# Multiple DAGs reliability model and fault-tolerant scheduling algorithm in cloud computing system

**Weipeng Jing[1, 2*], Yaqiu Liu[1, 2]**

[1]*The College of Information and Computer Engineering, Northeast Forestry University, Harbin 150040, China*

[2]*Heilongjiang province engineering technology research centre for forestry ecological big data storage and high performance (cloud) computing, Harbin 150040, China*

*Received 1 March 2014, www.tsi.lv*

**Abstract**

In this paper, in order to provide the reliable scientific workflow scheduling problem for cloud computing, a dynamic of RANK-Hierarchical algorithm is put forward which taking account of communication contention as well as supporting task dependencies (CCRH). A communication contention model is first defined, as soon as the earliest completion of the primary and backup task is deduced, besides the executive processor is limited, use dynamic hierarchical method and calculate of each DAG unfair degree factor for multiple DAGs scientific workflow. It can deal with the problem that multiple DAGs workflow comes at different time and have various kinds of structure. Both the theory and experiments have proved the algorithm not only improve the scheduling fairness of multiple DAGs workflow but also shorten the average execution Makespan effectively while meeting reliability constraints and meanwhile the produce well robustness.

*Keywords:* cloud computing, multiple DAGs, RANK-Hierarchical, reliability

## 1 Introduction

Cloud computing as a new computing model gets more and more attention. It is integrated by a variety of distributed computing, storage and application resources, meantime, realized multi-level virtualization and abstraction. It is efficient to make large-scale network resources form provide to user in reliable way [2]. Cloud computing as the next generation computing model plays an important role on scientific computing and commercial computing, it has been concerned by current academia and the business community. Now some typical cloud computing has been appeared, such as Google Cloud [1], Microsoft Cloud [14], Amazon EC2 [15] and IBM Cloud [16], these systems are committed to achieve web search, social network based on cloud computing.

In these fields of scientific computing applications, such as high-energy physics, astronomy, polymer materials, earth sciences, forestry resources and so on, due to huge task of data need to deal with, cloud computing system can provide powerful computing support. Great relevance and priority constraint relationship that may exist between the type of application computing tasks, so it should be on-demand dynamically provision, configuration, reconfigure and deprivation computing resource services in the cloud computing environment to achieve cloud computing scientific workflows high scalability and availability. The aim of resource scheduling is to achieve calculation, collection of storage resources and scheduling tasks to meet the relationship between spatial and temporal effectively. The Traverna [23], ASKALON [19], VGrADS [32], Pegasus [24] respectively to achieve distributed computing, scheduling management of storage resource.

In recent years, due to the dynamic expansion of cloud computing, high availability, resources assigned according to the need, some projects used cloud computing platform manage scientific workflow have been emerged. Such as the Amazon EC2 [15] can provide scalable, reliable, service-on-demand computing and storage services on scientific computing applications. Literature [30,31] describes the scientific workflow applied on the Amazon cloud platform's runtime and energy costs; addition, the ASKALON [19] and VGrADS [32] have been started to support scientific workflow applied on cloud computing platform.

In heterogeneous distributed environments (cloud computing, grid systems), use the DAG to describe task relationship for scientific workflow applications. The DAG workflow scheduling algorithm is divided into static scheduling algorithm and dynamic scheduling algorithm. The static scheduling algorithm is that assumption the overall structure and precedence constraints are known, execution time of the task can be calculated. So resources are allocated before the execution of the task, then no longer be adjusted. The dynamic scheduling algorithm can allocate resource dynamically based on workflow changed in the task execution process. Literature [3, 4] proves the static scheduling algorithm is better than the dynamic scheduling algorithm in different angles.

* ***Corresponding author*** e-mail: 39750600@qq.com

In the cloud computing scientific workflow applications, which faced mass intensive data processing performance:

1) in the scheduling process of cloud computing scientific workflow, existing multiple DAG submitted at the same time or submitted dynamically during calculation process, therefore it is required to the scheduling algorithm can meet the changes in dynamic environment;

2) in the enterprise-class workflow applications, it is need to set the trusted protection mechanism to tolerate failures when system is running;

3) the reasonable scheduling mechanism, it should to guard user that submit scientific computing request has little effect on data centre's load and position.

Therefore the dynamic scheduling algorithm which has a high reliability fault-tolerant ability is important in cloud computing to meet the users demand for dynamic tasks submitted. This paper provides solutions to the above questions by proposing an innovative dynamic of RANK-Hierarchical scheduling algorithms to maximize the performance and reliability.

The rest of this paper is organized as follows. In Section 2, we present system and mathematical models. As part of the system model, we design the processor model, task model, communication links model, the task priority to identify and underlying assumptions. In Section 3, we propose the multi-DAG scheduling algorithm CCRH. Simulation results show that proposed algorithms improve the performance compared to reference algorithms by varying number of DAGs parameters in Section 4. Prior related works are compared in Section 5 Finally, in Section 6, we conclude the paper by summarizing the comparison results and future work.

