TMR 容错计算故障恢复技术研究*)

李海山[1,2] 徐火生[2] 欧中红[2] 袁由光[2]

(哈尔滨工程大学计算机科学与技术学院 哈尔滨 150001)[1] (中船重工武汉 709 所 武汉 430074)[2]

摘 要 提出了一种可伸缩的 TMR 容错计算系统结构,根据 TMR 系统出现故障的情况,详细研究了其故障恢复模型和恢复策略。通过综合采用向前和向后恢复方法有效减少了由于实现容错功能而对系统运行进程完成时间的推延并进行了定量分析和验证。

关键词 向前恢复,容错计算,故障恢复模型和策略,系统性能

Research on Fault Recovery Techniques for TMR Fault-tolerant Computing

LI Hai-shan[1,2] XU Huo-sheng[2] OU Zhong-hong[2] YUAN You-guang[2]

(Institute of Computer Science and Technology, Harbin Engineering University, Harbin 150001, China)[1]

(Wuhan 709th Institute of CSIC, Wuhan 430074, China)[2]

Abstract The system architecture of a kind of scalable fault-tolerant computer with triple module redundancy is proposed. Detailed study on its fault recovery model and scheme corresponding to different fault occurrence situations is done. Through synthetically adopting forward recovery and roll-back recovery, the impact on the completion time delay of system processes due to the demand on fault tolerance can be efficiently cut down, which is quantitatively analyzed and validated.

Keywords Forward recovery, Fault-tolerant computing, Fault recovery model and scheme, System performance

1 引言

一个容错系统包括四个要素[1,3]:首先是故障检测,这是容错系统必不可少的环节,是其它环节的基础;其次是对出现的故障所造成的影响进行评估并限制其传播;三是对确定为不可恢复的故障进行处理,即重组,构成一个可工作的子系统;最后是对系统进行恢复,使系统回到正常状态,继续工作。

在检测出故障并利用重组处理了永久故障后,动态冗余的最后一步就是使系统恢复到正常的工作状态。恢复是处理检测到的可恢复故障或发生永久故障的可热拔插(或有备份替换)系统的重构、消除故障造成的影响并使系统继续运行的重要环节。只有在恢复环节之后,系统容错的目的才能得以完全实现。恢复通常使用特殊的软件实现,但需要有硬件机构的支持。

本文提出了一种可伸缩的 TMR 容错计算系统结构,根据 TMR 系统出现故障情况详细研究了其故障恢复模型和恢复策略,通过综合采用向前与向后恢复方法,有效减少了由于实现高可靠性而对系统性能的影响,并进行了定量分析和验证。

2 TMR 容错计算体系结构

本文以图 1 所示 TMR 容错计算机系统结构为蓝本,研究、讨论 TMR 容错计算故障恢复模型及恢复策略。图中虚线框由三台同构物理计算机 $PE_i(i=1,2,3)$组成 TMR 系统,对用户而言则是一台逻辑计算机。各 $PE_i(i=1,2,\cdots,n)$均由 PC_i、双端口网络控制器 NIC_i、容错管理模块 FMM_i(Fault-tolerant Management Module)组成。各 FMM_i 由高速千兆专用网 PHN(Proprietary High-speed Network)连接,PC_i 为符合 PICMG 2.0 R2.1 CompactPCI 规范的商用或兼容 PC 机,网络接口为 100/1000Mbps 自适应。FMM_i 最主要的任务就是通过软硬件相结合实现 PE_i 间同步、数据表决、故障检测、屏蔽、恢复等容错功能,通过 CPCI 与相应的 PC_i 通信,它是 TMR 的关键部件。为了满足实时需求,FMM_i 被设计为具备较强处理能力、通讯能力的智能容错模块,在 PE_i 执行任务的同时可并行进行容错处理。

