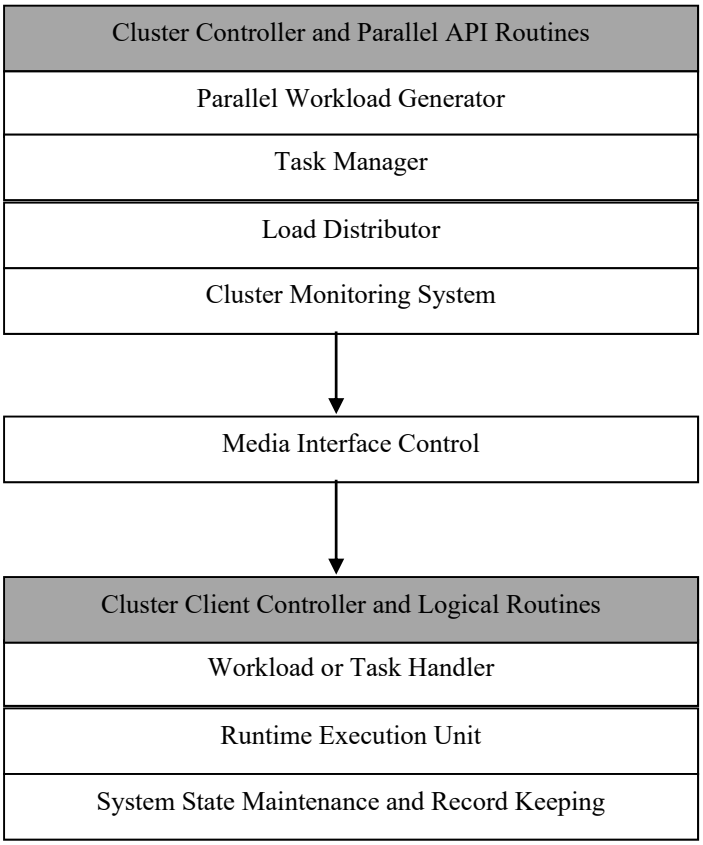

Fig. 1. Layered framework.

Basically, tasks are submitted and assigned to appropriate cluster node. Different types of distributions are performed, if the task has parallel behavior then task modules are distributed among cluster nodes, otherwise multiple tasks which have non-parallelized structure behavior are scheduled to different cluster nodes [5].

VI. Runtime Dependency Check

Cluster frameworks are well suited if there is no task interdependency. Usually, task manager firstly inspects the task for data source requirements, if the module interdependency exists there then system delays these tasks for current execution and performs schedule for other independent tasks sets. Some systems firstly schedule independent tasks and then tasks which are dependent upon previously scheduled independent tasks and so on. This method is more suitable to handle interdependency. Some compilers have inbuilt dependency checker which guides programmers during parallel logic construction. Normally when tasks are decomposed then their threads may have communication coupling. Stream linking is created to transfer bulk of data over which further data processing takes place. So better to check interdependency before decomposition, also modules which are dependent may be scheduled to same cluster or the task should be scheduled completely [6].

VII. CLUSTER ARCHITECTURES

There are usually three types of cluster architectures. Single ethernet wired cluster where master controller having a single interface ethernet card as described in Fig. 2. Whole communication with multiple clients takes place using this common media interface. The problem with this type of cluster interconnection is that whenever multiple clients respond or gets data from the servers simultaneously/concurrently there will be a communication delay; although work is parallelized but not be efficiently performed. Other problem due to congestion is the data loss. At the end of computation each node returns result which might be a heavily loaded the server or delayed other cluster nodes from responding their results.

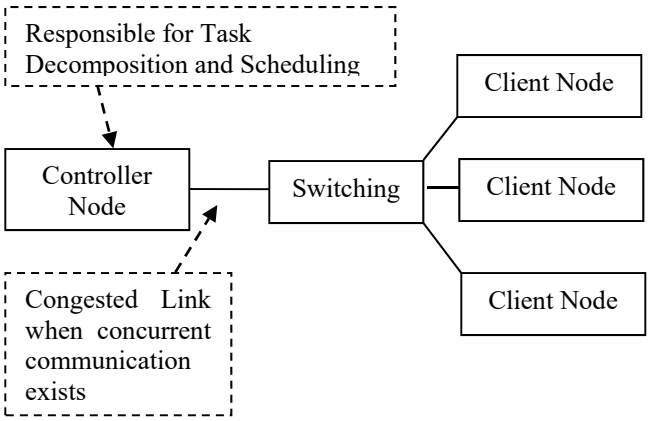

Fig. 2. SEWC architecture.

