Fault-Tolerant Scheduling Based on Periodic Tasks for Heterogeneous Systems

Wei Luo[1], Fumin Yang[1], Liping Pang[1], and Xiao Qin[2]

[1] School of Computer Science, HuaZhong University of Science and Technology, Wuhan 430074, P.R. China
free_xingezi@163.com, yangfm@routon.com, lppang@hust.edu.cn
[2] Department of Computer Science, New Mexico Institute of Mining and Technology, Socorro, New Mexico, 87801-4796, USA
xqin@cs.nmt.edu

Abstract. Most existing real-time fault-tolerant scheduling algorithms for heterogeneous distributed systems can achieve high reliability for non-preemptive and aperiodic tasks. However, the existing scheduling algorithms assume that status of each backup copy is either active or passive. To remedy this deficiency, we propose a novel reliability model tailored for preemptive periodic tasks. Next, we develop two real-time fault-tolerant algorithms (NRFTAHS and RDFTAHS) for heterogeneous distributed systems. NRFTAHS manages to assign tasks in a way to improve system schedulabilties, whereas RDFTAHS aims at boosting system reliability without adding extra hardware. Unlike the existing scheduling schemes, our algorithms consider backup copies in both active and passive forms. Therefore, our approaches are more flexible than the alternative ones. Finally, we quantitatively compare our schemes with two existing algorithms in terms of performability measured as a function of schedulability and reliability. Experiments results show that RDFTAHS substantially improves the overall performance over NRFTAH.

1 Introduction

With the development of high speed network and high performance computers, heterogeneous distributed systems have been widely applied for critical real-time systems, in which real-time and fault-tolerant abilities are two indispensable requirements.

To exploit high performances for real-time heterogeneous systems, much attention has been paid to real-time scheduling algorithms in context of heterogeneous systems. Ranaweer and Agrawal developed a scheduling scheme named SDTP for heterogeneous systems. Reliability costs was factored in some scheduling algorithms for tasks with precedence constrains [2][3]. Although schedulability is a main objective of these scheduling algorithms, the algorithms neither consider timing constraints nor support fault-tolerance. In addition, reliability models in these studies are geared to handle aperiodic, non-preemptive tasks.

Fault-tolerance, an inherent requirement of real-time systems, can be achieved in several approaches. One efficient fault tolerant technique, of course, is scheduling algorithms, among which the Primary-backup scheme plays an important role. In this approach, two versions of one task are scheduled on two different processors and an *acceptance test* is used to check the correctness of schedules [4,5,6]. The three variants of this scheme include active backup copy[4], passive backup copy[5], and primary backup copy overlapping[6]. Generally speaking, backup copy is always preferred to be executed as passive backup copy, because it can take the advantages of *backup copy overloading* and *backup copy de-allocation* technique to improve schedulability[4,5]. Primary backup copy overlapping technique is a tradeoff technique between the other two and can exploit the advantages of both the other two approaches[6].

Both active backup copies and passive backup copies have been incorporated into the Rate-Monotonic First-Fit assignment algorithm to provide fault-tolerance[7]. This scheme overcomes the drawbacks of timing constraints on backup copies to some extends. However, this scheduling algorithm neither considers heterogeneous systems nor takes system reliability into account.

Qin *et. al.* extensively studied real-time fault-tolerant scheduling algorithms based on heterogeneous distributed systems[8,9,10]. However, theses algorithms assume that status of each backup copy is either active or passive. Moreover, they only consider non-preemptive and aperiodic tasks.

