A Distributed Workflow Management System Model and its Scheduling Algorithm

Ding Changsong[1,2], Hu Zhoujun[1], Hu Zhigang[1], Li Xi[1]
[1]School of Information Science and Engineering, Central South University, Changsha 410083, China
[2]College of Physics Science and Information Engineering, Ji Shou University, Jishou 416000, China
E-mail:dinghongzhe@yeah.net

Abstract

Workflow management system model and workflow task scheduling algorithm are the critical research areas in combination of workflow and grid techniques. On the basis of active network, a distributed, scalable workflow management system model which can adapt to dynamic changes of grid resource is presented. Workflow tasks are partitioned according to the characteristic of workflow structure, and the entire deadline and cost are divided into sub-deadline and sub-cost for each task partition based on its length and communication traffic; Considering user's multi-QoS requirements and communication time between workflow tasks, a trust and cost based Comprehensive Beneficial Function and determination methods of trust's and cost's weights are provided. A scheduling algorithm TCD aims at comprehensive optimization of trust and cost is put forward and this algorithm tries to get an optimized solution for each task partition to achieve an optimized scheduling with multi-object for the entire workflow. Simulation results show that the algorithm can guarantee user's multi-QoS quite well.

1. Introduction

Grid workflow is an automation of a Grid process, in the whole or part, during which documents, information or data are passed from one Grid service to another for action, according to a set of procedural rules. Introducing workflow technique into Grid offers several advantages, such as (1) ability to build dynamic applications which orchestrate distributed resources, (2) utilizing resources that are located in a particular domain to increase throughput or reduce execution cost, (3) execution span multiple admini- strative domains to obtain specific processing capabi- lities, and (4) integration of multiple teams involved in management of different parts of the experiment workflow – thus promoting inter-organizational collaborations.

Current research on grid workflow mainly focuses on workflow management system model, its scheduling algorithm and so on. [3] presents a workflow management system architecture in which the control flow is centralized, lacking distributedness and parallelism. In [4] the performance of such centralized execution engines has been compared with two models of distributed execution engines. The simulations show that the decentralized models have much better performance. Distributed management system architecture—WORM is presented in [5], but in WORM the user's QoS requirements is not considered and workflow instance can't be executed in parallel. An algorithm is presented in [6] with the purpose of minimum cost while meeting the requirement of deadline, but it only considers task execution time and execution cost regardless of considering communication time and cost between tasks when calculating time and cost. [7] has discussed the importance of trust QoS to grid task scheduling and its concrete content in detail, but only one user QoS requirements is concerned during task scheduling. A multi-QoS optimal algorithm based on market model is put forward in [8], but the weight of each QoS requirement cannot be determined.

On the basis of active network, this paper investigates and builds a workflow management system model which is able to search grid service based on user's QoS requirements and adjust workflow execution process according to dynamic grid resource. A workflow task scheduling algorithm is also presented based on the algorithm which takes time and cost as its object in [6]. The algorithm presented in this paper considers two important factors—communication time between workflow tasks and trust relationship between gird nodes that provide grid services. This algorithm firstly partitions work- flow tasks and uses a deadline and cost assignment strategy to divide the entire workflow deadline and cost into sub-deadline and sub-cost of each task partition, and then schedules every

Supported by the National Natural Science Foundation of China under Grant No. 60673165 and 60433020.

978-0-7695-3540-1/08 $25.00 © 2008 IEEE
DOI 10.1109/FCST.2008.14

task partition according to its object function. Thereby the result of optimal solution for each task partition leads to an optimized scheduling with multi-object for the entire workflow. Due to the dynamic characteristic of Gird and non-accuracy of performance prediction, the schedule may fail, thus the failed tasks need to be rescheduled. In addition, workflow instance can be executed distributedly or in parallel by utilizing programmmability, scalability and distributedness of active network. As a result, the system overhead can be reduced and network workload can be balanced.

The reminder of this paper is organized as follows. Section 2 introduces the workflow management system model and the function of the main modules. Section 3 discusses multiple QoS-based workflow task scheduling algorithm in detail. The performance estimation and analysis will be discussed in section 4. Finally, section 5 concludes the paper.

2. Distributed workflow management system model

From the point of view of lifecycle, Grid workflow can be divided into three phases. (1)Workflow definition, namely a process of generating abstract workflow; (2)Workflow instance generation, namely a process of mapping workflow tasks to the Grid services; (3) Workflow execution, namely a process of executing workflow instance on Grid resource automatically according to the business flow. See Figure 1 for distributed workflow management system model. The functions of the main modules are showed as follows.

