Journal of Computer Science 5 (10): 745-750, 2009
ISSN 1549-3636

Job Type Approach for Deciding Job Scheduling in Grid Computing Systems

Asef AL-Khateeb, Rosni Abdullah and Nur'Aini Abdul Rashid
School of Computer Sciences, University Sains Malaysia,
11800 USM, Pulau Pinang, Malaysia

Abstract: Problem statement: Meta-scheduling has become very important due to the increased number of submitted jobs for execution. **Approach:** We considered the job type in the scheduling decision that was not considered previously. Each job can be categorized into two types namely, data-intensive and computational-intensive in a specific ratio. Job ratio reflected the exact level of the job type in two specific numbers in the form of ratio and was computed to match the appropriate sites for the jobs in order to decrease the job turnaround time. Moreover, the number of jobs in the queue was considered in the batch decision to ensure server-load balancing. **Results:** The new factor that we considered namely, the job ratio can reduce the job turnaround time by submitting jobs in batches rather than submitting the jobs one by one. **Conclusion:** Our proposed system can be implemented in any middleware to provide job scheduling service.

Key words: Meta-scheduling, job scheduling, job ratio, computational jobs, data-intensive jobs

INTRODUCTION

Currently, many applications can be divided into jobs that turned to grid computing[15] to meet their computational and data storage needs. The emergence of scientific applications and projects[16,17] leads to increased the number of applications. Using distributed grid resources can benefit these applications such as speeding up the applications process and utilizing the idle resources. However, this is possible only if the resources are scheduled well. Grid scheduling is defined as the process of making scheduling decisions involving allocating jobs to resources over multiple administrative domains. This can include searching multiple administrative domains to use a single machine or scheduling a single job to use multiple resources at a single site or multiple sites. Scheduling applications in a Grid environment is significantly more complicated than scheduling applications for a traditional supercomputer because of the heterogeneous nature of the Grid systems.

There are two different types of scheduler systems namely, the local scheduler and the global scheduler (Meta-Scheduler). The local scheduler schedules the jobs within its own managed site. Typically, these local schedulers cannot schedule jobs to some other available sites, rather it has localized control. The most popular local schedulers are: Load Sharing Facility (LSF)[8], the Open Portable Batch System (PBS)[10], Sun Grid Engine (SGE)[11] and Condor[9]. On the other hand Meta-Scheduler manages jobs among available sites in the grid environment. Although there are already available methods for Meat-scheduler, there is still a need to improve the scheduling techniques because the number of jobs running on grid has increased that cause the system to degrade. Therefore, our focus is to optimize the efficiency of the Meta-Scheduler by providing more enhanced techniques. The task of the scheduler is to dynamically identify and characterize the available resources and to allocate the most appropriate resources for the given jobs. The resources are typically heterogeneous, locally administered and accessible under different local access policies and thus the problem to be addressed in this study is how to select the best site for submitting the underlying jobs and how many jobs the system should submit each time.

In the previous study, the Meta-Scheduler decides the best site based on the number of jobs that is waiting in the queue. While other proposed methods select the criteria based on the data access cost; but these two main criteria are insufficient for the best decision, because they neglect the job types, which include: computational jobs and data-intensive jobs. The computational jobs are the jobs that require processing time more than data access time. However, data-intensive jobs are the jobs that require data access time more than processing time. Consequently, our scheduler selects the sites that have more processing capabilities

Corresponding Author: Asef Al-Khateeb, School of Computer Sciences, University Sains Malaysia, 11800 Penang, Malaysia

for the computational jobs and the sites that house the required data or very closed to the sites that house the required data in order to reduce the data access cost for data-intensive jobs. Knowing that the cost considered here is the time required for getting the required data.

In this study, the problem is viewed in two-fold: The first-fold is how to select the best site. The selection process based on the selection criteria. Identifying the criteria set is not an easy task, because most of the criteria are dynamic and changes over time. The second-fold is how many jobs the system can submit at a time. Grouping jobs together in one batch is a challenge problem because the number of jobs with each batch varies from one batch to another batch. Indeed, determining the numbers of jobs in each batch are subject to many factors such as: Job type, job requirements and resource availability.

The main objective of this study is to produce an enhanced Meta-scheduling system that provides: queuing service, matching service and batching service in order to establish server-load balancing and to reduce the job turnaround time, which is the time elapse from when the job is submitted to the grid site until the job finishes its execution.

Related work: Most previous Meta-scheduling work has considered data access cost and waiting time for jobs in the queue issues. We discuss study that has recognized the importance of these issues for job scheduling in the grid.

