Grid Workflow Scheduling based on Reliability Cost

Yongcai Tao, Hai Jin, Xuanhua Shi

Cluster and Grid Computing Lab
Services Computing Technology and System Lab
Huazhong University of Science and Technology, Wuhan, 430074, China
hjin@hust.edu.cn

ABSTRACT

Grid workflow is a complex and typical grid application, but owing to the highly dynamic feature of grid environments, resource unavailability is increasingly becoming severe and poses great challenges to grid workflow scheduling. Though fault recovery mechanism adopted in grid system guarantee the completion of jobs to some extent, but wasting system resources. To overcome the shortcoming, this paper proposes a Markov Chain based grid node availability prediction model which can efficiently predict grid nodes' availability in the future without adding significant overhead. Based on this model, the paper presents a grid workflow scheduling based on reliability cost (RCGS). The performance evaluation results demonstrate that RCGS improves the dependability of workflow execution and success ratio of tasks with low reliability cost.

Categories and Subject Descriptors

C2.4 [**Distributed Systems**]

General Terms: Design, Experimentation, Standardization

Keywords: Grid computing, Workflow, Reliability cost, Markov chain.

1. INTRODUCTION

As a novel and promising technology, grid offers us a new vision, infrastructure and trend for the coordinated resource sharing, problem-solving and services orchestration in dynamic, multi-institutional *Virtual Organizations* (VOs) by integrating various types of resources over Internet [1, 2]. Grid enables users to share all resources (computational resources, storage resources, communication resources, software resources, information resources, knowledge resources, etc.) and cooperate with others on the Internet. The advent of grid brings a bright future for scientific research, commerce and industry. Users, ranging from scientific researchers, workers, businessman to general customer, are utilizing grids to share, manage and process large data sets.

Grid workflow is a complex and typical grid application, distinguishing from the traditional workflow in: (1) efficiently utilizing wide-area distributed resources; (2) cooperating among heterogeneous organizations; (3) solving computing-intensive and data-intensive tasks; and (4) making workflow more like service flow to exploit OGSA idea. Grid workflow provides an approach to complex and collaborative scientific researches, like high-energy physics, geophysics, astronomy and bioinformatics.

Infoscale'07, June 6–8, 2007, Suzhou, China.

Currently, there are a number of tools supporting the description of workflow applications, such as: Petri Nets [22], UML (*Unified Modeling Language*) [23] and user-defined component. Graph-based modeling is more preferred by users compared with language-based modeling. So, grid workflows are typically represented by means of a *directed acyclic graph* (DAG) [24]. Each node in the graph denotes an executable task. Each directed edge denotes a precedence constraint between two tasks. The sink node cannot start execution until the source node has finished and the required amount of data from the source node has been transferred to the sink node.

Grid workflow can be seen as a collection of tasks that are processed on distributed resources in a well-defined order to accomplish a specific goal. The workflow scheduling problem is to allocate tasks onto resource nodes in such a way that precedence constraints are respected and the overall execution time is minimized, so it is a NP-complete problem, and many heuristics are proposed to obtain the optimal scheduling, such as min-min heuristic, max-min heuristic, sufferage heuristic [20]. However, due to the diverse failures and error conditions in grid environments, the unavailability of resource nodes is increasingly becoming severe and poses great challenges to grid workflow scheduling. For example, grid resources are mostly non-dedicated and can enter and depart without any prior notice. In addition, the change of resource local policy, the breakdown of software and hardware and the malfunction of network fabric can result in resource inaccessibility. Hence, jobs fail frequently and QoS can not be guaranteed.

To address these issues, existing grid systems generally resort to fault recovery mechanism [2], such as checkpoint, retry and replication. Although relieving the challenges to some extent, this mechanism sacrifices system resources. For example, checkpoint policy requires extra disk space and network bandwidth to record the job running information, e.g. intermediate results and data for continuing job without starting from scratch. Retry strategy re-schedules job at the same resource or other resource when job fails. Replication policy runs the job at multiple available resources. The fault recovery mechanism belongs to compensating methodology and can not prevent job failures in advance. To prevent the job failures proactively, the accurate information of temporal and spatial distribution of grid node availability in the future should be predicted. Thus, jobs can be scheduled onto resource node with long uptime instead of

upcoming failing nodes. So far, researchers mainly give attention to model and predict the characteristics of cluster nodes' availability. With the rapid expanding of gird scale, grid integrates not only cluster resources, but also more wide-area personal resources which are mostly non-dedicated. Therefore, the characteristics of node unavailability in grid environment distinguish greatly from that in cluster environment.

In this paper, we analyze the reliability of workflow execution in grid environment and propose a Markov Chain based grid node availability prediction model, which utilizes idle CPU cycles to predict node availability in the future without adding significant computational overhead. Furthermore, based on this model, the paper presents a novel grid workflow scheduling based on reliability cost (RCGS). Rational of RCGS is that it computes the reliability of node during task's running time and then makes scheduling decision based on the reliability cost of task. Finally, performance evaluation is conducted to compare RCGS with other scheduling algorithms, and performance evaluation results demonstrate that RCGS improves the dependability of grid workflow execution and the success ratio of tasks execution, and accordingly decreases the makespan of workflow execution.

