

Method for reducing cloud workflow completion time under the task interruption

Yawen WANG, Yunfei GUO, Wenyan LIU, Shumin HUO

National Digital Switching System Engineering and Technology Research Center, Zhengzhou, China

Abstract: As more and more large-scale scientific workflows are delivered to clouds, the business model of workflow-as-a-service is emerging. But there are many kinds of threats in the cloud environment, which can interrupt the task execution and extend the workflow completion time. As an important QoS parameter, the workflow completion time is determined by the critical task path. Therefore, critical path redundancy method is proposed to create a redundant path having the interact parallel relationship with the critical path, which can provide the protection for the tasks in the critical path and reduce the probability of the critical path interruption. Computing instance allocation is an essential part of the cloud workflow execution, since only the tasks assigned the instance can begin execution. In order to further reduce the workflow completion time, computing instance allocation algorithm based on HEFT (heterogeneous earliest finish time) is proposed. The algorithm considers diverse task dependency relationships and takes full advantages of the critical path redundancy method, which can improve the efficiency of workflow execution. Experimental results demonstrate that the proposed method can effectively reduce the cloud workflow completion time under the task interruption.

Key words: cloud workflow, task interruption, resource allocation, fault tolerance

doi: 10.11959/j.issn.2096-109x.2020018

1 Introduction

In many scientific research fields, such as quantum physics, astronomy, bioinformatics and so on, the process of scientific computing often consists of tens of thousands of steps, requiring large-scale data analysis and processing. In order to properly manage, arrange, execute and track these steps, the concept of scientific workflows is presented^[1-2]. As the improvement of the complexity of the scientific problems to be solved, the present large scientific workflows often need to be implemented on complex distributed computing envi-

ronments, such as supercomputers, distributed cluster systems and grid systems, but constructing such a system requires extraordinary expensive costs.

With the development of virtualization technologies, the emergence of cloud computing provides a new deployment and implementation solution for scientific workflow applications. The cloud computing takes the resource renting, application hosting and service outsourcing as the core, builds the computing environments through the integration of distributed resources to meet a variety of service requirements. As cloud customers, they no longer need hardware purchase and software deployment,

Received data: 2019-05-29; **Revised data:** 2019-08-09

Corresponding author: Yawen Wang, 15738321455@163.com

Foundation Items: The National Key R & D Program of China (2018YFB0804004), The Foundation for Innovative Research Groups of the National Natural Science Foundation of China (61521003)

they just need to pay some fees to easily access the required computing and storage resources through the internet. Due to low costs and the convenient resource access, more and more large-scale scientific workflows are delivered to clouds to complete.

In the process of scientific workflow execution, resource failures or malicious attacks will lead to the workflow interruption. To avoid such problems, Wang et al^[3] propose to implement checkpoint backup of the workflow by storing the workflow intermediate data. When the workflow is interrupted, the abnormal state can be fixed by backtracking to the most recent checkpoint. However, checkpoint backtracking will generate time overhead, resulting in a delay in workflow completion time. For this problem, a method to reduce the cloud workflow completion time under the task interruption is proposed. The main contributions of this paper are listed as follows.

1) A threat model for cloud workflows is proposed. In the model, tasks can be interrupted due to malicious attacks or resource failures, delaying the workflow completion. And the delay effects are related to the length of the path where the interrupted task is located, the longer the path, the greater the delay effects.

2) Critical path redundancy method is proposed, which creates a redundant path having the interact parallel relationship with the critical path to reduce the interruption probability of the critical path in the workflow.

3) A computing instance allocation algorithm based on HEFT algorithm is proposed, the algorithm takes into account diverse task dependency relationships and takes full advantages of the critical path redundancy method, which can improve the efficiency of workflow execution.

2 Related work

The workflow completion time is an important QoS parameter, since the consumers always specify

deadlines when submitting the workflow execution requests to clouds. Therefore, many scholars have studied how to ensure that the workflow can be completed before the deadline. Abrishami et al^[4] design a two-phase scheduling algorithm called partial critical paths to minimize the cost of workflow execution while meeting a user-defined deadline. Similarly, Wu et al^[5] present a two-step list scheduling method, which will rank tasks firstly and allocate each task a service which meets the sub-deadline and minimizes the cost. Calheiros et al^[6] use the available spare time and budget surpluses between leased cloud computing resources to replicate tasks, thereby ensuring that tasks are completed before deadlines. Considering the execution costs and transmission costs comprehensively, on the basis of genetic algorithm, Verma et al^[7] present a deadline constrained scheduling method, which can minimize the cost of the workflow execution while meeting the deadline for delivering the result. Currently, multi-core processors are widely used in clouds, for this reason, Deldari et al^[8] propose a deadline-constrained workflow scheduling algorithm for multicore resources in the cloud. Lin et al^[9] propose an online-scheduling strategy for continuous submitted scientific workflows on hybrid clouds, aiming to complete the deadline-constrained applications as fast as possible at a lower price. Zheng et al^[10] combine the DVFS (dynamic voltage and frequency scaling) technique and heuristic algorithm to reduce overall energy consumption of a workflow given an allocation of tasks to hosts and a deadline to complete the execution. Sahni et al^[11] exploit the advantages offered by cloud computing while taking into account the virtual machine performance variability and instance acquisition delay to identify a just-in-time schedule of a deadline constrained scientific workflow at lesser costs. However, the above literatures do not consider the situation where the tasks are abnormally interrupted. The multi-tenant coexistence service mode of the

cloud platform can easily introduce malicious attacks. Consequently, the task interruption during the workflow execution should not be ignored.

In order to avoid the task interruption during the workflow execution, Yao et al^[12] consider that the process of rescheduling in cloud systems have great similarities to the immune system which kept the body stable by removing the intrusive antigens, so an immune system inspired failure-aware rescheduling algorithm for the workflow task in clouds is designed to improve the reliability of the workflow. Task redundancy method is often used for improving the workflow reliability. Ding et al^[13] propose a primary-backup workflow scheduling strategy, which copies the same task into two replicas and schedules the two replicas to different computing instances. In reference[14], the authors propose cloud scientific workflow execution system based on mimic defense technology, in which each task is executed by three computing instances. Considering the cost and reliability together, Poola et al^[15-16] propose to use spot instances to reduce the costs of the task redundancy method. In reference[17], the authors combine rescheduling algorithm and task redundancy together to design a fault-tolerant workflow scheduling algorithm. From the above literatures, it can be found that rescheduling and task redundancy are two common methods to improve the workflow reliability. However, if the task is interrupted, rescheduling will take some time to restore the workflow state, and the time spent will delay the workflow completion. Task redundancy can seamlessly restores abnormal task states, but if each task is replicated into multiple copies and dispatched to different computing instances, it will consume a lot of resources. In reference[18], the authors propose that redundancy method should be used for protecting the important components. Based on this idea, it is proposed to use task redundancy method to protect the important tasks in the workflow.