## 2 System model

In this section, we introduce a scheduling model in cloud computing provider, which consists of processor, tasks and communication link. In cloud computing, the parallel tasks of scientific workflow applications can use the weights of nodes and edges to represent a directed acyclic graph (DAG), the following is formal definition:

**Definition 1**: Node and weights of side of the DAG Figure can use the four-array $G=(V,E,w,c)$, which $V=\{v_1,v_2,v_3...v_N\}$ represents the number of tasks, $E=\{e_{ij}\mid v_i,v_{\ j}\in V\}$ represents the communication edge combination of dependency between tasks, $w(v_i)$ represents computational cost of the task, $c(e_{ij})$ represents communication cost between $v_i$ and $v_j$

**Definition 2**: The collection $\{v_x\in V:e_{xi}\in E\}$ represents the task of $v_i$ 's predecessor node set, denoted by $pred(v_i)$. The collection $\{v_x\in V:e_{ix}\in E\}$ represents the task of $v_i$ 's successor node set, denoted by $succ(v_i)$. If $pred(v_i)=\varnothing$, the task node $v_i$ is the entry node, expressed as $v_{entry}$. If $succ(v_i)=\varnothing$, the task node $v_i$ is the exit node, expressed as $v_{exit}$.

**Definition 3**: cloud computing environment use a variety of heterogeneous computing platforms built environment, set of heterogeneous processors described as $P=\{P_1,P_2\cdots,P_M\}$, $M$, which indicates the number of heterogeneous processors. The processor $P_k$ on the task $v_i$ of primary's start time and completion time represent as $t_s^p(v_i,p_k)$, $t_f^p(v_i,p_k)$ respectively; the task $v_j$ of backup's start time and completion time represent as $t_s^B(v_j,p_k)$, $t_f^B(v_j,p_k)$ respectively. Primary and secondary version of the task $v_i$ scheduling processor represent as $P^p(v_i)$ and $P^B(v_i)$ respectively.

**Definition 4**: the cloud computing system is network structure of any interconnection. Denote any processor's communication between $P_h$ and $P_k$ as $\ell_{hk}$.

In order to better illustrate the problem, we make the following assumptions:

1) The CPU time used by the task switching and process scheduler is negligible;

2) At the same time only exists one processor failure, it is impossible existing two processor failure at the same time. And fault according to the Poisson distribution;

3) The failure of the processor is fail-stop mode, the processor status is normal or failure to stop, while ignoring the fault detection time;

4) The multi-DAG workflow tasks arrive at any moment randomly in order to meet user's demand in cloud computing environment.

5) The communications link of cloud computing system is the arbitrary interconnection duplex structure, task communication only allow the same direction in the same time.

## 3 Multi-DAG scheduling algorithm CCRH

Priority of static scheduling method is the key to determine task priority, so computation of task priority has efficient impact on scheduling algorithm. The priority determination method of HEFT [18] algorithm is a widely typical algorithms applied to the actual. For example, the ASKALON [19] system also applied HEFT algorithm, and to prove the validity of scheduling DAG.

In the scientific workflow applications of cloud computing, as the DAG task reached dynamically, therefore a static priority method of calculation the multiple DAG mission priority cannot be used, this paper proposes a DAG scheduling algorithm mining dynamic and static, dynamic scheduling algorithm processes dynamically reached DAG task on the hierarchical, while static points to single DAG scheduling tasks in accordance with the static method.

## 3.1 CHECKING THE PDF FILE

Single DAG task priority uses static scheduling method, the algorithm set tasks $v_i$ and $v_j$ with dependent manner inspired by HEFT [18], and $v_j$ run directly dependent on the operating results of $v_i$. Considering the computing and communication's general consumption computed tasks priority:

$$rank(v_i) = \overline{w(v_i)} + \max_{j \in succ(v_i)} \{\overline{c(e_{ij})} + rank(v_j)\}, \quad (1)$$

where $\overline{w(v_i)}$ represents average execution costs on all processors of task $v_i$, $c(e_{ij})$ represents average communication cost between task $v_i$ and $v_j$. The priority of task $v_i$ is the largest value that plus direct successor task priority and communication with its own computational cost. Priority of all tasks is traversing the task graph upward from export task; the export task's priority is defined as:

$$rank(v_{exit}) = w(v_{exit}). \quad (2)$$

This paper considers the competition of communication link, the basic idea of the scheduling in the communication section is treat the computing nodes and communication in the DAG equally. Therefore, communication contention scheduling algorithm should not only consider the processor scheduling, but also consider the communication link scheduling among processors [25, 26]. In order to better solve the link communication competition, this article uses any cloud computing environment, network interconnect heterogeneous computing system communication path to find the shortest path search algorithm based on insertion strategy [26]. Defined $LST(e_{ij}, \ell)$, $LFT(e_{ij}, \ell)$ as $e_{ij}$ communication start time and completion time in the communication link $\ell$ and: $LFT(e_{ij}, \ell) \geq LST(e_{ij}, \ell) + c(e_{ij})$.

## 3.2 PRIMARY AND BACKUP TASKING SCHEDULING

In order to improve the reliability of the cloud computing system, in this paper the primary and backup scheduling method is used to achieve fault tolerance which performs redundant tasks in the backup processor, while ensuring the real-time nature of the task. And in order to improve system performance, this paper uses overlapped primary and backup tasks to determine the earliest start time of primary and backup task.

### *3.2.1 The primary task scheduling*

First the primary task is consider to schedule, according to the backup completion time of the set of predecessor $pred(v_j)$, the start time of the primary task $v_j$ has following three situations:

1) $t_s^p(v_j, p) > \max_{v_i \in pred(v_j)} \{t_f^B(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$, The start time of the primary task $v_j$ is greater than the maximum of the latest completion time of the backup task set $pred(v_j)$ and data transmission time in the communication link, then if the processor where the primary of any task has failed, the task $v_j$ can successfully receive the message which sent by all the predecessor task.