Multi-ethernet wired cluster is the improvement over single ethernet wired architecture as expressed in Fig. 3. As the master link is congested when concurrent communication exists. So multi-ethernet architecture consists of multiple network interface cards where each communication line mapped with different cluster nodes. One to one pair mapping may be created or to make cost effective one to many cluster nodes depending upon the link bandwidth may be established.

One to one mapping of communication link is very expensive so depending upon the speed and bandwidth of communication link multiple cluster nodes may be mapped with a single communication line as described in Fig. 4.

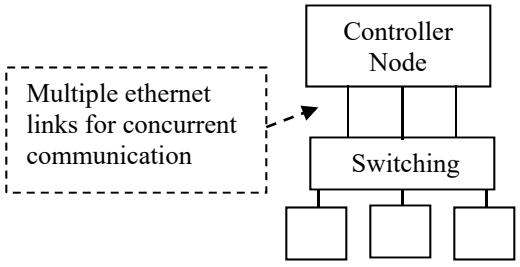

Fig. 3. MEWC architecture Type-I.

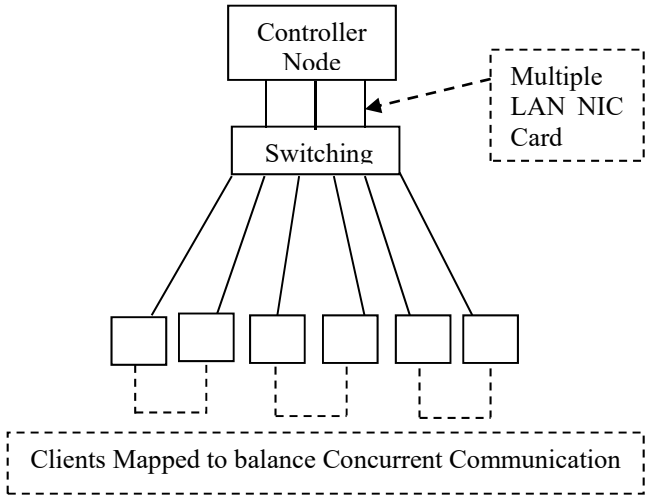

Fig. 4. MEWC architecture Type-II.

Basically parallel cluster communication implies concurrency during read or writes operation. This will create congestion over the server media interface as described above. So cluster implementation must follows multiple network interfacing in order to make dedicated transmission flow. Although implementation will be little bit costly. But now a day's NIC cards are easily available from garbage hardware machines which can be easily plugged into PCI slots, even the PCI extension cards are available. Other system employs USB network interface cards making NIC hub. Note that even with multiple ethernet interface cards server to client communication handling is sequential but clients to server communication is parallel. When multiple clients communicate with the servers memory, they have parallel lines as well but when server transmits computation requests it identifies the free client node and then selects appropriate mapped link to that client and so on. This process makes server transmission as sequential but as the cluster nodes comes into progression concurrency will be improved. Other type of architecture which will improve the network communication to avoid barriers is the broadcasting networks illustrated in Fig. 5, BWAC architecture, where a communication access point will manage the transmission. Each machine now has a Wireless NIC cards along with a broadcasting antennas. Access point devices with multiple antennas covering Omni-directional links configured to manage Wireless LAN systems. Use of broadcasting will help to manage intermediate congestion which will be the result if wired clusters are implemented. Wireless systems provide just a directional flow. In order to make multiple cluster interconnections, more than one access points may be configured at local level. Level of topology depends upon the functionality given and requirement of parallelizing problems.

Wireless LAN systems are less error prone than wired LAN system, communication will be fast, establishment of a cluster like a grid may be easy as no crowd of wires exists as compare to wired networks transmission links are monitored multiple times because of RJ45 jack connections.

