Deadline Guarantee Enhanced Scheduling of Scientific Workflow Applications in Grid

Chaokun Yan*, Huimin Luo
School of Computer and Information Engineering, Henan University, Kaifeng, China
School of Information Science and Engineering, Central South University, Changsha, China
Email: {ckyango, luohm1980}@yahoo.com.cn

Zhigang Hu, Xi Li
School of Information Science and Engineering, Central South University, Changsha, China
Email: {zghu, lixi}@mail.csu.edu.cn

Yanping Zhang
Department of Genome Oriented Bioinformatics, Technical University of Munich, Freising, Germany
Email: zhang@wzw.tum.de

***Abstract*—Many scientific workflow applications often have deadline constraint such that all tasks of application need to be finished within user-specified time. Though Grid systems have high performance, which provide an important infrastructure for executing scientific workflow, dynamic nature of Grid resources makes it very difficult to schedule workflow and satisfy the given deadline. For this issue, a M/M/C queuing model was adopted to model dynamic service capacity of grid resource. Under dynamic, unreliable grid environment, based on queuing system theory, a probability of meeting deadline for each task can be calculated. The detail analytic method and proof was given in the paper. Specifically, a novel probability evaluation based scheduling algorithm is proposed to address the problem of workflow' deadline guarantee. The results of large scale simulation experiments with real Protein annotation workflow demonstrate that our algorithm can remarkably improve the predictability of workflow completion and guarantee user's deadline requirements.**

***Index Terms*—Scientific Workflow, Deadline, unreliable, Grid, Markov, queuing model**

I. INTRODUCTION

Large scale computations from various scientific endeavors such as drug discovery, weather prediction, medical surgery and other applications are composed as a sequence of dependent operations or workflows [1]. Grid computing (also knows as computational grid) is proposed and considered as the foundation of next generation Internet [2], which provisioning an infrastructures with high computational power for these scientific applications. The last decade has seen an unprecedented growth in Grid infrastructures which nowadays enables large scale deployment and execution of scientific applications, such as TeraGrid, Earth System Grid, Enabling Grids for E-scienceE, etc [3]. Among these research domains, specially, a class of scientific applications must complete in a timely manner to generate appropriate results. LEAD [4] and GLCFS [5] are examples of such deadline-sensitive workflows, which require careful coordination of workflow tasks with underlying grid resources.

In the literature of workflow scheduling, existing research efforts mainly focus on minimizing the execution time of the scientific applications or optimizing certain Quality of Service (QoS) metric in application deadline. Proposed algorithms always assume the performance data of Grid sites can be obtained and trusted before schedule. However, Grid infrastructure in general is not a dedicated system, which leads to load variations or unavailable in Grid resource nodes. The dynamic nature of resource has been the most significant obstacles in grid workflow scheduling. Therefore, aforementioned scheduling strategy based static resource information cannot absolutely guarantee the successful completion of scientific workflow and satisfy the user' QoS requirements. To remedy this situation and improve the predictability of workflow scheduling under deadline constraint, a stochastic service model is used in this paper to characterize the load and service capacity of grid resources, a scheduling algorithm for deadline-sensitive scientific workflow is proposed.

The remainder of the paper is organized as follows. We survey related work in section II. Then section III describes related concepts and problem formulation in details. Scheduling Algorithm for Deadline Sensitive Scientific Workflow (SADSSW) is presented in section IV. Section V gives the experimental setup and describes the results of algorithm for the execution of a Protein annotation workflow, followed by concluding remarks in section VI.

II. RELATED WORKS

Manuscript received January 1, 2012; accepted July 1, 2012.
Project number: NSFC (60970038 and 61272148).
Corresponding author: ckyango@yahoo.com.cn(Chaokun Yan)

doi:10.4304/jcp.8.4.842-850

Many scientific projects always involve exabyte-scale computing to process the huge data sets. Moreover, for some scientific applications, these large scale data need to be processed before predefine deadline. Nowadays, although cloud computing [6] platforms such as Amazon EC2 has generated significant attention from HPC and scientific research, Grid systems is still an important platform for scientist to deploy and execute their applications. The work in this paper will focus on workflow scheduling problem in Grid environment.

The issue of workflow scheduling in grid systems has been extensively investigated. So far, overall execution time of workflow has been a major concern QoS metric. In order to optimization execution time of workflow in heterogeneous distributed environments, Topcuoglu [7] proposed a Heterogeneous-Earliest-Finish-Time (HEFT) algorithm used in ASKALON [8] workflow manager, which is a powerful DAG scheduling algorithm designed for heterogeneous computational environments. Some researchers formulated the workflow scheduling problem as a special case of the DAG scheduling problem. Most of DAG scheduling algorithms in traditional parallel and distributed systems was proposed to optimize the execution time, such as MCP, DCP. However, those algorithms cannot be well applied to Grid Environments considering the remarkable difference in the architecture and characteristics between Grid and traditional workstation or cluster. Some intelligent algorithm, such as genetic algorithm (GA), particle swarm optimization (PSO) and ant colony optimization (ACO) have been successfully applied to a wide range of applications such as function optimization, task assignment, and scheduling problems. For instance, an optimization algorithm of grid task scheduling is brought forward by using classification strategies to improve particle swarm algorithm in [9]. For the issue of scheduling workflow in grid environment, Jia Yu [10] proposed a new type of genetic algorithm, which has been used to the broker of Gridbus Project [11] in university of Melbourne.

In many cases, though proposed many intelligent algorithms can get approximate optimal solution, those algorithms have high computational complexity making it unsuitable for practical dynamic grid environments. Moreover, deadline constraint is not considered in these researches.