2) $t_s^p(v_j, p) > \max_{v_i \in pred(v_j)} \{t_f^p(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$ and $t_f^p(v_j, p) < \max_{v_i \in pred(v_j)} \{t_f^B(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$. The start time of the primary task $v_j$ is less than the maximum of the latest completion time of the backup task in the task set $pred(v_j)$ and data transmission time in the communication link. In this case, if the start time of the backup task is less $\max_{v_i \in pred(v_j)} \{t_f^B(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$, then when the processor where the task $v_i$ fails, the task $v_j$ cannot successfully receive the message sent by the entire predecessor task, and cannot get the right results. Therefore, the start time of the backup task $v_j$ is greater than $\max_{v_i \in pred(v_j)} \{t_f^B(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$ that meet the fault tolerance of the system.

3) $t_s^p(v_j, p) > \max_{v_i \in pred(v_j)} \{t_f^p(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$, $t_s^p(v_j, p) < \max_{v_i \in pred(v_j)} \{t_f^B(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$ and the completion time $t_s^f(v_j, p) > \max_{v_i \in pred(v_j)} \{t_f^B(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$. The start time of the primary task $v_j$ is greater than the maximum of the latest completion time of the backup task set $pred(v_j)$ and data transmission time in the communication link, and the completion time is less than the maximum completion time of the backup.

In CCRH, when the algorithm schedules the primary of the different DAG task, the primary task can be considered to be independent and non-priority task, and its constrains of independent scheduling is only with their own priority, looking for the processor of the earlier start to complete task based on scheduling processor queue.

### *3.2.2 The backup task scheduling*

In this section, we analysis the earliest start time of executing the backup task $v_j$. First we define the constraints of scheduling the primary task. When the schedule of the primary task $v_j$ meet the state (1) or (3), the start time of its backup task must meet:

$$t_s^B(v_j, p_h) > t_s^p(v_j, p_k) . \quad (3)$$

When the schedule of the primary task $v_j$ meet the state (2), the start time of its backup task must meet:

$$t_s^B(v_j, p_h) > \max_{v_i \in pred(v_j)} \{t_f^B(v_i, p), \underset{v_i \in pred(v_j)}{LFT}(e_{ij}, \ell)\} . \quad (4)$$

In order to achieve the fault-tolerant of the system, the processor of Cloud computing environment be scheduled by the backup task $v_j$ also need to meet the processor constraint: When the schedule of the primary task $v_j$ meet the state (1), the processor scheduled by its backup task meet:

$$P^B(v_j) \notin \{ \bigcup_{v_i \in pred(v_j)} P^p(v_i) \} . \quad (5)$$

When the schedule of the primary task $v_j$ meet the state (2) or (3), $pred(v_j)^2$ represents the task set that meeting the state (2) or (3) in set $pred(v_j)$, $pd(v_j)$ represents the task set that exist indirect and direct dependence and meet the state (2) or (3) of the task $v_j$:

$$pd(v_j) = \{pred(v_j)^2\} \cup \{ \bigcup_{v_i \in pred(v_j)^2} pd(v_i) \} . \quad (6)$$

So

$$P^B(v_j) \notin \{ \{ \bigcup_{v_i \in pred(v_j)} P_{pd}^p(v_i) \} \cup \{ \bigcup_{v_i \in pred(v_j)} P^p(v_i) \} \} . \quad (7)$$

Here $P_{pd}^p(v_i)$ represents the processor where all primary task in $pd(v_i)$.

**Lemma:** the earliest start time of the backup task $v_j$ is $\max_{i \in pred(v_j)} \{t_f^B(v_i, p), \underset{i \in pred(v_j)}{LFT}(e_{ij}, \ell)\}$ and its backup task cannot be dispatched to the processor(the virtual machine) $\{ \bigcup_{v_i \in pred(v_j)} P_{pd}^p(v_j) \} \cup \{ \bigcup_{v_i \in pred(v_j)} P^p(v_j) \}$.

**Proof:** assume that the start time of the backup task $v_j$ is less than $\max_{i \in pred(v_i)} \{t_f^B(v_i, p), \underset{i \in pred(v_i)}{LFT}(e_{ij}, \ell)\}$, then when the processor where the backup of $pred(v_j)$ complete time last failure, need to execute the backup task, $v_j$ will not be able to receive the messages sent, and task $v_j$ fail. Assume the backup task $v_j$ is scheduled to the processor $\{ \bigcup_{v_i \in pred(v_j)} P_{pd}^p(v_i) \} \cup \{ \bigcup_{v_i \in pred(v_j)} P^p(v_i) \}$, then when the processor $P^p(v_j)$ fail, and the malfunction of its precursor node do not recover, then the backup task $v_j$ will not run properly. So the assumption is not true.

The goal of scientific workflow task scheduling in cloud computing environment is getting the earliest completion time (*Makespan*) of the task. The earliest completion time of all DAG tasks is the exit node completion time of the backup task.

$$Makespan = t_f^B(v_{exit}, p) . \quad (8)$$

The earliest start time of CCRH looking for is calculating the earliest completion time of the backup task in scheduling strategy of the entire task.

### 3.3 MULTI-DAG HIERARCHICAL SCHEDULING

In DAG scheduling model of the cloud computing system, as the DAG workflow a will compete with other DAG workflow for the same set of computing resources, so the *Makespan* (the time from submit DAG a to finish the last task) of the workflow a is likely longer than the *Makespan* which it use the cloud computing environment separately, these two *Makespan* can be represented as $M_{multi}(a)$ and $M_{own}(a)$ separately. Literature [18] *Slowdown* is described this ratio: $Slowdown = M_{multi}(a) / M_{own}(a)$, so the inequities factor $Unfaines(s)$ of a scheduling algorithm s is defined as :

$$Unfaines(s) = \sum_{\forall a \in A} \left| Slowdown(a) - Avgslowdown \right| , \quad (9)$$

where $A$ is a set of being given multi-DAG. $AvgSlowdown$ is the average of *Slowdown* of all DAG, i.e. $AvgSlowdown = \frac{1}{|A|} \sum_{\forall a \in A} Slowdown(a)$, $|A|$ represent the base of set *A*, $Unfaines(s)$ is an important indicator that be used to measure the unfair degree of multi-DAG scheduling algorithm.

