INT J COMPUT COMMUN, ISSN 1841-9836
7(5):990-999, December, 2012.

A Fault-Tolerant Scheduling Algorithm using Hybrid Overloading Technology for Dynamic Grouping based Multiprocessor Systems

X.-B. Yu, J.-S. Zhao, C.-W. Zheng, X.-H. Hu

Yu Xing-biao
1.Institute of Software Chinese Academy of Sciences
4# South Fourth Street, Zhongguancun, Beijing, P.R. China
2.Graduate University of Chinese Academy of Sciences
80# Zhongguancun East Road, Haidian District, Beijing, P.R. China
E-mail: eliteman5317@hotmail.com

Zhao Jun-suo, Zheng Chang-wen, Hu Xiao-hui
Institute of Software Chinese Academy of Sciences
4# South Fourth Street, Zhongguancun, Beijing, P.R. China

Abstract:
In order to extend the application area of fault-tolerant scheduling algorithm based on hybrid overloading for multiprocessor and increase the fault-tolerant number of processors, we propose a new fault-tolerant scheduling algorithm, which is based on hybrid overloading and dynamic grouping for multiprocessor by combining logic grouping strategy for processors in primary backup overloading and backup backup overloading.This algorithm presents the formalization of the dynamic grouping for processors in fault-tolerant scheduling based on hybrid overloading and enlarges the task number included in overloading task link. In the process of fault-tolerant scheduling the processors are dynamically divided into some groups based on overloading task link, so as to keep good scheduling success ratio and enhance the fault-tolerant performance of processors. Both theoretical analysis and simulation experiment prove this algorithm's effectiveness respectively.

Keywords: dynamic grouping; fault-tolerant; overloading; primary backup; backup backup

1 Introduction

Real-time embedded system has been applied in many fields such as military, aeronautics, astronautics and communication, and the relevant research also has made important progress. The result of task scheduling in real-time system lies on not only scheduling correctness but also time restriction. The multiprocessor platform of real-time system takes advantage of the technology of resource redundancy and time redundancy in order to meet the demand of scheduling correctness and system reliability, among them primary-backup overloading(PB) [1, 2] and backup-backup overloading(BB) [2–4]approach is most important one. Supposing that there is just a processor fault in same period, the scheduling time of different versions of different tasks is overloaded in scheduling period of same processor for PB overloading , and the scheduling time of backup-backup versions of different tasks is overloaded in scheduling period of same processor for BB overloading. The fault-tolerant scheduling technology of hybrid overloading [2, 6–8] is the combination of PB overloading and BB overloading with better scheduling efficiency and fault-tolerant performance of processor, but still supposing there is only a processor fault in same period. Logic grouping strategy [1] for processor based on PB overloading and BB overloading dynamically divides processors into groups in the process of task scheduling so as to tolerate a processor fault in every group, which is more adaptable to apply practically. But this strategy yet supposes the task number of overloading task chain is not more than two tasks, therefore the assigning of grouping size and group number is not enough flexible to be suited for hybrid

overloading. In order to simplify the application of overloading method in scheduling algorithm, some basic application principles of overloading technology were introduced. [2]

In this paper Dynamic Grouping PB-BB Algorithm(DG_PB-BB Algorithm) combines the advantage of two overloading scheduling technology and logic grouping in the precondition of application principle for overloading technology, not only efficiently improving the efficiency of task scheduling but also enlarging tolerance area of the process. DG_PB-BB Algorithm has better application value than no grouping algorithm and logic grouping algorithm.

2 System Model

System is composed of m same processor nodes based on real time multiprocessor platform with shared memory, centralized dispatcher and processor communication medium without communication cost. Processor responsible for scheduling is dispatcher and responsible for execution is executer, which runs in parallel with dispatcher. New real time task is received by dispatcher and centralized dispatched to form assignment queue to execute on each processor.

