ISSN 1000-9825, CODEN RUXUEW
*Journal of Software*, Vol.21, No.2, February 2010, pp.218−230
doi: 10.3724/SP.J.1001.2010.03787

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

# 可扩展的多周期检查点设置*

慈轶为+, 张 展, 左德承, 吴智博, 杨孝宗

(哈尔滨工业大学 计算机科学与技术学院,黑龙江 哈尔滨 150001)

## Scalable Time-Based Multi-Cycle Checkpointing

CI Yi-Wei+, ZHANG Zhan, ZUO De-Cheng, WU Zhi-Bo, YANG Xiao-Zong

(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

+ Corresponding author: E-mail: ciyiwei@hotmail.com

**Ci YW, Zhang Z, Zuo DC, Wu ZB, Yang XZ. Scalable time-based multi-cycle checkpointing. *Journal of Software*, 2010,21(2):218−230.** http://www.jos.org.cn/1000-9825/3787.htm

**Abstract**: In this paper, a time-based multi-cycle checkpointing approach, allowing each process to take checkpoints with its own checkpoint cycle, is proposed. To ensure the advancement of consistent global checkpoint, checkpoint cycles can be adjusted according to a "$\mathcal{P}$-pattern". In the proposed approach, processes will be divided into zones, so that dependency tracking required for checkpoint cycle adjustment can be restricted in the zone scope. It makes the time-based multi-cycle checkpointing more scalable.

**Key words**: fault-tolerance; checkpoint; dependency tracking

**摘 要**: 提出了一种多周期检查点设置方法.它允许各个进程采用不同周期进行检查点设置.为了保证一致全局检查点的向前推进,检查点周期可以根据一个$\mathcal{P}$模式进行调整.在所提出的方法中,进程可以进行组划分处理,从而用于检查点周期调整的依赖跟踪可被限定在组内,同时也将使基于时间的多周期检查点设置具有较好的可扩展性.

**关键词**: 容错;检查点;依赖跟踪

**中图法分类号**: TP311 **文献标识码**: A

在大规模分布式环境中,往往难以确保所有节点总能正常运转.这对于需要长时间运行的分布式计算程序来说是不希望看到的.试想,单个进程的故障就会引发整个程序被迫重新执行,那么很可能带来可观的计算损失.为了尽量避免这样的现象发生,在分布式容错系统中,通过周期性的状态保存,失效的进程可以恢复到某些正确的状态而无须完全重新执行.也就是说,系统将可获得计算损失控制的能力.考虑到进程对计算损失的敏感程度不尽相同,一些进程需要严格控制计算损失;而同时,另一些进程可能并无这样的要求.这就需要一种能够针对不同进程采用不同频率进行状态保存的策略.这里所说的进程状态信息称为检查点(checkpoint).在通常的检查点设置算法中,以较高自主性设置检查点的代价往往是卷回距离难以控制.如何使二者兼顾就成为一个至关重要的问题.通常的检查点设置算法有 3 类:非协同[1−3]、协同[4−12]以及通信引发[13−20].其中,非协同检查点设

* Supported by the National High-Tech Research and Development Plan of China under Grant Nos.2008AA01A204, 2009AA01A404 (国家高技术研究发展计划(863)); the State Key Laboratory of High-End Server & Storage Technology of China under Grant No.2009HSSA07 (高效能服务器和存储技术国家重点实验室开放基金项目)

Received 2009-04-16; Revised 2009-09-11; Accepted 2009-12-07

置算法具有最好的自主性,但是这类算法会受到多米诺效应(domino-effect)[21]的影响,从而导致难以找到可用检查点进而带来较大的计算损失.对于协同检查点设置算法,计算损失可以得到很好的控制,这主要得益于进程的检查点设置几乎是同步完成的,一旦某个进程失效,就可以选择其最近的检查点进行恢复.但是我们可以看到,这类算法中,进程检查点设置的自主性是较弱的.基于时间的检查点设置算法[8−12]虽然通过局部时间的使用避免了通常的协同检查点设置算法在保证进程检查点设置同步过程中发送的控制消息,但是这类算法往往需要各个进程采用相同的检查点设置周期,因此检查点设置的自主性仍然较弱.通信引发检查点设置算法(communication induced checkpointing,简称CIC)的提出,在一定程度上解决了检查点设置的自主性以及卷回距离控制两个方面的问题.CIC算法可借助一种隐式同步来避免无用检查点的产生,但这一过程将会引发强迫检查点的设置.文献[18]指出,这有可能破坏CIC算法检查点设置的自主性.同时,CIC算法在恢复过程中仍有可能使用一些并不是最新的检查点进行恢复.各个进程的卷回距离与通信模式息息相关,因此往往难以做到恢复距离的准确控制.

出于提高检查点设置自主性以及有效控制卷回距离两个方面的考虑,我们提出了一种基于时间的多周期检查点设置算法.该算法允许各个进程使用不同周期进行检查点设置.为了确保一致全局检查点(consistent global checkpoint,简称CGC)总是向前推进的,从而控制计算损失,我们提出了一种使得一个$\mathcal{Z}$模式[22]能够满足的混合策略.其特点在于,可以令系统中的进程分为不同组,各组进程可以采用直接策略或者间接策略进行检查点周期的调整.这种分组机制可有效控制间接策略中依赖跟踪的深度,以使检查点协议具有较好的可扩展性,从而适应大规模应用.实验结果表明,本文提出的算法可以在相对较低的检查点设置开销下获得较小的卷回偏移.

本文第 1 节给出系统模型.第 2 节给出可扩展的多周期检查点设置算法(scalable time based multi-cycle checkpointing,简称 STBMC),并证明该算法具有不存在无用检查点的特性.第 3 节进行算法的性能分析.最后给出结论.

## 1 系统模型

我们假设进程间的相互通信仅通过消息传递,消息传递的延迟任意但却是有限的.每个进程将会执行消息发送、消息提交以及内部操作.消息提交是指消息接收后真正提交给应用.这里用$send(m)$和$deliver(m)$分别表示消息$m$的发送以及提交事件.此外,假设每个进程具有一个期望的检查点设置周期.进程$P_i$的期望检查点设置周期用$T_i$表示.令$\tau_0$表示系统中的最短检查点周期,$\tau_d$表示系统中的基准检查点周期,要求各个进程的期望检查点周期满足:

1) $T_i>\tau_d \Rightarrow (T_i \bmod \tau_d)=0$;

2) $T_i\le\tau_d \Rightarrow (\tau_d \bmod T_i)=0$.

其中,$\tau_d$满足$(\tau_d \bmod \tau_0)=0$.

### 1.1 一致全局检查点

这里用$C_{i,n}$表示进程$P_i$索引为$n$的检查点.需要指出,同一进程检查点的索引是递增的但并不一定连续.$I_{i,x}$表示进程$P_i$的第$x$个检查点间隔.$CI(I_{i,x})$表示检查点间隔$I_{i,x}$结束时所设置检查点的索引.$I_{i,x}$直接依赖于$I_{j,y}$当且仅当存在一个消息$m$在进程$P_j$的检查点间隔$I_{j,y}$发送并且在进程$P_i$的检查点间隔$I_{i,x}$提交.我们用符号$\prec$表示两个检查点间隔的直接依赖关系,$I_{i,x}$直接依赖于$I_{j,y}$,表示为$I_{j,y}\prec I_{i,x}$.用$\prec^c$表示$\prec$的传递闭包,$I_{j,y}\prec^c I_{i,x}$表示$I_{i,x}$依赖于$I_{j,y}$.对于两个检查点的先后关系,可根据检查点间隔的直接依赖关系来确定.这里,我们用$\rightarrow$表示先于关系,$C_{i,k}$先于$C_{j,l}$即$C_{i,k}\rightarrow C_{j,l}$.先于关系的定义如下:

**定义 1**. 设$CI(I_{i,x})=k$,$CI(I_{j,y})=l$.$C_{i,k}\rightarrow C_{j,l}$当且仅当下述条件之一成立:

1) $(i=j)\wedge(k<l)$.

2) $\exists I_{i,u},I_{j,v}:(I_{i,u}\prec I_{j,v})\wedge(u>x)\wedge(v\le y)$.

