

Short Paper

Survivability and Makespan Driven Scheduling Algorithm for Grid Workflow Applications*

SHU-PENG WANG, XIAO-CHUN YUN AND XIANG-ZHAN YU

Department of Computer Science and Technology

Harbin Institute of Technology

Harbin, 150001, P.R. China

With the development of Grid technologies, scientists and engineers obtain large computing capability to execute large-scale workflow applications. However the Grids also bring a troublesome problem. In the dynamic, complex and unbounded Grid systems, malicious attacks and hardware faults, which are inevitable, can have a serious effect on the execution of applications. Therefore it is necessary to enhance survivability of the applications in Grid systems. Scheduling is the first step of executing applications on Grids, and can have a significant impact on survivability of the applications. Unfortunately, most of current scheduling algorithms only address the objective of maximizing the performance of applications, but ignore survivability of applications. In this paper, we devise a survivability and makespan driven scheduling (*SMDS*) algorithm for scientific workflow applications in Grid systems. Different from the traditional scheduling algorithms, this algorithm addresses the objectives of maximizing the survivability and minimizing the makespan of workflow applications at the same time, and can trade off the survivability against the makespan of workflow applications. Many simulation experiments are taken to evaluate the performance of the *SMDS* algorithm. The results confirm that the algorithm can enhance survivability of the workflow applications, and can produce different schedules to meet different requirements by adjusting the weight parameter in its cost function.

Keywords: grid, scheduling algorithms, makespan, survivability, scientific workflow

1. INTRODUCTION

Grids [1, 2] are emerging as a global cyber-infrastructure by integrating large numbers of heterogeneous, distributed resources: computing resources, data storage systems, instruments etc. While providing large computing capability, the large-scale and highly dynamic Grid systems also bring a troublesome problem. In the complex and unbounded Grid system, malicious attacks and hardware failures, which are uncontrollable and inevitable, can result in the failure of Grid resources which can affect the execution of applications in Grid systems. Especially for the large-scale workflow applications, which need taking advantage of vast Grid resources and running for many days, failures of Grid resources may make them rarely finish. Therefore it is significant to enhance survivability of the workflow applications in Grid systems (The definition of survivability will be

Received November 1, 2005; revised April 26, 2006; accepted May 8, 2006.

Communicated by Sy-Yen Kuo.

* This paper was supported by the National Basic Research (973) Program of China (2005CB321806).

discussed in section 4 in detail). Scheduling is the first step to execute workflow applications on Grid systems, and the scheduling decisions can affect survivability of the Grid workflow applications significantly. Appropriate schedules can enhance survivability of workflow applications greatly, and decrease the probability of rescheduling that can add an extra overhead to the execution process of applications. Therefore it is necessary to address survivability of Grid workflow applications when making scheduling decisions. Unfortunately, most of current scheduling algorithms for Grid workflows ignore this problem. Initially, the performance of the workflow applications is addressed by researchers, and a number of Grid workflow scheduling algorithms are developed to minimize the makespan of workflow applications [3-8]. Recently the Qos requirements of users (*e.g.* deadline and budget) are considered, and several scheduling strategies for Grid workflows based on users' Qos requirements are proposed [9]. However, survivability of Grid workflow applications is still ignored.

In this paper, we study the approach to enhance survivability of scientific workflow applications in the scheduling process. A survivability and makespan driven scheduling algorithm (*SMDS*), which addresses survivability and makespan of workflow applications at the same time, is devised. In order to testify the efficiency of this scheduling algorithm, large numbers of simulation experiments are taken on a simulation platform that is implemented by extending SimGrid. The simulation results confirm that the algorithm can enhance survivability of workflow applications efficiently, and can produce different schedules to meet different requirements by adjusting the weight parameter in its cost function.

The rest of the paper is organized as follows. In the next section we present the related work in the realm of Grid workflow scheduling. Section 3 describes the model of systems and scientific workflow applications. Section 4 presents the definition and mathematical model for survivability of scientific workflow applications. Section 5 presents the *SMDS* algorithm for the Grid workflow applications. Section 6 shows the experiments and results. Section 7 concludes the paper with summary and future directions.

2. RELATED WORKS

In recent years, researchers devise some heuristic scheduling algorithms for Grid workflow applications. Paper [5] implements scheduling algorithms for DAG-based workflows using Min-Min, Min-Max, and Suffrage heuristics. Paper [7] promotes the use of Artificial Intelligence planning techniques and employs Greedy randomized adaptive search techniques for Grid workflows scheduling. Prodan *et al.* [10] uses classical generic algorithms with cycle elimination techniques to minimize the execution time of non-DAG based workflow on Grids. Jim Blythe devises *task-based* and *workflow-based* scheduling algorithm for Grid workflows using min-min heuristic [11]. The GridLab group [12] applies genetic algorithms, simulated annealing and Tabu search to workflow scheduling. The above schemes are all constructed to minimize the makespan or execution time of applications, thereby ignoring survivability of the workflow applications in Grids.

Now the Qos requirements of users (*e.g.* deadline and budget) are addressed by some researchers, and several Qos-based scheduling heuristics for Grid workflows have been devised. Jia Yu [9] proposes a Qos-based workflow scheduling heuristic which

minimizes the execution cost of workflow applications while meeting the deadline requirements of users. However these scheduling strategies also do not consider survivability of Grid workflow applications, so the desired Qos often can not be guaranteed in real Grid environments.

