International Journal of Security and Its Applications
Vol.9, No.8 (2015), pp.427-438
http://dx.doi.org/10.14257/ijsia.2015.9.8.38

Heuristic Grid Resource Scheduling Algorithm based on Group of Task and Secondary Distribution

Zhongping Zhang, Yupeng Feng, Shan Zhang and Ying Sun

School of Information Science and Engineering, Yanshan University, Qinhuangdao, 066004, China;
zpzhang@ysu.edu.cn;250771441@qq.com

Abstract

Grid resource scheduling policies directly affect the performance of the grid, so the grid resource scheduling algorithm for grid research has become a hot spot. In this paper, based on grid heterogeneity to resource scheduling of load balancing, effective resource utilization and minimize task sets the span of time (Makespan) as the goal, propose a heuristic grid resource scheduling algorithm based on Group of Task and Secondary Distribution, the basic idea is dividing the set of tasks into two groups by using the standard deviation, the one is long tasks group, another is short tasks group .The longest task in long tasks group will be allocated to the worst performance to execute, the rest can be done in the same manner. Tasks in short tasks group will be executed by the Min-Min algorithm , and then through secondary dispatch ideas to make the grid system to quickly reach each resource load balancing , improving resource utilization efficiency and minimizing the task set time span. Finally, the simulation model benchmark Braun et al. demonstrates the effectiveness of the algorithm.

Keywords: *resource scheduling; standard deviation; secondary distribution; load balancing; makespan*

1. Introduction

Grid is a kind of infrastructure which appears in the end of the 20th century, it can provide independent, universal and cheap distributed computing [1].Grid is a kind of heterogeneous computing systems which can maximize resource utilization due to the heterogeneity of tasks and resources. The key to raise the utilization ratio of resources in the heterogeneous environment is to distributes tasks which is going to be performed to resources and establish a best map matching between set of tasks and resources. But it becomes complicated to schedule grid resource because the grid is a heterogeneous computing system. The key of the grid resources scheduling is how to choose the scheduling algorithm and a good scheduling algorithm can improve the overall performance of the grid system.

2. Related Work

Scheduling algorithm based on the load balancing has been studied deeply in a classic distributed system and in the dedicated resources, but the scheduling algorithm can't show the good performance because of the heterogeneity, scalability and autonomy of the environment of grid. It is because of these reasons that scheduling algorithm which is in the environment of grid has caused the industry experts and scholars to explore and research continuously.

(1) Min-Min algorithm

Min-Min algorithm is a important and heuristic scheduling algorithm of grid resources [2]. Min-Min algorithm improves the effective utilization of resources [3] because it is simple, rapid and effective.

Min-Min algorithm is a important and heuristic scheduling algorithm of grid resources [2]. Min-Min algorithm improves the effective utilization of resources [3] because it is simple, rapid and effective.

The realization of Min-Min algorithm it to preferentially scheduler the task which is accomplished in a shortest time and it can shorten the finish time of all the tasks. But the tasks which must be executed in a longer time must wait to the free time of resource to be executed. When the long task is less in the task set, it can cause the long task not to be dispatched in a long time. On the other hand, it may occur the situation that Most tasks are executed in rare resources and the rest resources are free, so it can cause the unbalanced load and low utilization rate of total resources. Scheduling strategy of the scheduling algorithm is to calculate the expected finish time of the task on every resources and then find the shortest finish time of tasks on every resources. Then we can select tasks which is finished in a shortest time and the corresponding resources among every task set which is finished in a shortest time and then allocate resources to the task and delete the task and update the available time of resources. Repeat the above operation for the rest of the task set. The classical Min-Min algorithm achieves good performance in the stage of the rise of grid technology. But with the constant and complex changes of grid environment, the performance of the Min-Min algorithm is getting worse. The basic reason is that it adopts the strategy that it always dispatchs short job to resources of strong ability of calculation. Causing the resources of strong ability of calculation overload, but the resources of week ability of calculation are relatively free. It leads to load imbalance of the overall resources and low rates of overall resource utilization.