The rest of the paper is organized as follows. Section 2 reviews the related work. Section 3 introduces the reliability analysis of DAG. Grid node availability prediction model is designed in section 4. Then, the grid workflow scheduling based on reliability cost is proposed in section 5. The performance evaluation is conducted in Section 6. Finally, we conclude and give some future work about our research in Section 7.

2. RELATED WORK

There is a vast amount of research work on grid workflow scheduling. Scheduling models usually adopt DAG. Scheduling strategies can be categorized into performance-driven, market-driven and trust-driven [3].

Performance-driven strategy tries to submit jobs onto resources and achieve optimal performance for users and system (e.g. minimum job execution time and high throughput) [4, 5, 20]. Most heuristics for DAG scheduling on heterogeneous systems belong to the performance-driven strategy [26, 27, 28]. The work in [6, 7, 8] utilizes market-driven strategy during the job assignment. In the systems, bids are collected from eligible resource providers for each task. If the execution time satisfies user's requirement, a bid with lower price will be chosen as the optimal bid. Recently, some research works have strived to address the scheduling problems with reliability optimization [9, 10, 11, 12]. In CCOF project [9] and GridSec project [10, 11], trust-driven scheduling strategy is adopted to map jobs onto appropriate resources according to their trust levels. This strategy avoids selecting malicious nodes and non-reputable resources, and intends to increase the system reliability. However, it does not take the job completion prediction time into consideration. In [12], reliable scheduling adopts "Reliability Cost" as the indicator which considers how reliable a given system is when a group of tasks are assigned to it, but the failure rate of grid nodes and the communication links between them adopted in the reliability cost model is set experimentally.

To support reliable job scheduling, more attention is paid to model the characteristics of node failure so as to measure the node reliability, but which mainly focuses on homogeneous computing environments [13, 14, 15]. The work in [13, 14, 15] investigates the error logs of cluster systems and concludes that the time between reboots of nodes is best modeled by a Weibull distribution with shape parameters of less than 1, implying that a node becomes more reliable the longer it has been operating. Based on the observed characteristic of node failure, researchers also design resources allocation strategy to improve service availability. However, due to the highly dynamic feature of grid environment, grid node failures are more stochastic than clusters. Hence, current models can not adapt well to grid environment.

Recently, much efforts study the resource availability prediction in grid environment. In [16], machine availability in enterprise systems is analyzed, but the results are only meaningful for the considered application domain. The authors of [17] develop a multi-state availability model based on semi-Markov to predict future resource availability. However, the applications of FGCS in [17] are confined to be CPU-bound batch programs, which are sequential or comprise multiple tasks with little or no inter-task communication. So, FGCS is not suitable for grid workflow consisting of multiple tasks with much precedence and data dependence.

3. RELIABILITY ANALYSIS OF DAG

Directed acyclic graph (DAG) is an efficient model to represent grid workflow. A DAG $G=<V, E, W>$ is a node-weighted and edge-weighted directed graph, where $V=<u_1, u_2,\ldots, u_n>$ is the set of nodes, with each node denoting a task, $E \subseteq V \times V$ is the weighted edge set that defines the precedence relationship among nodes in V. The weight on each edge, $w_{ij} \in W$, denotes the volume of data being transmitted from node u_i to node u_j. $P=\{P_1, P_2,\ldots, P_M\}$ represents the resources of a grid system. For each task $u_i \in V$, $T(u_i)$ represents the execution time on each resource node: $T(u_i)=\{t_1(i), t_2(i),\ldots, t_M(i)\}$, where $t_j(i)$ represents the execution time of u_i on P_j.

System reliability is defined as the probability that system will not fail during the running time of tasks. Considering a grid system with M resource nodes, $P=\{P_1, P_2,\ldots, P_M\}$, and a DAG containing n nodes, $V=<u_1, u_2,\ldots, u_n>$. Let x_{ij} be a binary number that denotes whether task u_i is assigned to P_j or not. Let s_{ij} be the probability of resource node P_j not to fail during the running time of task u_i on P_j. So, the probability of the system not to fail is:

$$\Pr = \prod_{j=1}^{M} \prod_{i=1}^{N} \left(s_{ij}^{\,x_{ij}\left(t_j(i) + \sum_{p \in Prec(i)} \sum_{k=1}^{M} (w_{pi} x_{pk} D_{kj}) + SL_j\right)} \right) \quad (1)$$

where $\sum_{p \in Prec(i)} \sum_{k=1}^{M} (w_{pi} x_{pk} D_{kj}) + SL_j$ denotes the execution latency of task u_i, the time that task u_i spends to fetch the needed data from its preceded nodes before execution. SL_j represents the scheduling length of resource node P_j. When x_{ij} is small,

$$\Pr \approx \prod_{j=1}^{M} \prod_{i=1}^{N} \left(e^{-(1-s_{ij}) x_{ij} \left(t_j(i) + \sum_{p \in Prec(i)} \sum_{k=1}^{M} (w_{pi} x_{pk} D_{kj}) + SL_j\right)} \right) \quad (2)$$