3 The workflow model

3.1 The analysis of the workflow completion time

A workflow can be described by a directed acyclic graph (DAG), in which each task is represented by a node and each data dependency between tasks is represented by a directed edge between the corresponding nodes. There is a workflow as shown in Figure 1, it can be described by $G=(V,E)$, where $V=\{v_1, v_2, \dots, v_6\}$ represents the task set, each task will be executed in a computing instance generated from the computing resource pool and the corresponding task execution time is $T=\{t_1, t_2, \dots, t_6\}$, each instance can not execute multiple tasks at the same time. The $E=\{e_{1,2}, \dots, e_{5,6}\}$ represents dependent data between tasks, using B to represent the transmission bandwidth between the computing instances, the data transfer time $t_{i,j}$ between tasks v_i and v_j can be calculated by (1).

$$t_{i,j} = \begin{cases} 0 & , \text{if } v_i \text{ and } v_j \text{ belong to the same instance} \\ \frac{e_{i,j}}{B}, & \text{else} \end{cases} \quad (1)$$

Some tasks in the workflow have multiple predecessor tasks like v_5 , they begin to be executed only when all their predecessor tasks have been finished. So the completion time t_G of the workflow G can be represented by (2).

$$\begin{aligned} t_a &= t_1 + t_{1,2} + t_2 + t_{2,3} + t_3 + t_{3,6} \\ t_b &= \max(t_1 + t_{1,2} + t_2 + t_{2,5}, t_1 + t_{1,4} + t_4 + t_{4,5}) + t_5 + t_{5,6} \\ t_G &= \max(t_a, t_b) + t_6 = \max(t_a + t_6, t_b + t_6) \end{aligned} \quad (2)$$

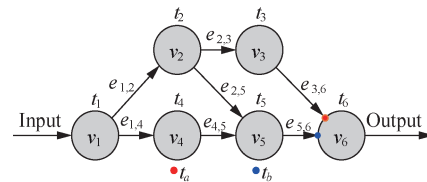


Figure 1 A sample workflow

3.2 The definition of the path length in the workflow

A workflow can be divided into multiple paths, as shown in Figure 2. Given a path p_k in which the task set is represented by $V_{p_k} = \{v_1, v_2, \dots, v_m\}$, the corresponding task execution time is represented by $T_{p_k} = \{t_1, t_2, \dots, t_m\}$, the amount of dependent data is denoted by $E_{p_k} = \{e_{1,2}, e_{2,3}, \dots, e_{m-1,m}\}$ and the transmission bandwidth between instances is B . Then we define the path length l_{p_k} by (3).

$$l_{p_k} = \sum_{i=1}^m t_i + \sum_{j=1}^{m-1} t_{j,j+1} \quad (3)$$

$$t_{j,j+1} = \begin{cases} 0, & \text{if } v_j \text{ and } v_{j+1} \text{ belong to the same instance} \\ \frac{e_{j,j+1}}{B}, & \text{else} \end{cases}$$

After the definition of the path length, the completion time t_G of the workflow shown in Figure 1 can be denoted by (4).

$$\begin{aligned} t_a &= l_{p_1} - t_6 \\ t_b &= \max(l_{p_2} - t_6, l_{p_3} - t_6) \\ t_G &= \max(t_a, t_b) + t_6 = \max(l_{p_1}, l_{p_2}, l_{p_3}) \end{aligned} \quad (4)$$

Therefore, the workflow completion time is decided by the longest path in the workflow, the longest path is called the critical path in the workflow.

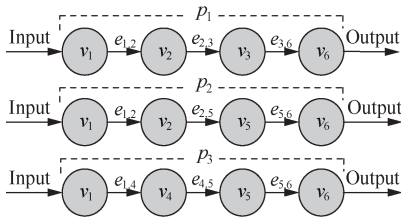


Figure 2 The path division in the workflow

3.3 The identification of the critical path in the workflow

There are many kinds of computing instances on the cloud platform, different kinds of computing instances have different computing ability, so if the same task is assigned to different computing in-

stances for execution, the required execution time will be different. Therefore, in practical applications, the path length can not be used directly to identify the critical path since each task execution time is uncertain before computing instance allocation. For this problem, an algorithm proposed by Topcuoglu^[19] to achieve the critical path identification is adopted.

In this algorithm, all tasks in the workflow need to be ordered based on upward ranking $rank_u$ and downward ranking $rank_d$. The $rank_u$ of a task v_i is recursively defined by

$$rank_u(v_i) = \bar{w}_i + \max_{v_j \in \text{succ}(v_i)} (\bar{c}_{i,j} + rank_u(v_j)) \quad (5)$$

where $\text{succ}(v_i)$ is the set of immediate successors of task v_i , $\bar{c}_{i,j}$ is the average communication time of edge (i, j) and $\bar{w}_i$ is the average execution time of task v_i . The calculation of upward rank should start from the last task in the workflow and traverse the task graph upward. For the last task v_{last} , the value of $rank_u$ is

$$rank_u(v_{\text{last}}) = \bar{w}_{\text{last}} \quad (6)$$

the $rank_d$ of a task v_i is defined as

$$rank_d(v_i) = \max_{v_j \in \text{pred}(v_i)} (rank_d(v_j) + \bar{w}_j + \bar{c}_{j,i}) \quad (7)$$

The calculation of $rank_d$ should start from the initial task in the workflow. For the initial task, the $rank_d$ value is 0. After calculating upward and downward rank for each task in the workflow, $rank_{u+d}$ is defined as

$$rank_{u+d}(v_i) = rank_u(v_i) + rank_d(v_i) \quad (8)$$

The tasks having the largest $rank_{u+d}$ constitute the critical path in the workflow.