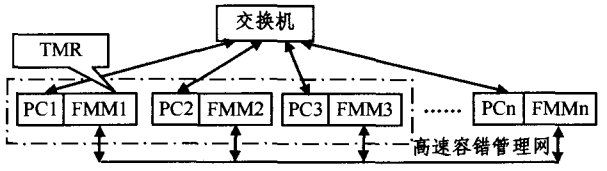

图 1 三模冗余 TMR 系统结构示意图

TMR 中相同的用户任务或进程(以后不加区别,均以进程代替)复制在三个模块上同步执行(同步控制器及同步策略、方法等同步方面的问题不是本文研究的内容,在此不作详细分析,可参见文献[5]),各模块的输出经过表决以择多原则表决后取认为正确的作为系统输出。

3 术语及符号

下面描述在本文中将用到的术语和符号。

检查点间隔[6-8]:上一个检查点完成(进程状态保存和比较)时刻与下一个检查点完成时刻之间的时间间隔,用 $I_j(j=$

*) 本文受国防重点预先研究项目(413160201)资助。**李海山** 博士研究生,主要研究方向为容错技术与可信计算、分布计算;**袁由光** 研究员,博士生导师,主要研究方向为计算机系统机构、可信计算。

$0,1,\cdots,n$)表示，I_0 表示进程开始执行点，I_n 表示进程执行结束点，I_j 和 I_{j+1} 表示相邻的两检查点。

检查点机制：为了在应用程序或硬件模块出现故障时能有效的恢复，在应用程序执行过程中的某些点进行的进程状态保存，比较、故障恢复。

有效执行时间：应用程序完成其有效功能所用时间即在无故障，没有检查点机制时应用程序的执行时间。

A,B,C 代表三个模块或运行在三个模块上的同一进程的三个复制(后面描述中不再区别，统称为模块)。

t_0 表示检查点间隔 I_j 的起点也即检查点间隔 I_{j-1} 的终点。

CP_{aj}，CP_{bj}，CP_{cj} 分别表示 A,B,C 三个模块在第 j 个检查点间隔 I_j 的终点设置的检查点，分别称为 A,B,C 的第 j 个检查点。

$CP_{xj}=CP_{yk}$ ($x,y\in\{A,B,C\}$，$x\neq y$)表示模块 x 的第 j 个检查点与模块 y 的第 k 个检查点一致。在本文研究的 TMR 中由于各模块同步运行，因此只需考虑 $j=k$ 的情况。

CP_j 表示系统的第 j 个检查点，即检查点间隔 I_j 结束时三模表决认为正确的检查点。

n 表示进程运行过程中设置的等距检查点数。

t_{ch} 表示检查点设置开始到结束并比较时间。

t_{cp} 表示模块间的检查点拷贝并恢复时间。

t_r 表示进程状态恢复到前一个检查点的时间。

t_u 表示一个检查点间隔内应用程序有效执行时间。

T_u 表示应用程序有效执行时间，即 $T_u=nt_u$。

t_{α} 表示备份模块检查点设置并与工作模块相应检查点比较时间。

t_{σ} 表示工作模块的正确检查点拷贝到备份模块并在备份模块上从检查点处开始启动运行的时间。

t_w 表示工作模块等待与备份模块的表决结果时间，$t_w=\max(t_{\sigma}+t_{\alpha}-t_{ch},0)$。

T 表示检查点间隔时间，$T=t_u+t_{ch}$。

$P(x)$表示发生事件 x 的概率。

设 τ_j 表示执行最后 j 个检查点间隔所需时间，$0\leqslant j\leqslant n$，则 τ_n 就是进程从开始到结束的整个执行时间，τ_{n-1} 表示进程执行完第一个检查点后到结束还需要运行的时间，$\tau_0=0$ 则为进程结束时刻。

