A Scheduling Method for Divisible Workload Problem in Grid Environments

Nguyen The Loc
Japan Advance Institute of Science and Technology
1-1, Asahidai, Nomi, Ishikawa, Japan 923-1292
lnguyen@jaist.ac.jp

Said Elnaffar
College of IT, UAE University
Al-Ain, UAE
elnaffar@uaeu.ac.ae

Takuya Katayama, Ho Tu Bao
Japan Advance Institute of Science and Technology
1-1, Asahidai, Nomi, Ishikawa, Japan 923-1292
{katayama, bao}@jaist.ac.jp

Abstract

Scheduling divisible workloads in distributed systems has been one of the interesting research problems over the last few years. Most of the scheduling algorithms previously introduced are based on the master-worker model. However, the majority of these algorithms assume that workers are dedicated machines, which is a wrong assumption in distributed environments such as Grids. In this work, we propose a dynamic scheduling methodology that takes into account the three prominent aspects of Grids: heterogeneity, dynamicity, and uncertainty. Our contribution is threefold. First, we present an analytical model for processing local and Grid tasks at each non-dedicated worker. Second, we present a simple prediction method to forecast the available CPU capacity and bandwidth at each worker. Third, we introduce a dynamic, multi-round scheduling algorithm.

Keywords: divisible tasks; dynamic scheduling algorithm; multi-round algorithm; Grid computing; performance prediction.

1 Introduction

A critical issue for the performance of a Grid is the task-scheduling problem, i.e. the problem of how to divide an application's workload into many parts and assign them to computers of the Grid, here thereafter called workers, so that the execution time is minimum.

Many algorithms for scheduling divisible workloads [1, 2, 8, 10] assume that computational resources are dedicated. This assumption renders these algorithms impractical in distributed environments such as Grids where computational resources are expected to serve local tasks in addition to the Grid tasks. Another shortcoming in these algorithms is that they do not take the dynamicity of Grids into account.

Our contribution is threefold. First, we present a model to represent a worker's activity with respect to processing local and external Grid tasks. Unlike the work done in [1, 2, 8, 10], this model help estimate the computing power of a worker under the fluctuation of number of local and Grid applications in the system. Second, we provide a simple method for predicting the computing power of processors, i.e. the portion of original CPU power that the owner can donate to Grid applications. Third, we incorporate the performance model and the prediction method described above into the UMR (Uniform Multi-Round) algorithm [8], which is originally a static scheduling algorithm.

The rest of this paper is organized as follows. Section 2 reviews some of the static and dynamic scheduling algorithms. Section 3 briefly describes our heterogeneous computation platform. Section 4 introduces our dynamic scheduling methodology. Section 5 concludes the paper and sketches future work.

2 Related work

Single round algorithm [1, 3] is the early and most simple way for the scheduling problem. As showed in [1], for a large workload, the single-round approach is not efficient due to a large idle timing suffered by the last worker to receive its chunk. Multi-round scheduling algorithm was introduced firstly in [2] but authors assume that the computation and communication startup times are zero, therefore this algorithm does not reflect correctly the real conditions in Grids. The studies in [1, 8, 9, 10] focus on affine model in which computation and communication startup time are different from zero. UMR [8] is the only algorithm that computes the approximately optimal number of rounds and the sizes of workload chunks. In fact, our method is inspired by the UMR model [8].

All above static algorithms assume that the performance of workers are stable during execution, which make them impractical for Grid applications. RUMR [9] is designed to tolerate performance prediction errors by using Factoring method, however all of its parameters are fixed before RUMR starts, which makes RUMR a non-adaptive scheduling algorithm. Apparently, dynamic algorithms [6, 7, 11] are more appropriate for Grids. Our method falls in this category, and in our knowledge, it is the first dynamic method for divisible workload. In [11], the authors use M/M/1 queue to model the tasks processing, however [11] lacks an efficient prediction strategy because it is merely based on probability parameters. On the other hand, the efforts in [6, 7, 11] are not for concerned with divisible workloads.

