ISSN 1000-0054
CN 11-2223/N
清华大学学报（自然科学版）2006 年 第 46 卷 第 1 期
J Tsinghua Univ (Sci & Tech), 2006, Vol. 46, No. 1
18/41
67-69,110

面向集群的消息传递并行程序容错系统

薛瑞尼， 张悠慧， 陈文光， 郑纬民

（清华大学 计算机科学与技术系，北京 100084）

摘 要：为了保证大规模集群系统的可靠性和可用性，设计并实现了一个面向集群消息传递并行程序的容错系统。该系统采用检查点设置与卷回恢复技术，提出了基于内存排除的退出重进入并行环境策略，实现了对用户程序完全透明的容错功能、进程迁移以及系统自动重构。实验结果表明：检查点设置和系统恢复开销小于 10%，符合大规模并行程序容错功能的要求。该系统提高了集群系统的可靠性和可用性，其设计结构和实现方法可以方便地移植到其他消息传递系统。

关键词：容错技术；检查点；卷回恢复；消息传递接口；并行程序

中图分类号：TP 302.8 **文献标识码**：A

文章编号：1000-0054(2006)01-0067-03

Fault tolerance for cluster-oriented MPI parallel applications

XUE Ruini, ZHANG Youhui, CHEN Wenguang, ZHENG Weimin

(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)

Abstract: A fault tolerant run time system was developed for cluster-oriented message passing interface (MPI) parallel applications to guarantee system reliability and availability in high performance clusters. This system uses the checkpointing and rollback recovery technique, with user lever transparent fault tolerance, process migration, and system auto reconfiguration based on an "exit and reenter" parallel environment strategy. Test results suggest that the overhead is less then 10% to satisfy the basic requirements of parallel fault tolerance. The system improves the cluster reliability and availability and its structure and implementation scheme can be conveniently ported to other message passing systems.

Key words: fault tolerance; checkpointing; rollback recovery; message passing interface; parallel application

集群系统在高性能计算领域发展迅猛。1997 年 11 月 Top500 中只有一个集群系统，到 2004 年 7 月，这个数字激增到 291，约占系统总数 58.2%[1]。并行程序是发挥集群系统计算能力最直接的方法，然而事实显示由于各种原因导致的系统失效致使大规模并行程序很难顺利完成，其原因在于并行系统可靠性随着节点数目的增加而下降[2]。可靠性成为影响系统可用性和扩展性的重要因素，容错成为并行系统的必备功能。

消息传递接口（message passing interface，MPI）[3]是并行编程的主流，目前有以下相关的容错系统：BLCR[4]在内核级实现，支持多线程，与 LAM (local area multicomputer)配合可以对简单的 MPI 程序进行检查点以及恢复。MPICH-V[5]以 MPICH 为基础，依赖于通道缓存在不可靠节点上运行。CoCheck[6]则基于其独立的 MPI 实现 tuMPI，通过普通 MPI 消息捎带附加标记消息来触发检查点过程，通过一个独立的资源管理模块完成网络状态的协调和并行程序的恢复。上述系统在恢复时均需要重构并行环境，且不支持进程迁移和系统重构。

本文设计实现的 MPI 并行程序容错系统（checkpoint-based rollback recovery and migration system for MPI applications，ChaRM）以扩展性好、移植性强的 MPICH[7]为基础，实现了低开销用户透明的容错功能，以及进程动态迁移和系统自动重构。ChaRM 支持 IA-32 和 IA-64 平台，可应用于各类集群系统。

1 ChaRM 的结构设计

1.1 系统模型和检查点协议

出错即停故障模型是指，部件失效后立即停止运行，同时其他部件可以探测到此故障的发生[8]。此模型符合集群系统物理特性，可以捕捉各种故障，且便于计算机建模。检查点设置与卷回恢复

收稿日期：2004-12-15
基金项目：国家“八六三”高技术项目（2002AA1Z2103）
作者简介：薛瑞尼(1981-)，男(汉)，山西，硕士研究生。
通讯联系人：郑纬民，教授，E-mail：zwm-dcs@tsinghua.edu.cn