$\overline{\tau_j}$ 表示 τ_j 的数学期望，即 $\overline{\tau_j}=E(\tau_j)$。

4 TMR 容错计算故障恢复模型及策略

故障恢复模型及恢复策略依赖于三个模块故障的发生情形。为了描述简洁方便，本文分析时做如下假设：

1. TMR 周期性地对进程做检查点设置，在此我们以等距检查点讨论。

2. 两个模块不会出现相同的故障状态。

3. 各模块在检查点设置结束后立即把检查点提交表决，即表决比较检查点设置时保存的进程状态。通过择多原则，定位故障；

4. 每个模块的故障分布遵从失效率为 λ 的泊松分布。

4.1 故障恢复模型及恢复策略描述

在检查点设置时刻，各模块进行进程状态保存，记此检查点间隔为 I_j，完成后立即提交表决器表决(可是完整的进程状态表决，也可是进程状态的特征值表决)。

情形一：三个模块无故障。三模表决一致即 $CP_{aj}=CP_{bj}=CP_{cj}$，各模块仍按正常时序继续运行。

情形二：一个模块发生故障，不失一般性设 A 模块故障，其它正常，此情况下采用故障前向恢复法。检查点间隔 I_j 结束后，三模表决结果不一致即 $CP_{aj}\neq CP_{bj}=CP_{cj}$，通过把各模块提交的表决数据分别与表决结果比较，检测并定位出故障模块 A，启动故障恢复机制——把 CP_{bj} 或 CP_{cj} 拷贝到模块 A，用正确运行状态替换模块 A 的故障状态，使其恢复到正确轨道，如图 2 所示，A,B,C 同步继续向前运行，开始第 $j+1$ 个检查点间隔的执行。

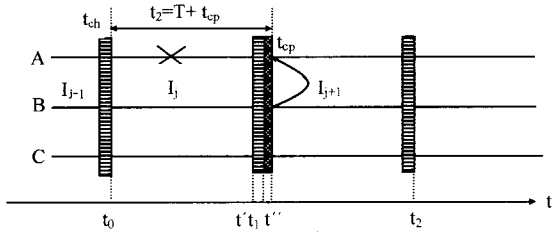

图 2 单模故障

若比较发现 $CP_{aj}\neq CP_{bj}\neq CP_{cj}$，说明在一个检查点间隔内有两个或三个模块同时故障，此时把检查点 CP_{j-1} 拷贝给 PE_i ($i>3$)(称为 D 模块)，通过该检查点保持的进程状态，在 D 上执行该进程的检查点间隔 I_j，同时 A,B,C 继续执行检查点间隔 I_{j+1}，D 在 I_j 结束时设置检查点 CP_{dj}，通过 CP_{dj} 与 CP_{aj}，CP_{bj}，CP_{cj} 比较来定位故障并根据故障发生情况采取不同恢复策略 —— 向前或向后卷回恢复。此种情况下分如下几种情形。

情形三：只有两个模块(不妨设为 A,B)在间隔 I_j 中发生故障，C 和 D 在 I_j 和 I_{j+1} 中无故障，如图 3。D 在检查点间隔 I_j 结束时，比较发现 $CP_{dj}=CP_{cj}$，则把 CP_{cj+1} 复制给 A 和 B 进行恢复，D 继续 I_{j+1}，A,B,C 继续 I_{j+2} 的执行，D 在 I_{j+1} 结束时设置检查点 CP_{dj+1} 并将其与 CP_{cj+1} 比较，有 $CP_{dj+1}=CP_{cj+1}$，所以前面对 A,B 的恢复是正确的。

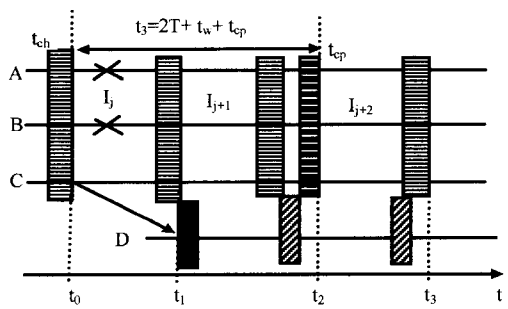

图 3 向前故障恢复

这种情形适用于下表中出现的几种故障情况(X 表示是否故障对结果不影响)

No.	各模块在 I_j 中的状态				各模块在 I_{j+1} 中的状态			
	A	B	C	D	A	B	C	D
1	F	F	T	T	X	X	T	T
2	F	T	F	T	X	T	X	T
3	T	F	F	T	T	X	X	T