Aimed at scheduling scientific workflow with deadline constraint, some existing works mainly focus on optimizing other QoS metric in deadline, such as cost or reliability. A method based on row generator technique for solving the discrete time/cost trade-off problem was proposed in [12]. Yuan designed DBL (deadline bottom level) [13] algorithm which generated iteratively feasible solutions in iterative heuristics from leveling-based initial solutions to optimize cost. The above researches mainly focused on transforming the structure of DAGs to task schedule based on static performance of grid resources, which can not guarantee the deadline due to dynamicity of target resources.

As we know, the dynamicity of resource mainly shows in the following aspects: load variations in links and processing node, unavailability or failure of resource nodes. Above-mentioned scheduling algorithms will not guarantee the successful completion of workflow application because the negative impact of load on queuing time of task and resource failure was not considered carefully. To improve the predictability of workflow execution, fault tolerance scheduling or advanced reservation is always adopted to remedy the situation. However, though resorting to fault tolerance mechanisms for workflow scheduling such as replication or resubmission can alleviate the impact of unreliability of the environment, fault tolerance often leads to unnecessarily resource consumption and degrades the QoS of grid system. Many researchers have been interested in taking advance reservation as an effective technique for workflow scheduling to improve predictability. For instance, Xiao et al [14] present a novel relaxed reservation policy, which can be applied to workflow scheduling. M. Wieczorek et al. [15] presented an extension to devise and implement advance reservation as part of the scheduling and resource management services of the ASKALON. However, advance reservation also brings many negative effects on resources and task scheduling, such as resource fragment and lower resource utilizing.

Instead of avoiding the dynamicity of resource by reservation or alleviating the impact of dynamicity on the scheduling performance by fault tolerance, some dynamic scheduling strategies based on performance evaluation and prediction have been widely paid attention to by researchers. A resource performance fluctuation aware workflow scheduling algorithm (PFAS) is an example for scheduling workflow based on just-in-time information in literature [16]. Instead of using a static task ranking approach of a DAG scheduling algorithm, PFAS updates task ranks and constructs the critical path dynamically in the scheduling procedure according to the change in performance of grid resources. Experiments show that the scheduling performance of PFAS measured in makespan is much better than HEFT algorithm. Because running scientific application in overloaded Grid environment often become latency-bound and suffers from serious deadline missing. As a result, the queuing time of task in available resource should be considered in the scheduling procedure. The real grid scheduler in VGrADS [17] predicted the queuing time of tasks and selected resource based on it, while the evaluation metric is limited to the specified grid system.

Unlike the aforementioned work, this paper will mainly focus on measuring dynamic service capacity of resources and scheduling workflows based on the evaluation results. Unlimited to the specified grid systems, we would try to mask resource heterogeneity via providing a uniform metric to evaluate the service capacity of grid resource.

III. PROBLEM FORMULATION

A workflow scheduling flow can be depicted as Figure1. First, user can submit their execution request by WMS (workflow management system) Portal. Different

Grid portals support different levels of application parallelism and they are typically tailored to one particular Grid. Grid Portal can provide the necessary user interface for the workflow concepts and the workflow application hosting environments. User can compose their workflow applications at high level of abstraction using various modeling tool, such as UML. Then, the abstract workflow representation is submitted to the Workflow Meta-Scheduler for transparent execution onto the underlying infrastructure.

Planner is the core module of meta-scheduler, which is responsible for deciding the execution sequence of each task. When all parent tasks of one task have been completed, the task will be put into Task Pool to wait to be scheduled. The Task dispatcher is responsible for mapping the tasks of scientific workflow to underlying resources. Before making a mapping decision, it needs to send a request of query to the Grid Catalogue Service to get the list of candidate resources. Performance Evaluator is responsible for estimating the execution performance for candidate resources according to the performance data from the underlying Infrastructures. Finally most suitable grid resource will be selected to execute tasks by Task Dispatcher. Grid Services are mainly responsible for supporting the functions of other modules, such as Resource Management, Information Service, Security Guarantee, et al. Once receiving tasks from Task Dispatcher, Local Scheduler will execute those tasks according to local scheduling policy.

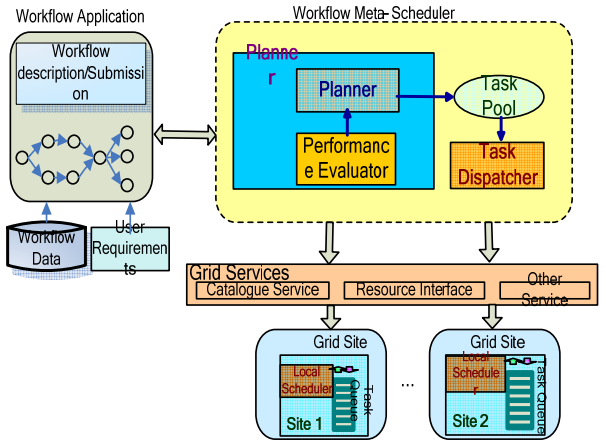


Figure1.Scientific Workflow Execution Architecture

The workflow structure, also referred as workflow pattern, indicates the temporal relationship between these tasks, which can be described as a DAG. In the paper, we introduce a two-tuple representation $<G,D>$ to denote a scientific application, where $G=<T,E>$ is the workflow structure described by a DAG. Let set $T=\{t_i \mid 1\le i\le m\}$ be the collection of tasks in the workflow. Let set $E=\{e_{ij} \mid 0<i,j\le n, i\ne j\}$ indicate the dependency and precedence constraint between tasks. Each edge is denoted by (t_i,t_j), where t_i is called an immediate parent task of t_j, and t_i is the immediate child task of t_i. A child task cannot be executed until all of its parent tasks have been completed.

