Scheduling Algorithm with Potential Behaviors

Jianhua Jiang
[1]College of Computer Science and Technology, Jilin University, Changchun, China
[2]Department of Information, Changchun Taxation College, Changchun, China
Email: jianhuajiang@yahoo.com

Huifang Ji, Gaochao Xu and Xiaohui Wei
[1]College of Computer Science and Technology, Jilin University, Changchun, China
Email: jihuifang_1986@yahoo.cn,{xugc, weixh}@jlu.edu.cn

***Abstract*—Scheduling algorithm for batch-mode data-intensive jobs is a key issue in data-intensive Grid applications. It focuses on how to minimize the overhead of transferring the required data set to the executing grid site. Existing approaches pay attention to the access cost of a data-intensive job at each executing grid site for replicating the required data set. However, they neglect the influence from potential behaviors of jobs in the waiting queue at each grid site when the access cost is evaluated. In this paper, we consider the influence of potential behaviors on the access cost, and propose a data-intensive job scheduling algorithm with potential behaviors. Furthermore, the causation of potential behaviors is analyzed. The simulation result in OptorSim shows that it has better performance in mean job time of all jobs, total number of replications, total number of local files accesses and effective network usage than the scheduling algorithm based on access cost.**

***Index Terms*—distributed computing, grid computing, data grid, job scheduling, access cost, replica replacement**

I. INTRODUCTION

With the development of grid computing, Data Grid [1,2] as an important branch of Grid Computing focuses on supporting an efficient management mechanism for controlled sharing and large amounts of distributed data. Data-intensive jobs are one major kind of jobs in Data Grid, and their scheduling strategies are regarded as one of the most important research fields. Batch-mode data-intensive jobs require access to a large amount of data (terabytes or petabytes). Therefore, it is the key how to process with the minimal access cost. Existing data-driven job scheduling algorithms [3-7] have been proposed to solve this issue by evaluating the access cost based on some important influencing factors, such as network bandwidth, network load, CPU workload etc. However, distributions of data set between scheduling and run time are always changing dynamically when replica replacements occur frequently during this period at each grid site. The potential behaviors of jobs in the waiting queue are supposed to be analyzed if batch-mode data-intensive jobs are required to be scheduled.

In this paper, we address the overall processing performance in the scheduling algorithm based on Access Cost with Potential Behaviors (ACPB). ACPB, which is an application-centric scheduling strategy and the evolution of scheduling algorithm based on access cost, focuses on the influence of potential behaviors of jobs in the waiting queue at each grid site on the heuristic function of access cost. The underlying influencing factors to potential behaviors are analyzed and discussed. To get the situation of potential behaviors, a simple and decentralized feedback mechanism is given to support the job scheduling process. The features of ACPB are summarized as bellow:

a) The access cost is recalculated with potential behaviors.

b) The native replica replacement strategy and the length of waiting queue are considered as two main important influencing factors on potential behaviors.

c) A decentralized feedback mechanism is created to support resource broker to make decisions in OptorSim.

To evaluate the effect of ACPB, it is simulated by OptorSim [8] and compared with the scheduling algorithm based on access cost (AC) [3]. The simulation result shows that ACPB has better performance in mean job time, number of replications, total number of local files accesses and effective network usage than these of the existing scheduling algorithm based on Access Cost (AC).

The rest of this paper is organized as follows. In section 2, gives an overview of previous work on job scheduling algorithms. In section 3, the proposed scheduling algorithm based on Access Cost with Potential Behaviors (ACPB) is presented. In section 4, simulation experiments with ACPB and AC in OptorSim are performed and discussed. Section 5 concludes the paper and outlines some future research work.

II. RELATED WORK

According to the different performance goals, the scheduling systems or algorithms can be classified into

This work is partially supported by National Natural Science Foundation of China (Grant No.60703024) and Jilin Province of Science and Technology under Grant No.20070122 and 20060532.
Gaochao Xu is the corresponding author.

three categories: system-centric, economy-based and application-centric systems.

A system-centric scheduling system focuses on the overall performance of the whole set of jobs and the whole Grid system. Condor [9] aims to increase utilization of workstation by hunting idle workstations for sharing job execution. Condor follows an approach to scheduling that lies between the centralized and decentralized scheme. For scheduling jobs, each workstation itself is responsible for maintaining the local queue of jobs to be run and scheduling the jobs onto idle workstations for execution.

Condor-G [10] leverages the advantages of both Condor and Globus Toolkit [11]. Globus Toolkit is a software infrastructure for setting up a Grid environment across multiple administrative domains, which supports resources management, secure file transfer, information discovery and secure authentication and authorization in a Grid environment. Based on Condor, Condor-G makes use of the mechanisms provided by Globus Toolkit to cross the boundaries of real institutions, aiming at utilizing the idle workstations among these institutions.

An economy-based scheduling system is based on the idea of market economy. Under this scheme, scheduling decisions are made based on the economy model. Nimrod-G [12] implements a relationship mechanism of producer and consumer in Data Grid. Economic methods include: commodity market model, posted price model, bargaining model, tender/contract-net model, auction model, bid-based proportional resource sharing model, trading model, and monopoly/oligopoly model. These models compare the different costs of replica replacement and remote access.

An application-centric scheduling system tries to maximize the performance of individual applications. Adaptive scheduling model, such as AppLes [5], is one of the most important application-centric scheduling systems. AppLes focuses on the run-time resource availability and the development of scheduling agent for parallel meta computing. Each scheduling agent is implemented by the task mapping mechanism. For scheduling decision, scheduling agents will consider the different requirement for different applications based on their load prediction and dynamic resource availabilities. To gain the precise information of dynamic resource availabilities and other influencing factors, AppLes uses NWS (Network Weather Service) [13] to monitor these factors.