Although numerous algorithms have been developed with respect to real-time fault-tolerant scheduling for distributed systems, to the best of our knowledge no work has been done on reliability-driven real-time fault-tolerant scheduling tailored for periodic tasks for heterogeneous distributed systems. In this study, a novel reliability model for real-time periodic tasks is proposed by extending the conventional reliability model designed for aperiodic tasks. In our approach, the primary backup copy approach is leveraged to tolerate single processor failures. Furthermore, two real-time fault-tolerant algorithms are devised for heterogeneous distributed systems. The first algorithm named NRFTAHS aims at assigning tasks in a way to improve schedulabilty of system, while the second algorithm termed as RDFTAHS employs the reliability measure as a major objective for scheduling real-time tasks. To quantify the combined metric of schedulability and reliability, the *Performability* measure is introduced. Finally, simulation experiments were conducted to compare the two algorithms in several aspects. The experiments results indicate that RDFTAHS performs significantly better than NRFTAHS with respect to reliability with marginal degradation in schedulability and, therefore, RDFTAHS substantially improves the overall performance over NRFTAH.

The paper is organized as follows: In section 2, a system model and assumptions are presented. Section 3 proposes reliability model for periodic tasks. Two novel real-time fault-tolerant scheduling algorithms are outlined in Section 4. Simulation experiments and performance analysis are presented in section 5. Finally, Section 6 concludes the paper by summarizing the main contribution of this paper and commenting on future directions of this work.

2 Systems Model

Our paper considers a typical heterogeneous distributed systems consisting of a set of tasks and a set of processors which are characterized as follows.

- A set of processors

  $\Omega = \{p_1, p_2, \ldots, p_M\}$
  $R = (\lambda_1, \lambda_2, \ldots, \lambda_M)$

Here, Ω is the processor set, p_i is the i-th processor and M is the total number of processor. All processors in the heterogeneous systems are connected by high-speed network. In this model, processor failures are assumed to be independent, and follow a Poison Process with a constant failure rate. R denotes the failure rates vector, wherein λ_i is the failure rate of p_i.

- A set of primary copy of real-time tasks

  $\Gamma = \{\tau_1, \tau_2, \tau_3, \ldots, \tau_N\}$
  $\tau_i = (C_i, T_i)\ (i = 1,2,\ldots,N)$

Here, Γ is the set of tasks, τ_i is the i-th task, and N is the number of tasks which are periodic, independent and preemptive. C_i denotes an execution time vector: $C_i = [c(i,1), c(i,2),\ldots,c(i,M)]$. Where $c(i, j)$ denotes the execution time of taskτ_i on processor p_j. T_i denotes the period of τ_i.

- A set of backup copy of real-time tasks

  $B\Gamma = \{\beta_1, \beta_2, \beta_3, \ldots, \beta_N\}$
  $\beta_i = (D_i, T_i)\ i = 1,2,\ldots,N$

Here, $B\Gamma$ is the set of backup copy of real-time tasks Γ. β_i is the corresponding backup copy of τ_i. Hence, D_i is the execution time vector of β_i. In our mode, it is assumed that backup copy and primary copy of a task are completely identical, that is: $D_i = [c(i,1), c(i,2),\ldots,c(i,M)]$. Correspondently, T_i denotes the period of β_i.

In our system model, as in [7], the backup copy has two statuses: *passive* backup-copy and *active* backup copy. When we assign a task, we assign its primary copy before assigning the backup copy. The status of backup copy is determined by the following:

$$Status(\beta_i) = \begin{cases} passive & T_i - R_{ij} > D_{ik} \\ active & T_i - R_{ij} \le D_{ik} \end{cases} \qquad (1)$$
$$where\ P(\tau_i) = P_j\ and\ P(\beta_i) = P_k$$

Here, R_{ij} denotes the WCRT(worst case response time) of τ_i which is assigned to P_j. For ease of presentation, γ_i represents a primary copy or a backup copy, namely, $\gamma_i = \tau_i$ or β_i.

To concentrate on our concerned problems, we make the following assumptions about failure characteristic of the hardware:

1. Hardware provides fault isolation mechanism, that is a faulty processor cannot cause incorrect behaviors in a non-faulty processors;
2. Processors fail in a fail-stop manner, which means a processor is either operational or cease functioning;
3. The failure of a processor is detected by the remaining ones within the closest completion time of a task scheduled on the faulty processor.