In the literature [7], the method of the multi-DAG task scheduling is: sorting the new task and the remaining tasks in DAG ascending according to the weight. If the weight of the new DAG task is always less than that the remaining DAG tasks, then the new DAG task is not scheduled, which will lead that the new DAG task cannot be scheduled as the weight.

So this paper proposes the multi-DAG scheduling method based on layer, the basic principle is to stratify the every DAG arrived in cloud computing environment any time, and the every layer of the last DAG is merged to that one where the DAG task do not be performed. Then it sorts every layer ascending on the basis of the task weight. So it will avoid the problem of the time span increasing due to the remaining task of previous DAG not be scheduling.

The concrete steps are as follows:

1) To stratify each DAG task in the scientific workflow processing.eg: in the 0 moment DAG-A arrives, then DAG-Ai(i=1,2…m) represent the i-th layer of the DAG-A.

2) To calculate the priority weight of all the tasks in every DAG according to the formula.

3) If it is single DAG, sort the task descending base on the priority weight, and submit to the schedule queue, then schedule in turn. The order is the primary task first and then the backup. Otherwise go to step (4).

4) If it is multi-DAG, merge the first layer of the last DAG to the next layer of the current DAG task.eg: in the t moment, when DAG-B arrives, the DAG-Ai task is performing, then put the DAG-B1 task to DAG-Ai+1 layer.

5) The tasks of every layer sort ascending according to the weight, e.g.: the tasks of DAG-Ai+1 sort ascending according to the weight, submit the schedule queue, then schedule the task in turn. The order is also the primary task first and then the backup.

6) Calculate unfair degree factor *Unfaines*(*s*) according to the Equation (9), and sort ascending. Schedule task to the processor low *Unfaines*(*s*) priority.

7) If the unfair degree factors *Unfaines*(*s*) of the multiple DAG equal, then schedule tasks in turn according to the *Makespan*.

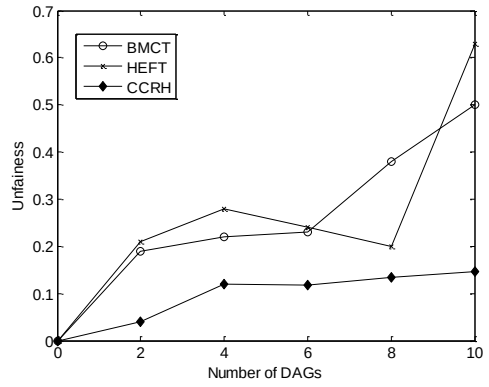
## 4 Test results and analysis

The simulation of the algorithm is compared with HEFT [18], BMCT [17] in the fairness of the fair factor scheduling, the scheduling time, the processor utilization and the task running time(the multi-DAG task of HEFT adopt the same way of stratifying), and compared with MaxAR [29] in robustness. In order to reflect the advantage of the algorithm in the scientific workflow better, we use four types of DAG task: random DAG task, FFT, Laplace and Fork-join, in which every type of the DAG contains 2-10 DAG task, and every DAG contains 10-50 task.

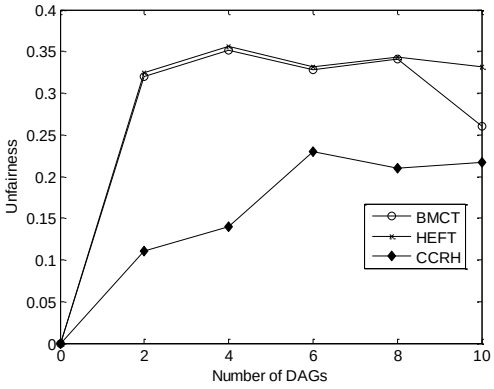
The experimental environment has Inter®Xeon E7420 2.13GHz，RAM 4G, the cloud computing environment of 1T hard disk. And use CCR to describe the ratio of communication and computing in DAG task graph, the value of the CCR select random number in 0.1-1.

### 4.1 THE FAIRNESS

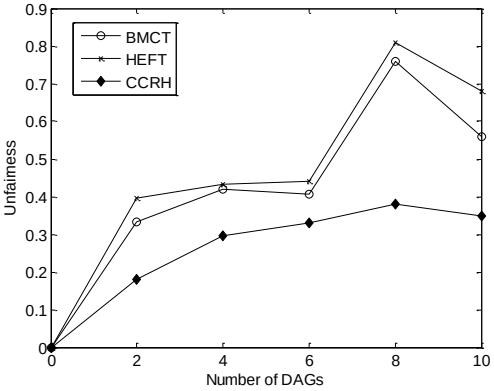
Compare the fairness of CCRH, HEFT and BMCT algorithm for different scientific workflow DAG Figure. Figure 1 a–d represent the fairness of the three algorithms in the random DAG Task FFT, Laplace, Fork-join graph respectively. The HEFT and BMCT using the same layered approach, its fairness do not have much difference. As CCRH use dynamical method, its fairness has improved greatly, but do not appear larger hopping phenomenon.



