International Journal of Networking & Parallel Computing
Volume 2, Issue II, April-May 2013

Batch Scheduling based on QOS in HPC - A Survey

Reshmi R, Dr. D Shanthi

Assistant Professor, Department of Computer Science & Engineering, SBM College of Engineering, Dindigul, TamilNadu, India.

Professor, Department of Computer Science & Engineering, PSNA College of Engineering & Technology, Dindigul, TamilNadu, India.

ABSTRACT

High performance distributed applications require high performance and Quality Of Service (QOS). Because of its complex runtime environment and with these requirements, it makes such applications extremely challenging. Performance evaluation and prediction of these applications in computing environment is a very difficult task. High performance clusters gains the relevance, as a common place for such computational intensive applications. Such applications typically run on a set of heterogeneous machines with dynamically varying loads. These heterogeneous machines are connected by heterogeneous networks possibly supporting a wide variety of communication protocols. The basic problem in parallel computing is how to execute a parallel program on a collection of heterogeneous processors, that is processors with different and possibly changing speeds. Many scheduling algorithms are designed to execute a job efficiently in a parallel computing environment. The goal of this paper is to study on recent batch scheduling methods used for resource allocation across multiple nodes in multiple ways and the impact of these algorithms. This paper presents a review of the literature on batch-processing problems by analysing QOS metrics in high performance computing environment. Herein we discuss about the area and the significance of scheduling, based on papers reviewed prevoiusly.This paper proposes an approach for batch scheduling of jobs by considering certain QOS metrics.

Indexing terms/Keywords

Batch Processors, Dynamic allocation, Quality Of Service, Scheduling

Academic Discipline And Sub-Disciplines

Information & Technology

SUBJECT CLASSIFICATION

Parallel Computing

TYPE (METHOD/APPROACH)

Survey

INTRODUCTION

Traditionally large scale computational problems have been solved using massive parallel computers located at research centres and national labs. An approach for solving large problem is to simultaneously use multiple parallel computers located at different sites. Many of these likely to be multiuser machines with widely and dynamically varying load. The computational resources are in turn connected together using a set of communication resources. This communication resources includes heterogeneous network such as high performance networks, ATM networks, custom-high performance networks, or combinations of these.

A change is eminent in nature, and this seems certainly to be true in the market of HPC. This market was progressively undergone rapid change of vendors, architectures, technologies and the usage of systems. The performance of microprocessors has been increasing in an exponential way for the last four decades and so the scope of computer's processing ability does not have a fixed definition. A computer is considered to be high performance if it uses multiple processors (tens, hundreds or even thousands) connected together by some network to achieve well above the performance of a single processor. Use of multiple processors to enhance computing performance is known as parallel computing. Parallel computing includes several computations carried out simultaneously. In parallel computation, the jobs must be scheduled in a better way to achieve better results.

The first systematic approach to scheduling problem was undertaken in the mid of 1950s. In the second half of the seventies, the introduction of vector computer systems marked the beginning of modern supercomputing. And in the eighties, the integration of vector computation in conventional computing environment became more important. Thereafter Massive Parallel Processing (MPP), which utilizes a large number of processors to perform a set of coordinated computations in parallel, became successful. The price/performance ratios contributed to the success. Later it was replaced by microprocessor based Symmetric Multiprocessor Systems (SMS) which are tightly coupled multiprocessor systems with a pool of homogenous processors running independently on different data and with the capability of sharing resources.

Scheduling the execution of parallel algorithms on parallel computer is an important and challenging area in current research. It is the job of a scheduler to determine when, where and how the given task should run and upon that direct the resource managers appropriately. Parallel computers can be used to solve scheduling problems very fast. Although the importance of parallel scheduling algorithms has been widely recognized, only few results are obtained so far.