如果消息$m$的提交导致$C_{i,k}\rightarrow C_{j,l}$,那么我们说$m$使得$C_{i,k}$和$C_{j,l}$有序.在给出一致全局检查点的定义前,首先对

全局检查点进行说明.全局检查点即由每个进程提供一个检查点而构成的检查点集合.在该检查点集合中,如果任意两个检查点都是无序的,那么这个全局检查点可称为一致全局检查点.可以设想,如果消息$m$使得某个全局检查点中的两个检查点$C_{i,k}$和$C_{j,l}$有序,那么在使用$C_{i,k}$和$C_{j,l}$恢复时将会出现$m$的接收事件存在,而发送事件不存在的现象.这种情况是应当避免的.一致全局检查点的定义如下:

**定义 2**. 全局检查点$C$是一致的当且仅当$\forall C_{i,k},C_{j,l}\in C(i\neq j),(C_{i,k}\nrightarrow C_{j,l})\wedge(C_{j,l}\nrightarrow C_{i,k})$.

下面我们将给出无用检查点[19]的定义.在此之前,需要首先定义Z-path.Z-path描述了检查点之间的弱先后关系.文献[23]指出,一致全局检查点中的任意两个检查点间不存在Z-path.

**定义 3**. Z-path由一组消息$[m_1,m_2,m_3,\ldots,m_q](q\geq 1)$组成,$\forall l,1\leq l<q,deliver(m_l)$与$send(m_{l+1})$被同一进程执行,并且满足下述条件之一:

1) $deliver(m_l)$在$send(m_{l+1})$之前的检查点间隔执行.

2) $deliver(m_l)$与$send(m_{l+1})$在同一检查点间隔执行.

如果检查点$C_{i,k}$与$C_{j,l}$间存在一条Z-path$[m_1,m_2,m_3,\ldots,m_q]$,并且$m_1$发送于$C_{i,k}$设置之后,$m_q$提交于$C_{j,l}$设置之前,我们说有一条由$C_{i,k}$到$C_{j,l}$的Z-path,表示为$C_{i,k}\rightsquigarrow_z C_{j,l}$.

**定义 4**. 如果$(send(m_1)\in I_{i,x})\wedge(deliver(m_q)\in I_{i,x'})\wedge(x'<x)$,那么Z-path$[m_1,m_2,m_3,\ldots,m_q]$称为Z-cycle.

**定义 5**. 如果一个检查点卷入到一个Z-cycle中,那么这个检查点是无用的,即$C_{i,k}$无用当且仅当$C_{i,k}\rightsquigarrow_z C_{i,k}$.

### 1.2 时间偏移

我们假设各个进程的局部时间是松散同步的.为了控制进程局部时间的漂移,局部时间需要定期地进行再同步处理.假设系统中存在一个协调者进程,每隔一段时间会重新调整各个进程的局部时间,从而完成局部时间的同步.这里,用$R_i(t)$表示当进程$P_i$的局部时间达到$t$时对应的实际时间.对于进程的局部时间以及检查点设置时间我们有如下假设(其中,$\tau_{cpt}$表示设置一个检查点的耗时,$\tau_{syn}$表示局部时间同步的周期):

**TA1**: $|R_i(t+t')-R_j(t)-t'|\leq\varepsilon(\varepsilon<<\tau_0,0\leq t'<\tau_{syn})$.

**TA2**: $t<t'\Leftrightarrow R_i(t)<R_i(t')$.

**TA3**: $\tau_{cpt}<<\tau_0$.

## 2 多周期检查点设置

### 2.1 基本思想

多周期检查点算法允许各个进程按照不同的周期进行检查点设置,这虽然可以提高检查点设置的自主性,但与此同时会带来另一方面的问题,即一些以较短周期设置检查点的进程可能不能从频繁的检查点设置中得到较小的计算损失,我们称其为跳跃效应.跳跃效应主要表示进程在恢复过程中不能选择近期设置的检查点进行恢复,而只能选择相对久远的检查点.如图 1 所示,当进程$P_i$失效后,所有进程被迫恢复到初始检查点,$P_j$与$P_k$的最后4个检查点都未能用于构造一致全局检查点.在图2中,通过对$P_i$和$P_l$检查点周期进行临时调整,全局一致检查点可从$\{C_{i,0},C_{j,0},C_{k,0},C_{l,0}\}$推进到$\{C_{i,1},C_{j,4},C_{k,4},C_{l,1}\}$.

图 2 中所描述的检查点模式具有以下特点:如果两个检查点$C_{i,k}$和$C_{j,l}$有序,那么可以找到一个检查点$C_{i,k'}$(或者$C_{j,l'}$)使得$C_{i,k'}$与$C_{j,l}$无序(或者$C_{i,k}$与$C_{j,l'}$无序).这样,$C_{i,k'}$与$C_{j,l}$就可能被包含在同一个一致全局检查点当中.例如,图 2 中$C_{i,0}$与$C_{j,4}$是有序的,但$C_{i,1}$与$C_{j,4}$则是无序的.这里,我们将上述检查点模式称为$\mathcal{P}$模式,对于$\mathcal{P}$模式的具体定义如下:

**定义 6**. $\mathcal{P}\cong\forall I_{i,x},I_{j,y}:I_{j,y}\prec^c I_{i,x}\Rightarrow\Diamond(T_{j,y}\leq T_{i,x})$,其中,$\Diamond$表示“最终”.

$\mathcal{P}$模式下任意两个检查点$C_{i,k}$与$C_{j,l}$(假设$CI(I_{i,x})=k,CI(I_{j,y})=l$),如果$I_{i,x}$依赖于$I_{j,y}$,那么检查点$C_{j,l}$的设置最终不会迟于$C_{i,k}$.$\mathcal{P}$模式具有以下特性:

**引理 1**. 如果$\mathcal{P}$模式满足,那么有$C_{j,l}\rightsquigarrow_z C_{i,k}\Rightarrow\Diamond(T_{j,y+1}\leq T_{i,x})$,其中,$k=CI(I_{i,x}),l=CI(I_{j,y})$.

**定理 1**. 如果$\mathcal{P}$模式满足,那么不存在无用检查点.

证明:采用反证法.假设存在无用检查点,那么$\exists C_{i,k}$:$C_{i,k} \rightsquigarrow_z C_{i,k}$.设$CI(I_{i,x})=k$,根据引理 1,我们有$\Diamond(T_{i,x+1} \le T_{i,x})$.这与$T_{i,x}<T_{i,x+1}$相矛盾,故假设不真. □

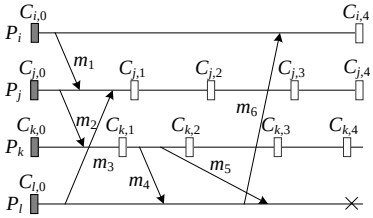



Fig.1 Skipping effect

图 1 跳跃效应

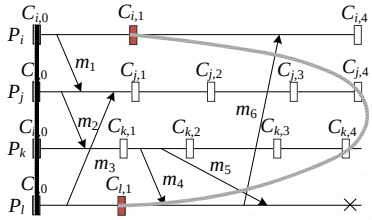



Fig.2 Advancement of CGC

图 2 一致全局检查点的推进

### 2.2 检查点的设置

当进程的局部时间达到检查点设置时刻时,如果当前的检查点间隔(假设为$I_{i,x}$)依赖于另一个检查点间隔(假设为$I_{j,y}$)并且$T_{j,y}>T_{i,x}$,由于此时$\mathcal{P}$模式未被满足,检查点周期需要作出调整.若$\mathcal{P}$模式满足,一个新的检查点可被设置.对检查点周期调整的具体描述我们将在后面给出.