User Interface: Grid user submits task and its requirements through UI, the UI generates abstract workflow according to the task that submitted by the user and parses user's QoS parameters.

Grid Service Directory: It records information of available services in Grid, such as service type, capacity, position and so on. QoS-based service discovery is supported and the historical data related to task execution is provided for performance prediction. This service can be implemented by extending Globus MDS.

Workflow Optimizing: This module optimizes the abstract workflow according to the available service discovered. For example, it needs to adjust the workflow because some activities of the workflow can't be executed in parallel. So, the entire workflow generation process is dynamic.

Performance Prediction: In order to reserve resource in advance and divide deadline, we need to predict execution time of workflow tasks on available service. Define task executing time equals the ratio of task length to service capacity, task length is denoted by

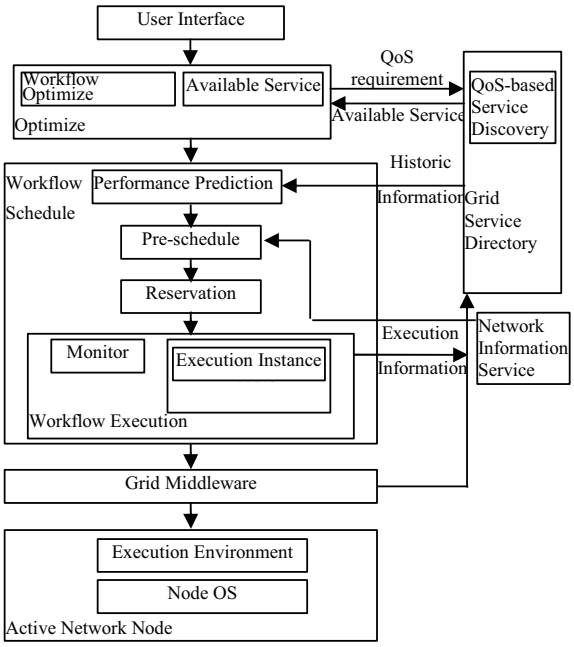


Figure 1. Workflow management system model

number of million instructions (MI), and service capacity is denoted by number of million instructions per second.

Pre-schedule: Based on available services, current minimum communication routing which can be found according to the method discussed in next section by Network Information Service which monitors the status of network, and the optimal matching of resource and workflow task is found according to the Comprehensive Beneficial Function.

Workflow Execution Instance: Workflow execution instance is a piece of code which defines workflow execution process, the code is the data package transferring on active network and can be executed automatically on active network nodes. Grid scheduler can use some tool to translate abstract workflow described by a description language such as OWL-WS [12] into this execution code.

Active Network Node: The node can not only store and transfer but also execute date package, it provides an execution environment for workflow instance code. The operating system such as Janos [13] of the node manages access requirements put forward by active network execution environment. And when there is a resource added to Grid, it only needs to join in or rebuild an active network node, so the model is scalable.

3. Muti-QoS based workflow task scheduling algorithm

For QoS-based workflow task scheduling problem, deadline and cost are two QoS parameters considered commonly, however, two shortages still exits. On one hand, workflow is a set of tasks, and it is an automatic process of executing ordered tasks. The communication between these tasks especially data-intensive tasks is a vital factor that affects workflow execution time and cost. On the other hand, trust relationship between Grid nodes is also an important factor affecting stability of task execution. In fact, uncertainty of service requester or service provider caused by untrusty will reduce quality of service. Therefore, trust degree between Grid nodes is another important guarantee for providing high quality service for user. With careful consideration of time and cost, this paper proposes a trust, cost and deadline based scheduling algorithm(TCD). Besides, the algorithm lays strong emphasis on communication time and trust degree and can determine the weight of each QoS objective.

(1)Communication time. We assume that task T_i is executed on service S_i^p, task T_j is executed on service S_j^q, T_i is predecessor of T_j. So there exits a shortest communication path between Grid node which provides service S_i^p and Grid node which provides service S_j^q. We also assume that the weight of edges in Grid nodes linked graph(see Figure 2) denotes communication time or communication cost which can be calculated according to the information provided by Network Information Service, so the shortest communication time or the lowest communication cost between these nodes can be found in the light of Dijkstra algorithm.