Real time tasks to be scheduled are independent aperiodic and nonpreemptalbe scheduling with primary-backup copy technology. Each task t_i has two versions, namely primary copy(pr_i) and backup copy(bk_i) with identical attributes and resource requirements. Task sets $\tau = \{t_i | i = 1, 2, \cdots, n\}$, t_i=(a_i,r_i ,c_i,d_i), a_i is the start time of task t_i, r_i is the ready time of task t_i, c_i is the maximal execution time of task t_i, d_i is the deadline time of task t_i. Processor sets $\omega = \{p_i | i = 1, 2, \cdots, m\}$. $pr_i \rightarrow bk_i$ is task chain, if other tasks are overloading scheduled on task t_i, then it is overloading task chain. The parameters of system model are defined as follows:

Definition 1. $st(t_i)$ is the start scheduling time of task t_i. $ft(t_i)$ is the finish scheduling time of task t_i. $r_i \leq st(pr_i) \leq ft(pr_i) \leq st(bk_i) \leq ft(bk_i) \leq d_i$.

Definition 2. $proc(pr_i)$ is the processor on which the primary task is scheduled, $proc(bk_i)$ is the processor on which the backup task is scheduled, $proc(pr_i) \neq proc(bk_i)$.

Definition 3. $s(t_i)$ is the time interval on which the primary or backup task is scheduled. $s(pr_i) \cap s(bk_i) = \phi$.

Definition 4. $n_{cascade}$ is the cascade number of overloaded tasks within a processor group and a time slot. $groupsize(g_i)$ is the processor number of a processor group g_i. $n_{cascade}$=1 represents the scheduling assignment of no task overloading, $n_{cascade}$= groupsize(g_i) represents the task can not be overloading scheduled.$1 \leq n_{cascade} \leq groupsize(g_i)$.

Definition 5. $t_{overload}$ is the part time of a task overloaded on other tasks. $0 \leq t_{overload} \leq c_i$.

3 A Fault-tolerant Scheduling Algorithm Using Hybrid Overloading Technology for Dynamic Grouping Based Multiprocessor Systems(DG_PB-BB)

This paper states the strategy of dynamic grouping of fault-tolerant processor based on fault-tolerant scheduling technology of hybrid overloading and the finishing of task scheduling with the help of AP(Allocation Parameter) [5,6]algorithm. In DG_PB-BB Algorithm the system can tolerate more than two processor faults at same moment and the tasks of overloading chain can not again be limited as two numbers.

3.1 Validity checking

1. Backups can be overloaded on any task. Primaries can be overloaded only on backups.
2. If a primary pr_i is overloaded on a backup bk_j, then st(pr_i)≥ft(pr_j).
3. A overloading task chain should not be looped. A overloading task chain is relating to some processors and the maximal primary number of a overloading task chain is groupsize(g_i)-1 in the group g_i.
4. No more than one processor is belong to different processor groups. Every processor group has at least two processors.
5. The size of every processor group is possibly unequal and dynamically changes in the process of task scheduling.
6. The backups and primaries of same task are scheduled on same processor group. Overloading scheduling of task copy can only happen within same processor group.
7. If proc(pr_i)=proc(pr_j) within same processor group, then s(bk_i)∩s(bk_j)=ϕ, stating the scheduling time of task bk_i and bk_j can not be overloading.

3.2 Scheduling algorithm

Next specifically describing DG_PB-BB algorithm. In DG_PB-BB algorithm processor grouping is concluded as four conditions, formalizably described as following:

1. If the copy of task t_i, pr_i and bk_i are not overloaded on other tasks,then the processors occupied by two copies form a new processor group g_e.
   g_e=form_group(proc(pr_i),proc(bk_i)).
2. If pr_i is not overloaded on other tasks, but bk_i is, then the task chain in which task t_k overloaded on bk_i is situated links with the task chain in which $pr_i \rightarrow bk_i$ is situated to extend as a new processor group g_e.
   (proc(pr_k),proc(bk_k))∈group(g_e),
   expand_group(g_e,proc(pr_i)).
3. If bk_i is not overloaded on other tasks, but pr_i is, then the task chain in which task t_k overloaded on pr_i is situated links with the task chain in which $pr_i \rightarrow bk_i$ is situated to extend as a new processor group g_e.
   (proc(pr_k),proc(bk_k))∈group(g_e),
   expand_group(g_e,proc(bk_i)).
4. If both of pr_i and bk_i is overloaded on other tasks, then the task chain in which task t_k overloaded on bk_i is situated links with the task chain in which task t_j overloaded on pr_i is situated to extend as a new processor group g_e.
   (proc(pr_j),proc(bk_j))∈group(g_e),
   (proc(pr_k),proc(bk_k))∈group(g_f),
   g_e=expand_group(g_e,g_f).