(2) Max-Min algorithm

The basic idea of Max-Min algorithm is similar to Min-Min algorithm, the scheduling policy of Min-Min algorithm is always a priority to the task of scheduling short, resulting in a long time not to perform the long task, so if there are the larger number of short tasks, and fewer number of long tasks in the collection, long tasks can be priority scheduling, reducing the waiting time of long tasks. The basic idea of Max-Min scheduling algorithm is a long task priority scheduling, the resource of strong computing capability allocates to a long task. Max-Min scheduling algorithm has better load balance and better time complexity. But when there are a larger number of long tasks in the centralized case, because the priority of long task scheduling, so may increase the system's Makespan.

(3) Min-mean algorithm

Min-mean algorithm is a typical resource Scheduling Algorithm of secondary distribution [4,5], there are two stages during the execution of the algorithm, the first stage is pre-scheduling, the use of Min-Min algorithm to pre-scheduling, and the second stage, the resource that execution time is longer than meanCT definite to heavy load resources, for tasks of heavy load on resources, if assigned to other grid resources can reduce the Makespan of the overall task set, then reallocated to other resources, so as to balance the load, get smaller Makespan.

(4) OPT-Min-Min algorithm

OPT-Min-Min scheduling algorithm [6] is similar to the Min-mean algorithm, all of which belong to the idea of secondary distribution applications. Difference between the two algorithms is that the definition of a heavy load resources. After using the Min-Min algorithm pre-scheduling, OPT-Min-Min scheduling algorithm defines the resource of the longest execution time as heavy load resources, for tasks of heavy load, if the resources allocated to other loads can reduce Makespan, assigned out, update Makespan, re-find the longest execution resources, and so on, until it can not be allocated so far. In the task sets of more long tasks, OPT-Min-Min scheduling algorithm is able to demonstrate a good load balancing and higher resource utilization.

(5) Improved Max-Min algorithm

Improved Max-Min scheduling algorithm, namely Max-Min heuristic improved job scheduling algorithm [7-8]. The algorithm idea is to have the tasks of the maximum execution time assigned to the appropriate resources, which can reduce the time to complete the job, thus minimizing the task set's Makespan.

(6) Balance Max-Min algorithm

Balance Max-Min scheduling algorithm [9], namely Max-Min heuristic grid resource scheduling algorithm based on load balancing. Further reduce the length of time to complete, load imbalance, and low resource utilization for the Max-Min algorithm, Balance Max-Min scheduling algorithm uses the idea of redistribution to reduce completion time, balancing the load and improving the effective utilization of resources. Firstly, using Max-Min algorithm to pre-scheduling. On the basis of pre-scheduling, the resource of the longest execution time definite as heavy load resources, for tasks of heavy load, if the resources allocated to other loads can be reduced Makespan, allocate out, update Makespan, rediscover the resources of the most long execution time , and so on, until the date can not be allocated. In a task set of more short tasks, Balance Max-Min algorithm can achieve high efficiency.

(7) $QPS_{Max-Min<>Min-Min}$ algorithm

$QPS_{Max-Min <> Min-Min}$ algorithm is based on QoS and the predict mechanisms of Min-Min and Max-Min algorithm [10]. Firstly, the algorithm divides the tasks of the task set by QoS into high and low QoS. first, the task for the high QoS is scheduled. According to the characteristics of applicable Min-Min and Max-Min scheduling algorithm, namely Min-Min algorithm is more applicable to the situation of the long task than of short tasks, while Max-Min algorithm is more applicable to the case of short tasks than of long tasks, so a prediction mechanism predicts the distribution of long-short tasks in task set, you can better to choose scheduling policy. The idea of this algorithm is based on SD standard deviation (Standard Deviation) [11] which is calculated the expectations of every task in task set to the choice of the scheduling algorithm, if the deviation of any two jobs expected completion time is greater than the standard deviation in position appears the first half of the task queue, Min-Min algorithm is used, otherwise, the use of Max-Min algorithm; then, for low QoS tasks, which also performs the above operation.