3 Reliability Model Based on Periodic Tasks

In this section, we attempt to address the issue of reliability for periodic tasks in heterogeneous systems. In [2], reliability is defined as the probability that the system can run an entire task set successfully. Furthermore, the definition of reliability costs is proposed. However, the definition of reliability costs is based on aperiodic, non-preemptive tasks set which is not applicable for periodic, preemptive tasks set. Because, usually, the periodic tasks will run periodically and will not cease until we force it to or external events. To solve this issue, we firstly analyze the characteristics of periodic tasks and the failure characteristic of heterogeneous processors, and then a definition of reliability costs based on periodic, preemptive tasks set is investigated.

It is worth noting that not only the periodic tasks are periodic in nature, but also the behavior of real-time tasks set as whole is periodic. The hyperperiod H can be seen as the period of the tasks set. H is defined as the least common factor of periods of all tasks sets, namely, $H = \mathrm{lcm}\{T_i|\tau_i \in \Gamma\}$).

It is also noted that the processor failures are assumed to be independent and follow a Poisson Process with a constant failure rate. Moreover, because the Poisson Process is a stable incremental process, the processors have equal fault probability during any equal time interval on the same processor. Thus, we can study the reliability of tasks set in a hyperperiod as the metric of reliability of the system.

To simplify our discussion without losing generality, we have the following assumptions:

1. If a processor fails when it is period, the failed processor will be replaced by a spare processor immediately. So, we do not consider it as a critical failure.
2. If a processor fails while the processor is working, we can use a spare processor and certain processor replacing mechanism(e.g., *FTRMFF-Replacing* presented in [7]) to recover the system to the non-faulty state after some time.

With the above two assumptions along with the single processor failure assumption, we can safely only consider the system reliability in fault-free scenario. Moreover, we only need to consider the effects of processors failure on tasks while processor is working. Therefore, we can redefine the system reliability based on periodic task set as the probability of system can run the entire task set in a hyperperiod while no critical failure occurs. Thus,

$$Reliability = \exp\Big(-\sum_{k=1}^{M} \Big(\sum_{P(\tau_i)=P_k} \lambda_k * C[i,k] * H/T_i + \sum_{P(\tau_i)=P_k}^{Status(\beta i)=active} \lambda_k * D[i,k] * H/T_i\Big)\Big) \qquad (2)$$

$$= \exp\Big(-H * \sum_{k=1}^{M} \lambda_k \Big(\sum_{P(\tau_i)=P_k} C[i,k]/T_i + \sum_{P(\tau_i)=P_k}^{Status(\beta i)=active} D[i,k]/T_i\Big)\Big)$$

According to the above definition of reliability, we can derive the definition of reliability costs based on our system model as.

Definition 1. When we assign both primary copy set Γ and backup copy set $B\Gamma$ of a set of real-time tasks on a set of heterogeneous processors Ω. The system Reliability-Costs is defined as follows:

$$Reliability\ Costs(\Gamma, B\Gamma, \Omega) = \sum_{k=1}^{M} \lambda_k \Big(\sum_{P(\tau_i)=P_k} C[i,k] \big/ T_i + \sum_{P(\beta i)=P_k}^{Status(\beta i)=active} D[i,k] \big/ T_i \Big) \tag{3}$$

Clearly, in order to increase the reliability of system we have to reduce the reliability costs as much as possible. The first item in the parentheses of (2) denotes the unreliability contributed by primary copy on their corresponding processors, while the second item is due to the active backup copy. This inspires us that allocating primary copy and active backup copy with greater load to more reliable processors might be good heuristic to decrease the reliability costs. Moreover, it is highly desirable to make as many backup copies as possible to be executed as passive status. Because, on one hand, it can increase the system schedulability, and can decrease the reliability costs on the other hand.