Fig.1 states four conditions of dynamic processor grouping. OTC is overloading task chain. OTC(A) represents task chain $pr_1 \rightarrow bk_1$ and the processors occupied by it form a new processor group, which is the first condition of dynamic processor grouping. Within OTC(B_1), the task chain t_4 is situated in links with $pr_3 \rightarrow bk_3$ task chain and the processors occupied by it form

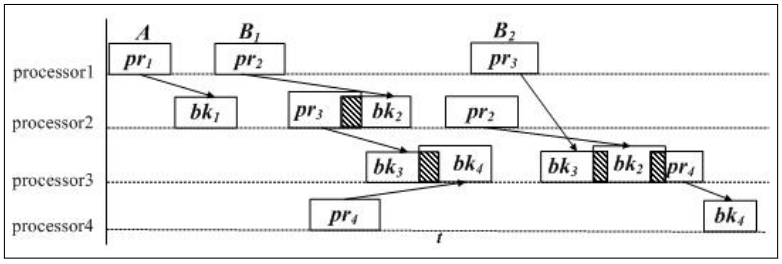


Figure 1: dynamic processor grouping

a new processor group, showing the second condition of dynamic processor grouping. Within OTC(B_2), the task chain t_4 is situated in links with task chain $pr_2 \rightarrow bk_2$ and the processors occupied by it form a new processor group, expressing the third condition of dynamic processor grouping. Within OTC(B_1), task chain $pr_3 \rightarrow bk_3$ links with both task chain t_2 and t_4, and the processors occupied by it form a new processor group, which is the fourth condition of dynamic processor grouping.

DG_PB-BB algorithm is based on AP scheduling strategy of hybrid overloading. New tasks form ready scheduling queue according to FCFS(First Come First Server) method. Bigger The distance of start scheduling time between primaries and backups, smaller the effect of backups to task receiving ratio, and it means start scheduling time of primaries is as early as possible, st(pr_i)$\rightarrow r_i$, finish scheduling time of backups is as late as possible, ft(bk_i)$\rightarrow d_i$. In order to increase task receiving ratio, tasks contained in task sets of hybrid overloading should be as much as possible and scheduling time occupied by them should be as few as possible. In the process of task scheduling the strategy of dynamic processor grouping is adopted to enhance tolerant level of the processor. The principle of processor grouping is the processors occupied by every overloading task chain constitute one group, if new overloading task chain has no relationship with old overloading task chain, then it should add a new processor to schedule new overloading task chain again to form a new processor group.

If e=groupsize(g_e),then:

$$AP[pr_i, p_j, ft(pr_i)] = \begin{cases} \frac{d_i - ft(pr_i)}{d_i - r_i} \frac{1}{e} & n_{cascade} = 1 \\ \frac{d_i - ft(pr_i)_i}{d_i - r_i} \frac{n_{cascade}}{e} \frac{t_{overload}}{c_i} & 1 < n_{cascade} \leq e \end{cases}$$

$$AP[pr_i, p_j] = \max\{AP[pr_i, p_j, ft(pr_i)]\}$$

$$AP[bk_i, p_j, st(bk_i)] = \begin{cases} \frac{st(bk_i) - r_i}{d_i - r_i} \frac{1}{e} & n_{cascade} = 1 \\ \frac{st(bk_i) - r_i}{d_i - r_i} \frac{n_{cascade}}{e} \frac{t_{overload}}{c_i} & 1 < n_{cascade} \leq e \end{cases}$$

$$AP[bk_i, p_j] = \max\{AP[bk_i, p_j, st(bk_i)]\}$$

AP[pr_i,p_j,ft(pr_i)] is evaluation factor of scheduling scheme that a new primary pr_i is assigned to schedule on processor j. similarly,AP[bk_i,p_j,ft(bk_i)] is evaluation factor of scheduling scheme that a new backup bk_i is assigned to schedule on processor j. AP is bigger, showing that the scheme of scheduling assignment is more optimal and scheduling efficiency is better.

