软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2013,24(Suppl.(2)):89−98
©中国科学院软件研究所版权所有.
E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

超级计算环境容错机制*

赵 毅, 曹宗雁, 朱 鹏, 迟学斌

(中国科学院 计算机网络信息中心 超级计算中心,北京 100190)

通讯作者: 赵毅, E-mail: zhaoyi@sccas.cn, http://www.sccas.cn

摘 要: 中国科学院超级计算环境是整合了包括总中心、分中心和所级中心计算资源的 3 层架构超级计算环境.为提升超级计算环境的可靠性,提供稳定、可靠的计算服务,其容错机制的研究成为超级计算环境的一个研究重点.在对容错基本思想及各类计算机容错技术进行充分调研的基础上,提出一种适用于超级计算环境的容错框架,依据该框架给出了不同层次的容错方案,并对不同层次的容错开销进行了分析和比较,验证了不同层次容错方案对应用程序所带来的影响.

关键词: 超级计算;容错框架;检查点设置/回卷恢复;容错开销

中文引用格式: 赵毅,曹宗雁,朱鹏,迟学斌.超级计算环境容错机制.软件学报,2013,24(Suppl.(2)):89−98. http://www.jos.org.cn/1000-9825/13027.htm

英文引用格式: Zhao Y, Cao ZY, Zhu P, Chi XB. Fault-Tolerant mechanism in supercomputing environment. Ruan Jian Xue Bao/Journal of Software, 2013,24(Suppl.(2)):89−98 (in Chinese). http://www.jos.org.cn/1000-9825/13027.htm

Fault-Tolerant Mechanism in Supercomputing Environment

ZHAO Yi, CAO Zong-Yan, ZHU Peng, CHI Xue-Bin

(Supercomputing Center, Computer Network Information Center, The Chinese Academy of Sciences, Beijing 100190, China)

Corresponding author: ZHAO Yi, E-mail: zhaoyi@sccas.cn, http://www.sccas.cn

Abstract: The three layers supercomputing environment of Chinese Academy of Sciences is built to integrate the computing resources of the head center in Beijing, eight regional centers and several campus-level centers. To enhance the reliability of the supercomputing environment and provide stable and reliable computing services, the fault-tolerant mechanism research has become a research priority of the supercomputing environment. In this paper, the fault-tolerant basic concepts and computer fault-tolerant technologies are introduced at first. Next, a fault-tolerant framework of the supercomputing environment is proposed. Then the fault-tolerant solutions of different levels based on the framework and the performance test results in Deepcomp 7000 are presented. Finally, the fault-tolerant overheads of different levels are compared and analyzed to verify the impact on the application.

Key words: supercomputing; fault-tolerant framework; checkpoint/rollback recovery; fault-tolerant overheads

中国科学院在“十一五”信息化专项中设立了“超级计算环境建设与应用”项目,目标是整合中国科学院的超级计算资源,形成由总中心、分中心及所级中心组成的中国科学院 3 层架构的超级计算环境(supercomputing environment,简称 SCE),如图 1 所示.3 层架构的超级计算环境包括 1 个总中心,8 个分中心和 20 多家所级中心,覆盖全国,接入和登录节点多,用户和作业数目巨大,系统规模庞大,层次结构复杂,这给硬件可靠性和软件可用性带来了很大的威胁和严峻的挑战.因此,如何提升整个超级计算环境的可靠性,使之能够提供稳定、可靠的服务,是 3 层架构超级计算环境的一个研究重点.

超级计算环境的可靠性/容错研究,涵盖面很广,之前的研究工作大部分都针对某一类容错问题进行容错技

* 基金项目: 国家高技术研究发展计划(863)(2011AA01A205); 中国科学院知识创新工程青年人才领域项目(CNIC_QN_10004); 中国科学院青年创新促进会基金

收稿时间: 2012-08-05; 定稿时间: 2013-07-22

© 中国科学院软件研究所 http://www.jos.org.cn

术的研究,而本文从超级计算环境的全局出发,首先介绍容错概念和常用的容错技术,在此基础上提出超级计算环境的容错框架及容错方案,并对不同层次的容错方案进行性能测试和分析.