**规则 1**. $CI(I_{i,x})=T_{i,x}/\tau_0$.

这里要求任何检查点周期均为$\tau_0$的整数倍,可知$CI(I_{i,x})$为整数.需要指出的是,同一进程检查点的索引是递增的,但并不一定连续.此外,为了保证尽可能采用期望检查点周期进行检查点设置,当完成一次检查点设置后,新的检查点设置时刻可采用下述方法计算:

**规则 2**. $T_{i,x+1}=\lfloor (T_{i,x}+T_i)/T_i \rfloor T_i$.

图 3 给出了进程$P_i$和$P_j$的执行过程,其中,$P_i$和$P_j$的当前检查点间隔分别为$I_{i,3}$和$I_{j,4}$,$T_{j,4}=T_{i,2}=4\tau_0$.若存在一个消息$m$在检查点间隔$I_{i,3}$内发送,如果$m$在$P_j$的检查点间隔$I_{j,4}$提交,那么由于$T_{j,4}<T_{i,3}$,为了仍然能够使得$\mathcal{P}$模式满足,$T_{j,4}$需要调整到$T_{i,3}$.然而,假如在$m$提交前不存在$I_{k,z}$使得$(I_{k,z} \prec^c I_{j,4}) \wedge (T_{k,z}>T_{j,4})$,那么如果$I_{j,4}$在$m$提交前结束,检查点时刻$T_{j,4}$的调整就可以避免.这里,为了使检查点设置的进度可以被其他进程所感知,在每个消息中携带一个时间索引.$m.TI$表示携带在消息$m$中的时间索引.令$t_i$表示进程$P_i$的局部时间,$m.TI$的计算方法如下:

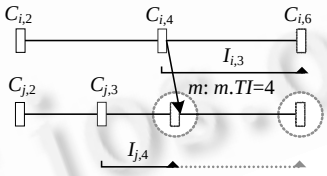



Fig.3 Time index

图 3 时间索引

**规则 3**. $m.TI=\lfloor t_i/\tau_0 \rfloor$.

其中,$t_i$具体指发送消息$m$时$P_i$的局部时间.当一个进程$P_j$接收到消息$m$时发现局部时间$t_j$要先于$m.TI\cdot\tau_0$,那么$t_j$将被更新为$m.TI\cdot\tau_0$.这里,$m.TI\cdot\tau_0$表示根据时间索引还原出的发送$m$进程所处的一个时间间隔起始.如果这时$t_j$达到一个检查点设置时刻,那么$P_j$将会启动检查点设置,而后才可提交消息$m$.当然,一个新的检查点能否最终设置,还取决于此时$\mathcal{P}$模式是否满足.在图 3 中,如果不存在$I_{k,z}$使得$(I_{k,z} \prec^c I_{j,4}) \wedge (T_{k,z}>T_{j,4})$,那么$I_{j,4}$将在提交$m$前结束.检查点设置的形式化描述如图 4 所示.检查点索引具有如下性质:

**性质 1**. $\forall m$:$deliver(m) \in I_{i,x} \Rightarrow CI(I_{i,x-1}) \le m.TI < CI(I_{i,x})$.

**When $P_i$'s local time $t_i$ reaches $T_{i,x}$**
/* Check if $\mathcal{P}$-pattern is followed */
1. **if** $\neg(\exists I_{j,y}):(I_{j,y}\prec^c I_{i,x})\wedge(T_{j,y}>T_{i,x})$ **then**
2. Take a checkpoint;
3. **else**
4. Update checkpoint time;
5. **endif**

**When $P_j$ receives a message $m$**
/* The current checkpoint interval is $I_{i,x}$ */
1. **if** $m.TI\cdot\tau_0\geq T_{i,x}$ **then**
2. $t_i\leftarrow T_{i,x}$;
/* Check if $\mathcal{P}$-pattern is followed */
3. **if** $\neg(\exists I_{j,y}):(I_{j,y}\prec^c I_{i,x})\wedge(T_{j,y}>T_{i,x})$ **then**
4. Take a checkpoint;
5. **else**
6. Update checkpoint time;
7. **endif**
8. **elseif** $m.TI\cdot\tau_0>t_i$ **then**
9. $t_i\leftarrow m.TI\cdot\tau_0$; /* Update local time*/
10. **endif**
11. Deliver $m$;

Fig.4 Checkpointing
图 4 检查点的设置

### 2.3 检查点周期的调整

为了使$\mathcal{P}$模式满足,要求任意检查点间隔$I_{i,x}$与$I_{j,y}$,如果$I_{j,y}\prec^c I_{i,x}$,那么必须保证最终$T_{j,y}\leq T_{i,x}$.这里,我们给出 3 种检查点周期调整策略,分别为直接策略、间接策略以及混合策略.在直接策略下,检查点周期会被主动调整从而防止违背$\mathcal{P}$模式的可能.在这一过程中不需要收集任何依赖信息,但是进行这样的检查点周期调整将会导致检查点数量的增加.与其相对的是采用间接的检查点周期调整策略,当一个进程通过依赖信息收集发现$\mathcal{P}$模式未被满足时,该进程的检查点周期需要作调整.在这种模式下,依赖信息收集的开销与系统中进程的数目相关.一方面为使检查点算法能够适用于大规模环境,另一方面为了有效地控制检查点设置开销,我们采用一种混合策略,将直接策略与间接策略进行了有机的结合.通过分组机制,当需要进行依赖信息收集时,可使参与进程控制在同组范围内.这样,可以通过控制组规模来控制依赖信息收集开销.后面我们将证明在混合策略下,$\mathcal{P}$模式仍可得到满足.在介绍混合策略之前,我们首先给出直接策略以及间接策略.

#### 2.3.1 直接策略

当存在一个消息$m$由检查点间隔$I_{j,y}$发送至$I_{i,x}$时(即$I_{j,y}\prec I_{i,x}$),根据性质 1,若接收进程的期望检查点周期是$\tau$的整数倍,那么$T_{i,x}\geq\lfloor(m.TI\cdot\tau_0+\tau)/\tau\rfloor\tau$.如果$T_{j,y}$迟于$\lfloor(m.TI\cdot\tau_0+\tau)/\tau\rfloor\tau$,则$\mathcal{P}$模式可能无法满足.此时,$I_{j,y}\prec I_{i,x}$称为一个可疑依赖.为使这些可疑依赖不破坏$\mathcal{P}$模式,我们可以令$T_{j,y}$直接调整至$\lfloor(t_j+\tau)/\tau\rfloor\tau$,即采用了一种发送端检查点周期直调的方式.该策略如下:

**规则 4**. 当一个进程$P_j$发送消息到$P_i$时(假设$t_j$指示$P_j$此时的局部时间),如果$P_i$的期望检查点周期是$\tau$的整数倍,那么$P_j$的当前检查点设置时刻$T_{j,y}$将调整至$\lfloor(t_j+\tau)/\tau\rfloor\tau$.

如图 5 所示($T_i=4\tau,T_j=2\tau,T_k=\tau,T_l=4\tau$),当进程$P_i$发送消息$m_1$时($t_i<\tau$),其将检查点周期调整为$\tau$.当设置完检查点$C_{i,1}$后,根据规则 2,$P_i$将检查点周期调整为 3$\tau$.类似地有,当$P_j$发送消息$m_5$时($2\tau<t_j<3\tau$),$P_j$将检查点周期调整为$\tau$.直接策略具有以下性质:

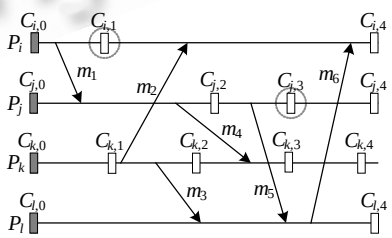



Fig.5 Direct scheme
图 5 直接策略

**定理 2**. 如果$P_i$的期望检查点周期$T_i$满足($T_i \bmod \tau$)=0,其中,($\tau \bmod \tau_0$)=0,那么,若$I_{j,y}\prec I_{i,x}$且$T_{j,y}\leq\lfloor(t_j+\tau)/\tau\rfloor\tau$,则有$T_{j,y}\leq T_{i,x}$.

证明:假设消息$m$由$I_{j,y}$发送至$I_{i,x}$.根据性质 1,有$T_{i,x}$迟于$m.TI\cdot\tau_0$.由于$T_i$是$\tau$的整数倍,因此$T_{i,x}\geq\lfloor(m.TI\cdot\tau_0+\tau)/\tau\rfloor\tau$.根据规则 3,$t_j=(x+m.TI)\tau_0$,其中,$0\leq x<1$.由于$(\tau \bmod \tau_0)=0$,故$\tau=k\tau_0(k\geq1)$.因此,$\lfloor(t_j+\tau)/\tau\rfloor\tau$可以写成$\lfloor(x+m.TI)/k+1\rfloor\tau$.设$m.TI=ck+b$,其中,$b=(m.TI \bmod k)$,那么$\lfloor(t_j+\tau)/\tau\rfloor\tau$可写成$(c+1+\lfloor(x+b)/k\rfloor)\tau$.由于$x<1,b<k$且$b$为整数,故$\lfloor(x+b)/k\rfloor=\lfloor b/k\rfloor=0$.因此,$\lfloor(t_j+\tau)/\tau\rfloor\tau=\lfloor(m.TI\cdot\tau_0+\tau)/\tau\rfloor\tau$.由于$T_{j,y}\leq\lfloor(t_j+\tau)/\tau\rfloor\tau$,所以$T_{j,y}\leq T_{i,x}$. □

由定理 2 可知,如果任意进程的期望检查点周期是$\tau$的整数倍,则当采用直接策略进行检查点周期调整时,$\mathcal{P}$模式可以满足.