4 Proposed Scheduling Algorithms

In this section, we propose two algorithms for scheduling periodic tasks set along with their corresponding backup copy on a heterogeneous distributed system. The objective of the first algorithm, NRFTAHS, is to maximize the schedulability of the system and does no take reliability into account. By contrast, the other one, RDFTAHS, tries to minimize the total reliability costs of the system while retaining the schedulability of the system.

As [7], before task copies are assigned, both primary copy and backup copy are ordered by increasing period to simplify our algorithm. Thus, tasks are assigned to processors following the order:

$$\tau_1, \beta_1, \tau_2, \beta_2, \ldots, \tau_N, \beta_N \tag{4}$$

4.1 NRFTAHS

The main objective of the NRFTAHS is to maximize the schedulability of the system. The gists of NRFTAHS are:

1. Try to assign primary copy to a processor on which the execution time is shortest.
2. Try to schedule backup copy as a passive copy whenever possible.
3. If a backup copy has to be scheduled as an active backup copy, the algorithm endeavors to assign backup copy to processor on which the execution time is minimal.

Before presenting our algorithm, an execution time order vector for each task is introduced.

Definition 2. Given a set of real-time tasks Γ and a set of processors Ω. An execution time order vector for each task τ_i is defined as exec_order(i) = $[et_{i1}, et_{i2}, \ldots, et_{iM}]$, where et_{ij} ($1 \leqslant j \leqslant M$) is the processor number. exec_order(i) is sorted in the order of non-decreasing execution time of τ_i on processor set Ω, namely:

$$\forall i \in [1,N], \forall j,k \in [1,M] : (j < k \rightarrow (C[i, et_{ij}] \leq C[i, et_{ik}]))$$

The algorithm NRFTAHS is described as follow:

```
Algorithm: NRFTAHS
1) Initialization: reorder primary and backup copies following decreasing RM priorities as (4); generate execution time vector Exec_order(i) of each task i;
2) for i ←1,2,…, N do /* allocating processor to both primary and backup copies of N tasks*/
3)   found_active ← FALSE;
4)   for k ← 1, 2, …, M do
5)     for s←1, 2, …, M, s≠k do
         Step 5.1: Check the schedulability of $\tau_i$ and $\beta_i$ on processor exec_orer[i, k] and exec_orer[i, k], respectively. Determine the status of $\beta_i$ according to (1);
         Step 5.2: If both $\tau_i$ and $\beta_i$ can successfully scheduled on processor exec_orer[i, k] and processor exec_orer[i, s] respectively, and status($\beta_i$) = passive, then go to (2);
         Step 5.3: if both $\tau_i$ and $\beta_i$ can successfully scheduled on processor exec_orer[i, k] and processor exec_orer[i, s] respectively, status($\beta_i$) = active and found_active = FALSE, then found_active←TRUE, save the processor exec_orer[i, k], exec_orer[i, s] number to temporary variable;
6)     end for
7)   end for
8)   if (found_active = TRUE)
9)       Set P($\tau_i$), P ($\beta_i$) to the processor recorded in step 5);
10)  else
11)      ret_value ← FAIL; return ret_value;
12)  end if
13) end for
14) ret_value ← SUCCESS return ret_value;
```

4.2 RDFTAHS

RDFTAHS consider the heterogeneities of reliability costs, thereby improving the reliability of the system without extra hardware costs. The objective of RDFTAHS is to minimize the reliability costs. The gists of NRFTAHS are:

1. Try to assign primary copy to a processor on which the reliability cost is minimal.
2. Try to schedule backup copy as a passive copy whenever possible.
3. If a backup copy has to be scheduled as an active backup copy, the algorithm endeavors to assigned backup copy to processor on which the reliability costs is minimal.

Before presenting our algorithm, A reliability costs vector is defined.