情形四：在 I_j 内 A,B,C,D 至少三个模块故障，则卷回到 I_j 开始时刻进行恢复，如图 4。

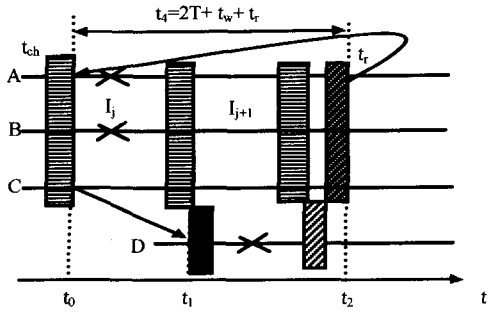

图 4　卷回恢复一

这种情形适用于下表中出现的几种故障情况(X 表示是否故障对结果不影响)

No.	各模块在 I_j 中的状态			
	A	B	C	D
1	F	F	F	X
2	X	F	F	F
3	F	X	F	F
4	F	F	X	F

情形五:只有两个模块(不妨设为 A,B)在间隔 I_j 中发生故障,C 和 D 在 I_j 中无故障,但在 I_{j+1} 中 C,D 至少有一个发生故障,即 D 在 I_j 结束时,比较发现 $CP_{dj}=CP_{cj}$,则把 CP_{cj+1} 复制给 A 和 B 进行恢复,D 继续 I_{j+1},A,B,C 继续 I_{j+2} 的执行;D 在 I_{j+1} 结束时设置检查点 CP_{dj+1} 并将其与 CP_{cj+1} 比较,有 $CP_{dj+1}\neq CP_{cj+1}$,则卷回到 I_{j+1} 开始时刻把 CP_{cj} 复制给 A 和 B 进行恢复,如图 5。

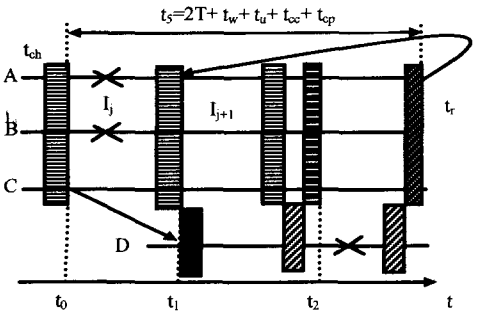

图 5　卷回恢复二

这种情形适用于下表中出现的几种故障情况(X 表示是否故障对结果不影响)

No.	各模块在 I_j 中的状态				各模块在 I_{j+1} 中的状态			
	A	B	C	D	A	B	C	D
1	F	F	T	T	X	X	F	X
2	F	F	T	T	X	X	X	F
3	F	T	F	T	X	X	X	F
4	F	T	F	T	X	F	X	X
5	T	F	F	F	X	X	F	X
6	T	F	F	X	X	X	X	F

4.2　性能分析

设 P_1,P_2,P_3,P_4,P_5 分别表示发生上述五种情形的概率,则有:

$P_1=$ $P(I_j$ 间隔内 A,B,C 均无故障$)=e^{-3\lambda T}$;

$P_2=P(I_j$ 间隔内只有 A 故障$)+$ $P(I_j$ 间隔内只有 B 故障$)+$ $P(I_j$ 间隔内只有 C 故障$)$

$=3(1-e^{-\lambda T})e^{-2\lambda T}$;

$P_3=P(I_j$ 中 A,B 故障,而 C,D 在 I_j,I_{j+1} 中无故障$)+$ $P(I_j$ 中 B,C 故障,而 A,D 在 I_j,I_{j+1} 中无故障$)+$ $P(I_j$ 中 A,C 故障,而 B,D 在 I_j,I_{j+1} 中无故障$)$

$=3(1-e^{-\lambda T})^2e^{-\lambda T}e^{-\lambda T}e^{-\lambda(t_{cr}+t_u+t_{cc})}e^{-\lambda(t_u+t_{cc})}$

