Enhanced Adaptive Scheduling for the Grid Harvest Service

Wushou Sliamu and Yong Hou
College of Information Science and Engineering
Xinjiang University, China, 830046
{wushour, houyong}@xju.edu.cn

Junwei Cao
Research Institute of Information Technology
Tsinghua National Laboratory for Information Science and Technology
Tsinghua University, Beijing 100084, China
jcao@tsinghua.edu.cn

Abstract

Grid technology and applications become mainstream distributed computing research in recent years. The Grid Harvest Service, GHS, is one of non-dedicated computing grid scheduling systems; it has been widely used in Grid research field. The contribution of GHS is providing appropriate prediction for long-term applications different from AppLes which is designed for short-term predictions. GHS maps metatasks using the Min-Min algorithm in a uniform log-term application. Min-Min is a highly efficient algorithm in an uniform workload environment that leads to load unbalance and low performance when the workload is not uniform. In order to solve the problem, a novel task scheduling algorithm is proposed in this work, which is an adaptive task scheduling algorithm based on Min-Min and Max-Min (A-MM). A-MM merges the high efficiency of traditional Min-Min scheduling with load balance of traditional Max-Min scheduling. The simulation and experimental result show that A-MM has better performance and scalability than the Min-Min of GHS.

1. Introduction

Large scale of computation and high costs of supercomputers make it desirable to utilize computational resources distributed on networks to solve problems in science, engineering and commerce. Lots of efforts have been done and many projects have been defined, such as Condor [1], Globus [2] and Nimrod [3].

Currently, there are many distributed and grid systems to be researched and used, which can provide different kind services to users. One kind of Grid systems is Non-dedicated grid, which assumes that guarantees cannot be obtained from the system. A Grid task is allowed to execute when there is no local jobs running. If any local jobs or other jobs arrive, the Grid task is either suspended or terminated. For example, in SETI@home, Entropia, and Condor, a Grid task is allowed to run only when no keyboard or mouse activities are detected on local resources.

Performance-prediction based task scheduling system has been researched in non-dedicated grid world [4], and there are many grid system based on non-dedicated grid, such as AppLes[7] and RPS[8]. In these systems, the CPU availability is estimated by Network Weather Service (NWS)[10]. However, NWS is designed for short-term predictions (up to 5 minutes) and cannot provide appropriate prediction for long-term applications. GHS is a performance evaluation and task scheduling system for solving large-scale applications in a shared environment.

GHS schedule long-term applications by Min-Min algorithm. Min-Min is a simple and highly efficient algorithm. However, in real Grid society, as we know, scheduling system always dispose tasks including long-term and short-term applications, whose workloads are not uniform. Through a plenty of experiments [5], we can conclude that the performance of Min-Min is bad in the condition where the workload of applications are not uniform. So how to solve the problem of Min-Min bad performance in the environment is a serious issue.

The experimental results in [6] show that in the condition when there are many uneven applications waiting to be scheduled, the Min-Min algorithm is unable to balance the load well since it usually schedules small tasks first, on the contrary, the Max-Min algorithm may result in a better balance for a

978-0-7695-3570-8/09 $25.00 © 2009 IEEE
DOI 10.1109/WCSE.2009.308

Authorized licensed use limited to: MIT Libraries. Downloaded on December 6, 2009 at 06:41 from IEEE Xplore. Restrictions apply.

given mapping. Thus, in this paper, we try to absorb the merit of both Min-Min and Max-Min and propose a new adaptive task scheduling algorithm, namely the Adaptive Min-Min and Max-Min (A-MM) algorithm. The new algorithm not only retains the advantage of the Min-Min algorithm but also achieves the good load balance.

This paper is organized as follows. In section 2, the GHS system is analyzed. In section 3, the new scheduling algorithm is introduced. In section 4, the experimental results are presented and discussed. We conclude this study in section 5.

2. The Grid Harvest Service

2.1 The architecture of GHS

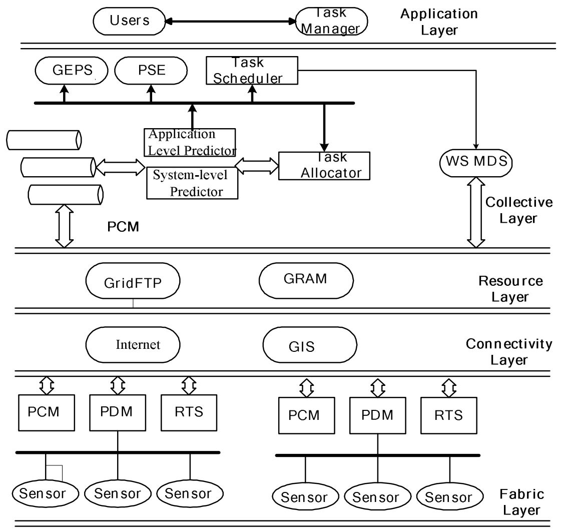

Figure 1. Architecture of GHS performance evaluation and task scheduling system.

