Parallel Job Scheduling with Time-varying Constraints for Heterogeneous Multiple-Cluster Systems

Weizhe Zhang\textsuperscript{a,c}, Hongli Zhang\textsuperscript{a}, Xinran Liu\textsuperscript{b}, Xuemai Gu\textsuperscript{c}

\textsuperscript{a} School of Computer Science and Technology, Harbin Institute of Technology, Harbin, 150001, China
\{zwz, zhl\}@pact518.hit.edu.cn
\textsuperscript{b} National Computer Network Emergency Response Technical Team/Coordination Center of China, Beijing 100029, China
lxr@mail.nisac.gov.cn
\textsuperscript{c} Communication Research Center, Harbin Institute of Technology, Harbin 150001, China
xmgu@hitcrc.hit.edu.cn

Abstract

Parallel job scheduling in multiple cluster system is a critical aspect to facilitate the processor cooperation. However, co-allocation, heterogeneity and time-varying emerge as tough challenges for the design of multiple cluster job scheduling models and algorithms. This paper presents a new multiple cluster job scheduling scheme based on the time-varying job performance model for a dynamic, heterogeneous multiple cluster environment. Four multiple cluster scheduling algorithms based on different heuristic resource selection strategies are introduced. Experiments indicate that the scheduler and the algorithms are effective and perform better than the single cluster and other multiple cluster scheduling algorithms.

Keywords: Network computing, multi-cluster system, job scheduling, resource selection, time-varying

1. Introduction

Parallel Job scheduling is a complex problem, even in a single parallel computer. However, multiple cluster systems or the Grid\textsuperscript{[1]}, compared to the classical parallel computers, pose several technical challenges that introduce an additional degree of complexity to the scheduling problem while amplifying the existing ones. Therefore, it is necessary to point out the intrinsic nature of multiple cluster job scheduling that is different from parallel computer scheduling as follows:

1) Co-allocation or multisite scheduling: a multiple cluster system is typically composed of several clusters from geographically distributed organizations. Parallel jobs should be scheduled to spread to more than one clusters in order to run simultaneously on several clusters without considering the resource limitation from one single cluster. The scheduling algorithm should be capable of coordinating these resources from different clusters;

2) Heterogeneity: in a real-world scenario, hardware and software resources from different clusters may have a rich diversity. Heterogeneous scheduling issues are highlighted which simply do not occur in "single-chassis" sequential or parallel machines;

3) Time-varying: Previous studies in algorithms for multiple cluster scheduling use a fixed-penalty model for scheduling across cluster boundaries. Actually, this type of characterization is not sensitive to the time-varying contention for bandwidth in the inter-cluster communication links and it impacts the execution time of co-allocated jobs that share network resources. An alternative bandwidth-centric job performance model should be considered taking into account time-varying network utilization to capture the interaction and impact of simultaneously co-allocating jobs across multiple clusters.

Based on the above essences of the multiple cluster systems, this paper concentrates on the heterogeneous multiple clusters scheduling with time-varying constraints. We propose a time-varying performance model for multiple cluster jobs. A multiple cluster scheduling scheme and four novel time-varying scheduling algorithms are introduced.
The organization of this paper is as follows. We introduce related works and clarify our motivation in section 2. Section 3 presents the multiple cluster system model and time-varying job performance models. The multiple cluster scheduling schema and four new algorithms are discussed in detail in section 4. Section 5 defines the experiment set and performance metrics used for evaluation. Initial performance comparison under different scenarios is presented in section 6. Finally, we conclude this paper and give a discussion in section 7.

2. Related works and our motivation

Considerable research has been conducted over the last decade on the topic of job scheduling for parallel systems. Much of this research has been presented at the annual Workshops on Job Scheduling Strategies for Parallel Processing [2] and the International Heterogeneous Computing Workshop [3]. Moreover, Feitelson, Rudolph and Schwiegelshohn have written a couple of surveys [4, 5] to report the current art and state for parallel job scheduling on the supercomputer. Recent trends for parallel job scheduling in workstation clusters are summarized in [6], where the parallel job scheduling on the multiple clusters plays more and more important role.

More recently, Ernemann, Hamscher and Yahyapour [7] perform simulations to evaluate the effects of a global grid constituted by the compute centers. Their results show that the average weighted response time of all submitted jobs decrease up to about 30% for a global grid. In [8], the authors analyzed the problem of executing a parallel application on a multiple cluster environment. They presented some simulations where multisite execution was beneficial compared with job-sharing, even for an additional communication overhead of about 25%.

Another important research about co-allocation or multisite site scheduling is presented in [9-13]. Bucur and Epema [9] assess the influence on the mean response time of the job structure and size, the sizes of the clusters in the system, the ratio of the speeds of local and wide-area communications, and of the presence of a single or of multiple queues in the system.