The scheduling model based on access cost, such as Chameleon [14], is a branch of application-centric scheduling strategy. Chameleon calculates the cost by the relationship between the grid site which holds data files and the grid site where the job will be run. The job is scheduled by the access cost of these different grid sites. The scheduling algorithms based on access cost can be classified into two categories based on if prediction techniques are adopted or not. Application-centric scheduling algorithms based on prediction techniques [15, 16, 17] use statistics techniques to predict the resource behavior. Traditional scheduling methods focus on the application of statistics.

1. Schopf [17] introduces a stochastic scheduling method in scheduling input data to processors at run-time. It assumes the mean of predicted completion time follows a normal distribution, and the scheduler will schedule a job on a grid site with higher power and lower variability. The limitation of this approach is the assumption of the distribution of the period of job executing to be a normal distribution. Therefore, this assumption should be verified in an experiment. Furthermore, the behaviors of jobs executed at each grid site are not analyzed.

2. Yang and Schopf [16] introduce interval predication and interval variance predication as two more predictive measurements instead of one-step estimate. The grid site with lower interval variance is considered more "reliable", so a scheduler assigns less work to higher variance grid sites. However, it does not explore the reason why the higher variance it leads to.

3. Vazhdukai [15] uses a regression technique to predict data transfer in Grid systems. It uses the liner regression model, quasi-linear regression model, or polynomial regression model to predict the dependent variable.

There are two major scheduling algorithms [3, 8] based on access cost in OptorSim.

- Access cost scheduling algorithm
- Queue access cost scheduling algorithm

Access cost scheduling algorithm (AC) [3] schedules data-intensive jobs to the grid site with minimal access cost. Access cost is an estimated value, which is based on the network status, for obtaining all files required by this job at the scheduling time. This scheduling algorithm considers the importance of file distribution, whereas it neglects the influencing factor of the length of jobs in the waiting queue. The proposed scheduling algorithm is given based on this traditional scheduling algorithm. Compared with our proposed algorithm, AC supposes that the data distribution in Data Grid will be the same between the scheduling and run time. However, it will be different when replica replacements occur frequently at each grid site. ACPB regards the potential behaviors of jobs in the waiting queue as an important influencing factor to the access cost.

Queue access cost scheduling algorithm [3] schedules data-intensive jobs to the grid site, which has the minimal total estimated access cost of all the jobs in the waiting queue. For each job in the queue the access cost is calculated as same as the access cost algorithm (AC). It also neglects the importance of potential behaviors of jobs in the waiting queue.

All in all, these two scheduling algorithms above in OptorSim focus on how to calculate the traditional access cost. However, the situations of data distributions between the scheduling and running time will be different when replica replacements are often in each grid site. Therefore, it is necessary for us to seek the causation. This paper assumes that the replica replacement strategy and the length of jobs in waiting queue at the local grid site are two direct influencing factors on the variation of data distribution.

To test this hypothesis, which is that the replica replacement occurs frequently influences the access cost significantly, this paper proposes a scheduling algorithm based on access cost with potential behaviors of jobs. The experiment is performed in OptorSim, and the traditional access cost scheduling algorithm (AC) has been compared with our proposed algorithm.

The main contribution of our scheduling algorithm, compared to the previous work, is considering the potential behaviors of jobs in the waiting queue at each grid site as an important influencing factor to the access cost, and it help us to predict the real access cost with more accurate at scheduling time.

III. Scheduling Algorithm with Potential Behaviors

A. Scheduling Algorithm

To predict the data distribution of each grid site at run time, the potential behaviors of jobs in the waiting queue should be analyzed and inferred. In this section, the influencing factors on existing access cost approach is given firstly, then predictive access cost is defined secondly, and the proposed scheduling algorithm is depicted in detail finally.

1) Influencing factors

Existing access cost can be mainly evaluated by the processing time for replica creation when the target site is fixed or assumed. The processing time for replica creation includes:

- the time for replica set location
- the time for making replica selection decision
- the time for replica transferring

How to shorten the processing time for replica creation is the key. The relevant influencing factors are discussed in papers [12, 18-24] as below:

- *Network topology*: Network topology is always considered as the key factor to replica transferring.
- *Network bandwidth*: The replica in Data Grid is very huge, so the replica transferring is affected by the network bandwidth directly.
- *CPU load*: Replica creation will be much affected if the CPU load is heavy.
- *I/O bandwidth*: If the source node has a high I/O bandwidth, the time of replica transferring from this site is reduced greatly.
- *Access probability*: The grid sites in Data Grid have different access probability in the recent history. The grid site which has a high access probability has the high priority to be selected as the source node.
- *Job queue*: If jobs in the waiting queue are long at a grid site, replica transferring will be delayed for a long time. Therefore, selecting these grid sites which have a shorter length of waiting queues is better choice.

These six factors are classified into two categories: static factors and dynamic factors. Network topology, network bandwidth and I/O bandwidth are static factors; CPU load, access probability and job queue are dynamic factors. It is easy to get the values of static factors without any extra tools, but we need a feedback mechanism to monitor dynamic factors when required. We will give the feedback mechanism in the next subsection.

2) Predictive access cost

In order to get the predictive access cost, some relevant definitions are listed as bellow.

Define 1:

$$P(f_i) = \frac{F(f_i, T)}{\sum_{k=0}^{n} F(f_k, T)} \quad . \tag{1}$$

Where T refers to the duration to the present and $F(f_i,T)$ is the frequency of f_i within T.