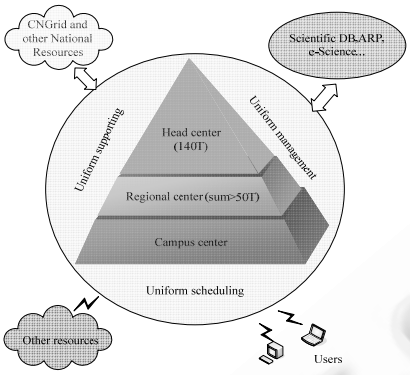

Fig.1 Three layers supercomputing environment diagram of Chinese Academy of Sciences
图 1 中国科学院超级计算环境 3 层结构示意图

1 容错技术概述

1.1 基本概念

一般而言,提高计算机的可靠性有两种方法:一种是避错,就是避免出现故障,第 2 种是容错,要求计算机在出现故障的情况下容忍故障的存在.容错技术最早由约翰•冯•诺依曼(John Von Neumann)提出,是指当系统在出现一个或多个硬件或软件方面故障或错误的情况下仍能保证不间断地提供正常服务的方法和技术[1].

1.2 硬件容错技术

容错技术最早产生于硬件领域,在高性能计算机中通常采取避错、冗余、在线替换等硬件容错技术[2].

- 避错技术.这是指采用正确的设计和质量控制方法尽量避免把故障引进系统即减少器件的失效率.如在高性能计算机中尽量选用低热阻的封装、低功耗的设计和高效的冷却方式等.
- 部件冗余.关键部件及单点故障部位,如电源、风扇、时钟源、监控处理器等都采用冗余设计,冗余方式有 1+1,*N*+1 或 *N*+*M* 等.
- 数据通路冗余.许多计算机中采用了数据通路的冗余措施.如访问系统磁盘阵列 RAID 的冗余路径,访问 I/O 服务器的冗余路径.
- 信息冗余.纠错码和检错码都是信息冗余技术,在计算机中,CPU、Cache、片内 RAM、存储器接口和数据总线等通常采用纠错码进行防护;采用检错码、监视定时器判定是否存在总线超时或网络连接故障.
- 重组技术.重组是指当检测出一个不可恢复的故障后,系统用备用的部件代替有故障的部件,称为后援备份;如果没有备用部件,则隔离故障部件,实现系统降级使用,称为缓慢降级.
- 恢复技术.对于瞬态的可恢复故障,采用恢复技术消除故障的影响,使系统重新运行.如消息重传、指令重试等.
- 在线替换.在高性能计算机中,尤其是刀片式架构中,普遍采用了模块化的冗余结构,实现了背板、互连、电源、网络、风扇等关键部件的冗余和热插拔,从而大大提高了系统的可靠性和可维护性.

1.3 软件容错技术

1.3.1 检查点设置/回卷恢复

检查点设置/回卷恢复(checkpoint and rollback recovery)是指,在程序的执行过程中,定期地或由命令触发把程序的状态(称为检查点)保存到可靠的存储介质上,在发生故障之后,程序重新启动并恢复到前一次保存的程

© 中国科学院软件研究所 http://www.jos.org.cn

序状态,从检查点状态继续执行,把计算损失减小到状态保存时刻到故障发生时刻这段时间所作的计算,避免了程序从头开始执行[3].

随着检查点设置/回卷恢复技术得到了越来越广泛的重视,出现了一系列的检查点库,包括串行检查点库和并行检查点库.串行检查点库主要有 BLCR[4],Condor[5],libckpt[6],CRAK[7]等,并行检查点库主要有 LAM-MPI,MVAPICH2,FT-MPI,Cocheck[8]等.

根据程序状态保存的方式,可以将检查点技术分为系统级检查点(system-level checkpointing)和应用级检查点(application-level checkpointing).系统级检查点是将所有进程的地址空间内容、寄存器信息和通信库状态存储到稳定存储介质上,应用级检查点是通过修改应用程序,程序代码直接指定检查点操作需要保存的变量和数据,产生最小的检查点.

1.3.2 消息日志

回卷恢复还有一种重要的容错技术,即消息日志,是指将消息事件记录下来,在恢复时通过重演记录的消息事件来恢复之前的状态.消息日志可以分为 3 种:悲观日志、乐观日志和因果日志.