DG_PB-BB Algorithm

1. The copy of first task, pr_i and bk_i are assigned respectively processor p_1 and p_2 to form a new group g_e. g_e is present processor group, G is assigned processor group. schedule(pr_i)$\rightarrow p_1$,schedule(bk_i)$\rightarrow p_2$,g_e=form_group(p_1,p_2),validity(),G=g_e.

2. **(1)** Within assigned processor group G, next task t_i finishes scheduling with the adoption

of scheduling technology of hybrid overloading and allocation parameter.
for task t_i,
while validity()$\neq$ failed
AP(pr_i,p_{j1})=next[max(AP(pr_i,p_j))]or AP(bk_i,p_{j2})=next[max(AP(bk_i,p_j))],
$p_j \in$ group(G),$j = 1, 2, \cdots, m$,$j_1 \neq j_2$,
schedule(pr_i)$\rightarrow p_{j1}$,schedule(bk_i)$\rightarrow p_{j2}$,validity().

(2) If task t_i can not be scheduled within assigned processor group, then it should be to add a new processor into assigned processor group G to schedule task t_i again and judge whether new and old overloading task chain can be united according to four conditions. Otherwise to judge whether task chain $pr_i \rightarrow bk_i$ and task chain included in processor group G are separated each other, and whether processor group g_e formed by task chain in which task t_i is situated is really contained in group G. If it is true, then group g_e should be decomposed, otherwise task t_i is dealt with according four conditions. After processors are assigned completely, if new task can not be scheduled, then fault-tolerant scheduling algorithm of dynamic grouping is started again according to principle of processor grouping.

if schedule(t_i)=failed in G
then {for new processor p_k, G=$p_k \cup G$,go to (1) (constraint p_k=$p_{j1} \vee p_k$=p_{j2})
if (combine(link($pr_i \rightarrow bk_i$),(group(G) $\rightarrow$ $link$))=failed)

then condition 1,validity()
else condition i,validity(),go to (1) until $j_1 \vee$ j_2=m}
else
{if ((combine(link($pr_i \rightarrow bk_i$),(group(G) $\rightarrow$ $link$))=failed) and
$form_group(pr_i, bk_i) \in$group(G))

then decompose(form_group(pr_i, bk_i))
else condition i,validity(),go to (1) until $j_1 \vee j_2$=m}

Next giving an example of DG_PB-BB algorithm to express the thinking of algorithm.T_1=(2, 2,2,7.5),T_2=(4,4,2,8),T_3=(4.5,4.5,2,9),T_4=(5,5,2,10),T_5=(5,5,4,13), T_6=(6,6,3.5,14.5),m=6. Fig.2 describes DG_PB- BB algorithm scheduling example of above queue.

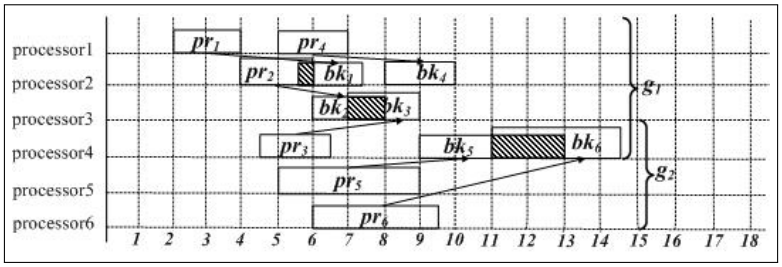


Figure 2: DG_PB-BB algorithm

Firstly m=2, processor p_1 and p_2 form group g_1.St(pr_2)=4 ,m=3 ,proc(bk_2)=p_3, pr_2 and bk_1 can be PB overloading ,p_3 is added into group g_1.st(pr_3)=4.5,if m=3, pr_3 and bk_3 can not be scheduled within group g_1, then m=4, bk_3 and bk_2 are BB overloading, proc(pr_3)=p_4, p_4 is added into group g_1. st(pr_4)=5,m=4, pr_4 and bk_4 can be scheduled within group g_1,proc(pr_4)=p_1,