Some works on tasks scheduling in traditional distributed systems has addressed the issue of failures of computing resources. To reduce the effect of failures on applications, reliable scheduling heuristics for applications are proposed to minimize the probability of failures of applications [13]. In order to trade off the conflicting requirements between minimizing the execution time and the probability of failure of an application, list scheduling heuristics taking into account both execution time and reliability are devised [14, 15]. However, they only consider the hardware failure and ignored the malicious attacks. Also they do not consider the prediction inaccuracy in dynamic Grid systems. So these scheduling algorithms are not appropriate for open, highly dynamic, complex and heterogeneous Grid systems.

3. MODELS AND ASSUMPTIONS

The Grid system is composed of a number of non-dedicated hosts and each host is composed of a number of machines, which may be homogeneous or heterogeneous. Without the exclusive control of the local hosts, the Grid scheduler does not have control over them. The network topology of a Grid system is modeled using a connected, undirected graph $G = (M^P \cup M^U, N)$, where M^P denotes the set of computation machines, M^U denotes the set of communication machines and N denotes the set of communication links. Let $m_i \in M^P \cup M^U$ denote a machine (a computation machine or a communication machine), $m_i^P \in M^P$ denote a computation machine, $m_j^U \in M^U$ denote a communication machine, and $n_{k,l} \in N$ denote the communication link between machine m_k and machine m_l . Let $R = M^P \cup M^U \cup N$ denote the set of resources in a Grid system; an element $r_i \in R$ refers to either a machine or a network link. Let $I(m_k^P, m_l^P)$ denote the expected transmission time of sending one byte of data from machine m_k^P to m_l^P .

Considering the states of the Grid resources, the following assumptions are made. Assume resources have two states: {"normal", "failure"}. When a computation machine is in "failure" state, tasks can not execute on it; and when network links or communication machines are in "failure" state, files can not transmit through them. Hardware failures or malicious attacks all can result that resources turn into "failure" state, in which Grid resources can not provide appropriate execution environment for tasks. The Grid resources have different hardware, software and security policies, so they have different capability to withstand the malicious attacks or hardware failures, and different probability that they turn into "failure" state. Therefore allocating different resources for applications can significantly affect their survivability. Assume the failures of Grid resources caused by malicious attacks and hardware faults follow a Poisson process [16] and each resource $r_i \in R$ is accordingly associated with a constant failure rate λ_{r_i} .

A scientific workflow application is represented using a directed acyclic graph (DAG) (see Fig. 1) $T = \{V, E\}$, where $V = \{v_1, v_2, \dots, v_n\}$ is the set of tasks, and the set E of edges represents file dependencies between tasks. Let $EET(v_i, m_j^P)$ denote the estimated execution time of task v_i on machine m_j^P ($m_j^P \in M^P$). Let $e_{i,j} = \{v_i, v_j\} \in E$ indicate the data files transmitted from task v_i to v_j , and $|e_{i,j}|$ denote the volume of data files that will

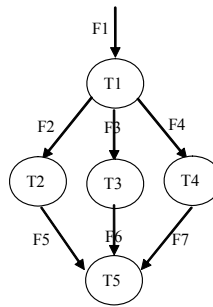


Fig. 1. A grid workflow DAG.

be transmitted from task v_i to task v_j upon completion of task v_i . When task v_i and v_j are assigned to the same computational machine, the time for transmitting data files from task v_i to v_j is 0; otherwise when task v_i and v_j are assigned to different computational machines, the time for transmitting data files from task v_i to v_j is $|e_{ij}| \times I(m(v_i), m(v_j))$, where $m(v_i)$ denotes the computation machine to which task v_i is scheduled.

The makespan of the workflow application T is defined as:

$$MK(T) = \max_{v_i \in V} \{AFT(v_i)\} \quad (1)$$

where $AFT(v_i)$ denotes actual finish time.

4. SURVIVABILITY MODEL OF GRID WORKFLOW APPLICATIONS

In this section, the definition and the mathematical model of survivability of scientific workflow applications is presented. There have been a number of different definitions on system survivability [17]. The main function of a scientific workflow application is to process the scientific data correctly and return the correct results to users. According the characteristic of scientific workflow applications, we use the definition of system survivability which is similar to that in [18].

Definition 1 System survivability is the capability of a system to provide critical services for users.

Definition 2 Survivability of scientific workflow applications is the probability that they return correct results of data processing.

For a scientific workflow application, only when all tasks in the scientific workflow execute correctly and all data files transmit between tasks normally, the correct results of data processing can be got. And according to section 3, normal execution of tasks and transmission of data files requires that the resources used by them must be in “normal” state during the time that these tasks are executing and data files are transmitting. Let $R(T, X)$ denote the set of Grid resources on which the workflow application T executes under scheduling decision X , and r_i ($r_i \in R(T, X)$) denote a Grid resource assigned to application T under scheduling decision X . And let t_{ri} denote the time during which Grid re-

source r_i is used by tasks or data files of workflow T under the schedule X , and $P(r_i, t)$ ($r_i \in R(T, X)$) denote the probability that resource r_i remains in “normal” state during time t . Since the failure of a Grid resource is governed by a Poisson process, $P(r_i, t)$ can be computed as:

$$P(r_i, t) = e^{-\lambda_{r_i} \times t}. \quad (2)$$

Then survivability of the scientific workflow application T under the schedule X can be computed as:

$$Sur(T, X) = \prod_{m_i \in R(T, X)} p(m_i, t_{m_i}) \times \prod_{n_{k,l} \in R(T, X)} p(n_{k,l}, t_{n_{k,l}}) = e^{-\sum_{r_i \in R(T, X)} \lambda_{r_i} \times t_{r_i}}. \quad (3)$$

5. SCHEDULING ALGORITHMS

5.1 Definitions of Notations

To facilitate the presentation of the algorithm, some of the notations are listed in section 3, other necessary notations and properties are listed in the following Table 1.

Table 1. Definitions of notations.

