第33卷 第12期 计 算 机 工 程 2007年6月
Vol.33 No.12 Computer Engineering June 2007
·软件技术与数据库· 文章编号：1000—3428(2007)12—0084—03 文献标识码：A 中图分类号：TP391

一种降低并行程序检查点开销的方法

周小成[1,2]，孙凝晖[2]，霍志刚[1,2]，马 捷[2]
(1. 中国科学院研究生院，北京 100080；2. 中国科学院计算技术研究所，北京 100080)

摘 要：检查点设置和卷回恢复是提高系统可靠性和实现容错计算的有效途径，其性能通常用开销率来评价，而检查点开销是影响开销率的主要因素。针对目前并行程序运行时存在较多通信阻塞时间的现状，该文在写时复制检查点缓存的基础上提出了一种进一步降低检查点开销的方法。通过控制状态保存线程的调度和选择合适的状态保存粒度，该方法能很好地利用通信阻塞时间隐藏状态保存线程运行时带来的开销，从而能进一步降低开销率。

关键词：检查点设置和卷回恢复；检查点开销；通信阻塞时间

Method for Reducing Checkpoint Overhead of Parallel Program

ZHOU Xiaocheng[1,2], SUN Ninghui[2], HUO Zhigang[1,2], MA Jie[2]
(1. Graduate School of Chinese Academy of Sciences, Beijing 100080;
2. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080)

【Abstract】 Checkpointing and rollback recovery is an effect way to improve system reliability and implement fault-tolerant computation. It is usually evaluated by overhead ratio, which is primarily effected by checkpoint overhead. As there is much communication blocking time while parallel program is running, a method based on copy-on-write checkpoint buffering is proposed to further reduce checkpoint overhead. By controlling the running of checkpointing thread and selecting a suitable granularity, the method can hide the overhead caused by checkpointing thread very well and thus reduce overhead ratio.

【Key words】 Checkpointing and rollback recovery; Checkpoint overhead; Communication blocking time

1 概述

检查点设置和卷回恢复(checkpointing and rollback recovery)是在规模不断扩大的机群系统中进行故障恢复简单而有效的方法。系统在应用正常运行过程中定期保存其运行状态(检查点设置)，当系统发生故障时，在修复该故障后，系统根据最近保存的检查点即可恢复应用的运行(卷回恢复)。由于减少了系统发生故障时的损失，当应用需要在有一定故障率的系统中长时间运行时，检查点设置和卷回恢复技术可以有效降低其运行时间。

检查点设置和卷回恢复技术的性能通常用开销率(overhead Ratio)来评价，其定义为应用每单位有效运行时间由于系统故障及检查点设置和卷回恢复增加的运行时间[1]。检查点设置的性能是影响开销率的重要因素。衡量检查点设置性能主要有两个指标：检查点开销(checkpoint overhead)和检查点延迟(checkpoint latency)。检查点开销是指每做一次检查点设置操作应用增加的运行时间；检查点延迟是指从发起一个检查点设置操作到这个检查点可用之间的时间间隔。相对于检查点延迟，检查点开销对开销率的影响更大[1]。

检查点开销隐藏是降低检查点开销的一类重要的方法，如检查点缓存(checkpoint buffering)和写时复制检查点缓存(copy-on-write checkpoint buffering)等。通过使程序的正常操作和保存检查点状态的操作并行执行，检查点缓存和写时复制检查点缓存都能大幅度降低检查点开销。由于避免了不必要的内存拷贝，写时复制检查点缓存效果更好[2]。

通过测试可以看到，即使应用了写时复制检查点缓存技术，设置检查点时仍然存在较大的检查点开销。因此，针对目前并行程序由于进程间需要同步和交换数据而存在较多通信阻塞时间的现状，本文在写时复制检查点缓存的基础上提出了一种利用并行程序的通信阻塞时间来隐藏检查点开销的方法。通过控制状态保存线程的调度和选择合适的状态保存粒度，该方法能使状态保存线程的运行时间和目标进程的通信阻塞时间尽量重合，从而可以进一步隐藏状态保存线程运行时带来的开销。测试结果表明，该优化方法在写时复制检查点缓存的基础上进一步降低了检查点开销，从而进一步降低了开销率。