Programming parallel applications is difficult and not worth the effort unless large performance gains can be realized. Scheduling is a key part of the workload management software which usually performs queuing, scheduling, monitoring, resource management and accounting. The critical part of scheduling is to balance policy enforcement with resource optimization in order to pick the best job to run. A scheduler selects the best job to run based on the defined policies and available resources, then executes the job and after completion, cleans up and loops for the next job. Individual nodes can be allocated with number of tasks which is less than its maximum load. The load of the processor is the sum of the processing times of the tasks assigned to it. A parallel computation consists of a number of tasks, T_1, T_2, T_3 .... T_n that have to be executed by a number of parallel processors P_1, P_2, P_3........ P_m. The task T_j requires processing time P_j and is to be processed sequentially by exactly one processor. The length of a schedule is the maximum load of any processor. The aim is to find a schedule with minimum length which powers better performance. i.e. a classical problem in scheduling theory is to compute a minimal length scheduler for executing 'n' unit length tasks on 'm' identical processors constrained by a precedence relation.

In parallel computation, the scheduling and the mapping tasks are considered to be the most critical problem. Parallel computation solves a problem by breaking it into subtasks and working on those subtasks at the same time. These application subtasks are assigned to parallel machines for execution, by an efficient scheduler that guarantees minimum execution time and satisfies load balancing between processors of parallel machines. High performance computing ensures better performance by the utilization of an efficient scheduling algorithm. These parallel machines infrastructure may be homogenous or heterogonous. Homogenous systems use same machine power and performance. While heterogeneous infrastructure includes machines that differ in their performance, speed and interconnection.

Batch scheduling is a standard method for sharing a cluster among HPC. The main issue is, it inherently limits overall resource utilization. Batch processors uses integral resource allocation with no time sharing of nodes, then incoming jobs can be postponed while some nodes sits idle.

The performance measures can be classified into two as user oriented and system oriented measures. The user oriented measure includes quantities such as yield, response time, turnaround time etc. The system utilization in terms refers to number of processor included in execution cycle. Here synchronization must takes place at the end of every cycle, while certain processors wait for others to complete the job. So it needs a perfect load balancing system for efficient execution.

International Journal of Networking & Parallel Computing
Volume 2, Issue II, April-May 2013

This paper is organized as follows: In section 2, it gives a statistical analysis and background knowledge of the works undertaken in the scheduling area. Section 3 gives a literature survey. Section 4 gives proposed system. The conclusion arrived at section 5.

2. STATE- OF –THE- ART

The sharing of computing resources amongest competing instances of applications, or jobs, within a single system has been supported by operating systems via time-sharing, for decades. The second half of the eighties witnessed a shift to scheduling of jobs in parallel system to avoid bottlenecks. However many conventional methods and algorithms emerged in the nineties which emphasised the importance of scheduling classification for better system performance.