In order to maximize Pr, we need to minimize:

$$\sum_{j=1}^{M}\sum_{i=1}^{N}(1-s_{ij})x_{ij}(t_j(i)+\sum_{p\in Prec(i)}\sum_{k=1}^{M}(w_{pi}x_{pk}D_{kj})+SL_j) \quad (3)$$

According to the above analysis, we define the reliability cost R_{ij} of successful execution of task u_i on resource node P_j as follow:

$$R_{ij}=(1-s_{ij})(t_j(i)+\sum_{p\in Prec(i)}\sum_{k=1}^{M}(w_{pi}x_{pk}D_{kj})+SL_j) \quad (4)$$

Correspondingly, *RC* is defined to be the reliability cost of successful execution of workflow DAG *V* on grid system *P* as follow:

$$RC=\sum_{i=1}^{N}\sum_{j=1}^{M}x_{ij}R_{ij} \quad (5)$$

Thus, to improve the reliability of workflow execution, we need to minimize RC. The lower RC is, the higher the reliability.

4. GRID NODE AVAILABILITY PREDICTION

In the section, a *Markov Chain* (MC) based grid node availability prediction model is designed, and we adopt *Time To Failure* (TTF) of nodes as the metric of node availability.

4.1 Markov Model

Markov model is usually utilized to model stochastic processes in many application fields. *Discrete-time Markov chain* (DTMC) is a process that consists of a finite number of states $M(S_1, S_2,\ldots, S_m)$ and $M\times M$ known state transition matrix *P*. In matrix *P*, P_{ij} is the probability of moving from state S_i to state S_j [18].

DTMC can be used to predict the state occurrence probability in the future. Suppose at time *k*, system state is S_i ($1\le i\le M$) and the distribution of S_i is $P_k(S_i)=e_i$, where e_i is $1\times M$ row vector, the value at location *i* is 1, and others is 0. Thus, we can predict the distribution of S_i at time *k*+1 as:

$$P_{k+1}(S_i)=P_k(S_i)P=e_iP \quad (6)$$

At time *k*+2, the distribution of S_i is:

$$P_{k+2}(S_i)=P_{k+1}(S_i)P=e_iP^2 \quad (7)$$

At time *k*+*n*, the distribution of S_i is:

$$P_{k+n}(S_i)=P_{k+n-1}(S_i)P=e_iP^n \quad (8)$$

So, with DTMC, we can obtain the distribution of state S_i at this time and next time, and therefore we can get the occurrence probability of each state at each time.

4.2 MC Based Node Availability Prediction Model

Resource nodes and network are volatile and failures can occur at any time in grid environment. As a result, TTF of nodes at each time is stochastic. So Markov model can be used to model the stochastic process of nodes' TTF. In the prediction model, TTF is the system state: $M(S_1, S_2,\ldots, S_m)$. *P* is $M\times M$ state transition matrix.

In the Markov model described above, the state set *M* and state transition matrix *P* are invariable. The dynamic nature of grid requires large storage space for *M* and *P*, which makes the model complex and unpractical for grid system. In order to address this issue, we present an adaptive MC based grid node TTF prediction model which can dynamically amend *M* and *P*.

When a resource node fails, a new TTF is produced (called TTF_{new}). Then *M* would be traversed. If there exists state S_i whose absolute difference value and TTF_{new} is less than the specified value, S_i would be modified to be the average of S_i and TTF_{new} and the number of state transition would be increased by 1. If there does not exist this state, new state S_{m+1} would be created. At the same time, *P* would be emended correspondingly as follow:

$$P_{ij}=\frac{n_{ij}}{\sum_{K}n_{ik}} \quad (9)$$

where n_{ij} represents the transition number from state *i* to state *j* at *K* failures. $\sum_K n_{ik}$ denotes the all state transition number of *K* failures. The pseudo-code of state *M* and matrix *P* amendment algorithm is illustrated in Algorithm 1 below:

Algorithm 1: State *M* and matrix *P* amendment algorithm
Input: New TTF at time *k*, *M*, *P*
Output: New *M* and new *P*
Step 1: Amending state space *M*
  While *i* <= size of *M* do
    If$|S_i-TTF_{new}|$ <=specified value then
      $S_i \leftarrow \frac{S_i+TTF_{new}}{2}$;
      n_{ki} ++;
      break;
    Endif
    insert a new state S_{m+1} into *M*;
    $n_{k(i+1)}=1$;
    dimension of *P* increases by 1;
  Endwhile
Step 2: Amending matrix *P*
  For *i*=1 to the row size of *P* do
    N=0;
    For *j*=1 to the column size of *P* do
      $N=N+n_{ij}$;
    Endfor
    For *j*=1 to the column size of *P* do
      $P_{ij} \leftarrow \frac{n_{ij}}{N}$;
    Endfor
  Endfor