2 方法设计

2.1 检查点开销分析

一个并行检查点系统中的检查点开销主要包括获取系统全局一致状态[3]的协议开销和保存检查点状态的开销。当采用写时复制检查点缓存技术保存检查点状态时，由于需要启动一个状态保存线程在目标进程运行的同时保存检查点状态，因此除了目标进程运行过程中修改页面引起的复制页面的开销外，保存检查点状态的开销还包括状态保存线程占用处理器时可能导致的开销。

表1给出了在本文测试平台上对LINPACK设置一个检查点时的开销分布。测试时每个进程的检查点状态大小约为

基金项目：中国科学院新一代机群关键技术研究基金资助项目(KGCX2-SW-116)
作者简介：周小成(1979－)，男，博士生，主研方向：机群通信系统和容错技术；孙凝晖，研究员；霍志刚，博士生；马 捷，副研究员
收稿日期：2006-06-30 **E-mail：**zxc@ncic.ac.cn

282.38MB。从表 1 中可以看出：应用写时复制检查点缓存技术后，仍然存在较大的检查点开销，当并行程序中的进程数增加(获取全局一致状态协议的开销将会增加)或检查点状态大小增大(状态保存线程的开销将会增加)时，检查点开销将随之增加；获取一致状态协议的开销较小，状态保存线程的开销是检查点开销中的主要部分。因此，本文的研究重点是如何进一步隐藏由于状态保存线程引起的开销。

表 1 检查点开销分布

	开销/s	占用比例/%
一致状态协议	0.13	5.24
写时复制	0.65	26.21
状态保存线程	0.97	39.11
其它	0.73	29.44
总计	2.48	100

2.2 并行程序通信阻塞时间

由于应用固有的串行特性，并行程序的各个进程并不能完全并行，它们在运行过程中需要同步和交换数据来进行协作。因此，并行程序在运行过程中存在大量进程间同步和交换数据导致的通信阻塞时间。表 2 中的数据表明了这一点：LINPACK、IS(C)和 MM5 在本文的测试平台上运行时通信阻塞时间分别占全部运行时间的 12.96%、47.19%和 6.14%。并行程序的各进程间在运行过程中需要更多的同步和交换数据操作，当节点规模扩大时，通信阻塞时间比例将进一步增加。

表 2 通信阻塞时间比例

	运行/s	通信阻塞/s	占用比例/%
LINPACK	135.438	12.137	12.96
IS(C)	69.945	33.009	47.19
MM5	113.070	6.941	6.14

2.3 利用通信阻塞时间隐藏检查点开销的方法

从上面的分析可以看出，即使应用了写时复制检查点缓存技术，设置检查点时仍然存在较大的检查点开销。同时，由于进程间的同步和交换数据操作，并行程序在运行过程中存在大量通信阻塞时间。因此本文试图在写时复制检查点缓存技术的基础上，通过控制状态保存线程的调度，利用并行程序的通信阻塞时间进一步隐藏检查点开销。图 1 给出了一个利用并行程序的通信阻塞时间隐藏检查点开销的示意图。在图中，假设并行程序正常运行时，各进程独占一个 CPU，不存在进程间通信之外的 I/O 操作，因此目标进程只有“运行”和“通信阻塞”两种状态。当发起一个检查点设置操作时，并行程序中各进程将停止运行一段时间用于获取一致状态，然后继续运行并启动状态保存线程负责保存检查点状态到本地硬盘。由于 DMA 技术的应用，向硬盘中写数据的过程 CPU 无须参与，因此状态保存线程大部分时间都处于等待写硬盘操作完成的状态。

当 CPU 数量足够多时，目标进程可以和状态保存线程完全不干扰地并发执行(图 1(a))。但在大部分系统中，并行程序运行时节点上的 CPU 都被占用，因此当状态保存线程被调度执行时，目标进程需要等待调度。此过程中，目标进程等待调度的时间可能和其运行过程中的通信阻塞时间不重合、重合一部分或完全重合，前两种情况将导致保存检查点状态的开销(图 1(b))。如果能够控制状态保存线程被调度执行的时机，使得只有当目标进程处于通信阻塞状态时状态保存线程才被调度执行，那么目标进程等待调度的时间将和其通信阻塞时间基本重合，从而能隐藏状态保存线程运行时带来的开销。由于需要等待调度时机，该技术可能会带来检查点延迟的增加(图 1(c))。