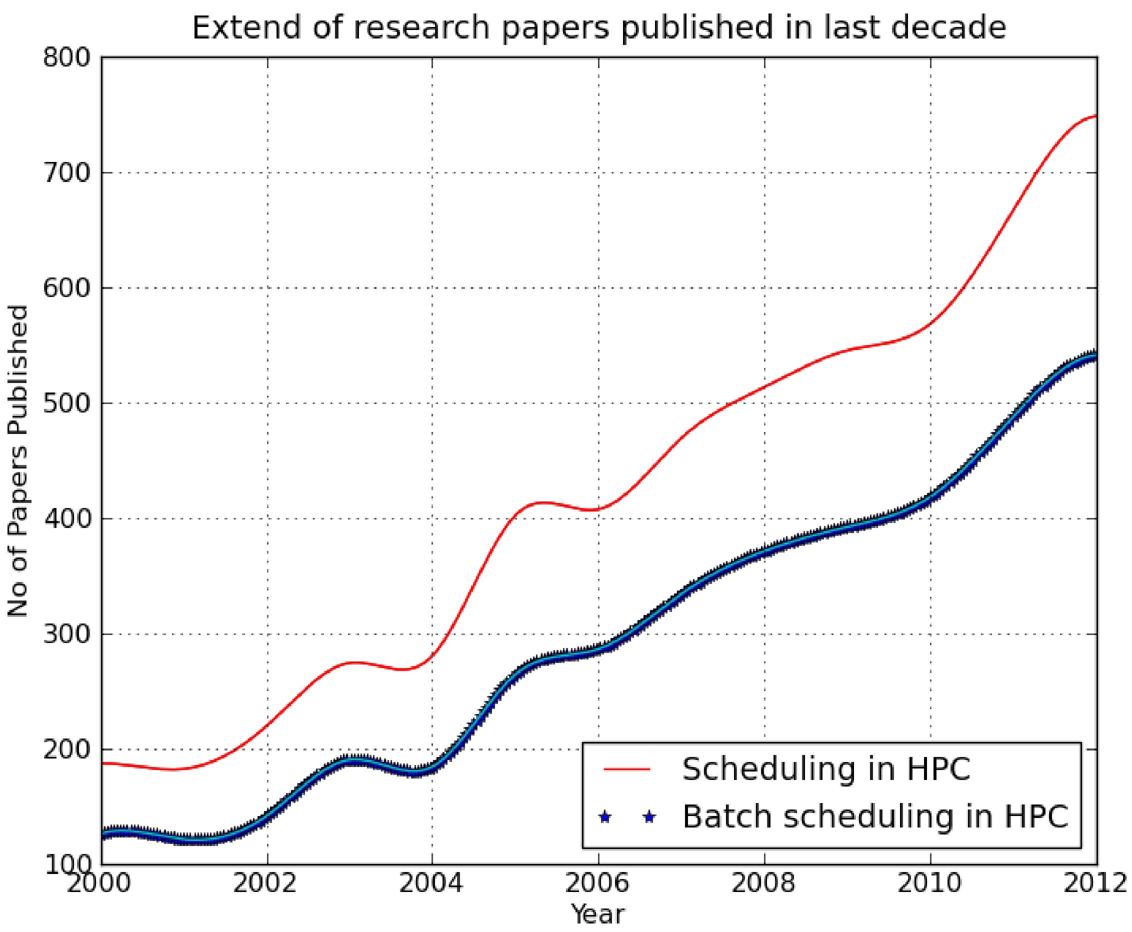

Fig1: Scheduling vs Batch Scheduling – Statistics

A statistics of works, undertaken in scheduling problems during the past decade reveals that most of the research papers published were addressing based on dynamic scheduling. This statistics is illustrated in figure 1.

Dynamic scheduling implies planning over time periods wherein scheduling problems are influenced by internal and external factors at different time levels. Scheduling area can be classified into real-time scheduling, dynamic scheduling and static scheduling. Real-time scheduling determines the sequence of execution of tasks with deadlines. Static scheduling is used when we have extremely time critical threads for which no delays can be accepted. Dynamic scheduling calculates priorities during execution of the system. Its goal is to adapt dynamically changing progress in self sustained manner.

Scheduling is an allocation of processor resources to jobs and its tasks overtime. Better scheduling leads to quality of service. Fault-tolerant scheduling plays a significant role in improving system reliability of clusters. Although extensive fault-tolerant scheduling algorithms have been proposed for real-time tasks in parallel and distributed systems, quality of service (QOS) requirements of tasks have not been taken into account.

2.1 Allocation in batch scheduling

Batch scheduling is a standard method for sharing nodes among high performance computing users.

Batch scheduler is traditional method for executing background tasks based on date and time during which resources were available for batch window. Batch processing collects input data into batches of files and are subsequently processed by the program. Batch systems always hide the complexity of the underlying architecture which enables the user to submit jobs in a processing queue. A batch processor is responsible for managing these jobs, locating and allocating resources needed, staging data and tracking job execution.

