Real-time Fault-tolerant Scheduling Algorithm for Distributed Computing Systems

Yun Ling, Yi Ouyang
College of Computer Science and Information Engineering
Zhejiang Gongshang University
Postal code: 310018
P.R.CHINA
{yling, oyy}@mail.zjgsu.edu.cn

Journal of Digital Information Management

ABSTRACT: *This article proposes a Distributed Real-time Fault-tolerant model, priority Real-time Fault-tolerant algorithm and computational architecture of Distributed Real-time Fault-tolerant. According to this model, the problem of how to schedule a weighted Directed Acyclic Graph (DAG) in Distributed computing system for high reliability can be solved in the presence of multi-processors faults. When some tasks in a DAG have dependence on each other, a task must be scheduled to make sure that maximum communication efficiency and high reliability can be guaranteed due to a processor failure. Firstly, this paper propose task model, communication model and reliability model for evaluating fault-tolerant performance of the system, and define the priority of a task so that a critical task is defined as one with the highest priority. To add some constrained conditions that do not influence the earliest start time of its successors in the process of scheduling task, The Distributed Real-time Fault-tolerant Scheduling Algorithm (DRFACS) targets maximizing reliability to dynamically schedule dependent, non-preemptive, non-periodic real-time tasks to improve the quality of service through scheduling in the case of arising massive resource failures. Finally, Experimental results demonstrate the feasibility of the proposed algorithm.*

Categories and Subject Descriptors:
D.4.7 [Organization and Design]; Distributed systems **C.4 [Performance of Systems];** Fault tolerance

General Terms: Distributed computing, Fault-tolerance systems

Keywords: Fault-tolerant Scheduling Algorithm, Directed Acyclic Graph (DAG), Communication reliabiity

Received: 15 March 2012, Revised 28 May 2012, Accepted 9 June 2012

1. Introduction

Fault tolerance, communication efficiency and reliability are important requirements in distributed computing, which often includes geographically distributed nodes co-operating in executing tasks. The study on fault-tolerance plays a key role in distributed computing, especially when multi-processors fail. In the last decade the field of Distributed Computing Systems had an outstanding evolution from scientific areas to commercial areas. With the development of these areas, looking for cheap and powerful computing platform around the corner - the Distributed computing system came into being. Distributed computing is to enable distributed computing tasks on a large number of distributed computers. The main idea of Distributed computing is the computing resources on the Internet all together and it mainly focused on the feature of high reliability. However, large-scale Distributed computing resources used by the system are highly dynamic and heterogeneous resources. The inherent state of environment is unreliable, so the distributed computing platform has greater possibility of errors than the traditional condition. The challenge in Distributed computing is to tolerate large-scale distributed system or eliminate these errors, which is caused by computer hardware failures, software errors and failures of other sources [1-2].

2. Related works

High reliability and minimized communication time involved in the research are few. The case for multi-treatment failure, some methods have been proposed, such as FTBAR [4], FTSA, MC-FTSA [5], CAFT [6]. FTBAR is based on List heuristic active replication technology. Scheduling algorithm, which allows a task scheduling $N+1$ copies to different processors, they can be parallel execution and tolerate up to N processors failure. However, a large number of copies of the task and assigning these tasks to processors can cause a lot of communication latency and improve the computing complexity degrees. In order to solve the communication problem, FTSA proposed a new data structure to reduce the large number of redundant

communication it by mapped to the communication between the processor and task as one to one relationship. However, the reduction of communication will affect the start time of communication task. The cause is that FTSA didn't take into account the start time of follow-up tasks so that the execution time of DAG is not optimized. CAFT, MC-FTSA from communication and time complexity point, by multiple copies of the task mapped to different processors, they have minimized the number of communication and tolerance of processor failure has been largely restricted, that is, the less the number of processors can tolerate less communications failure. However, in most of the previous work only constrained experimental settings are considered. In contrast, we focus on reliability-driven.

The paper focus on the case in large-scale resources, especially multi-processor failure, the reliability of the consideration of dynamic scheduling, from interdependent scheduling objective, non-preemptive, to non-periodic tasks in real time algorithm, named it-DRFACS, attempts to improve scheduling system reliability, while the task scheduling algorithms to fully consider the time and communication time of inter-processor, making the algorithm more realistic and more accurate.