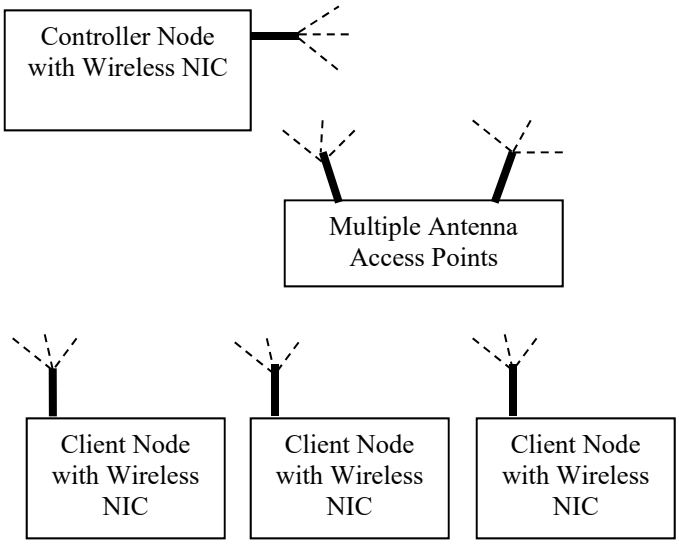

Fig. 5. BWAC architecture.

VIII. Parallel Workload

In order to achieve parallel aspects, implemented algorithms must have concurrent modules. So that job logic must be distributed in a parallel fashion. Parallelism follows the concept of space sharing and time sharing policy structures. In space sharing policy structures multiple processors are allocated to single active job. In time sharing policy structure multiple jobs are allocated to a single active processor. Usually MIMD systems are implemented in TSS (Time Sharing System) and SIMD systems are implemented in space sharing system. Space sharing system requires a lot of efforts to distribute problem domain and consolidation domain units at the end of computations. Space sharing systems requires processing demands for each job and must have bulk of processor space available. So to implement parallel schedulers managing processor space is more important than managing workload. Following types of parallel logics are build [7], [8]:

a) Fixed Demand Rigid Jobs – Here a job requests some number of processor and must be executed whenever that demand is fulfilled. Otherwise job must have to wait and scheduler gets next job set allocation. This policy structure is inflexible because once the processing units are allocated; they must be allocated throughout the jobs execution life span.

b) Scheduler oriented moldable Jobs – Another type of job structures where scheduler set the job demand initially depending upon its parallel behavior or its concurrent modules. Note that this allocation is again fixed once allocation made no change will happen in demand settlement during execution. For implementing this type of scheduler, there is a need of code inspection. Management of processor availability is must because as much of the free processor space available, the scheduler can schedule many jobs or there will be increased throughput.

c) Malleable Jobs – Malleable jobs are more beneficial than moldable because here scheduler modify the job demand at any time because parallelism may change throughout the job execution. This will also manage the processor space, runtime change in demand will provides efficient mode of allocation.

d) Growing Jobs – Evolvable also known as growing tasks are similar to the structure of malleable processes, such applications demands are not controlled by schedulers but application itself provides information for change in processing needs. In other words application itself manages when to adjust processor requirements. In malleable scheduling scheduler decides when to change processing demands but in evolvable decision is taken by the task during runtime parallelism [9].

Task level parallel schedulers are one step ahead to the pipeline schedulers where n-number of tasks are filled in the pipeline unit and then tasks are passed away from each unit one by one in overlapped fashion. Task level parallel schedulers requires great amount of programming efforts. Load balancing is a high priority work required to manage performance when degradation happens. Effective scheduling specifies that better task allocation rather than performing task replacement/adjustment in order to balance load in between. Adjusting tasks later on to and from processors queues is more costly than initial effective allocation; wait time of tasks may increase due to revolving around queues [10].

IX. Cluster Analysis

Clusters architectures are analyzed according to the different parameters involved in the model. Table 1 describes the communication behavior of the cluster. Analysis also covers traditional matrix multiplication algorithm on SEWC type I and BWAC architectures by taking different matrix sizes as described in Tables 2 & 3. Fig. 6 and 7 expresses the simulation results.