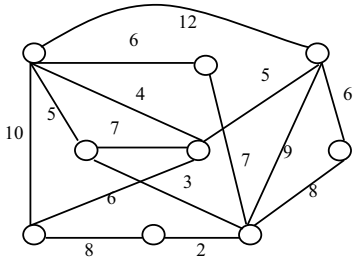


Figure 2. Grid node linked graph

(2)Trust degree. Using trust relationship between people in sociology for reference, the trust relationship in Grid is based on point-to-point and can be passed. The trust degree of node i to node j is denoted by $trust(i,j)$. $trust(i,j)$ is defined as a value which equals the ratio of times of node j which executes tasks submitted by node i successfully in a recent period to times of submission from node i to node j.

3.1. Problem formulation

Workflow task scheduling problem can be expressed as: given a process which contains a number of activities, some data should be transferred between these activities; given some Grid nodes, each node contains some kinds of resource and an amount of the resource, and physical bandwidth and delay exit in one node or between nodes; given each activity some execution option, choose a best one to reach some optimal objective.

Workflow activities are often described by Directed Acyclic Graph (DAG). We call the task whose In-Degree is zero as entry task, denoted as T_{entry}; and the task whose Out-Degree is zero as exit task, denoted as T_{exit}. Workflow scheduling problem can be denoted as a four-tuple $\Omega(\Gamma,\Lambda,D,C)$. In which, let element in set Γ denote task $T_i, 1 \le i \le n$ in workflow, namely $\Gamma = \{T_i, 1 \le i \le n\}$, corresponding to node set of DAG. Let element in set Λ denote execution sequence of task (T_i, T_j), corresponding to directed edges set of DAG and T_i is the predecessor of T_j. Let D denote workflow deadline specified by the user. Let C denote workflow cost specified by the user.

Let m be the number of available services. There are a set of services S_i^j $(1 \le i \le n, 1 \le j \le m_i, m_i \le m)$ which are capable of executing task T_i, but only one service should be selected for executing the task.

The basic steps of workflow scheduling process can be described as follows.

(1) Discover available service based on user's QoS requirements and predict execution time of each task;

(2) Partition workflow task according to the structure of DAG;

(3) Divide workflow deadline and cost into each task partition according to the task length and the communication traffic;

(4) For each task partition, query available time slot, make an optimized pre-scheduling according to the objective function and then make a reservation;

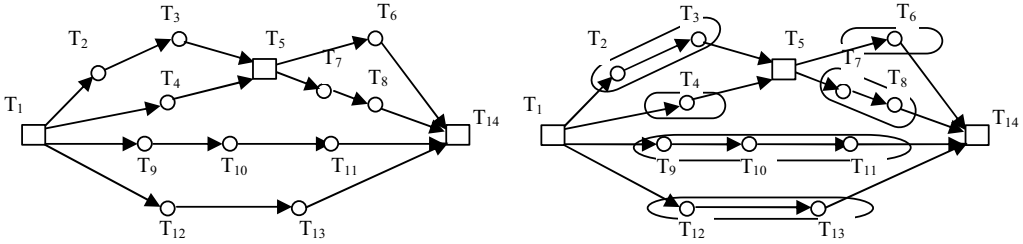
(5) Execute workflow instance, reschedule task when the initial scheduling fail.

3.2. Workflow task partition

Using methodology of task partition in [6] for reference, we categorize workflow task to be either a synchronization task or a simple task. A synchronization task is defined as a task whose in-degree or out-degree is more than one, such as T_1, T_5, T_{14} in Figure 3. Other tasks whose in-degree and out-degree equals one

are simple tasks. Let a branch be a set of interdependent simple tasks that are executed sequentially between two synchronization tasks, such as $\{T_2,T_3\}$, $\{T_4\}$, $\{T_6\}$, $\{T_7,T_8\}$, $\{T_9,T_{10},T_{11}\}$, $\{T_{12},T_{13}\}$ in Figure 3.

Workflow task partition is also described by DAG and denoted as $G(V,E,D,C)$. Thereinto, task partition $V_i \in V(1 \le i \le k+h)$, k and h denotes amount of synchronization tasks and number of branches respectively.



