PROCEEDINGS OF SCIENCE

An Improved Job Scheduling Algorithm Based on the First Distribution and Redistribution Strategy for Grid Computing

Tianwei Ni[1] [2]
Department of Electronics and Information Engineering
Hohai University Wentian College, Maanshan 243031 Anhui China
E-mail: tianwei5689@126.com

Baolin Zheng
Department of Information Engineering
Henan vocational and technical college, 450046 Zhengzhou China
E-mail: zhengbaolin_2001@163.com

Jinzhu Lin
Department of Electronics and Information Engineering
Hohai University Wentian College, Maanshan 243031 Anhui China
E-mail: jinzhu5689@126.com

As the traditional job scheduling algorithms ignore the impact of resource fragmentation, there are some insufficiencies in terms of the job execution time for grid computing. Since deficiency occurs to these traditional job scheduling algorithms , why not reduce the resource fragmentation probability by the way of combining backfilling with the priority scheduling strategy? If it should be, how to be regulated? This paper argues that the treatment of resource recovery should be the solution to a considerable resource fragmentation problem. We present an improved job scheduling algorithm which optimizes the traditional grid scheduling algorithms based on the first distribution and the redistribution strategy of jobs. The algorithm considers the resource recycling so specifically that it can distribute the job efficiently. Simulation experiment results demonstrate that the improved job scheduling algorithm can greatly improve the resource utilization rate; at meanwhile, it can also track the system throughput rate with excellent performance.

CENet2015
12-13 September 2015
Shanghai, China

[1]Speaker
[2]Corresponding Author

PoS(CENet2015)060

1. Introduction

Job scheduling is not only an essential function to the grid computing, but also a key factor of improving the system performance and system efficiency in the grid computing. It is necessary to achieve an effective job scheduling algorithm in order to improve the resource utilization. The traditional grid job scheduling algorithms such as Suffrage-C [1], Min-min [2], Max-min [3] and WQ [4] etc. can set the job priority automatically. Once an idle resource is found, the high priority assignment will be submitted automatically and running. If we ensure normal operation of these algorithms, we have to provide detailed information in respect of the job and the processor. In fact, it is unpossible to get the detailed information; therefore, it reduces the system response time, etc. In order to better match suitable resources for job, this paper presents an improved job scheduling algorithm, which uses the first distribution and redistribution strategy in the grid environment while not requiring detailed processor information and job information.

The paper is organized as follows: firstly, the introduction is made to the design of the system scheduling model in Section 2; secondly, Section 3 presents an improved algorithm based on the first distribution and redistribution strategy concretely; thirdly, in Section4, our experiments are discussed. In the end, Section 5 concludes the paper and gives a look to the future trends.

2. System Scheduling Model

In the grid computing, one resource could meet the needs of multiple jobs and one job can be matched by multiple resources. As shown in Fig. 1, the size of a job refers to the number of instructions contained within this job; and the processing capacity of a processor refers to the processor capable of handling the number of instructions in an unit time. The grid scheduler shall arrange these jobs as assigned to the distributed by the processors under certain criteria [5]. The grid scheduler is to choose a job waiting to be assigned to a processor according to the job size. Once the resource is adequate to meet the needs of the job, the job will run promptly.

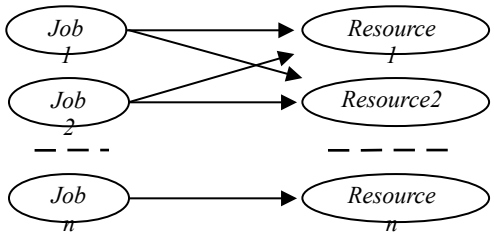

Figure 1: Surjection from Job Collection to Resource Collection

The job scheduler assigns these tasks to the heterogeneous and distributed processors under certain criteria. The key to the job scheduling is the determination of such criteria in respect of the dispatch objective function[5]. The performance of the processor is affected by the local job and varies through time in the grid computing. In dealing with such situation, This paper uses a new standard, the TPCC (total processor cycle consumption)[6]. This criterion has taken into account the variation of the processing capability of the grid through time and proved that it is equivalent to the makespan. This paper measured the performance of schedulers in the light of TPCC. The number of instructions processed by the processor P in $(t,t+1)$ is represented by $n_{p,t}$. T represents a collection of n independently working sets. Each task can be replaced by a three tuple $< v, p, t >$, among which, $v \in T$, $1 \le p \le m$, where, m is the number of processor and t is the starting time for each task. The TPCC formula is shown as follows:

$$\text{TPCC}= \sum_{t=0}^{makespan} \sum_{p=1}^{m} n_{p,t} \qquad (2.1)$$

It can be seen from Formula (2.1) that a scheduler's TPCC is in proportion to its makespan. This paper discusses an improved job scheduling algorithm for the grid computing based on TPCC. The time of matching resources with tasks will be added to the makespan, as shown in Formula (2.2):

$$\text{makespan} = \max_{p=1}^{m}(S_p + \theta_p) \tag{2.2}$$

Where S_p represents the time for the calculation of processor P; θ_p represents all the tasks in respect of the dispatch time span before Processor P calculates the tasks, which includes the time used for the task creation and the acquisition of resources, etc..