Notation	Definition
$D(v)$	The set of predecessors of task v , $D(v) = \{v_i (v_i, v) \in E\}$
$S(v)$	The set of successors of task v , $S(v) = \{v_i (v, v_i) \in E\}$
$EAT(v_i, m_j^P)$	The expected time when the computation machine m_j^P ($m_j^P \in M^P$) will be available for the execution of the task v_i ($v_i \in V$).
$FST(v_i, v_k)$	The expected time when files start transmitting from task v_i ($v_i \in D(v_k)$) to task v_k ($v_k \in V$).
$FAT(v_i, v_k, m_j^P)$	The expected time when the files from task v_k ($v_k \in D(v_i)$) will be available for the task v_i if task v_i is scheduled to the computation machine m_j^P ($m_j^P \in M^P$).
$FAT(v_i, m_j^P)$	The expected time when all the files required by task v_i will be available if task v_i ($v_i \in V$) is scheduled to computation machine m_j^P ($m_j^P \in M^P$).
$EST(v_i, m_j^P)$	The estimated time when task v_i ($v_i \in V$) start to execute on computation machine m_j^P ($m_j^P \in M^P$).
$ECT(v_i, m_j^P)$	The estimated time when task v_i ($v_i \in V$) will be completed on computation machine m_j^P ($m_j^P \in M^P$).

According to section 3, $FAT(v_i, v_k, m_j^P)$ can be computed as:

$$FAT(v_i, v_k, m_j^P) = \begin{cases} ECT(v_k, m_j^P) & \text{if } m(v_k) = m_j^P \\ FST(v_k, v_i) + |e_{k,j}| \times I(m(v_k), m_j^P) & \text{otherwise} \end{cases}. \quad (4)$$

Then $FAT(v_i, m_j^P)$ can be computed as:

$$FAT(v_i, m_j^P) = \max_{v_k \in D(v_i)} \{FAT(v_i, v_k, m_j^P)\}. \quad (5)$$

Before a task start to execute on a computation machine, all files required by it must have be received and the computation machine must be available, so $EST(v_i, m_j^P)$ can be computed as:

$$EST(v_i, m_j^P) = \max(EAT(v_i, m_j^P), FAT(v_i, m_j^P)). \quad (6)$$

Finally the estimated complete time $ECT(v_i, m_j^P)$ of task v_i can be computed as:

$$ECT(v_i, m_j^P) = EST(v_i, m_j^P) + EET(v_i, m_j^P). \quad (7)$$

5.2 Cost Functions

Scheduling of DAG-based workflow applications is a NP-complete problem, so heuristic solutions are needed by the scheduling algorithms for Grid workflow applications. The cost functions, which can guide scheduling algorithms to produce schedules, are critical for the scheduling heuristics. The Eq. (8) presents the cost function used by the makespan-driven scheduling heuristics for DAG-based workflow applications [11].

$$C_M(v_i, m_j^P) = ECT(v_i, m_j^P). \quad (8)$$

The *SMDS* scheduling algorithm accounts for the objective of minimizing the makespan as well as the objective of maximizing survivability of workflow applications. The cost function for this algorithm can be obtained by combining a new term $C_S(v_i, m_j^P)$, which must be able to guide the algorithm to enhance survivability of the applications, with the term $C_M(v_i, m_j^P)$. The approach to compute $C_S(v_i, m_j^P)$ is given in following.

Let $P(v_i, m_j^P)$ denote the probability that task v_i execute successfully when it is assigned to computation machine m_j^P , $P^E(v_i, m_j^P)$ denote the probability that machine m_j^P is in “normal” state during the time when task v_i executes on it, and $P^F(v_i, m_j^P)$ denote the probability that task v_i receives all data files required by it from its predecessors if task v_i is assigned to computation machine m_j^P . Successful execution of a task on a machine requires that all data files required by the task must have been received and the machine must be in “normal” state during the execution of the task v_i . Thus, $P(v_i, m_j^P)$ can be computed as:

$$P(v_i, m_j^P) = P^E(v_i, m_j^P) \times P^F(v_i, m_j^P). \quad (9)$$

According to Eq. (2), $P^E(v_i, m_j^P)$ can be computed as:

$$P^E(v_i, m_j^P) = e^{-\lambda_{m_j^P} \times EET(v_i, m_j^P)}. \quad (10)$$

Let $P(e_{k,i})$ denote the probability that data files $e_{k,i}$ transmits from task v_k to task v_i successfully. Then $P^F(v_i, m_j^P)$ can be computed as:

$$P^F(v_i, m_j^P) = \prod_{v_k \in D(v_i)} P(e_{k,i}). \quad (11)$$

The complexity to compute $P(e_{ij})$ is NP-hard [19]. In this paper a computational inexpensive method, which is proposed in [14], is used to compute $P(e_{ij})$. Let $H_i(m_s^P, m_t^P) = \{r^i_1, r^i_2, \dots, r^i_{li}\}$, $1 \leq i \leq Q_{s,t}$ denote the i th simple path between machines m_s^P and m_t^P , where $Q_{s,t}$ denotes the number of different simple paths between these two machines and $r^i_1 = m_s^P$ and $r^i_{li} = m_t^P$. And let $R_s(m_s^P, m_t^P)$ denote the set of resources (including communication links and machines) that are common to all the paths between machines m_s^P and m_t^P , which can be represented as $R_s(m_s^P, m_t^P) = \{r_x \mid r_x \in \bigcap_{i=1}^{Q_{s,t}} H_i(m_s^P, m_t^P)\}$. Then $P(e_{ij})$ can be computed as:

$$P(e_{i,j}) = 1 - \sum_{r_k \in R_s(m(v_i), m(v_j))} \lambda_{r_k} \times (|e_{i,j}| \times I(m(v_i), m(v_j))). \tag{12}$$