2.3.2 间接策略

间接策略的思想是:如果进程$P_i$发觉检查点间隔$I_{i,x}$直接依赖于$I_{j,y}$并且检查点设置时刻$T_{i,x}$早于$T_{j,y}$,那么$T_{i,x}$将被更新为$T_{j,y}$.在具体实现中,可通过在消息中携带发送进程即将到达的检查点时刻,从而令消息接收进程作出相应的判断.需要指出的是,简单调整接收进程的检查点周期并不一定使$\mathcal{P}$模式满足,这就需要在进程$P_i$的局部时间$t_i$达到检查点设置时刻$T_{i,x}$后作进一步核实.通过依赖信息收集,如果$P_i$发现存在一个$I_{k,z}$满足$(I_{k,z}\prec^c I_{i,x})\wedge(T_{k,z}>T_{i,x})$,那么此时$P_i$同样需要将$T_{i,x}$调整到至少$T_{k,z}$.如果没有这样的$I_{k,z}$存在,那么此时才能设置一个新的检查点.显然,间接策略下,$\mathcal{P}$模式能够满足.间接策略的描述如下:

**规则 5**. 如果存在$I_{j,y}$使得$(I_{j,y}\prec^c I_{i,x})\wedge(T_{j,y}>T_{i,x})$,那么检查点设置时刻$T_{i,x}$需要被更新为$T_{j,y}$.

如图 6 所示($T_i=4\tau_0,T_j=\tau_0,T_k=2\tau_0,T_l=\tau_0$),当接收消息$m_1$时,$P_j$发现$I_{k,1}\prec I_{j,1}$并且$T_{k,1}>T_{j,1}(T_{k,1}=2\tau_0,T_{j,1}=\tau_0)$.$P_j$将$T_{j,1}$调整到 $2\tau_0$.类似地有,当接收消息$m_5$时,$P_j$发现$I_{i,1}\prec I_{j,2}$并且$T_{i,1}>T_{j,2}$,$P_j$将$T_{j,2}$调整到 $4\tau_0$.此外,当$P_l$的局部时间达到检查点设置时刻 $3\tau_0$时,通过依赖信息收集,$P_l$可以发现$I_{i,1}\prec^c I_{l,3}$,因此需要将$T_{l,3}$调整到 $4\tau_0$.此时,$P_l$的检查点周期将暂时扩大到 $2\tau_0$.

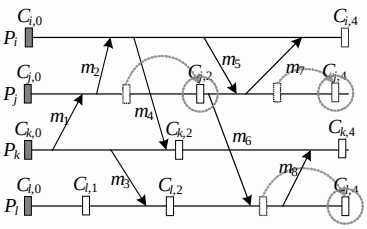



Fig.6 Indirect scheme

图 6 间接策略

下面我们将证明检查点设置时刻$T_{i,x}$调整的上界是确定的.在此之前,我们将给出所有与$I_{i,x}$有依赖关系的间隔$I_{j,y}$的起始时间上界.令符号⊙表示"起初".

**引理 2**. 设$\tau'$为各进程期望检查点周期的最小公倍数,经过间接检查点周期调整后,如果$I_{j,y}\prec^c I_{i,x}$,则有$T_{j,y-1}<\lfloor(T_{i,x}+\tau')/\tau'\rfloor\tau'$.

证明:用反证法.假设$\exists I_{j,y}:I_{j,y}\prec^c I_{i,x}$且$T_{j,y-1}\geq\lfloor(T_{i,x}+\tau')/\tau'\rfloor\tau'$.设$\Delta=\lfloor(T_{i,x}+\tau')/\tau'\rfloor\tau'$.不失一般性,假设任意$I_{l,r}$,如果$(I_{j,y}\prec^c I_{l,r})\wedge((I_{l,r}\prec^c I_{i,x})\vee(I_{l,r}=I_{i,x}))$,那么$T_{l,r-1}$早于$\Delta$.由于$\Delta$是$P_l$的检查点设置时刻,$T_{l,r}$起初不迟于$\Delta$(即⊙$(T_{l,r}\leq\Delta)$).令$I_{u,s}$为满足$(I_{j,y}\prec I_{u,s})\wedge((I_{u,s}\prec^c I_{i,x})\vee(I_{u,s}=I_{i,x}))$的检查点间隔,假设$m$的提交导致$I_{j,y}\prec I_{u,s}$.由于⊙$(T_{u,s}\leq\Delta)$并且$T_{j,y-1}\geq\Delta$,根据性质 1,在提交$m$前存在一个检查点间隔$I_{v,t}$其满足$(I_{v,t}\prec^c I_{i',x'})\wedge(I_{i',x'}\prec I_{u,s})\wedge(T_{v,t}>\Delta)\wedge(T_{i',x'-1}<\Delta)$.其中,$I_{i',x'}\neq I_{i,x}$.如果$T_{v,t-1}<\Delta$,那么有⊙$(T_{v,t}\leq\Delta)$并且$\exists I_{v',t'}:(I_{v',t'}\prec I_{v,t})\wedge(T_{v',t'}>\Delta)$.因此,只有当存在检查点间隔$I_{j',y'}$其满足$(I_{j',y'}\prec^c I_{i',x'})\wedge(T_{j',y'-1}\geq\Delta)$时,存在有限个检查点间隔使$I_{i',x'}$依赖.由此,我们找到了一对检查点间隔$I_{j',y'}$和$I_{i',x'}$,满足$(I_{j',y'}\prec^c I_{i',x'})\wedge(I_{i',x'}\prec^c I_{i,x})\wedge(T_{j',y'-1}\geq\Delta)\wedge(T_{i',x'-1}<\Delta)$.以此方法,我们同样还可得到另一对检查点间隔$I_{j'',y''}$和$I_{i'',x''}$,同样满足$(I_{j'',y''}\prec^c I_{i'',x''})\wedge(I_{i'',x''}\prec^c I_{i',x'})\wedge(T_{j'',y''-1}\geq\Delta)\wedge(T_{i'',x''-1}<\Delta)$.这里,$I_{i'',x''}$不同于$I_{i',x'}$和$I_{i,x}$.如上所述,我们将可找到无穷多个检查点间隔与当前的$I_{i,x}$有依赖关系,这与当前的任何一个检查点间隔仅存在有限个检查点间隔与其有依赖关系相矛盾. □

**定理 3**. 设$\tau'$为各进程期望检查点周期的最小公倍数,经过间接检查点周期调整后,如果$I_{j,y} \prec^C I_{i,x}$,那么有

$$T_{j,y} \leq \lfloor (T_{i,x}+\tau')/\tau' \rfloor \tau'.$$

定理 3 可根据引理 2 获得.对于$I_{j,y} \prec^C I_{i,x}$,由于$T_{j,y-1} \leq \lfloor (T_{i,x}+\tau')/\tau' \rfloor \tau'$,⊙$(T_{j,y} \leq \lfloor (T_{i,x}+\tau')/\tau' \rfloor \tau')$.如果$T_{j,y}$被迫进行调整,那么$I_{j,y}$一定依赖于一个检查点间隔$I_{k,z}, T_{k,z}>T_{j,y}$并且$P_k$的检查点设置时刻$T_{k,z}$尚未被调整.根据引理 2,$T_{k,z-1}<\lfloor (T_{i,x}+\tau')/\tau' \rfloor \tau'$,由此可知$T_{k,z} \leq \lfloor (T_{i,x}+\tau')/\tau' \rfloor \tau'$,因此,$T_{j,y}$调整后不会迟于$\lfloor (T_{i,x}+\tau')/\tau' \rfloor \tau'$.

**推论 1**. 设$\tau'$为各进程期望检查点周期的最小公倍数,经过间接检查点周期调整后,检查点设置时刻$T_{i,x}$不会超过$\lfloor (T_{i,x}+\tau')/\tau' \rfloor \tau'$.

2.3.3 混合策略

在本节中,我们将给出一个混合策略,从而将直接策略与间接策略结合起来,并且保证$\mathcal{P}$模式满足.在混合策略下,系统中的进程将被分配到不同组中,每个组的进程采用直接策略或者间接策略进行检查点周期调整.在采用直接策略的组中,进程是不允许进行检查点依赖信息收集的,因此,当产生跨组消息时会产生这样的情况,即采用间接策略组的进程依赖于采用直接策略组的进程.由于此时采用直接策略组进程的依赖信息无法获得,故$\mathcal{P}$模式可能不满足.这就需要在发送跨组消息时进行处理,从而可仅通过有限的依赖跟踪来保证$\mathcal{P}$模式满足.这里,我们将允许依赖跟踪的组称为I-Zone,其中的进程采用一种类似于间接策略的检查点周期调整策略;将不允许依赖跟踪的组称为D-Zone,采用直接策略进行检查点的周期调整.由于D-Zone中进程进行检查点周期调整时不需要进行依赖信息的收集,因此我们不对D-Zone规模作任何限定.换句话说,系统中仅需要维护一个D-Zone即可.这里,我们根据$\tau_d$将进程分配到不同的组,所有期望检查点周期小于$\tau_d$的进程被分配到各个I-Zone,其他进程被分配到一个D-Zone中.在进一步介绍混合策略之前,我们首先给出组内路径的定义.

