系 统 仿 真 学 报
JOURNAL OF SYSTEM SIMULATION
Vol. 16 No. 10
Oct. 2004

用时间序列分析方法动态确定检查点时间间隔

梁 蓓，张大方，杨金民，季 洁
(湖南大学计算机与通信学院，湖南 长沙 410082)

摘 要： 设置检查点是容错计算机系统进行故障恢复的重要手段。检查点的开销则是影响其性能的一个主要因素。许多文献已指出检查点开销主要是来自于对进程状态的保存。文章采用时间序列分析的方法对进程状态大小变化情况进行分析和预测，动态选择恰当的检查点时间间隔，减少检查点时需保存的进程状态量，从而减小了检查点的开销。

关键词： 容错；检查点开销；时间序列分析；进程检查点
文章编号： 1004-731X (2004) 10-2350-04 **中图分类号：** TP 302.08 **文献标识码：** A

Searching the interval of checkpoint dynamically with the method of Time-Series Analysis

LIANG Bei, ZHANG Da-fang, YANG Jin-min, JI Jie
(College of Computer&Communication, Hunan University, Changsha 410082, China)

Abstract: Checkpoint is an important technology to save and rollback the state of the Process. The overhead of checkpoint is a major factor that has effect on the performance. Some papers had mentioned that the overhead of checkpoint mainly dues to save the process's state into stable storage. In this paper, using Time Series Analysis method to modeling and forecast the state size of process, we can select the appropriate interval of checkpoint, which can reduce the overhead.
Keywords: fault tolerance; checkpoint overhead; Time Series Analysis; checkpointing

引 言

设置检查点是容错系统进行故障恢复的重要手段。进程检查点机制是在进程正常运行的适当时刻设置检查点，保存进程状态，当出现故障时，进程卷回到检查点状态，通过重试，进行故障恢复，从而避免从头开始执行，减少计算损失。

设置进程检查点的开销有两个主要来源[1]。一个是系统在每个检查点时都必须完成保存进程状态的操作，这部分开销的大小主要由进程状态的大小决定。另一个主要来源是因故障而引起的卷回，因为每次卷回都会损失已经完成的计算。

许多检查点的研究工作都着眼于如何减小检查点的开销。当检查点的开销不随时间变化或者仅知道检查点的平均开销时，文献[2]提出，使用固定的检查点间隔策略存在一个最优的检查点间隔。但大部分程序在执行过程中，进程状态大小是通过分配和释放内存块动态改变的，因而检查点的开销也是随时间而不断变化的。文献[3]指出检查点的主要开销在于将进程状态写入固定存储器中，所以减少检查点开销关键在于控制检查点文件大小。固定间隔的检查点放置策略没有考虑到检查点时刻进程状态大小，即检查点文件大小，仅用该策略可能使得某些检查点时刻所需要保存文件很大，增大了检查点的开销。由于进程状态大小是动态改变的，所以很可能在固定间隔的检查点时刻附近存在提交或释放内存块的操作改变了进程状态大小，所以如果我们选择在进程状态比较小的时刻放置检查点，就可以减小检查点的开销。

我们通过定时对进程地址空间进行扫描获知进程提交物理存储器的大小，即可得知该时刻进程状态的大小 m_t。利用离散时间序列$\{m_t\}$，用时间序列分析的方法建立进程状态大小模型，利用该模型预测相邻几个时刻进程状态大小，以帮助确定检查点时刻。

本文在第二部分介绍动态确定检查点间隔算法；第三部分介绍利用时间序列分析方法对进程状态大小变化情况建模；最后是实验结果与本文结论。

1 动态确定检查点间隔算法

假设系统故障模型为$e^{\lambda t}$，当检查点的开销不随时间变化或者仅知检查点的平均开销时，文献[2]提出了一种固定间隔大小的最优检查点放置策略。设 $R(t)$是整个程序的检查点开销率，$\bar{c}$ 是检查点的平均开销，t 为设置检查点的时间周期。

$$R(t)=\frac{e^{\lambda t}+\lambda\bar{c}-1}{\lambda t}-1 \tag{1}$$

使得 $R(t)$达到最小的检查点间隔是

$$\tilde{t}\approx\sqrt{2\bar{c}/\lambda} \tag{2}$$

由于程序在执行过程中，进程状态大小是通过分配和释放内存块动态改变的，因而选择不同时刻做检查点所需保存的数据量大小也是不断变化的。固定间隔的检查点放置策略

收稿日期： 2003-09-15 **修回日期：** 2004-03-04
基金项目： 国家自然科学基金资助项目（60273070）
作者简介：梁 蓓(1978-)，女，湖南长沙人，硕士生，研究方向为软件容错；**张大方**(1959-)，男，教授，博导，研究方向为容错计算、网络测试等；**杨金民**(1967-)，男，博士生，研究方向为容错计算、软件容错；**季 洁**(1957-)，女，工程师，研究方向为计算机应用。