So Eq. (9) becomes:

$$P(v_i, m_j^P) = e^{-\lambda_{m_j^P} \times EET(v_i, m_j^P)} \times \prod_{v_k \in D(v_i)} \left(1 - \sum_{r_l \in R_s(m(v_k), m(v_i))} \lambda_{r_l} \times (|e_{k,i}| \times I(m(v_k), m(v_i))) \right). \tag{13}$$

When e^x is substituted for $1 + x$, Eq. (13) becomes

$$P(v_i, m_j^P) = e^{-\lambda_{m_j^P} \times EET(v_i, m_j^P)} \times \prod_{v_k \in D(v_i)} e^{-\sum_{r_l \in R_s(m(v_k), m(v_i))} \lambda_{r_l} \times (|e_{k,i}| \times I(m(v_k), m(v_i)))}. \tag{14}$$

Then $C_S(v_i, m_j^P)$ is defined as:

$$C_S(v_i, m_j^P) = \lambda_{m_j^P} \times EET(v_i, m_j^P) + \sum_{v_k \in D(v_i)} \left(\sum_{r_l \in R_s(m(v_k), m(v_i))} \lambda_{r_l} \times (|e_{k,i}| \times I(m(v_k), m(v_i))) \right). \tag{15}$$

It is clear that maximizing $P(v_i, m_j^P)$ is equivalent to minimizing $C_S(v_i, m_j^P)$. Finally, the cost function for the SMDS algorithm can be computed as:

$$C_{SM}(v_i, m_j^P) = \alpha \times (C_M(v_i, m_j^P)/MK_{\max}) + (1 - \alpha) \times (C_S(v_i, m_j^P)/Sur_{\max}) \quad (0 \leq \alpha \leq 1) \tag{16}$$

where $MK_{\max} = \max_{v_i \in V_R, m_j^P \in M^P} \{C_M(v_i, m_j^P)\}$ and $Sur_{\max} = \max_{v_i \in V_R, m_j^P \in M^P} \{C_S(v_i, m_j^P)\}$.

5.3 Algorithm Description

Fig. 2 presents the SMDS algorithm. As can be seen from Fig. 2, only the tasks which have no unscheduled predecessors are considered every time. For each task v_i in the task set V_R and each machine m_j^P in the set M^P_A of available computation machines, the cost $C_M(v_i, m_j^P)$ and $C_S(v_i, m_j^P)$ for every task-machine pair (v_i, m_j^P) is computed, and the values of MK_{\max} and Sur_{\max} are found. Then the cost $C_{SM}(v_i, m_j^P)$ for each task-machine pair (v_i, m_j^P) is computed, and the task-machine pair (v_i, m_j^P) with the minimum $C_{SM}(v_i, m_j^P)$ value is found. The task v_i with the minimum $C_{SM}(v_i, m_j^P)$ value is scheduled in next. This is repeated until all the tasks have been scheduled.

```

The SMDS Algorithm
Input:  $G = (M^B \cap M^U, N)$ ,  $T = \{V, E\}$  /* Grid System, Workflow DAG */
Output: scheduling decisions  $X$ ,  $MK(T, X)$  and  $Sur(T, X)$ 
(1)  $Q = \text{null}$ ; /* the list which saves the tasks having been assigned */
(2) While  $V \neq \emptyset$  do
(3) Select tasks that have no unscheduled predecessors from  $V$  and put them into  $V_R$ ;
/*  $V_R$  saves the tasks that have no unscheduled predecessors */
(4) Obtain the set  $M^P_A$  of available computation machines
(5) While  $V_R \neq \emptyset$  do
(6)  $MK_{\max} = -\infty$ ;  $Sur_{\max} = -\infty$ ;
(7)  $TM = \emptyset$  /*  $TM$  is the set of task-machine pairs */
(8) for each task  $v_i$  in  $V_R$  do
(9) for each computation machine  $m^p_j$  in  $M^P_A$  do
(10) Compute  $C_M(v_i, m^p_j)$  and  $C_S(v_i, m^p_j)$ ;
(11)  $MK_{\max} = \max(C_M(v_i, m^p_j), MK_{\max})$ ;
(12)  $Sur_{\max} = \max(C_S(v_i, m^p_j), Sur_{\max})$ ;
(13)  $TM = TM \cup \{(v_i, m^p_j)\}$ ;
(14) end for
(15) end for
(16) Compute the cost  $C_{SM}(v_i, m^p_j)$  for every task-machine pair  $(v_i, m^p_j)$  in  $TM$ ;
(17) Select the task-machine pair  $(v_i, m^p_j)$  with the minimum  $C_{SM}(v_i, m^p_j)$  from  $TM$ ;
(18) Assign task  $v_i$  to  $m^p_j$ ;
(19) Insert  $v$  into  $Q$ ;  $V_R \leftarrow V_R - \{v\}$ ; mark  $v$  as assigned;
(20) Update information related to computation machines and tasks;
(21) end while
(22)  $V \leftarrow V - V_R$ ;
(23) end while
(24) return  $(X, MK(T, X), Sur(T, X))$ 

```

Fig. 2. The survivability and makespan driven scheduling algorithm.

In the *SMDS* algorithm, the parameter α plays a critical role in the trade-off between the makespan and survivability of the Grid workflow applications. When $\alpha = 1$, the *SMDS* algorithm will turn into the traditional makespan-driven scheduling algorithm, and when $\alpha = 0$ the *SMDS* algorithm will turn into the scheduling algorithm which only address the objective of maximizing survivability of the application, which is referred as survivability-driven scheduling algorithm.