(checkpoint and rollback recovery, CRR)容错机制包括两个方面:检查点过程保存程序的运行状态(程序状态被写入检查点文件),卷回过程则将程序恢复至被保存点的状态。因此,失效程序可以卷回至被保存的状态恢复运行,避免重新启动,实现一定范围内的故障容忍。

文[8]中讨论了各种并行程序检查点设置协议。协调式检查点设置协议是指所有进程在保存各自状态之前,必须同步以清空中途消息,保证整个并行程序处于全局一致性状态。协调式检查点设置协议需要每个进程只维护一个检查点文件,空间开销小,无须垃圾回收。ChaRM 采用此协议。

1.2 层次结构设计

ChaRM 的设计基于两个前提条件:首先,系统构筑在出错即停模型之上;其次,进程通信的基础是可靠的FIFO 信道,这一点由MPICH 的通信语义保证。图1 描述了ChaRM 的层次逻辑结构,虚线框表示每个MPI 任务由通信进程和计算进程组成。

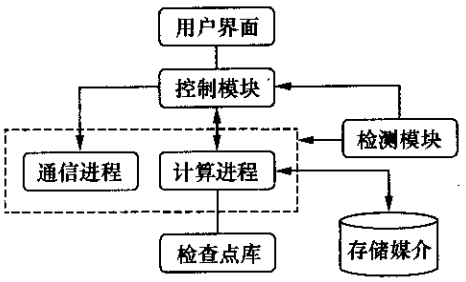


图1 ChaRM 的逻辑结构

管理模块控制所有MPI 任务,负责收集任务信息、发出控制信息。ChaRM 将通信进程作为管理模块和计算进程之间传递消息的中介。管理模块将控制命令广播至通信进程,然后各通信进程分别通知计算进程。计算进程和管理模块间的直接连接是二者交换信息的通道,如注册、注销以及同步等过程。

检查点库为我们自行开发的Thckpt,是ChaRM 中一个独立的模块。Thckpt 只保存计算进程的状态,如同对串行程序进行检查点一样。除了提供基本的CRR API 来完成检查点和恢复任务,Thckpt 还提供了其他4 个接口:pre_checkpoint、post_checkpoint、pre_recover 和post_recover,分别用来在检查点和恢复的开始和结束阶段完成针对计算进程的准备和处理工作。这种设计降低了Thckpt 和计算进程的耦合性,提高了适应不同程序的灵活性。

Thckpt 将检查点文件写入可靠存储媒介,并在卷回的时候读取此文件进行状态恢复。将检查点文件和计算进程置于同一节点的策略会在节点发生故障时无法恢复进程。设置专门的检查点文件服务器可以解决这个问题,但会导致额外的网络传输开销。

故障检测模块是CRR 系统的重要组成部分。智能的检测模块辅助系统决策检查点时机。ChaRM 的检测模块结合了普通的Heart Beat 技术和Intel IPMI 技术。Heart Beat 检测进程失效,IPMI 汇报整个系统状态,预测可能发生的故障。检测模块对收集到的数据进行分析,做出决策并通知管理模块是否进行检查点设置或将可能发生故障节点上的进程迁移至可靠节点。

用户界面图形接口作为ChaRM 的控制前端可以启动MPI 任务,显示各任务的位置和运行状态。它将所有用户命令传递至管理模块,并由管理模块将控制消息广播。

2 ChaRM 的实现

2.1 启动及注册

ChaRM 要求所有MPI 任务启动后进行注册,以便在整个运行过程中进行控制。注册信息包括计算进程MPI 号和进程号、通信进程进程号、监听端口以及节点名称。管理模块通过通信进程端口与通信进程交互,进而控制计算进程。注册过程在计算任务开始之前,并且异步完成。

2.2 设置检查点

ChaRM 收到检查点设置消息后即启动两步协调式检查点设置过程:

1) 管理模块广播检查点设置消息,计算进程接收到此消息后暂停计算,清空通信通道,并设置临时检查点,然后向管理模块发送确认请求;