In a workflow G, an entry task does not have any parent task and an exit task does not have any child task. Without loss of generality, in the paper, we assume that there is only entry task and exit task in workflow G. If there multiple entry tasks and exit tasks in real workflow application, we connect them to a zero-cost pseudo entry or exit task. In addition, D denotes the deadline constraint for scientific workflow. We assume that the overall deadline D can be divided into sub-deadline $d_i(1\le i\le m)$ for each task $t_i(1\le i\le m)$ in a workflow.

We assume that a grid system is composed of N resource nodes, represented as $(R_1,\ldots,R_N)$. Each resource node can be seen a cluster with c_i computing elements. Because of some reasons such as computer failure, hardware upgrade, the real available number of computer is not always to hold the value c_i. For the issue, in order to describe the situation, for a cluster with c_i computing elements, a birth–death process could be modeled with failure rate λ_{si} and repair rate μ_{si}.

According to the theory of birth-death process, assume that state X represents that the number of available computing elements is X ($X=\{0,1,2,\ldots c_i\}$). It is clear that X is a stochastic variable. Denote $p_X(i)$ as the steady probability for the cluster i staying at state k. So, it is simple to derive $p(X=k)$ by

$$p_0=\frac{1}{1+\sum_{k=1}^{n}\frac{n!\mu_{si}^k}{k!(n-j)!\lambda_{si}^k}}. \quad (1)$$

$$p(X=k)=\left(\frac{n!\mu_{si}^k}{k!(n-k)!\lambda_{si}^k}\right)\cdot p_0 \quad (k=1,2,\ldots,N). \quad (2)$$

Once the steady number of computing elements has been obtained, the key of scientific workflow scheduling immensely depends on an estimation of performance for candidate resource nodes. Performance estimation is the prediction of performance of task execution on resource node and is crucial for generating an efficient schedule. Especially for workflow scheduling with deadline constraint, though deadline of each task in workflow can be obtained by some deadline distribution strategy, deadline of sub-task could not be guaranteed because of the performance fluctuation of resource node. There are related works in performance modeling of distributed networks computing in dedicated and non-dedicated systems. Different performance estimation approaches can be applied to different research field. Recently, many researchers [18-19] have applied queuing system to describe the working of grid resources. These studies show that queuing system is capable of precisely describing the working and workload model of grid resources in dynamic environments. Similarly, in the

paper, we adopt M/M/C queuing system to describe resource nodes' workloads and their dynamic service process depicted as a three-tuple $< \lambda_i, \mu_i, n_i >$, in which λ_i represents mean time interval between task arrivals on R_i, μ_i and n_i are mean serving time and number of resources of R_i respectively.

Assume that we have the steady number k_i of available computing elements for every resource node according to (1) and (2). Then, considering the M/M/C queue model, we can get the number of waiting tasks in the schedule queue of resource node. Once a task of workflow has been ready, then the task dispatcher can decide the most suitable resource node for the task according to theorem 1 as follows.

Theorem 1 (Deadline Guarantee Probability) Given that a resource R_i with three parameters μ_i, λ_i, c_i, then for each task t with deadline d_m, the probability that R_i can complete task t before d_m, which we call Deadline Guarantee Probability, can be calculated as follows:

$$P(t<d_m)=\sum_{k_i=1}^{c_i} p_{k_i} \cdot \left(\sum_{n=0}^{k_i} \delta \cdot \frac{(\lambda_i/\mu_i)^n}{n!} + \sum_{j=1}^{k_i \cdot \mu_i \cdot d_m - 1} \delta \cdot \frac{(\lambda_i/\mu_i)^{j+k_i}}{k_i^j \cdot k_i!}\right) \quad (3)$$

For the proof, see Appendix A.

So, in the paper, the workflow scheduling is to map every task of workflow application onto a suitable resource to maximize the probability of completion before deadline, we adopt heuristic method to obtain a mapping solution between task and resource. Proposed scheduling algorithm is described in details in section IV.

IV. SCHEDULING ALGORITHM

In the paper, we take the "deadline guarantee probability" as the important metric for resource selection. So the key idea of the scheduling algorithm proposed is to satisfy the deadline constraint by finding the best resource with highest deadline guarantee probability based on theorem 1.

The primary task of scheduling algorithm is deadline distribution, which refers to assign users' overall deadline into every task. There exist some strategies for workflow deadline distribution, such as distribution strategy of Pegasus [20], DTL(deadline top level) [21], DBL(deadline bottom level) [22]. DTL and DBL distribute overall deadline of workflow into each subtasks by dividing all subtasks into groups using a forward or backward method. In DBL, the starting time of a task in each group is determined by the maximum finish time among those of its predecessors rather than the finish time of its parent group, which is adopted by DTL. The total time float is allocated equally to each group, which manage to enlarge the cost optimization intervals of all tasks. Comparatively, DBL can obtain better cost optimization than DTL. However, our paper mainly focuses on the performance evaluation of grid resource by queuing model. Moreover, DTL is a simple and easy-to-implement method with low time-complexity, which has been applied to Gridbus Scheduler, which means it is a very practical method. So DTL is adopted to distribute overall deadline of workflow into each subtask and get a sub-deadline vector of all workflow tasks $\{d_1, \ldots, d_m\}$.

Some definitions related are introduced firstly. Workflow tasks are first categorized as either synchronization tasks or simple tasks. A synchronization task is defined as a task which has an in-degree or out-degree greater than one. For example, T_1, T_5, T_{14} in fig.2 are synchronization tasks. Comparatively speaking, simple task is defined as a task which has both in-degree and out-degree equal to one. A branch is a set of simple tasks between two synchronization tasks. For example, $\{T_2, T_3\}$, $\{T_4\}$, $\{T_6\}$, $\{T_7, T_8\}$, $\{T_9, T_{10}, T_{11}\}$, $\{T_{12}, T_{13}\}$ are branches in fig.2.