With batch scheduling, users submit requests to run jobs, which are placed in a queue and waits to be granted an allocation. The job has exclusive access to these nodes for a bounded duration. In the case of batch scheduling, it limits the overall resource utilization [1]. i.e., when a job utilizes only a fraction of resources then rest will be wasted. Also the use of integral resource allocation without any time-sharing between nodes [3], [2] postpones incoming jobs even while some nodes remain idle. Another drawback of batch scheduling point outs lack of user concerns i.e. is certain parameters are not directly related to relevant user-centric metrics [2].

2.1.1 Scheduling by fractional resource allocation

The paper [6] addresses remedy for batch scheduling problems by allocating fractional resources that can be modified by changing allocated resource fractions and by migrating job tasks to different nodes instead of pausing any jobs. It also defines objective performance metrics and develops algorithms that attempt to optimize batch scheduling. Certain parameters like stretch and yield determine the prioritization of batch execution. With the help of greedy task mapping method the authors propose to reschedule jobs selected for pausing, thereby avoids starvation of any newly submitted shortcoming jobs. Resource allocation considers maximum stretch which defines deadlines based on job's release date and processing time. Dynamic Fractional Resource Scheduling (DFRS) algorithm leads to a dramatic improvement over batch scheduling in terms of maximum stretch. Resource allocation decisions must be based on estimates of jobs' resource needs. One simple solution is to use virtual machine (VM) instance monitoring mechanisms. VM monitor can limit its resource usage. All VM monitors are in turn under the control of a VM Manager that specifies resource usage constraints for all instances. The VM Manager can also pre-empt instances, and migrate instances among physical nodes. Users submit job requests to the cluster. Each job consists of one or more tasks to be executed in parallel, each task running within a VM instance. This paper aims to design algorithms to make sound resource allocation decisions.

2.1.2 Scheduling with setup times

Another major work that focused in batch scheduling is based on set up time. The solution of classical batch scheduling problem with identical jobs and setup times to minimize flow time is known for around twenty five years. The importance and applications of scheduling models with explicit considerations of setup times have been discussed in several studies since the mid-1960s.

It can be done in two approaches; one is by constructing optimal batches and other by using uniform parallel machines.

In two uniform batch scheduling problem [5], they construct batch sizes using decreasing arithmetic sequence on each of the machines. Also proposes a simple greedy rounding procedure to obtain integer batch sizes. Here an optimal number of batches are obtained by forcing the last term of decreasing arithmetic sequence to be non-negative. The authors proposes an optimal algorithm which defines the upper bound on number of batches on both machines which reduces the total computational efforts to O(n).The authors studied single machine scheduling problem by keeping two competing agents where each of agents needs to process a set of jobs in order to optimize its objective function, minimizing flow time [8]. In this paper they proposes an optimal solution to minimize the overall flow time by keeping upper bound on the flow time of second agent as that of minimum flow time of first agent. The solution procedure introduced in this paper requires no more than an effort of O(n) time.

2.1.3 Scheduling by minimizing makespan

In scheduling procedure, minimizing setup time leads to minimization of flow time. In the case of a single batch processing system, the problem of minimizing make span leads to processing of multiple jobs in the machine simultaneously. The paper [7] investigates this problem. This paper focuses on the context of parallel batch scheduling. The processing time of a batch is equal to the longest processing time of all the jobs in that batch. Once a batch is being processed, the batch-processing machine cannot be interrupted; no jobs can be removed from the machine until the process is completed. Each job is characterized by release time, processing time, and job size. The problem study is formulated as mixed integer programming model. Based on this model the aim to minimize completion time can be defined to Minimize $C_{max} = S^k + P^k$. i.e., the make span for a batch processing machine equal to completion time of latest batch processed. The authors proposes a constructive based meta heuristic method, Ant Colony Optimization which generate solutions from scratch by adding solution components iteratively to an initially empty solution set until the feasible solution is obtained. They also considers the waste and idle space of batch b, WIS(S) which equals to the sum of the waste space and the idle space of batch b. The smaller WIS(S) gives better solution, so they construct batch job to reduce WIS(S) of batch jobs together. It also introduces a candidate list strategy to restrict the number of available choices to be considered at each construction step. The authors recommend a new heuristic method to construct heuristic information which exploits the specific knowledge from the problem which is used in combination with pheromone trails to build solution together. The heuristic information is constructed using the knowledge of minimizing the C_{max} under solution building procedures.