proc(bk_4)=p_2,and task chain $pr_4 \rightarrow bk_4$ is separated from old overloading task chain within group g_1 and really included in group g_1, so that task chain $pr_4 \rightarrow bk_4$ is combined into group g_1. st(pr_5)=5,m=4,pr_5 and bk_5 can not be scheduled within group g_1, so it is to extend a new processor p_5 and rescheduling pr_5 and bk_5 within processor scheduling sets composed of p_5 and group g_1, proc(pr_5)=p_5, proc(bk_5)=p_4.As a result, task chain $pr_5 \rightarrow bk_5$ is separated from old overloading task chain within group g_1 and not really included in group g_1, therefore it is to extend task chain $pr_5 \rightarrow bk_5$ as group g_2. st(pr_6)=6, task t_i can not be scheduled neither in group g_1 nor group g_2, so that a new processor p_6 is added into group g_2 and task t_i is rescheduled within extended group g_2, proc(pr_6)=p_6, proc(bk_6)=p_4, bk_5 and bk_6 is BB overloaded on processor p_4. All processors have been assigned completely so that DG_PB-BB algorithm is restarted from processor p_1 while there are new tasks arriving into scheduling queue.

3.3 Algorithm analysis

If task sets showed in Fig.2 are scheduled by AP scheduling algorithm of hybrid overloading without grouping(PB-BB_AP Algorithm),although it only needs four processors to finish scheduling, but can just tolerate a processor fault and is too strict to apply widely. DG_PB-BB algorithm needs to increase two processors to finish scheduling, but new processor group g_2 in system makes fault-tolerant number of processor extend as two processors so as to strengthen the reliability of system. Time complexity of PB-BB_AP algorithm is $O[N^2 \cdot m \cdot (m-1)]$, N is average task number of task sets on which a processor has ever scheduled, m is processor number of the system. If in DG_PB-BB algorithm average processor number of group g_e is k, then time complexity of DG_PB-BB algorithm is $O[N^2 \cdot k \cdot (k-1)]$.In regard to DG_PB-BB algorithm, comparing with PB-BB_AP algorithm, the algorithm cost decreases and the system reliability increases, but the guarantee ratio decreases, because PB-BB_AP algorithm is an ideal method.

3.4 Theory testification

The theory testification follows the methodology used in [8] and is based on the following assumptions:

- All tasks have unit worst execution time, for example: c_i=1.
- Backup slots are preallocated in the schedule.
- FIFO scheduling strategy is used.
- Task deadlines follow uniform distribution [W_{min},W_{max}],called deadline window. If $P_{win}(w)$ is the probability that an arriving task has a relative deadline w, then $P_{win}(w) = 1/(W_{max} - W_{min} + 1), W_{min} \leq w \leq W_{max}$.
- Task arrivals follow uniform distribution[0,A_{max}], with mean A_{av}=$A_{max}/2$. If $P_{ar}(k)$ is the probability of k tasks arriving at a given time, then $P_{ar}(k) = 1/(A_{max}+1), 0 \leq k \leq A_{max}$.

A simple pre-allocation policy for BB overloading is to reserve a slot for backups every n time slots on each processor. Backup slots on the three processors can be staggered. For a task t_i, bk_i is scheduled immediately after pr_i with probability 0.5 and is scheduled two slots later than pr_i with probability 0.5.

In PB overloading there are three different types of time(0,1 and 2), if (t-1)mod 3=i, any time t has a type of i. At any time t, the number of primaries that can be scheduled to start at that time is s_0 if t is of type 0, s_1 if t is of type 1, s_2 if t is of type 2.