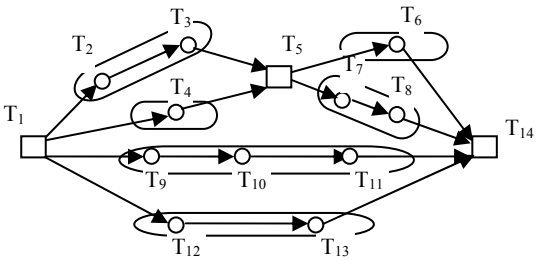


Figure 2. Workflow example

After task partitioning, workflow tasks can be clustered into some partitions. As a result, the original workflow can be transformed into a workflow partitioning graph. Fig.3 is a workflow partitioning graph for fig.2.

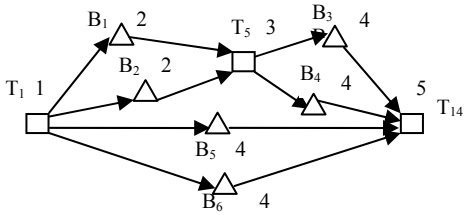


Figure 3. Workflow partitioning graph

The deadline assignment strategy can be described as followings.

(1) Depth-First Search was adopted firstly to travel workflow G. Then the longest path will be found, we can mark its task vertex from one to gn. For example, the longest path in figure 3 is $\{T_1, B_1, T_5, B_3, T_{14}\}$ and $\{T_1, B_2, T_5, B_4, T_{14}\}$.

(2) Then, we mark the other paths by depth-first algorithm from end vertex of workflow G. Mark the unmarked vertex from gn in inverted sequence. The task partitions with same mark will be assigned same sub-deadline. As a result, we can get the sub-deadline vector of all task partitions by assigning sub-deadline for the longest paths.

(3) The actual computation length for all tasks in workflow is different. In the paper, we assign the sub-deadline based on the proportion of computation length of every task partition. If a task partition has larger computation length, then it should be assigned more time to execute. What needs to be pointed out is that the branch with biggest computation length should be selected when there are several branches between two synchronization tasks.

Through the above analysis, we can obtain the proportion of computation length for all task partitions as follows:

$$QT[T_1]:QT[V_2]:\cdots:QT[V_{gn}] \quad (4)$$

Then, the assigned deadline of each partition can be defined by:

$$d[V_i]=\frac{QT[V_i]}{\sum_{k=1}^{gn}QT[V_k]} \quad (5)$$

Where, the denominator in above express represents the overall computation length of a longest path.

(4) In the same way as (5), we can get the sub-deadline of every task in a task partition. Finally, we can obtain a sub-deadline vector for all workflow tasks, which is represented as $\{d_1,\ldots,d_m\}$.

The equation (3) is the criterion of our scheduling algorithm. Maybe several tasks are scheduled to the same resource node. We use an array Q of wait queue when selecting resource. That is, as a task is scheduled to a resource node, the probability of completion before its deadline for the same resource would be decreased. So the probability $P(t<d_m)$ can be modified as

$$P(t<d_m)=\sum_{k_i=1}^{c_i}p_{k_i}\cdot\left(\sum_{n=0}^{k_i}\delta\cdot\frac{(\lambda_i/\mu_i)^n}{n!}+\sum_{j=1}^{Q}\delta\cdot\frac{(\lambda_i/\mu_i)^{j+k_i}}{k_i^{\,j}\cdot k_i!}\right) \quad (6)$$

In the following, we present a scheduling algorithm for deadline sensitive scientific workflow, SADSSW. In the algorithm, once obtaining the sub-deadline assignment of all subtasks, the task dispatcher begins to schedule the subtasks of workflow level by level. The pseudo code of scheduling algorithm is described as follows. The tasks are ordered in non-increasing relative deadline order.

INPUT: $R_i(1\le i\le n)$,, λ_i,μ_i,c_i,p_i, $<G,D>$..

OUTPUT: task-resource mapping $mapping(t_i,r_j)$

1. Begin
2. Request average processing capacity from available Grid resource for each task and task characteristics
3. Distribute Deadline D over all tasks of workflow according to (6)
4. Initialization Task Pool Array Q=Null, indicator i=1;
5. Put the entry task of workflow into Task Pool Q;
6. While Q is not empty
7. Get first task t_i from Q
8. for $j=1$ to n in Task Pool
9. calculate $P(t_i<d_m)$ using formula (6);
10. select the biggest P value as candidate resource of t_i;
11. end for
12. Put the children of task t_i into Task Pool Q when all parent tasks of t_i have been scheduled;
13. $i++$;
14. end while
15. output task-CE mapping $mapping(t_i,r_j)$.

End

To compute the time complexity of our proposed algorithm, suppose the number of tasks of workflow is m. We assume that the maximum number of the available grid resources for a task is n, the maximize width of workflow is l. It first distributes the overall deadline to all tasks (Line 2-3). Adjacency list was adopted as graph storage structure in the algorithm, so the time complexity of graph travel (Depth-First-Search) is O(m+e), where m and e represent the number of tasks and edges respectively. Once getting the longest path, the time complexity of assigning deadline is k*O(gn), where gn and k represent the number of workflow partitions and tasks in each partition respectively.

After that, it starts to schedule tasks of the workflow from the entry task (Line 5). Then, select the resource node for each of sorted tasks in turn according to (6) (Line 7-11), the time complexity is O(n), where n represent the number of resource. When the number of tasks is m, this process (Line 7-11) will be repeatedly m times to generate mapping result. The time complexity is O(m*n).Once all parents of a task have been scheduled, the task is ready for scheduling and then the dispatcher put it into queue (Line 12), the time complexity can be considered as a constant time.