2.2 Allocation using Dispatching rules

A widely used approach to real-world scheduling, in which problems are often characterised by a highly complex and dynamic environment, involves use of dispatching rules [12].

Dispatching rules are simple heuristics. This implies that whenever a machine is available, it determines the job with the highest priority among the jobs waiting to be processed next on that machine. The computation of priorities is typically based on local information, which allows dispatching rules to be executed quickly, irrespective of the complexity of the overall problem. Though each scheduling decision is taken at the latest moment, it is able to react with dynamic changes. Dispatching rules take scheduling decisions on the basis of current local conditions without assessing the negative impact a decision might have on the decision-making at other work centres in the future. Many research works in the past had proposed alternative methods for automatic creation of dispatching rules.

3. An Overview of works in scheduling

Table 1 summarizes work strategies in batch scheduling.

Table1 : Summarization of works in Batch scheduling

No	Paper Name	Work Done	Future Scope	A Review
1	Dynamic Fractional Resource Scheduling versus Batch scheduling	* Do fractional resource allocations that can be modified by changing allocated fraction and by migrating job tasks to different nodes. * Defines objective performance metrics and develops algorithms that attempt to optimize it.	Scheduling algorithms should be improved with a strategy for reducing the yield of long running jobs. This method can be extended to heterogeneous system. Can implement mechanisms to include user priorities.	Due to heuristics approach It can allow a newly submitted short-running job to be stalled Indefinitely.
2	Predictable High Performance Computing Using Feedback Control and Admission Control	* Designs an Admission Controller which relies on prediction About a job's resource requirements based on priority and deadline. * Designs a Feedback Controller – CPU scheduler which regulates the Scheduling based on job's progress.	Support to parallel jobs, VM performance overhead, and Control stability.	The approach is better for the Jobs that are deadline sensitive. Though its batch scheduling, the remaining (idle) CPU cycles from deadline scheduler can be used for the jobs that do not require firm deadline guarantee.

3	Tackling Resource Variations Through Adaptive Multicore Execution Frameworks	* Adaptive scheduling and core reordering scheme incorporates with static multicore schedules so as to predictably tolerate runtime core Degradations due to unpredictable device failures, thermal stress, resource competitions or preemptions.	These approaches deliver a framework capable of intensive static analysis that complements dynamic reconfigurations in the face of resource variation-based unpredictability in future multicore systems.	An application with strong intertask dependencies will limits the possible reordering Choices.
4	Complexity Analysis of Checkpoint Scheduling with Variable Cost	* Designs a performance model which exhibits the impact of the Checkpoints operations and the lost computation due to failures. * Analyze computational complexity of the problem of scheduling checkpoints with variable costs for general failure distributions.	Can include advanced failure detection mechanisms into the proposed scheme. Can focus on eventual performance gaps between schedules obtained with the different failure distribution models.	Non preallocated checkpoint scheduling and block scheduling are strongly Interleaved.
5	Supporting HPC Analytics Applications with Access Patterns using Data Restructuring and Data-centric Scheduling Techniques in MapReduce	* MRAP data restructuring is provided to improve the performance of access patterns which access data non-contiguously in small regions to Perform analysis. * MRAP data centric scheduling scheme reduces amount of data chunks transfers over a network, based on two scheduling approach virtual split and weight set cover problem.	Develop more real world HPC data analytical applications using MRAP and also explore new Applications with their different access patterns.	If the I/O access pattern of the application is not consistent over different executions of same analysis operation then Such applications will not benefit much.
6	Probabilistic Reservation Services for Large-Scale Batch-Scheduled Systems	* Describes new methods for reducing uncertainty with queuing delays in batch jobs - Queue bounds estimation from time series - Virtual advanced reservations for queues - Coallocation service that uses VARQ	The parameter chosen are extremely aggressive by design which will increase the cost and this should be explicitly managed.	Priority of paused job has to be changed and reconfigure advanced reservation system in periodic intervals.
7	Impact of Workload and System Parameters On Next Generation Cluster Scheduling Mechanisms	* Evaluation of numerous parallel coscheduling mechanisms using a Unified framework. * Heuristics approach gives a better yield without compromising performance	Can extend the work to show how buffering and latency tolerance can be used to minimize the impact of non coscheduled execution	Guaranteed Quality-of-Service is expected when workloads are communication intensive.