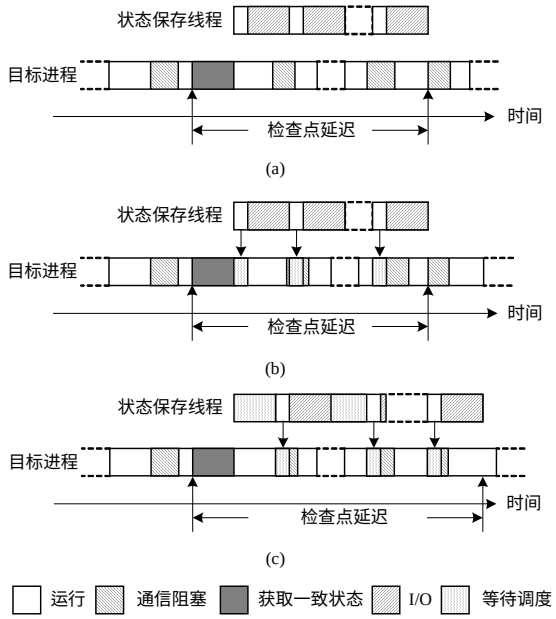

图 1 利用通信阻塞时间隐藏检查点开销的方法

为实现以上隐藏检查点开销的思路，需要解决以下两个关键问题：(1)控制状态保存线程的调度，使得只有当目标进程进入“通信阻塞”状态后状态保存线程才可以立即被调度执行。(2)选择合适的状态保存粒度，即状态保存线程被调度执行时，保存多少检查点状态后再次进入睡眠。

3 实现

本文的原型系统基于 BLCR(berkeley lab checkpoint/restart)和 LAM/MPI 而实现。BLCR 是美国伯克利国家实验室开发的一个检查点系统。利用 BLCR 提供的检查点设置回调机制，LAM/MPI 实现了 MPI 应用的并行检查点。

由于 BLCR 不支持写时复制检查点缓存，因此本文在 BLCR 中首先增加了对写时复制检查点缓存的支持。同时，为了避免不必要的文件缓存的开销，本文修改了 BLCR 使得检查点状态以非缓存方式(Direct I/O)写入检查点文件。从 4.2 节的实验结果可以看到，这两个修改大幅度降低了检查点开销。

3.1 控制状态保存线程的调度

本文利用一个全局信号量 *s* 以及设置调度优先级来控制状态保存线程的调度。状态保存线程的运行流程如图 2 所示。

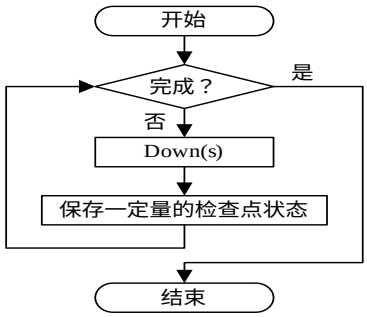

图 2 状态保存线程运行流程

由于信号量 *s* 的初始值为 0，因此状态保存线程首次被调度执行时将进入睡眠状态等待被唤醒。本文修改了 LAM/MPI 中的底层通信接口，当目标进程运行到等待通信操

作完成时，检查该信号量，如果发现状态保存线程处于睡眠状态则将其唤醒。为了确保状态保存线程在被唤醒后能立即被调度执行，设置其调度优先级为高优先级。状态保存线程被唤醒后，在保存一定量的检查点状态后再次进入睡眠状态。重复此过程直至保存所有的检查点状态。

3.2 状态保存粒度的选择

为了使得目标进程等待调度的时间与其通信阻塞时间尽量重合和充分利用目标进程的通信阻塞时间，可以根据目标进程的通信阻塞时间的大小动态决定状态保存的粒度。但不同应用在运行时的通信阻塞时间分布不同，同时为了避免动态决定状态保存粒度带来的复杂性，本文采用了面向应用的静态粒度选择策略。

某个应用在某个特定平台上的运行过程中，其通信阻塞时间的分布大致固定。如果状态保存线程某次被调度时的执行时间T_1大于此时的通信阻塞时间T_2，则只能隐藏T_2/T_1部分的开销。如果状态保存粒度减少，T_1随之减少，被隐藏的开销百分比则增大。当T_1小于等于T_2时，状态保存线程被调度后的执行时间能被完全隐藏。选择状态保存粒度还需要考虑其对检查点延迟的影响。状态保存线程每次被调度时保存的状态越多，其需要调度的次数越少，因此其等待调度的时间越少，检查点延迟也就越小。根据测试结果，本文测试时选择的粒度大小为 256 个页面。