Define 2:

$L(G_i)$ refers to the length of jobs in the waiting queue at the grid site G_i.

Define 3:

$T(R_i)$ refers to the time to live of the report R_i.

Define 4:

$P_{PA}(f_i)= P(f_i)\times L(G_i)/Maxlength.$ (2)

Where $P_{PA}(f_i)$ refers Probability of Potential Access of f_i.

Define 5:

$N(G_i, G_j)$ refers the bandwidth capacity between grid site G_i and grid site G_j.

Define 6:

$P_{ACF}(f_i)=Min(P(f_i))\times(Sizeof(f_i)/N(G_m,G_n)))$. (3)

Where $P_{ACF}(f_i)$ refers to the Predictive Access Cost of f_i.

Define 7:

$$P_{AC}(G_i) = \sum_{k=0}^{N} P_{ACF}(f_k) \cdot \tag{4}$$

Where N refers to the number of required files for the scheduling job.

Therefore, each candidate grid site as target node can be calculated which holds the minimal Predictive Access Cost will be chosen as the final target node.

3) Scheduling algorithm

From fig. 1, we can describe the scheduling workflow as follows:

a) The resource broker gets a data-intensive job from the waiting queue in the master grid site.

b) The required data set of this job is analyzed and grouped as a set.

c) The distribution status of replicas is gained based on the required files of this job, and the grid sites in this distribution will be grouped as a candidate set.

d) Iteratively calculating the access cost for each grid site in the candidate set based on latest reports and the status of data distribution.

e) Choosing the grid site with minimal access cost, and scheduling the data-intensive job to this site finally.

Therefore, the proposed approach is an application-centric scheduling strategy. Each job is scheduled to the grid site with predictive minimal access cost at run-time.

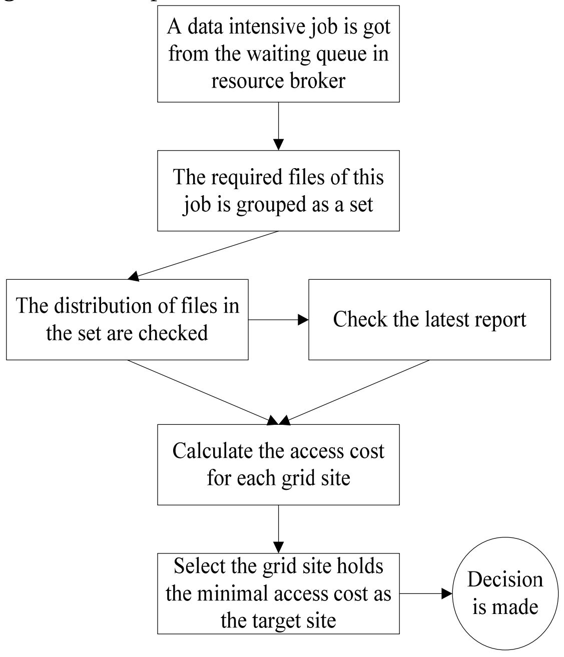
A data intensive job is got from the waiting queue in resource broker
The required files of this job is grouped as a set
The distribution of files in the set are checked
Check the latest report
Calculate the access cost for each grid site
Select the grid site holds the minimal access cost as the target site
Decision is made

Figure 1. Scheduling algorithm workflow.

B. *Replica Feedback Mechanism*

Replica feedback mechanism is a communication mechanism between grid sites and the resource broker in Data Grid, and also the infrastructure of ACPB. This section includes three subsections: report content, report operations and report workflow.

1) *Report content*

In order to predict the future access cost of a data-intensive job, four factors are included in report content. Declaration of replica replacement strategy and the access probabilities of replicas in local grid site are the first two factors. The length of jobs in waiting queue in local grid site is the third factor. Time to live of this report is the last factor on inference the future access cost of a data-intensive job. These four factors can be depicted in the following:

a) *Declaration of replica replacement strategy:* it indicates which replica replacement strategy the gird site uses, such as LFU, LRU etc.

b) *The recent access probabilities of replicas in local grid site:* this is the major content in the report content which indicates the future access probabilities of replicas in local grid site.

c) *The length of jobs in the waiting queue:* it tells that whether the local grid site is busy or not, and the probability of replica replacement occurring.

d) *Time to live of this report:* it is used to evaluate the period of validity of this report.

2) *Report Operations*

The replica feedback mechanism is to communicate between the resource broker and grid sites in Data Grid. A report queue is created and maintained in each grid site, and the latest report is accessed by the resource broker before making an informed scheduling decision.

The main operations for the report queue are summarized as follows:

a) addNewReport(report,ttl). A latest report is created when a replica replacement is occurred, and it will be inserted into the local report queue if allowed. The permission condition is that if all reports in the report queue were invalid, any new report is allowed to be added. If the report is added successfully, it returns true. Otherwise, it returns false with different reasons for failure.

b) readLatestReport(). The resource broker will access the report queue before making a scheduling decision. If no latest report, it returns null. Otherwise, it returns the latest report in the report queue.

c) displayReportQueue(). This method is applied to checking reports in the report queue.

3) *Report Workflow*

To introduce the replica feedback mechanism clearly, the report workflow is depicted in fig. 2.

From fig. 2, it shows that report is generated by the local grid site when replica replacement occurs, and the report will be inserted into its local report queue if a certain condition is satisfied. If not, the operation of adding a new report will fail. The local report queue is maintained by itself, and it is queried by resource broker to gain the latest report. The query operation will be performed before the job scheduling decision has been made.