Figure 1 shows the relationship among the major components of the GHS system. Referring the OGSA criterion, GHS is divided into four layers: Fabric Layer, Collectivity Layer, Resource Layer, and Application Layer. Many components in each Layer cooperate together to finish Grid tasks. Major components are as follows:

Application Level Predictor, which estimates the application performance based on the map information provided by the Task Allocator component and the system information provided by the System-level Predictor component.

System Level Predictor analyzes resource usage patterns to provide an estimation of system performance in a future period. The inputs of this component are the preprocessed system parameters stored in a performance database.

Task Allocator decides how to partition a parallel program or how to group subtasks of a meta-task. The inputs of this component are the application characteristics (workload, application type, dependency of subtasks of an application), and the output is a mapping of subtasks on a given set of computing resources.

Task Scheduler determines a scheduling plan for a large-scale application to provide an optimal or near-optimal solution for its running in a shared environment.

Performance Data Management (PDM) implements data filtering and reduction techniques for extrapolating system and application performance information on each resource.

Reschedule Trigger System (RTS) analyzes the collected application and the performance data of resource to detect whether resources present abnormal performance from their historical records. If an abnormal resource behavior is identified, application will be rescheduled to prevent potential performance loss.

Grid-enabled Programming System (GEPS) and *Problem Solving Environment (PSE)* are served by the Prediction component.

Performance Communication Manager (PCM) component is used to collect performance data, which could be exchanged through the GridFTP services protocol based on the communication infrastructure provided by the GSI service in the Connectivity layer. The GridFTP service protocol can be used to handle the transfer of applications and their data files and the GRAM can be used to dispatch subtasks of applications on resources.

2.2 The scheduling of GHS

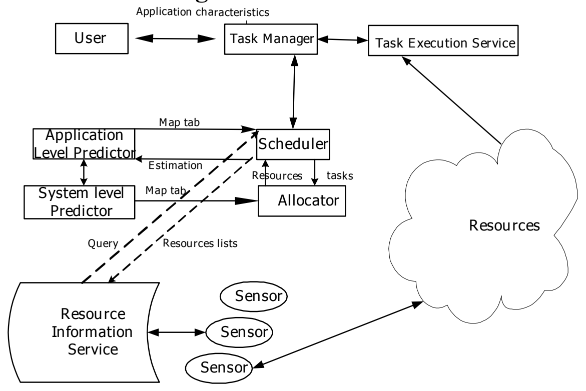

Figure 2. A framework of GHS scheduling system.

Figure 2 shows the process of GHS scheduling system. GHS has five orders to finish a scheduling of a task: the resource measurement order (smeas), system-

Authorized licensed use limited to: MIT Libraries. Downloaded on December 6, 2009 at 06:41 from IEEE Xplore. Restrictions apply.

level predictor order (syspred), application-level predictor order (apppred), meta-task scheduler order (mtsc) and parallel program scheduler order (pgsc). It works as following: firstly, collecting resource utilization, job arrival rate, job service time, job service time standard deviation of resources during 1000 hours, and sending them into dynamic array of ghsmeasrue.log, the smeas order calculates the average of data stored in ghsmeasrue.log during each hour and send them into the parameter of date, meanUtil, meanLambda, meanServ and meanStd of ghsmeasrue.log. The syspred order will collects all of the parameter from ghsmeasure.log, and gets the even value of job arrival rate in each hour, named valueHour, from predictable in a given period, and get the even value of job arrival rate in every day, named valueDay, in a given period.