3 The Divisible workload scheduling problem in Grid environments

Let us consider a computation Grid, in that, a master process controls N worker processes and each process runs in a particular computer. We denote the total workload by W_{total}, the master can divide it into arbitrary chunks and delivers them to appropriate workers. We assume that the master uses its network connection in a sequential fashion, i.e., it does not send chunks to some workers simultaneously. The communication and computation platforms of our system are heterogeneous. Workers can receive data from network and perform computation simultaneously.

3.1 Notation

- N: the number of workers, M: the number of rounds.
- W_{total}: the total amount of workload.
- $chunk_{j,i}$: the fraction of total workload W_{total} that the master deliver to worker$_i$ in round$_j$ ($i = 1, 2, ..., N, j = 1, 2, ..., M$).
- S_i: computational speed of the worker$_i$, its measure is the number of units of workload performed per second.
- ES_i: estimated average speed of worker$_i$ for Grid tasks on the next round. ES_i is derived from equation (12).
- $Tcomp_{j,i}$: computation time required for worker $_i$ to process chunk$_{j,i}$.
- $Tcomm_{j,i}$: communication time required for master to send chunk$_{j,i}$ to worker$_i$

$$Tcomm_{j,i} = nLat_i + \frac{chunk_{j,i}}{B_i} \tag{1}$$

$$Tcomp_{j,i} = cLat_i + \frac{chunk_{j,i}}{ES_i} \tag{2}$$

where $cLat_i$ is a fixed overhead time for starting a computation in worker$_i$ and $nLat_i$ is the overhead time incurred by the master to initiate a data transfer to worker$_i$.

- B_i: the data transfer rate, of the connection link between the master and worker $_i$.
- $round_j$: the amount of workload

$$round_j = \sum_{i=1}^{N} chunk_{j,i} \tag{3}$$

UMR [8] makes the the time required for each worker to process its workload during a round constant, $const_j$

$$cLat_i + \frac{chunk_{j,i}}{ES_i} = const_j \tag{4}$$

$$chunk_{j,i} = \alpha_i \times round_j + \beta_i \quad (i, j = 1, ..., N) \tag{5}$$

where

$$\alpha_i = \frac{ES_i}{\sum_{k=1}^{N} ES_k} \tag{6}$$

$$\beta_i = \frac{ES_i}{\sum_{k=1}^{N} ES_k} \sum_{k=1}^{N} (ES_k \times cLat_k) - ES_i \times cLat_i \tag{7}$$

3.2 Problem statement

The task scheduling problem in non-dedicated environments can be defined as follows. Given:

- Divisible workload W_{total} that reside at the master

- Non-dedicated computational platform consists of the master and N workers, computational speed of the worker$_i$ is S$_i$ with latency $cLat_i$.
- Data transfer rate of the connection link between the master and worker$_i$ is B$_i$ with latency $nLat_i$
- S$_i$ vary over time ($i = 1, 2, ..., N$). This is nature of non-dedicated environments.

Our ultimate question is: given the above Grid settings, in what proportion should the workload W_{total} be split up among the heterogeneous, dynamic workers so that the overall execution time is minimum? Formally, we need to minimize the following objective function:

$$max_{i=1,2,...,N}\left(Tcomm_{1,i} + \sum_{j=1}^{M} Tcomp_{j,i}\right) \rightarrow min \quad (8)$$

4 The proposed method

Proposed method for this problem consists of two steps.

1. Predicting an adaptive factor (explained below).
2. Scheduling tasks.

In order to minimize the execution time, we have to carry out two tasks. First, the performance of workers should be predicted effectively. The proposed method performs this task by using the Grid computation model described in Sec. 4.1 and applying the Mixed Tendency-based strategy (Sec. 4.2). Second, scheduling the workload (Sec. 4.3) is carried out by using the UMR algorithm [8] after integrating it with our CPU prediction mechanism.

4.1 Grid computation model

Most static scheduling algorithms [1, 2, 8, 10] assume that execution time is well-known based on the assumption that workers have fixed, predefined CPU speeds. On a nondedicated, dynamic platform such as Grid, these assumptions are not realistic. Thus in this paper we present a new model of executing local and Grid tasks at a given, non-dedicated worker.