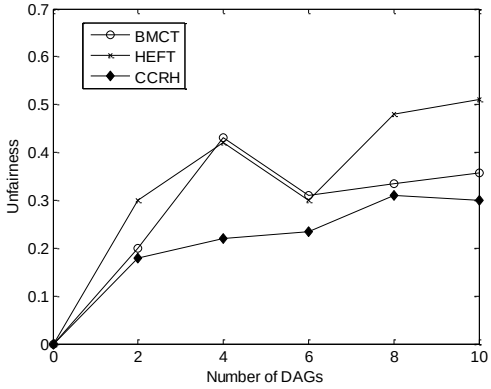

a) Random DAG




b) FFT




c) Laplace

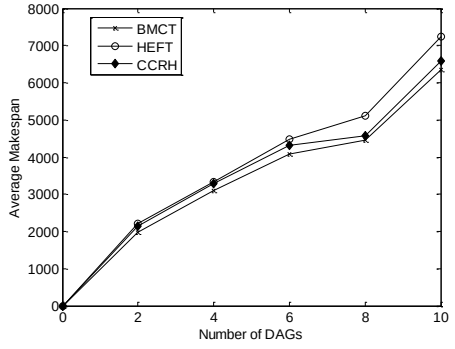



d) Fork-join

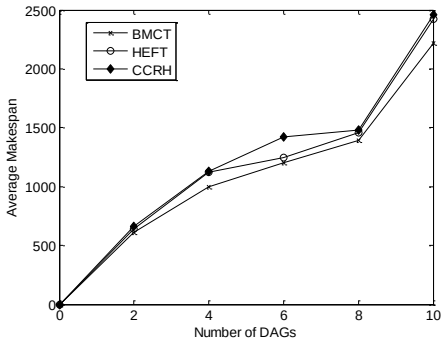
FIGURE 1 The Comparison of algorithm fairness

## 4.2 THE MAKESPAN ALGORITHM

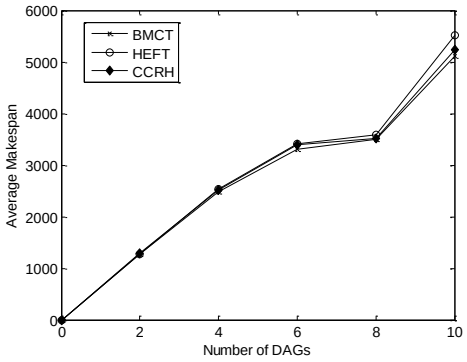
Compare the *Makespan* of CCRH, HEFT and BMCT algorithm for different scientific workflow DAG Figure. Figures 2 a–d represent *Makespan* in random DAG Tasks, FFT, Laplace, Fork-join respectively on three algorithms. There is little difference can be seen from Figure 2 in the three algorithms' *Makespan*. The main reason is that the three algorithms select a similar-priority comparison algorithm. BMCT is better than HEFT as BMCT considers communication constraints between tasks and as CCRH adopt the technology of primary and secondary version to improve system reliability, but its backup algorithm increases its *Makespan*.



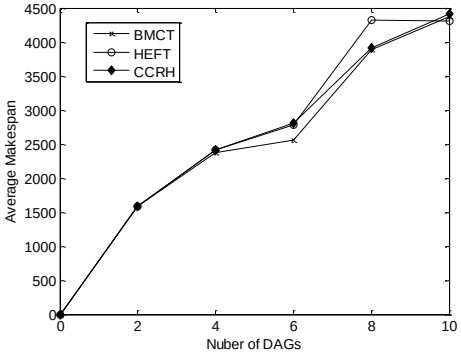

a) Random DAG




b) FFT




c) Laplace




d) Fork-join

FIGURE 2 The Comparison of average Makespan algorithm

## 4.3 THE LARGE-SCALE DATA COMPUTING TIME

In order to better test the overall performance of algorithm, by analyzing running time of 100 DAG in randomly generated environment. It is can be seen that HEFT performance the best of three algorithms, BMCT is poor. The CCRH use the technology of primary and secondary versions to improve the reliability, meanwhile expense its running time.

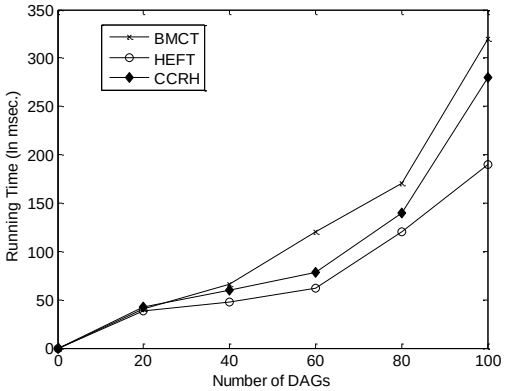



FIGURE 3 The Comparison of resource utilization

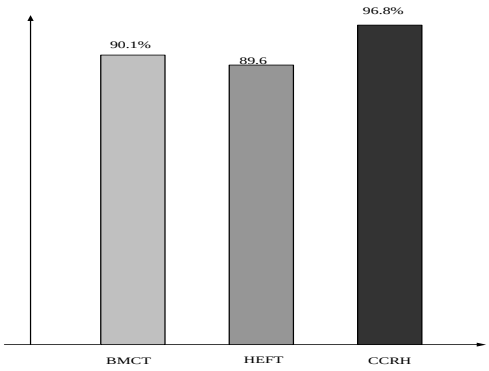



FIGURE 4 The large – scale data computing time

## 4.4 FOOTNOTES THE RESOURCE UTILIZATION

In order to better reflect utilization of the algorithm on cloud environment resources, we research the average utilization rate of the processor at four different scientific workflow loads. It is can be seen to from Figure.4 CCRH has a higher processing utilization. Thus CCRH have better benefits in the cloud environment that resource usage accounting