悲观日志假定任何非确定事件之后都可能发生故障,需要采用同步的方式把日志信息保存到稳定存储上,虽然在故障恢复时只有故障过程需要回卷,简化了恢复操作,但无错执行时会导致很高的开销.乐观日志是将日志信息以异步的方式保存到稳定存储上,减少了没有失效发生时执行的开销,但失效恢复时回卷恢复较为复杂.因果日志结合了乐观日志和悲观日志两者的优点,通过在乐观和悲观日志中寻找一个平衡点来实现降低无错执行阶段的开销和简单的恢复过程,但这些优势是以复杂的恢复协议为代价的.

1.3.3 冗余计算

冗余计算技术是指,通过同步地对运行中的原始进程做一个或多个复制,当一个进程失效时,通过其冗余副本代替其正常执行的方法[9].同一个进程的主进程通过多播操作直接与所有的副本进程进行通信,副本进程同步主进程的连接状态,在这个过程中,需要使用选举和协商算法以保证主进程和副本进程事务处理及数据传输的一致性.冗余计算技术的优点是 MPI 程序无需任何修改即可运行,整个系统的冗余计算及一致性保证由系统完成,但是每个进程最少需要一个副本,且要保证副本与主进程保持一致性,这些开销是巨大的.

1.3.4 基于算法的容错

对于某些应用还有一种重要的容错方法,即基于算法的容错技术.1984 年,Huang 和 Abraham 提出了基于算法的容错技术(algorithm based fault-tolerant,简称 ABFT),针对矩阵操作利用冗余数据检测和恢复多处理器系统中的计算错误[10].随后,针对不同数值算法设计了相应的 ABFT 算法,如矩阵 LU、QR 分解,快速傅里叶变换算法等.但是,多数 ABFT 关注于错误检测,并未涉及错误恢复;而且 ABFT 主要面向计算错误,难以有效处理超级计算机中出现的各类系统错误;ABFT 需要在原算法的基础上设计复杂的容错方法,需要程序员修改源代码,这些都限制了 ABFT 的应用范围.

1.3.5 MPI 容错技术

MPI(message passing interface)[11]是由全世界工业、科研和政府部门联合建立的一个消息传递编程标准,是当前高性能计算领域最主要的编程环境.MPI-1 和 MPI-2 标准只是对如何写一个正确的 MPI 应用程序进行了规范说明,并没有提供容错的策略.目前 MPI 的容错能力主要由 MPI 的实现者设计和提供.MPI-3 标准考虑到 MPI 程序的容错,对 MPI 标准进行了改进,如允许进程失效及进程失效后 MPI 通信器重建,允许程序降级执行等.

• MPICH2[12]是一个得到广泛采用的高性能 MPI 实现.MPICH2 对并行应用的容错主要体现在其提供的通信器相关的错误处理函数上,需要通过应用程序自己对错误的处理方法来实现并行应用的容错运行.

• FT-MPI[13−15]修改了 MPI 标准的一些接口语义,在应用加载时派生冗余的替代任务和通信器,当系统出现故障时,通过将所有的任务切换到备份的通信器来恢复计算的执行.

• MVAPICH2[16,17]是基于 Infiniband[18]的 MPICH2 实现,它借助伯克利实验室开发的单机检查点软件包 BLCR,实现了 Infiniband 下 MPI 并行程序的检查点设置/回卷恢复的功能.

• LAM-MPI[19]是 Indiana 大学开放系统实验室开发的开源 MPI 版本,它基于伯克利实验室开发的单机检查点软件包 BLCR,实现了对 MPI 并行程序的检查点设置/回卷恢复功能.

© 中国科学院软件研究所 http://www.jos.org.cn

• OpenMPI[20,21]结合多个项目的容错技术思想,对 MPI 并行程序提供容错支持,定义了一个容错框架,集成多种单机检查点协议,实现了一个并行检查点系统.

1.3.6 其他容错技术

除了传统的 MPI 并行编程模型,一些其他分布式并行编程模型也在逐步发展.如美国 UIUC 计算机学院 PPL 实验室基于处理器虚拟化技术的 Charm++和 Google 公司的 Map-Reduce 计算模型都具有很好的容错性.