(a) before partition (b) after partition

Figure 3. Workflow task partition

A task partition V_i includes the following attributes: the closed cost $\cos t[V_i]$, denotes the cost that execution cost of the task partition can't exceed; the most expensive cost $cst[V_i]$, denotes the cost while scheduling the most expensive service to execute the task partition, which equals the sum of the most expensive cost of each task in the task partition; cost $ct[V_i]$, denotes actual cost of the task partition based on performance prediction; start time $st[V_i]$, denotes the time from which the task partition can start; deadline $dl[V_i]$, we can obtain the equation: $st[V_i] = \max(dl[P_i])$, P_i is a set of predecessor task of V_i; execution time $et[V_i]$, denotes actual execution time of the task partition based on performance prediction; communication time $t_{comm(V_i,V_j)}$, denotes communication time between task partition V_i and V_j (V_i is the predecessor of V_j), it equals communication time between the last task $T_{(i,last)}$ of task partition V_i and the first task $T_{(j,1)}$ of task partition V_j; communica- tion time $t_{comm(V_i)}$, denotes sum of communication time between tasks in task partition V_i; task length $QT(V_i)$, denotes sum of task length of each task in task partition V_i, namely $QT(V_i) = \sum QT(T_i)$ (task $T_i \in V_i$); communication traffic $QC(V_i,V_j)$, denotes communication traffic between task partition V_i and V_j (V_i is the predecessor of V_j), it equals communication traffic between the last task $T_{(i,last)}$ of task partition V_i and the first task $T_{(j,1)}$ of task partition V_j; communi- cation traffic $QC(V_i)$, denotes the sum of communication traffic of each task in task partition V_i, namely $QC(V_i) = \sum QC(T_i)$ (task $T_i \in V_i$), if task partition V_i is a synchronization task, then $QC(V_i) = 0$.

3.3. Divide deadline and cost

The entire workflow deadline is divided into sub-deadline of each task partition, and then each task has a sub-deadline. Deadline contains following properties: (1) A synchronization task can be executed only after finishing all of its predecessor tasks. (2) Between two synchronization tasks there are no less than two branches, in order to avoid waiting execution of other branches, any branch can select a service with lower cost while meeting the requirement of sub-deadline, thus the execution cost can be reduced. For example, see Figure 3, task partition $\{T_2,T_3\}$ and $\{T_4\}$ have the same deadline, the task partition with shorter execution time(suppose $\{T_4\}$) can choose a slower service in performance in order to reduce cost, lest waiting execution of task partition $\{T_2,T_3\}$.(3) If every task partition can be finished before its sub-deadline, then the workflow can meet the requirement of the user's deadline.(4) The divided sub-deadline should not less than the sum of the minimum execution time and the minimum communication time.

According to the properties above, the detailed approach of dividing deadline is as follows.

(1) Depth first then breadth first search Graph G, find the longest path and mark the node

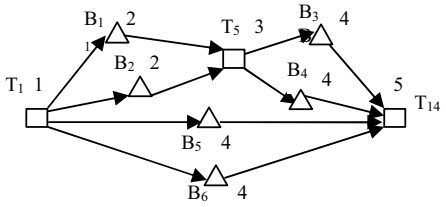


Figure 4.Workflow task partition graph

s in the longest path from 1 to N. For example, the longest path in Figure 4 is $\{T_1, B_1, T_5, B_3, T_{14}\}$ or $\{T_1, B_2, T_5, B_4, T_{14}\}$.

(2) Mark other paths. Depth first search Graph G backwards, mark unmarked nodes started with N reversely. The sub-deadline of the task partitions are the same if their marks are the same. So finishing dividing deadline in the longest path, each sub-deadlines for each partition is obtained.

(3) When dividing deadline, the communication time between two task partitions is assigned to the successor task partition. The dividing proportion of deadline is calculated according to the sum of communication traffic between task partitions, task partition length and communication traffic between tasks in the task partition. In the longest path, for multi-branches between two synchronization tasks, choose the larger sum of communication traffic and task partition length. By calculating task length and communication traffic of each task partition in the longest path, we can obtain the dividing proportion:

$$QT_1 : QC(V_1,V_2) + QT(V_2) + QC(V_2) : \cdots : QC(V_i,V_j) + QT(V_j) + QC(V_j) : \\ : \cdots : QC(V_{n-1},V_n) + QT(V_n) + QC(V_n) \quad (1)$$

So, the deadline of the entry task is:

$$dl[V_1] = \frac{QT_1}{QT_1 + \sum_{i=2}^{N-1} QC(V_i,V_j) + QT(V_j) + QC(V_j)} D \quad (2)$$

The deadlines of other tasks are:

$$dl[V_j] = \frac{QC(V_i,V_j) + QT(V_j) + QC(V_j)}{QT_1 + \sum_{i=2}^{N-1} QC(V_i,V_j) + QT(V_j) + QC(V_j)} D \quad (3)$$