3. System model and problem statement

3.1 Task Model

The tasks and the relationships among them can be simulated by DAG (a weighted Directed Acyclic Graph), as shown in Figure 1. Let $T = (V, E)$, where V is a set of nodes corresponding to all non-periodic, non-preemptive properties of real-time tasks. Meanwhile, the tasks are still divided into independent and dependent ones. The independent one has no relationship connected with others. The dependent task means that it is the result of a task which will affects the implementation of others. E is a set of edges corresponding to all the priority relations between tasks and communication between tasks. $v = |V|$ is the number of nodes, $e = |E|$ is the number of edges. In the DAG, if a node does not have predecessor node, it can be beginning node. If a node does not have successor node, it can be end node. For a task i, $Sdp(i)$ represent the set of predecessor nodes. $Sda(i)$ represents the set of successor nodes of task i. We use $w(i, j)$ to represent the amount of data sent from the task i to task j.

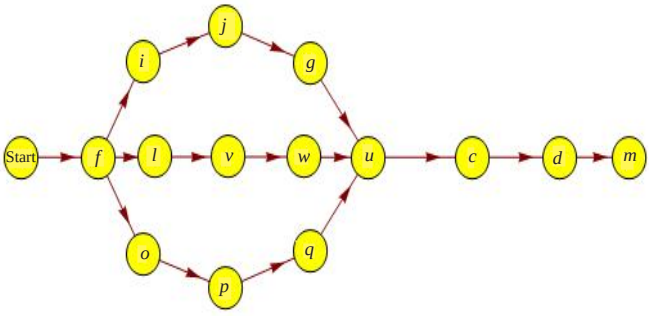


Figure 1. A Weighted Directed Acyclic Graph

3.2 Communication Model

There are a set $P = \{P1, P2, \ldots, Pm\}$ representing a set of processors in Distributed computing system. It can be also divided into heterogeneous and homogeneous processor. Distributed computing system is heterogeneous processors and these processors are fully connected. The link L_{kb} shows that the processor R_k and R_b is connection. We use $\varepsilon : V \times P \rightarrow R$ to simulate heterogeneous computing tasks, and it means each processor on each task execution time $\varepsilon(i, P_j), 1 \le j \le m$. The property of communication heterogeneous can use $W(i, j) = v(i, j) \times d(R_k, R_b)$ to represent, which means task i map to processor . means required for time in sending unit length of data. If the task is deployed to the same processor, the communication time is zero. Task mapping matrix X is a 0-1 matrix, means the relationship between the tasks in DAG be mapping to the processors on the system. x_{ij}=1 represents task i be mapping to the processor P_j, and otherwise $x_{ij} = 0$.

3.3 Reliability Model

Since many real-time systems are non-deterministic environment in which even dangerous, system fault tolerance is necessary and important. In order to effectively verify the level of fault-tolerant systems, a reliability model needs to be proposed, it is generally assumed that the error arrival rate is stable and the error probability at any time intervals meets Poisson distribution [7-8]. Thus, by the following derivation, the reliability mode will be used to evaluate the performance of our proposed fault-tolerant scheduling strategy.

We define the reliability of the model is similar to Literature [9]. We assume that the probability of permanent failure obeys the Poisson distribution and they are mutually independent. Literature [10] proposed a K-Timely Scheduling (K-TFT scheduling), which think that there are K processors working failure, but the process of the task scheduling meet the deadline still. This paper scheduling goal is that as scheduling multi-processor failure appears, we can reach K-TFT also. At the same time, fault-tolerant scheduling for each task has a schedule matrix $X = [x_{I_i J}]_{m \times (Z+1)}$, and assume $\varepsilon + 1$ the copy of a task assigned to a processor, when a copy of the task i assigned to the processor P_j, $x_{I_i J} = 1$, the otherwise $x_{I_i j} = 0$, where $i = 1, 2, \ldots (\varepsilon+1), j = 1, 2, \ldots m$.

The processor failure may occur during idle time, but the reliability of the model does not take into account the processor failures in this time. It is caused by two reasons: firstly, in addition to affecting system reliability, the idle time of the failure will affect only the task completion time; secondly, the use of a spare cell to replace the failed processor can be a good deal of processor failure, which means that for reliability analysis of a similar failure is not critical.