To sum up, the time complexity of SADSSW is O(m*n). Comparatively speaking, the number of available grid sites for scientific workflow is much less than the number of tasks of workflow, which mean $n<<m$. So the execution time of Line 7-11 will not take a long time. As a result, the scheduling algorithm (SADSSW) is a practical solution.

V. Experiments and Analysis

A. Experimental Setup

We simulate the Grid environment by using the GridSim5.0 toolkit and implemented our scheduling algorithm on a single machine with Intel Core Dou processor and 2G RAM. We have changed some classes of GridSim to support for parameters of our algorithm. In our simulations, the Grid consists of 9 sites based on the valid data on April 1, 2010 of Grid'5000 as shown in table 1. Based on the analysis of Workloads Archive of Grid'5000 in [22] we conclude that parameter λ which is the interval between task arrivals and parameter μ which

TABLE1.

CONFIGURATION DATA OF RESOURCE NODES

ID	Grid site	Processor number	Avg. processing power(GHZ)
1	Orsay	684	2.15
2	Grenoble	68	2.5
3	Lyon	260	2.22
4	Rennes	376	2.47
5	Sophia	400	2.26
6	Bordeaux	424	2.6
7	Lille	290	2.53
8	Nancy	518	1.99

is task execution time follow exponential distribution, and their values should be uniformly distributed in [0, 1] and [1, 2] respectively.

Nowadays, some efforts such as Computer Failure Data Repository [23], the Desktop Grid Failure Traces [24] and Failure Trace Archive [25] have led to making failure-related data public. For required data, we can use statistical methods and maximum likelihood estimation to parameterize the distributions. For instance, literature [26-28] analyzes failure data and model the time between failures, and the repair time. The related researches lay important foundation for our scheduling algorithm. The detailed description how to get the characteristic of resource failure is out of the scope of this paper. In our experiment, the exponential failure model is used to evaluate the birth-death Markov process of available computing elements shown in Section III. Three different failure conditions are defined according to the value of failure rate λ_{si} for each resource node, which are named mild failure, moderate failure and severe failure. We assume that the corresponding value of λ_{si} are uniformly distributed in [0.01, 0.03], [0.03, 0.06], [0.06, 0.1] respectively. The repair rate μ_{si} of each computing unit is predetermined and set as a fixed value, ie.1.2.

Similar to literature [29], a real protein annotation workflow [30] was adopted to test proposed algorithm in the experiment. Figure 5 give its architecture and task characteristic data, such as task name and their computation requirement. For example, SignaIP is a workflow task with 300000 Million instruction computation length in the first workflow level.

As user's deadline parameter is not recorded in the trace data, we adopt a similar experimental methodology in [31] to model the parameter as: $D = \sigma T$, where T is the execution time of workflow in the best resource of all tasks in critical path of the currently scheduled workflow, and σ is the urgent factor of the deadline which is taken from the set {1.5, 2.0, 2.5}.

The proposed SADSSW is compared with PFAS and the scheduling algorithms in the practical grid systems VGrADS with different resource workloads and deadline constraints. For each scenario, we did the experiments 50

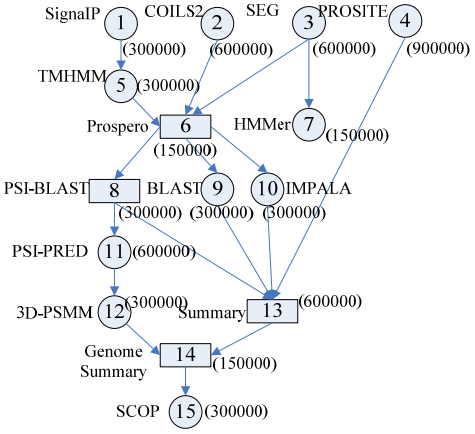


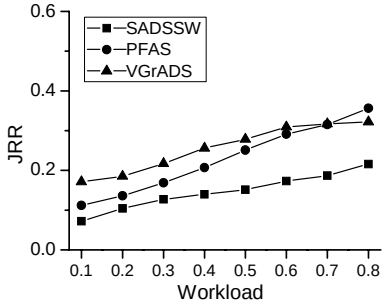
Figure 5. Protein annotation workflow.

times independently and statistic the experimental results. Then we take the average value in JRR (job rejection rate) metric to evaluate the performance of three algorithms. The JRR means the ratio of the number of unfinished tasks (job means task in the experiments of this paper) and the total number of workflow tasks.

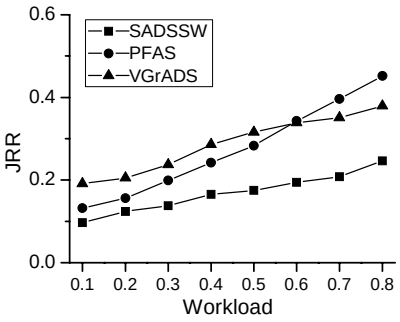
B. Experimental Results and Analysis

1) Comparison of JRR

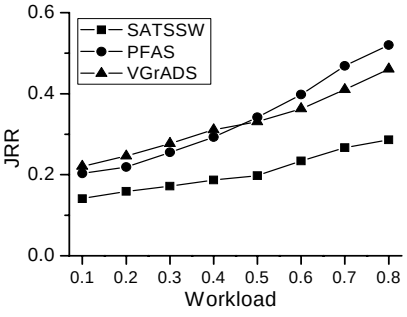
In Fig.6, we show the JRR comparison with different resource workloads under three failure conditions.