$=2(1-e^{-\lambda T})^2e^{-\lambda(2T+t_{cr}+2t_u+2t_{cc})}$;

$P_4=P(I_j$ 中 A,B,C 故障$)+$ $P(I_j$ 中 B,C,D 故障$)+$ $P(I_j$ 中 A,C,D 故障$)+$ $P(I_j$ 中 A,B,D 故障$)$

$=(1-e^{-\lambda T})^3+3(1-e^{-\lambda T})^2e^{-\lambda T}(1-e^{-\lambda(t_{cr}+t_u+t_{cc})})$

$=2(1-e^{-\lambda T})^2e^{-\lambda(2T+t_{cr}+2t_u+2t_{cc})}$;

$P_5=P(I_j$ 中 A,B 故障,而 C,D 在 I_j 中无故障,但在 I_{j+1} 中至少有一个故障$)+P(I_j$ 中 B,C 故障,而 A,D 在 I_j 中无故障,但在 I_{j+1} 中至少有一个故障$)+$ $P(I_j$ 中 A,C 故障,而 B,D 在 I_j 中无故障,但在 I_{j+1} 中至少有一个故障$)$

$=3(1-e^{-\lambda T})^2e^{-\lambda T}e^{-\lambda(t_{cr}+t_u+t_{cc})}(1-e^{-\lambda T}e^{-\lambda(t_u+t_{cc})})$

$=3(1-e^{-\lambda T})e^{-\lambda(T+t_{cr}+t_u+t_{cc})}(1-e^{-\lambda(T+t_u+t_{cc})})$;

显然有:$P_1+P_2+P_3+P_4+P_5=1$;

设 t_1,t_2,t_3,t_4,t_5 分别表示完成上述五种情形所用时间,则

$t_1=T,t_2=T+t_{cp}$, $t_3=2T+t_w+t_r$

$t_4=2T+t_w+t_{cp}$, $t_5=2T+t_w+t_u+t_{cc}+t_r$

因为在进程的最后两个检查点间隔中发生故障时,只能直接使用检查点卷回法进行向后恢复,所以有:

$$\overline{\tau_1}=P_1t_1+(1-P_1)(\overline{\tau_1}+T+t_r) \tag{1}$$

$$\overline{\tau_2}=2\,\overline{\tau_1} \tag{2}$$

当 $n\geqslant 3$ 时,根据数学期望的定义可得下列递归式:

$$\overline{\tau_n}=P_1(\overline{\tau_{n-1}}+t_1)+P_2(\overline{\tau_{n-1}}+t_2)+P_3(\overline{\tau_{n-2}}+t_3)+P_4(\overline{\tau_n}+t_4)+P_5(\overline{\tau_{n-1}}+t_5) \tag{3}$$

解(1)式得:

$$\overline{\tau_1}=t_1+\frac{1-P_1}{P_1}(T+t_r)=T+(\frac{1}{P_1}-1)(T+t_r)=\frac{T+t_r}{e^{-3\lambda T}}-t_r \tag{4}$$

解(3)式得:

$$\overline{\tau_n}=\frac{Q_3}{1+Q_3}((Q_1t_1+Q_2t_2+Q_3t_3+Q_4t_4+Q_5t_5)((n-2)Q_3^{-1}+\frac{1-(-Q_3)^{n-2}}{1+Q_3})+\overline{\tau_1}(Q_3^{-1}+(-Q_3)^{n-1})+(\overline{\tau_2}-(Q_1+Q_4+Q_5)\overline{\tau_1})(Q_3^{-1}+(-Q_3)^{n-2})) \tag{5}$$

其中:

$Q_1=\frac{P_1}{(1-P_4)}$, $Q_2=\frac{P_2}{(1-P_4)}$, $Q_3=\frac{P_3}{(1-P_4)}$,

$Q_4=\frac{P_4}{(1-P_4)}$, $Q_5=\frac{P_5}{(1-P_4)}$

$\overline{\tau_2}$ 由(2)式和(4)式给出。

如果没有模块 D,则在情形三时即在一个检查点间隔内出现两个模块故障时,就需要通过卷回到前一个检查点进行恢复,所以在这种情况下有:

$P_1=$ $P(I_j$ 间隔内 A,B,C 均无故障$)=e^{-3\lambda T}$;

$P_2=P(I_j$ 间隔内只有 A 故障$)+$ $P(I_j$ 间隔内只有 B 故障$)+$ $P(I_j$ 间隔内只有 C 故障$)$

$=3(1-e^{-\lambda T})e^{-2\lambda T}$;

(下转第 132 页)

显优于用傅立叶变换提取的参数，然而用不同小波函数提取的参数识别率也不一样，本文通过试验 db6 效果为最佳。而且用混沌神经网络的识别率明显优于用 BP 神经网络识别率。在以后的研究中，为了进一步提高语音识别的识别率，可以考虑以下几个方面：更合理地选择较多的输入样本；更好地对输入数据进行处理，提取较好的特征参数，尤其是动静态组合的特征参数；改进神经网络的结构等等。神经网络语音识别的研究将会提高到一个全新的水平。

参 考 文 献

[1] Juang B H. The past, present, and future of speech proceesing. IEEE Signal Processing Magzine, May, 1998

[2] Ryeu J K, Chung H S. Chaotic recurrent neural networks and their application to speech recognition[J]. Neurocomputing, 1996, 13(2-4): 281/294

[3] Rabineer L R, juang B H. Fundamentals of speech Processing and Recognition[M]. Prentice-Hall, 1993

[4] 何强，何英. MATLAB 扩展编成[M]. 北京：清华大学出版社，2002，6

[5] Rioul O, Vetterli M. Wavelets and signal processing. IEEE signal processing Mag. pps. October 1991: 14-38

[6] Wassner H, Chollet G. New Cepstral Representation Using Wavelet Analysis an Spectral Transformation for Robust Speech Recognition// Proceedings. Acoustics, Speech, and Signal Processing. IEEE, 2001: 260-263

[7] Torres Humberto M, Rufiner Hugo L. Automatic Speaker Identification by means of Mel Cepstrum, Wavelet and Wavelets Packets// Proceedings of the 22nd Annual EMBS international Conference. IL Chicago: July 2000: 978-981

[8] Aihara K, Takabe T, Toyoda M. Chaotic neural networks[J]. PhysLettA, 1990(144): 333-340

[9] 王旭，王宏，王文辉. 人工神经元网络原理与应用[M]. 沈阳：东北大学出版社，2000：41-46

[10] 任晓林，等. 基于混沌神经网络的语音识别方法[M]. 上海：上海交通大学学报，1999，33(12)：1517-1520

(上接第 121 页)

$P_3 = P(I_j$ 中 A, B 故障，C 无故障$) + P(I_j$ 中 B, C 故障，A 无故障$) + P(I_j$ 中 A, C 故障，B 无故障$) + P(I_j$ 中 A, B, C 均故障$)$

$= 3(1-e^{-\lambda T})^2 e^{-\lambda T} + (1-e^{-\lambda T})^3$；

显然有：$P_1 + P_2 + P_3 = 1$；

同样在最后一个检查点间隔中有：

$$\overline{\tau_1} = P_1 t_1 + (1-P_1)(\overline{\tau_1} + T + t_r)$$

即

$$\overline{\tau_1} = t_1 + \frac{1-P_1}{P_1}(T+t_r) = T + (\frac{1}{P_1} - 1)(T+t_r) = \frac{T+t_r}{e^{-\lambda T}} - t_r \tag{6}$$

而当 $n \geqslant 2$ 时，根据数学期望的定义可得下列递归式：

$$\overline{\tau_n} = P_1(\overline{\tau_{n-1}} + t_1) + P_2(\overline{\tau_{n-1}} + t_2) + P_3(\overline{\tau_n} + t_3) \tag{7}$$

由(7)式解得

$$\overline{\tau_n} = \overline{\tau_{n-1}} + (t_1 + \frac{P_2}{P_1} t_2) = \overline{\tau_{n-2}} + 2(t_1 + \frac{P_2}{P_1} t_2)$$