Most of the scheduling algorithms described above cover only part of the nature of the multiple clusters systems. First, many scheduling algorithms neglect the possibility of jobs co-allocation across the boundary of clusters [8]. Second, all the clusters from different domains are assumed to be homogenous without taking the heterogeneity into consideration [9-13]. Moreover, less attention is paid on the dynamic characterization of job communications. Previous work tends to characterize jobs by assigning a fixed execution-time penalty [7-13], which is not sensitive to the time-varying contention for the inter-cluster communication links. This research aims to extend our previous work presented in [14] and [15] by replacing the static communication model with a more dynamic job performance model in heterogeneous multiple clusters.

3. Time-varying models for heterogeneous multiple cluster

In this section, we introduce our time-varying model for the multiple clusters. The section is organized as follows. Section 3.1 shows the architecture of multiple cluster scheduling system. In section 3.2, the time-varying performance models of job running across the multiple clusters are discussed.

3.1. Architecture of multiple cluster scheduling system

Models for multiple cluster systems can be divided into five parts: job, cluster, local scheduler, domain scheduler and meta-scheduler models as shown in figure 1.

![Fig 1. Architecture of multiple cluster scheduling system](image)

(i) Job model. In our model, a parallel job requires more than one node and unlike independent sequential jobs, it requests processors simultaneously in more than one cluster. A task as part of a job runs a single node. Tasks communicate by exchanging messages over the network. All tasks of a job start and finish in the same time, which implies that all the nodes allocated to a job are being simultaneously occupied and released. The jobs from all the users are independent and do not have precedence constraints.

A job shares the nodes in a space-sharing rather than in a time-sharing fashion. Jobs are rigid without
preemption. Furthermore, our job model is limited for batch jobs, which are dominant on most MPP systems.

(ii) Cluster model. We model a multiple cluster system consisting of geographically distributed and independent clusters. We assume that each participating cluster is a massively parallel processor system (MPP), which consists of several nodes. Each node has its own processor, local memory, and hard disk. The nodes are homogeneous in the same clusters but the nodes in different clusters have different speeds and quantity. As to the communication network, we assume that all intra-cluster communication links have the same capacity, as all the inter-clusters are linked. Since LAN links are faster than WAN links, we assume that the speed of intra-cluster links is significantly higher than that of inter-cluster links.

(iii) Local scheduler. The local scheduler is responsible for reporting the status of the clusters, starting the jobs after allocation by the domain scheduler and collecting the execution results for local users. They do not participate in the decision for job scheduling.

(iv) Domain scheduler. The domain scheduler receives the job requests from different clusters and gets the clusters’ information from local schedulers. Our multiple cluster scheduling algorithms in the domain scheduler are responsible for allocating and mapping the job requests to the nodes. After the scheduling decision is made, the domain scheduler informs the local schedulers to deploy and transfer jobs to the destination nodes.

(v) Meta-scheduler. In order to efficiently leverage the collective computational power of a multi-cluster, special scheduling agents are required to balance the workload for different domains. We refer to these schedulers as meta-schedulers. In general, we consider a meta-scheduler to be collection of software, that decides where, when, and how to keep the multiple cluster systems in balance.

3.2. Job performance model for heterogeneous multiple cluster systems

An accurate performance prediction model is the foundation of scheduling algorithms’ design and evaluation. In this subsection, we introduce the time-varying job performance model in a heterogeneous multiple cluster systems.

The total running time of job \( T \) includes waiting time in queue \( T_W \), execution time across different clusters \( T_E \), and communication time \( T_C \), which presented as follows:

\[
T = T_W + T_E + T_C
\]

Assume the number of sites is \( N \). The job executes simultaneously at clusters \( j, k, \ldots, m \), where \( 1 \leq i, j, k, m \leq N \). The heterogeneity of different clusters is denoted by a heterogeneous factor \( h_0 \), where \( 0 < h_0 \). Also, we assume that the execution time of the job on some cluster is \( T_{i} \), whose heterogeneous factor is \( h_i \). Consider all the tasks of a job are terminated at the same time. Thus, the predicted execution time \( T_E \) should be determined by the worst performance cluster as follows:

\[
T_E = T_0 \times \max(h_i, h_k, \ldots, h_m)/h_0
\]

The communication time \( T_C \) is affected by frequencies of trigger events, job communication models, network topology between different clusters. When running jobs terminate or new jobs are triggered to run, the bandwidth between different clusters is released or congested. The residual time of running jobs is affected. Thus, the communication model of the job should be dynamic and time-varying. Based on the above consideration, the time-varying model of job communication time on multiple cluster systems is presented as follows:

\[
T_C = \sum_{i=1}^{n} \Delta T_c^i + T_c^{(Re,i)}
\]