(a) Mild failure condition



(b) Moderate failure condition



(b) Severe failure condition

Figure 6. JRR Comparison for different failure condition

As shown in Fig.6 (a), The JRR of three algorithms increases with increase in workload. For mild failure condition, the average JRR of SADSSW will be better than PFAS by 24.3% and VGrADS by 32.5%. This is due to the fact that queuing time of workflow tasks will account for a greater proportion of the schedule length as the resource workloads are heavy and resource failure can be ignored. When failure happens frequently, the gap of JRR between SADSSW and others increase dramatically. As we can see in Fig.6 (c), the average JRR of SADSSW will be better than PFAS by 35.8% and VGrADS by 43.7% correspondingly. Comparatively, JRR of the proposed algorithm SADSSW has been consistently lower than PFAS and VGrADS. As a result, we can see: (1) the workloads of resource and failure condition have significant impact on the execution of workflows; (2) the M/M/C considering resource failure can be used to evaluate dynamic service capacity of grid node, which comprehensively considers execution time and queuing time of each task.

2) Performance impact of deadline and failure rate

We investigate the impact of different deadline and failure condition on scheduling performance for these three algorithms. When resource lies in mild failure, the average JRR comparisons of three algorithms are shown in Fig.7.

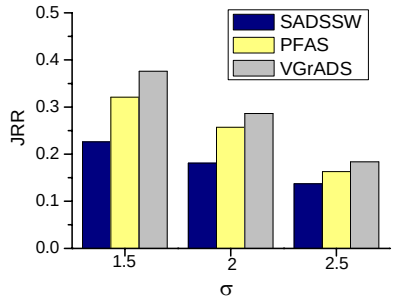


Fig.7 Deadline impact on workflow performance.

As we can see, when deadline constraint is very strict, three algorithms all have relatively high JRR. Task σ =1.5 as example, the JRR of SADSSW is better than VGrADS by 45.8%, and PFAS by 36.9%. As deadline gets looser, successful number of tasks for three algorithms will increase gradually.

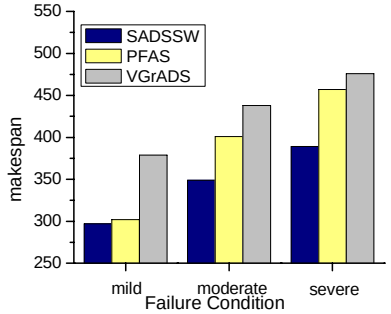


Fig.8 Failure condition on workflow performance.

As shown in Fig.8, when failure becomes very frequent, makespan of three algorithms will increase dramatically. This shows that failure has big negative impact on resource performance. However, the average makespan of SADSSW is the least one of the three scheduling algorithms for different failure condition.

Accordingly, it is not difficult to draw the conclusion that SADSSW can produce better schedules than other algorithms. This is because SADSSW can find the best resource to execute each task by evaluating dynamic service capacity of resources. Both the workload fluctuation and reliability status of resource are considered by SADSSW, which also indicates proposed algorithm have better adaptability for deadline requirement in dynamic grid environment. Experimental results further verify that proposed model can describe the real executive characteristic of grid resource.

VI. CONCLUSION

Guaranteeing deadline of scientific workflow in grid is a hard problem. Dynamicity of resource node is the biggest obstacle. So, a birth-death process is built to model the availability of grid node firstly. Then we adopt M/M/C queuing system to characterize dynamic workload and service capacity of resource. Based on queuing system theory, a probability of deadline guarantee was put forward. Finally, a scheduling algorithm for deadline-sensitive scientific workflow is proposed. The experimental results prove that our algorithm is better than others in terms of adaptability and predictability under deadline constraint. In future, we will consider performance impact of deadline distribution and propose better deadline distribution strategy.

APPENDIX A PROOF FOR THEOREM 1

Theorem 1 Given that a resource R_i with three parameters μ_i , λ_i and c_i , each task t with deadline d_m , when the failure condition of resource is considered, the probability that R_i can complete task t before d_m is

$$P(t<d_m)=\sum_{k_i=1}^{c_i} p_{k_i}\cdot\left(\sum_{n=0}^{k_i}\delta\cdot\frac{(\lambda_i/\mu_i)^n}{n!}+\sum_{j=1}^{k_i\cdot\mu_i\cdot d_m-1}\delta\cdot\frac{(\lambda_i/\mu_i)^{j+k_i}}{k_i^{\,j}\cdot k_i!}\right)$$

Where, $\sum_{k_i=1}^{c_i} p_{k_i}$ can be calculated by formula (1) and (2) as shown in Section III.

Proof.

Let a stochastic variable ψ represent the number of tasks waiting in the queue of R_i . According to queuing theory [30], the probability that the number of tasks waiting on CE_i is k should be:

$$\Pr\{\psi=k\}=\begin{cases}\delta\cdot\dfrac{\rho_i^{\,k+c_i}\cdot c_i^{\,c_i}}{c_i!}, & k>0\\ \sum_{n=0}^{c_i}\delta\cdot\dfrac{(\rho_i\cdot c_i)^n}{n!}, & k=0\end{cases}\qquad(7)$$

According to the property of M/M/c_i queuing system, $c_i \cdot \mu_i$ is the service rate of resource R_i, i.e. the average number of tasks completed on R_i in a unit of time. Therefore, the average number of tasks that R_i can finish in d_k units of time should be $c_i \cdot \mu_i \cdot d_k$. So, according to queuing theory, the probability that the task t can be completed by R_i before deadline d_k is as follows

$$\Pr\{\varphi \le d_k\} = \sum_{n=0}^{c_i} \delta \cdot \frac{(\rho_i \cdot c_i)^n}{n!} + \sum_{k=1}^{c_i \cdot \mu_i \cdot d_k - 1} \delta \cdot \frac{\rho_i^{k+c_i} \cdot c_i^{c_i}}{c_i!} \quad (8)$$

Where $\delta = \left[\sum_{n=1}^{c_i-1} \frac{(\rho_i \cdot c_i)^n}{n!} + \frac{(\rho_i \cdot c_i)^{c_i}}{c_i! \cdot (1-\rho_i)} \right]^{-1}$, φ is a random variable representing actual completion time of this task.