During the execution of a Grid task on a certain worker, some local tasks may arrive causing to interrupt the execution of the lower priority Grid tasks. The arrival of the local tasks of worker$_i$ is assumed to follow a Poisson distribution with arrival rate λ_i, their execution process follows an exponential distribution with service rate μ_i and the local task process in the worker is an M/M/1 queueing system [4] ($i = 1, 2, ..., N$).

The execution time Tcomp$_{j,i}$ of chunk$_{j,i}$ on the worker$_i$ can be expressed as:

$$Tcomp_{j,i} = X_1 + Y_1 + X_2 + Y_2 + ... + X_{NL} + Y_{NL} \quad (9)$$

where:

- NL: the number of local tasks which arrive during the execution of chunk$_{j,i}$.
- Y_k: execution time of the local task k ($k = 1, 2, ..., NL$), these are independent identical distribution random variables.
- X_k: execution time of k^{th} section of $chunk_{j,i}$ ($k = 1, 2, ..., NL$). We have:

$$X_1 + X_2 + ... + X_{NL} = \frac{chunk_{j,i}}{S_i} \quad (10)$$

From the M/M/1 queueing theory [4] we have:

$$E\left(NL\right) = \frac{\lambda_i chunk_{j,i}}{S_i} \qquad E\left(Y_k\right) = \frac{1}{\mu_i - \lambda_i} \quad (11)$$

Because of NL and Y$_k$ are independent random variables ($k = 1, 2, ..., NL$) we derive

$$E\left(Tcomp_{j,i}\right) = E\left(Tcomp_{j,i}|NL\right) = \sum_{k=1}^{NL} X_k + \\ + \sum_{k=1}^{NL} E(Y_k) = \frac{chunk_{j,i}}{S_i} + E(NL) \times E(Y_k) = \\ = \frac{chunk_{j,i}}{S_i(1-\rho_i)} \quad (where\ \rho_i = \lambda_i/\mu_i) \quad (12)$$

λ_i, μ_i, ρ_i are representative on the long run but cannot be used to estimate the imminent execution time that will take place a given worker. Therefore, we introduce the $adaptive factor$ δ_i, which represents the performance of worker$_i$ and it is initialized by 1 (i.e., full availability of computational capacity) in the first round. Now the expected value of the execution time of chunk$_{j,i}$ is

$$\frac{chunk_{j,i} \times \delta_i}{S_i(1-\rho_i)} \quad (13)$$

The actual power of workers delivered to Grid varies over time, therefore we have to predict the adaptive factor δ_i as the below section.

4.2 Predicting the adaptive factor δ

In this section we consider worker$_i$ only, thus we will delete the character i in the notations, for example we write δ instead of δ_i. We periodically measure δ and obtain the original preceding value time series $C = c_1, c_2, ..., c_n$. Data point c_i is value of δ at time point i.
M: aggregation degree, calculated as
M = *execution time of a round* $\times$ *frequency of original time series*
$\Delta = \delta_1, \delta_2, ..., \delta_k (k = \lceil n/M \rceil)$: the interval CPU load time series, calculated as

$$\delta_i = \frac{\sum_{j=1}^{M} c_{n-(k-i+1)M+j}}{M} \qquad (i = 1, 2, ..., k) \tag{14}$$

Each value δ_i is the average value of adaptive factor over a round. After collecting the original time series C and creating interval time series Δ, we apply the Mixed Tendency-based strategy [6, 7] to estimate the value in the next round δ_{k+1}.