The reliability cost, C_{ij}, of copy of task i is decided by the

failure rate λ_j of processors and execution time of copy of task i. Therefore, the reliability cost of processors is the sum of all copies of the reliability cost during the task scheduling.

Given a failure rate vector $\Delta = (\lambda_1, \lambda_2, \ldots \lambda_m)$, a definite schedule X and a task i, the processor reliability cost can be represent:

$$C_{PN}(\Delta, X, I = \sum_j^m \sum_{i=1}^{z+1} - \lambda_j x_{I_{ij}} c_{ij}) \quad (1)$$

So, the sum of reliability cost of processors is:

$$C_p = \sum_{I=1}^n C_{PN}(\Delta, X, I) \quad (2)$$

Before estimating the reliability cost of the connection processors, we first introduce a set of messages E_{kb}, it contains all the messages from the processor R_k to R_b that is:

$$E_{kb} = \{(v_{I_i}, v_{I_j}) \mid e_{ij>0} \wedge x_{I_j b} = 1|\}, \forall 1 \le k, b \le m, k \ne b \quad (3)$$

The failure rate of the connect processors is expressed as μ_{kb}. The reliability cost of a message $e_{ij} \in E_{kb}$ is decided by μ_{kb} and $w_{kb}|e_{ij}|$. Therefore, the reliability cost of message can be calculated as $-\mu_{kb} x_{I_i k} x_{I_j b} w_{kb}|e_{ij}| = -\mu_{kb} w_{kb}|e_{ij}|$. Based on reliability cost of a message-based definition, we can express the reliability cost of the connection processors as $C_{kb}(M, X, I, J)$. It is the total cost of the reliability of all information through this path, more specifically it can be get as the following equation (4), where M is the matrix of connection failure rate ($[\mu_{kb}]_{m \times m}$).

$$C_{kb}(M, X, I, J) = \sum_{i=1}^{z+1} \sum_{j=1}^{z+1} (-\mu_{kb} x_{I_i k} x_{I_j b} w_{kb}|e_{ij}|) \quad (4)$$

Due to the assumption that the system contains n tasks, the reliability cost of the connection processors is:

$$C_{kb} = \sum_{I=1}^n \sum_{J=1, J \ne I}^n C_{kb}(M, X, I, J) \quad (5)$$

Therefore, we can derive the total reliability cost of the connection processors RCLINK, it is the connect reliability cost of all connections, that is

$$C_{LINK} = \sum_{k=1}^m \sum_{b=1, b \ne k}^n C_{kb} \quad (6)$$

Now, we can determine the cost of system reliability RC, which is connected to all processors and the total cost of reliability. Therefore, the cost of system reliability can be expressed as:

$$C_{sum} = C_p + C_{LINK} \quad (7)$$

For a given distributed computing system, its reliability can be expressed as:

$$R(\Delta, M, X, I) = \exp(-C_{sum}) = \exp(-C_p) \times (-C_{LINK}) \quad (8)$$

4. Scheduling algorithm

In this section, we propose a fault-tolerant scheduling algorithm, which aims to ensure maximum reliability under completion time, and the task mapping process can tolerate any number of processor failures. DRFACS method use active replication technique to assign $\varepsilon + 1$ copies of the task on to different processors.

The DRFACS is a greedy heuristic algorithms based on the importance and reliability of tasks. Local DAG uses the idle task the length of the longest path to express the importance of task. A task is idle, and if it does not dispatch and at the same time, all of its predecessors have been dispatched. S is a set of scheduled tasks, U is a set of not scheduled tasks, $U_f \in U$ is the set of idle tasks. Once a task being scheduled to the processor, we will get the start time $t_j^{i,\varepsilon}$ and completion time $t_j^{i,f}$.

Most real-time scheduling using priority-based scheduling algorithm, this algorithm assigns a priority to the task, priority scheduling at each scheduling time with the highest priority task execution. According to priority assignment, real-time scheduling can be divided into static priority scheduling and dynamic priority scheduling. Static priority scheduling algorithm with good mathematics because of its theoretical basis, and can handle complex task model, is widely regarded as the main hard real-time system design basis [12-13]. Unlike static priority scheduling algorithms, dynamic priority scheduling algorithm based on the task resource requirements to dynamically assign priority, the aim is to resource allocation and scheduling with greater flexibility to the distributed computing system scheduling task execution. Therefore, the dynamic priority scheduling policy is more suitable for the scheduling of distributed computing systems analysis and research.