2) 管理模块接收到所有进程的确认请求后立即广播成功确认消息,否则广播失败反馈消息,计算进程根据收到的消息确定临时检查点是否有效。

为了保证并行程序处于全局一致性状态,在设置检查点之前必须清空通信通道,确保中途消息到达目的任务。各计算进程广播信道清空消息(end of channel, EOC)之后进入EOC 消息接收过程。当计算进程接收到所有来自其他进程的EOC 消息后,就可以设置检查点。由于清空信道是全交换通信方式(如果进程数为 n,消息数目即为 n^2),容易导致死锁。死锁的解决方案见图2,其中a—d 为如下步骤:

a) 源进程0 创建套接字以便目的进程1 连接,然后向目的进程通信进程发出连接请求;

b) 源进程屏蔽请求信号,等待目的进程主动

连接；

c）接收到连接请求后，目的进程的通信进程通过信号通知目的进程；

d）目的进程主动连接源进程。

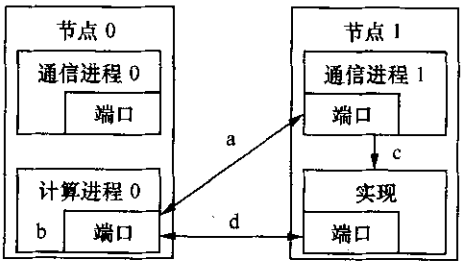


图 2 P4 连接建立过程

如果源进程MPI 号小于目的进程MPI 号，连接过程按照a—d 进行。如果源进程MPI 号大于目的进程MPI 号，源进程在d 释放屏蔽信号，并与目的进程交换角色，重新开始建立连接。这种策略保证了最终的连接都是MPI 号较大的进程主动连接MPI 号较小的进程，避免了死锁的产生。

2.3 卷回恢复

计算进程由通信模块和计算模块两部分组成。计算模块独立于通信模块，所有进程的通信模块共同形成整个并行环境，并在程序启动时自动建立。通常情况下，检查点将计算进程的整个虚拟地址空间完全保存下来，不区分通信和计算模块。因此，卷回过程必须重构并行环境，过程复杂，容易出错。

ChaRM 在设置检查点的时候采用“内存排除”技术将通信模块排除，只保存计算模块，即被保存的计算进程“退出”了并行环境。在卷回恢复时，只有计算模块被恢复，现有的并行环境不受破坏，即计算进程“重进入”并行环境。这种基于内存排除的退出重进入并行环境的策略避免了并行环境重构，降低了系统的复杂性和开销，有利于透明容错。

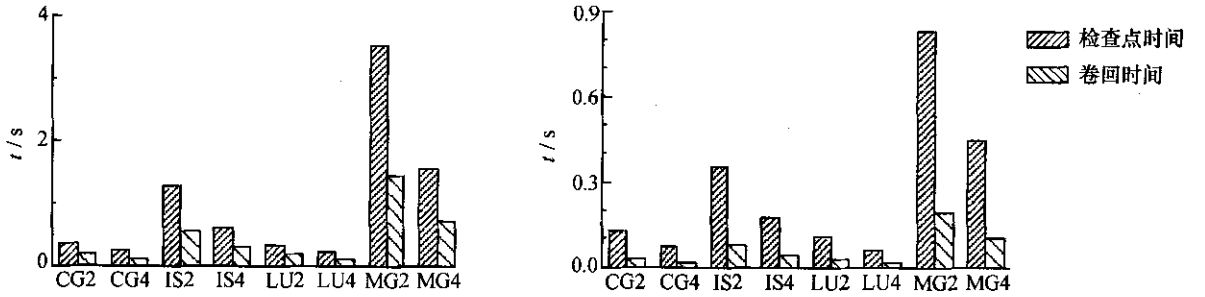
3 性能测试

衡量并行容错系统的性能有三个重要的参量：检查点时间、卷回时间和同步时间，分别表示单纯完成检查点的开销、单纯完成卷回的开销和所有进程同步的开销。我们选择 MPICH-1.2.5.2 和 NAS NPB 并行程序测试集，分别测试了ChaRM 在IA-32 和IA-64 平台下的性能。IA-32 环境为 4×Xeon、700 MHz、1 G SDRAM、Linux-2.4.20、100 MB 以太网；IA-64 环境为 2×Itanium、1 GHz、2 GB SDRAM、Linux-2.4.18、100 MB 以太网。测试结果见表 1 和图 3。