## 4.5 THE ROBUST OF ALGORITHM

In view of existing research for DAG task's scheduling algorithm, there is little on the robust of algorithm itself. The literature [28] tests 20 kinds of heuristic scheduling algorithm's robustness by introducing the standard deviation of Makespan. The literature and existing results assuming that the scheduler can obtain the information of the computing nodes at any time, and in practical applications, especially in large-scale cloud computing platforms, the collection of node information is affected by the competition in the communication link, node load factors, literature [29] proposed a novel robustness test method, which can be an measure the performance of algorithm effectively.

In order to reflect the scheduling model based on communication competitive advantage, in the simulation test, we assume that the scheduler compute nodes information in every 2 milliseconds. Testing the robustness of the algorithm on three parts: degradation approximation factor, average wait time and decay degree of critical path. Compare with the literature [29] MaxAR algorithm. The approximation factor defined as: $\eta = \frac{t_{\max}}{t_{\min}}$, here $t_{\max} = \max_{i=1...N}^{k=1...M} t_f^p(v_i, p_k)$ is the maximum completion time of the backup task $v_i$ in the processor set, $t_{\min} = \min_{i=1...N}^{k=1...M} t_f^p(v_i, p_k)$ is the minimum completion time of the backup task $v_i$ in the processor set; average waiting time of the critical path is $\overline{cpw} = \frac{1}{N}\sum_{i=1}^{N}(LFT(e_{ij}, \ell) - t_f^B(v_i, p_k))$ ; the average attenuation of the critical path is defined as follows: $cps_i = 1 + \frac{LFT(e_{ij}, \ell) - t_f^B(v_i, p_k)}{t_f^B(v_i, p_k)}$. It can be seen from Figure.5 due to use the tolerant mechanism of primary and secondary version, the CCRH performs the worst performance on approximate, but as CCRH can calculate the optimal start time of current task, thus the approximate factor attenuates in an acceptable range.

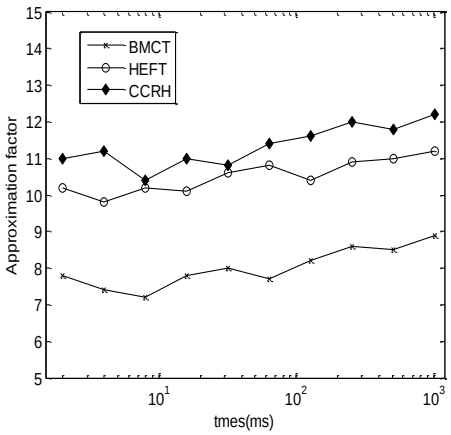



FIGURE 5 The comparison of approximate factor

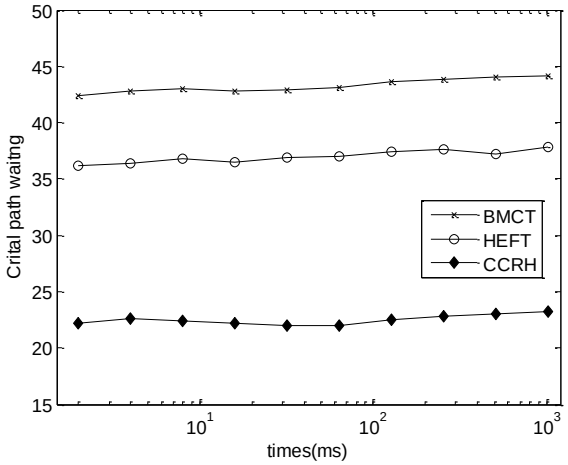



FIGURE 6 The average waiting time of critical path

Figure 6 shows the changes of critical path on average waiting time in the node information within the update interval. As CCRH fully considers link communications competition, its waiting time is more accurately reflect the algorithm to obtain the actual performance.

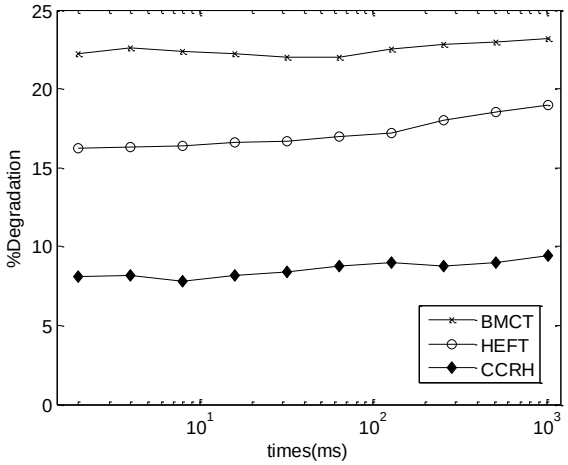



FIGURE 7 The average attenuation of critical path

It is can be seen from Figure.7 that the CCRH can calculate the optimal start time of the current task, the average attenuation of the critical path performs better. Meanwhile, in Figure 6 and Figure 7, with an increasing of node information's interval, so the algorithm robustness are affected, so choose the appropriate time to update the node information is the key to affect algorithm robustness.

## 5 Related work

The core idea of cloud computing is to manage and schedule a large number of computing resources connected by a network, and to constitute a pool of computing resources on-demand service to users. The key issues is how to schedule the resource fast and reasonable in cloud computing. So to schedule multiple DAG task is a way in effect of improving scientific calculation in scientific workflow applications of cloud computing.