An idle task with highest priority is a critical task. The priority of a task by the $H(i) + L(i)$ to decide, where $H(i)$ means that the dynamic high level and $L(i)$ means low level of static, we can use the two methods [5-6] to calculate them.

Because system does not know what the task assigned to the processor, we consider the worst case communication to use the average task execution time $\overline{\varepsilon(i)} = \sum_j^m \varepsilon(i, P_j)$ and the average communication time $\overline{W(i,j)} = v(i,j) \times \bar{d}$ between tasks, where $\bar{d}$ means that the system per unit length of data sent between processors, $v(i,j)$ means the bandwidth of node i to node j, the average delay. $t_j^{P(j),i}$ means the task i completion time on the processor j, that is the start time on the task processor, which continued late in the task of all messages before the arrival processor, but also will be available later than the processor available time $t^{P_j, eat} = \max_{j \in s}(x_{ij} \times t_j^{P_j, f})$. Therefore, the task's start time:

$$t_j^{P(j),a} = \max(\max_{j \in S_{dp(j)}}(t_j^{P(j),f} + W(i,j)\}, t^{P_j, eat}) \quad (9)$$

To provide a good measure of the importance of task: the more important task, along this path the more the task of deployment. DRFACS consider the heterogeneous

computing systems and following objectives: (1) delay constraint, it is able to tolerate the permanent failure of any number of processors; (2) to accurately calculate the critical task to make the important tasks to be completed earlier; (3) system reliability.

Each step in the mapping process, DRFACS chooses a critical task at the same time using the following equation to simulate the mapping of tasks to all processors:

$$\forall 1 \le j \le m, t_j^{P(j),f} = \mathrm{EXP}[\lambda * \varepsilon(i, P_j) + (1-\lambda) * \Lambda(\{t_{u^k}^{P(u^k),f}, W(u^k, j), t^{P_j, eat}\})] \quad (10)$$

Where λ means regulatory factor, Λ is the spending time mapping function which determined by average communication time $W(i,j)$, execution time $t_j^{P(j),a}$,processor available time $t^{P_j, eat}$.

Lemma 1: Under Active Replication strategy, if and only if $P(i^k) \neq P(i^{k+1})$, k =1, 2, ..., ε, the task has å processors in the case of permanent failures can ensure truly implementation.

Proof: Because ε +1 copies of the task mapped to ε +1 different processors, so if ε processors fail, then there is $P(i_z)$ $(1 \le z \le \varepsilon + 1)$ no failure and will be executed successfully. However, if there is a processor $P(i_u)$ $(1 \le u \le \varepsilon + 1)$, and $P(i_u) = P(i_z) = P_w$, when P_w failed, then i_u and i_z could not be executed successfully.

However, we found that start time and reliability have conflict at sometimes, if fully consider the reliability, the task will result in a longer response time thus affecting the earliest start time of predecessors of task. Only consider start time will affect the reliability of the system. Therefore, we introduce a parameter $t_{r-a}(i)$ to limit the task response time, and it can lead to a balance between the start time and reliability.

$$t_{r-a}(i) = \min_{j \in S_{ds}(i)} \{\max_{q \in S_{dq}(j)} \{t_q^{P(q),f} + W(q,j)\}\} \quad (11)$$

DRFACS algorithm:

Step1: Parameters initialize.

Set the processors failed number for system support the maximum number ε;

Calculate each $L(i)$ of task i in DAG, and initialize $H(i)$=0 of each task i;

Determine the system's processors set $P = \{P1, P2, \ldots, Pm\}$;

Initialize $S = \emptyset$; $U = V$, idle task list $\alpha = \emptyset$; Let initial task insert the list α;

Step2: Using Dynamic Programming method to calculate the scheduled tasks

Select the highest priority task $i \leftarrow H(\alpha)$; Initialization reliability $r \leftarrow 0$,and $t_j^{P,f} \leftarrow \infty$;

For each processor P_j do

compute $t_j^{P_j, ef}$ using equation (10); calculate the earliest completion time

if tasks i can be completed on P $(z+1)$, and execution time is less than $t_{r-a}(i)$

Using equation (8) to detect R_j to make sure the task i in the processor $P(z+1)$ and connection reliability;

if $((R_j > R)$ *or* $(R_j = R)$ *and* $(t_j^{P,f} > t_j^{P_j, ef})$

Then schedule the task i to the ε +1 processors;

End for

if the task i is not possible for the processor, then return (FAIL);

Step3: Update scheduling queue.