**定义 7**. 组内路径$\mathcal{Z}$由一串消息$[m_1,m_2,m_3,\ldots,m_q](q \geq 1)$构成.对于$\mathcal{Z}$上的任何一个消息$m_k$,假设$m_k$从进程$P_i$发送至进程$P_j$,$m_k$满足下述条件:

1) $deliver(m_k)$与$send(m_{k+1})(1 \leq k<q)$在同一个检查点间隔执行;

2) $P_i$与$P_j$在同一个组中.

这里,用$I_{j,y} \rightsquigarrow_{\mathcal{I}} I_{i,x}$表示有一条组内路径$[m_1,m_2,m_3,\ldots,m_q]$由$I_{j,y}$至$I_{i,x}$,其中,$m_1$在间隔$I_{j,y}$发送,$m_q$在间隔$I_{i,x}$被提交.下面给出混合策略下检查点周期的调整方法.

**规则 6**. 混合策略分 3 种情况处理:

1) 假设消息$m$的传递导致$I_{j,y} \prec I_{i,x}$,$t_j$指示发送$m$时$P_j$的局部时间:

   a) 如果$m$是一个跨组消息或$P_j$为D-Zone进程,那么$T_{j,y}$直接调整为$\lfloor (t_j+\tau_d)/\tau_d \rfloor \tau_d$;

   b) 如果$m$是一个跨组消息且$P_i$接收$m$时$T_{j,y}>T_{i,x}$,那么$T_{i,x}$调整为$T_{j,y}$.

2) 对于I-Zone进程,如果存在组内路径$\mathcal{Z}$且满足$(I_{j,y} \rightsquigarrow_{\mathcal{I}} I_{i,x}) \wedge (T_{j,y}>T_{i,x})$,那么$T_{i,x}$调整至$T_{j,y}$.

下面对规则 6 的检查点周期调整进行解释.在情况 1a)中,当$m$是一个跨组消息且$P_j$是I-Zone进程时,$P_j$也采用了直接调整方式,检查点设置时刻的选择方式类似于一个D-Zone消息发送进程.这是为了确保接收进程的检查点设置时刻$T_{i,x}$根据$T_{j,y}$进行调整后不会再次调整,从而避免了跨组的依赖信息收集.需要指出的是,如果消息接收进程为D-Zone进程,那么接收进程无须进行任何被动的检查点周期调整.因此,情况 1b)中,$P_i$实际上一定是一个I-Zone进程.如果$T_{j,y}>T_{i,x}$,为了保证$\mathcal{P}$模式满足,需要延长$P_i$的检查点周期.情况 2)给出了I-Zone进程通过组内路径收集依赖关系信息后进行的检查点周期调整,这里采用了间接检查点周期调整策略.在依赖信息收集过程中,若存在$I_{j,y} \prec^C I_{i,x}$,且检查点$C_{j,l}(l=CI(I_{j,y-1}))$已经设置,如果检查点设置符合$\mathcal{P}$模式,则此时不存在$I_{k,z}:(I_{k,z} \prec^C I_{j,y-1}) \wedge (T_{k,z} \geq T_{j,y})$,因此,$I_{k,z} \prec^C I_{j,y-1}$对于$I_{i,x}$的调整实际上是无用的.在跟踪过程中,如果存在$\mathcal{Z}:I_{j,y} \rightsquigarrow_{\mathcal{I}} I_{i,x}(T_{j,y} \leq T_{i,x})$并且$I_{j,y}$已经结束,那么所有$I_{j,y}$依赖的检查点间隔不会引起$T_{i,x}$的调整,因此无须进一步沿$\mathcal{Z}$收集依赖信息.

图 7 所示为混合策略下检查点周期调整的一个示例($\tau_d=2\tau_0, T_1=4\tau_0, T_2=4\tau_0, T_3=4\tau_0, T_4=4\tau_0, T_5=\tau_0, T_6=\tau_0, T_7=\tau_0, T_8=\tau_0$).当消息$m_5$由$P_6$发送时,由于$P_3$是D-Zone进程,$P_6$需要将其检查点周期调整到 $2\tau_0$,然后才可以发送$m_5$.当进程$P_6$的检查点$C_{6,2}$设置完成后,$P_6$的检查点周期恢复到$\tau_0$.当$P_1$发送$m_1$时$(t_1<\tau_0)$,$P_1$将其检查点设置时刻调整到最

近的且为$\tau_d$整数倍的一个时刻 $2\tau_0$.类似地,当$P_3$发送$m_3(t_3<\tau_0)$时,$P_3$将检查点设置时刻调整为 $2\tau_0$.在$P_5$接收到$m_3$后,由于$T_{3,1}>T_{5,1}$,$T_{5,1}$被调整到 $2\tau_0$.当$P_8$的局部时间达到检查点设置时刻 $3\tau_0$时,由于$I_{6,2}\rightsquigarrow_I I_{8,3}$且$T_{6,2}>T_{8,3}$,$T_{8,3}$被调整为 $4\tau_0$($T_{6,2}=4\tau_0$).类似地,当$t_5$达到$3\tau_0$时,$P_5$需要将检查点时刻$T_{5,2}$调整到$4\tau_0$.

下面我们将证明在混合策略下,即使采用有限的依赖信息收集,仍然可以保证$\mathcal{Z}$模式是可以满足的.首先需要指出,I-Zone 中的每个进程,其检查点设置时刻的调整是存在上界的.

**定理 4**. 假设$P_i$是I-Zone进程,$I_{i,x}$为其当前检查点间隔,那么$T_{i,x}$所能调整到的最大值不超过$\lfloor(T_{i,x}+\tau_d)/\tau_d\rfloor\tau_d$.

证明:$T_{i,x}$进行调整有如下两种可能:

情况 1:$P_i$在$I_{i,x}$接收到来自其他组的消息$m$,假设$m$发送自$I_{j,y}$,有$T_{i,x}<T_{j,y}$.$m$可在$I_{i,x}$提交,除非$m.TI\cdot\tau_0<T_{i,x}$.根据规则 6,$T_{j,y}=\lfloor(m.TI\cdot\tau_0+\tau_d)/\tau_d\rfloor\tau_d$.由于$m.TI\cdot\tau_0<T_{i,x}$,故$T_{j,y}\leqslant\lfloor(T_{i,x}+\tau_d)/\tau_d\rfloor\tau_d$.因此,$T_{i,x}$调整后不超过$\lfloor(T_{i,x}+\tau_d)/\tau_d\rfloor\tau_d$.

情况 2:I-Zone中存在组内路径$\mathcal{Z}$且满足$(I_{j,y}\rightsquigarrow_I I_{i,x})\wedge(T_{j,y}>T_{i,x})$.由于$\tau_d$是$P_i$组内进程期望检查点周期的公倍数,$\lfloor(T_{i,x}+\tau_d)/\tau_d\rfloor\tau_d$是一个公共检查点设置时刻.由于依赖信息仅收集自同组进程,根据推论 1,$T_{i,x}$调整后不超过$\lfloor(T_{i,x}+\tau_d)/\tau_d\rfloor\tau_d$. □

**性质 2**. 如果$P_i$,$P_j$在同一组中,那么若$I_{j,y}\prec I_{i,x}$,则有$\Diamond(T_{j,y}\leq T_{i,x})$.

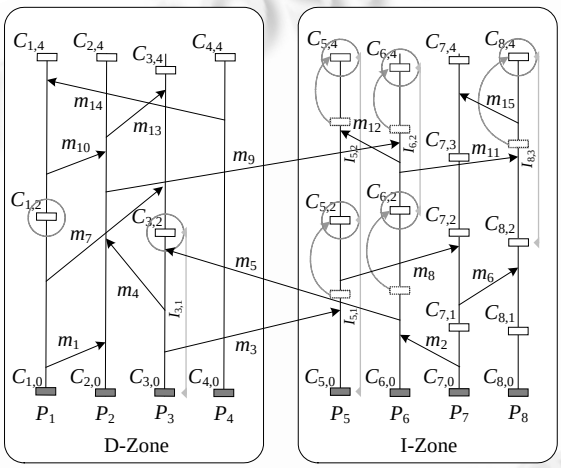



Fig.7 Hybrid scheme

图 7 混合策略

**定理 5**. 对任意检查点间隔$I_{i,x}$,$I_{j,y}$,如果$I_{j,y}\prec I_{i,x}$,那么$\Diamond(T_{j,y}\leq T_{i,x})$.

证明:如果$P_i$,$P_j$在同一组中,可知$\Diamond(T_{j,y}\leq T_{i,x})$.这里,我们仅讨论$P_i$,$P_j$处于不同组的情形,假设消息$m$的提交导致$I_{j,y}\prec I_{i,x}$.