• Charm++是一个 C++实现的并行运行环境,并实现了一个 MPI 标准的 AMPI(adaptive MPI).其主要特点是处理器虚拟化.用户所看到的是 Chare Object,然后通过 Charm++提供的函数来调用其他 Chare Object 中的方法,Charm++将这些函数调用转化为消息传递,其调度器负责将 Chare Object 调度到真实的处理器上去,并可以在各个处理器之间进行动态迁移,保证负载均衡,同时具有天然的容错性.

• Map-Reduce 模型将并行计算过程抽象为两个函数:Map 和 Reduce,Map 操作读取文件系统对应的文件块,执行 Map 函数里的相关操作得到中间结果,Reduce 操作则将 Map 操作得到的中间结果归约成最终结果.Map-Reduce 可以通过增加计算机来扩充新的计算节点,由此获得海量的计算能力,并具有很强的容错能力.但是 Map-Reduce 模型仅适合任务易于分解且各个任务可独立处理、无须交互的应用.

1.4 网格容错技术

网格环境下出现的错误通常可以分为两类:作业级错误和网格级错误.对于作业级错误常用的容错方法包括:出错重试(retry)、替代资源(alternate resource)、检查点(checkpoint)和作业复制(replication)[22].网格级的容错通常和网格中间件及其他网格软件的实现相关,包括信息服务的可靠性、数据服务的可靠性和作业的调度管理策略等方面的内容.因为与不同网格的具体实现相关,这里不对网格环境的容错作过多探讨.

2 超级计算环境容错框架

超级计算环境建设的一个重要目标是环境的稳定运行,服务不受部分硬件或软件功能失效的影响.我们结合各类容错技术提出了一种适用于超级计算环境的容错框架,指导各层次的容错设计,进而构造出具有高可靠性的超级计算环境.在图 2 所示的超级计算环境容错体系结构框图中,将超级计算环境分为超级计算网格层和计算节点层,计算节点层又分为应用层、系统层、核心层和硬件层,不同层次给出了对应的容错措施.

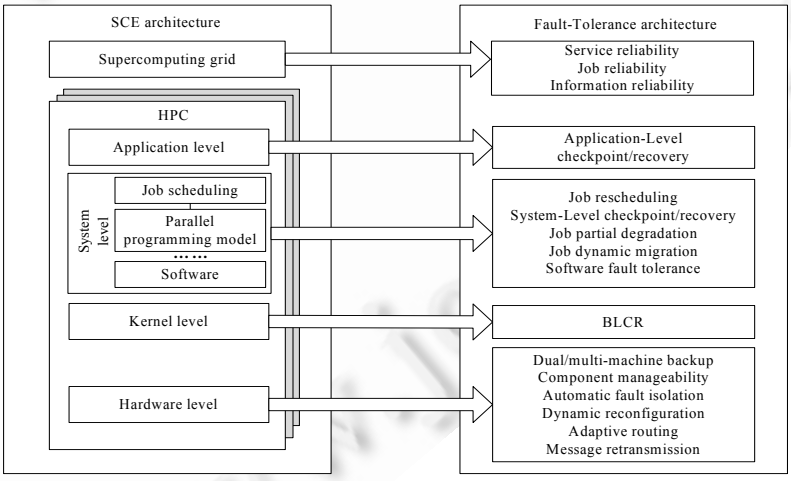

Fig.2 Fault-Tolerance architecture diagram of supercomputing environment

图 2 超级计算环境容错体系结构框图

2.1 超级计算网格层

超级计算网格层的可靠性包括服务的可靠性、信息服务的可靠性和作业的可靠性.服务的可靠性包括超

© 中国科学院软件研究所 http://www.jos.org.cn

级计算环境访问的可靠性和网格服务器的可靠性;信息服务的可靠性主要是解决信息收集和状态更新这两个方面的可靠性问题;作业的可靠性涉及作业的调度机制和迁移策略[23].

2.2 计算节点应用层

用户对自己的课题都有比较深刻的理解,由用户在编写程序时根据需要插入适当的代码,使作业运行过程中可以有选择地进行状态保留,当作业被终止后,一旦执行环境满足需要,还可以从特定位置继续运行,我们称之为应用级容错或应用级保留恢复.应用级容错需要保留的数据少,容错开销小,成功率高.