The processor will be assigned to the task i when the reliability of processor reach the maximum; take task i into S, at the same time updating the priority of the predecessors;

The idle predecessors of task i will be placed in α;

Update U, $U \leftarrow U(i)$;

Update information of messages;

Time Complexity Analysis: For the time complexity of the algorithm, it calculates bl (i), which takes O $(e + v)$ (e is the number of edges and v is the number of nodes.), and insert and delete nodes need the cost of O $(\log|\alpha|)$ $(|\alpha| \le w$, w is the DAG's width). During the execution of each task, the action of insert delete only once, so the management for á of time complexity is $O(v\log w)$. Analysis algorithm, we found that the major computing costing at iteration. Loop need v times. The head node needs O $(v\log w)$ to find. Computing tasks at the same time the start time or completion time need $O(|S_{dp}(i)|(\varepsilon+1)m)$ (m is the processor number), the reason is that the predecessor of all copies in task i need for testing on each processor P_j (j =1,2,…,m) to detect. Also, because $\varepsilon < m$, the entire cost of v cycles is $O(em2)$. Next, we want to update the priority of task's direct successors, and the cost is $O(|S_{dp}(i)|)$, the total cost in v times iteration is O (e). So the time complexity of the DRFACS algorithm is O $(em2 + v\log w)$.

5. Performance test

In this section, we will evaluate the performance of the proposed method. We randomly generated task graphs, the parameters designing in accordance with the method of the literature [Benoit et al.2008]. We use three parameters to describe the resulting task graph: (1) the number of tasks from [100,150] range of uniform sampling; (2) corresponding to each task input/out edges is set to [1,3]; (3) The task graph of the computation and communication ratio: CCR (G) by increments of 0.2 from [0.2,2.0] range choosing, that is - the slowest of each

edge of each task communication time and the slowest execution time ratio, and if CCR $(G) < 1$, means that in contrast to the task execution time it receives a small amount of communication time. The number of processors is set to {20,40,80}, the failures number of processors range from {3,7}, these processors have different processing speed, assuming that they obey the [1.0,10] uniform distribution while their failure rates can be balanced from the MINR and MAXR. In literature [11], MINR means per hour 1.0 × 10-6, MAXR accordance with 0.5 × 10-6 the incremental changes between 3.5 × 10-6 and 7.5 × 10-6. At the same time, taking into account the heterogeneous communication system, the connections between task and messages capacity of the unit message passing delay can be sampling, respectively, from the range [0.5,1] and [50,150] in the uniform selection. These criteria from the scheduling ability, latency and reliability point of view to describe the performance of the algorithm.

5.1 Results and discussions

From scheduling, reliability and communication efficiency three aspects, we compare the experiment results. In order to obtain stable and accurate results, our experiment was repeated 20 times to get average value. At the same time, our distributed computing system includes 20 processors, similar to 40 and 80 processors will have similar results. We have to evaluate this part of the impact of CCR, which represents the communication time and execution time ratio. The high CCR value represents the relative value of large execution time, communication time is rather large, CCR value is randomly generated.

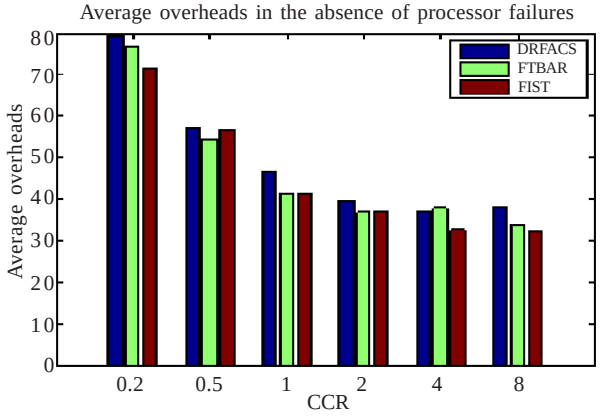
We present in this Section the performance results on the fault-tolerance heuristic. We compute the fault-tolerance overhead in the following way:

In contrast to the application's execution time, scheduling time will be ignored [5-6].