The result of prediction in a given period is Alpha * valueHour +Beta * valueDay. Alpha and Beta could be obtained from experience. apppred order estimates the job service time and job service time standard deviation. the cumulative distribution function of the application completion time on a machine is

$$P_r\{T \le t\} = \begin{cases} 0 & \textit{otherwise}, \\ \prod_{k=1}^{q} e^{-\lambda_k w_k/\tau_k} + (1-e^{-\lambda_k wk/\tau_k}) P_r\{U(S_k) \le t - W_k / \tau \mid S_k > 0\}, & t \ge W_k, \end{cases}$$

where $P_r\{T \le t\}$ calculates the cumulative distribution function of the application completion time (T) and T_k is the subtask completion time on machine k, name M_k. The arrival of local jobs in machine k follows a Poisson distribution with λ_k. w_k is the workload of the remote task on M_k. S_k is the number of interruptions encountered on M_k. t_k is task completion time on M_k and τ_k is the capability of computing on M_k. $U(S_k)$ is the completime of the grid task on the condition of S_k. The mtsc order is a program which generates a scheduling plan for a given resource set and a given meta-task. We will discuss the meta-task scheduling in this paper. The pgsc order is a program which generates a parallel scheduling plan.

3. Enhanced adaptive scheduling

In this section, we will introduce A-MM algorithm and its implementation into GHS in details.

As mentioned in [5], the grid resources set is $R=\{r_1,r_2,...,r_n\}$, where there are n resources, $r_1,r_2,...,r_n$, which could be used by the scheduler. The grid tasks set is $J=\{j_1,j_2,...,j_m\}$, and there are m tasks, $j_1,j_2,...,j_m$, which demand the resource. Expected time to compute (*ETC*) matrix is a model of the given service grid system, where the ETC_{ij} is the expected execution time of task i on machine j without any load.

Definition 1. The judgment array of A-MM, *Min_Time*. We can find the minimum ETC_{ij} from each task *ETC* of task *i*, named $Min\_Time(i)= ETC_{ij}$, on every resources, and all of the ETC_{ij} compose a set of number, named *Min_Time*.

Definition 2. Relative Standard Deviation, *RSD*.

$$RSD = \frac{s}{\overline{x}},$$

where

$$s = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \overline{x})^2}{n-1}}.$$

s is the standard deviation of the sample, which represents the degree of the dispersion of it. $\bar{x}$ is the mean value of the sample. *RSD* could well indicate the degree of dispersion of a set of values.

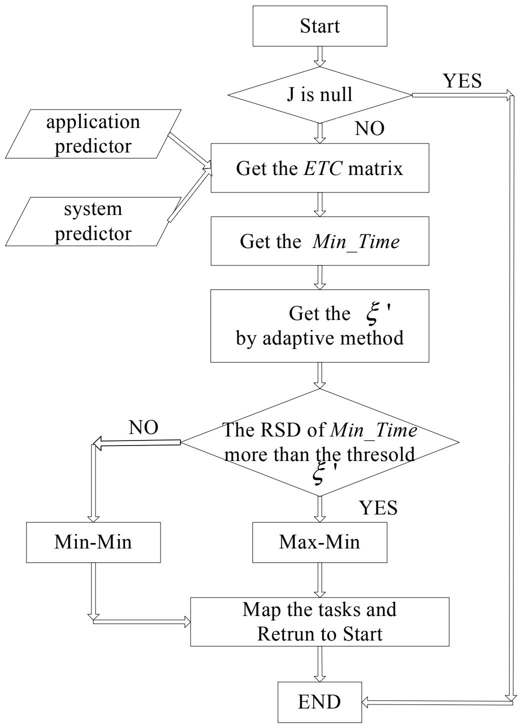

Figure 3. The flowchart of A-MM scheduling

Figure 3 gives the description of A-MM algorithm: Firstly we obtain the *ETC* matrix through the application predictor and the system predictor. Secondly, we get the value of *ETC* and compose the *Min_Time*. Thirdly, we get the value of RSD of the *Min_Time*, and denoted as ξ. A-MM algorithm will use the property of the RSD, which could well indicate the degree of dispersion of a set of value, to judge the degree of the dispersion of the *Min_Time*. If ξ is

Authorized licensed use limited to: MIT Libraries. Downloaded on December 6, 2009 at 06:41 from IEEE Xplore. Restrictions apply.

smaller than the critical value of *RSD* of *Min_Time*, which is named ξ' and could be obtained by means of experiences, we regard the degree of dispersion of *Min_Time* is lower, and it is believed the *ETC* values of tasks are uniform, so the scheduler will select the Min-Min algorithm that has better performance in the uniform environment. Otherwise, the dispersion degree of *Min_Time* is higher, for the *ETC* values of tasks are not uniform, consequently, and the scheduler will select the Max-Min algorithm to assign tasks. As a result, A-MM algorithm can obtain much better performance than that of either Min-Min or Max-Min algorithm through the selecting mechanism described above.