In a heterogeneous grid environment, when the long task is less and the short is relatively more in task set, the complete time of the system is very likely depend on short missions' execution time, when the heterogeneous of resources is high, using of Max -Min algorithm will make a lot of short tasks on the resources of poor performance implementation, thereby increasing the completion time of the system, if using Min-Min algorithm will not be scheduled of a long time task, therefore ,this paper packet secondary distribution grid resource scheduling heuristic algorithm GTSD (heuristic grid resource scheduling algorithm Based on Group of Task and Second Distribution) algorithm based on task-based, balancing load fundamentally, improving resource utilization, minimizing the Makespan of the task set , improving grid system performance, achieving customer satisfaction.

3. GTSD Algorithm

3.1 Algorithm Thoughts

The definition of the GTSD Algorithm:

Definition 1 : Long Tasks Group: The tasks appellation after the SD standard deviation.

Definition 2: Short Tasks Group: The tasks appellation before the SD standard deviation.

Definition 3: Heavy Load Resources: resources needs the maximum Completion Time

Definition 4: Light Load Resources: all the other load resources except the Heavy Load Resources

The GTSD thoughts presented in this paper: grouped by the SD standard deviation, each one in Long Tasks Group is assigned to inferior calculable resources to execute in sequence of execution time from long to short while tasks in the Short Tasks Group are pre-scheduled by Min-Min algorithm. After then, all the tasks from the Heavy Load Resources are assigned to the Light Load level to execute with the adoption of secondary allocation. Under the guidance of this ideology, the balance and resource utilization rate of the whole scheduling process is kept well and the accumulation of the Makespan can be minimized.

3.2 The Secondary Allocation Algorithm Based on the Grouping Task

The GTSD algorithm presented in the paper realize the balance of the loads, the enhancement of resource utilization rate and the minimality of the task sets mainly through two procedures: the sectionalization of tasks and secondary allocation.

The primary task is to sectionalize tasks and its criterion is to locate the place beyond the SD standard deviation. The calculation of the SD standard deviation needs the following preparations:

Calculate the expected time every task needs. The computational formula for the expected time can be gained by formula (1); the expected execution time ETij of the total tasks can be gained through its quantity, instructions, the network bandwidth and the computation speed;then its ETC matrix [12]can be got. Detailed methods: First, permute the tasks on task set ascendingly according to the expected time and arrange them into the TQ(task queue) . During the first cycle, all the tasks on the original task set are to be calculated seperately about its expected time of the whole resources.

$$CT_{ij} = ET_{ij} + r_j \tag{1}$$

In formula (1), CT_{ij} refers to the expected completion time; ET_{ij} refers to the expected execution time; r_j refers to the ready time awaiting computation for the grid resources. During the second cycle, it singles out the minimum completion time and matches it to the related resources.

Under the situation that one task matches to several grid resources, it will match the task to the resource with the minimum resource utilization in order to obtain a better balance of the loads.The efficient resource utilization can be got by the formula (2).

$$ru_j = \frac{\sum_{i=1}^{s}\left(te_i - ts_i\right)}{T} \tag{2}$$

The formula (2) refers to the proportion of the resources executed on the resource R_j than on the total resources. In the formula, s refers to the lumped quantity; tis refers to the starting time when task t_i runs on the resource R_j; te_i refers to the ti ending time task t_i runs on the resource R_j; T refers to the executed time of the whole application which is the time that the difference between the maximal task ending time and the minimal task starting time until now.T can be got by the formula (3).

$$T = \max\left(te_i\right) - \min\left(ts_i\right) \tag{3}$$

In which, i refers to the tasks which have been executed by far.