4.2.1 Prediction strategy

Algorithm 4.1: MIXEDTENDENCY-BASED()

```
procedure INCREMENTVALUEADAPTATION()
  Mean = $\left(\sum_{i=1}^{n} \delta_i\right)/n$;
  RealIncValue = $\delta_T - \delta_{T-1}$ ;
  NormalInc = IncrementValue + (RealIncValue-
             - IncrementValue) $\times$ AdaptDegree;
  if ($\delta_T$ < Mean)
    then IncrementValue = NormalInc;
    else
      PastGreater = (number of data points
                      greater than $\delta_T$) / n;
      TurningPointInc = IncrementValue $\times$
                      $\times$ PastGreater ;
      IncrementValue = Min(NormalInc,
                      TurningPointInc);

main
  if ($\delta_{T-1} < \delta_T$)              //Tendency is increase
    then
      INCREMENTVALUEADAPTATION()
      $P_{T+1} = \delta_T$ + IncrementValue;
  else if ($\delta_{T-1} > \delta_T$)         //Tendency is decrease
    then
      DECREMENTFACTORADAPTATION()
      $P_{T+1} = \delta_T \times$ DecrementFactor;
```

Formally, Mixed Tendency-based prediction [6, 7] strategies can be expressed as above. The adaptation process in case of Increase and Decrease are similar. δ_T is the current value of adaptive factor, and P_{T+1} is the predicted value for δ_{T+1}. AdaptDegree is optional parameter that expresses the adaptation degree of the variation, its value can ranger from 0 to 1. Now we predict that the average speed ES$_i$ of the worker$_i$ on the next round is

$$ES_i = \frac{S_i \times (1 - \rho_i)}{\delta_i} \tag{15}$$

where δ_i is predicted as explained above. Henceforth, we will use ES$_i$ to denote the speed of of worker$_i$.

4.3 Scheduling tasks

4.3.1 Induction on chunk sizes

We rewrite here the deductions and constraints of [8, 10]. While worker N process chunk$_j$, master send (N-1) chunks to (N-1) remaining workers. To maximize bandwidth utilization, the master must finish sending the last chunk$_{j+1,N}$ of round$_j$ to the last worker N before the worker N finish processing chunk$_{j,N}$, so we have

$$round_j = \theta^j \times (round_0 - \eta) \tag{16}$$

where

$$\theta = \left(\sum_{i=1}^{N} \frac{ES_i}{B_i}\right)^{-1} \tag{17}$$

$$\eta = \left(\sum_{i=1}^{N} \frac{ES_i}{B_i} - 1\right)^{-1} \times$$

$$\times \left(\sum_{i=1}^{N}(ES_i \times cLat_i) - \sum_{i=1}^{N} ES_i \times \sum_{i=1}^{N}\left(\frac{\beta_i}{B_i} + nLat_i\right)\right) \tag{18}$$

4.3.2 Constrained minimization problem

Our objective is to minimize the execution time of total workload W_{total}

$$Ex(M, round_0) = \\ = \sum_{j=0}^{M-1} const_j + \tfrac{1}{2}\sum_{i=1}^{N}\left(\tfrac{chunk_{0,i}}{B_i} + nLat_i\right) \tag{19}$$

$$G(M, round_0) = M \times \eta + \frac{round_0 - \eta}{1 - \theta} \times (1 - \theta^M) - W_{total} \tag{20}$$

where M and $round_0$ are unknowns.

$$round_0 = \frac{1-\theta}{1-\theta^M}(W_{total} - M \times \eta) + \eta \tag{21}$$

where M is solution of the following equation

$$\frac{(M \times \eta - W_{total}) \times \theta^M ln\theta \sum_{i=1}^{N} \frac{\alpha_i}{B_i}}{(1-\theta^M)} + \eta \sum_{i=1}^{N} \frac{\alpha_i}{B_i} - \\ -2\frac{1-\theta^M}{1-\theta} \times \frac{\sum_{i=1}^{N}(ES_i \times cLat_i)}{\sum_{i=1}^{N} ES_i} = 0 \quad (22)$$

After obtain the value of round$_0$ from (21), we can use (16) to compute round$_i$. Subsequently, (5) can be used to obtain chunk$_{j,i}$ $\forall i, j$