5. RELIABLE SCHEDULING OF GRID WORKFLOW

Scheduling DAGs in grid environment mainly exploits heuristics based on list scheduling and grouping scheduling. In list scheduling, a weight is assigned to each node and edge of DAG and these weights are used to prioritize the nodes. Then, task nodes in DAG are subsequently assigned to grid resource in this order. Whereas grouping scheduling groups the tasks according to

the weights of nodes and edges, and tasks in same group are independent of each other and can be scheduled independently. Based on the listing and grouping scheduling, we presents a novel workflow scheduling based on reliability cost (RCGS).

5.1 Grid Workflow Scheduling based on Reliability Cost (RCGS)

RCGS consists of three phases: ranking, grouping, and scheduling independent tasks within each group. First, a weight is assigned to each node and edge of DAG. This is based on averaging all possible values of the cost of node (or edge) on each resource (or combination of resources). With this weight, upward ranking is computed and each node of DAG is assigned a rank value. The rank value of node i, $rank_i$, is recursively defined as follow:

$$rank_i = W_i + \max_{\forall \in Succ_i} (W_{ij} + rank_j) \quad (10)$$

where W_i is the weight of node i, $Succ_i$ is the set of immediate successors of node i and W_{ij} is the weight of the edge connecting nodes i and j.

Second, nodes of DAG are sorted in descending order by their rank value. With this order, they are divided into different groups as follows. The first node (e.g., the node with the highest rank value) is added to a group numbered 0. If successive nodes in descending order by their rank value are independent with all the nodes already assigned to the group (namely, there is no dependence between them in the DAG), they are placed in the same group. Reversely, if there is dependence, a new group will be created and the new group's number is the current group's number increased by one, and then the node with the smallest rank value is the member of the new group. Again, subsequent task nodes will be assigned to different groups. The final outcome is a set of ordered groups.

Third, according to the ascending order of groups' number, the independent tasks within each group are scheduled using different heuristics, such as max-min heuristic, min-min heuristic. In RCGS, the independent tasks in each group are scheduled with consideration of the reliability of resource nodes and tasks' execution time. Namely, the task would be tried to be scheduled to the resource node on which task has the lowest reliability cost while satisfying QoS requirements (e.g. completion time), which enhancing the reliability of task execution.

The algorithm of RCGS is shown in Algorithm 2. Although RCGS may seem similar to the hybrid heuristic in [19], there is a fundamental difference. Hybrid heuristic aims to minimize the makespan of DAG during scheduling independent tasks, whereas, RCGS targets to improve the probability of successful execution of DAG, and correspondingly shortens the makespan.

Algorithm 2: RCGS algorithm
Input: DAG, P (set of grid resource)
Output: Scheduling set of tasks/resource nodes.
Step 1: Assign a weight to each node and edge of DAG
For each task i in DAG do

$$W_i = \frac{\sum_{j=1}^{M} t_j(i)}{M};$$

Endfor
For each edge in DAG do

$$W_{ij} = \frac{\sum_{i=f}^{M}\sum_{j=l}^{M}(k_{ij} \times w_{ij})}{2 \times C_M^2};$$

Endfor
Step 2: Computing the rank value of nodes of DAG
For each task i in DAG do

$$rank_i = W_i + \max_{\forall \in Succ_i} (W_{ij} + rank_j);$$

$R[\,] \leftarrow$ sorting ranki in descending order;
Endfor
Step 3: Grouping the nodes of DAG
$G_0 = \{\}$; $i = 0$; $k = 0$;
While $R[k] \neq \varnothing$ do
If $\exists$ dependence between $R[k]$ and $\forall\ u_j$ ($u_j \in G_i$) then
i++;
creating $G_i = \{\ \}$;
Endif
$G_i = \{\ \} \leftarrow R[k]$;
k++;
Endwhile
Step 4: Scheduling independent tasks in each group
scheduling tasks in ascending order;

5.2 Scheduling Independent Tasks in Each Group

Existing heuristics are developed at the assumption that resources are dedicated and no fails occur, not considering the reliability of resource nodes. In this sub-section, based on the availability prediction of grid resource nodes, a reliable scheduling algorithm is proposed.