情况 1:$P_i$位于I-Zone.由于$m$在$I_{i,x}$提交,根据性质 1,有$m.TI\cdot\tau_0<T_{i,x}$.若$T_{i,x}<T_{j,y}$,由于$m.TI\cdot\tau_0<T_{i,x}$并且$T_{j,y}=\lfloor(m.TI\cdot\tau_0+\tau_d)/\tau_d\rfloor\tau_d$,有$\lfloor(T_{i,x}+\tau_d)/\tau_d\rfloor\tau_d=\lfloor(m.TI\cdot\tau_0+\tau_d)/\tau_d\rfloor\tau_d$.根据规则 6,$T_{i,x}$将调整到$T_{j,y}$.根据定理 4,此后$T_{i,x}$不发生改变,发送进程的检查点设置时刻$T_{j,y}$亦不会再次改变,故$\Diamond(T_{j,y}\leq T_{i,x})$成立.

情况 2:$P_i$位于D-Zone.根据规则 6,$T_{j,y}=\lfloor(m.TI\cdot\tau_0+\tau_d)/\tau_d\rfloor\tau_d$.由于$m$在$I_{i,x}$提交,根据性质 1,有$m.TI\cdot\tau_0<T_{i,x}$.由于$T_i$是$\tau_d$的整数倍,因此有$T_{i,x}\geq\lfloor(m.TI\cdot\tau_0+\tau_d)/\tau_d\rfloor\tau_d$.加之$P_j$发送$m$后不会调整$T_{j,y}$,因此,$\Diamond(T_{j,y}\leq T_{i,x})$成立. □

**推论 2**. 在混合策略下的检查点周期调整有$\forall I_{i,x},I_{j,y}:I_{j,y}\prec^c I_{i,x}\Rightarrow\Diamond(T_{j,y}\leq T_{i,x})$.

## 3 性能分析

在本节中,我们将对比STBMC协议与CIC协议的性能.CIC协议同样允许进程采用不同周期进行检查点设置.CIC协议通常分为 3 类[14]:严格无Z-path(strictly Z-path free,简称SZPF)、无Z-path(Z-path free,简称ZPF)以及无Z-cycle(Z-cycle free,简称ZCF).由于ZCF类协议对于避免非因果Z-path的要求最低,因此往往可以获得相对较

低的检查点开销.比较常见的ZCF类协议有Manivannan-Singhal提出的MS[15]协议、Helary等人提出的HMNR[19]协议.在MS与HMNR协议中,表示逻辑时间的控制信息将会携带在每个消息中发送给其他进程.通过该信息进程,可以决定是否通过设置强迫检查点来保证逻辑时间沿着Z-path总是递增的,从而避免无用检查点的出现.在MS协议中,恢复阶段允许设置强迫检查点以降低计算损失.该机制同样可以用于HMNR以及STBMC协议中.考虑到恢复的效率,这里我们关注恢复阶段无强迫检查点要求下各个协议带来的计算损失.在仿真过程中,我们作如下假设:

1) 每个进程存在一个期望检查点设置周期且允许不同.进程$P_i$的期望检查点周期即$T_i$.进程的检查点周期允许临时调整.
2) 假设当前的检查点设置时刻为$T_{i,x}$,检查点设置后,新的检查点设置时刻初始化为$\lfloor (T_{i,x}+T_i)/T_i \rfloor T_i$.
3) 进程在恢复过程中选择最大一致状态进行恢复.

### 3.1 仿真模型

为了获得各个协议在突发通信模式下的性能,这里我们给出一个仿真模型,与其他一些模型,如文献[13,22,24,25]相类似.该模型可以表示为一个六元组$\langle N_p, T_{adj}, c, p_{com}, p_{snd}, \mu\rangle$,其中,$N_p$是系统中进程的数目.对于每个进程$P_i$来说,存在一些偏好进程(favorite process)[26−28]与其通信频繁,$P_i$会将消息随机地发送给相对于它而言的偏好进程.每个进程的偏好进程是在初始时随机选择的,并且每经过$T_{adj}$时间可以重新选择.每个进程的偏好进程数目服从均匀分布$N_p \times U(c)$[24].对于每个进程来说存在两种状态,即通信状态和静默状态.进程进入通信状态的概率为$p_{com}$.当一个进程进入通信状态后执行内部操作以及通信操作,其概率分别为 $1-p_{snd}$和$p_{snd}$.执行每个操作的时间服从指数分布,该指数分布具有均值$\mu(\mu=1)$.需要指出的是,在每个通信操作结束时,消息方可发送至目标进程.如果一个进程进入了静默状态,那么它仅执行内部操作,不会发送任何消息.每个进程持续一个状态的时间为$\tau_0$.在仿真过程中,每个进程会产生$n_f$个故障,故障产生时间服从均匀分布$T \times U(0.5)$,其中,$T(T=1000)$是仿真时间.用$f_b$表示基本检查点频率,$f_b=\tau_0/T$.本文所使用的性能指标如下:

1) 卷回偏移(rollback deviation,简称RD).RD用于描述卷回距离与期望卷回距离的偏差,这里我们假定$T_i/2$是$P_i$的期望卷回距离,$cl_i$是进程$P_i$的平均卷回距离.RD可以写成如下形式:

$$RD = \sum_i \left| \frac{cl_i}{T_i} - \frac{1}{2} \right|.$$

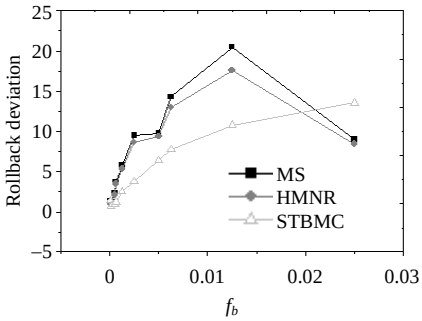
2) 总计算损失(total computation loss,简称TCL). $TCL = \sum_i tcl_i$, 其中,$tcl_i$是进程$P_i$的总卷回距离.

3) 检查点强度比(checkpoint intensity ratio,简称CIR).CIR是实际检查点数目与基本检查点数目的比值.假设系统存在$n$种期望检查点周期$\eta_1, \eta_2, \ldots, \eta_n$.用$N_k$表示期望检查点周期等于$\eta_k$的进程数目,$N_{basic}$表示基本检查点数目.$N_{basic}$可写成 $N_{basic} = T\sum_i N_i / \eta_i$.

### 3.2 仿真结果

在仿真过程中,参数设置如下:$T_{adj}=50$,$p_{snd}=0.2$,$p_{com}=0.5$,$n_f$ =100.图 8~图 11 给出了各个协议的卷回偏移.可以看出,当$f_b$还没有变得很高时,STBMC的卷回偏移要小于其他协议;随着$f_b$的提高,将会出现更多的跨组消息,这时进程更趋向于采用基准检查点周期进行检查点设置,这意味着进程常常不能采用其期望检查点周期,进而影响了对卷回距离的控制.但是,由于本文所述的进程分组实际上是一个逻辑分组,在实际应用过程中,可以通过分组优化来降低跨组消息数目,从而减小检查点周期的变化.

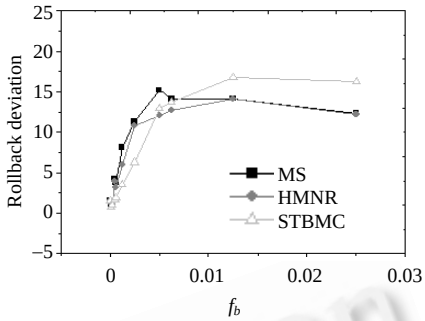
图 12~图 15 给出了进程总的计算损失.STBMC在$f_b$偏小时表现良好;随着$f_b$的增大,STBMC的总计算损失将会超过MS和HMNR;但是可以看出,STBMC卷回偏移的变化要远小于计算损失的变化.这主要由于STBMC具有使临近检查点用于恢复的能力;而在MS和HMNR协议中可能出现恢复阶段临近检查点被跳过的现象,其带来卷回偏移的增大.这在期望检查点周期较小的那些进程上体现得更为明显.实际上,卷回偏移体现了假设各进程期望检查点周期内的计算重要性相同时的一种计算损失.