4 The threat model for the workflow

In the cloud computing system, many methods can be used to break the execution of cloud workflows. For example, adversaries can bypass logical isolation between VMs via side channels to attack the VMs performing the workflow^[20]. A kind of vulnerabilities called VM Escape can be employed to help attackers access privilege domain of

Hypervisor, then all the VMs supported by this Hypervisor^[21] will be compromised easily. A malicious user can perform guest denial of service attack by starving there sources of the server, the VM applications running on the same server will be slowed down or even interrupted^[22]. Furthermore, there are some attacks targeting hardware of the cloud^[23], if the hardware of the physical machine on which the cloud workflow is executed fails, the cloud workflow will inevitably be interrupted.

If an interruption occurs when a task v_k is being executed, extra time t_E will be needed to restore v_k , so the time spent on v_k will increase by t_E . It is assumed that v_k will be interrupted with the probability of p , t_k denotes the execution time of v_k , so the actual time t_k^a spent on v_k can be denoted by (9).

$$t_k^a = \begin{cases} t_k + t_E, & \text{the probability of } p \\ t_k, & \text{the probability of } (1-p) \end{cases} \quad (9)$$

Although the interruption of the task in the workflow will generate extra task execution time, the delay effect on the workflow completion caused by the task interruption is uncertain. The workflow shown in Figure 1 is used as an example. If the interruption occurs in the process of v_3 execution, the workflow completion time will change to (10).

$$t_G^{v_3} = \max(t_a + t_E, t_b) + t_6 = \max(l_{p_1} + t_E, l_{p_2}, l_{p_3}) \quad (10)$$

Then (11) is used to show the relationship between $t_G^{v_3}$ and the length of the path where v_3 is located.

$$t_G^{v_3} = \begin{cases} t_E + M, & \text{if } l_{p_1} = M \\ t_E + l_{p_1}, & \text{if } M - t_E \leq l_{p_1} < M \\ M, & \text{if } l_{p_1} < M - t_E \end{cases} \quad (11)$$

$$M = \max(l_{p_1}, l_{p_2}, l_{p_3})$$

It demonstrates that the workflow completion time under the task interruption is related to the length of the path in which the interrupted task is located. The length of the path where the interrupted

task is located longer, the delay effect is greater.

5 The proposed method

From (11) it can be found that there are two ways which can reduce the workflow completion time under the task interruption. One way is reducing the probability of the interruption of the tasks on the longest path. The other is reducing the length of the longest path in the workflow, i.e. M in (11).

In this section, firstly the critical path redundancy method will be presented, which can reduce the probability of task interruption occurring on the critical path. Then computing instance allocation algorithm is proposed to improve the overall workflow execution efficiency, reducing M in (11).

5.1 Critical path redundancy method

It has been discussed that reducing the interruption probability of the tasks on the critical path is a way to shorten the workflow completion time under the task interruption. Based on this idea, critical path redundancy method is proposed to provide the protection for the tasks in the critical path.

In this method, the redundant replicas will be built for each task in the critical path, all the redundant replicas will generate a new path called redundant path, the path combined the critical path and the redundant path is called the enhanced path. After building the redundant path, the relationship between the critical path and the redundant path should be determined. There are two types of relationships, independent parallel relationship and interactive parallel relationship, as shown respectively in Figure 3 (a) and (b), v'_k denotes the redundant task of v_k . It is assumed that the probability p of each task being interrupted is the same. In the independent parallel relationship, the enhanced path will be interrupted when at least one task interruption occur in the critical path and all the redundant paths, the interruption probability p_{ind} is denoted by (12).

$$p_{ind} = [1 - (1-p)^n]^k \quad (12)$$

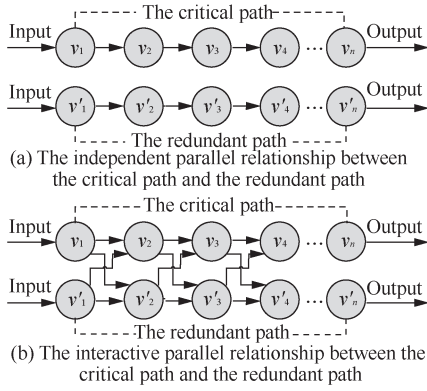


Figure 3 The independent parallel relationship and the interactive parallel relationship between the critical path and the redundant path

where n denotes the number of tasks included in the critical path, k denotes the number of redundant replicas for each task. In the interactive parallel relationship, v_2 will begin to be executed if v_1 or v'_1 has been finished. So the enhanced path interruption will occur when the same task in all the paths have been interrupted, interruption probability p_{int} is denoted by (13).

$$p_{\text{int}} = 1 - (1 - p^k)^n \quad (13)$$

The curves of p_{ind} and p_{int} were plotted with n changes when $k = 2$, as shown in Figure 4. From Figure 4 it can be found that compared with the independent parallel relationship, the interactive parallel relationship can provide higher security. Therefore, in the critical path redundancy method, we choose to build the interactive parallel relationship between the critical path and the redundant path, as shown in Figure 5.

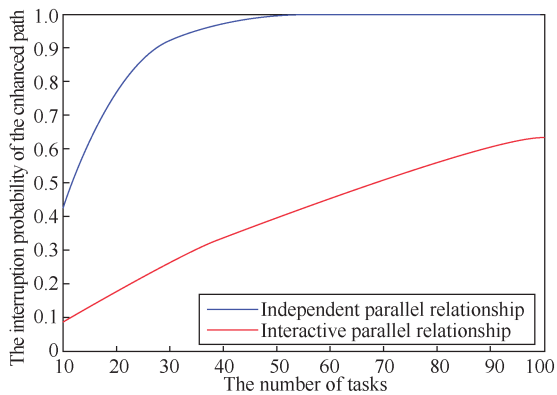


Figure 4 The security comparison between the independent parallel relationship and the interactive parallel relationship

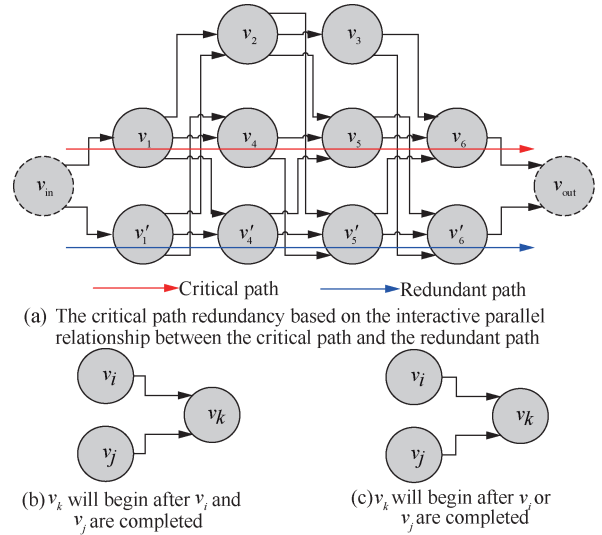


Figure 5 The illustration of critical path redundancy method

5.2 Computing instance allocation algorithm

In the workflow, only the tasks that have been assigned computing instances from the cloud platform can be executed, so computing instance allocation is an essential part of cloud workflow execution.