The reliable scheduling algorithm consists of two steps. At the first step, independent tasks are sorted in descending order by their average completion time on all resource nodes. The completion time of task i includes the execution time and the transfer time for needed data as follow:

$$MFT_i = W_i + \sum_{k \in Parents(i)} W_{ki} \quad (11)$$

At the second step, according to the descending order of the average completion time of tasks, the task with highest average completion time is chosen. The completion time of task i on resource node j is computed as follow:

$$FT_{ij} = t_j(i) + \sum_{p \in \Pr ec(i)} \sum_{k=1}^{M} (w_{pi} x_{pk} D_{kj}) \quad (12)$$

Then, using the prediction model in Section 4, the reliability of resource node j during the running time of task i is predicted as follows. Assume that the state of resource node j is S_q at time t_q, we can predict its state distribution at time t_q+1: (P_{q1} P_{q2} … P_{qM}). S_k ($1 \le k \le m$) denotes the TTF of resource node j. In order to guarantee the reliable execution of task i on resource node j, the following condition should be satisfied, $S_k - (T_{now} - ST_j) > FT_{ij}$. Here, T_{now} denotes the current time and ST_{ij} represents the startup time

of node *j*. So, the reliability of successful execution of task *i* on resource node *j* can be obtained as follow:

$$r_{ij} = \frac{\sum_{k=x}^{M} S_k P_{qk}}{\sum_{k=1}^{M} S_k P_{qk}}$$

$$S_k - (T_{now} - ST_j) > FT_{ij} \quad (13)$$

Accordingly, its reliability cost can be computed:

$$R_{ij} = (1 - r_{ij}) \times (FT_{ij} + SL_{ij}) \quad (14)$$

The specific scheduling process is shown in Algorithm 3.

Algorithm 3: Scheduling independent tasks
Input: A set of tasks, a set of resource nodes.
Output: Scheduling set of tasks/resource nodes.
Step 1: Sorting task nodes
For each task i do

$$MFT_i = W_i + \sum_{k \in Parents(i)} W_{ki};$$

Endfor
$T[]$ ← task nodes in descending order by their average predicted completion time on all resource nodes;
Step 2: Scheduling tasks
For task $T[i]$ do
For each resource node *j* do

$$FT_{ij} = t_j(i) + \sum_{p \in \mathrm{Prec}(i)} \sum_{k=1}^{M} (w_{pi} x_{pk} D_{kj});$$

$$r_{ij} = \frac{\sum_{k=x}^{M} S_k P_{qk}}{\sum_{k=1}^{M} S_k P_{qk}}; \quad // S_k - (T_{now} - ST_j) > FT_{ij}$$

$R_{ij} = (1 - r_{ij}) \times (FT_{ij} + SL_{ij})$;
Endfor
$T[i]$ ← resource node *j* with lowest R_{ij} value
Endfor

6. PERFORMANCE EVALUATION

6.1 The Settings

In order to evaluate the performance of RCGS for scheduling DAGs proposed in the paper, we compare it with HEFT [25] and Hybrid min-min heuristic in terms of reliability cost, makespan of DAG and success ratio of tasks [20]. The experiments are carried out in a real grid environment. The testbed includes two sites, one is at Cluster and Grid Computing Lab (CGCL) at Wuhan, China, and the other is at National Hydro Electric Energy Simulation Laboratory (NHEESL) at Wuhan, China. There is a 16 nodes cluster linked by 100Mbps switched Ethernet in CGCL, each node is equipped with Pentium III processor at 1GHz and 512MB memory, and the operating system is Red Hat Linux 9.0. There is a 12 nodes cluster linked by 100Mbps switched Ethernet in NHEESL, each is composed of IA 64 processor at 1.3GHz and 2GB memory, and the operating system is Red Hat Linux 9.0. The testbed is composed of four nodes in CGCL and four nodes in NHEESL. The prototype of RCGS is implemented with JAVA. The grid platforms are deployed with *ChinaGrid Support Platform* (CGSP) [21]. We collect the running log of these six grid platforms for three months and use them to create MC based node availability prediction model to predict grid nodes' availability in the future. The two scheduling algorithms are as follows:

- **HEFT:** Heterogeneous Earliest-Finish-Time [25]. HEFT covers two major phases: a task prioritizing phase for computing the priorities of all tasks and a resource selection phase for selecting the tasks in the order of their priorities and scheduling each selected task on its "best" resource, which minimizes the task's finish time. It selects the task with the highest upward rank at each step. The selected task is then assigned to the resource which minimizes its earliest finish time with an insertion-based approach. The upward rank of a task is the length of the critical path (i.e., the longest path) from the task to an exit task, including the computation cost of the task.
- **Min-min heuristic:** Min-min heuristic has two scheduling steps. In the first step, for each job, the resource having the minimum predicted completion time is found. In the second step, the job having the minimum predicted completion time value is chosen to be scheduled. This is done iteratively until all the jobs have been mapped. Min-min heuristic is widely adopted in grid job scheduling system.