Each job modeled in this study performs all-to-all global communication patterns. The job \( K \) requires the \( n \) nodes in total, \( n_i \) means the nodes inside the cluster with the link \( j \), and the bandwidth between nodes is denoted as \( BW_{BN} \). The total bandwidth of link \( j \) occupied by the job \( K \) is denoted as \( BW_{j}^{i} \):

\[
BW_{j}^{i} = BW_{BN} \times n_j \times (n - n_j)/(n - 1)
\]

\[
BW_{const}^{j} = \text{const bandwidth of link } j \text{. When the event } i \text{ is triggered, the congestion factor is}
\]

\[
CF_{j}^{i} = BW_{const}^{i} \sum_{i=\text{link}} BW_{j}^{i}
\]

If \( CF_{j}^{i} > 1.0 \), the residual communication time \( T_{c}^{Re,i} \) will not be affected, otherwise \( T_{c}^{Re,i} \) should be determined by the worst \( CF_{j}^{i} \):

\[
FF_{i} = \begin{cases} 1 & \text{if } \forall \text{ Link } \text{} \min (CF_{j}^{i}), \text{} \text{ if } 3 \text{ Link } \text{} \text{CF}_{j}^{i} < 1.0 \end{cases}
\]
4. Multiple cluster scheduling algorithms

This section is organized as follows. The scheme of multiple cluster scheduling algorithms will be introduced in the section 4.1. The resource selection algorithms, which are the key components, will be presented in the section 4.2.

4.1. Multiple cluster scheduling algorithm scheme

From the formula (1) to (3), the important factors affecting the total running time $\Delta T$ are waiting time in queue $T_w$, the heterogeneous factor $h_i$ and the communication cost $BWBN$. Thus, in this section a multiple cluster scheduling algorithm scheme is proposed to decrease the waiting time in queue $T_w$, choose the lower heterogeneous factor $h_i$ with the bandwidth-awareness.

**Algorithm 1.** Multiple cluster scheduling algorithm scheme

**Input:** (1) Job queue (2) Clusters aggregate
**Output:** (1) Mapping results

**Variables:**
1. $\Delta t$ inter-schedule interval
2. $Currentjob$ first unmapped job in the job queue
3. $AssignedCluster$ indicate the resource selection results
4. $CurrentJob\rightarrow next$ the first unmapped job in the current job queue
5. $Allocatedjob()$ based on the time-varying job performance model.

1. (Initialization) Check the status of the job running queue. If some job finished, release the resources, go back to step 1; otherwise, updating the remaining time of jobs in queue.
2. (Updating) if the job queue is empty, then wait $\Delta t$ interval and recheck the status of the job queue; otherwise, collect the clusters’ state from the local schedulers and the job requests from the job queue.
3. (Mapping) for each unmapped job request in the current job queue
   a. $CurrentJob\leftarrow$ the first unmapped job in the current job queue
   b. $AssignedCluster\leftarrow Allocatedjob(CurrentJob)$
   c. $Return$ the mapping results and go back to step 1.

4.2. Resource selection algorithms

In this section, we propose four multiple cluster resource selection sub-algorithms in the procedure $Allocatedjob()$ based on the time-varying job performance model.

**Algorithm 2.** Multiple cluster resource selection schema

**Input:** (1) $CurrentJob$ (2) Clusters status
**Output:** (1) Mapping results of the best resource allocation

**Variables:**
1. $\Delta t$ inter-schedule interval
2. $Currentjob$ first unmapped job in the job queue
3. $AvailNodes$ the number of free nodes in the cluster
4. $HeterRate$ the heterogeneous factor of the cluster
5. $Saturation$ the degree of network saturation
6. $SortedQueue$ the clusters queue sorted according to the $AvailNodes$, $HeterRate$ or/and $Saturation$
7. $RemainNodes$ the nodes required by the $Currentjob$

1. (Initialization) (a) If the $Currentjob$ is the end of the waiting queue, then quit. If the $Currentjob = null$, then $Currentjob = Currentjob\rightarrow next$;
   b. Sort the clusters queue according to the $AvailNodes$, $HeterRate$ or/and $Saturation$, and initialize the $SortedQueue$
   c. Set the $RemainNodes$ = the nodes request of $Currentjob$
2. If the request of $Currentjob$ exceeds the total node number from all sits, then drop the job and go to step 1.
3. If the node request of $Currentjob$ exceeds all the idle node number, then go to step 1.
4. (Selection) searching in the $SortedQueue$, if the idle nodes of current cluster are smaller than the $Currentjob$’s request, then $RemainNodes = RemainNode - AvailNodes$; Allocate all the idle nodes in the cluster to the $Currentjob$
   b. Searching in the $SortedQueue$, if the idle nodes of current cluster are larger than the $Currentjob$’s request, then allocate the $AvailNodes$ in the cluster to the $Currentjob$
5. (Updating) Update the status of the cluster queues, and go to step 1.