没有考虑到检查点时刻进程状态大小，所以仅用此方法可能使得检查点时刻所需要保存的数据很大，有时会加大检查点开销。

检查点的机制需将进程状态保存到固定存储器，固定存储器通常指的是硬盘。因此，进程状态的大小是影响检查点开销的一个重要因素。进程执行过程中其状态大小是随内存块分配和释放动态改变的。通过定时对进程地址空间进行扫描获知进程提交物理存储器的大小，可得知该时刻进程状态的大小 m_t。t 时刻，检查点的开销 c 与进程状态大小 m_t 存在如下关系：

$$c=a+m_t/b \tag{3}$$

其中 a 为进程做检查点固有的时间开销，b 为写固定存储器的速度。因此 c 随 m_t 增大而增大。

文献[3]考虑到检查点时刻进程状态大小对检查点开销的影响，提出一个动态的检查点放置策略。该方法假设进程只有 2 种状态 $s_1,s_2,s_1<s_2$。当进程状态为 s_1 时，设检查点开销为 c_1；当进程状态为 s_2 时，设检查点开销为 c_2，$c_1<c_2$。算法定义了 t_1，t_2，$t_1<t_2$。该算法提出怎样确定放置检查点的时间，T 是间隔上次检查点的时间。算法规则如下：

1) if $T<t_1$，不做检查点。

2) if $t_1\le T<t_2$ 并且进程状态为 s_1，则在 T 时刻放置检查点。

3) if $T=t_2$，在 T 时刻放置检查点。检查点开销为 c_2。

该算法为了避免做过多的检查点而引起高开销，限制了检查点时间间隔的下限 t_1。但如果过长时间不做检查点，就会引起进程发生错误时，浪费的计算过多，重新计算时间过长，因此限定了检查点时间间隔的上限 t_2。如果在 t_1 与 t_2 之间发现了小的进程状态，就放置检查点。如果进程一直运行至 t_2 仍然没有变化为小的进程状态，那么就在上限 t_2 放置检查点尽管此时有较大的进程状态。该文章假设进程两种状态的迁移过程服从马尔克夫链的迁移过程，分析得出 t_1 和 t_2 的理论值。

结合上述算法思路，我们提出了一种动态确定检查点时间间隔的算法。算法的基本思想是结合该进程历史检查点的开销记录估计下一个检查点时间间隔

$$\tilde{t}=\min(\sqrt{2\bar{c}/\lambda},MaxWait) \tag{4}$$

其中 $\bar{c}$ 是该进程运行过程中历史检查点的平均开销，$MaxWait$ 是最大的检查点间隔。在其附近的区间$[t_1,t_2]$内选取使得该间隔开销率达到最小的点放置检查点。算法的关键问题在于如何确定 t_1、t_2 的取值以及如何预测$[t_1,t_2]$间开销率最小的点。

根据(1)式，当 λ 不变，平均开销 $\bar{c}$ 保持大致稳定时，检查点开销率 $R(t)$与检查点间隔 t 存在如下关系，如图 1。

可见当间隔 t 过小时，检查点的开销率会因为多次做检查点而增大；当间隔过大时，检查点的开销率会因为回卷恢复的时间增加而增大。通过让检查点的开销率 $R(t)$小于等于某确定值 R_0 来确定 t_1、t_2 的取值。