Jiang *at el*.[1] consider the job behavior in the job waiting queue as an important factor for scheduling algorithm. The data access cost also is aggregated with the job waiting queue in order to reduce the job turnaround time. Results obtained from the simulator show better system performance than other works that consider only the data access cost in the scheduling decisions. Knowing that the data access cost may include: Data transfer time, data locations, storage access latency and response time.

Ranganathan and Foster[2] propose a framework that include Meta-scheduling, local scheduler and dataset scheduler. Results show that each scheduler may influence other scheduler, Chicago Grid simulator has been designed and implemented for achieving this work only and thus the Chicago Grid simulator is a customize simulation tool. Regarding to the external scheduler that also called meta-scheduler, the data access cost and the jobs waiting in queue are the only criteria that considered in the decisions.

Elmroth and Tordsson[3] the main goal of this study is to shortest time for job completion including the time for file staging, jobs waiting time in the queue and job execution time. This goal is achieved by matching the jobs to the appropriate resources using prediction method because the resources are heterogeneous and administrated by different administrator domains. Matching jobs with resources is done by a decentralized brokering system. Each user in the grid has its own broker and the broker inquire other brokers of other recourses in order to find the best resource. The decentralized system is scalable, but the decisions are made without global vision and thus the scheduling results will be degraded.

Shi *at el*.[4] provide an adaptive meta-scheduling for data-intensive application. The data access cost, jobs in a waiting queue and the computational grid are all considered in the proposed system. The best site to be selected for the underlying job should provide the highest computation (i.e., high speed of CPU). A good balancing between the computational power and data access cost that exist in grid site is achieved.

Ernemann *at el*.[5] proposed a meta-scheduler that considers the geographic site location by categorize the sites into time zones. The scheduler selects the site that provides the nearest distance to the required data for the job execution to reduce the job completion time. Hence, the data access cost is the main factor that focused in the scheduler.

Zhang *at el*.[6] propose a resource selection algorithm for the jobs. Each job can be submitted into multisite in order to reduce job execution time. But, in this method the job turnaround time may increases since the data transfer time increase among the selected multisite. On another hand this method may be worthy in grid environments that the number of available sites bigger than the submitted jobs, but this kind of grid environment is very rare occurred in reality.

Anjum *at el*.[7] Data Intensive and Network Aware (DIANA) is a meta-scheduler engine that schedule jobs, jobs priorities, jobs queue bulk submission and other access cost and computational capability in order to enable efficient global scheduling.

Some of previous approaches have been used greedy algorithm where jobs are submitted to a best site location (grid resource) without assessing the global cost of this action. However, this action can lead to a skewing in the distribution of resources and can result in large queue, reduced system performance and throughput and degradation for the remainder of the jobs. Obtaining local objectives may lead to achieve global objectives at the system, but it may takes more time in the optimization process and can only achieve global objective to some extends while in our system, global knowledge of jobs and grid resource are aggregated and used in the scheduler engine.

All the previous studies assume that all the jobs are same. Very few studies consider that the jobs can be either data-intensive jobs or computational jobs in same ratio. Our main contribution is to analyze the job type into two values. Values reflect the exact job type and how much ratio for each type in order to enable more efficient matching of the jobs to the appropriate resources. More details about how to measure the job ratio using predication method. On another side, the grid site capability is provided by ISP and thus the computational jobs submit into the sites that have more computational power and the data-intensive jobs submit into the sites that have less data access cost.

MATERIALS AND METHODS

Our solution encapsulates in a system that we termed as: Job-Based Meta-Scheduler System (JBMS), which consists of three main components namely, Job Analyzer and Monitor (JAM), Job Decider (JD) and Job Batcher (JB) as shown in Fig. 1.