Definition 3. Given a set of real-time tasks Γ, corresponding backup copy BΓ and a set of processors Ω. A reliability costs vector of τ_i and β_i is defined as $RCV_i = [rcv_{i1}, \ldots, rcv_{iM}]$, where $rcv_{ij} = (rc_{ij}, \rho_{ij})$. rcv_{ij} is derived from vector C_i and Ω. Namely, $rc_{ij} = (C[i, \rho_{ij}]/T_i) \times \lambda \rho_{ij}$. Elements in RCV_i is sorted in order of non-decreasing reliability costs, namely:

$$\forall i \in [1, N], \forall j, k \in [1, M] : (j < k \rightarrow (rc_{ij} \leq rc_{ik}));$$

The algorithm RDFTAHS is described as follow:

Algorithm: RDFTAHS

```
1) Initialization: reorder primary and backup copies following decreasing RM priorities as (4);generate execution time vector exec_order(i) and reliability costs vector $RCV_i$ for each task i;
2) for i ←1, 2, ……N do
3) found_active←FALS, tmp_rc←0 ;
4) for k ← 1, 2, …,M do
5) for s←1, 2, ……M, s≠k do
     Step 5.1: Assign primary copy $\tau_i$ to processor $RCV_i.\rho_{ik}$, which follows the order defined by reliability costs vector. Assign backup copy $\beta_i$ to processor exec_order(i, s), which follows the order defined by execution time order vector; Status of $\beta_i$ is determined by (1);
    Step 5.2: if both $\tau_i$ and $\beta_i$ are schedulable, and status($\beta_i$) = passive, then assign$\tau_i$ and $\beta_i$ to processor $RCV_i.\rho_{ik}$ and Exec_order(i, s), respectively; go to (2);
    Step 5.3: if both $\tau_i$ and $\beta_i$ are schedulable, and status ($\beta_{i)}$ = active;
       Step 5.3.1: if found_active = FALSE, then calculate the reliability costs of $\beta_i$ on processor exec_order(i, s), tmp_rc; found_active ← TRUE ; save processor $RCV_i.\rho_{ik}$ and exec_order(i, s) to temporary variable.
       Step 5.3.2: if found_active = TRUE, and the reliability costs of $\beta_i$ on processor exec_order(i, s) is smaller than tmp_rc, then set tmp_rc ← reliability costs of $\beta_i$ on processor exec_order(i, s); save processor $RCV_i.\rho_{ik}$ and exec_order(i, s) to temporary variable;
6)  end for
7) end for
8) if (found_active = TRUE)
9)     Set P($\tau_i$) and P($\beta_i$) with the processor number saved at Step 5.3; Set status($\beta_i$) ←active;
10) else ret_value ← FAIL; return ret_value;
12) end if
13) end for
14) ret_value ← SUCCESS; return ret_value;
```

5 Performance Evaluation

In this section, a number of simulations are carried out to evaluate the two algorithms proposed in the paper and compare them in several aspects. Three performance metrics are used to capture there aspects of real-time fault-tolerant scheduling. The first metric is Reliability Costs, defined in (3); The second one is *Schedulability*, defined to be the minimal number of processors(MNP) required by a certain number of tasks. In order to comprehensively measure the performance of our algorithms, we introduce a new metric, *Performability*, defined to be the product of the *Schedulability* and Reliability Costs. Formally:

$$Performability(\Gamma, B\Gamma, \Omega)=Schedulability\times Reliability\ Costs(\Gamma, B\Gamma, \Omega)$$

Here, *Schedulability* is the MNP of a set of primary copy Γ along with its corresponding backup copy set $B\Gamma$ scheduled by any algorithms proposed above. *Reliability Costs* is obtained by a scheduling Γ and $B\Gamma$ on MNP processors. Clearly, the smaller *Performability*, the better overall performances.