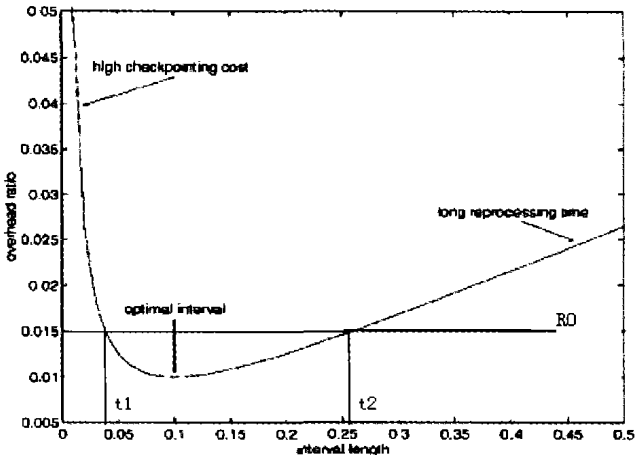

图 1 $R(t)$ 与 t 的关系

确定 t_1、t_2 后，如何选取$[t_1,t_2]$间开销率最小的时刻放置检查点是需要解决的第二个问题。除非预先知道进程在 t_1 与 t_2 之间状态大小变化的情况，否则无法选择$[t_1,t_2]$间开销率最小的时刻。预知进程状态大小的变化是比较困难的。进程状态大小实际上就是进程提交物理存储器的大小，而进程提交和释放物理存储器具有以下特性：

- 物理存储器的提交和释放具有对称的特性
- 物理存储器使用具有循环的特性

因此，我们可以利用时间序列分析法给进程提交物理存储器的情况建模预测进程状态大小[4]，具体方法在下一节进行论述。

我们利用时间序列对进程提交物理存储器大小建立数学模型预测其变化情况，提出了一种新的检查点放置算法。首先，利用该进程历史检查点的开销记录按照（4）式估计下一个检查点时间间隔 $\tilde{t}$。设定 R_0，求出对应 t_1、t_2 的取值。T 仍为间隔上次检查点的时间，确定检查点时刻规则如下：

1) if $0\le T<t_1$，通过固定时间间隔 Δt 对进程提交物理存储器大小进行扫描，记录进程状态大小的时间序列 $\{m_t\}$。

2) if $T=t_1$，若$\{m_t\}$为常数，设 t_0 为该段预测的最佳检查点时间间隔，则 $t_0=\tilde{t}$。

若$\{m_t\}$不为常数，则利用时间序列$\{m_t\}$采用第 3 小节中介绍的方法建立进程状态大小模型，并预测出 $t_1<T\le t_2$ 的 $k=(t_2-t_1)/\Delta t$ 个时间点的进程状态大小 m_i，$i=1..k$。然后，根据式（3）和式（1）分别计算每个时间点的 c_i 和 $r(t_i)$，其中使 $r(t)$达到最小的 i 记为 i_0，$t_0=T+i_0\Delta t$。

3) if $T=t_0$, 则放置检查点。

目标程序执行过程中，通过采样如果其进程状态大小不变，该算法选取间隔大小为 $\tilde{t}$ 放置检查点，并且不建模，对程序也不会有附加的开销。当进程状态大小变化时，算法为其建立进程状态大小模型进行预测，选取$[t_1,t_2]$间检查点开销率最小的时刻放置检查点。

2 时间序列分析方法为进程状态大小建模

时间序列预测方法是系统正常运行时，对其进行有规律的数据采集，并加以必要的数据处理和数学计算，估计出系统对象的数学模型，根据模型预测系统数据。时间序列预测方法中关键是进行数学建模。数学建模包括模型类和模型结构参数的确定两部分。根据进程状态大小的特性，可选用随机序列线性模型进行建模。

随机序列线性模型分为三类，自回归模型（AR），滑动平均模型(MA)和自回归滑动平均模型(ARMA)。模型类的确定需根据采集的时间序列数据的特性来决定。

建模过程如下：

1) 计算样本序列的标准自相关函数和偏自相关函数

随机序列 m_t 的 n 次观察值，则样本均值为

$$\overline{m} = \frac{1}{n}\sum_{t=1}^{n} m_t \tag{5}$$

随机序列 m_t 的样本自协方差函数为

$$\hat{r}_k = \frac{1}{n}\sum_{t=1}^{n-k}(m_{t+k} - \overline{m})(y_t - \bar{y}) \tag{6}$$

则样本自相关函数为

$$\hat{\rho}_k = \hat{r}_k \Big/ \hat{r}_0 \tag{7}$$

样本的偏相关函数由下述递推公式[8]进行计算：

$$\begin{aligned} &\varphi_{11} = \hat{\rho}_1 \\ &\varphi_{k-1k-1} = (\hat{\rho}_{k-1} - \sum_{j=1}^{k}\hat{\rho}_{k-1-j}\hat{\varphi}_{kj}) \times (1 - \sum_{j=1}^{k}\hat{\rho}_j\hat{\varphi}_{kj})^{-1} \\ &\hat{\varphi}_{k+1j} = \hat{\varphi}_{kj} - \hat{\varphi}_{k+1k+1}\hat{\varphi}_{kk-j+1} \end{aligned} \tag{8}$$