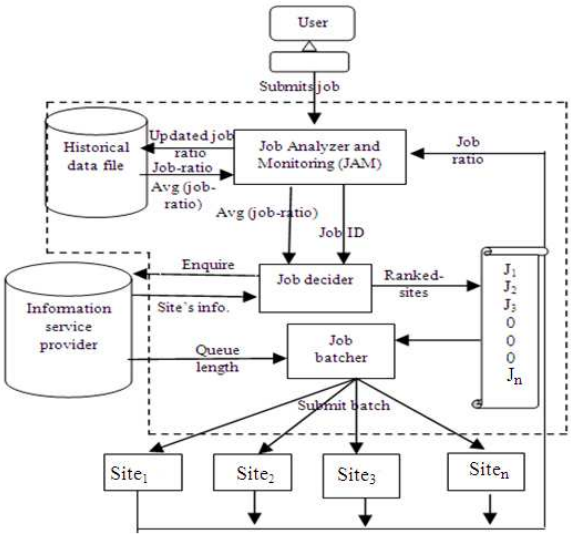

Fig. 1: Job-Based Meta-Scheduler System (JBMS) architecture

Job Analyzer and Monitor (JAM): The JAM component is responsible for analyzing the received job in terms of job type. The job types considered in this context are: Computational Jobs (CJ) and Data-Intensive Jobs (DIJ). The computational jobs are the jobs that require more execution time than data access time which represents the time required for input and output (I/O) operations, while the data-intensive jobs are the jobs that require more data access time than execution time. According to the job type the system can match the jobs with the appropriate resource. Therefore, our system matches the computational jobs with the sites that have more computational power which are represented by the number of the processes and their cycle speed. However, the system matches the data-intensive jobs with the sites that are very close to the required data files location in order to reduce data access time.

The key concept of our proposed solution is not only determined the job type, but also the job type percentage ratio or simply Job-Ratio (JR), which determines the exact job percentage requirement of either computation or data-intensive jobs. Each job has its own ratio that is divided into two values, the first value indicates the job computation execution requirement and the second value indicates the data intensive I/O requirement. For example, a job has ratio 3:1, this means that the job needs 75% computational execution and 25% data intensive I/O. This ratio will be used latter by the JD component to decide the exact appropriate site for submitting the jobs. In this context, the ratio is playing a major role in the selection decision which weights for each job type.

The JAM computes the JR from the historical data file which keeps information about the previously recorded jobs. Typically, the jobs as software programs require similar computational power and data access each time they execute, but there are some differences based on the program instructions flow. The job may use three files in one execution and may use four files in another execution. Therefore, JAM record the amount of time required for execution and the amount of time required for data access for all the jobs in the grid in the history file. The following time, when a job is submitted to the system, JAM computes the average of the execution time and the average of the data access time to come up with the JR.

If the job does not exist in the history file as if the job is being executed for the first time. In this case, JAM assigns an estimated JR for the underlying job depending on the average of other similar jobs. Once the JR is computed for the jobs, the JAM passes these jobs to the JD that manages the selection process and passes these jobs to the JB that manages the priorities and batches and submits the jobs to the appropriate site location for execution. JAM monitors the jobs under execution by measuring the job execution time and the data access time for each job in order to save the new data into the history file for further decisions. Due to the amount of times the job is executed the JR will be more accurate and reflect the exact job type.

The JAM measures the Job Execution Time (JET) by using the following equation:

$$JET = JTT - JIOT \quad (1)$$

Where:

JTT = Job Turnaround time, which is the period of time when the job is submitted into the site until the job finishes the execution

JIOT = Job Input and Output Time, which is the total time required for I/O operations where the job is being executed

Likewise, the JAM also monitors the jobs under execution by using some other existing tools and measures the JIOT by the following equation:

$$JIOT = JTT - JET \quad (2)$$

Accordingly, the job-ratio that has the form CJR: DIJR is computed as follows:

$$CJR = \frac{JET}{JTT} \times 100 \quad (3)$$

$$DIJR = \frac{JIOT}{JTT} \times 100 \quad (4)$$

CJR and DIJR are multiplied by 100% for clarity and to deal with CJR and DIJR as percentage ratios. Therefore, the JR (CJR: DIJR) are the values that reflect the exact job required time for execution and the required time for data access. Such that:

- CJR represents the execution time required by a job
- DIJR represents the I/O operations time required by a job

Job Decider (JD): The JD is responsible for finding the appropriate site for the underlying job. JD ranks the available sites for each job according to the following steps:

Step 1: There are several types of jobs which require more of the computational process and need more processing power to submit for the site which has more capability processing power. Thus the Process Power Ratio (PPR) is computed as follows:

$$PPR = \frac{Process\ power_{(Site)}}{\sum_{i=1}^{n} Process\ power_{(Site)}} \quad (5)$$

Step 2: Data-intensive jobs are the jobs that require data access more than processing power. The Data Power Ratio (DPR) is computed as follows:

$$DPR = 1 - \frac{Data\ access\ cost_{(site)}}{\sum_{i=1}^{n} Data\ access\ cost_{(site)}} \quad (6)$$