Using FIFO scheduling is equal to maintaining a task queue, to which arriving tasks are appended. Given that the number of task that can be scheduled on each time unit is known, then the position of a task in the Q indicates its scheduled start time. In BB overloading two tasks can be scheduled on each time (one slot is reserved for backups). If at the beginning of time slot t, a task t_i is the qth task in Q, then t_i is scheduled to execute at time slot $t+g_q^{BB}$. g_q^{BB} is the time at which a task, whose position in the Q is $q\,(q=1,2,\cdots,2W_{\max})$, will be executed and is defined as $g_q^{BB}=\lfloor\frac{q}{2}\rfloor$. In PB overloading s_0,s_1 and s_2 tasks can be scheduled on a given time slot t depending on whether t is of type 0,1,or 2 respectively. The time g_q^{PB} is defined as g_q^{PB} =(i+j+l),$\sum_{c=1}^{i} s_0+\sum_{c=1}^{j} s_1+\sum_{c=1}^{l} s_2 \le q-1, |i-j|\le 1, |j-l|\le 1, |l-i|\le 1$. where $i\ge j\ge l$ if t is of type 0, $j\ge l\ge i$ if t is of type 1, and $l\ge i\ge j$ if t is of type 2. When a task t_i arrives at time t, its schedulability depends on the length of Q and on the relative deadline w_i of the task. In BB overloading, if t_i is appended at position q of Q and $w_i\ge g_q^{BB}$, then the primary task pr_i is guaranteed to execute before $t+w_i$, Moreover, if $w_i\ge g_q^{BB}+2$, then bk_i is also guaranteed to execute before time $t+w_i$. In PB overloading, if t_i is appended at position q of Q and $w_i\ge g_q^{PB}$, then the primary task pr_i is guaranteed to execute before $t+w_i$, Moreover, if $w_i\ge g_q^{PB}+2$, then bk_i is also guaranteed to execute before time $t+w_i$. Let $p_{q,k}$ be the probability that one of the k tasks is rejected when the queue size is q, and its value is the probability that the relative deadline of the task is smaller than $g_b^*+\delta$,*=PB or BB, δ=1 or 2.

g_b^{logic} is the time at which a task, whose position in the Q is b, will be executed in the PB-BB overloading scheduling strategy of logic grouping, $b=q+k/2$. Showing as t_0-t_{12} in Fig.3, processor p_1-p_{12} are divided into 4 groups and every group has 3 processors, adopting with scheduling strategy of BB overloading, processor p_{13}-p_{24} are also divided into 4 group and every group has 3 processors, adopting with scheduling strategy of PB overloading. Above method is described as PB-BB overloading scheduling strategy of logic grouping.

$$g_b^{logic}=\frac{q+k/2}{\left\lfloor\frac{(i+j+l)(n-n/3)+\sum_{c=1}^{i} s_0+\sum_{c=1}^{j} s_1+\sum_{c=1}^{l} s_2}{}\right\rfloor},$$

$$\sum_{c=1}^{i} s_0+\sum_{c=1}^{j} s_1+\sum_{c=1}^{l} s_2 \le q+k/2-1, |i-j|\le 1, |j-l|\le 1, |l-i|\le 1.$$

$g_b^{dynamic}$ is the time at which a task, whose position in the Q is b, will be executed in the PB-BB overloading scheduling strategy of dynamic grouping, $b=q+k/2$. Different with logic grouping, in dynamic grouping s_0=3, s_1=4 and s_2=2. Describe as t_{12}-t_{24} in Fig.3, processor p_1-p_{12} is divided into 3 groups and every group has 4 processors, adopting with scheduling strategy of BB overloading, processor p_{13}-p_{24} are also divided 3 groups and every group has 4 processors, adopting with scheduling strategy of PB overloading, Above method is described as PB-BB overloading scheduling strategy of dynamic grouping.

$$g_b^{dynamic}=\frac{q+k/2}{\left\lfloor\frac{(i+j+l)(n-n/4)+\sum_{c=1}^{i} s_0+\sum_{c=1}^{j} s_1+\sum_{c=1}^{l} s_2}{}\right\rfloor}, \sum_{c=1}^{i} s_0+\sum_{c=1}^{j} s_1+\sum_{c=1}^{l} s_2 \le q+k/2-1, |i-j|\le 1, |j-l|\le 1, |l-i|\le 1.$$

Obviously, $g_b^{dynamic}<g_b^{logic}$, $g_b^{dynamic}$ decreases more quickly than g_b^{logic} with increasing n, therefore DG_PB-BB algorithm is more efficient than LG_PB-BB algorithm with increasing n.