The average completion time aveCT is shown in the formula (4). The SD standard deviation of the expected completion time of the whole tasks can be got by the formula (5).

$$aveCT = \frac{\sum_{i=1}^{s} CT_{ij}}{s} \tag{4}$$

$$SD = \sqrt{\frac{\sum_{i=1}^{s}(\mathrm{CT}_{ij} - \mathrm{aveCT})^2}{s}} \tag{5}$$

After calculating the SD, single out the place greater than SD among the task queue TQ. Then, note all the tasks behind the place as the Long Sequence Task Group and the other part in front of the place as the Short Task Group. Assign the inferior calculable resources to execute in sequence of tasks from maximum to minimum first, and pre-dispatched the tasks in the place less than SD by Min-Min algorithm.

After the sectionalization and pre-scheduling of the tasks, the balance of the loads is broken. We use the secondary allocation to assign the tasks in Heavy Load Resources to the Lesser Load Resources to keep balance. These are the specific operations of the secondary allocation:

1)Permute the tasks in Heavy Load Resources ascendingly according to the expected time

2)Proposed that the task with the minimal executed time is assigned to all the other resources renewedly, refresh the executed time of all the resources.

①If among all the resources, the maximal executed time is less than Makespan, the task will be assigned to the resources with the minimal executed time.

②If among all the resources, the maximal executed time is larger than Makespan, we should consider selecting other tasks on this task set for the renewed allocation.

If the procedure①and②have tasks to assign again, renew executed time of all the resources and obtain Makespan again.

Executing procedure ① and ② until the maximal executed time can no longer be assigned.The algorithm ends and we get the final Makespan.

According to the description in chapter 3.2, we present the Heuristic Grid Resource Scheduling Algorithm Based on Group of Task and Secondary Distribution shown as the algorithm 1:

The Algorithm 1:Heuristic Grid Resource Scheduling Algorithm Based on Group of Task and Secondary Distribution

```
Input: Task Set $T_i$(i=1,2,…,m), Resource Set Rj(j=1,2,...,n),
     ETC(m*n) Matrix
Output: Completion Time Makespan, Task Scheduling Table
GTSD($T_i$, $R_j$, ETC)
BEGIN
// Predict initial data resources
1)    For all the tasks $T_i$
2)      For all the resources $R_j$
3)        $CT_{ij}=ET_{ij}+r_j$
4)      END FOR
5)     END FOR
6)    Calculate the earliest completion time and its related resources
7)    Calculate SD
8)    permute the free tasks ascendingly according to the completed time and arrange them
into the task queue TQ.
9)    Find the place p where two continous $CT_{ij}$'s difference larger than SD.
10)   Do Num($T_i$-S) Long Sequence Tasks
11)     Assign $T_i$ to $R_j$ which has the latest completion time without assignment
12)     Delete $T_i$ in list MT
13)     Refresh the ready time for $R_j$
14)     Noted $RC_j$ in resource set as assigned
15)     Refresh $CT_{ij}$ for all the i
16)   End Do
17)   Do Num S Short Tasks
18)     Find Task $T_i$ with the minimal completion time
19)     Assign Resource $R_j$ which has the minimal and earliest time to $T_i$
20)     Delete $T_i$  in  List MT
21)     Refresh the ready time for $R_j$
22)      Refresh $CT_{ij}$ for all the i
23)   End DO
24)    DO WHILE with Makespan resources RmaxCT has tasks to assign
25)    permute the tasks on RmaxCT ascendingly according to the executed time
26)    FOR all the tasks TmaxCT assigned to resources RmaxCT
27)     FOR another n-1resources
28)     IF($CT_j$+ET<Makespan && $CT_j$+ET minimum)
29)      Reschedule T to resource $R_j$
30)      Refresh CT
31)     END IF
32)    END FOR
33)    IF all the tasks assigned, out of the cycle
34)   END FOR
35)    permute the resources ascendingly according to the executed time $CT_j$
36)    END DO
37)   END IF
38)   Output Makespan and Task Scheduling Table
```

3.3 Analysis of the Algorithm

In the situation when massive short tasks needs to be scheduled while the long tasks are less than them, if we take the Min-Min scheduling algorithm, it will result in unbalance due to the lack of the scheduling on long tasks; if we take the Min-Min scheduling algorithm, it will result in the time-lasting lack of the scheduling on prior tasks and although having been scheduled, it will be executed on inferior calculable resources and this will reduce the completion time of the whole task set and cause the low resource utilization .