Performance is evaluated using two different types of DAGs: random (generated as explained in [26]), Laplace commonly used in other similar studies (see [27, 28]). In each case, we randomly generate 10 DAGs, each DAG consists of from 10 to 100 tasks, and they are scheduled to different heterogeneous resources mentioned above. For each task in the DAGs, the estimated execution time on each different machine is randomly generated from a uniform distribution in the interval of 50 to 100 time units, while the *communication-to-computation ratio* (CCR) is also randomly chosen from the interval 0.1 to 1.

6.2 Performance Evaluation Results

Figure 1 and Figure 2 show the reliability cost of three scheduling algorithms. The reliability cost of DAG is computed according to Formula 5. From the results, RCGS outperforms HEFT and Min-min heuristic. This is mainly because that both HEFT and Min-min heuristic aim to select the resource node on which task has lowest finish time, whereas, RCGS tries to schedule tasks to the resource node with lowest reliability cost while satisfying tasks' QoS requirements (e.g. completion time, cost).

Figure 3 and Figure 4 show the comparison of makespan of DAG between RCGS and HEFT and Min-min heuristic. RCGS performs better and HEFT is poor, especially while the number of tasks of DAG increases. In HEFT, the rank value of task *i* is defined to be the longest path from task *i* to exit task. HEFT does not consider the topology of resource node and transfer load between resource nodes, so it can not dynamically reflect the change of DAG during scheduling. In RCGS, although the rank value of task is also defined statically at the beginning, but it considers the communication cost between resource nodes. Min-min heuristic is similar to RCGS. However, like HEFT, it also

does not consider the reliability of resource nodes while scheduling independent tasks. So, when the number of tasks of DAG increases, tasks would fail frequently while adopting HEFT and Min-min heuristic scheduling and re-scheduling them would result in the longer makespan of DAG.

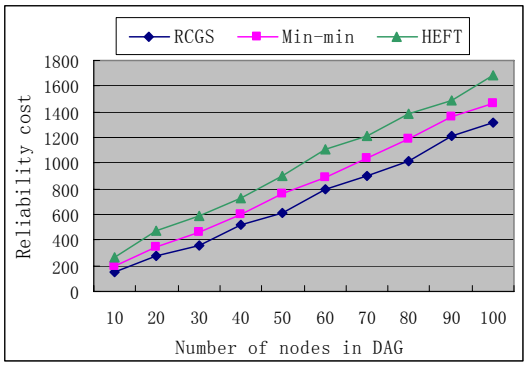

Figure 1. Reliability cost of three scheduling algorithms on random DAGs

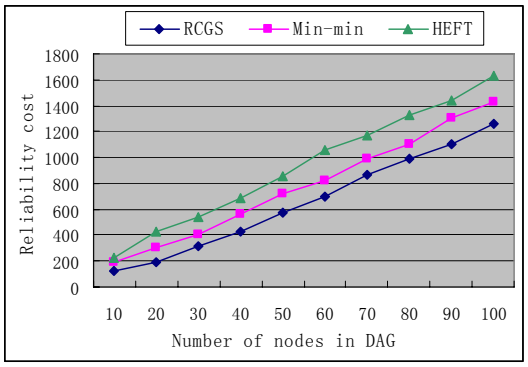

Figure 2. Reliability cost of three scheduling algorithms on Laplace DAGs

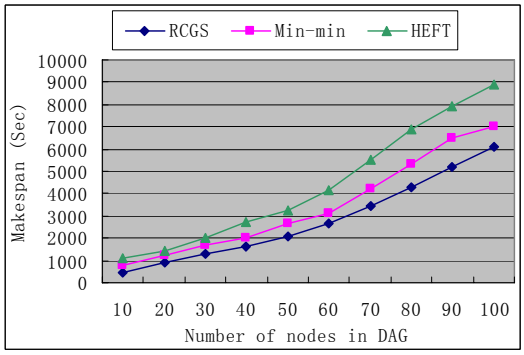

Figure 3. Makespan of DAG of three scheduling algorithms on random DAGs

Figure 5 and Figure 6 show the success ratio of task execution of DAG by adopting RCGS, HEFT and Min-min heuristic algorithms. As shown in Figures 5 and 6, there is little difference in success ratio of jobs of three algorithms when the number of jobs of DAGs is low. However, with the increasing of the number of jobs of DAGs, RCGS improves the success ratio of task about 10 percent higher than HEFT and Min-min heuristic algorithms. This phenomenon can be explained by the fact that RCGS, exploiting reliability cost as the main scheduling metric, tries to schedule tasks to the reliable resource nodes. HEFT and Min-min heuristic do not take into account resource nodes' reliability while scheduling so that the resources, to which tasks are assigned, are prone to fail and the failure ratio of tasks is high.