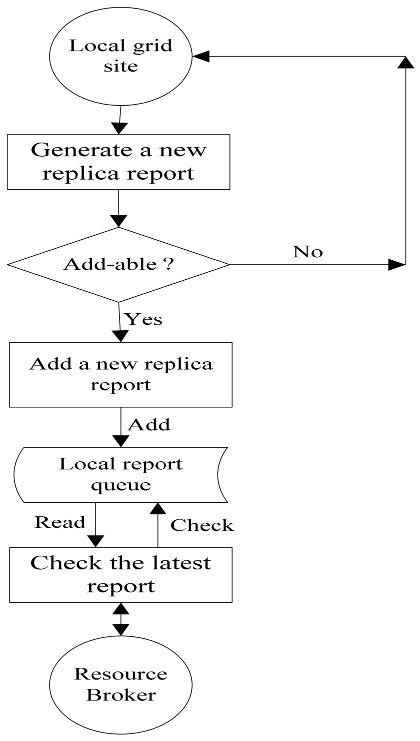

Figure 2. Report Workflow.

IV. Experiments and Discussion

A. *Experiment Configuration*

Experiment configuration in OptorSim includes parameter configuration, grid configuration and job configuration.

1) Parameter configuration

The basic simulation parameters are set up by means of parameter configuration file. There are three parts of parameters in this configuration file. The first part defines the grid topology and resources, the second part defines the jobs and their associated files, and the last part defines the relevant parameters and algorithms to use. The most important parameters include: the access pattern through which the jobs access files; the job submission pattern by which the users submit jobs to the resource broker; the level and variability of non-grid traffic present and the optimisation algorithms to use. A full description of each is in the OptorSim user guide [4]. In our experiment, the access pattern is sequential access, which means that the files are sequentially accessed. The interval of job delay is defined as the delay of resource broker submitting each job, and the value is 2500 ms. The optimization algorithm of replica replacement we adopted is LFU, which always deletes least frequently accessed file to replicate the required file. For experiment environment, the CPU is Intel core T2400 1.83 GHz and the capacity of memory is 1G.

2) Grid Configuration

The topology of CMS testbed is depicted in fig. 3.

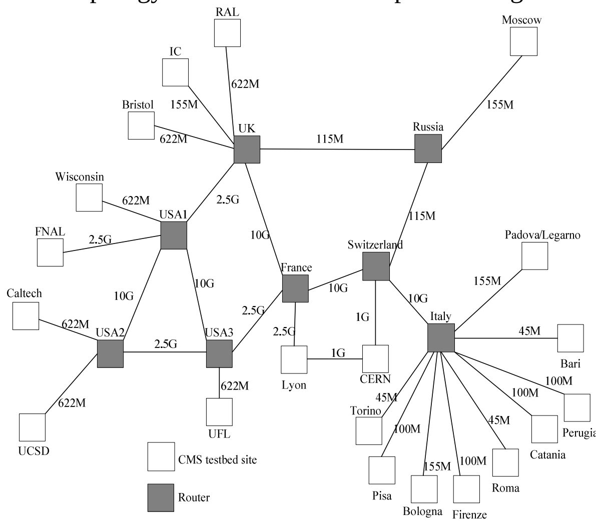

Figure 3. CMS Data Challenge 2002 grid topology.

From fig.3, CERN and FNAL are given SEs of 100 GB capacity and no CEs in the CMS testbed. SE refers to Storage Element, and CE refers to Computing Element. All master files are stored at one of these sites. Every other site is given 50 GB of storage and a CE with one worker node.

3) Job Configuration

Initially, all files were placed on the CERN Storage Element. There are six job types, with no overlap between the set of files each job accessed. Each set of files is assumed to have the capacity of 10GByte files.

From table 1, highptlepjob represents this kind of jobs with highest selection probability, least required files and minimal execution time, while highptphotjob represents that kind of jobs with lowest selection probability, the largest number of files and the maximal execution time in this paper. These six types of jobs represent major scenarios of applications with different selection probabilities and different numbers of required files in Data Grid. To make sure the experiment more valuable, we give two assumptions as bellow:

- The job with more files has lower selection probability. In reality, the number of jobs with fewer files is more than that with more files, and so is it in Data Grid.
- The execution time of the job which requires fewer data files is much shorter. This assumption is common sense.

Table 1. Selection Probability, the Max Number of Required Files and Execution Time for Each Job Type.

Job type	Selection probability	Number of files	Execution time (ms)
jpsijob	0.1	12	8
highptlepjob	0.5	2	2
incelecjob	0.15	5	4
incmuonjob	0.07	14	10
highptphotjob	0.03	58	25
zbbbarjob	0.15	6	4

B. *Experiment Comparison and Discussion*

1) Experiment Comparison

We have implemented ACPB algorithm and compared it with scheduling algorithm based on access cost (AC) in OptorSim. Mean job time, number of replications, and effective network usage are chosen as evaluation criteria and illustrated in detail as follows.

We give two different perspectives to the experiment of ACPB and AC.

- Different queue length with the same number of jobs.
- Different numbers of jobs with the same queue length.

a) Different queue length with the same number of jobs.

We assume there are 1000 batch-mode data-intensive jobs required to be processed in CMS testbed. ACPB and AC are processed with different maximum queue length of 25, 50, 75, 100, 125, 150 and 175. The simulation result is shown in the following figures.