It has been discussed that another way to reduce the workflow completion time under the task interruption is shortening the length of the longest path in the workflow, i.e. M in (11), which can be realized by designing reasonable computing instance allocation strategy, since the interdependent tasks in the same instance do not need to transfer dependent data through network, which can save the time spent on the data transmission.

At the stage of computing instance allocation, a computing instance allocation algorithm is proposed. The proposed algorithm is improved from HEFT algorithm^[19] which has proven to be one of the most effective task scheduling algorithms in distributed environments.

In HEFT algorithm, two important attributes are defined, which are EST (earliest execution start time) and EFT (earliest execution finish time). $EST(v_i, p_x)$ and $EFT(v_i, p_x)$ represent earliest execution start time and earliest execution finish time of task v_i on computing instance p_x , respectively^[19]. For the initial task v_{initial} , $EST(v_{\text{initial}}, p_x) = 0$. For the rest tasks in the workflow, EFT and EST

values are calculated recursively, starting from the initial task, as shown in (14) and (15), respectively.

$$EST(v_i, p_x) = \max\{avail[x], \max_{v_m \in \text{pred}(v_i)} \{AFT(v_m) + e_{m,i}\}\} \quad (14)$$

$$EFT(v_i, p_x) = w_{i,x} + EST(v_i, p_x) \quad (15)$$

where $avail[x]$ denotes the earliest time when computing instance p_x is ready for task execution, $\text{pred}(v_i)$ denotes the set of immediate predecessor tasks of task v_i . $e_{m,i}$ denotes the communication time between v_m and v_i , if v_m and v_i are scheduled on the same computing instance, $e_{m,i}$ is 0. $w_{i,x}$ represents estimated execution time to complete task v_i on computing instance p_x . $AFT(v_m)$ is the actual finish time of task v_m , after v_m has been allocated a computing instance p_x , the earliest finish time will be the actual finish time. Therefore, in order to calculate EFT value of task v_i , all immediate predecessor tasks of v_i must have been allocated computing instances.

HEFT algorithm only considers the situation where task v_i can start to be executed only if all immediate predecessor tasks of v_i have been finished. However, from Figure 5 it can be found that there are two kinds of relationship between dependent tasks. In Figure 5 (b), task v_k will begin to be executed after v_i and v_j are completed, so $EST(v_k, p_x)$ can be calculated by (16).

$$EST(v_k, p_x) = \max\{avail[x], \max\{AFT(v_i) + e_{i,k}, AFT(v_j) + e_{j,k}\}\} \quad (16)$$

In Figure 5 (c), task v_k will begin to be executed after v_i or v_j are completed, so $EST(v_k, p_x)$ can be calculated by (17).

$$EST(v_k, p_x) = \max\{avail[x], \min\{AFT(v_i) + e_{i,k}, AFT(v_j) + e_{j,k}\}\} \quad (17)$$

Therefore, in the proposed computing instance allocation algorithm, before allocating a computing instance for task v_k , all immediate predecessor

tasks of v_k should be classified first. The immediate predecessor tasks with the relationship of “AND” and “OR” will be represented by $\text{pred_and}(v_k)$ and $\text{pred_or}(v_k)$, respectively. Then (14) can be transformed into (18).

$$EST(v_k, p_x) = \max\left\{avail[x], \max\left\{\max_{v_m \in \text{pred_and}(v_k)} \{AFT(v_m) + e_{m,k}\}, \min_{v_n \in \text{pred_or}(v_k)} \{AFT(v_n) + e_{n,k}\}\right\}\right\} \quad (18)$$

The proposed computing instance allocation algorithm is shown in Algorithm 1.

Algorithm 1 Computing instance allocation algorithm

- 1) Calculating $rank_u$ for all tasks;
- 2) Sorting the tasks in a scheduling list based on $rank_u$ values;
- 3) while there are unscheduled tasks in the list
- 4) Choosing the first task v_i from the scheduling list;
- 5) for each computing instance p_k
- 6) Calculating $EST(v_i, p_k)$ by (18);
- 7) Calculating $EFT(v_i, p_k)$ by (15);
- 8) end
- 9) Assigning task v_i to the computing instance p_x which minimizes EFT of task v_i
- 10) end

5.3 The procedure of the proposed method

In this section, a workflow shown is used in Figure 1 to illustrate the procedure of the proposed method. It is assumed that there are 3 computing instances, the time required for each task execution on different computing instances is shown in Table 1. The communication time of each edge is shown in Table 2.

Table 1 The time required for each task execution on different computing instances

computing instances	Tasks					
	v_1	v_2	v_3	v_4	v_5	v_6
p_1	8	7	9	10	6	8
p_2	9	8	10	11	7	9
p_3	10	9	11	12	8	10

Table 2 The communication time of each edge in the workflow

Edge	$e_{1,2}$	$e_{1,4}$	$e_{2,3}$	$e_{2,5}$	$e_{3,6}$	$e_{5,6}$
Time	1	3	1	2	2	1

1) Calculating rank_u , rank_d and rank_{u+d} for each task in the workflow, the results are shown in Table 3. According to rank_{u+d} values, v_1 , v_4 , v_5 and v_6 constitute the critical path.

Table 3 rank_u , rank_d and rank_{u+d} value of each task in the workflow

v_i	$\text{rank}_u(v_i)$	$\text{rank}_d(v_i)$	$\text{rank}_{u+d}(v_i)$
v_1	43	0	43
v_2	30	10	40
v_3	21	19	40
v_4	31	12	43
v_5	17	26	43
v_6	9	34	43

2) Employing the critical path redundancy method to build a redundant path, as shown in Figure 5. The execution time of v_{in} and v_{out} are 0.

3) Recalculating rank_u for each task in the workflow. Then sorting the tasks in a scheduling list by the order of rank_u values, it is $\{v_1, v'_1\} \rightarrow \{v_4, v'_4\} \rightarrow v_2 \rightarrow v_3 \rightarrow \{v_5, v'_5\} \rightarrow \{v_6, v'_6\}$. $\text{rank}_u(v_i)$ is equal to $\text{rank}_u(v'_i)$, so the order of v_i and v'_i doesn't matter.