Some of the relevant multi-DAG scheduling algorithms have been proposed, e.g.: literature [17] proposes a way of performing the multi-DAG task in turn, which result in producing a lot of idle wait time in the processor and prolonging the running time. Literature [5, 7, 12, 13] proposes a way of using a single DAG schedule way to schedule a complex DAG which is composed by multiple DAG. In [5], it proposes that multiple DAG merge into a complex DAG, then allocate resource for the complex DAG layer. In [7], it proposes a Planner-guided scheduling policy, which uses dynamic PANK-HYBD to schedule the priority for the multi-DAG tasks. These algorithms, however, does not consider the case that the DAG arrive at different times. In addition, literature [6] analysis [5, 7, 12, 13, 17] shows that the simple merger of DAG does not enhance the performance of algorithm significantly. Literature [13] proposes a multi-DAG scheduling algorithm that based on service time face to date-intensive applications. However this algorithm does not solve the problem which the running time increase because of the remaining tasks do not be scheduled in the former DAG. It is one of the keys in the cloud computing environment about scheduling problem that how to effectively solve the multi-DAG scheduling.

Moreover, the main goal of these algorithms is to explore the best completion time of the whole task, and ignore the task reliance. As the cloud computing is a new service model based on the large-scale low-cost service cluster, which hardware and software easily fail due to their own reasons or external factor. Literature [8] proposes to copy the task fully and define the position of this backup. In literature [9], an algorithm is proposed to meet the best Makespan and reliability, which improve the performance by putting the task to the computing nodes that has the smallest failure rate. Literature [10] proposes to improve the reliability by copy task based on literature [9], and schedule the task to the processor of load lightest. It proposes a fault-tolerant scheduling way of the priority constraint and the reliability cost driven, which emphasizes "the strong primary copy", and demand that the task must received the result of its all predecessor node, so this algorithm only consider the predecessor node of the task, without considering the completion of all nodes task. In literature [8, 10, 11], it uses the copy way in compromising the reliability and system performance. However, these methods only judge the copy task itself, without calculate the start time of coping the task truly, which affects the algorithm performance.

And these foregoing algorithms assume that the processors of any network are fully connected and it can receive the correlation information between scheduler and processor and between processors at any time. However, in practical applications, this assumption is untenable in the complex cloud computing environment. Its studies have shown that the scheduler algorithm considered the competition in the communication links can improve the accuracy grade effectively in Literature [27]. Literature [25] proposes a communication competition model in heterogeneous computing environment, and uses it to prove the validity of the scheduling algorithm, but this model is to consider the case of any network interconnection. In literature [26], it achieves the search and scheduler problem of processor in any interconnection network by the shortest path search algorithm in the communication competition model.

## 6 Contribution and future work

Against the reliable scheduling problem of scientific workflow in cloud computing system, this paper put forward a new method which use primary and secondary version to improve the system fault tolerance and dynamic hierarchical scheduling, the scheme has solved the problem when the multiple DAG task in quite different weights, the time span of DAG which arrived before will not be increased as the remaining tasks delays in scheduling. Simulation results show that in the premise of reliability requirements, the algorithm in fairness, Makespan, resource utilization, system run time showed better performance. The next step is to research in a given real cloud computing system architecture, how to solve the scheduling policies reliability under different failure probability, and through the different DAG scientific workflow load verify the algorithm's validity.

## Acknowledgment

The work described in this paper is supported by the Fundamental Research Funds for the Central Universities (DL13CB05) and the Application technology research and development in Harbin (2013AE1CE007) and Technological innovation talent research project in Harbin (2013RFXXJ089).

## References

[1] http://www.googlecloud.com/
[2] Boss G, Malladi P, Quan D, Legregni L, Hall H 2007 Cloud computing *IBM White Paper*
[3] Wieczorek M, Prodan R, Fahringer T 2005 Scheduling of scientific workflows in the Askalon grid environment *SIGMOD Record* **3**(34) 56-62
[4] Mandal A, Kennedy K, Koelbel, C, Marin G, Mellor-Crummey J, Liu B, Johnsson L 2005 Scheduling strategies for mapping application workflows onto the grid *Proceedings of the 14th International Symposium on High Performance Distributed Computing (HPDC 2005)* North Carolina USA 125-34
[5] Iverson M, Ozguner F 1999 Hierarchical, competitive scheduling of multiple dags in a dynamic heterogeneous environment. .Distributed Systems Engineering, 1999 **3**(6) 112-20
[6] Zhao H, Sakellariou R 2006 Scheduling multiple DAGs onto heterogeneous systems *Proceedings of the 15th Heterogeneous Computing Workshop (HCW)* Rhodes Island Greece