Data Access Cost is the aggregated cost for all required files for the underlying job and for each data access file the cost is the transfer time which is computed as follows:

$$Transfer\ time = \frac{File\ size_{(MB)}}{Bandwidth_{(MB/SEC)}} \quad (7)$$

Step 3: The number of the jobs in the queue should be considered in the selection process before submitting any group of jobs and to know the required time which remains in the queue, in order to ensure server load-balancing among the grid sites. Thus the Queue Ratio (QR) is computed as follows:

$$QR = 1 - \frac{No.\ of\ jobs\ in\ queue}{\sum_{i=1}^{n} No.\ of\ jobs\ in\ queue} \quad (8)$$

Step 4: For each site, the site ranked as follows:

$$Site\ rank = (PPR*CJR) + (DPR*DIJR) + QR \quad (9)$$

Step 5: Sort the site rank in descending order for each job in the job handler queue.

Job Batcher (JB): The JB collects the jobs in batches and sends the jobs to the selected ranked site. JB is responsible for determining the number of jobs for each batch based on the JAM and the Grid Information Service (GIS)[13,18] such as Network Weather Service (NWS)[14]. The JB gets the ranked sites from the JD and decides the job batching by performing the following steps:

Step 1: For each site, the available number of jobs in the site will be the number of the jobs in the batch and computed as follows:

$$Available\ No.\ of\ jobs = Queue\ lenght - current\ No.\ of\ jobs \quad (10)$$

Step 2: For each job in the job queue handler, the first job is assigned to the site that has the first rank. If the queue of the underlying site is full, then the job will be assigned to the second ranked site and so forth until all the jobs are assigned to the related sites.

Step 3: JB submits the jobs related to each batch to the correspondence site.

Step 4: JAM monitors all the jobs under execution and measures the JET and JDAT. Accordingly, the new job ratio will be computed and updated in the history file.TET and JDAT is computed for each job in the grid. Each time any job executed, the job ratio will be update. Since the grid sites are varying in their capability power, reference point should be used to uniform the PP for all the sites in order to expose the JET and JTT, as computed by the following equation:

$$RJTT = \frac{PP_{(site)}}{RPP_{(site)}} * JTT \quad (11)$$

Where:
RJTT = Ratio Job Turnaround Time
PP = Process Power for the Job Site
RPP = Reference Process Power which is constant value

Step 5: Very huge data may reordered in the historical file gradually over time. Therefore, a new method is used in order to keep only the average ratio by computing the new average each time a specific job is executed, as the following equation:

$$NewAv = \frac{OldAv * JETN + RJTT}{JETN + 1} \quad (12)$$

NewAV =New Average
OldAv = Old Average
JETN = Job Execution Time Number

RESULTS AND DISCUSSION

Table 1 reflects the results of applying the underlying solution. Each job of the submitted jobs has a ranked value for all the available sites. Related to these ranks the jobs batcher can submit the jobs to the selected ranked site as batches after calculating the available number of jobs in the site's queue. Table 2 shows the available number of jobs and job batches for each site.

This study describes the scheduling decision of jobs in grid computing systems. We have considered a new factor namely, the job ratio that can reduce the job turnaround time as a main objective of this study and the most desired issue for grid users. On another hand submitting jobs in batches is more efficient than submitting the jobs one by one as a result.

Table 1: Jobs and related ranked sites example

Job ID	Ranked site				
Job 1	1	5	4	2	3
Job 2	3	4	5	1	2
Job 3	2	1	5	4	3
Job 4	5	3	1	2	4
Job 5	4	1	3	2	5
Job 6	2	4	1	5	3
Job 7	3	1	5	4	2
Job 8	1	4	3	2	5
Job 9	5	4	3	1	2
Job 10	2	1	4	3	5

Table 2: Jobs batched example

Site ID	Available No. of jobs	Batches job
1	2	J_1, J_8
2	1	J_3
3	3	J_2, J_7, J_{10}
4	2	J_5, J_6
5	2	J_4, J_9

CONCLUSION

In this study, we have introduced a job resource matching policy. The job execution time and the data access time for each job is monitored and computed to provide the Job-Ratio. An elaborate prediction function is produced for computing the Job-Ratio based on the history file and other grid services tools. Also we introduced job batches policy that based on jobs' priorities and sites capabilities in order to reduce the time and balancing the workload among grid sites. Our system can be implemented in real grid middleware such as Globus[12].

ACKNOWLEDGEMENT

This study has been funded by University Sains Malaysia**.** Under the short-term Research Grant for "KNOWLEDGPROCESSING PLATFORM" [1001/PKOMP/817002].