式中 $\hat{\rho}_t$ 是样本标准自相关函数。

2) 模型识别

AR(p)、MA(q)与 ARMA(p,q)模型具有各自的不同特征，如何确定选择哪种模型进行预测就是模型识别的过程。根据各类模型的自相关与偏自相关函数的具体特征进行模型识别。

(1) AR(p)模型的识别方法

若 m_t 的偏自相关函数 φ_{kk} 在 p 步以后截尾，即 $k>p$ 时 $\varphi_{kk}=0$,而且它的自相关函数 ρ_k "拖尾"，则可断言此序列为自回归序列 AR(p)。但是，当 $k>p$ 时 $\varphi_{kk}=0$ 仅是理论上的，实际上的样本偏自相关函数 $\hat{\varphi}_{kk}$ 仅是理论偏自相关函数的一个估计值，由于样本随机性，免不了有误差。因此，当 $k>p$ 时 $\hat{\varphi}_{kk}$ 不会全为零，而是在零的上下波动。

(2) MA(q)模型的识别方法

若随机序列的自相关函数截尾，即自 q 步以后有 $\rho_k=0$。$k>q$ 时，而它的偏自相关函数拖尾，则可断言此序列是滑动平均 MA(q)序列。但是，在 $k>q$ 以后，$\rho_k=0$ 仅是理论上的，实际的样本自相关函数不会在 q 步以后全为零，而是在零的上下波动。

(3) ARMA(p,q)模型的识别方法

若随机序列的自相关函数与偏自相关函数均是拖尾，则可判断此序列是自回归滑动平均序列。

3) 参数的确定

由模型识别，可以选择模型的类型。确定模型阶后，需要确定模型参数，才能求得模型。由于篇幅有限，这里仅介绍 AR（p）模型的参数确定方法。其余模型的参数确定方法参见[5]。

根据[5]中的方法，实现了 AR 模型自动辨别机。该辨别机根据时间序列的观测数据自动的决定 AR 模型的阶 p 和参数 α 。

AR（p）满足如下方程：

$$x_t = a_1 x_{t-1} + a_2 x_{t-2} + \cdots + a_p x_{t-p} + e_t \tag{9}$$

$a_1, a_2, ..., a_p$ 是需要估计的未知参数，p 为模型的阶。

a) 确定模型的阶 p

AR 模型自动辨别机采用了 F 检验法定阶。由低阶到高阶用递推最小二乘法（RLS）分别建立 AR（1），AR（2），...，AR（p+1）共 p+1 个模型，并在相邻两个模型间进行 F 检验，即可决定模型的阶 p。

AR 模型阶的 F 检验公式为：

$$F = \frac{(A_1 - A_0)(N-p)}{A_0} \tag{10}$$

其中 A_0 =AR（p）的残差平方和 RSS；A_1 =AR（p-1）的残差平方和 RSS；N 为观测数据数目。

b) 参数估计

AR 模型自动辨别机的参数 α 估计采用递推最小二乘法（RLS），公式为：

$$\hat{\alpha}(N+1) = \hat{\alpha}(N) + k(N+1)(x_{N+1} - \varphi^T(N+1)\hat{\alpha}(N)) \tag{11}$$

$$k(N+1) = \frac{p(N)\varphi(N+1)}{1 + \varphi^T(N+1)p(N)\varphi(N+1)} \tag{12}$$

$$p(N+1) = [I - k(N+1)\varphi^T(N+1)]p(N) \tag{13}$$

其中 $\hat{\alpha}(0)=0$,p（N）的初值为 $p(0)=\mu I$, μ 为很大的正数，如 $\mu=10^4$。

4) 模型的检验

通过模型识别与参数估计，建立起相应的随机线性数学模型。为判断所建模型是否合理，以及为进一步改善已建立的模型，有必要对已建立的模型作检验或诊断检验。模型的诊断检验包括模型的平稳性分析，残差分析检验和过拟合检验三个方面。对建立的模型进行检验后，如果模型合理，则可进行进程状态大小预测。