		benefits		
8	Improving Performance in HPC Centers by Modeling Users Through an Evolutionary Virtual Interface	* Develops a user behavior modeling system which learns user behavior in a way that allows predicting the real usage of system resource to achieve more accurate scheduling	Can improve by adapting the schedule dynamically to the fact that every user has a different perception of what a good service is.	Obtain better and consequently more useful values of resource request.
9	A Self-guided Genetic Algorithm for permutation Flowshop scheduling problems.	* Proposes an algorithm by applying it to deal with the NP-complete flowshop scheduling problem to minimize the makespan. * Compares different self-guided GA algorithms	Enhancing the proposed algorithm by using problem-specific knowledge.	Broadening the idea to deal With multi-objective optimization Problem.
10	Timely Result-Data Offloading for Improved HPC Center Scratch Provisioning and Serviceability	* Design a combination of both a staged as well as a decentralized offloading scheme for job output data. * Develop a decision making component that factors in parameters such as a center's purge deadline, user delivery schedule and a snapshot of current network conditions between the center and the end user, to determine the most suitable approach to offload.	Explore this approach with strategic placement, selection of intermediate nodes between an HPC center and end-user destinations.	The offloading approach effectively utilizes orthogonal residual bandwidth and can serve as an alternative to direct transfers, which may not always be feasible, optimal, or fault-tolerant.
12	Job Scheduling for Multi-User MapReduce Clusters	* Designed a model (FAIR) for MapReduce cluster, that provides isolation, guarantees a minimum share to each user, and achieve statistical Multiplexing	Aspects of MapReduce such as data locality and task interdependence inverses the performance.	Delayed scheduling gives better options

4. PROPOSED SYSTEM

A new approach is proposed which seeks remedy for batch scheduling by creating optimal batch sizes and by selecting better node for execution, that is, which has minimum execution time.

Algorithm

1. Generate an empty batch as the current batch b, where processing time and release time of this batch is zero.
2. Allocate unscheduled jobs in ready queue to the created batch based on Batch-Size rule.
3. Continue with steps 1 & 2 until all jobs got allotted in any batch.
4. For each batch calculate average execution time and based on it rank the batches to prioritize for execution.
5. If the rank < threshold then assign batches ascending in a local queue else in global queue.
6. If global queue is not empty then map the jobs from batch in the ordered list to processor with minimum execution time else do mapping for local queue.
7. If a process completes its execution calculate computer performance parameters such as waiting time, turnaround time, slowdown time and total completion time.

Batch size rule states that a batch can start its execution, if the fraction of maximum number of jobs with processor execution capacity is greater than or equal to a critical number.

The nodes that having minimum execution time can be find by assigning grades for each nodes based on the parameters involved in QOS metrics. QOS will be the weighted average based on the relative grade of these parameters. The node with highest grade will have maximum utilization. CPU load, CPU speed, memory usage, fan speed such parameters can be included in QOS metrics.