$N_1$=8,$N_2$=8,$N_3$=8,$\eta_1=\tau_0,\eta_2=4\tau_0,\eta_3=16\tau_0,\tau_d=\eta_2,c$=0.1

Fig.8

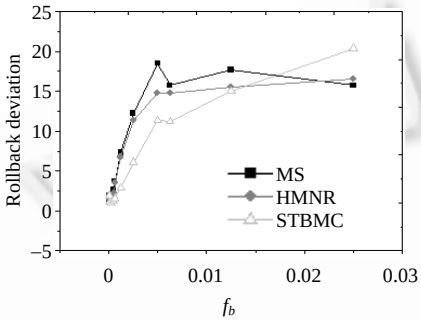
图 8




$N_1$=8,$N_2$=8,$N_3$=8,$\eta_1=\tau_0,\eta_2=4\tau_0,\eta_3=16\tau_0,\tau_d=\eta_2,c$=0.5

Fig.9

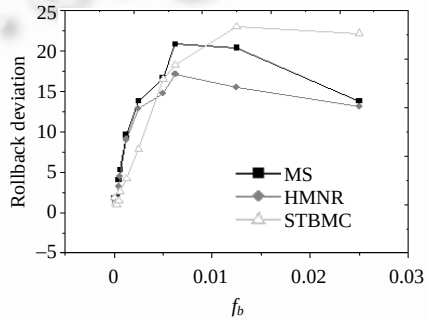
图 9




$N_1$=8,$N_2$=8,$N_3$=8,$N_4$=8,$\eta_1=\tau_0,\eta_2=2\tau_0,\eta_3=4\tau_0,\eta_4=16\tau_0,\tau_d=\eta_3,c$=0.1

Fig.10

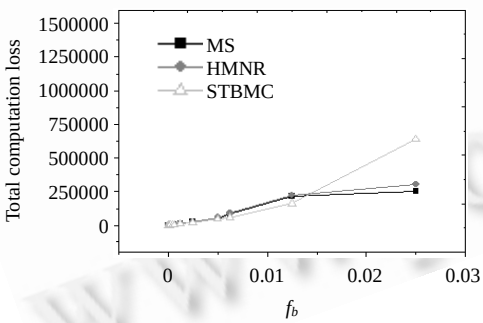
图 10




$N_1$=8,$N_2$=8,$N_3$=8,$N_4$=8,$\eta_1=\tau_0,\eta_2=2\tau_0,\eta_3=4\tau_0,\eta_4=16\tau_0,\tau_d=\eta_3,c$=0.5

Fig.11

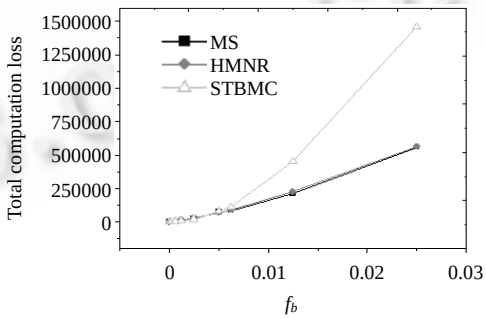
图 11




$N_1$=8,$N_2$=8,$N_3$=8,$\eta_1=\tau_0,\eta_2=4\tau_0,\eta_3=16\tau_0,\tau_d=\eta_2,c$=0.1

Fig.12

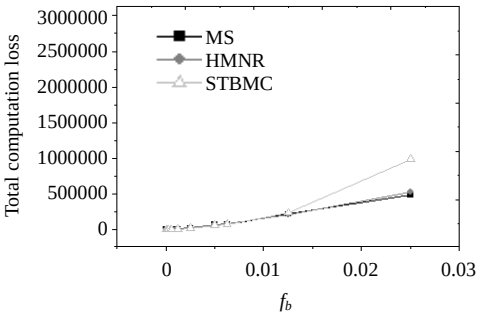
图 12




$N_1$=8,$N_2$=8,$N_3$=8,$\eta_1=\tau_0,\eta_2=4\tau_0,\eta_3=16\tau_0,\tau_d=\eta_2,c$=0.5

Fig.13

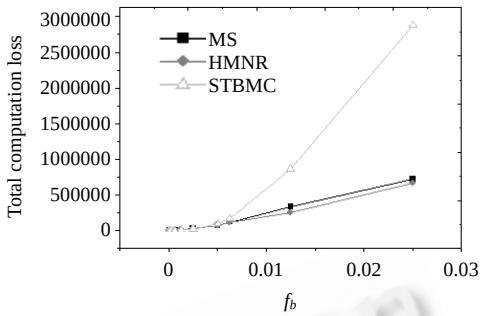
图 13




$N_1$=8,$N_2$=8,$N_3$=8,$N_4$=8, $\eta_1=\tau_0, \eta_2=2\tau_0, \eta_3=4\tau_0, \eta_4=16\tau_0, \tau_d=\eta_3, c$=0.1

Fig.14

图 14

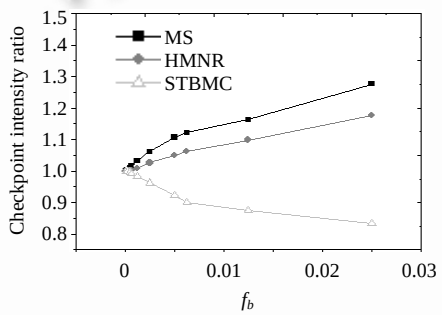



$N_1$=8,$N_2$=8,$N_3$=8,$N_4$=8, $\eta_1=\tau_0, \eta_2=2\tau_0, \eta_3=4\tau_0, \eta_4=16\tau_0, \tau_d=\eta_3, c$=0.5

Fig.15

图 15

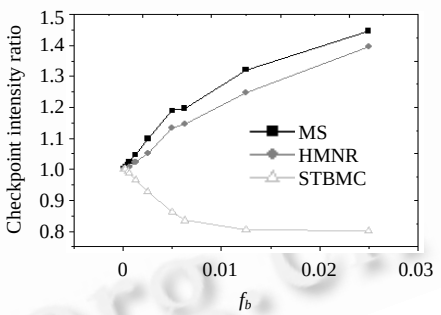
图 16~图 19 给出了检查点强度比.检查点强度比体现了协议的检查点开销.可以看出,STBMC的CIR要小于其他协议.在STBMC中,不会出现由于接收到的消息中携带了一个较新的逻辑时间而引发设置额外检查点的情况.所有的额外检查点仅当采用直接策略时会被设置,其存在于D-Zone内的消息发送进程.然而在HMNR和MS协议中均会出现随着较新逻辑时间通过发送消息的扩散带来强迫检查点数目增长的现象.可以看出,当每个$\tau_0$内消息发送数目增多时,HMNR以及MS协议的检查点开销会很快增长.我们可以看到,STBMC是在较小的检查点开销下完成计算损失控制的,这与仅通过频繁的检查点设置来控制计算损失的方法有着明显的不同.