The description in detail of A-MM algorithm in GHS is proposed in Figure 4.

```
Assumption: a meta-task composed of a number of independent subtasks, $S_r=\{t_1,t_2,\ldots,t_p\}$ and a list of machines $\{m_1,m_2,\ldots,m_q\}$. Each subtask is indivisible.
Objective: find a task group $m_k(1\le k\le q)$ for machine $G_k=\{t_{k1},t_{k2},\ldots,t_{kn}\}$
--------------------------------------------------------------
Begin
/*$C_k$ denotes the current estimated completion time of the assigned subtasks on machine $m_k$ */
$C_k=0, G_k=\phi, (1\le k\le q); i=1;$
While $S_r \ne \phi$
  For $t_i \in S_r$ Do
    $j=1;$
    While $j<q$
$E(T_{i,j})=w_i/[\tau_j*(1-\rho_j)];$
/*$E(T_{i,j})$is the execution time of $t_i$ on machine $m_j$*/
$j=j+1;$
End While
Find machine $k$ where $C_k+E(T_{i,j})$ is minimal;
End For
Compose all of $C_k+E(T_{i,k})$ into a one metric $[C_k+E(T_{i,k})]$ , named Min_Time;
Canculate the RSD value of $[C_k+E(T_{i,k})]$, $\xi$;
While, get $\xi'$, the critical value of RSD of Min_Time, by means of daptive method:
$\xi'$ take the value between 0.1 and 1 each 0.05 in each process of choosing, select the optimal value of $\xi'$ from all of them as the judgment key of A-MM;
If($\xi>\xi'$)
Find a map of $(t_u, m_v)$ where $C_v+E(T_{u,v})$ is maximal;
Else Find a map of $(t_u, m_v)$ where $C_v+E(T_{u,v})$ is minimal;
$G_v=G_u\cup\{T_u\}; C_k=C_k+E(T_{u,v});$
$S_r=S_r-\{T_u\}; i=i+1;$
End While
Return $G_k(1\le k\le q);$
End
```

Figure 4. A-MM scheduling algorithm in GHS

w_k is the workload of the remote task on M_k, and it is a static value. τ_j, ρ_j will be obtained from performance prediction component, T_u , m_v and G_v stand for task *u*, machine *v* and task group on machine *v* respectively.

4. Simulation results and performance evaluation

In this section, we evaluate the performance of A-MM through implementing it into GHS and compare the performance of A-MM with Min-Min in different scenarios. All the experiments run in Solaris. In this paper, we will set a scenario with 5 resources and 4 groups tasks, including 5, 15, 30, 45 tasks respectively. The result can be seen in Figure5.

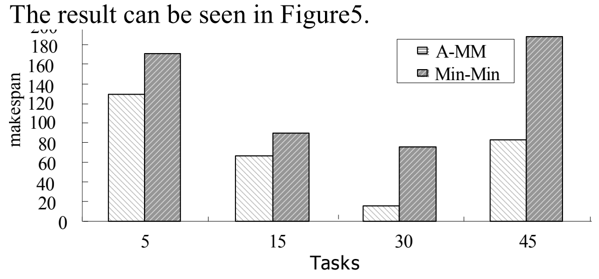

Figure 5. Makespan of A-MM and algorithm in GHS

As show in Figure 5, for all the four scenarios, the A-MM outperforms the Min-Min. Through exploiting the merit of the Min-Min and Max-Min, A-MM could exert the potential of the efficiency of Min-Min and the load balance of Max-Min, obtaining the better performance than others by choosing the optimal value of ξ' , A-MM algorithm could get much better performance in most situations than Min-Min.

Figures 6 and 7 show the comparison of the scalability of tasks between A-MM and Min-Min of GHS, which include 5 resources and 5～50 tasks in Figure 6 and 10～50 service resources and 20 tasks in Figure 7.

Authorized licensed use limited to: MIT Libraries. Downloaded on December 6, 2009 at 06:41 from IEEE Xplore. Restrictions apply.