4 性能测试

4.1 测试平台和测试集

本文的测试平台由 2 个节点组成。每个节点装有 2 个主频为 1.6GHz 的 AMD 处理器(Opeteron 242)，2GB 的内存，希捷 SCSI 的硬盘。节点间用千兆以太网相连。

本文的测试集包括 LINPACK 基准测试程序、NPB(Nas Parallel Benchmark)基准测试程序中的 IS 测试程序和中尺度数值模拟系统 MM5。在运行 LINPACK 基准测试程序时，规模大小为 12 000，分块大小为 60。NPB 基准测试程序中的 IS 选择 C 类测试(NPB 中的测试程序根据问题的规模可以分为 A、B 和 C 3 类)。在测试中，每个节点上加载两个进程。设置检查点时，检查点状态保存到本地硬盘。

4.2 性能比较

本节比较了用 BLCR、实现写时复制缓存后的 BLCR(Copy-on-write，COW)以及经过本文进一步优化后的 BLCR(Using-blocking-time，UBT)对本文的测试集设置一次检查点的检查点开销和检查点延迟。因为对不同程序设置检查点时的检查点状态大小不同，所以本文以“s/MB”为单位进行比较。根据测试结果和一些合理的假设参数，本文对开销率也进行了比较。

表 3 检查点开销的比较 **(s/MB)**

	BLCR	COW	UBT
LINPACK	0.019 250	0.008 789	0.004 703
IS(C)	0.021 848	0.005 527	0.003 740
MM5	0.031 085	0.020 293	0.016 160

表 3 和表 4 分别给出了检查点开销和检查点延迟的比较。从表中可以看出，本文的优化方法在写时复制缓存的基础上进一步降低了检查点开销。对 LINPACK、IS(C)和 MM5，检查点开销分别降低了 51.13%、23.24%和 26.05%。除 LINPACK 外，对 IS(C)和 MM5 设置一次检查点操作的检查点延迟并没有显著的增加。对 LINPACK、IS(C)和 MM5，检查点延迟分别增加了 115.74%、34.43%和 19.04%。

表 4 检查点延迟的比较 **(s/MB)**

	BLCR	COW	UBT
LINPACK	0.068 415	0.065 358	0.216 474
IS(C)	0.067 898	0.064 851	0.075 972
MM5	0.091 858	0.071 535	0.073 174

根据测试平台的统计数据，本文假设其平均无故障时间(MTTF)为 3 200h，平均恢复时间为 0.14h。根据文献[1]中的公式，开销率的计算结果如表 5 所示。从表中可以看出，由于降低了设置检查点时的开销，本文的优化方法在写时复制检查点缓存技术的基础上进一步降低了开销率。对 LINPACK、IS(C)和 MM5，开销率分别降低了 25.20%、18.08%和 9.42%。

表 5 开销率的比较

	BLCR	COW	UBT
LINPACK	0.000 983 2	0.000 701 0	0.000 524 4
IS(C)	0.001 221 6	0.000 657 1	0.000 538 3
MM5	0.000 477 5	0.000 394 0	0.000 356 9

5 结论及以后的工作

写时复制检查点缓存是目前最有效的降低检查点开销的方法。而本文的测试结果表明，通过控制状态保存线程的调度和选择合适的状态保存粒度，本优化方法在写时复制检查点缓存的基础上利用并行程序的通信阻塞时间进一步降低了检查点开销。尽管其使得检查点延迟有所增加，但由于检查点开销对开销率的影响更大，计算结果表明本文的优化方法能大幅度降低开销率。

下一步将进行如下 3 方面的工作：

(1)改进粒度选择策略，目前面向应用的静态粒度选择策略并不能充分利用并行程序的通信阻塞时间。

(2)研究如何控制检查点延迟的大小。本文中状态保存线程能否被调度执行完全取决于目标并行程序的运行情况，检查点延迟的大小无法控制。根据文献[1]中的分析，只有控制检查点延迟的大小在一定范围之内时，减少检查点开销的优化才有利于降低开销率。

(3)在更大规模的系统中应用高速通信网络测试本文优化技术的效果。

参考文献

1 Nitin H, Vaidya. Impact of Checkpoint Latency on Overhead Ratio of a Checkpointing Scheme[J]. Information Processing Letters, 1997, 46(8): 942-947.

2 Plank J S. Efficient Checkpointing on MIMD Architectures[D]. Princeton University, 1993.

3 Chandy K M, Lamport L. Distributed Snapshots: Determining Global States of Distributed Systems[J]. ACM Transactions on Computer Systems, 1985, 3(1): 63-75.