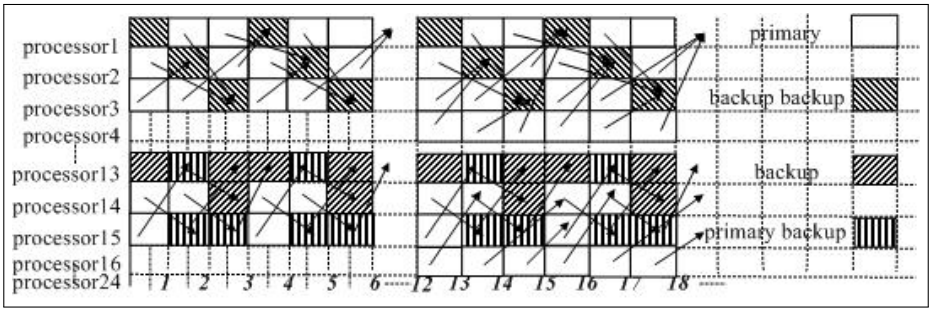


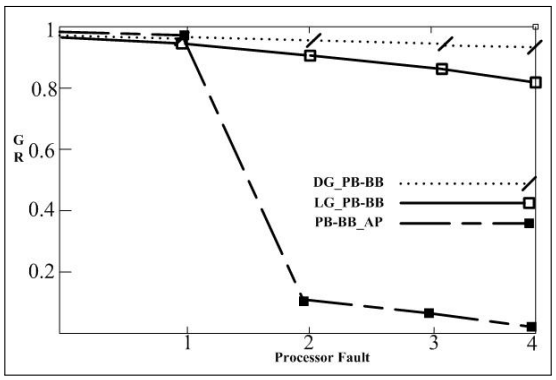
Figure 3: theroy testification of LG_PB-BB and DG_PB-BB

4 Simulation Experiment

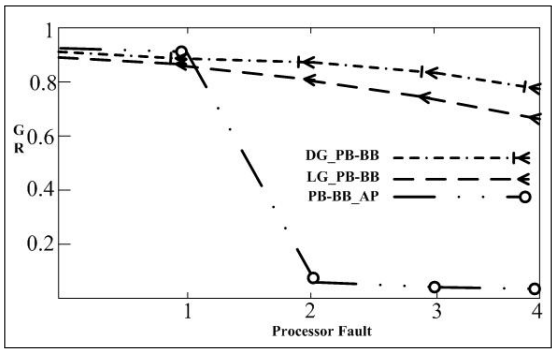
Supposing LG_PB-BB represents fault-tolerant scheduling algorithm of hybrid overloading based on the strategy of logic grouping. Simulation experiment mainly compares with change situation of the guarantee ratio(GR) in variety of fault-tolerant number of processor in circumstance of different task load for DG_PB-BB, LG_PB-BB and PB-BB_AP algorithm respectively. The guarantee ratio=the number of tasks guaranteed/the number of tasks arrived. Simulation parameters as following:

- The inter-arrival time of tasks follows exponential distribution with mean θ. θ=8.
- The inter-arrival time of faults follows exponential distribution with mean λ. λ=200.
- The execution time of a task is chosen uniformly [2,8].
- The deadline of a task is chosen uniformly $[r_i + c_i, r_i + R * c_i]$, where$R \geq 1$.
- Processor number m=10, task number n=50.
- R(task laxity) represents flexibility time task t_i can stay in ready queue in precondition of finishing scheduling before deadline. R=3.
- L (task load) is the expected number of task arrivals per mean service time. $L = C/\theta$, C is the mean execution time, θ is the inter-arrival average time of tasks t_i. Bigger L is, more the average load of processor is and lower the guarantee ratio is.