3. Algorithm Flow Description

An improved job scheduling algorithm is proposed in this paper. The scheduling algorithm considers TPCC as the criterion and achieves the minimum of TPCC by the scheduling strategy. The required data structure should be defined before the algorithm is proposed, as shown in Fig. 2.

1) $Wait\_queue$: a FIFO queue.

2) Run_queue: the queue of running job.

3) P_i: represents the job priority. The job scheduler decides the operation procedure to accept parameters of the system resources at the priority level in case of dealing with multiple working procedures, as shown in Formula (3.1):

$$P_i = p[i] + l * St \tag{3.1}$$

where $p[i]$ is the initial value of priority, St represents the job status and l represents constant factor.

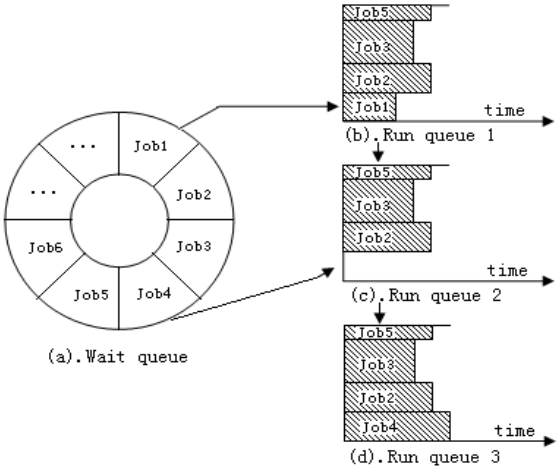

Figure 2 : Data Structure of System Scheduler

The data structure is divided into a wait queue to submit job and a global run queue as shown in Fig. 2. $Wait\_queue$ represents a wait queue to submit job and the wait queue is "first in and first out" (FIFO) queue. There is a waiting for submission to the local scheduler in this queue. Run_queue represents a global run queue which is the running job. Abscissa refers to the run length and the ordinate refers to the number of resources Used to run job. The job is selected to run by the scheduler in turn in this $Wait\_queue$; but if some resources can't satisfy the operation needs, the job will continue to wait while increasing the priority of the job as well. When the system released some resources, the resources will be allocated for the job which is waited.

The Pseudocode of job scheduling algorithm based on data structure of Fig. 2 is designed as follows:

Void Scheduler (set *T*, set *P*)

{**if**（job t_i accomplished）
for the job t_i in *Wait _ queue*
if (*Pi* != *null*)
if (*Pi* = =*Pmax*)
if (resources can satisfy the operation needs)
the scheduler assigns resources to the job on the basis of backfilling
else
P_i ++
next job is dispatched in *Wait _ queue*
end if
else if (P_i= =P_j)
if (resources can satisfy the operation needs)
the scheduler assigns the resource to the job on the basis of FCFS
else
P_i ++
select a job on the basis of backfilling
end if
else
select a job on the basis of priority strategy
end if
else
P_i ++
when the system released some resources, which will be allocated for the job which is waited on the basis of backfilling
end if
end if
}End

4. Simulation and Experiment

This paper uses GridSim to simulate the system's performance[7]. GridSim is a grid simulator tool based on discrete event simulation package for Java. It supports the heterogeneous style resources and the application of the modeling and simulation.

This simulation compares the performance of the improved algorithm with the EASY-backfillling algorithm as proposed in reference under different scheduling times. It is the algorithm that is a typical aggressive backfilling algorithm [8]. According to the simulation requirements, the algorithm sets two routers of network topology with one used to connect the scheduler and the other to connect all the computing resources; and the bandwidth between two routers is 10Mb/sec. The grid resource is set in the space-shared mode and the network transmission delay is 20 ms. The simulation results indicate the improved algorithm can greatly improve the resource utilization rate; at meanwhile, it can also track the system throughput rate with excellent performance, as shown in Fig. 3.