2.3 计算节点系统层

计算节点系统层对应较多的容错手段,诸如作业重新调度、系统级保留恢复、作业局部降级、作业动态迁移和软件容错集成.作业重新调度也可以说是作业回卷,是作业系统自动完成的一种容错措施,若作业提交时指定了回卷属性,在作业运行过程中因部分资源发生故障退出,作业系统自动地以原有的运行方式避开故障节点重新提交作业;系统级保留恢复是作业系统和并行编程环境共同完成的一种容错措施,作业在运行过程中按预先指定的周期或者事件触发地对作业进行全局保留,形成检查点文件,保留作业运行的阶段性成果,成功保留过的作业,可以继续运行或根据需要(机时不够或者环境故障)停止运行,作业恢复时从最近的检查点文件恢复作业运行;作业局部降级基于某些并行编程模型(如 Charm++),当部分资源发生故障时,作业进行局部降级,甩掉故障节点,重构作业环境,降级完成后,剩余的任务可以继续运行;作业动态迁移是作业运行中当部分节点发生故障时,作业系统自动分配冗余节点代替故障节点,并将本来在故障节点上运行的任务迁移到新分配的冗余节点上运行,完成作业的动态迁移.作业的动态迁移需要作业系统和并行编程模型的共同支持;软件的容错集成是指,对于具有保留恢复功能的软件实现其保留恢复功能与作业系统的集成,使得商业软件用户也可以像其他用户那样在提交作业时指定保留周期,并通过系统命令恢复作业执行.

2.4 计算节点核心层

BLCR 是伯克利实验室开发的内核级检查点库,集成安装在系统内核,实现用户透明的检查点操作.BLCR 本身是单机检查点库,通过与支持 BLCR 的 MPI 库(如 MVAPICH2)的集成来实现并行应用的检查点功能.

2.5 计算节点硬件层

硬件层最常见的容错措施就是双/多机备份,此外,刀片箱、服务器、交换机等部件都具有可管理、故障自动隔离、动态重构等功能,系统网络的自适应路由、消息重传、路由重构等措施保证了网络环境的容错.

3 不同层次容错方案及性能测试

在超级计算网格环境的建设过程中,重点解决了服务的可靠性和网格服务器的可靠性问题,作业的可靠性方面主要采用的重新调度的策略,因此网格层面的容错不对应用程序性能产生直接影响.而计算节点硬件层的容错,主要是在环境建设初期考虑各部件的容错设计,保证系统的可靠性,同样也不会给应用本身带来影响.因此,下文主要对计算节点系统层(包括核心层)、应用层及商业软件的容错方案进行性能测试和分析.

测试环境为深腾 7000 上的刀片节点,每个刀片配置两颗 Intel Xeon E5450 四核处理器,主频 3.0GHz,32GB 内存,配 4x DDR infiniband 网卡;操作系统是 Red Hat Enterprise Linux Server release5.1,内核版本 2.6.18-53.e15;安装 LSF 作业管理系统,mvapich2-1.2p1,采用全局共享的 SNFS 文件系统[24].

3.1 计算节点系统层

MVAPICH2 借助 BLCR 实现了 MPI 应用的检查点设置/回卷恢复功能.深腾 7000 上通过 Platform 公司的 LSF 资源管理软件来进行资源管理和作业调度,用户需要通过 LSF 提交和运行自己的作业,无法直接使用 MVAPICH2 和 BLCR 相关的检查点设置/回卷恢复命令.但是,BLCR 并没有实现与 LSF 的集成,无法告知 LSF 挂起、检查点设置和重启作业,我们通过编写相关的作业加载程序,修改 MVAPICH2 作业加载模式并定义容错的 MPI 类型等,实现了 LSF+MVAPICH2+BLCR 容错方案的集成,在深腾 7000 上实现了对用户完全透明的 MPI

© 中国科学院软件研究所 http://www.jos.org.cn

并行程序的系统级容错.图 3 给出了该方案检查点设置/回卷恢复机制的状态图.