For dividing workflow cost, we obtain the dividing proportion by calculating task length and communication traffic in all task partition. Then the closed costs of each task partition are as follows:

$$\cos t[V_1] = \frac{QT_1}{QT_1 + \sum_{i=2}^{N-1} QC(V_i,V_j) + QT(V_j) + QC(V_j)} C \quad (4)$$

$$\cos t[V_j] = \frac{QC(V_i,V_j) + QT(V_j) + QC(V_j)}{QT_1 + \sum_{i=2}^{N-1} QC(V_i,V_j) + QT(V_j) + QC(V_j)} C \quad (5)$$

3.4. Pre-schedule

The idea of pre-scheduling is that schedule task to the service that satisfies cost and trust optimization while before the deadline. If optimization of scheduleing each task partition can be achieved, then the entire workflow scheduling is optimized.

For entry task: only cost to be considered. Thus, the cheapest service is selected to execute the task while meeting the requirement of the deadline.

For other tasks(including tasks in a task partition and synchronization tasks): select a service with the largest value of the Comprehensive Benefit Function to execute the task while meeting the requirement of the deadline. The Comprehensive Benefit Function is:

$$\begin{cases} Benificial(\mathrm{q}) = \omega_1 \times trust(p,q) / \max\{trust(p,q)\} + \omega_2 \times \min\{ct(q)\} / ct(\mathrm{q}) \\ s.t.\ \ \omega_1 + \omega_2 = 1, t_{comm(V_i,V_j)} + t_{et(V_j)} + t_{comm(V_j)} \le dl[V_j], ct[V_j] \le \cos t[V_j] \end{cases} \quad (6)$$

Thereinto, ω_1 and ω_2 denote trust's and cost's weight respectively. Task V_i and V_j are executed on Grid node p and q respectively, the node q has the largest value of the Comprehensive Benefit Function for scheduling task V_j. In order to avoid bringing more cost due to rescheduling, schedule a task whose execution time is near to its deadline to a more reliable node, namely ω_1 should be bigger; schedule a task whose execution cost is near to its sub-cost to a cheaper node, namely ω_2 should be bigger. In multiple objective scheduling problems, the determin- ation of each objective's weight plays an important role in optimal match between tasks and resources. The value of ω_1 and ω_2 always adjust dynamically according to execution time, deadline and execution cost, the closed cost. See lemma1 for their values.

Lemma1 The relationship between ω_1 and ω_2 is equivalent to that between $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$, and $\omega_v = \frac{1}{c_v \sum_{v=1}^{2} \frac{1}{c_v}}$ ($v = 1,2$. See formula(11) for c_v).

Proof: The probability of rescheduling is less when scheduling task to a more reliable Grid node than to a less reliable Grid node. $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$ denote the propinquity degree of execution time and deadline, execution cost and the closed cost respectively. From the angle of rescheduling, the relationship between $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$ reflects the weight of trust and cost. So, the relationship between ω_1 and ω_2 is equivalent to that between $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$, and we can obtain the value of ω_1 and ω_2 by determining the relation- ship between $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$.

For determining the relationship between $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$, we can model as following: As mentioned above, the number of service on which task T_i can

execute is m_i. Take $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$ as evaluation index, the relationship between $\frac{et[V_i]}{dl[V_i]}$ and $\frac{ct[V_i]}{\cos t[V_i]}$ can be obtained by evaluating m_i services. The elements $x_{uv}(u=1,2,\cdots,m_i;v=1,2)$ of matrix X represent the two index of each service, the evaluation vector is $e_u=(\omega_1,\omega_2)\cdot(x_{u1},x_{u2})^T$.

1) Denote $x_v^*=\max_{1\le u\le 2}\{x_{uv}\},(v=1,2)$, the evaluation vector $e^*=(\omega_1,\omega_2)\cdot(x_1^*,x_2^*)^T$.

$$\begin{cases}\min Y=\frac{1}{m_i}\sum_{u=1}^{m_i}(\sum_{v=1}^{2}(x_{uv}-x_v^*)^2{\omega_v}^2)\\ s.t.\omega_1+\omega_2=1\end{cases} \quad (7)$$

2) Denote $\overline{x_v}=\frac{1}{m}\sum_{u=1}^{m_i}x_{uv},(v=1,2)$, the evaluation vector $\overline{e_u}=(\omega_1,\omega_2)\cdot(\overline{x_1},\overline{x_2})^T$.