The time complexity of the *SMDS* algorithm is $O(|V| \times |V| \times |M^P|)$, where $|V|$ denotes the number of tasks in the set V and $|M^P|$ denote the number of machines in the set M^P . In order to assign one task, it will take $O(|V_R| \times |M^P_A|)$ time to compute $C_M(v_i, m^p_j)$ and $C_S(v_i, m^p_j)$, and determine MK_{\max} and Sur_{\max} .

6. SIMULATION EXPERIMENTS

To evaluate the performance of the proposed scheduling algorithm, we present several sets of experimental results obtained from extensive simulations in this section. In section 6.1, we describe the simulator platform, the parameters related to the simulator platform, and the performance metrics of interest. Section 6.2 presents simulation results showing the effectiveness of the *SMDS* algorithm on randomly generated workflow graphs under the assumption of prediction accuracy. In section 6.3, we show how the accuracy of the prediction affects the performance of the *SMDS* algorithm.

6.1 The Experimental Platform

The experiments are taken based on a simulation platform implemented by extending the SimGrid simulator [20]. In the simulation platform, computation machines are characterized by processing capability, utilization, and failure rate, communication resources are characterized by bandwidth, latency, and failure rate. The information about the Grid resources can be inputted into the simulation platform by configuration files.

The Grid system used in the simulation experiments is composed of 4 non-dedicated sites (see Fig. 3), which is described as follows: (1) The number of computer machines in every site is 4, 5, 6, 5 respectively, and the number of communication machines in every site is 1. (2) The computing capability in MIPS (Million Instructions Per Second) of every computation machine is uniformly distributed in the range from 500 to 2500. The scale of these ranges approximates the level of computational heterogeneity. (3) The utilization rate of computation machines and links is uniformly distributed in the range from 0.3 to 1. (4) The failure rate for each machine is chosen uniformly between 1×10^{-4} /hour and 1×10^{-3} /hour. (5) The bandwidth of links in sites is uniformly distributed in the range from 60Mbps/s to 100Mbps/s, and the bandwidth of links between sites is uniformly distributed in the range from 2Mbps/s to 10Mbps/s. The scale of these ranges approximates the level of communicational heterogeneity. (6) The failure rate of links in sites is chosen uniformly between 1×10^{-4} /hour and 1×10^{-3} /hour, and the failure rate of links between sites is chosen uniformly between 1×10^{-3} /hour and 5×10^{-3} /hour.

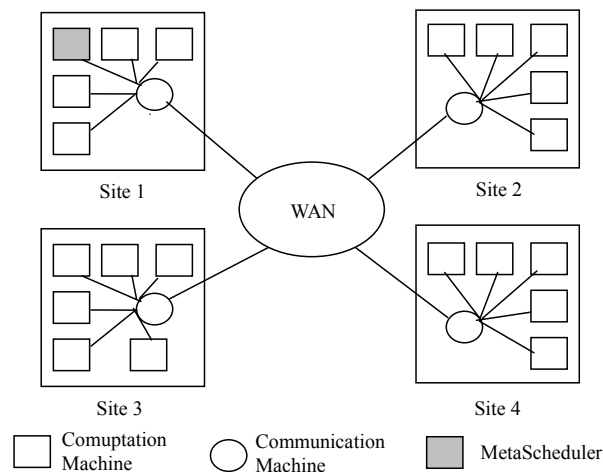


Fig. 3. The simulation grid system.

In simulation experiments, the randomly generated workflow graphs are considered. The length in MI (million Instructions) of each task in the randomly generated workflow graphs is uniformly chosen between 2×10^4 and 2×10^6 . For each workflow, the overloads of the files produced by tasks are drawn according the communication-to-computation ratio (CCR). Compute-intensive applications can be modeled by assuming $CCR = 0.1$, whereas, data-intensive applications can be modeled by assuming $CCR = 10$. The

number of tasks in workflow graphs is fixed as 100, 200, 300, 400 or 500. In order to evaluate the impact of the weight parameter α on the schedules, the performance of the *SMDS* algorithm with different α are compared. In the simulation experiments, the value of α is set as 0, 0.3, 0.5, and 1.

The performance measures in our simulation study are makespan and survivability. And all experiments data presented in this paper are the average of 100 experiment results.

6.2 Evaluation of the *SMDS* Algorithm in the Case of Prediction Accuracy

In this section, the performance of the *SMDS* algorithms is evaluated under the assumption that the performance predictions are 100% accurate. In experiments, randomly generated workflow graphs are used for the evaluation, and two scenarios of $CCR = 0.1$, and $CCR = 10$ are considered.

The simulation results for workflow applications in the case of prediction accuracy are shown in Figs. 4-5. As is shown by these figures, while the size of the workflow application increases, the makespan of the workflow application increases as expected. In parallel to this increase, survivability of the workflow application decreases. This is due to the fact that when the size of the workflow application increases, more resources will be used and will run for a longer period. As a result, the workflow applications will be more susceptible to malicious attacks and hardware failures.

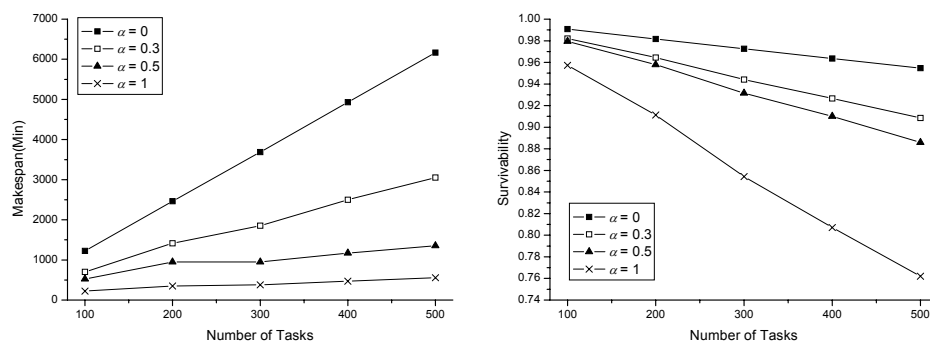


Fig. 4. Makespan and survivability of the *SMDS* algorithm for workflow applications with $CCR = 0.1$.