4.4 Overview of the proposed algorithm

Algorithm 4.2: PROPOSEDALGORITHM()

```
Collect the value of $\{B_i,\ S_i, \lambda_i, \mu_i, \rho_i\}$
Use equation (15) to derive $\{ES_i\}$ $(i = 1, 2, ..., N)$
Compute $M$, $round_0$, $\{chunk_{0,i}\}$ $(i = 1, 2, ..., N)$
$W_{remains} = W_{total} - round_0$;
Deliver $\{chunk_{0,i}\}$ to $\{worker_i\}$  $(i = 1, 2, ..., N)$
repeat
  // Processing on round$_j$
  Collect items of the series C of last round
  Use Tendency-based Predictor to obtain $\{ \delta_i \}$
                                        $(i = 1, 2, ..., N)$
  Use equation (15), (16) to derive $round_j$ and
                                $\{ES_i\}$ $(i = 1, 2, ..., N)$
  if $(round_j > W_{remains})$
    then $round_j = W_{remains}$;
  $W_{remains} = W_{remains} - round_j$;
  Deliver $\{chunk_{j,i}\}$ to $\{worker_i\}$ $(i = 1, 2, ..., N)$
until $W_{remains} = 0$;
```

5 Conclusion

In this paper we presented a dynamic scheduling method that is based on the UMR algorithm and the M/M/1 model. We discussed a task execution model that describes the processing of local and Grid tasks each individual machine. Then we used this model to predict the performance of these worker machines. Based on the estimated performance of each worker, we decide on how to distribute workload chunks. The prediction of workers' performance takes place the beginning of each round based on the historical values observed in the previous rounds.

In the future, we consider three extensions of the current work. First, we would like to remove the constraint that the master can not send data to many workers at the same time because current platforms, such as WAN, support this capability. Second, we will factor in the time needed to ship the results back to the master. Finally, we have noticed that the majority of present algorithms assume that the execution time is is proportional to the size of the data, therefore the relation between computation time and transfer time is linear (see equations (1,2) in Section 3). We believe that the real relation between them is more complex and it largely depends on the characteristics of the data that need processing.

References

[1] O. Beaumont, A. Legrand, and Y. Robert. Scheduling divisible workloads on heterogeneous platforms. *Parallel Computing*, 29(9), September 2003.

[2] V. Bharadwaj, D.Ghose, V.Mani, and T. G. Robertazzi. *Scheduling Divisible Loads in Parallel and Distributed Systems.* IEEE Computer Society Press, 1996.

[3] J. Blazewicz, M. Drozdowski, and M. Markiewicz. Divisible task scheduling-concept and verification. *Parallel Computing*, 25(7):87–98, January 1999.

[4] A. Papoulis and S. U. Pillai. *Probbility, Random Variables, and Stochastic Processes.* McGraw-Hill, 2002.

[5] R. Wolski. Dynamically forecasting network performance using the network weather service. *Journal of Cluster Computing*, 1998.

[6] L. Yang, J. Schopf, and I. Foster. Conservative scheduling: Using predicted variance to improve scheduling decision in dynamic environments. *Super-Computing 2003, Phoenix, Arizona USA*, November 2003.

[7] L. Yang, J. Schopf, and I. Foster. Homeostatic and tendency-based cpu load predictions. *International Parallel and Distributed Processing Symposium (IPDPS'03) Nice, France*, April 2003.

[8] Y. Yang and H. Casanova. Multi-round algorithm for scheduling divisible workloads application: Analysis and experimental evaluation. *Technical Report CS2002-0721, Dept. of Computer Science and Engineering, University of California*, 2002.

[9] Y. Yang and H. Casanova. Rumr: Robust scheduling for divisible workloads. *12th IEEE International Symposium on High Performance Distributed Computing (HPDC'03) Seattle, Washington, USA*, 2003.

[10] Y. Yang and H. Casanova. Umr: A multi-round algorithm for scheduling divisible workloads. *Proceeding of the International Parallel and Distributed Processing Symposium (IPDPS'03), Nice, France*, April 2003.

[11] Y. Zhang, Y. Inoguchi, and H. Shen. A dynamic task scheduling algorithm for grid computing system. *Second International Symposium on Parallel and Distributed Processing and Applications (ISPA'2004)*, December 2004.