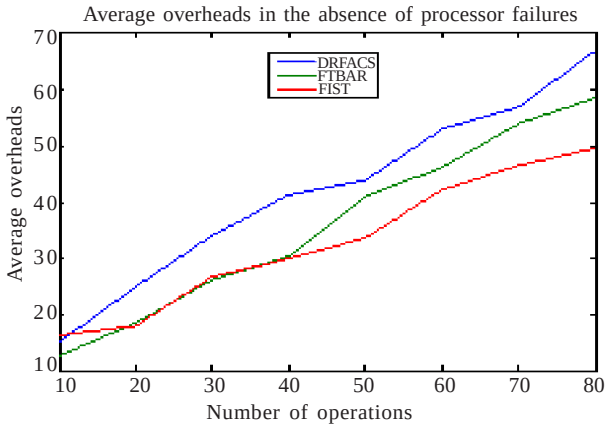
$Overheads = \frac{F_1 - F_2}{F_1} * 100$,where F_1 is Fault Tolerant Schedule Length. F_2 non Fault tolerant schedule length.

Figure 2 (a, b) shows that As the CCR increases, DRFACS also gradually improve the reliability, when the CCR value is very small, DRFACS the scheduling ability better than FTSA, FTBAR. But with the increase in the value of CCR, DRFACS performance was gradually reduced, and FTSA, FTBAR methods even worse than DRFACS trend. The results show that when CCR is small DRFACS would be much better than FTSA, FTBAR, when CCR is large, FTSA, FTBAR is also worse than DRFACS. The reason is the increase with the CCR value will increase the proportion of communication time, FTSA, FTBAR have a strong focus on communication time of our algorithm design process for communication time and execution time of balance, so they will be slightly better than the results DRFACS show that our algorithm is in the scheduling of an acceptable range.

Figure 3 (a, b) shows that with the increase of CCR, DRFACS, FTSA and FTBAR delays have tended to decline. The reason is mainly the increase of communication time to reduce the execution time with the CCR's value increasing. When the three processors failure, DRFACS, FTSA and FTBAR relationship among the three tender to stable, and when failure occurs between the 7 processors, FTBAR volatility is very strong. The DRFACS, FTSA clearly superior to FTBAR performance. This is mainly because the factors are taken into account will affect the delay, the process involves the communication time and execution time of the relationship. This shows that with the change in the number of processor failures, the performance of the proposed DRFACS method quite good performance, and good scalability.

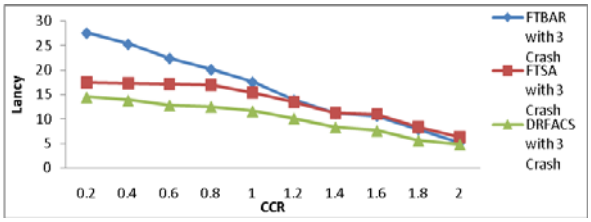


(a)

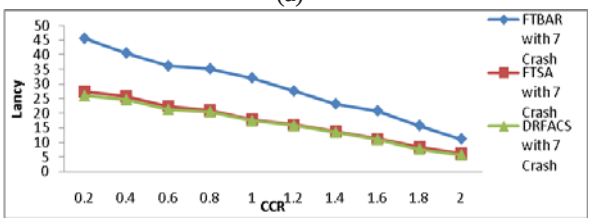


(b)

Figure 2. Scheduling capability and CCR



(a)



(b)

Figure 3. Delay time and CCR

6. Conclusions

This paper considered the situation of multi-processor failure, and defined system reliability model, the time constraints and the critical tasks. Based on these definitions, we proposed a reliability-based fault-tolerant scheduling algorithm. And from the scheduling, latency and reliability of the three aspects, comparing the simulation results show that the proposed algorithm has good performance, especially to ensure that scheduling and delay, while greatly improving the reliability of the system. Real-time distributed computing system fault-tolerant scheduling algorithm is still in an early stage, follow-up will improve and expand the reliability of the existing distributed computing system, and study the reliability of a hybrid scheduling model, which contains a mixture of periodic task and non-periodic task model, and consider the reliability of communication between tasks. Our method integrate with the physical environment at the same time, and simulation experiment has not yet detected a problem from the algorithm, the future research can use real data for further testing the availability of the method in real computing environment.