TABLE I. Wired and Wireless Cluster Analysis

Factor/Type	SEWC	MEWC-I	MEWC-II	BWAC
Data Access Speed	√	√√√	√√	√√√√
Concurrency Level	√	√√√	√√	√√√
Safe State	√	√√√	√√	√√√
Delay/Latency	√√√	√	√√	√
Data Loss	√√√	√	√√	√
Congestion/Traffic	√√√	×	√√	×
Collision Rate	√√	×	√	×
Mobility	×	×	×	√
Cost Effectiveness	√√	√√√√	√√√	√
Degree of Parallelism	√√	√√√√	√√√	√√√
Scalability	√√√	√√	√	√√√√

TABLE II. SEWC Matrix Analysis

Matrix Size (n) /Cluster Nodes SEWC-(Sec)	2000	4000	8000	12000
4	6.82	12.45	18.55	29.45
8	4.65	7.85	10.53	19.23
10	3.58	9.67	8.34	13.42

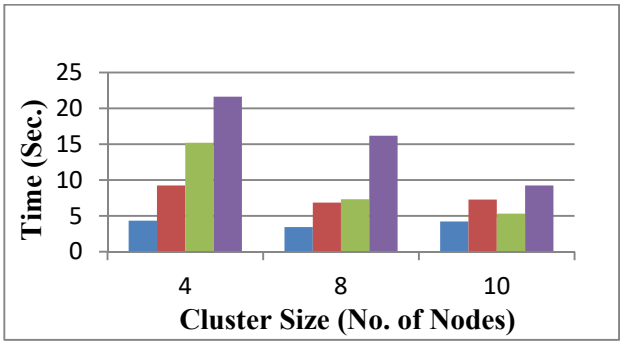

Fig. 6. SEWC Matrix Analysis.

TABLE III. BWAC MATRIX ANALYSIS

Matrix Size (n) /Cluster Nodes BWAC-(Sec)	2000	4000	8000	12000
4	4.34	9.25	15.21	21.63
8	3.44	6.85	7.33	16.19
10	4.22	7.29	5.32	9.24

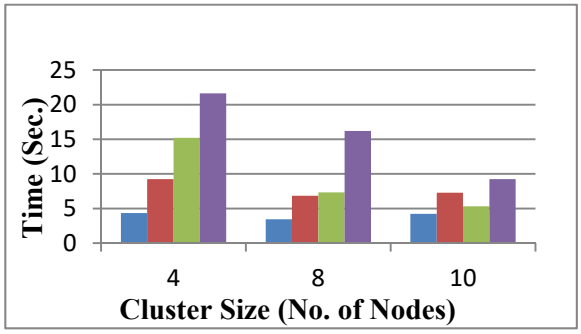

Fig. 7. BWAC Matrix Analysis.

Below is the analysis of multi ethernet wired LAN cluster. Four ethernet cards are used to make minimum one to one data communication possible for each node, maximum four cluster nodes are handled by each ethernet communication (Table 4). Simulation results are illustrated in Fig. 8.

TABLE IV. MEWC MATRIX ANALYSIS

Matrix Size (n) /Cluster Nodes MEWC-(Sec) Ethernet cards 4	2000	4000	8000	12000
4	5.24	10.15	17.81	23.31
8	3.89	6.77	9.23	17.67
16	1.45	3.78	4.44	7.42

Fig. 8. MEWC Matrix Analysis.

X. CONTRIBUTION REVIEW

Technical contribution covered in the research is about analysis of computation and data intensive matrix multiplication. Traditional algorithm is used as previous literature cover analysis of Strassen's and Winograd approach to wired clusters. Matrix multiplication problem is considered because of high degree of parallel decomposition exists and data intensive work will be generated. Our work focuses an improvement over existing systems using wireless clusters. Wireless systems proofs high concurrency than wired clusters. Disadvantages behind wired clusters are that their implementations have maximum data transfer delays/data loss, also initial and ending transmissions creates polling. Other limitations of wired clusters are that they are not true parallel systems. Workload during parallel simulation implementation is taken using random no. generator, as matrix source is generated at run time before actual distribution is performed. Multiple scenarios of workload are taken and average analysis is considered from each run-time execution. Wireless cluster may have interference problems but not severe, if happens, this will be recovered using re-transmission whenever system detects delayed response from a particular cluster node. Current implementation provides algorithmic structure over TCP/IP network; programmatic layered frame work is described for parallel task decomposition and distribution. As described large level grid oriented computation is not implemented, our work is to utilize the low speed computation processors as parallelizing complex problems. Frequency standard follows in ethernet cluster is 802.3 with 10/100Mbps over shared ethernet communication. In wireless cluster the wireless access point transmission rate is 54Mbps for each network broadcasting with frequency used is 2.4 Ghz with 802.11g standard. Factors mainly considered are described in Table 1, maximizing degree of parallelism level without possible delay and loss of data. Other factors considered scalability and decreasing level of polling during broadcasting to various cluster nodes. Traffic controlled using wireless broadcast access because of concurrent arrival rate. Further if multiple clusters are interconnected they may have particular set of tasks associated. This system may help in future parallel hardware implementation. High degree of programming using socket connection exists in this work. Existing literature follows divide and conquer approach to matrix multiplication. These will be compared with traditional matrix multiplication improvement over wireless topology. Algorithm follows only data partitioning and distribution according to a particular cluster set. Strassen and Winograd methods use both data partitioning as well as divide and conquer algorithm structure. In comparison to existing work results are improved. Conclusion stated that as the nodes in a particular cluster set are increased their must be the increase in problem domain also, otherwise the system consumes more time in partitioning and final results consolidation than actual computation [11], [12].