Our simulations are presented for large task sets with periodic tasks which are generated according to following parameters:

1) *Periods of tasks* (T_i)—a value generated randomly distributed in [0,500];
2) *Execution time of any task on any processor*(C[i, j])—a value taken from a random distribution in the interval $0<C[i, j] \leqslant \alpha T_i$, parameter $\alpha = \max_{i=1,\dots,N,j=1,\dots,M} C[i,j]/T_i$, which represents the maximum load occurring in the task set on all processors. Three values are chosen for α, namely, 0.2,0.5, and 0.8.
3) *Size of any task set*(L)—a value selected from a specific set, namely, [200, 400, 600, 800, 1000].

Besides, for the heterogeneous systems, the failure rate(FR) for each processor is uniformly selected between the range 0.95 to $1.05*10^{-6}$/hour(10^{-4})[11].

5.1 Reliability

In this experiment, the Reliability Costs of the two algorithms are evaluated. Fig.1 displays the Reliability Costs obtained by two algorithms as a function of size of task set. The number of processor (*Processor_Num*) is in proportional to the size of task set. Formally:

$$Processor\_Num = (L*15)/200$$

From Fig.1, it is clear that all values of Reliability Costs increase as L increase, as expected. For fixed task set, Reliability Costs also increase as α increase. This is because the bigger L or α, the more computation time are needed; therefore, the Reliability Costs increase. Most importantly, it is observed that RDFTAHS performs better NRFTAHS, in terms of reliability costs.

5.2 Schedulability

Another important metric for real-time fault-tolerant scheduling algorithms is *Schedulability*. Here, the *Schedulability* is defined as the minimal number of processors(MNP) to which all tasks, together with their corresponding backup copy, can be scheduled to finish before their specific deadlines. Therefore, we devise an algorithm, called Find Minimal Number of Processors, to find the MNP of a give task set[13].

Fig.2 illustrates the simulation results. Actually, RDFTAHS requires only one more processor than NRFTAHS in most cases. This result indicates that NRFTAHS is a little inferior to RDFTAHS.

5.3 Performability

In order to compare the overall performance, the third experiment is carried out to compare the two algorithms in terms of *Performability*.

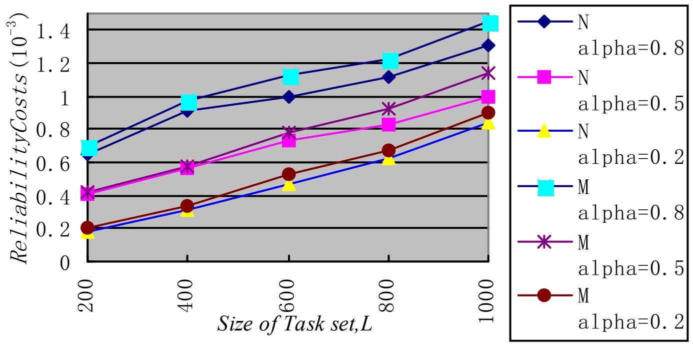

Fig. 1. Comparison of the Reliability Costs between NRFTAHS and RDFTAHS.

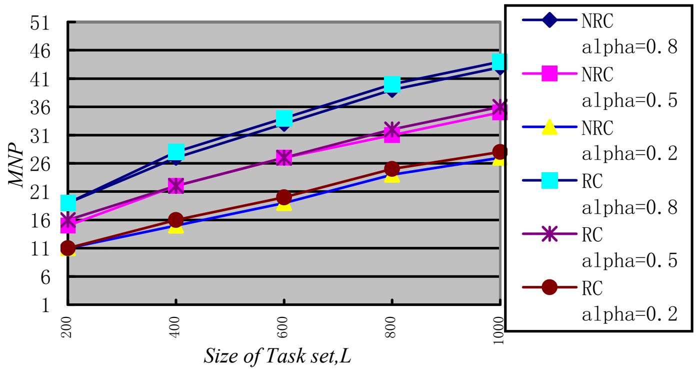

Fig. 2. Comparison of the MNP between NRFTAHS and RDFTAHS.