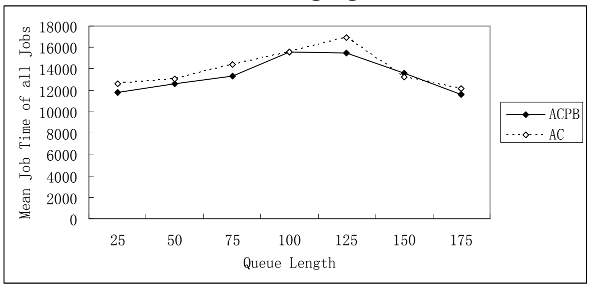

Figure 4. Mean job time of all jobs on grid vs. queue length.

From fig. 4, it shows that mean job time of all jobs on grid is increased with the increasing of queue length. ACPB spends less time than AC with the equal length of queue as illustrated.

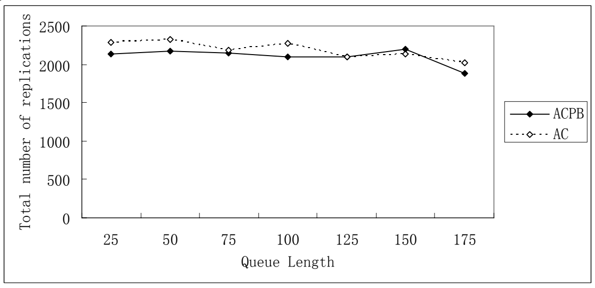

Figure 5. Total number of replications vs. queue length.

From fig. 5, in general, total number of replications for ACPB is much less than AC when the length of queue is less than 125. As the length of queue is increased, total number of replications is relatively reduced. However, there is an outlier when the queue length equals 175.

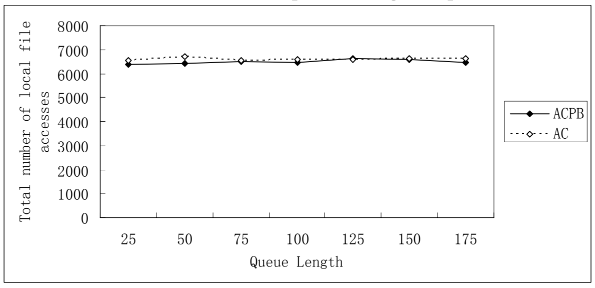

Figure 6. Total number of local file accesses vs. queue length.

From fig. 6, it depicts that total number of local file accesses of ACPB is less than AC except the situation when the queue length is 125.

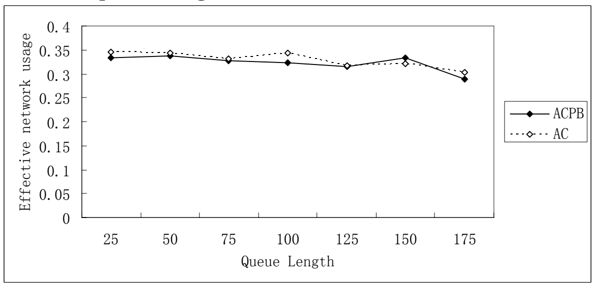

Figure 7. Effective network usage vs. queue length.

From fig. 7, it tells that effective network usage of ACPB is less than AC. As the length of queue is increased, effective network usage of ACPB is reduced. When the maximum queue length is 150, the effective network usage of ACPB is a little higher than AC.

Therefore, from fig. 4, 5, 6 and 7, ACPB has better performance than AC in evaluation criteria of mean job time of all jobs on grid, total number of replications, total number of local file accesses and effective network usage. However, ACPB does not have an advantage compared with AC when the length of queue is more than 125.

b) Different numbers of jobs with the same queue length.

From the first perspective, we know that ACPB has better performance when the queue length is less than 125 and it is not the same situation as the queue length is more than 125. To verify our conclusion, we do more experiments with the queue length of 50 and 150 separately. The results of these experiments are given in figures.

- Maximum queue length is 50.

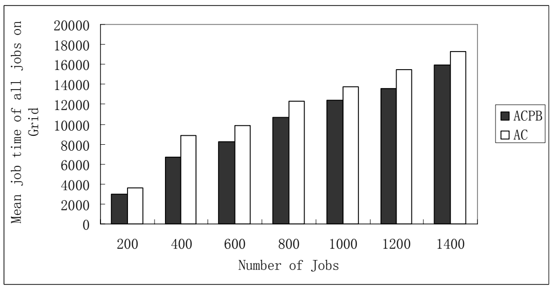

Figure 8. Mean job time for all jobs of ACPB vs. AC.

Fig. 8 indicates that the ACPB's mean job time for different number of jobs is less than AC's. As the number of jobs increases, the required mean job time for both ACPB and AC is increasing. However, the ACPB algorithm needs less mean job time, and the tendency is obvious when the number of jobs is more than 400.

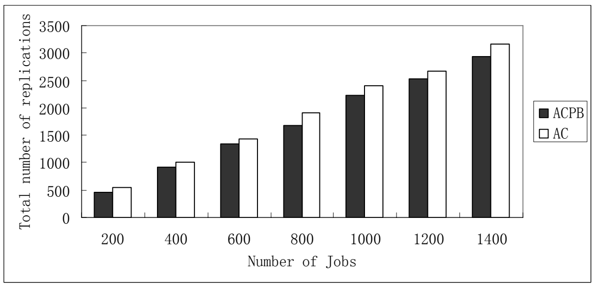

Figure 9. Total number of replications of ACPB vs. AC.

From fig. 9, total number of replications required for ACPB is less than AC. As the number of jobs increases, the required total number of replications for both algorithms is increasing. The AC algorithm requires more replications than ACPB when the number of jobs is same, no matter what the number of jobs is.