表 1 测试程序的 3 种平均开销相对运行总时间的比例

%

测试程序	开销		
	检查点	卷回恢复	同步
CG	0.84	0.40	0.06
IS	3.88	1.55	0.05
LU	0.45	0.02	0.06
MG	7.38	2.42	0.34



(a) IA-32 (b) IA-64

图 3 NPB 测试程序系统开销

表1 为3 种开销相对运行总时间的比例。图 3 为包括同步时间的绝对开销（CG2 表示测试程序CG 选择标准数据结合 A，并启动两个进程，即 CG.A.2；其他依此类推）。根据以上数据，可以得到如下结论：1）文[9]中指出将并行系统检查点和恢复时间与总运行时间的比例控制在10%以内比较合理。ChaRM 的开销远远低于 10%，满足并行系统的要求；2）进程越多，开销越小。这个现象是由于“进程绝对数目太少”而造成。因为进程绝对数目很少的时候，同步开销远小于检查点和卷回恢复的开销，故而整体开销主要表现为检查点和恢复的损耗，即读写检查点文件的开销。随着进程数目的增加，检查点文件相应变小，导致读写时间变短，所以整体开销下降。但是，随着进程绝对数目进一步增加，同步开销会逐渐增大，可能成为CRR 的性能瓶颈。

（下转第 110 页）

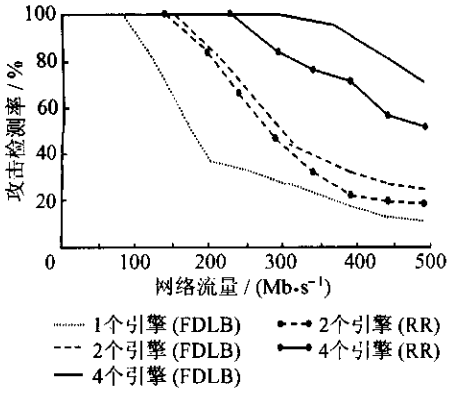


图2 FDLB 算法与轮转算法的效果比较

4 结 论

本文专门探讨了可用于 nIDS 的负载均衡策略和算法,在提出基于多引擎并行处理 nIDS 框架的基础上,分析了 3 种实用的 nIDS 负载均衡策略,重点阐述了一种基于流的动态负载均衡算法——FDLB 算法。实验结果表明,采用多引擎并行处理技术,可以有效克服高负载下 nIDS 出现的丢包问题,显著提高大流量下 nIDS 的检测性能。同时,实验结果也证明了 FDLB 算法是非常有效的,在大流量多引擎的情况下,其负载均衡效果要比简单的轮转算法好得多。

参考文献 (References)

[1] Schaelicke L, Slabach T, Moore B, et al. Characterizing the performance of network intrusion detection sensors [A]. Proceedings of the Sixth International Symposium on Recent Advances in Intrusion Detection (RAID 2003) [C]. Lecture Notes in Computer Science, Springer-Verlag, 2003.

[2] Coit J, Staniford S, McAlerney J. Towards faster string matching for intrusion detection or exceeding the speed of snort [A]. Proc DARPA Information Survivability Conference and Exposition (DISCEX II '02) [C]. Los Alamitos, Calif: IEEE CS Press, 2001. 367-373.

[3] Edwards S. Vulnerabilities of Network Intrusion Detection Systems: Realizing and Overcoming the Risks [Z]. Toplayer Networks, 2002.

[4] Kruegel C, Valeur F, Vigna G, et al. Stateful intrusion detection for high-speed networks [A]. Proceedings of the IEEE Symposium on Security and Privacy [C]. Berkeley, CA: IEEE, 2002. 285-294.

[5] Asser N. Tantawi, Don Towsley. Optimal static load balancing in distributed computer systems [J]. *Journal of the ACM*, 1985, **32**(2): 445-465.