$= \cdots\cdots$

$$= \overline{\tau_1} + (n-1)(t_1 + \frac{P_2}{P_1} t_2) \tag{8}$$

将(6)式代入(8)式得

$$\overline{\tau_n} = n(t_1 + \frac{P_2}{P_1} t_2) = n(\frac{T+t_r}{e^{-\lambda T}} - t_r) \tag{9}$$

4.3 性能比较

设 $\overline{\tau_{n|f}}$ 表示 TMR 系统中，进程执行期间至少发生一次故障时进程完成时间的数学期望值，则由上节计算得到的 $\overline{\tau_n}$，有

$$\overline{\tau_{n|f}} = \frac{\overline{\tau_n} - P_1^n nT}{1 - P_1^n} \tag{10}$$

将(5)，(9)分别代入(10)就可得到两种恢复策略各自的 $\overline{\tau_{n|f}}$。

从图 6 中我们可看到有模块 D 参加(称为策略 2)要比没有模块 D 参加(称为策略 1)故障定位时对进程的影响小，因为在前一种方案中在模块出现故障时尽量采用向前恢复，避免向后恢复。λ 的大小对 $\overline{\tau_{n|f}}$ 有一定影响，因为 λ 越大模块发生故障的可能性越大。图 6 是在 $\lambda = 1/1000$ 和 $\lambda = 1/4000$ 时 $\overline{\tau_{n|f}}$ 与设置得检查点数的关系图，从图中还可看到并非检查点设置得越多越好。

例 1：各参数设置如下（单位：时间单位）

T_u	t_{cp}	t_r	t_{cc}	t_{ch}	t_{cr}
100	0.4	0.4	0.6	0.6	0.4

结束语 为了节约资源，文中模块 D 在 TMR 工作正常时可作它用，也可以是图 1 中负载较轻的计算机，还可被几个 TMR 共享。从本文的计算可知，在使用 TMR 提高系统可靠性的同时，通过增加少量资源可有效提高 TMR 的效率，实现高可靠、高可用和高性能。

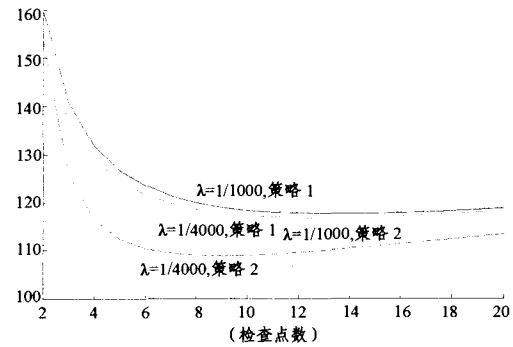

图 6 $\overline{\tau_{n|f}}$ 与检查点个数的关系图

参 考 文 献

[1] Yuan Youguang. The Reliability Techniques in Real-Time Systems (in Chinese). Beijing: Tsinghua University Press, September 1995

[2] 西沃赖克 D P，斯沃兹 R S. 可靠系统的设计理论与实践(上、下). 袁由光、曹泽翰、刘志模，等译. 科学出版社，1988，1993

[3] 袁由光，陈以农. 容错与避错技术及其应用. 科学出版社，1992.

[4] 戴新发. 基于主动任务复制的透明容错计算研究与实现. 哈尔滨工程大学博士学位论文. 2005. 3

[5] WANG Kuochen, WANG Chien-Chun. A Cost-Effective Forward Recovery Checkpointing Scheme in Multiprocessor Systems. Journal of Information Science and Engineering, 2000, 16: 65-80

[6] Long J, Fuchs W K, Abraham J A. Forward Recovery Using Checkpointing in Parallel Systems// Proc. 19th Int'l Conf. Parallel Processing. Aug. 1990: 272-275

[7] Ziv A, Bruck J. Analysis of Checkpointing Schemes for Multiprocessor Systems// Proc. 13th Symp. Reliable Distributed Systems. Oct. 1994: 52-61