XI. CONCLUSION AND FUTURE WORK

Wireless clusters are more beneficial than wired clusters, also provides mobility to the system interconnection. Benefits from broadcasting facility provide transmission speed during concurrent access which may be delayed during wired

communication as described. Analysis covers some parameters and an execution of a matrix multiplication algorithm. Conclusion from the research is that as the cluster size and problem size increases, the performance becomes visible. This will depend upon the size of the problem domain. Smaller size modules when executed over larger sized cluster will degrade the performance. So both the sizes, of the cluster and the problem domain increases, respectively. Above results are captured by executing system simulation build via java programming using sockets connection and multi-threading. In SEWC twisted pair cables cath5e and gigabit ethernet cards along with Linux OS with dual core processors. BWAC clusters are implemented via single antennas wireless access points and ethernet cards capable of transmitting 150Mbps speed. Future work of this research may include analysis of other different problem domains. Further image compressions, encoding, hashing algorithms may be analyzed over wireless clusters.

REFERENCES

[1] A. Chhabra, "A Cluster Based Parallel Computing framework (CBPCF) for Performance Evaluation of Parallel Applications," International journal of computer theory and engineering, vol.2, pp.1793-8201, 2010.

[2] A. Arora, "Cluster Based Performance Evaluation of Runlength Image Compression," IJCA Foundation of Computer Science New York, vol.33, 2011.

[3] J. Moscicki, "Processing Moldable Tasks on the Grid: Late job binding with light weight user-level overlays," in Elsevier Science, Publisher B. V. Amsterdam, The Netherlands, pp. 725-736, 2011.

[4] V. Nguyen, R. Kirner, "Demand Based Scheduling Priorities for Performance Optimization of Stream Programs on Parallel Platforms," 13 International Conference, ICA3PP, Springer Berlin Proceeding Part-1 pp.18-20, 2013.

[5] K. Huang, "Moldable Job Scheduling for HPC as a Service with Application Speedup Model and Execution Time Information," Journal of Convergence, vol. 4, pp. 14-22, December 2011.

[6] S. Kolliosopoulos, G. Steiner, "Partially Ordered Knapsack and Application to Scheduling Elsevier," vol. 155, issue 8 pp. 889-897, 2007.

[7] S. Bagga, D. Garg, "Moldable Load Scheduling Using Demand Adjustable Policies," ICACCI Galgotia Noida, IEEE Xplore, pp. 143-150, 2014.

[8] A. Nasr, "Task Scheduling Optimization in Heterogeneous Distributed System," IJCA, Foundation of Computer Science pp. 0975-8887, 2014.

[9] M. Etinski, "Parallel Job Scheduling for Power constrained HPC systems," Elsevier Science, B. V. Amsterdam, The Netherlands, pp. 615-630, 2012.

[10] G. Sabin, M. Lang, "Moldable Parallel Job Scheduling Using Job Efficiency: An Iterative Approach,"12 international workshop JSSPP, Springer Berlin, Saint Malo, France, pp. 94-114, 2006.

[11] A. Stephen, P. Daniel,"Dynamic Scheduling of Parallel Jobs with QOS demand in Multi-Cluster and Grid," University of Warwick, 5th IEEE/ACM international Workshop on Grid Computing , pp. 402-409, 2014.

[12] G. Feitelson, "Job Scheduling Strategies for Parallel Processing," IPPS/SPDP 99 WS, JSSPP 99,978-3-540-66676-9.