As mentioned above (Section III), the real number of computing elements in a grid resource is not equal to c_i assuredly. It is necessary to decide the actual available number of computing elements for each grid node. According to formula (1) and (2), we can obtain it.

Combining formula (1), (2) and (8), we have that

$$\begin{aligned} P(t < d_m) &= P\{X = k, \varphi \le d_m\} \\ &= P\{X = k\} P\{\varphi \le d_m \mid X = k\} \end{aligned} \quad (9)$$

Where $P\{X = k\}$ is the probability density function of waiting queue length, which can be obtained according to the theory of queuing system in [32]. Therefore, we can calculate $P(t < d_m)$ as following

$$\begin{aligned} &\Pr\{t \le d_m\} \\ &= \Pr\{X = k_i\} \Pr\{\omega \le d_m \mid X = k_i\} \\ &= \sum_{k_i=1}^{c_i} p_{k_i} \cdot \left(\sum_{n=0}^{k_i} \delta \cdot \frac{(\lambda_i/\mu_i)^n}{n!} + \sum_{j=1}^{k_i \cdot \mu_i \cdot d_m - 1} \delta \cdot \frac{(\lambda_i/\mu_i)^{j+k_i}}{k_i^{\,j} \cdot k_i!} \right) \end{aligned}$$

End.

ACKNOWLEDGMENT

We appreciate the valuable comments from anonymous reviewers. This work was supported by the National Nature Science Foundation of China under grant No.60970038.

REFERENCES

[1] L. Ramakrishnan, J.S. Chase, D. Gannon, et al, "Deadline-sensitive workflow orchestration without explicit resource control", Journal of Parallel and Distributed Computing, vol.71, no.3, pp.343-353, 2011. "doi: 10.1016/ j.jpdc. 2010.11.010"

[2] J. Xu, A.Y.S. Lam, V.O.K. Li, "Chemical Reaction Optimization for Task Scheduling in Grid Computing," IEEE Transactions on Parallel and Distributed Systems, vol.22, no.10, pp.1624-1631, 2011. "doi:10.1109/ TPDS.2011.35"

[3] W. Gentzsch, D. Girou, A. Kennedy, et al, "DEISA-Distributed European Infrastructure for Supercomputing Applications", Journal of Grid Computing, vol.9, no.2, pp.259-277,2011. "doi:10.1007/s10723-011-9183-2"

[4] U.U. Turuncoglu, S. Murphy, C. Deluca, N. Dalfes, "A Scientific Workflow Environment for Earth System Related Studies", Computer and Geosciences, vol.37, no.7, pp.943-952, 2011. "doi:10.1016/j.cageo.2010.11.013"

[5] Great Lakes Coastal Forecasting System, http://www.glerl.noaa.gov/res/glcfs/, 2010

[6] L. Wang, J.F. Zhan, W.S. Shi, Y. Liang, "In Cloud, Can Scientific Communities Benefit from the Economies of Scale", IEEE Transactions on Parallel and Distributed Systems,vol.23, no.2, pp.296-303, 2012. "doi:10.1109/ TPDS.2011.144"

[7] H.Topcuoglu,S.Hariri,M.Wu, "Performance Effective and Low-complexity Task Scheduling for Heterogeneous Computing", IEEE Transactions on Parallel and Distributed Systems,vol.13, no.3, pp.260-274, 2002. "doi:10.1109/71.993206"

[8] T. Fahringer, A. Jugravu, S. Pllana, "ASKALON: A Tool Set for Cluster and Grid Computing", Concurrency and Computation: Practice and Experience, vol.17, no.2-4, pp.143-169, 2005. "doi:10.1002/cpe.v17:2/4"

[9] S.B. Zhong, H.E. Zhongshi, "Application of particle swarm optimization algorithm based on classification strategies to grid task scheduling", Journal of software, vol.7, no.1, pp.118-124, 2012. "doi:10.4304/jsw.7.1.118-124"

[10] J.Yu, R.Buya, "Scheduling Scientific Workflow Applications with Deadline and Budget Constraints Using Generic Algorithms", Scientific Programming Journal, vol.14, pp.2127-2130, 2006. "doi:10.1126/science.-1065467"

[11] Gridbus middleware, http://www.cloudbus.org, 2010

[12] Akkan C, Drexl A, Kimms A, "Network decomposition-based benchmark result s for the discrete time-cost tradeoff problem", European Journal of Operational Research, vol.165, no.2, pp.339-358, 2005. "doi:10.1016/ j.ejor.2004.04.006"

[13] Y.C. Yuan, X.P. Li, Q. Wang, X. Zhu, "Deadline division-based heuristic for cost optimization in workflow scheduling", Information Science, vol.179, no.15, pp.2562-2575, 2009. "doi:10.1016/j.ins.2009.01.035"

[14] P. Xiao, Z.G. Hu, "Two-dimension relaxed reservation policy for independent tasks in grid computing", Journal of software, vol.6, no.8, 2011, pp.1395-1402, 2011. "doi: 10.4304/jsw.6.8.1395-1402"