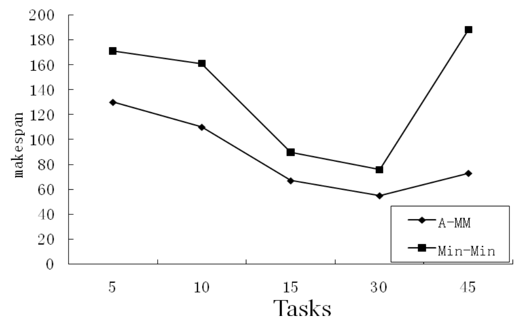

Figure 6 . The scalability of resources

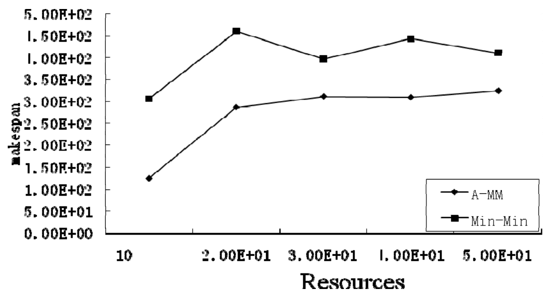

Figure 7. The scalability of tasks

As can be seen, the line of A-MM is more close to linearity than Min-Min algorithm, and therefore, A-MM has much better scalability of tasks and resources than that of Min-Min and Max-Min [9].

5. Conclusions

In this paper, we thoroughly analyze GHS and propose a new algorithm, A-MM, to solve the problem of unbalanced load scheduling algorithm of GHS. A-MM merges the efficiency of Min-Min and the load balance of Max-Min and show the better performance than original Min-Min. Meanwhile, A-MM is robust in the scalabilities of the tasks and the resources.

As for future work, we plan to expand the scale of simulations, and multi-dimensional QoS should be embedded into the task scheduling.

Acknowledgment

This research is supported by the priming scientific research foundation for the junior teachers in XinJiang University (grants No.070309), sponsored by Ministry of Science and Technology of China under the national 863 high-tech R&D program (grants No. 2006AA10Z237, No. 2007AA01Z179 and No. 2008AA01Z118), and National Science Foundation of China (grant No. 60803017).

References

[1] R. Raman, M. Livny, and M. Solomon, “Matchmaking: Distributed Resource Management for High Throughput Computing”, *Proceedings of the 7th IEEE International Symposium on High Performance Distributed Computing*, Chicago, IL, July 1998, pp.140-146.

[2] I. Foster, “Globus Toolkit Version 4: Software for Service-Oriented Systems,” *IFIP International Conference on Network and Parallel Computing*, Springer-Verlag LNCS 3779, 2005, pp. 2-13.

[3] D. Abramson, R. Sosic, J. Giddy, B. Hall, Nimrod: a tool for performing parametised simulations using distributed workstations, in: *Proceedings of the 4th IEEE Symposium on High Performance Distributed Computing*, Virginia, August 1995, pp.112-121.

[4] X.-H. Sun, M. Wu, “Grid Harvest Service: A system for long-term, application-level task scheduling”, in: *Proceedings of 2003 IEEE International Parallel and Distributed Processing Symposium*, Nice, France, April 2003.

[5] Yong Hou, Jiong Yu and Turgun, NDA-MM: A New Adaptive Task Scheduling Algorithm Based on the Non-dedicated Constraint Grid, *Sixth International Conference on Grid and Cooperative Computing*, 2007.8, pp.275-281.

[6] T. D. Braun, H. J. Siegel, N. Beck, L. L.Bölöni, M. Maheswaran, A. I.Reuther, J. P.Roberstson, M. D.Theys, Y. Bin, D. Hensgen, R. F.Freund, “A Comparison of Eleven Static Heuristics for Mapping a Class of Independent Tasks onto Heterogeneous Distributed Computing Systems”, *Parallel and Distributed Computing*, 2001.61(6), pp.810-837.

[7] H. Casanova, G. Obertelli, F. Berman, Rich Wolski, “The AppLeS parameter sweep template: user-level middleware for the Grid”, in: *Proceedings of Super Computer 2000*, Dallas, TX, November 2000, pp.60-79.

[8] P. Dinda, D. O’Hallaron, Host load prediction using linear models, *Cluster Computing*, 3 (2000) 265–280.

[9] Sun Xian He, DT Rover. NLR Center, VA Hampton., “Scalability of parallel algorithm-machine combinations”, *Parallel and Distributed Systems*, 1994.5(6), pp.599-613.

[10] Richard Wolski, Neil T.Spring,and Jim Hayes, “The network weather service: a distributed resource performance forecasting service for metacomputing”, *Future Generation Computing Systems*, Oct.1999, pp.757-768.

Authorized licensed use limited to: MIT Libraries. Downloaded on December 6, 2009 at 06:41 from IEEE Xplore. Restrictions apply.