$$\begin{cases}\max Z=\frac{1}{m_i}\sum_{u=1}^{m_i}(\sum_{v=1}^{2}(x_{uv}-\overline{x_v})^2{\omega_v}^2)\\ s.t.\omega_1+\omega_2=1\end{cases} \quad (8)$$

3) We can get the relationship of each index after getting the value of ω_1 and ω_2 meeting following condition.

$$\begin{cases}\mathrm{minf}(\omega_1,\omega_2)=\mathrm{minY}-\mathrm{maxZ}\\ s.t.\omega_1+\omega_2=1\end{cases} \quad (9)$$

4) Put formula (7) and (8) into formula (9):

$$f(\omega_1,\omega_2)=\frac{1}{m_i}\sum_{u=1}^{m_i}(\sum_{v=1}^{2}(x_{uv}-x_v^*)^2{\omega_v}^2)-\frac{1}{m_i}\sum_{u=1}^{m_i}(\sum_{v=1}^{2}(x_{uv}-\overline{x_v})^2{\omega_v}^2)$$
$$=\sum_{v=1}^{2}{\omega_v}^2\mathrm{c_v} \quad (10)$$

Thereinto, $\mathrm{c_v}=\frac{1}{m_i}\sum_{u=1}^{m_i}[(x_{uv}-x_v^*)^2-(x_{uv}-\overline{x_v})^2]$
$$=(\overline{x_v}-x_v^*)^2 \quad (11)$$

Suppose Largrange function

$$\mathrm{L}(\omega_1,\omega_2,\lambda)=\sum_{v=1}^{2}{\omega_v}^2\mathrm{c_v}+2\lambda(\sum_{v=1}^{2}\omega_v-1) \quad (12)$$

Evaluate formula (12) for derivative, suppose $\frac{\partial \mathrm{L}}{\partial \omega_\mathrm{v}}=0$, then obtain $\omega_v=-\frac{\lambda}{\mathrm{c_v}}$ (13)

Evaluate formula (9) and (13), obtain

$$\lambda=-\frac{1}{\sum_{v=1}^{2}\frac{1}{\mathrm{c_v}}} \quad (14)$$

Then evaluate formula (13) and (14), obtain

$$\omega_\mathrm{v}=\frac{1}{\mathrm{c_v}\sum_{v=1}^{2}\frac{1}{\mathrm{c_v}}}. \quad (15)$$

3.5. Reschedule

The probability of rescheduling is not high because the scheduling algorithm presented in this paper is based on resource reservation. However, it needs to reschedule the failed tasks once the pre-schedule is invalid because of dynamic changes of Grid environment. The solving strategy of rescheduling is that readjust the sub-deadline or recalculate the optimal scheduling.

For scheduling the failed task, if one or more service(s) that can perform this task before the deadline can still be found, it just needs to reschedule this task to the service that has the maximal value of Comprehensive Benefit Function. Rescheduling does not affect the scheduling of successor tasks. Contrarily, it needs to adjust the deadline of the failed task. The adjustment strategy adopts the way of ¡¶level by level¡¸ which means that it can reduce the deadline of immediately successor or indirect successor of the failed task. The adjustment principle is that the changes of deadline of successor task should not be too sharp and should affect as little successor tasks as possible, for fearing that recalculate the deadline of successor tasks and changing resource reservation, which may induce the decline of system performance.

4. Performance evaluation

We simulate the algorithm proposed in the paper through a Grid simulator GridSim, and draw a comparison with Deadline-MDP algorithm in [6] and Hybrid Balanced Minimum Completion Time (HBMCT) algorithm in [13]. Three metrics, that is task completion time, cost and service refuse rate (SRR), are used to evaluate the above three algorithms. Deadline-MDP algorithm aims at the least cost to execute task while meeting the requirement of dead- line; HBMCT algorithm aims at the least execution time after task ranking and grouping according to cost.