4) Performing computing instance allocation algorithm.

6 Experiment

In this section, three experiments are designed.

① Evaluating the workflow completion time of the critical path redundancy method. ② Evaluating the workflow completion time of the proposed computing instance algorithm. ③ Evaluating the resource requirements of the proposed method.

The experiments are conducted with the Workflow Sim toolkit^[24], which is an open source

software designed for workflow scheduling simulation in clouds. In this software, workflows are described in XML language, which contains the information about tasks and links. The proposed method in this paper mainly includes two steps: the critical path redundancy and computing instance allocation. Similarly, critical path redundancy module is constructed and computing instance allocation module. Critical path redundancy module firstly reads in the original XML file representing workflows. Afterwards it will find the tasks contained in the critical path and build a new redundant path by the proposed method. Then the module will output the new generated workflow in the format of XML. After that computing instance allocation module will be read in the new generated XML file, then it will perform the proposed computing instance allocation algorithm and output the results.

The workflows used for experiments are CyberShake, Epigenomics, Montage and Inspiral, which are published by Pegasus project^[25-26] and widely used to measure the performance of scheduling algorithms. For each type of workflow, four kinds of DAG with different number of jobs and edges are introduced. For example, CyberShake workflow has 4 kinds with 30, 50, 100 and 1 000 tasks, respectively. In this experiment, each kinds of workflow with maximum tasks is used for evaluation. The DAG characteristics of these workflows, including the number of nodes and edges, average data size and average task runtime are given in Table 4, the structures of them are shown in Figure 6.

6.1 Evaluating the workflow completion time of the critical path redundancy method

In this experiment, the number of available computing instances is 120, the computing ability of each instance is randomly generated from 800 to 1 200 MIPS. It is supposed that each task in the workflow has the same probability of interruption and the time of task interruption is equal to the

Table 4 The parameters of workflows used for experiments^[12]

Workflow	Number of tasks	Number of edges	Average data size/MB	Average task runtime/s
CyberShake(1 000)	1 000	3 988	102.29	22.71
Epigenomics(997)	997	3 228	388.59	3 858.67
Montage(1 000)	1 000	4 485	3.21	11.36
Inspirai(1 000)	1 000	3 246	8.90	227.25

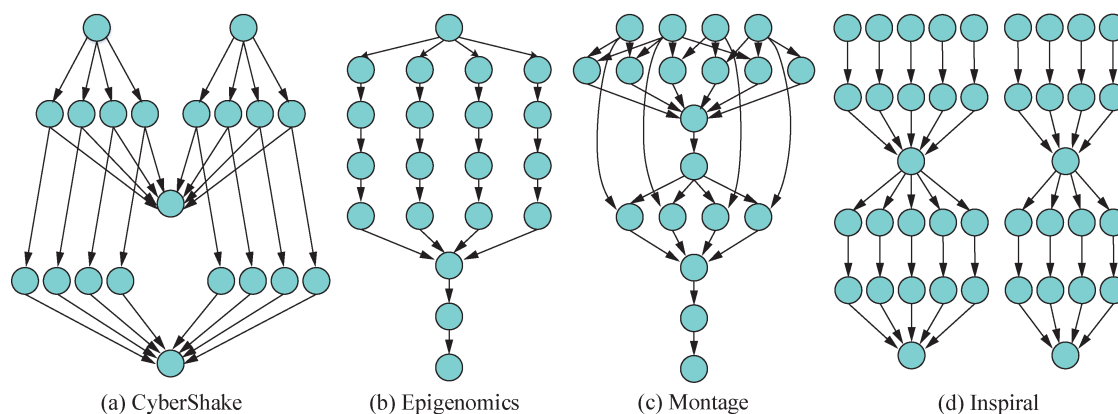


Figure 6 The structures of workflows used for experiments

average task runtime. The probability of task interruption is regarded as the variant to record the average workflow completion time. For comparison, 3 types of workflows are created, which are the original workflow, the workflow with non-critical path redundancy and the workflow with critical path redundancy. The original workflow adopts HEFT algorithm to allocate computing instances and the others employ the proposed computing instance allocation algorithm. The evaluation results are shown in Figure 7. From the figure it can be found that the starting and ending points of the three curves are roughly the same. The starting point represents that the tasks will never be interrupted and the ending point represents that the task must be interrupted. At these points, the length of the critical path in these workflows are the same, so the three curves share the same starting and ending points. Between the starting point and the ending point, the average completion time of path redundancy method is shorter than the original workflow, since the redundant path can reduce the

impact of task interruption on workflow completion time to some extent. Compared with the non-critical path redundancy, critical path redundancy method has the better result, because the workflow completion time is decided by the length of the critical path in the workflow, the protection for the critical path will bring the greatest security gain.

6.2 Evaluating the workflow completion time of the proposed computing instance allocation algorithm

In this experiment, the number of available computing instances is 120, the computing ability of each instance is randomly generated from 800 to 1 200 MIPS. For comparison, the proposed algorithm, HEFT and CPOP^[19] algorithms are used for computing instance allocation for the workflow with critical path redundancy, respectively. We record the average workflow completion time for each computing instance allocation algorithm under 3 task interruption probabilities (0, 0.2 and 0.4), the results are shown in Figure 8. From the figure it can be found that the