The GTSD Algorithm presented in the paper absolutely solves the above-mentioned problems.It divides the task set into the Long Sequence Task Group and Short Task Group, applies different scheduling policies to make tasks able to match the resources and minimize the Makespan and balance the whole system's loads. When executing GTSD Algorithm, the inputs m and the resources n are limited; the time complexity about the executive cycle is O(m*n);the executive cycle 10)-23)'s time complexity is O(m); the executive cycle 24)-37)'s time complexity is O(m*n) ; the whole time complexity is O(2m*n+m); the output is Task Set Makespan and Task scheduling table Table.The situation about infinite loop will never appear.

4. Instance Analysis

The following instances are used to verify the effection of the GTSD Algorithm. Proposed the user submit the Task set {T_1, T_2, T_3, T_4, T_5, T_6} to the Grid Resource Allocation Manager. There are three available resources through the resource searching and the task set is {R_1, R_2, R_3}.All the ETC matrix in task set is shown as the Table 1:

Table 1. ETC Matrix

	R_1	R_2	R_3
T_1	3	4	5
T_2	5	9	10
T_3	7	16	17
T_4	9	22	23
T_5	11	33	34
T_6	28	29	30

First,permute the minimal expected completion time on the original task set ascendingly {T_1，T_2，T_3，T_4，T_5，T_6}.Calculate SD=9.922 through the formula(5).The place lager than SD is between T_5 and T_6.Divide the task set into two groups: Long Sequence Task Group:{T_6} and Short Task Group {T_1，T_2，T_3，T_4，T_5}.Assign the Task T_6 which executes the longest time to resource R_3 which has the most inferior calculable resource. Then scheduling the short tasks T_1~T_5 through the Min-Min algorithm. The consequencee is shown as the Figure 1.

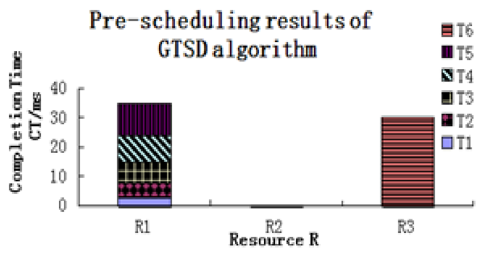


Figure 1. Pre-scheduling Results of GTSD Algorithm

After Pre-scheduling, the executive time of R_1 become 35ms while the R_2 is still 0ms and R_3 is 30ms. Permute the Resource group {R_1, R_2, R_3} ascendingly according to the executive time and get the new resource group {R_2, R_3, R_1} and we will get the Makespan resource {T_1, T_2, T_3, T_4, T_5} on R_1.According to GTSD Algorithm procedure 28, we can assign T_1 to R_2 and get the re-scheduling results: Makespan=32ms shown in Figure 2.

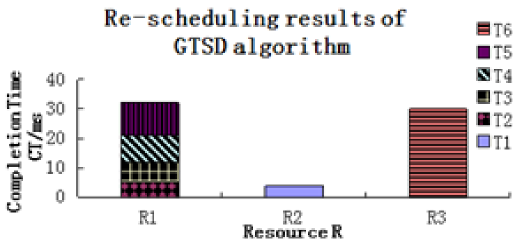


Figure 2. Re-scheduling Results of GTSD Algorithm

After re-scheduling, the executive time of R_1 becomes 32ms; the one of R_2 becomes 4ms; and the one of R_3 is still 30ms.Permute the resource group {R_1, R_2, R_3} ascendingly according to the executive time and get the new resource group {R_2, R_3, R_1} and we will get the Makespan resource {T_2, T_3, T_4, T_5} on R_1.According to GTSD Algorithm procedure 28, we can assign T_2 to R_2 and get the re-scheduling results: Makespan=30ms shown in Figure 3.