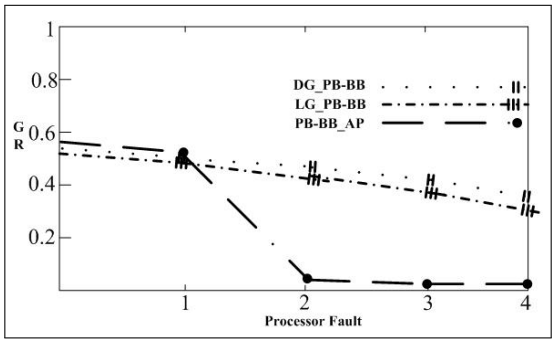
In Fig.4 (a),(b) and (c) respectively show the relationship of processor fault and guarantee ratio in DG_PB-BB,LG_PB-BB and PB-BB_AP algorithm for the system of L=0.25,L=0.5 and L=1. Experiment results prove GR decreases along with L and processor faults increase. When there is only one processor fault in DG_PB-BB,LG_PB-BB and PB-BB_AP algorithm, the differences of GR for the system of L=0.25,L=0.5 and L=1 are small. When processor fault increases to more than two faults, GR for three kinds of task load in PB-BB_AP algorithm are all low and failure possibility of task scheduling is high. When L=0.25 and L=0.5 in DG_PB-BB algorithm GR enhances significantly comparing with LG_PB-BB algorithm, and for the system of L=1 the difference of GR between two algorithm is small, stating in the system of not full load task DG_PB-BB algorithm can tolerate processor fault better than LG_PB-BB and PB-BB_AP algorithm.



(a) L=0.25



(b) L=0.5



(c) L=1

Fig.4. comparison with dynamic grouping, logic grouping and no group algorithm

5 Conclusions

This paper improves task number and grouping strategy included in overloading task chain and proposes DG_PB-BB algorithm based on PB-BB_AP algorithm according to logic grouping strategy of PB overloading and BB overloading. Simulation experiment shows DG_PB-BB algorithm not only has good guarantee ratio of task scheduling, but also improves fault-tolerant level of processor, with better application valuation.

Main creative achievements include 1)introducing the formalization of processor dynamic grouping in fault-tolerant scheduling technology of hybrid overloading, 2)proposing the method of processor dynamic grouping based on overloading task chain,and 3)extending task number included in overloading task chain and increasing fault-tolerant level of processor by the adoption of processor dynamic grouping.

Bibliography

[1] R.Al-Omari,Arun K.Somani,G.Manimarna,Efficient overloading techniques for primary-backup scheduling in real-time systems, *J.Parallel and Distributed Computing*,64:629-648,2004.

[2] Wei Sun,Naixue Xiong,Laurence T.Yang,Chunming Rong, Towards free task overloading in passive replication based real-time multiprocessors, *10th IEEE International Conference on Computer and Information Technology*, 1735-1742, 2010.

[3] Bindu Mirle,Albert M.K.Cheng, *Simulation fault-tolerant scheduling on real-time multiprocessor systems using primary backup overloading*, University of Houston, 1-10,2006.

[4] R.Al-Omari,Arun K.Somani,G.Manimarna, An adaptive scheme for fault-tolerant scheduling of soft real-time tasks in multiprocessor systems, *J.Parallel and Distributed Computing*, 65:595-608, 2005.

[5] W.Sun,Y.Zhang,C.Yu,X.Defago,Y.Inoguchi, Dynamic scheduling real-time task using primary-backup overloading strategy for multiprocessor systems, *IEICE Transactions on Information and Systems*, 796-806, 2008.

[6] W.Sun,Y.Zhang,C.Yu,X.Defago,Y.Inoguchi,Real-time task scheduling using extended overloading technique for multiprocessor system, *11th IEEE Symposium on Distributed Simulation and Real-time Applications*, 95-102, 2007.

[7] W.Sun,Y.Zhang,C.Yu,X.Defago,Y.Inoguchi,Hybrid overloading and stochastic analysis for redundant scheduling in real-time multiprocessor systems, *26th IEEE International Symposium on Reliable Distributed Systems*, 265-274, 2007.

[8] G. Manimaran, C. Siva Ram Murthy,A fault-tolerant dynamic scheduling algorithm for multiprocessor real-time systems and its analysis, *IEEE Trans. Parallel Distributed System* , 9(11):1137-1152, 1998.