When the cluster queue $SortedQueue$ is sorted according to the $AvailNodes$, the heuristic scheduling algorithm is called **Biggest Free Node Priority**. The second algorithm is called **Smallest Heterogeneous Factor Priority** sorted according to the $HeterRate$. The third algorithm is called **Smallest Network Congestion Priority** sorted according to the
Saturation. At last, the forth algorithm is called Smallest Heterogeneous Free Node Priority sorted according to all these three factors.

The computational complexity of the algorithms depends on the resource selections. They reduce the complexity by adopting the heuristic and an upper bound on its complexity is $O(n\log(n)+m)$, $n$ is the number of clusters, $m$ is the number of the job in the queue.

5. Experiments setup

A discrete event simulation environment is developed to study the effect of multiple cluster scheduling algorithms about the different job input cases. The configuration of the clusters and network are based on a real-world multiple cluster platform “Virtual Computing Environment”, which is supported by the National Grand Fundamental Research 973 Program of China.

We configure three clusters named Beijing (360 nodes), Shanghai (128 nodes) and Harbin (64 nodes) with a total of 552 nodes. The bandwidth capacity is 1,000Mbps within a single site while that varies from 10Kbps to 10Mbps between different sites. The network topology connecting different clusters is star style. The inter-schedule interval $\Delta t$ is set to one second. The job logs are from the collection of workload logs, Dror Feitelson’s archive. We select the first 10,000 job subnet of the Cornell Theory Center (CTC workload), based on an IBM RS6000/SP parallel computer with 430 nodes.

We use the job completion time (Makespan) as the metric to evaluate the algorithms performance:

$$\text{Makespan} = \max(T_1,T_2,\ldots,T_n,T_j)$$  \hspace{1cm} (8)

6. Experiments results and performance evaluation

In this section, we evaluate the performance of BFNP, SHFP, SNCP, and SHFNP algorithms against the single cluster algorithms (SCCA) proposed in [8, 14] and another two multiple cluster algorithms (No-Share, Idea). The performance is evaluated on the Makespan metric.

We assume that the heterogeneous factors of three clusters are 1.0, 2.0 and 4.0 respectively. The average node request of jobs is 40 nodes.

Figure 3 (a) shows that SCCA has best performance in total, the worst is the No-Share algorithm which is nearly three times than SCCA algorithm. Figure 3 (b) shows that in the four time-varying heuristic scheduling algorithms; BFNP has the best makespan than the other three. When the BNBW is lower than 80M, the makespan of BFNP even perform better than the Idea algorithm. Because BFNP algorithms first choose the cluster who owns the largest free nodes, it makes the highest possibility of the job to run on the single cluster and decrease the network saturation deeply. The makespan of SHFNP is better than SHFP and SNCP. Also, figure 3 (b) shows that when the BNBW is lower than 40M, almost all the heuristic algorithms perform better than the Idea algorithm. Considering the Idea algorithm does not adopt the time-varying performance model, some difference will be introduced, so our time-varying algorithms show potential better performance.
7. Conclusion and future work

In this paper we point out that co-allocation, heterogeneity and time-varying are the intrinsic nature of multiple cluster scheduling different from parallel computer scheduling. A multiple cluster system model is introduced in the real-world scenario, which allows the jobs to run across site boundaries. Dynamic job performance model with time-varying constraints is introduced. Four multiple cluster scheduling algorithms are proposed to adaptively select and map jobs to heterogeneous resource combinations. Initial experimental results show that the four algorithms scales well and perform better than the other single cluster and multiple cluster scheduling algorithms in makespan metric.

This work is just a first step to exploit the nature of multiple cluster scheduling and there are still many works remaining for further exploration. For example, many systems are connected to the multiple clusters, so the continuous availability and work must be guaranteed.

Acknowledgment This work was supported in part by the Natural Science Foundation of China under Grant No. 60703014, the National Grand Fundamental Research 973 Program of China under Grant No.G2005CB321806, the National High-Tech Research and Development Plan of China under Grant No.2006AA02Z334, the Development Program for Outstanding Young Teachers in Harbin Institute of Technology under Grant No. HITQNJIS.2007.034, the Specialized Research Fund for the Doctoral Program of Higher Education(SRFDP) under Grant No. 20070213044, the China Postdoctoral Science Foundation under Grant No. 20070410263, the Hei Long Jiang Postdoctoral Foundation under Grant No. LBH-Z07108

References

http://www.cs.umass.edu/~rsnbrg/hcw2005/hcw05_prev_workshops.html

1848