Fig. 4 presents the performance of the *SMDS* algorithm in the scenario of $CCR = 0.1$. According to Fig. 4, when $\alpha = 0$, the makespan is the worse, but the survivability is the optimal; when $\alpha = 1$, the makespan is the optimal, but the survivability is the worst; and with the increase of the weight parameter α , the makespan becomes better and the survivability becomes worse. And with the increase of α , the makespan is better, however the survivability is worse. The results show that the objectives of minimizing the makespan and maximizing survivability are conflict and the enhancement of survivability of applications is at the expense of increasing the makespan. And the *SMDS* can efficiently trade off the makespan against survivability of the computation-intensive workflow applications, and can produce different schedules to meet different requirements of users by

adjusting the value of the weight parameter α .

The simulation results for the data-intensive workflow applications are presented in Fig. 5. As is shown by this figure, the decrease of the weight parameter α in the cost function raises the increase of survivability of the workflow applications. The survivability of applications is the best when $\alpha = 0$, and the worst when $\alpha = 1$. This is similar to that in the case of computation-intensive workflow applications.

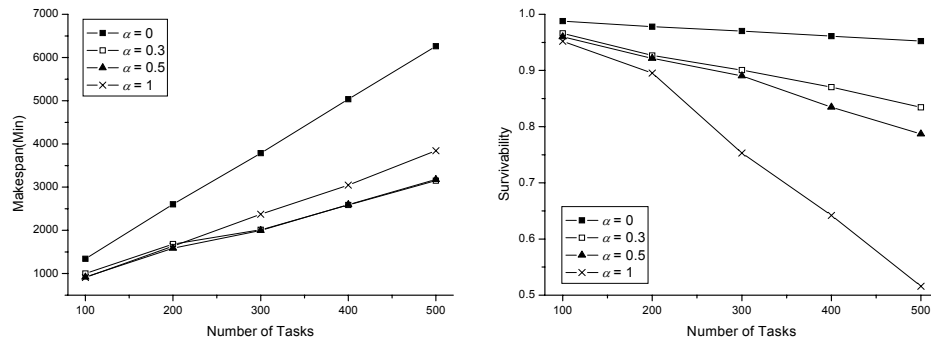


Fig. 5. Makespan and survivability of the *SMDS* algorithm for workflow applications with $CCR = 10$.

Table 2. Comparison between the scheduling results when the *SMDS* algorithm obtains the optimal makespan and that when $\alpha = 1$ for workflow applications with $CCR = 10$.

Number of Tasks	Value of α where makespan is the optimal	The optimal makespan	The corresponding survivability	Makespan ($\alpha = 1$)	Survivability ($\alpha = 1$)
100	0.9	894.7	0.96	909.2	0.95
200	0.64	1,572.5	0.92	1,635.5	0.90
300	0.44	1992	0.89	2,372.1	0.75
400	0.38	2,567.8	0.86	3,047.6	0.64
500	0.32	3,147.2	0.83	3,844.9	0.52

However, regarding to the makespan of the workflow applications, an interesting phenomenon different from that in the computation-intensive scenarios is found. That is, the makespan of the workflow application may not be the optimal when $\alpha = 1$, and the makespan when $0 < \alpha < 1$ can be better than that when $\alpha = 1$. Table 2 compares the scheduling results when the *SMDS* algorithm obtains the optimal makespan with that when $\alpha = 1$. According to this table, for data-intensive workflow applications by adjusting the weight parameter α we can obtain the optimal makespan which is better than that when $\alpha = 1$, and the survivability when makespan is the optimal is also higher than that when $\alpha = 1$. And with the increase of the scale of the workflow applications, the difference between the schedule results when the makespan is optimal and that when $\alpha = 1$ is greater. The results confirm that in the data-intensive scenarios it is necessary to addressing survivability of workflow applications when make scheduling decisions. And compared to the computation intensive applications, the influence of survivability on the communication intensive applications is greater.

The main reason for the above phenomenon is the serious impact of the large files of the data-intensive applications on the makespan-driven min-min scheduling heuristics (the *SMDS* algorithm with $\alpha = 1$) in the scenario of the data-intensive workflow applications. In general data-intensive scenarios, costly transfers of large files might be avoided by producing them where they are to be used, or at locations with a high-bandwidth connection. However, the makespan-driven min-min scheduling algorithm (the *SMDS* algorithm with $\alpha = 1$), making purely local decisions about each task, tends to distribute tasks evenly among available resources. In data-intensive scenarios, this increases the chance that a large-file will need to be transferred. So when $\alpha = 1$, the *SMDS* algorithm can not obtain the optimal makespan for the data-intensive applications in most cases. The term $C_S(v_i, m_j^P)$ can guide the *SMDS* algorithm to prevent the transfer of large files in some degree, so when the parameter value α is given a proper value between 0 and 1, the *SMDS* can obtain the optimal makespan.

From the above analysis, in the scenario of data-intensive workflow applications, the term $C_S(v_i, m_j^P)$ can not only guide the *SMDS* algorithm to enhance survivability of the workflow applications, but also guide the *SMDS* to obtain better makespan.