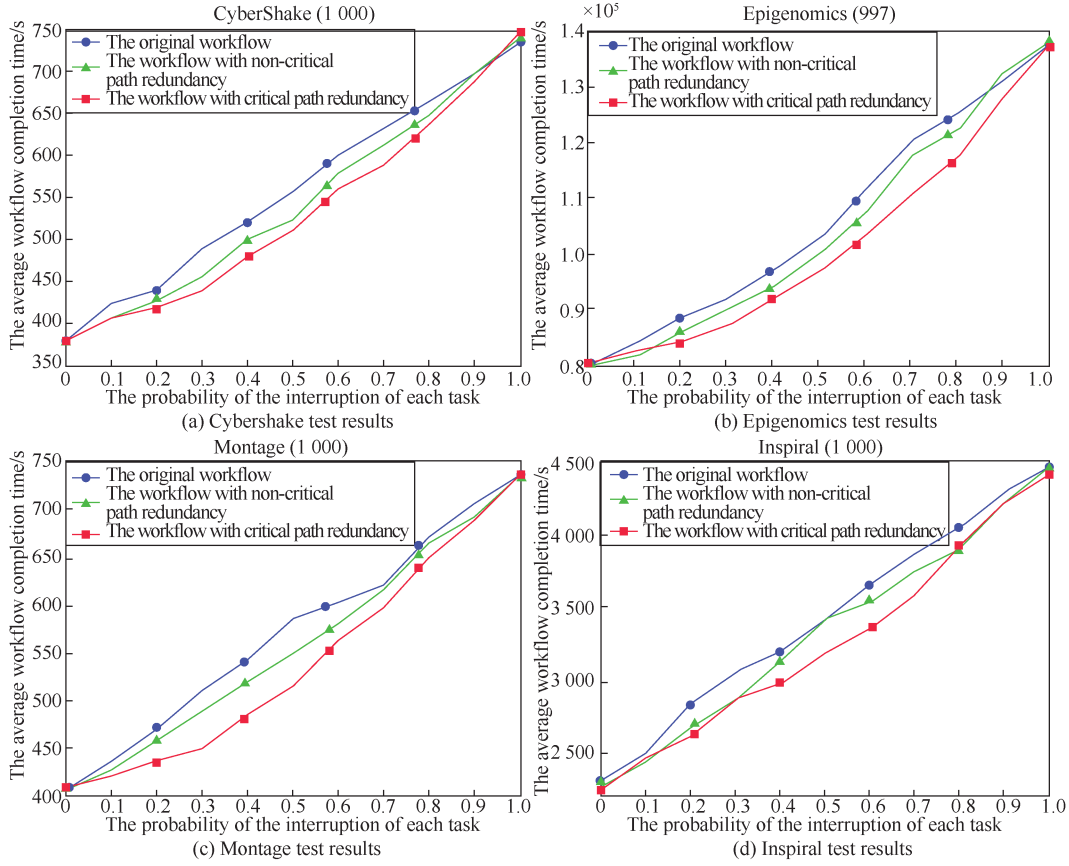


Figure 7 The evaluation results of the critical path redundancy method

proposed algorithm has the best effect, since diverse task dependencies are not considered in HEFT and CPOP algorithms. Using $t_{\text{start}}(v_i)$ to denote the time when task v_i start to be executed, $t_{\text{finish}}(v_i)$ denotes the time when task v_i is finished, $\text{pred}(v_i)$ represents all the immediate predecessor tasks of v_i . In these algorithms, the tasks can only be executed when all their predecessor tasks have been completed, so $t_{\text{start}}(v_i) > \max\{t_{\text{finish}}(\text{pred}(v_i))\}$. However, in the critical path redundancy method, the tasks in the critical path and the tasks in redundant path belong to backup relationships, so if a task has multiple predecessor tasks belonging to backup relationships, the task can start to be executed when one of predecessor tasks has been finished. The proposed computing instance allocation algorithm takes into account the backup relationship between tasks, so it can effectively improve the workflow execution efficiency.

6.3 Evaluating the resource requirements of the proposed method

In this experiment, the task interruption probability is set to 0.2, the number of available computing instances is variable (60, 80, 100 and 120). Average completion time of the original workflow and the workflow with critical path redundancy were recorded, the results are shown in Figure 9. From the figure it can be seen that increasing the number of available computing instances can effectively improve the efficiency of workflow execution, it is because there are two factors that determine the start time $t_{\text{start}}(v_i)$ of a task v_i , one is the longest completion time t_{longest} of all predecessor tasks of v_i , the other is the time when the resource reaches an idle state t_{idle} . $t_{\text{start}}(v_i) > \max\{t_{\text{longest}}, t_{\text{idle}}\}$. With the increase of the number of available computing instances, there