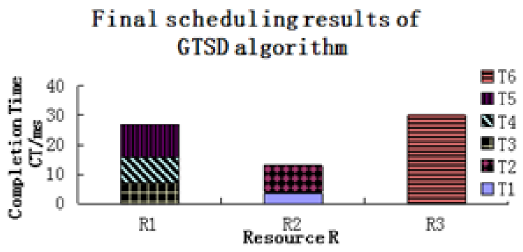


Figure 3. Final Scheduling Results of GTSD Algorithm

5. The Results and Performance Analysis

The paper quotes the classical grid simulation model standards raised by Braun’ s etc. [13, 14]. The standard originates from isomerism of tasks and resources to simulate four common types of grid scheduling algorithm: GTSD、Min-Min、Max-Min、Balance Max-Min under 12 situations in grid environment.

5.1. Experimental Environment

Dell T 110 II server, operating system of Microsoft Windows Small Business Server 2011, Xeon E3-1220V2 3.10GHz, 8GB internal storage.

5.2 Performance Analysis

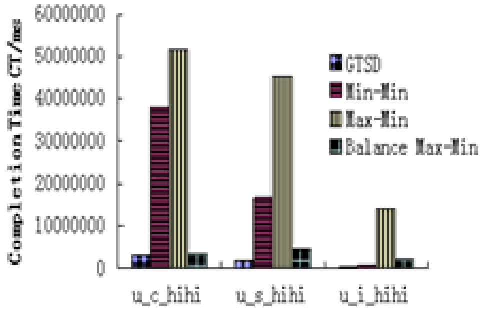
The experiment is conducted under the above experimental environment and the simulation model of the benchmark, choosing 128 resources and 1024 tasks to produce ETC matrix in a equally distributed way.Under this standard, we compare the executive time among the four algorithms :GTSD、Min-Min、Max-Min、Balance Max-Min.The results are shown in the Table 2[15].

Table 2. The Experimental Results of GTSD Algorithm and Other Three Algorithms

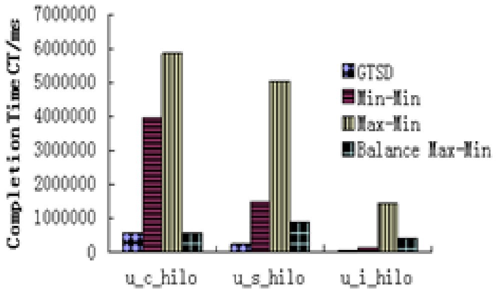
Instances	GTSD	Min-Min	Max-Min	Balance Max-Min
u_c_hihi	3.302×10^6	3.7919×10^7	5.17986×10^7	3.33501×10^6
u_c_hilo	5.654×10^5	3.9461×10^6	5.84904×10^6	5.76957×10^5
u_c_lohi	8.801×10^2	4.7742×10^3	5.83739×10^3	9.53354×10^2
u_c_lolo	1.193×10^2	4.7584×10^2	5.88147×10^2	1.31069×10^2
u_s_hihi	1.923×10^6	1.6727×10^7	4.53254×10^7	4.50027×10^6
u_s_hilo	2.292×10^5	1.4652×10^6	5.03382×10^6	8.85052×10^5
u_s_lohi	6.031×10^2	2.6292×10^3	4.87301×10^3	1.10051×10^3
u_s_lolo	9.107×10^1	2.8188×10^2	4.99644×10^2	1.55069×10^2
u_i_hihi	3.023×10^5	8.9004×10^5	1.41631×10^7	1.99065×10^6
u_i_hilo	4.244×10^4	1.2933×10^5	1.45977×10^6	4.15273×10^5
u_i_lohi	4.073×10^2	8.4122×10^2	3.04612×10^3	8.60166×10^2
u_i_lolo	5.015×10^1	7.6229×10^1	3.72865×10^2	1.18801×10^2

According to the scheduling results given in Table 2 of GTSD, Min-Min, Max-Min and Balance Max-Min four algorithms, we can draw the histogram representation, as shown in Figure 3.