This paper validates TCD algorithm by workflow for genome annotation in [14]. As is shown by Figure 5, the pane stands for task and the number nearby the pane stands for task workload. We establish the following simulation environment based on GridSim: 100 resources, the executing ability of resource is generated randomly between 300 and 3000MIPS, the price of resource is proportional to resource ability; trust relationship exists between any two Grid nodes, and the trust degree is generated randomly between 0 and 1; network bandwidth between nodes is generated randomly between 10M and 100M; I/O data between tasks is generated randomly between 10M and 100M; workflow tasks are scheduled by the above three algorithms respectively under deadlines of 4000, 5000,

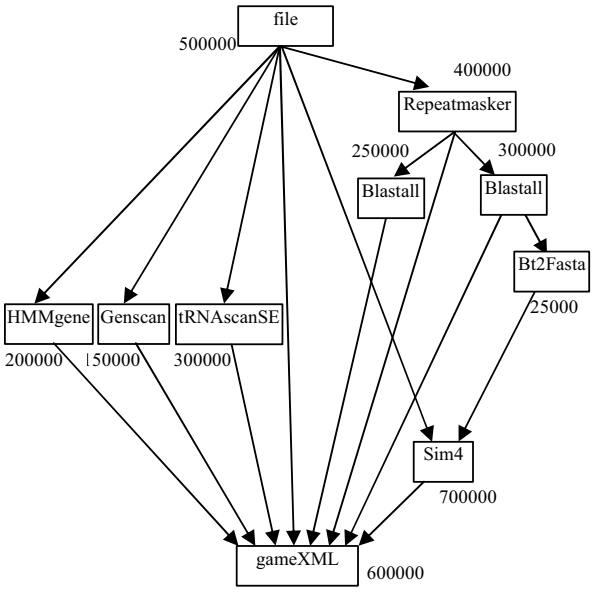


Figure 5. Workflow for genome annotation

6000 and 7000. For different scenarios, we test 50 times and take the average value as the final result.

Contrast of workflow completion time, workflow executing cost and service refuse rate under different deadlines scheduled by Deadline-MDP、HBMCT and TCD algorithm can be seen in Figure 6, 7, 8. With delay of deadline, for Deadline-MDP algorithm, cost decreases gradually and tends to be steady; completion time draws near to deadline in a certain range and tends to be steady; service refuse rate lessens gradually and tends to be steady. The cost tends to be steady because tasks have been scheduled to the cheapest services. For HBMCT algorithm, the completion time and cost changes a bit. For TCD algorithm, the completion time increases gradually but changes softly; the cost changes indefinitely; and it is not as good as Deadline-MDP and HBMCT algorithm in aspect of time and cost in some case, because Deadline-MDP and HBMCT pursues single objective. But TCD algorithm has a quite significant advantage in aspect of service refuse rate, so it is closer to the reality requirements because this algorithm takes into account of cost, communication time and trust between nodes in general.

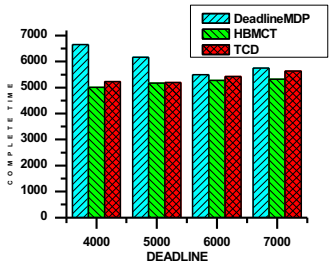


Figure 6. Workflow completion time

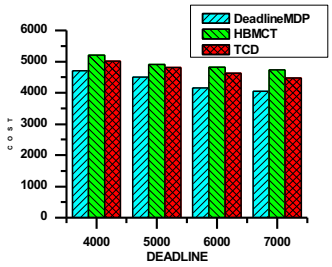


Figure 7. Workflow cost

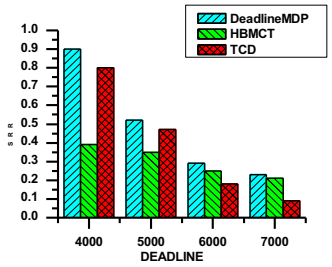


Figure 8. Workflow service refuse rate

As is shown in Figure 6,7,8. In the condition of quite urgent deadline 4000, none of the three algorithm can meet user's QoS requirement of deadl- ine (Note, in order to compare the performance of the three algorithms, the tasks which can't meet the dead- line requirement will still be scheduled to the resource with the shortest execution time); and the cost of the workflow scheduled by HBMCT and TCD algorithm is high because the two algorithms pursue services with fast process speed; for most of workflow tasks, neither of the algorithm can meet the deadline require- ment, so service refuse rate is high. Under the deadl- ine of 5000 and 6000, the variation range of complet- ion time of Deadline-MDP algorithm is wider and the completion time draws near to the deadline; cost may always keep high because HBMCT algorithm is not sensitive to deadline's variation, and service is refused since trust between nodes or communication time neglected; because the completion time of TCD algorithm varies softly, it fits for workflow with stricter deadline. Service refuse rate of Deadline-MDP algorithm can reduce rapidly, because there is enough time to execute the tasks. But compared with TCD algorit- hm, its service refuse rate is still high, because the former fails to consider the communication time between tasks and so that leading to failure of task scheduling. In the condition of a loose deadline such as 7000, the cost of the workflow scheduled by Deadline-MDP algorithm is the lowest and tends to be steady ultimately. Therefore, cost and completion time will vary softly under longer deadline. But in the condition of a loose deadline, because of overlooking trust relationship between Grid nodes, services may still be refused and the probability of rescheduling of Deadline-MDP and HBMCT algorithm is still high.