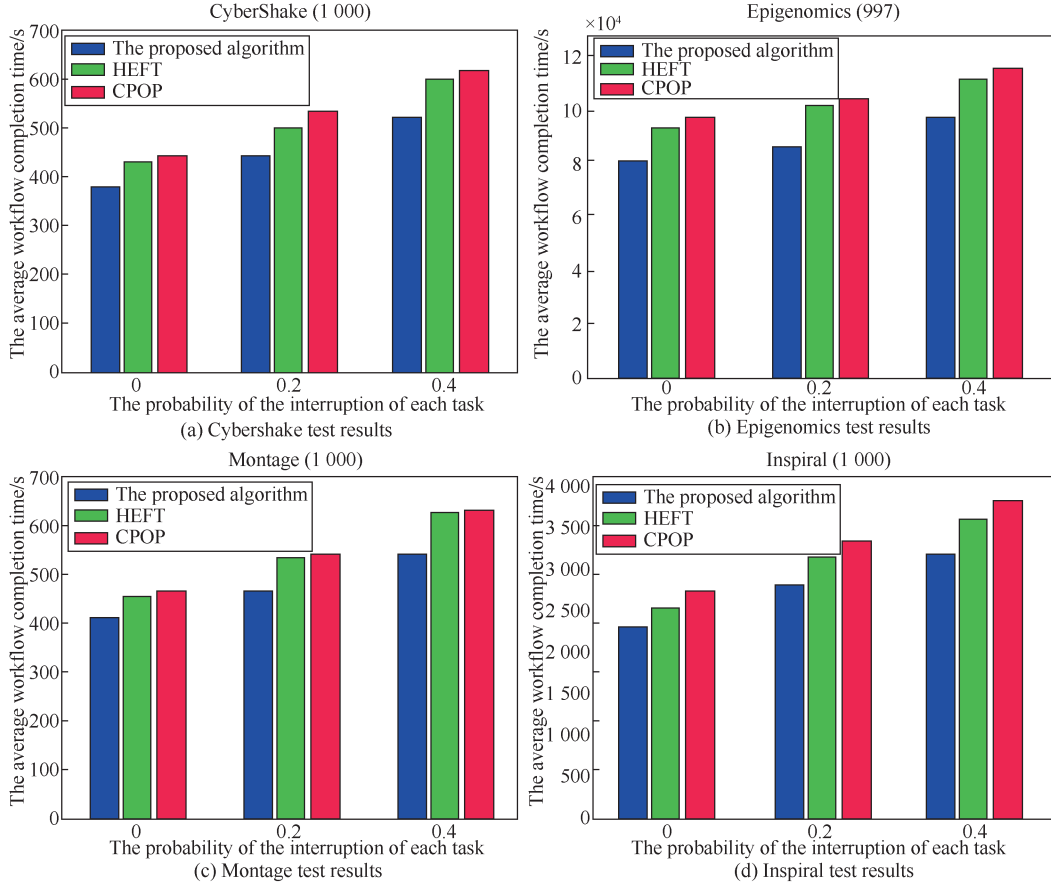


Figure 8 The evaluation results of computing instance allocation algorithm

are more idle instances during the workflow execution, which can reduce the value of t_{idle} and improve the efficiency of workflow execution. But the efficiency improvement is not endless, since t_{start} is affected by both $t_{longest}$ and t_{idle} , the increase in the number of available computing instances will affect the value of t_{idle} but will not affect $t_{longest}$. With the decrease of the number of available computing instances, compared with the original workflow, the performance of workflows with the critical path redundancy will deteriorate more rapidly. Because the purpose of the critical path redundancy method is to reduce the value of $t_{longest}$ under task interruption, if the number of available computing instances is small, t_{idle} will be the main factor in determining $t_{start}(v_i)$, the purpose of reducing $t_{longest}$ will become meaningless. Furthermore, the critical path redundancy method in-

creases the number of tasks in the workflow, more computing instances are needed to execute the added tasks, so the proposed method increases the demand for resources. Therefore, the proposed method can produce good results only when there are enough computing instances available.

7 Conclusion

The task interruption in the workflow will delay the workflow completion. For the problem, we firstly analyze and conclude that delay effects caused by the task interruption is related to the length of the path where the interrupted task is located, the longer the path, the greater the delay effects. Then we present the critical path redundancy method to create a redundant path having the interact parallel relationship between the critical path, reducing the interruption probability of the critical path. Furthermore, since

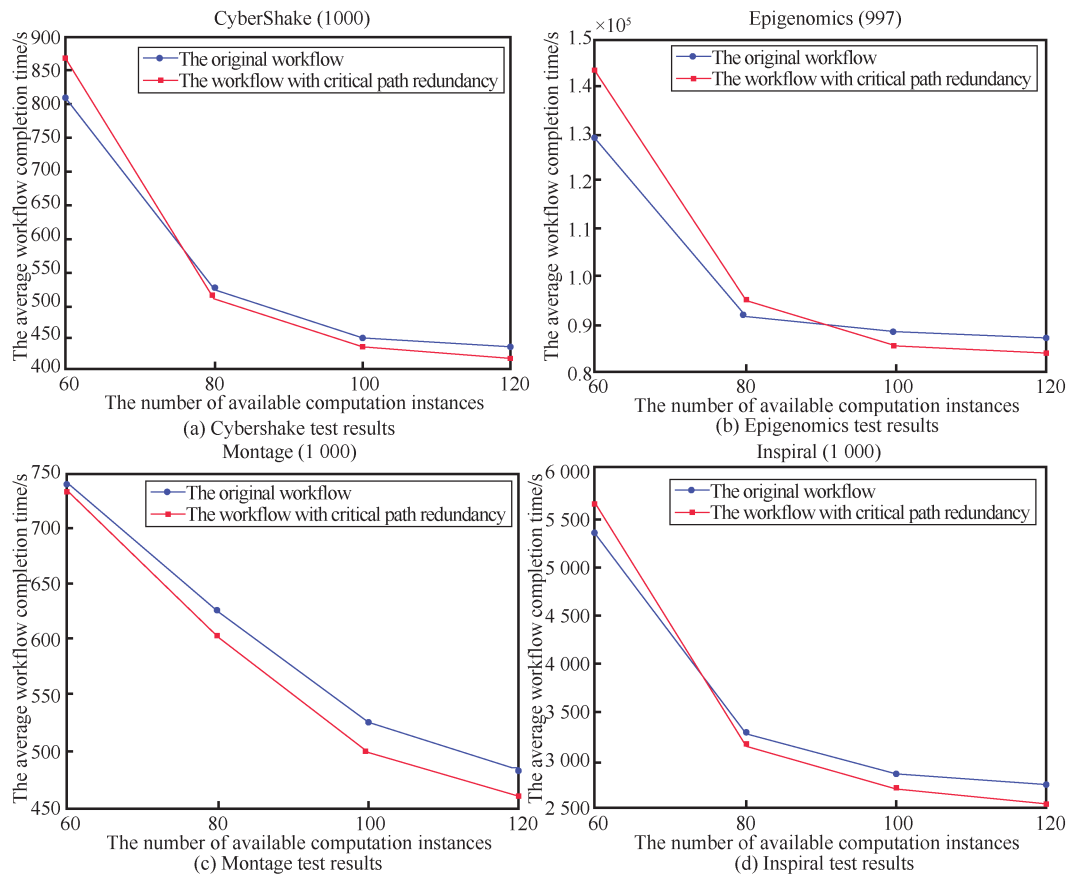


Figure 9 The evaluation results of resource requirements of the proposed method

only the tasks assigned the instance can begin execution, a computing instance allocation method based on HEFT algorithm is proposed, which takes into account diverse task dependency relationships and takes full advantages of the critical path redundancy method, improving the efficiency of workflow execution. The experiment is conducted on Workflow-Sim, four real world workflow models are used for tests. Test results demonstrate that the proposed method can effectively reduce the workflow completion time under the task interruption.

References:

- [1] LIU L, ZHANG M, BUYYYA R, et al. Deadline-constrained coevolutionary genetic algorithm for scientific workflow scheduling in cloud computing[J]. *Concurrency and Computation: Practice and Experience*, 2017, 29(5): 1-12.
- [2] RODRIGUEZ M A, BUYYYA R. A taxonomy and survey on scheduling algorithms for scientific workflows in IaaS cloud computing environments[J]. *Concurrency and Computation: Practice and Experience*, 2017, 29(8): 1-23.
- [3] WANG M, ZHU L, Chen J. Risk-aware checkpoint selection in cloud-based scientific workflow[C]//2012 Second International Conference on Cloud and Green Computing. 2012: 137-144.
- [4] ABRISHAMI S, NAGHIBZADEH M, EPEMA D H J. Deadline-constrained workflow scheduling algorithms for infrastructure as a service clouds[J]. *Future Generation Computer Systems*, 2013, 29(1): 158-169.
- [5] WU Q, ISHIKAWA F, ZHU Q, et al. Deadline-constrained cost optimization approaches for workflow scheduling in clouds[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2017, 28(12): 3401-3412.
- [6] CALHEIROS R N, BUYYYA R. Meeting deadlines of scientific workflows in public clouds with tasks replication[J]. *IEEE Transactions on Parallel and Distributed Systems*, 2013, 25(7): 1787-1796.
- [7] VERMA A, KAUSHAL S. Deadline constraint heuristic-based genetic algorithm for workflow scheduling in cloud[J]. *International Journal of Grid and Utility Computing*, 2014, 5(2): 96-106.
- [8] DELDARI A, NAGHIBZADEH M, ABRISHAMI S. CCA: a deadline-constrained workflow scheduling algorithm for multicore resources on the cloud[J]. *The journal of Supercomputing*, 2017, 73(2): 756-781.
- [9] LIN B, GUO W, LIN X. Online optimization scheduling for scientific workflows with deadline constraint on hybrid clouds[J]. *Concurrency*