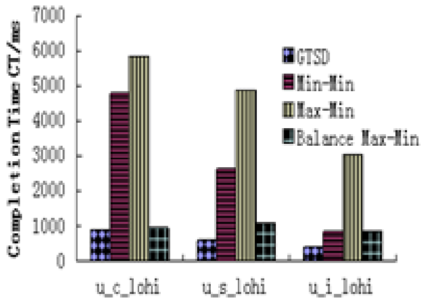
We can draw conclusions from the above experimental results: whatever the distribution of the tasks are, the GTSD is apparently superior to the other three ones, especially under the situation when long tasks are less than short tasks. The efficiency of the GTSD is far superior to the other three ones.



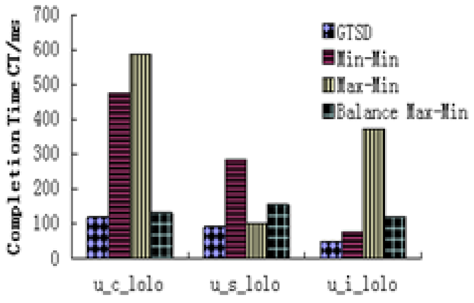
(a) The Experimental Results of hihi



(b) The Experimental Results of hilo



(c) The Experimental Results of lohi



(d) The Experimental Results of lolo

Figure 3. The Experimental Results of GTSD Algorithm and other Three Algorithms

6. Conclusion

The GTSD pointed out in the paper, aiming at the Max-Min algorithm about its unbalanced loads, low resource utilization rate and long completion time, adopts the heuristic algorithm and takes advantage of the prediction mechanism. According to the SD, we can anticipate the distribution about the standard of the tasks; then based on the place, we can anticipate the quantity of the long tasks and the short tasks. Dividing the task set into the Long Sequence Task Group and Short Task Group, we apply different scheduling policies to it, adopting the Long Sequence Task first to assign them to the inferior calculable resources. Min-Min algorithmic scheduling is adopted to the short tasks. After pre-scheduling, we employ secondary allocation, balancing loads rapidly by assigning heavy load resources to lesser load resources.It improves the efficient resource utilization and lowers the complexity of the algorithm. And our further direction is to select a more precise criterion to anticipate the distribution of the task set.

Acknowledgements

This work was financially supported by Hebei Provincial Natural Science Foundation of China (F2012203087),the Special Fund Project for National Natural Science Foundation of China (61040023) and National Natural Science Foundation of China (61073060).

References

[1] Ian Foster, Carl Kasselman, Steven Tuecke. The Anatomy of The Grid. Intl J. Supercomputer Application, 1-25 **(2001)**

[2] T Kokiavani, Dr D l. George Amalarethinam. Load balanced Min-Min algorithm for tatic meta-task scheduling in grid computing. International Journal of Computer Application, 20(2): 43-49, **(2011)**

[3] Liu Jue-Fu, Gang Li. An improved Min-Min grid tasks scheduling algorithm based on QoS constraints. In: Proceedings of Interational Conference on Optics, Photonics and Energy Engineering. **(2010)** 281—283; Wuhan, China

[4] Kamalam G K, Murali Bhaskaran V. A new heuristic approach: Min-mean algorithm for scheduling meta-tasks on heterogeneous computing systems. IJCSNS International Journal of Computer Science and Network Security, 10(1): 24-31 **(2010)**

[5] Kamalam G K, Murali Bhaskaran V. New enhanced heuristic Min-mean scheduling algorithm for scheduling meta-tasks on heterogeneous grid environment. European Journal of Scientific Research, 70(3): 423-430, **(2012)**