7. Acknowledges

This work is supported by the Zhejiang Provincial Natural Science Foundation of China (Grand No. LR12F02002) as well as the Scientific Research Fund of Zhejiang Province of China (Grand No. 2011C24008 and No. 2012C23121).

References

[1] Ling, Y., Luo, Z. S., Ge, Y. J. (2010). Fault-tolerant Scheduling Algorithm for Precedence Constrained Tasks in Grid Computing Systems with Communication Efficiency. *In: Proc. the 3rd International Conference on Information Sciences and Interaction Sciences,* p. 205-210, Jun.

[2] Xiaomin Zhu, Chuan He, Rong Ge, Peizhong Lu.(2011)Boosting adaptivity of fault-tolerant scheduling for real-time tasks with service requirements on clusters Original Research Article.*Journal of Systems and Software*, 84 (10) 1708-1716, October.

[3] Anne Benoit, Mourad Hakem, Yves Robert.(2009). Contention awareness and fault-tolerant scheduling for precedence constrained tasks in heterogeneous systems Original Research Article. *Parallel Computing,* 35 (2) 83-108, February.

[4] Girault, A., Kalla, H., Sighireanu, M., Sorel, Y. (2003). An algorithm for automatically obtaining distributed and fault-tolerant static schedules. *In: Proc. International Conference on Dependable Systems and Networks* (*DSN'03*), p. 102-106, Apr.

[5] Benoit, A., Hakem, M., Robert, Y. (2008). Fault Tolerant Scheduling of Precedence Task Graphs on Heterogeneous Platforms. *In: Proc. IEEE International Symposium on Parallel and Distributed Processing*, p. 1-8, Apr.

[6] Benoit, A., Hakem, M., Robert, Y. (2008). Realistic Model and Efficient Algorithms for Fault Tolerant Scheduling on Heterogeneous Platforms. *In: Proc. the 37th International Conference on Parallel Processing*, p. 411-416, Sept.

[7] Dongarra, J., Jeannot, E., Saule, E., Shi, Z. (2007). Bi-objective scheduling algorithms for optimizing makespan and reliability on heterogeneous systems. *In: Proc. the 19th Annual ACM Symposium on Parallel Algorithms and Architectures* (*SPAA'07*), p. 280-288, Jul.

[8] Qin, X., Jiang, H. (2001). Reliability-driven scheduling for parallel real-time jobs in heterogeneous systems. *In: Proc. the international Conference on Parallel Processing*, p. 131-138, Feb.

[9] Qin, X., Jiang, H. (2005). A dynamic and reliability-driven scheduling algorithm for parallel real-time jobs executing on heterogeneous clusters. *Journal of Parallel Distribution Computation*, 65 (2) 885-900.

[10] Oh Y., Son S.H. (1997). Scheduling real-time tasks for dependability. *Journal of Operational Research Society* 48 (6) 629-639.

[11] Qin, X., Han, Z., Pang, L., Li, S. (2000). Design and performance analysis of a hybrid real-Time scheduling algorithm with fault-tolerance. *Journal of software*, 11 (5) 686-693.

[12] Yang, C., Gui, W., Ji, L. (2003). A fault-tolerant scheduling algorithm of hybrid real-time tasks based on multiprocessors. *Chinese Journal of Computers*, 26 (11) 216-223.

[13] Xiaomin Zhu, Chuan He, Kenli Li, Xiao Qin. (2012). Adaptive energy-efficient scheduling for real-time tasks on DVS-enabled heterogeneous clusters. *Journal of Parallel and Distributed Computing*, 72 (6), June, p. 751-763.

Author biographies

Yun Ling is a Full Professor. His major research areas include computer networking, distributed computing and ubiquitous/pervasive computing. He has led many national and international research projects and industrial projects. He has researched on the ways of analyzing the performance of computing grid and put forward a range-based approach to distributed computing performance.

Yi Ouyang is a Associate Professor. His major research areas focus on visual computing subjects in the areas of computer vision and pattern recognition. He has researched on object detection, tracking, pose estimation, and recognition. Potential applications of his research reside in the domains of human-computer interaction, visual surveillance, and video analysis and content extraction.