5. Conclusion

Introducing workflow technique into Grid and the QoS-based task scheduling are a hotspot field in Grid research. On the basis of active network, a distributed, scalable workflow management system model which can adapt to dynamic characteristic of Grid is presented in this paper. Workflow instance can be executed in parallel in the model, thus, system overhead is reduced and network workload is balanced. The presented workflow task scheduling algorithm considers user's multi-QoS comprehensively. We evaluate the algorithm through GridSim simulator. Comparing with other two related algorithms, it shows that the algorithm we present can guarantee user's multi-QoS requirements quite well. Deadline is regarded as a limited condition in the paper, and in the future we will take time as the QoS objective and further research on task rescheduling with lower cost.

References

[1] J. Yu, R. Buyya. A Taxonomy of Workflow Management Systems for Grid Computing. Journal of Grid Computing, 2005, 3(34):171-200.

[2] W.M.P. van der Aalst, B.F. van Dongen, J.Herbst et al. Workflow Mining: A Survey of Issues and Approaches. Data & knowledge Engineering, 2003, 47(2):237-267.

[3] J. Cao, S. Jarvis, S. Saini, and G. Nudd. GridFlow: Workflow Management for Grid Computing. In: 3rd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGrid'03). Tokyo: IEEE Computer Society Press, 2003, 198-205.

[4] Y. Feng, W. Cai, and J. Cao. A Simulation Study of Job Workflow Execution Models over the Grid. In: 2nd International Workshop on Grid and Cooperative Computing. Shanghai, China, 2003, 935-943.

[5] J. Schneider, B. Linnert, L. Burchard. Distributed Workflow Management for Large-Scale Grid Environments. In: 2006 International Symposium on Applications and the Internet (SAINT'06), Phoenix: IEEE Computer Society, 2006, 229-235.

[6] J. Yu, R. Buyya, C. K. Tham. Cost-based Scheduling of Scientific Workflow Applications on Utility Grids. In: Proc. of the First International Conference on e-Science and Grid Computing, Melbourne: IEEE Computer Society, 2005, 140-147.

[7] W. Zhang, B. Fang et al. A Trust-QoS Enhanced Grid service Scheduling. Journal of Computer, 2006, 29(7):1157-1165.

[8] C. Li, L. Li. Optimal Multiple QoS Resource Scheduling In Grid Computing. In: Proc. of the 20th International Conference on Advanced Information Networking and Applications (AINA'06), Vienna: IEEE Computer Society, 2006, 240-246.

[9] D.L Tennenhouse et al. A Survey of Active Network Research. IEEE Communications Magazine, 1997, 35(1):80-86.

[10] F. Ren, Y. Ren, X. Shan. Research and Development of Active Network. Journal of Software, 2001, 12(11):1641-1622.

[11] Tennenhouse, D. Wetherall. Towards an Active Network Architecture. ACM SIGCOMM Computer Communications Review, 1996, 26(2):5-17.

[12] S. Beco, B. Cantalupo, L. Giammarino, N. Matskanis, M. Surridge. OWL-WS: A Workflow Ontology for Dynamic Grid Service Composition. In: Proc. of the First International Conference on e-Science and Grid Computing, Melbourne: IEEE Computer Society, 2005.

[13] P. Tullmann, M. Hibler, J. Lepreau. Janos: A Java-oriented OS for Active Network Nodes. In: Proc. of the DARPA Active Networks Conference and Exposition (DANCE`02). San Francisco, USA, 2002, 117-123.

[14] R. Sakellariou and H. Zhao. A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems, In: Proc. 18th International Parallel and Distributed Processing Symposium. Santa Fe: IEEE Computer Society Press, 2004, 111b.

[15] S. P. Shah, D. Y. M. He et al. Pegasys: Software for Executing and Integrating Analyses of Biological Sequences. http://www.biomedcentral.com/1471-2105/5/40.