[6] Keith W R, David D Y. Optimal load balancing and scheduling in a distributed computer system [J]. *Journal of the ACM*, 1991, **38**(3): 676-690.

(上接第 69 页)

4 结 论

本文所述的 MPI 并行程序容错系统已经成功地应用于数个中型规模的集群系统,为系统的可靠运行提供了保障。实验结果表明,分层控制的系统结构和基于"内存排除"的"退出重进入"策略减小了系统复杂度,降低了系统开销。本系统的设计方案和实现策略也适合其他消息传递系统,可以方便地进行移植。进一步的研究工作将改进检查点设置协议,降低同步开销。

参考文献 (References)

[1] Top500 Supercomputer Sites. TOP500 List [OL]. http://www.top500.org/, 2004.

[2] David P, Aaron B, Pete B, et al. Recovery-Oriented Computing (ROC): Motivation, Definition, Techniques, and Case Studies [R]. UCB-CSD-02-1175, USA: University of California Berkeley, 2002.

[3] Message Passing Interface Forum. MPI: A message passing interface standard [J]. *International Journal of Supercomputer Applications*, 1994, **8**(3/4): 159-416.

[4] Duell J, Hargrove P, Roman E. The Design and Implementation of Berkeley Lab's Linux Checkpoint/Restart [R]. LBNL-54941, USA: Berkeley Lab, 2003.

[5] Aurelien B, Franck C, Herault H, et al. MPICH-V2: A fault tolerant MPI for volatile nodes based on pessimistic sender based message logging [A]. SC2003. Igniting Innovation [C]. New York: ACM Press and IEEE Computer Society Press, 2003.

[6] Georg S. CoCheck: Checkpointing and process migration for MPI [A]. Proceedings of the 10th International Parallel Processing Symposium (IPPS '96) [C]. Honolulu, HA: IEEE Press, 1996.

[7] William G, Ewing L. A high-performance, portable implementation of the MPI message passing interface standard [J]. *Parallel Computing*, 1996, **22**(6): 789-828.

[8] Elmootazbellah N E, Lorenzo A, Wang Y M, et al. A Survey of Rollback - Recovery Protocols in Message-Passing Systems [R]. CMU-CS-96-181, USA: Carnegie Mellon University, 1996.

[9] Kai L, Jeffrey N, James P. Low-latency, concurrent checkpointing for parallel programs [J]. *IEEE Tran on Parallel and Distributed Systems*, 1994, **5**(8): 874-879.

面向集群的消息传递并行程序容错系统

作者： 薛瑞尼， 张悠慧， 陈文光， 郑纬民， XUE Ruini， ZHANG Youhui， CHEN Wenguang， ZHENG Weimin

作者单位： 清华大学,计算机科学与技术系,北京,100084

刊名： 清华大学学报（自然科学版） ISTIC EI PKU

英文刊名： JOURNAL OF TSINGHUA UNIVERSITY(SCIENCE AND TECHNOLOGY)

年，卷(期)： 2006,46(1)

参考文献(9条)

1. Kai L;Jeffrey N;James P Low-latency,concurrent checkpointing for parallel programs[外文期刊] 1994(08)

2. Elmootazbellah N E;Lorenzo A;Wang Y M A Survey of Rollback-Recovery Protocols in Message-Passing Systems 1996

3. William G;Ewing L A high-performance,portable implementation of the MPI message passing interface standard[外文期刊] 1996(06)

4. Georg S CoCheck:Checkpointing and process migration for MPI 1996

5. Aurelien B;Franck C;Herault H MPICH-V2:A fault tolerant MPI for volatile nodes based on pessimistic sender based message logging 2003

6. Duell J;Hargrove P;Roman E The Design and Implementation of Berkeley Lab's Linux Checkpoint/Restart 2003

7. Message Passing Interface Forum MPI:A message passing interface standard 1994(3/4)

8. David P;Aaron B;Pete B Recovery-Oriented Computing (ROC):Motivation,Definition,Techniques,and Case Studies 2002

9. Top500 Supercomputer Sites.TOP500 List 2004