5. CONCLUSION

In this paper we reviewed the literature on scheduling in parallel processors. Resource utilization and job scheduling is the fundamental criteria for achieving high performance in a parallel computing environment. In order to achieve the goal of high-performance, they need to adaptively utilize their computational and communication resources. Our approach that uses QOS is a critical input for node selection. The QOS is based on various measurable parameters of the nodes in a cluster.

REFERENCES

[1] C.B. Lee and A.E. Snavely, "Precise and Realistic Utility Functions for User-Centric Performance Analysis of Schedulers," Proc. 16th Int'l Symp. High Performance Distributed Computing (HPDC), pp. 107-116, 2007.

[2] D.G. Feitelson "Parallel Workloads Archive, "http://www.cs.huji.ac.il/labs/parallel/workload/, 2005.

[3] S.K. Setia, M.S. Squillante, and V.K. Naik, "The Impact of Job Memory Requirements on Gang-Scheduling Performance," ACM SIGMETRICS Performance Evaluation Review, vol. 26, no. 4, pp. 30-39, 1999.

[4] C.T. Ng , T.C.E. Cheng , Mikhail Y. Kovalyov, Ali Allahverdi, " A survey of scheduling problems with setup times or costs", European Journal of Operational Research 187 (2008) 985–1032.

[5] Baruch Mor, Gur Mosheiov, "Batch scheduling on uniform machines to minimize total flow-time",Computers & Operations Research 39 (2012) 571–575.

[6] Mark Stillwell, Frederic Vivien, and Henri Casanova, "Dynamic Fractional Resource Scheduling versus Batch Scheduling ", IEEE Transcations on Parallel and Distributed Systems, vol. 23, no . 3, March 2012.

[7] Rui Xu, Huaping Chen, Xueping Li, "Makespan minimization on single batch-processing machine via ant colony optimization",Computers & Operations Research 39 (2012) 582–593.

[8] Baruch Mor, Gur Mosheiov,"Single machine batch scheduling with two competing agents to minimize total flowtime",European Journal of Operational Research 215 (2011) 524–31.

[9] http://www.acpi.info/

[10] Kai Li, Ye Shi , Shan-lin Yang , Ba-yi Cheng , "Parallel machine scheduling problem to minimize the makespan with resource dependent processing times ",Applied Soft Computing 11 (2011) 5551–5557.

[11] Eugene Levner , Vladimir Kats, David Alcaide López de Pablo , T.C.E. Cheng, "Complexity of cyclic scheduling problems: A state-of-the-art survey", Computers & Industrial Engineering 59 (2010) 352–361.

[12] Christoph W. Pickardt , Torsten Hildebrandt, Jurgen Branke , Jens Heger , Bernd Scholz-Reiter, " Evolutionary generation of dispatching rule sets for complex dynamic scheduling problems", Int. J. Production Economics, 17 Febraury 2012 .

[13] Daniel Nurmi, Rich Wolski, and John Brevik, "Probabilistic Reservation Services for Large- Scale Batch-Scheduled Systems", IEEE Systems jouurnal, Vol. 3, no. 1,March 2009.

[14] Amina Haned , Ameur Soukhal, Mourad Boudhar , Nguyen Huynh Tuong , " Scheduling on parallel machines with preemption and transportation delays", Computers & Operations Research 39 (2012) 374–381.

[15] Sang-Min Park, Marty A. Humphrey, " Predictable High-Performance Computing Using Feedback Control and Admission Control",IEEE Transactions on Parallel and Distributed Systems, Vol. 22, no. 3, March 2011.

[16] Bernholdt, D.E., Armstrong, R.C., Allan, B.A.: Managing complexity in modern high end scientific computing through component-based software engineering. In: Proc. Of PCA Workshop on Productivity and Performance in High-End Computing (P-PHEC 2004),Madrid, Spain, IEEE Computer Society (2004).