REFERENCES

1. Jiang, J., H. Ji, G. Xu and X. Wei, 2008. Scheduling algorithm with potential behaviors. J. Comput., 3: 1-9. http://www.academypublisher.com/jcp/vol03/no12/jcp03125159.html
2. Ranganathan, K. and I. Foster, 2003. Simulation studies of computation and data scheduling algorithms for data grids. J. Grid Comput., 1: 53-62. http://www.metapress.com/link.asp?id=HJ25770J1825817U

3. Elmroth, E. and J. Tordsson, 2008. Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions. Future Generat. Comput. Syst., 24: 585-593. DOI: 10.1016/j.future.2007.06.001
4. Shi, X., H. Jin, W. Qiang and D. Zou, 2004. An adaptive meta-scheduler for data-intensive applications. Lecture Notes Comput. Sci., 3033: 830-837. DOI: 978-3-540-21993-4
5. Ernemann, C., V. Hamscher and R. Yahyapour, 2004. Benefits of global grid computing for job scheduling. Proceedings of the 5th IEEE/ACM International Workshop on Grid Computing, Nov. 8-8, IEEE Computer Society, Washington DC., USA., pp: 374-379. http://portal.acm.org/citation.cfm?id=1033267
6. Zhang, W., B. Fang, H. He, H. Zhang and M. Hu, 2004. Multisite Resource Selection and Scheduling Algorithm on Computational Grid. Proceedings of the 18th International Parallel and Distributed Processing Symposium, Apr. 26-30, IEEE Xplore Press, USA., pp: 105. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1303052
7. Anjum, A., R. McClatchey, A. Ali and I. Willers, 2006. Bulk scheduling with the DIANA scheduler. Nuclear Sci. IEEE. Trans., 53: 3818-3829. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=4033697
8. Zhou, S., X. Zheng, J. Wang and P. Delisle, 1993. A load sharing facility for large, heterogeneous distributed computer systems. Software-Pract. Exp., 23: 1305-1336. DOI: 10.1002/spe.4380231203
9. Thain, D., T. Tannenbaum and M. Livny, 2003. Condor and the Grid. In: Grid Computing: Making The Global Infrastructure a Reality, Fran Berman, A.J.G.H. (Ed.). John Wiley, Geoffrey Fox, ISBN: 0470853190, pp: 1012.
10. Robert, H., 1995. Job scheduling under the portable batch system, in job scheduling strategies for parallel processing. Lecture Notes Comput. Sci., 949: 279-294. http://portal.acm.org/citation.cfm?id=689372
11. Sundaram, B. *et al*., 2006. Sun grid engine package for OSCAR-a Google summer of code 2005 project. Proceeding of the 20th International Symposium on High-Performance Computing in an Advanced Collaborative Environment, May 14-17, IEEE Xplore Press, USA., pp: 41-41. DOI: 10.1109/HPCS.2006.42
12. Ferreira, L. and V. Berstis *at el*., 2002. Introduction to Grid Computing with Globus. 1st Edn., Copyright IBM., ISBN: 0-7384-9988-9, pp: 290.
13. Aktas, M.S., 2007. Information federation in grid information services. Department of Computer Science. Indiana University. http://grids.ucs.indiana.edu/ptliupages/publications/MehmetAktasThesis.pdf
14. Yousaf, M.M., M. Welzl and M.M. Junaid, 2007. Fog in the network weather service: A case for novel approaches. Proceedings of the 1st International Conference on Networks for Grid applications, Oct. 17-19, Lyon, France, pp: 1-6. http://portal.acm.org/citation.cfm?id=1386610.1386641
15. Foster, I., C. Kesselman and S. Tuecke, 2001. The Anatomy of the Grid: Enabling scalable virtual organizations. Lecture Notes Comput. Sci., 2150: 1-4. DOI: 10.1007/3-540-44681-8_1
16. Cho, K., 2007. A test of the interoperability of grid middleware for the korean high energy physics data grid system. Int. J. Comput. Sci. Network Sec., 7: 1-6. http://paper.ijcsns.org/07_book/200703/20070308.pdf
17. Holtman, K., 2001. CMS data grid system overview and requirements. Technical Report. http://citeseerx.ist.psu.edu/viewdoc/summary?DOI=10.1.1.20.9986
18. Czajkowski, K., S. Fitzgerald, I. Foster and C. Kesselman, 2001. Grid information services for distributed resource sharing. Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing, Aug. 7-9, IEEE Xplore Press, San Francisco, California, USA., pp: 181-194. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=945188