[15] R. Prodan, M. Wieczorek, "Negotiation-Based Scheduling of Scientific Grid Workflows Through Advance Reservations", Journal of Grid Computing, vol.8, pp.493-510, 2010. "doi:10.1007/s10723-010-9165-9"

[16] F. Dong, S.G. Akl, "PFAS: A Resource Performance Fluctuation Aware Workflow Scheduling Strategies for Grid Computing", In Proceeding of 21th international parallel and distributed processing symposium, pp.1-9, 2007. "doi:10.1109/IPDPS.2007.370328"

[17] L. Ramakrishnan, C. Koelbel, Y.S. Kee, et al, "VGrADS: enabling e-science workflows on grids and clouds with fault tolerance", In Proceedings of the conference on High Performance Computing Networking, pp.369-376, 2009. "doi:10.1145/1654059.1654107"

[18] V Berten, J Goossens, "On the Distribution of Sequential Jobs in Random Brokering for Heterogeneous Computational Grids", IEEE Transactions on Parallel and Distributed Systems, vol.17, no.2, pp.113-124, 2006. "doi:10.1109/TPDS.2006.27"

[19] P. Xiao, Z.G. Hu, "Deadline-Guarantee-Enhanced Co-allocation for Parameter Sweep Application in Grid," In Proceedings of International Conference on Communications(ICC'09), pp.1-5,2009.

[20] E. Deelman, G.Singh, M.H.Su, et al, "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed System", Scientific Programming Journal, vol.13, no.3, pp.219-237, 2005.
[21] J. Yu, R. Buyya, C. K. Tham, " Cost-based Scheduling of Scientific Workflow Applications on Utility Grids", In proceedings of the 1st International Conference on e-Science and Grid Computing, pp.140-147, 2005. "doi:10.1109/E-Science.2005.26"
[22] A. Iosup, H. Li, M. Jan, "The Grid Workloads Archive". Future Generation Computer Systems", vol.24, no.7, pp.672-686, 2008. "doi:10.1016/j.future.2008.02.003"
[23] B. Schroeder, G.A. Gibson, "The computer failure data repository", In workshop on reliability analysis of system failure data, 2007. "doi:10.1145/1188455.1188615"
[24] D. Kondo, G. Fedak, F. Cappello, A.A. Chien, "Characterizing resource availability in enterprise desktop grids", Future Generation Computer Systems", vol.24, no.7, pp.888-903, 2008. "doi:10.1016/j.future.2006.11.001"
[25] D. Kondo, B. Javadi, A. Iosup, D. Epema, "The failure trace archive: Enabling comparative analysis of failures in diverse distributed systems", In proceedings of 10th international symposium on cluster, cloud and Grid computing, 2010. "doi: 10.1109/CCGRID.2010.71"
[26] B. Schroeder, G.A. Gibson, "A large-scale study of failures in high-performance computing systems", IEEE Transactions on Dependable and Secure Computing, vol. 7, no. 4, pp.337-350, 2010. "doi:10.1109/TDSC.2009.4".
[27] A. Iosup, M. Jan, O. Sonmez, D. Epema, "On the dynamic resource availability in grids", In proceedings of 8th IEEE/ACM international conference on Grid computing, 2007. "doi: 10.1109/GRID.2007.4354112".
[28] R.K. Sahoo, A.Sivasubramaniam, M.S. Squillante, Y. Zhang, "Failure Data Analysis of A Large-Scale Heterogeneous Server Environments", In Proceedings of Dependable Systems and Networks, 2004. "10.1109/DSN.2004.1311948".
[29] J. Yu, R. Buyya, C.K. Tham, "Cost-based Scheduling of Scientific Workflow Applications on Utility Grid", In proceedings of 1st international conference on e-science and Grid computing, pp.140-147, 2005. "doi:10.1109/E-SCIENCE.2005.26".
[30] A. O'Brien, S. NewHOuse, J. Darlington, "Mapping of Scientific Workflow within the e-protein project to Distributed Resources", In Proceedings of UK e-science All Hands meetings, 2004.
[31] C. S. Yeo, R. Buyya, "Pricing for Utility-driven Resource Management and Allocation in Clusters", International Journal of High Performance Computing Applications, vol.21, no.4, pp.405-418D, 2007. "doi:10.1177 /1094342007083776"
[32] D.Gross, C.M. Harris, "Fundamentals of Queuing Theory", USA:John Wiley and Sons,1998.

Chaokun Yan was born in 1978 and received master degree in Central South University in 2004. Currently he is a Ph.D student at School of Information Science and Engineering in Central South University, Changsha, China. His research interests include Scientific Workflow, Grid Computing and Cloud Computing.

Huimin Luo was born in 1980 and received master degree in NorthWestern Polytechnical University in 2004 is a lecturer at School of Computer and Information Engineering in Henan University, Kaifeng, China. Her research interests include Scientific Workflow, Cloud Computing.

Zhigang Hu was born in 1963. He is a professor and Ph.D supervisor in School of Information Science and Engineering in Central South University, Changsha, China. Currently, he is the Chief Secretary of Computer Science in Hunan Province. His research interests include Parallel and Distributed Computing, Grid and Cloud Computing.

Xi Li was born in 1980. She received doctor degree in central south university in 2012. Currently she is a lecture in School of Information Science and Engineering in Central South University, Changsha, China. She was also visiting scholar to Chicago University from 2007 to 2008and her research interests include Scientific Workflow, Performance Evaluation, Grid Computing and Cloud Computing.

Yanping Zhang, was born in 1987 and received master degree in Central South University in 2011. Currently she is a Ph.D student at department of genome oiented bioinformatics in technical uiversity of munich, Freising, Germany. Her research interests include Scientific Computing, Computational proteomics.