3 实验结果与分析

在操作系统为 WinNT，配置为内存 128M，CPU 为 PIII800MHz 的 PC 机上,利用检查点系统 NTckpt[6]为进程加入检查点功能。为了体现动态确定间隔算法的特点，选取了内存变化明显的遗传算法的程序进行测试。分别采用动态间隔和固定间隔两种策略进行检查点的设置。

测试程序：瞬态电流测试中利用遗传算法进行测试生成。选取 $\lambda = 0.001$。采用动态时间间隔检查点放置方法，初始检查点间隔选择 60s，R_0=15%，*MaxWait*=80s，实验结果如表1。采用固定间隔检查点放置方法，检查点间隔选择60s，实验结果如表2。两种方法的检查点时间和开销比较如图2。

表1 采用动态时间间隔检查点放置方法

检查点序号	检查点时间（s）	检查点开销（s）	预测开销（s）	单个间隔的检查点开销率（%）
1	60	3.09	0	5.16
2	134	3.83	0.2	5.44
3	212	4.12	0.31	5.67
4	289	3.81	0.21	5.21
5	363	3.12	0.26	4.57
平均检查点开销率（%）			5.21	

表2 采用固定间隔检查点放置方法

检查点序号	检查点时间（s）	检查点开销（s）	单个间隔的检查点开销率（%）
1	60	3.09	5.16
2	120	3.78	6.30
3	180	4.01	6.68
4	240	3.68	6.43
5	300	3.76	6.26
6	360	3.12	5.2
平均检查点开销率（%）		6.01	

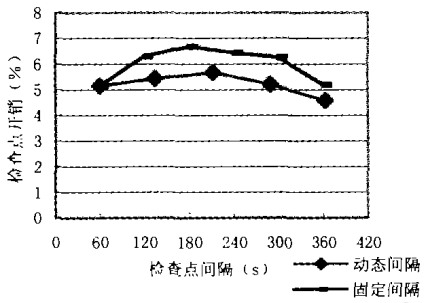

图2 两种方法检查点间隔及开销比较

实验数据表明，本文的动态间隔设置方法有如下优点：

1. 当用户对应用程序的特性不了解时，往往不能为其设置合适的检查点间隔。而检查点间隔是影响检查点开销的一个重要因数。动态间隔方法可以根据该进程历史的检查点开销记录动态选择合适的检查点间隔，降低检查点的平均开销。

2. 当程序在执行过程中进程状态大小发生时，动态间隔设置方法结合历史检查点开销记录大致估计下一个检查点时间区间，利用时间序列分析方法为进程状态建模并预测检查点开销率最小的时刻放置检查点，降低了检查点的平均开销率。

4 结论

本文基于固定间隔的检查点放置方法提出了一种动态确定检查点间隔的算法。该算法将进程检查点的放置限定在一个区间内，使得检查点开销率小于等于某固定值来确定区间上下限。放置检查点前，算法以固定频率对进程状态大小进行采样，用时间序列分析方法对进程状态大小建立数学模型，预测区间内几个时刻进程状态大小情况，并计算各个时刻检查点开销率，选取开销率最小的时刻放置检查点。试验结果显示，当程序运行过程中，进程状态变化明显时，该方法能够有效的减小检查点的开销。

今后的工作还需进一步研究采样频率的选取，以及它们对检查点开销的影响。

参考文献：

[1] Ziv A, Bruck J. Performance Optimization of Checkpointing Schemes with Task Duplication [J]. IEEE Trans Computers, 1997, 46(12): 1381-1386

[2] Duda A. The Effects of Checkpointing on Program Execution Time [J]. Information Processing Letters, 1983, 16: 221-229

[3] Ziv A, Bruck J, An On-Line Algorithm for Checkpoint Placement [J]. IEEE Transactions on Computers, 1997, 46(9): 976-985

[4] Jiman Hong, Sangsu Kim, On the choice of checkpoint interval using memory usage profile and adaptive time series analysis [J]. IEEE 2001 Pacific Rim International Symposium on Dependable Computing, 2001, 45-48

[5] 邓自立, 郭一新, 等. 现代时间序列分析及其应用[M]. 第一版. 北京: 知识出版社, 1989年6月.

[6] 梁蓓, 杨金民, 张大方. WinNT 进程检查点系统 NTckpt 的设计与实现 [J]. 计算机应用, 2003, 23(6): 23-25