[7] Yu Z, Shi W 2008 A planner-guided scheduling strategy for multiple workflow applications *Proceedings of the Parallel Processing – Workshops 2008 ICPP-W'08 International Conference Portland Oregon* USA 20081-8
[8] Feng J, Humphrey M 2004 Eliminating Replica Selection—Using Multiple Replicas to Accelerate Data Transfer on Grids *Proceedings of the Parallel and Distributed Systems (ICPADS 2004)* Newport Beach CA USA 359-66
[9] Dongarra J J, Jeannot E, Saule E, Shi Z 2007 Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems *Proceedings of the 19th Annual ACM Symposium on Parallel Algorithms and Architectures (SPAA 2007)* New York USA 280–8
[10] [Saule E, Trystram D 2009 Analyzing scheduling with ransient failures. *Information Processing Letters* **109**(11) 539-42
[11] [Qin X, Jiang H 2006 A novel fault-tolerant scheduling algorithm for precedence constrained tasks in real-time heterogeneous systems *Parallel Computing* **32**(5) 331-56
[12] [Hönig U, Schiffmann W 2006 A meta-algorithm for scheduling multiple dags in homogeneous system environments *Proceedings of the 18th International Conference on Parallel and Distributed Computing and Systems (PDCS 2006)* Dallas Texas USA
[13] Zhu L, Sun Z, Guo W, Jin Y, Sun W, Hu W 2007 Dynamic multi DAG scheduling algorithm for optical grid environment *SPIE 6784 Network Architectures, Management and Applications* 2007 67-84
[14] http://www.microsoft.com/azure
[15] http://aws.amazon.com/ec2/
[16] http://www.ibm.com/ibm/cloud/
[17] Sakellariou R, Zhao H 2004 A Hybrid Heuristic for DAG Scheduling on Heterogeneous Systems *Proceedings of the 13th Heterogeneous Computing Workshop(HCW 2004)* Santa Fe New Mexico USA
[18] Topcuoglu H, Hariri S, Wu M 2002 Performance effective and low-complexity task scheduling for heterogeneous computing *IEEE Transactions on Parallel and Distributed Systems* **13**(3) 260-74
[19] Wieczorek M, Prodan R, Fahringer T 2005 Scheduling of scientific workflows in the Askalon grid environment *SIGMOD Record* **3**(34) 56–62
[20] Pandey S, Wu L, Guru S, Buyya R 2010 A particle swarm optimization based heuristic for scheduling workflow applications in cloud computing environments *Proceedings of the 24th IEEE International Conference on Advanced Information Networking and Applications (AINA 2010)* Perth Australia 400-7
[21] Salehi M A, Buyya R 2010 Adapting market-oriented scheduling policies for cloud computing *Proceedings of the 10th Conference on Algorithms and Architectures for Parallel Processing (ICA3PP 2010)* Busan Korea 2010 351-62
[22] Casanova H, Legrand A, Zagorodnov D, Berman F 2000 Heuristics for scheduling parameter sweep applications in grid environments *Proceedings of the Heterogeneous Computing Workshop 2000* 349-63
[23] Oinn T, Addis M, Ferri J, Mavin D, Senger M. Green-wood M, Carver T, Glover K, Pocock M R, Wipat A, Li P 2004 Tavern: A tool for the composition and enactment of bioinformatics workflows Bioinformatics **20**(17) 3045-54
[24] Deelman E, Blythe J, Gil Y, Kesselman C, Mehta G, Patil S, Su M H, Vahi K, Livny M 2004 Pegasus: Mapping Scientific workflows onto the grid *Proceedings of the European Across Grids Conference* Nicosia Cyprus 11-20
[25] Sinnen O, Sousa L A 2006 Toward a realistic task scheduling model, IEEE Transactions on Parallel and Distributed Systems **17**(3) 263-75
[26] Tang X, Li K, Padua D 2010 Communication contention in APN list scheduling algorithm *Science in China (Series F Information Sciences)* **52**(1) 59-69
[27] Macey B S, Zomaya A Y 1998 Performance evaluation of CP list scheduling heuristics for communication intensive task graphs *Proceedings of the First Merged International Parallel Processing Symposium and Symposium on Parallel and Distributed Processing* March-April 1998 538-541
[28] Canon L-C, Jeannot E, Sakellariou R, Zheng W 2008 Comparative evaluation of the robustness of dag scheduling heuristics. *In Sergei Gorlatch, Paraskevi Fragopoulo, Thierry Priol editors, Integration Research in Grid Computing, Core GRID integration work-shop* Hersonissos Crete Greece 63–74
[29] Hirales-Carbajal A; Tchernykh A; Yahyapour R 2012 Multiple Workflow Scheduling Strategies with User Run Time Estimates on a Grid *Journal of Grid Computing* 2012 **10**(2) 325-46
[30] Juve G, Deelman R, Vahi K, Mehta G, Berriman B, Berman B P, Maechling P 2009 Scientific workflow applications on Amazon EC2 *Proceedings of the 5th IEEE International Conference on e-Science* 2009 59-66
[31] Deelman E 2010 Grids and clouds: making workflow applications work in heterogeneous distributed environments *Int. J. High Perform. Comput.* **24** 284-98
[32] Ramakrishnan L, Koelbel C, Kee Y-S, Wolski R, Nurmi D, Gannon D, Obertelli G, Yarkhan A, Mandal A, Huang T M, Thyagaraja K, Zagorodnov D 2009 VGrADS: enabling e-science workflows on Grids and clouds with fault tolerance *Proceedings of the Conference on High Performance Computing Networking, Storage and Analysis* New York USA **47** 1–12

## Authors

**Weipeng Jing, born in January, 1979, Heilongjiang**

**Current position, grades:** Lecture in Northeast Forestry University. Aa member of the CCF
**Scientific interests:** modelling and scheduling for distributed computing systems, system reliability estimation, fault tolerant computing and system reliability, distributed computing.

**Yaqiu Liu, born in February, 1971, Heilongjiang**

**Current position, grades:** Professor at the Northeast Forestry University.
**University studies:** M. Eng. in Control Theory and Engineering from Northeast Forestry University in 1999. PhD in Navigation, Guidance and Control from Harbin Institute of Technology in 2004.
**Scientific interests:** process control, distributed computing, cloud computing, intelligent control and soft computing, model reconstruction.