$N_1$=8,$N_2$=8,$N_3$=8, $\eta_1=\tau_0, \eta_2=4\tau_0, \eta_3=16\tau_0, \tau_d=\eta_2, c$=0.1

Fig.16

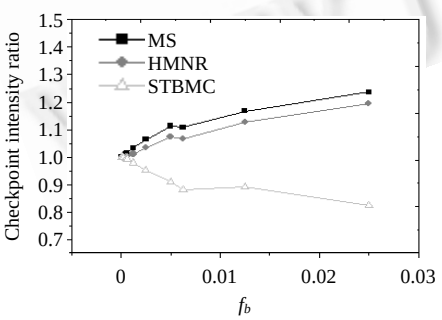
图 16




$N_1$=8,$N_2$=8,$N_3$=8, $\eta_1=\tau_0, \eta_2=4\tau_0, \eta_3=16\tau_0, \tau_d=\eta_2, c$=0.5

Fig.17

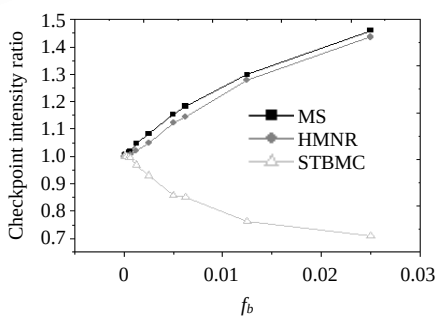
图 17




$N_1$=8,$N_2$=8,$N_3$=8,$N_4$=8, $\eta_1=\tau_0, \eta_2=2\tau_0, \eta_3=4\tau_0, \eta_4=16\tau_0, \tau_d=\eta_3, c$=0.1

Fig.18

图 18




$N_1$=8,$N_2$=8,$N_3$=8,$N_4$=8, $\eta_1=\tau_0, \eta_2=2\tau_0, \eta_3=4\tau_0, \eta_4=16\tau_0, \tau_d=\eta_3, c$=0.5

Fig.19

图 19

## 4 结 论

本文提出了一种基于时间的多周期检查点算法.该算法允许各个进程采用不同的周期进行检查点设置.为了避免恢复阶段的无用检查点,本文提出了一种混合检查点周期调整策略,通过这样的调整,使得RDT模式能够满足.在混合策略下,进程将被划分到多个逻辑组中,并且依赖信息仅对组内进程可见.如此,依赖信息收集量可以得到较好的控制,从而使检查点算法具有良好的可扩展性.此外,我们证明了在依赖信息局部可见条件下,混合策略能够保证RDT模式满足.实验结果显示,STBMC 是一种可通过较低检查点开销获得良好卷回距离控制的检查点算法.

**References**:


[1] Wang YM, Chung PY, Lin IJ, Fuchs WK. Checkpoint space reclamation for uncoordinated checkpointing in message-passing systems. IEEE Trans. on Parallel and Distributed Systems, 1995,6(5):546−554.

[2] Wang YM, Fuchs WK. Optimal message log reclamation for uncoordinated checkpointing. In: Proc. of the Conf. on Fault-Tolerant Parallel and Distributed Systems. Piscataway: IEEE Computer Society Press, 1995. 24−29.

[3] Gupta B, Rahimi S, Yang Y. A novel roll-back mechanism for performance enhancement of asynchronous checkpointing and recovery. Informatica, 2007,31(1):1−13.

[4] Elnozahy EN, Johnson DB, Zwaenepoel W. The performance of consistent checkpointing. In: Proc. of the 11th Symp. on Reliable Distributed Systems. 1992. 39−47.

[5] Koo R, Toueg S. Checkpointing and rollback-recovery for distributed systems. IEEE Trans. on Software Engineering, 1987, SE-13(1):23−31.

[6] Cao G, Singhal M. Low-Cost checkpointing with mutable checkpoints in mobile computing systems. In: Proc. of the 18th Int'l Conf. on Distributed Computing Systems. Piscataway: IEEE Computer Society Press, 1998. 464−471.

[7] Sakata TC, Garcia IC. Non-Blocking synchronous checkpointing based on rollback-dependency trackability. In: Proc. of the 25th IEEE Symp. on Reliable Distributed Systems. Piscataway: IEEE Computer Society Press, 2006. 411−420.

[8] Tong Z, Kain RY, Tsai WT. A low overhead checkpointing and rollback recovery scheme for distributed systems. In: Proc. of the 8th Symp. on Reliable Distributed Systems. Piscataway: IEEE Computer Society Press, 1989. 12−20.

[9] Cristian F, Jahanian F. A timestamp-based checkpointing protocol for long-lived distributed computations. In: Proc. of the 10th Symp. on Reliable Distributed Systems. Piscataway: IEEE Computer Society Press, 1991. 12−20.

[10] Kavanaugh GP, Sanders WH. Performance analysis of two time-based coordinated checkpointing protocols. In: Proc. of the Pacific Rim Int'l Symp. on Fault-Tolerant Systems. Los Alamitos: IEEE Computer Society Press, 1997. 194−201.

[11] Neves N, Fuchs WK. Using time to improve the performance of coordinated checkpointing. In: Proc. of the Int'l Computer Performance and Dependability Symp. Los Alamitos: IEEE Computer Society Press, 1996. 282−291.

[12] Neves N, Fuchs WK. Coordinated checkpointing without direct coordination. In: Proc. of the IEEE Int'l Computer Performance and Dependability Symp. 1998. 23−31.

[13] Baldoni R, Quaglia F, Fornara P. An index-based checkpointing algorithm for autonomous distributed systems. IEEE Trans. on Parallel and Distributed Systems, 1999,10(2):181−192.

[14] Manivannan D, Singhal M. Quasi-Synchronous checkpointing: Models, characterization, and classification. IEEE Trans. on Parallel and Distributed Systems, 1999,10(7):703−713.

[15] Manivannan D, Singhal M. A low-overhead recovery technique using quasi-synchronous checkpointing. In: Proc. of the 16th Int'l Conf. on Distributed Computing Systems. Piscataway: IEEE Computer Society Press, 1996. 100−107.

[16] Baldoni R, Helary JM, Mostefaoui A, Raynal M. A communication-induced checkpointing protocol that ensures rollback-dependency trackability. In: Proc. of the 27th Annual Int'l Symp. on Fault-Tolerant Computing. Washington: IEEE Computer Society Press, 1997. 68−77.

[17] Baldoni R, Quaglia F, Ciciani B. A VP-accordant checkpointing protocol preventing useless checkpoints. In: Proc. of the 17th IEEE Symp. on Reliable Distributed Systems. Los Alamitos: IEEE Computer Society Press, 1998. 61−67.

[18] Alvisi L, Elnozahy E, Rao S, Husain SA, Mel AD. An analysis of communication-induced checkpointing. In: Proc. of the 29th Annual Int'l Symp. on Fault-Tolerant Computing. Washington: IEEE Computer Society Press, 1999. 242−249.

[19] Helary JM, Mostefaoui A, Netzer RHB, Raynal M. Communication-Based prevention of useless checkpoints in distributed computations. Distributed Computing, 2000,13(1):29−43.

[20] Tsai J. On properties of RDT communication-induced checkpointing protocols. IEEE Trans. on Parallel and Distributed Systems, 2003,14(8):755−764.

[21] Randell B. System structure for software fault tolerance. IEEE Trans. on Software Engineering, 1975,SE-1(2):220−232.

[22] Ci YW, Zhang Z, Zuo DC, Wu ZB, Yang XZ. Communication-Based prevention of non-∧pattern. In: Proc. of the 28th IEEE Symp. on Reliable Distributed Systems. Washington: IEEE Computer Society Press, 2009. 129−134.

[23] Netzer RHB, Xu J. Necessary and sufficient conditions for consistent global snapshots. IEEE Trans. on Parallel and Distributed Systems, 1995,6(2):165−169.

[24] Alvisi L, Bhatia K, Marzullo K. Causality tracking in causal message-logging protocols. Distributed Computing, 2002,15(1):1−15.

[25] Kim J, Lilja DJ. Exploiting multiple heterogeneous networks to reduce communication costs in parallel programs. In: Proc. of the 6th Heterogeneous Computing Workshop. Los Alamitos: IEEE Computer Society Press, 1997. 83−95.

[26] Kim J, Lilja DJ. Characterization of communication patterns in message-passing parallel scientific application programs. In: Proc. of the 2nd Int'l Workshop on Network-Based Parallel Computing. London: Springer-Verlag, 1998. 202−216.

[27] Chodnekar S, Srinivasan V, Vaidya AS, Sivasubramaniam A, Das CR. Towards a communication characterization methodology for parallel applications. In: Proc. of the 3rd Int'l Symp. on High-Performance Computer Architecture. Los Alamitos: IEEE Computer Society Press, 1997. 310−319.

[28] Vetter JS, Mueller F. Communication characteristics of large-scale scientific applications for contemporary cluster architectures. In: Proc. of the Int'l Symp. on Parallel and Distributed Processing. Washington: IEEE Computer Society Press, 2002. 27−36.

**慈轶为**(1981－),男,博士生,CCF 学生会员,主要研究领域为容错计算,移动计算.

**张展**(1978－),男,博士,讲师,CCF 会员,主要研究领域为容错计算,移动计算.

**左德承**(1972－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为容错计算,移动计算.

**吴智博**(1954－),男,博士,教授,博士生导师,CCF 高级会员,主要研究领域为容错计算,移动计算.

**杨孝宗**(1939－),男,教授,博士生导师,CCF 高级会员,主要研究领域为容错计算,移动计算.