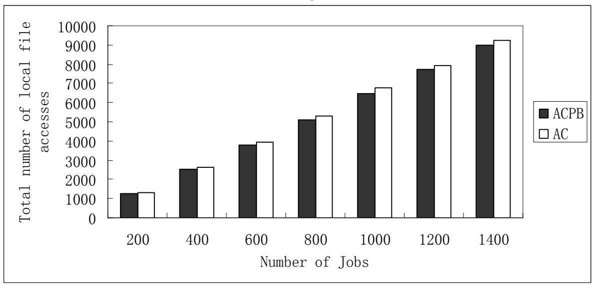

Figure 10. Total number of local file accesses of ACPB vs. AC.

From fig. 10, total number of local file accesses required for ACPB is less than AC. As the number of jobs increases, the required total number of local file accesses for both algorithms is increasing. The AC algorithm requires more local files accesses than ACPB when the number of jobs is equal.

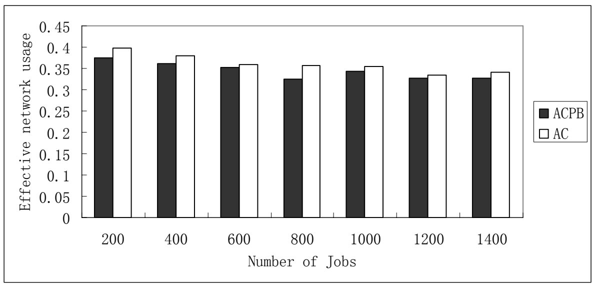

Figure 11. Effective network usage of ACPB vs. AC.

From fig. 11, effective network usage of ACPB is lower than AC. There is no significant difference for different amount of jobs; however, values of effective network usage of both algorithms have a decrease tendency when the number of jobs is increased. All in all, this phenomenon shows that effective network usage of Data Grid is not affected significantly by ACPB or AC.

From fig. 8,9,10 and 11, the proposed algorithm named as ACPB has better performance in mean job time, total number of replications, total number of local file accesses and effective network usage when the maximum queue length is 50 at each grid node. As we have discussed, the critical point of ACPB and AC is 125. To explore the characteristics of ACPB, we should consider another maximum queue length such as 150.

- Maximum queue length is 150.

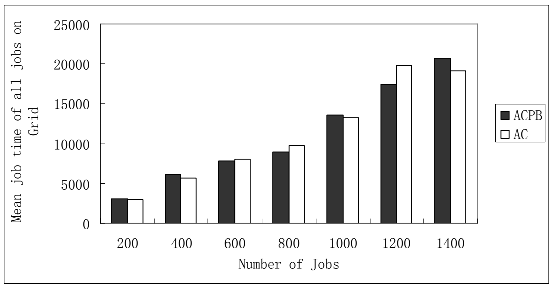

Figure 12. Mean job time for all jobs of ACPB vs. AC.

Fig. 12 indicates that the ACPB's mean job time for different number of jobs is less than AC's when the number of jobs is less than 1000. As the number of jobs increases, the required mean job time for both ACPB and AC is increasing. The mean job time of ACPB and that of AC are combined together. That is to say, there is no significant difference between ACPB and AC in mean job time when the number of jobs is more than or including 1000.

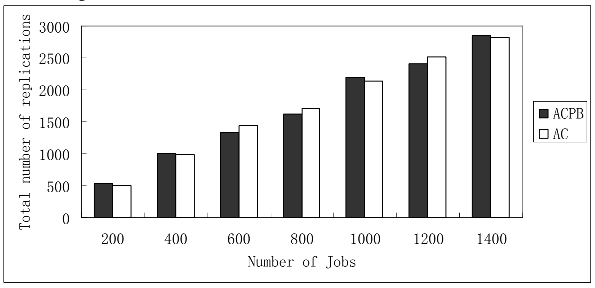

Figure 13. Total number of replications of ACPB vs. AC.

From fig. 13, the required total numbers of replications for both algorithms are increasing when the number of jobs increases. Total number of replications required for ACPB is less than AC when the number of jobs is less than 1000. However, total number of replications of ACPB and AC is combined together when the number of jobs is more than or including 1000.

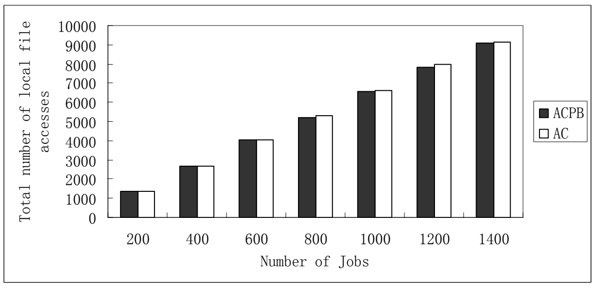

Figure 14. Total number of local file accesses of ACPB vs. AC.

From fig. 14, total number of local file accesses required for ACPB is less than AC. As the number of jobs increases, the required total number of local file accesses for both algorithms is increasing. The AC algorithm requires more local files accesses than ACPB when the number of jobs is same. However, the difference between ACPB and AC is not significant.

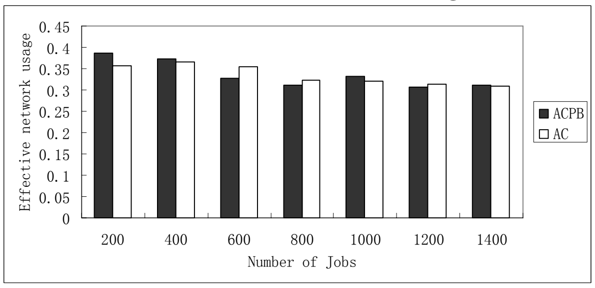

Figure 15. Effective network usage of ACPB vs. AC.