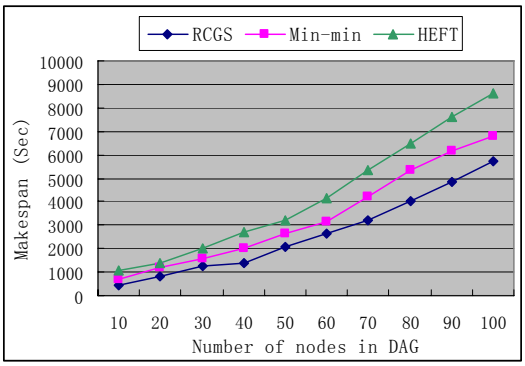

Figure 4. Makespan of DAG of three scheduling algorithms on Laplace DAGs

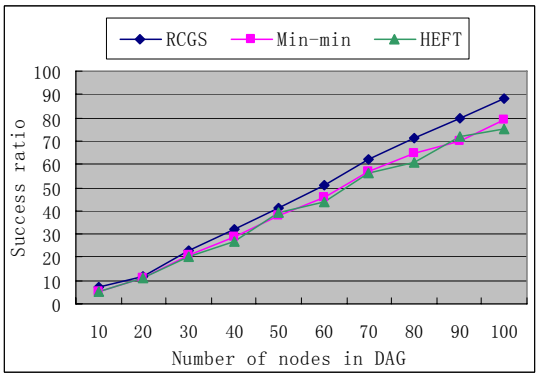

Figure 5. Success ratio of task execution of three scheduling algorithms on random DAGs

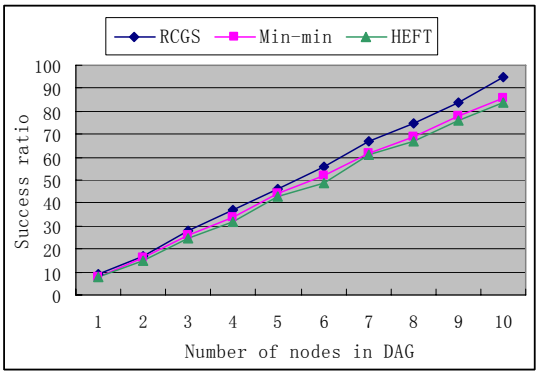

Figure 6. Success ratio of task execution of three scheduling algorithms on Laplace DAGs

7. CONCLUSION AND FUTURE WORK

The resources in grid environment are generally provided voluntarily and their availability fluctuates highly. Due to unexpected resource unavailability, grid jobs frequently fail. In

this paper, the reliability cost of DAG is analyzed. A MC based grid node availability prediction model is designed. Based on this model, a grid workflow scheduling based on reliability cost is presented, which considers both the reliability of resource nodes and tasks' completion time, providing fault-avoidance capability for grid workflow. Performance evaluation results prove that RCGS lowers the reliability cost and makespan of DAG, and improves the success ratio of tasks.

As our future work, we plan to perfect the node availability prediction model and further study reliable scheduling for grid workflow.

8. ACKNOWLEDGEMENTS

This paper is supported by Nation Science Foundation of China under grant No.90412010 and 60603058.

9. REFERENCES

[1] Foster, I. and Kesselman, C. The Grid: Blueprint for a New Computing Infrastructure. 2nd edition, Morgan Kaufmann, November 2003.

[2] Hwang, S. and Kesselman, C. Grid Workflow: A Flexible Failure Handling Framework for the Grid. In Proc. of twelfth IEEE International Symposium on High Performance Distributed Computing (HPDC-12), IEEE Computer Society Press, Los Alamitos, CA, USA, June 22-24, 2003, Seattle, Washington, USA, pp.126-137.

[3] Krauter, K., Buyya, R., and Maheswaran, M. A Taxonomy and Survey of Grid Resource Management Systems for Distributed Computing. Software Practice and Experience, 32(2):135-164, February 2002.

[4] Cooper, K., Dasgupata, A., and Kennedy, K., eds. New Grid Scheduling and Rescheduling Methods in the GrADS Project. NSF Next Generation Software Workshop, International Parallel and Distributed Processing Symposium, Santa Fe, IEEE CS Press, Los Alamitos, CA, USA, April 2004.

[5] Cao, J., Jarvis, S. A., Saini, S., and Nudd, G. R. GridFlow: Workflow Management for Grid Computing. In Proc. of 3rd International Symposium on Cluster Computing and the Grid (CCGrid), Tokyo, Japan, IEEE Computer Society Press, Los Alamitos, May 12-15, 2003.

[6] Buyya, R., Murshed, M., Abramson, D., and Venugopal, S. Scheduling Parameter Sweep Applications on Global Grids: A Deadline and Budget Constrained Cost-Time Optimisation Algorithm. Software: Practice and Experience (SPE) Journal, 35(5):491.512, April 2005.

[7] Sandholm, T. and Lai, K. Market-Based Resource Allocation using Price Prediction in a High Performance Computing Grid for Scientific Applications. In Proc. of Fifteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-15), IEEE Computer Society, June 19-23, 2006, Paris, France, pp.132-143.