- and Computation: Practice and Experience, 2016, 28(11): 3079-3095.
- [10] ZHENG W, HUANG S. An adaptive deadline constrained energy - efficient scheduling heuristic for workflows in clouds[J]. Concurrency and Computation: Practice and Experience, 2015, 27(18): 5590-5605.
- [11] SAHNI J, VIDYARTHI D P. A cost-effective deadline-constrained dynamic scheduling algorithm for scientific workflows in a cloud environment[J]. IEEE Transactions on Cloud Computing, 2015, 6(1): 2-18.
- [12] YAO G, DING Y, REN L, et al. An immune system-inspired re-scheduling algorithm for workflow in cloud systems[J]. Knowledge-Based Systems, 2016, 99: 39-50.
- [13] DING Y, YAO G, HAO K. Fault-tolerant elastic scheduling algorithm for workflow in cloud systems[J]. Information Sciences, 2017, 393: 47-65.
- [14] WANG Y, WU J, GUO Y, et al. Scientific workflow execution system based on mimic defense in the cloud environment[J]. Frontiers of Information Technology & Electronic Engineering, 2018, 19(12): 1522-1536.
- [15] POOLA D, RAMAMOCHANARAO K, BUYYA R. Fault-tolerant workflow scheduling using spot instances on clouds[J]. Procedia Computer Science, 2014, 29: 523-533.
- [16] POOLA D, RAMAMOCHANARAO K, BUYYA R. Enhancing reliability of workflow execution using task replication and spot instances[J]. ACM Transactions on Autonomous and Adaptive Systems (TAAS), 2016, 10(4): 30.
- [17] YAO G, DING Y, HAO K. Using imbalance characteristic for fault-tolerant workflow scheduling in cloud systems[J]. IEEE Transactions on Parallel and Distributed Systems, 2017, 28(12): 3671-3683.
- [18] ZHENG Z, ZHOU T C, LYU M R, et al. Component ranking for fault-tolerant cloud applications[J]. IEEE Transactions on Services Computing, 2011, 5(4): 540-550.
- [19] TOPCUOGLU H, HARIRI S, WU M. Performance-effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE transactions on parallel and distributed systems, 2002, 13(3): 260-274.
- [20] VERMA A, MITTAL M, CHHABRA B. The mutual authentication scheme to detect virtual side channel attack in cloud computing[J]. International Journal of Computer Science and Information Security (IJCSIS), 2017, 15(3).
- [21] GROBAUER B, WALLOSCHKE T, Stocker E. Understanding cloud computing vulnerabilities[J]. IEEE Security & privacy, 2010, 9(2): 50-57.
- [22] MISHRA P, PILLI E S, VARADHARAJAN V, et al. Intrusion detection techniques in cloud environment: A survey[J]. Journal of Network and Computer Applications, 2017, 77: 18-47.
- [23] STEWIN P, BYSTROV I. Understanding DMA malware[C]// International Conference on Detection of Intrusions and Malware, and Vulnerability Assessment. 2012: 21-41.
- [24] CHEN W, DEELMAN E. WorkflowSim: a toolkit for simulating scientific workflows in distributed environments[C]//2012 IEEE 8th International Conference on E-Science. 2012: 1-8.
- [25] JUVE G, CHERVENAK A, DEELMAN E, et al. Characterizing and profiling scientific workflows[J]. Future Generation Computer Systems, 2013, 29(3): 682-692.
- [26] DEELMAN E, VAHI K, JUVE G, et al. Pegasus, a workflow management system for science automation[J]. Future Generation Computer Systems, 2015, 46: 17-35.

About the authors:



Yanwen WANG (1990–), born in Henan. He is working on his Ph.D degree at National Digital Switching System Engineering and Technological Research Center. His research interests include cloud security and scientific workflow scheduling.



Yunfei GUO (1963–), born in Henan. He is a Ph.D supervisor and professor at National Digital Switching System Engineering and Technological Research Center. His research interests include cloud security and telecommunication network security.



Wenyan LIU (1986–), born in Henan. He is a lecturer at National Digital Switching System Engineering and Technological Research Center. His research interests include cloud security.



Shumin HUO (1985–), born in Shanxi. He is a lecturer at National Digital Switching System Engineering and Technological Research Center. His research interests include cloud security.

《网络与信息安全学报》

第二届编辑委员会

主 任 委 员：方滨兴

常务副主任委员：李风华

副 主 任 委 员：李 晖 俞能海 Kui Ren[USA] 刘建伟

邹德清 杜学绘 刘华鲁

委 员：（按姓氏笔画排序）

丁 勇 马建峰 王 标 王布宏 王连海

王良民 王国胤 王明生 王美琴 牛 犇

仇洪冰 田有亮 田俊峰 史国振 付章杰

冯 涛 朱建明 延志伟 仲 盛 向 涛

刘 哲 刘 渊 刘西蒙 刘哲理 闫 峥

孙东红 杨宏宇 肖 甫 肖 亮 吴 铤

吴晓平 邱卫东 何道敬 沈 超 张小松

张云勇 张红旗 张宏莉 张建标 张振江

陆月明 陈立全 陈兴蜀 陈性元 陈雷霆

苗春雨 林 鹏 易东山 赵 波 赵琉涛

咸鹤群 禹 勇 姚志强 秦 拯 贾 焰

殷丽华 翁 健 黄欣沂 曹春杰 彭国军

蒋文保 韩伟力 韩劲松 程 光 阚海斌

David Naccache [FRA] Gang Qu [USA] Jinjun Chen [AUS]

Willy Susilo [AUS] Yier Jin [USA]