Fig. 15 indicates that there is no significant difference between ACPB and AC in effective network usage. The value of effective network usage decreases when the number of jobs is increased. It also tells that the Data Grid will become stable when the maximum number of jobs is more than 800.

From fig. 12,13,14 and 15, the proposed algorithm named as ACPB has better performance in mean job time, total number of replications, and total number of local file accesses when the number of jobs is less than 1000 and the maximum queue length is 150 at each grid site. There is no significant difference between ACPB and AC when the number of jobs is more than or including 1000.

2) Discussion

Data-driven job scheduling algorithms focus on how to schedule the job to its suitable processing grid site. From the experiments illustrated above, we compare the evaluation criteria between ACPB and AC for different queue lengths. As the hypothesis we have given before, the queue length is an important influencing factor on potential behaviors of jobs in waiting queue. The hypothesis is verified when the length of jobs in waiting queue is less than 125. That is to say, there is a significant difference between our proposed scheduling algorithm named as ACPB and the scheduling algorithm based on access cost called AC when the maximum queue length at each grid site is less than 125. However, the hypothesis is not accepted when the maximum queue length is more than 125, which means ACPB can be replaced by AC when the maximum queue length is more than 125.

Queue length of 125 is a critical point. We attach importance to the value which is not a constant. In other words, there may be some influencing factors on this value. For example, it may be changed when different jobs with different data files in different Data Grid environments.

The result of different experiments illustrated in this paper indicates that there is a significant difference between ACPB and AC when the maximum queue length is less than 125. Therefore, the queue length is an important influencing factor on potential behaviors of jobs in the waiting queue. In the algorithm given, queue length which is put into calculating access cost is considered as an important influencing factor. The experiment demonstrates that the proposed algorithm gets better performance since the queue length factor has a positive influence when it is less than 125. At the same time, this factor has no or little influence on the value of access cost. Therefore, the proposed algorithm can be improved when the queue length is increasing.

The decentralized feedback mechanism for replica report is given to support decision making. This mechanism is very simple, effective and easy to be implemented. Based on this feedback mechanism, the latest report has its probability to be into the report queue if there is no valid report in it. In other words, the latest report will not be inserted into the report queue in time when the time to live of reports in the report queue is too long or replica replacement operations are frequently performed in a short period. Therefore, the time to live of reports should be dynamically given based on the frequency of replica replacement operations.

V. Conclusions and Future Work

The scheduling algorithm for data-intensive jobs is very important for Data Grid applications. Existing scheduling algorithms based on access cost are effective and efficient approaches. However, the influencing factors on access cost should be reconsidered, since existing algorithms neglect the importance of potential behaviors of jobs in the waiting queue. These potential behaviors can be predicted roughly based on the queue length of waiting jobs and the local replica replacement strategy. To verify this hypothesis that the potential behaviors of jobs in the waiting queue have an effect on the evaluation of access cost, we have proposed a scheduling algorithm based on access cost with potential behaviors and given a decentralized feedback mechanism to replica report.

From these preceding experiments and comparisons, we can see that the proposed algorithm named as ACPB has better performance than AC when the maximum queue length at each grid site is less than 125 in the Data Grid with same number of jobs required to be processed. However, there is no significant difference when the maximum queue length is more than 125. Based on experiments in OptorSim, ACPB shows that it has better performance in mean job time of total jobs on grid, total number of replications, total number of local file accesses and effective network usage than the scheduling algorithm based on access cost (AC) when the waiting queue's maximum length is less than 125 for each grid site.

Based on the discussion section, we have explained the reason why the experiment result is. For the future, we will explore the evaluation criteria of access cost and the relationship between the waiting queue length and access cost. Furthermore, we should redesign the decentralized feedback mechanism to collect the latest replica report quickly and to support resource broker to make decisions efficiently.

References

[1] A. Chervenak, I. Foster, C. Kesselman et al., "The Data Grid: towards an architecture for the distributed management and analysis of large scientific data sets," *Journal of Network and Computer Applications*, vol. 23, pp. 187-200, July 2001.

[2] W. Hoschek, F. J. Jaen-Martinez, A. Samar, H. Stockinger, and K. Stockinger, "Data management in an international data grid project," *Lecture Notes In Computer Science*, Vol.1971, pp.77-90, 2000.

[3] D. G. Cameron, R. Carvajal-Schiaffino, P. Millar, C. Nicholson, K. Stockinger, and F. Zini, "Evaluating scheduling and replica optimisation strategies in OptorSim," in proc. of Grid, Phoenix, Arizona, USA, November 2003, pp. 52-59.

[4] M. J. Lewis and A. Grimshaw, "The core legion object model," in proc. of 5th IEEE international Symposium on High Performance Distributed Computing, Syracuse, New York, USA, August 1996, pp.551-561.

[5] F. Berman, R. Wolski, and H. Casanova, "Adaptive computing on the grid using AppLeS," *IEEE Transactions on Parallel and Distributed Systems*, vol.14 (4), pp.369-382, April 2003.

[6] R. S. Chang, J. S. Chang, and S. Y. Lin, "Job scheduling and data replication on data grids," *Future Generation Computer Systems*, vol.23 (7), pp.846-860, August 2007.