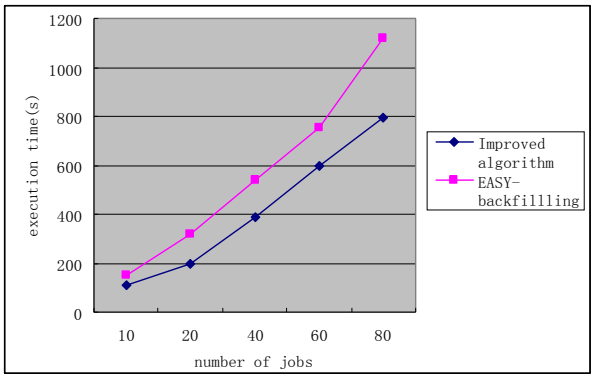

Figure 3 : Influence of Task Size on Execution Time

If the number of resources and the size of tasks remain the same, the number of the grid resource sets is 10 and the bandwidth of the link between each resource and router is 1Mb/sec with the task size between $1*10^9$ instructions/sec ~ $3*10^9$ instructions/sec. The simulation results indicate the performance of the improved algorithm is relatively advanced when compared with that of EASY-backfilling along with the increase in the number of jobs. This is because when the number of jobs increases, the job scheduling time saved by the use of the method is more obvious, thus saving TPCC tremendously, as shown in Fig. 4.

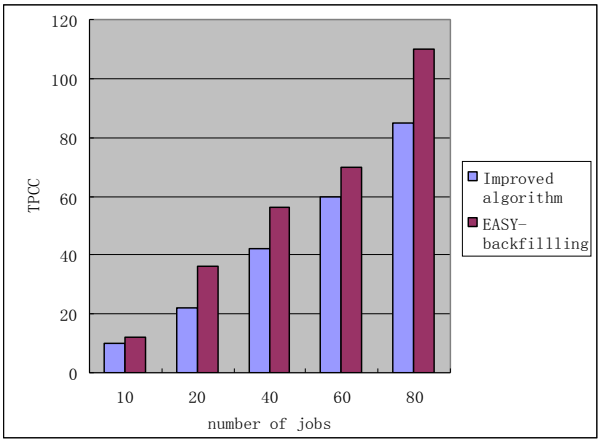

Figure 4 : Influence of the Task Size on TPCC

5. Conclusion

This paper mainly discusses an improved job scheduling algorithm based on the first distribution and redistribution strategy in the grid environment; besides, the design of the system scheduling model and the algorithm description are discussed. Upon the experiment verification, this algorithm can efficiently improve the system resource utilization rate and the system load balancing. Of course, there are some topics not covered herein, for example, whether the scheduler can make a decision on the starting time and ending time of each job is the basic standard that makes assessment of implementing resource reservation strategy. Hopefully, we are able of increasing the resource reservation algorithm to improve the system's throughput rate to certain extent.

References

[1] Casanova H, Legrand A, Zagorodnov D,Berman F. *Heuristics for scheduling arameter sweep applications in grid environments*[C]. Proceedings of the 9th heterogeneous computing workshop (HCW'2000). IEEE Computer Society Press.pp,349-363(2000)

[2] Braun TD, Siegel HJ, Beck N. *A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems*[J].Journal of 6 parallel and distributed computing,61(6), 810−837(2001)

[3] Maheswaran M., Ali S.,Siegel H.J.,Hensgen D.,Freund R.. *Dynamic matching and scheduling of a class of independent tasks onto heterogeneous computing systems*[C]. The 8th IEEE Heterogeneous Computing Workshop (HCW'1999). IEEE Computer Society Press.pp,30–44(1999)

[4] Graham R.L.. *Bounds for certain multiprocessing anomalies*[J].The Bell System Technical Journal. 45,1563–1581(1966)

[5] Lei Zhang，Yi Wang. *A Fault Tolerant Grid Scheduling Algorithm for Coarse-Grained Tasks in Parameter Sweep Applications*[J]. Journal of Hohai University(Natural Sciences),36(2),258-262(2008) " (In Chinese)

[6] Bucur I D, Epema H J. L*ocal versus global schedulers with processor co-allocation in multicluster systems*[J].Lecture Notes in Computer Science.2537,184-204(2002)

[7] Buyya R, Murshed M. *Gridsim: a toolkit for the modeling and simulation of distributed resource management and scheduling for grid computing*[J].The Journal of Concurrency and Computation:Practice and Experience.14(13-15),1175-1220(2002)

[8] Hong Jiang, Tianwei Ni. *PB-FCFS--A Task Scheduling Algorithm Based on FCFS and Backfilling Strategy for Grid Computing[*C]. The 2009 Joint Conference on Pervasive Computing(JCPC'2009) , IEEE Computer Society Press.pp,507-510(2009)