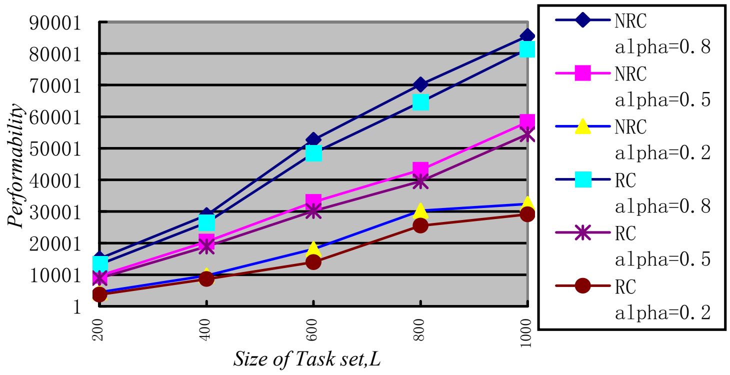

Fig. 3. Comparison of the *Performability* between NRFTAHS and RDFTAHS

Fig.3. reveals the Performability as a function of the size of task set. The number of processors used for each task set is MNP obtained in the previous experiment. As can be observed, RDFTAHS outperforms NRFTAHS considerably.

6 Conclusions

In this paper, we developed two real-time fault-tolerant scheduling algorithms for heterogeneous distributed systems and conduct extensive simulations about them in several aspects.Future studies in this area are two folders. First, we intend to study more efficient scheduling algorithms in the context of heterogeneous distributed systems. Second, we plan to extend our scheduling algorithms by incorporating precedence constrains and communication heterogeneities in distributed systems.

References

1. S. Ranaweera, and D.P. Agrawal. Scheduling of Periodic Time Critical Applications for Pipelined Execution on Heterogeneous Systems. In Proceeding of the 2001 International Conference on Parallel Processing, Spain, 2001
2. S. Srinivasan, and N.K. Jha. Safety and Reliability Driven Tasks Allocation in distributed Systems. IEEE Trans. Parallel and distributed systems, 10: 238-251, 1999
3. A. Dogan, and F. Ozguner. Reliable matching and scheduling of precedence-constrained tasks in heterogeneous distributed computing. In Proceeding of 29th International Conference on Parallel Processing, Spain, 2001.
4. C.H. Yang, G. Deconinck, and W.H. Gui, Fault-tolerant scheduling for real-time embedded control systems. Journal of Computer Science and Technology, 19:191-202, 2004.
5. H. Liu, and S.M. Fei. A Fault-Tolerant Scheduling Algorithm Based on EDF for Distributed Control Systems. Chinese Journal of Computers, 14:1371-1378, 2003.
6. A.l. Omari R, A.K. Somani, and G. Manimaran. An adaptive scheme for fault-tolerant scheduling of soft real-time tasks in multiprocessor systems. Journal of Parallel and Distributed Computing, 65: 595-608, 2005.
7. A.A Bertossi, L V Mancini, and F. Rossini. Fault-tolerant rate-monotonic first-fit scheduling in hard-real-time systems. IEEE Trans. Parallel and Distributed Systems, 10: 934-945, 1999.
8. X. Qin, Z.F Han, and L.P Pang. Towards Real-time Scheduling with Fault-tolerance in Heterogeneous Distributed Systems. Chinese Journal of Computers, 25: 121-124, 2002.
9. X Qin, and J. Hong, Dynamic, Reliability-driven Scheduling of Parallel Real-time Jobs in Heterogeneous Systems. In Proceeding of the 2001 International Conference on Parallel Processing, Spain, 2001.
10. X.Qin, J. Hong, and R.S. David, An Efficient Fault-tolerant Scheduling Algorithm for Real-time Tasks with Precedence Constraints in Heterogeneous Systems. In Proceeding of the 31st International Conference on Parallel Processing (ICPP), Canada, 2002.