6.3 Impact of Inaccurate Performance Prediction

In practice, the grid environments are highly dynamic. It is difficult to obtain accurate prediction for the execution time of tasks and the transfer time of files. So the impact of the prediction inaccuracy on scheduling algorithms is worthy of investigation. In this section, the *SMDS* algorithm is evaluated under different accuracies of prediction of computation time and communication time. In experiments, the prediction error is varied within a range from 0 to 90%.

Fig. 6 presents the experiment results of the *SMDS* algorithms for compute-intensive applications under prediction inaccuracy of computation time. As can be seen from these figures, the increase of the prediction error incurs the variation of the survivability and makespan. When $\alpha = 0$, the makespan and survivability decrease with the increase of the prediction error; when $0 < \alpha \leq 1$, the makespan and survivability increase with the increase of the prediction error. In addition, the increase of α raises the decrease of makespan and survivability in the case of prediction error. The results farther confirm that the objectives of maximizing the survivability and minimizing the makespan are conflict, and the *SMDS* algorithm still can trade off these two conflict objectives under the case of prediction inaccuracy for compute-intensive applications. Moreover, it should be noted that the results show that the prediction error dose not decrease the performance of the *SMDS* algorithm with $\alpha = 0$.

Also, we take simulation experiments to test the performance of the *SMDS* algorithm for data-intensive applications under the case of prediction inaccuracy of communication time. As can be seen from Fig. 7, with the increase of the prediction error the variations of the makespan and survivability in the case of $0 < \alpha \leq 1$ is greater than that in the case of $\alpha = 0$. When the prediction error is larger than a certain value, the makespan increases greatly and survivability decreases greatly with the increase of the prediction error in the case of $0 < \alpha \leq 1$. The results show survivability obtained by the makespan-driven min-min scheduling heuristic (the *SMDS* algorithm when $\alpha = 1$) can be very low when the prediction error is large, so under such schedules the workflow applications can

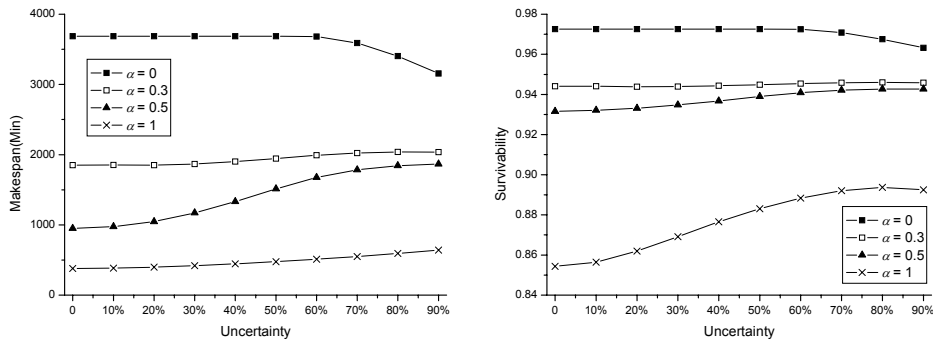


Fig. 6. Makespan and survivability for workflow applications with CCR = 0.1 in the case of uncertain compute time (number of tasks = 300).

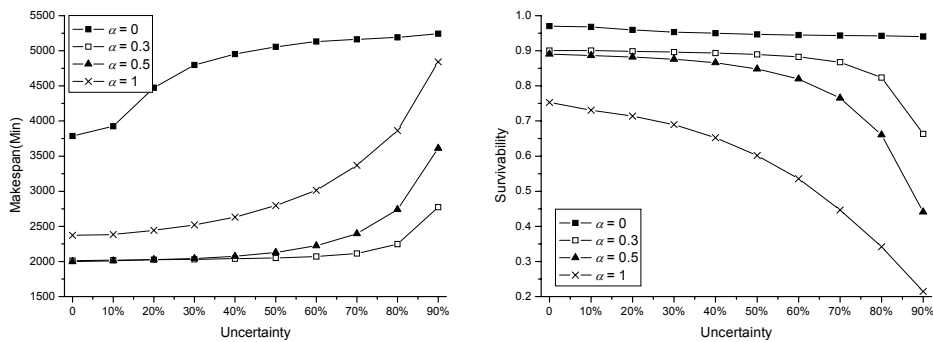


Fig. 7. Makespan and survivability for workflow applications with CCR = 10 in the case of uncertain communication time (number of tasks = 300).

not execute normally in Grids. From the results, we also can see that the survivability still follows this rule that the larger the value of α is, worse the survivability is. According to the above analysis, the *SMDS* algorithm with $\alpha = 0$ is more adapted to the data-intensive applications in the case of the prediction inaccuracy than that with $0 < \alpha \leq 1$.

7. CONCLUSIONS AND FUTURE WORKS

To confront the impact of malicious attacks and hardware failures in the complex and dynamic Grid environments, the approach to enhance survivability of scientific workflow applications in workflow scheduling process is studied. The *SMDS* algorithm which addresses the objectives of maximizing the survivability and minimizing the makespan of the workflow is devised. Many simulation experiments are taken to test the performance of the *SMDS* algorithm. The results confirm that the new term related to the survivability of the Grid workflows can guide the scheduling algorithm to make scheduling decisions that can enhance the survivability of workflow applications, and the *SMDS* algorithm can trade off these two objectives efficiently by adjusting the weight parameter α . Also, we test the performance of the algorithm under the case of prediction inaccuracy, and the results show the performance of the *SMDS* algorithm with $0 < \alpha \leq 1$ is more affected by

prediction errors than the *SMDS* algorithm with $\alpha = 0$. This further confirms that it is necessary to take account of survivability of applications in scheduling process.