[8] Venugopal, S., Buyya, R., and Winton, L. A Grid Service Broker for Scheduling Distributed Data-Oriented Applications on Global Grids. In Proc. of 2nd International Workshop on Middleware for Grid Computing (Middleware 2004), Toronto, Ontario, Canada, ACM Press, New York, NY, USA, October 18, 2004.

[9] Zhao, S. Y. and Lo, V. Result Verification and Trust-based Scheduling in Open Peer-to-Peer Cycle Sharing Systems. Technical Report, University of Oregon, USA, 2005.

[10] Song, S. S. and Hwang, K. Security Binding for Trusted Job Outsourcing in Open Computational Grids. IEEE Transactions on Parallel and Distributed Systems (TPDS), submitted May 2004, revised Dec. 2004.

[11] Song, S. S., Kwok, Y. K., and Hwang, K. Trusted Job Scheduling in Open Computational Grids: Security-Driven Heuristics and A Fast Genetic Algorithm. In Proc. of 19th IEEE International Parallel & Distributed Processing Symposium (IPDPS'05), Denver, CO, USA., IEEE Computer Society Press, Los Alamitos, CA, USA, April 4-8, 2005.

[12] He, Y., Shao, Z., Xiao, B., Zhuge, Q., and Sha, E. Reliability Driven Task Scheduling for Heterogeneous Systems. In Proc. of The Fifteenth IASTED International Conference on Parallel and Distributed Computing and Systems, Vol.1, 465-470, 11/2003.

[13] Sahoo, R., Sivasubramaniam, A., Squillante, M., and Zhang, Y. Failure data analysis of a large-scale heterogeneous server environment. In Proc. of International Conference on Dependable Systems and Networks (DSN), Florence, Italy, 2004.

[14] Heath, T., Martin, R., and Nguyen, T. D. Improving cluster availability using workstation validation. In Proc. of ACM SIGMETRICS 2002, Marina Del Rey, CA, 2002.

[15] Sahoo, R., Oliner, A. J., Rish, I., Gupta, M., Moreira, J. E., and Ma, S. Critical event prediction for proactive management in large-scale computing clusters. In Proc. of the ACM SIGKDD, pp.426-435, August 2003.

[16] Brevik, J., Nurmi, D., and Wolski, R. Automatic methods for predicting machine availability in desktop grid and peer-to-peer systems. In Proc. of CCGrid'04, pp.190-199, 2004.

[17] Ren, X. J., Lee, S., Eigenmann, R., and Bagchi, S. Resource Failure Prediction in Fine-Grained Cycle Sharing Systems. In Proc. of Fifteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-15), IEEE Computer Society, June 19-23, 2006, Paris, France, pp.93-104.

[18] Gilks, W. R., Richardson S., and Spiegelhalter, D. J. A. Introducing Markov chain Monte Carlo. pp.1-19, In Markov Chain Monte Carlo in Practice, Chapman & Hall, London.

[19] Sakellariou, R. and Zhao, H. A hybrid heuristic for dag scheduling on heterogeneous systems. In Proc. of 13th Heterogeneous Computing Workshop (HCW-2004), Santa Fe, New Mexico, USA, 2004.

[20] Mandal, A., Kennedy, K., Koelbel, C., Marin, G., Mellor-Crummey, J., Liu, B., and Johnsson, L. Scheduling strategies for mapping application workflows onto the grid. In Proc. of fourteenth IEEE International Symposium on High Performance Distributed Computing (HPDC-14), IEEE Computer Society, July 24-27 2005, Research Triangle Park, North Carolina, USA, pp.125-134.

[21] http://www.chinagrid.edu.cn/CGSP/

[22] Petri, C. A. Kommunikation mit Automaten. PhD Thesis, Institut für instrumentelle Mathematik, Bonn, 1962

[23] Object Management Group, Unified Modeling Language (UML), http://www.uml.org/

[24] Altintas, I., Birnbaum, A., Baldridge, K., Sudholt, W., Miller, M., Amoreira, C., Potier, Y., and Ludaescher, B. A. Framework for the Design and Reuse of Grid Workflows. In Proc. of International Workshop on Scientific Applications on Grid Computing (SAG'04), LNCS 3458, Springer, 2005.

[25] Topcuoglu, H., Hariri, S., and Wu, M. Performance-effective and low-complexity task scheduling for heterogeneous computing. IEEE Transactions on Parallel and Distributed Systems, 13(3):260–274, March 2002.

[26] Zhao, H. and Sakellarion, R. An experimental investigation into the rank function of the heterogeneous earliest finish time scheduling algorithm. In Proc. of Euro-Par 2003. Springer-Verlag, LNCS 2790, 2003.

[27] Sakellariou, R. and Zhao, H. A low-cost rescheduling policy for efficient mapping of workflows on grid systems. Scientific Programming, 12(4), December 2004, pp.253-262.

[28] Sakellariou, R. and Zhao, H. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems. In Proceedings of 13th Heterogeneous Computing Workshop (HCW 2004), 26-30 April 2004, Santa Fe, New Mexico, USA.