[7] M. Tang, B. S. Lee, X. Y. Tang, and C. K. Yeo, "The impact of data replication on job scheduling performance in the Data Grid," *Future Generation Computer Systems*, vol.22 (3), pp.254-268, February 2006.

[8] D. G. Cameron, A. P. Millar, and C. Nicholson, "Optorsim: a simulation tool for scheduling and replica optimisation in Data Grids," in proc. of CHEP, Interlaken, Switzerland, September 2004.

[9] M. Litzkow, M. Livny, and M. Mutka, "Condor-a hunter of idle workstation," in proc. of 8th International Conference of Distributed Computing Systems, San Jose, California, USA, June 1988, pp.204-111.

[10] J. Frey, T. Tannenbaum, et al, "Condor-G: a computation management agent for multi-Institutional grids," *Cluster Computing*, vol.5, pp. 237-246, July 2002.

[11] I. Foster and C. Kesselman. "Globus: a metacomputing infrastructure toolkit," *International Journal of Supercomputer Applications*, vol.11 (2), pp.115-128, April 1997.

[12] R. Buyya, D. Abramson, and J. Giddy, "Nimrod/G: an architecture of a resource management and scheduling system in a global computational grid," in proc. of HPC ASIA, Beijing, China, May 2000, pp.283-289.
[13] R. Wolski, N. T. Spring, and J. Hayes, "The network weather service: a distributed resource performance forecasting service for metacomputing," *Journal of Future Generation Computing Systems*, vol.15, pp.757-768, October 1999.
[14] G. Karypis, E. H. Han and V. Kumar, "Chameleon: hierarchical clustering using dynamic modeling," *Computer*, vol.32(8), pp.68-75, August 1999.
[15] S. Vazhkudai, and J. M. Schopf, "Using regression techniques to predict large data transfers," *The International Journal of High Performance Computing Applications*, vol.17 (3), pp. 249-268, August 2003.
[16] L. Y. Yang, J. M. Schopf, and I. Foster, "Conservative scheduling: using predicted variance to improve scheduling decisions in dynamic environments," in proc. of SC, Phoenix, USA, November 2003, pp.31-31.
[17] J. M. Schopf, and F. Berman, "Stochastic scheduling," in proc. of SC, Portland, Oregon, USA, November 1999, pp.48-48.
[18] X. H. Wei, W. W. Li, O. Tatebe, G. C. Xu, L. Hu, and J. b. Ju, "Implementing data aware scheduling in Gfarm using LSF scheduler plugin mechanism," in proc. of GCA, Las Vegas, Nevada, USA, 2005, pp.3-10.
[19] R. Kavitha and I. Foster, "Design and evaluation of dynamic replication strategies for a high performance data grid," in proc. of CHEP, Beijing, China, September 2001, pp.106-118.
[20] Y. Zhao and Y Hu, "GRESS - a grid replica selection service," in proc. of PDCS, Marina Del Rey, Canada, November 2003, pp.423-429.
[21] J. H. Jiang, G. C. Xu and X. H. Wei, "An enhanced data-aware scheduling algorithm for batch-mode data-intensive jobs on data grid," in proc. of ICHIT, Cheju Island, Korea, November 2006, pp.257-262.
[22] R. M. Rahman, K. Barker and R. Alhajj, "Replica selection in grid environment: a data-mining approach," in proc. of ACM SAC, Santa Fe, New Mexico, March 2005, pp.695-700.
[23] R. S. Chang, J. S. Chang and P. S. Lin, "An ant algorithm for balanced job scheduling in grids," *Future Generation Computer Systems*, vol.25 (1), pp.20-27, January 2009.
[24] S. Venugopal, and R. Buyya, "An SCP-based heuristic approach for scheduling distributed data-intensive applications on global grids," *Journal of Parallel and Distributed Computing*, vol.68 (4), pp.471-487, April 2008.
[25] K. Ranganathan and I. Foster, "Identifying dynamic replication strategies for a high performance data grid," *Lecture Notes In Computer Science*, Vol.2242, pp.75-86, 2001.
[26] L. Huy, P. Coddington, A. L. Wendelborn, et al., "A data-aware resource broker for data grids," in proc. of NPC, (Lecture Notes in Computer Science Vol. 3222), Wuhan, China, 2004, pp.73-82.
[27] E. Elmroth and J. Tordsson, "Grid resource brokering algorithms enabling advance reservations and resource selection based on performance predictions," *Future Generation Computer Systems*, vol.24 (6), pp.585-593, June 2008.

Jianhua Jiang received his B.E degree from Jilin University, China in 2003 and M.S degree from Jilin University in 2006. Currently, he is working towards the Ph.D. degree in the College of Computer Science and Technology at Jilin University. His research focus is data grid, data mining, business intelligence, OLAP and computer education. He has served as a technical reviewer for several international conferences and has published 8 papers in both technical and educational fields.

Huifang Ji received her B.E degree from Jilin University, China in 2008. Currently, she is working towards the M.S. degree in the College of Computer Science and Technology at Jilin University. Her research focus is job scheduling algorithms and replica management in data grid and data mining.

Gaochao Xu received his Ph.D. degree from Jilin University, Changchun, China in 1995, in computer architecture. His research focus is distributed computing, grid computing, data mining, and network security and computer education. He has served as a technical reviewer for several IEEE international conferences and has published more than 30 papers in both technical and educational fields.

Xiaohui Wei received his Ph.D. degree from Jilin University, Changchun, China in 1999, in computer architecture. His research focus is distributed computing, grid computing, and web intelligence and computer education. He has served as a technical reviewer for several IEEE international conferences and has published more than 20 papers in both technical and educational fields.