Regarding to the future, there are two important issues that need to be further studied. Firstly, the approach to find an appropriate value for the parameter α needs to be studied. This study can be combined with the studies on the QoS-based scheduling. Secondly, although survivability of workflow applications can be enhanced through making appropriate scheduling decisions, the failure of applications still can not be completely prevented in the complexity and dynamic Grid environments. Therefore scheduling strategies addressing survivability of workflow applications only can be seen as the first step to enhance survivability of applications in Grid systems, other approaches to enhance survivability of applications, such as rescheduling, still need to be studied and taken to farther enhance survivability of workflow applications. In next, we will study how to address survivability of applications in the rescheduling process.

REFERENCES

1. I. Foster and C. Kesselman, *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, San Francisco, 1999.
2. I. Foster, C. Kesselman, *et al.*, "The anatomy of the grid: enabling scalable virtual organizations," *International Journal of High Performance Computing Applications*, Vol. 15, 2001, pp. 200-222.
3. T. Tannenbaum, D. Wright, K. Miller, and M. Livny, "Condor – A distributed job scheduler," *Beowulf Cluster Computing with Linux*, in T. Sterling, ed., The MIT Press, MA, U.S.A., 2002.
4. T. Fahringer, *et al.*, "ASKALON: a tool set for cluster and grid computing," *Concurrency and Computation: Practice and Experience*, Vol. 17, 2005, pp. 143-169.
5. K. Cooper, *et al.*, "New grid scheduling and rescheduling methods in the GrADS project," in *Proceedings of the 18th International Parallel and Distributed Processing Symposium*, 2004, pp. 26-30.
6. A. Birnbaum, J. Hayes, W. Li, M. Miller, P. Bourne, and H. Casanova, "Grid workflow software for high-throughput proteome annotation pipeline," in *Proceedings of the 1st International Workshop on Life Science Grid (LSGRID)*, 2004, pp. 68-81.
7. E. Deelman, *et al.*, "Mapping abstract complex workflows onto grid environments," *Journal of Grid Computing*, Vol. 1, 2003, pp. 25-39.
8. G. Singh, E. Deelman, G. Mehta, K. Vahi, M. Su, B. Berriman, J. Good, J. Jacob, D. Katz, A. Lazzarini, K. Blackburn, and S. Koranda, "The pegasus portal: web based grid computing," in *Proceedings of the 20th Annual ACM Symposium on Applied Computing*, 2005, pp. 680-686.
9. J. Yu, R. Buyya, and C. K. Tham, "QoS-based scheduling of workflow applications on service grids," Technical Report No. GRIDS-TR-2005-8, Grid Computing and Distributed Systems Laboratory, University of Melbourne, Australia, June 9, 2005.
10. R. Prodan and T. Fahringer, "Dynamic scheduling of scientific workflow applications on the grid: a case study," in *Proceedings of 20th Annual ACM Symposium on Applied Computing*, 2005, pp. 687-694.
11. J. Blythe, S. Jain, E. Deelman, Y. Gil, K. Vahi, A. Mandal, and K. Kennedy, "Task

- scheduling strategies for workflow-based applications in grids,” in *Proceedings of IEEE International Symposium on Cluster Computing and Grid*, 2005, pp. 759-767.
12. M. Mika, *et al.*, “A metaheuristic approach to scheduling workflow jobs on a grid,” in *Grid Resource Management: State of the Art and Future Trends*, Kluwer Academic Publishers, 2003.
 13. A. Dogan and F. Ozguner, “Reliable scheduling of precedence constrained tasks using a genetic algorithm,” in *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Application*, 2000, pp. 549-555.
 14. A. Dogan and F. Ozguner, “Matching and scheduling algorithms for minimizing execution time and failure probability of applications in heterogeneous computing,” *IEEE Transactions on Parallel and Distributed Systems*, Vol. 13, 2002, pp. 308-323.
 15. A. Dogan and F. Ozguner, “Biobjective scheduling algorithms for execution time-reliability trade-off in heterogeneous computing systems,” *Computer Journal*, Vol. 48, 2005, pp. 300-314.
 16. J. S. Plank and W. R. Elwasif, “Experimental assessment of workstation failures and their impact on checkpointing systems,” in *Proceedings of the 28th International Symposium on Fault-Tolerant Computing*, 1998, pp. 48-57.
 17. V. R. Westmark, “A definition for information system survivability,” in *Proceedings of the 37th Hawaii International Conference on System Sciences*, Track 9, 2004, pp. 2086-2096.
 18. S. D. Moitra and S. L. Konda, “A simulation model for managing survivability of networked information systems,” Technical Report No. CMU/SEI-2000-TR-020, Software Engineering Institute, Carnegie Mellon University, 2000.
 19. M. O. Ball, “Computational complexity of network reliability analysis: an overview,” *IEEE Transactions on Reliability*, Vol. 27, 1978, pp. 206-211.
 20. The SimGrid Project, <http://grail.sdsc.edu/projects/simgrid>.

Shu-Peng Wang (王树鹏) received the M.S. degrees in Computer Science from Harbin Institute of Technology in 2002 and 2004. From 2004 to date, he is working for his doctoral degree at the computer science department of Harbin Institute of Technology. His research interests include network security, system survivability and grid computing.

Xiao-Chun Yun (云晓春) received the B.S. and Dr. degree in Computer science from Harbin Institute of Technology (HIT) in 1993 and 1998 respectively. He has been working in HIT from 1998 and been a Professor at computer science of HIT since 2002. His research interests include computer network, network security and network simulation.

Xiang-Zhan Yu (余祥湛) received the M.S. and Dr. degree in Computer Science from Harbin Institute of Technology in 1997 and 2005 respectively. He has been working in HIT since 1997 and been an associate Professor in Computer Science of HIT since 2005. His research interests include disaster tolerance, network security.