[6] Zhang Zhong-Ping, Wen Li-Juan. OPT-Min-Min: The grid resources scheduling algorithm based on load balance. Journal of Chinese Computer System, 35(7): 1573-1577 **(2014)**

[7] Devipriya S, Ramesh C. Improved Max-Min heuristic model for task scheduling in cloud. In:Proceedings of Green Computing, Communication and Conservation of Energy (ICGCE), **(2013)** 883—888; Chennai, India

[8] Li Xiao-Fang, Mao Ying-Chi, Xiao Xian-Jian. An improved Max-Min task-scheduling algorithm for elastic cloud.In: Proceedings of Computer, Consumer and Control (IS3C),2014 International Symposium, **(2014)** 340—343; Taichung, China

[9] Tarun Kumar Ghosh, Rajmohan Goswami, Sumit Bera. Load balanced static grid scheduling using Max-Min heuristic. In: Proceedings of Parallel Distributed and Grid Computing (PDGC), 2012 2nd IEEE International Conference, **(2012)** 419-423;Solan, India

[10] Singh M, Suri P K. QPS Max-Min<>min-min:A QoS based predictive Max-Min, Min-Min switcher algorithms for job scheduling in a grid. Information Technology Journal, 7(8): 1176-1181 **(2008)**

[11] Kobra Etminani, Naghibzadeh M. A Min-Min Max-Min selective algorihtm for grid task scheduling. In: Proceedings of Internet, 2007. ICI 2007. 3rd IEEE/IFIP International Conference in Central Asia, **(2007)**, 1-7; Tashkent, Uzbekistan

[12] Zhou Wei, Bu Yan-Ping, Zhou Ye-Qing, The application of an improved cultural algorithm in grid computing. In: Proceedings of Control and Decision Conference (CCDC), 2013 25th Chinese, **(2013)** 4565—4570; Guiyang, China

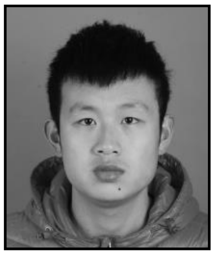
[13] Tracy D Braun, Howard Jay Siegel, Noah Beck. A comparison of eleven static heuristics for mapping a class of independent tasks onto heterogeneous distributed computing systems. Journal of Parallel and Distributed Computing, **(2001)** 61(6): 810--837

[14] Braun T, Siegel H, Beck N. A comparison study of static mapping heuristics for a class of meta-tasks on heterogeneous computing sy-stem. In: Proceedings of In 8th IEEE Heterogeneous Computing Workshop (HCW'99). **(1999)** 15-29; San Juan, Puerto Rico

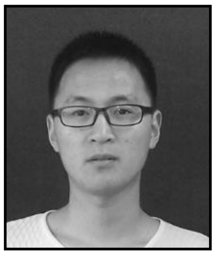
[15] Ahmadi S M, Masoumi B. Resource allocation in grid systems using collective case based reasoning methods and learning automata. In: Proceedings of AI & Robotics and 5th RoboCup Iran Open International Symposium (RIOS), 2013 3rd Joint Conference. **(2013)** 1-5;Tehran, Iran:

Authors

Zhongping Zhang, Male, he was born in 1972, professor, Ph.D., post-doctoral, CCF Senior Member (E20-0006458S). His main research interests are the grid computing, data mining and semi-structured data etc. He has undertaken 1 project of provincial level and participated in 2 projects funded by national natural science foundation of China. He rewarded the provincial Scientific and Technological Progress second-class Award. On the domestic and international academic conferences and journals, He published more than 100 papers, 20 of them are cited by EI.



Yupeng Feng, Male, Born in 1989, Current Master Students, the main research interest is grid computing.



Shan Zhang, Male, he was born in 1990, Current Master Students, the main research interest is cloud computing.

Ying Sun, Female, she was born in 1989, Current Master Students, the main research interest is data mining and the outlier detection.