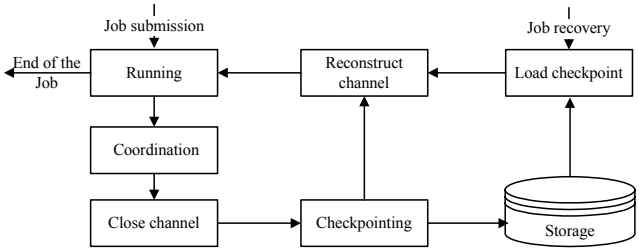

Fig.3 State diagram for checkpoint/rollback recovery
图 3 检查点设置/回卷恢复机制状态图

如图 3 所示,用户通过作业系统提交作业,设定检查点周期,在作业运行过程中,当需要进行检查点设置时,首先要协同所有进程发送检查点设置请求,关闭通信通道,然后调用本地检查点库保存作业进程地址空间的内存页面内容至文件系统,形成检查点文件,然后重建通信通道,作业继续运行.作业恢复的时候,作业进程读取检查点文件,重建通信通道,恢复作业运行.

我们在深腾 7000 上配置了包含 8 个刀片节点的容错队列,安装 BLCR 0.8.1,通过 LSF 提交美国 NAS 开发的 NPB2.4(NAS parallel Benchmark)并行测试程序对系统级容错的时间开销和检查点存储开销进行了测试.

- 检查点文件存储开销

如表 1 所示,检查点文件的大小与问题规模相关,随着问题规模的扩大,检查点文件的大小也会增加,给系统带来的存储开销也会增加.

Table 1 Checkpoint storage overhead
表 1 检查点文件存储开销

Test program	Checkpoint file size (MB+KB)	Checkpoint file size (MB)
lu.C.4	182×4+330	728
sp.C.4	346×4+330	1 384
lu.C.16	53×16+330	848
sp.C.16	101×16+330	1 616
lu.D.16	719.5×16+326	11 512
sp.D.16	1396.6×16+330	22 346

- 检查点设置与恢复时间开销

如表 2 和图 4、图 5 所示.

Table 2 Checkpoint and recovery time overhead
表 2 检查点设置与恢复时间开销

Test program	Program running time (s)	Checkpoint		Recovery	
		Time overhead (s)	Percentage (%)	Time overhead (s)	Percentage (%)
lu.C.4	868	11	1.3	71	8.2
sp.C.4	1 320	19	1.4	72	5.5
lu.C.16	166	15	9.0	11	6.6
sp.C.16	499	43	8.6	13	2.6
lu.D.16	8 018	203	2.5	105	1.3
sp.D.16	10 272	366	3.6	192	1.9

测试结果显示,在问题规模一定的情况下,随着作业规模的扩大,进程协同的开销增大,检查点设置的时间开销也随之增加;作业规模不变的情况下,随着问题规模的扩大,写检查点文件的开销增加,检查点设置的时间开销也明显增加.反之,在检查点恢复的时候,在问题规模一定的情况下,随着作业规模的扩大,读取检查点文件的速度会更快,因此检查点恢复的时间开销反而减少;而作业规模一定的情况下,随着问题规模的扩大,检查点恢复的时间开销会增加.测试中检查点设置和恢复的时间开销都在程序正常运行时间的 10%以内,一定程度上影响到了应用程序的执行效率,但即便如此,仍然还是可以接受的.

© 中国科学院软件研究所 http://www.jos.org.cn

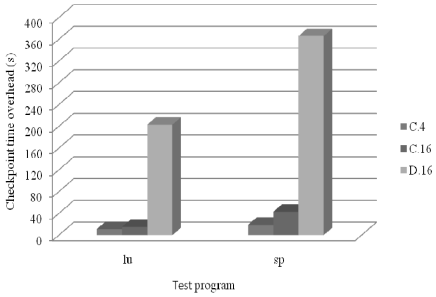

Fig.4 Checkpoint time overhead

图 4 检查点设置时间开销

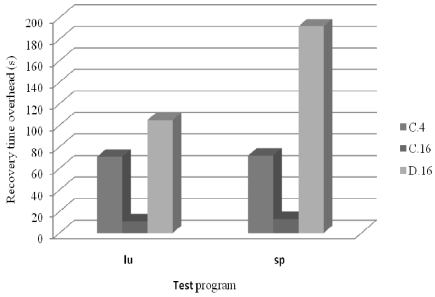

Fig.5 Recovery time overhead

图 5 检查点恢复时间开销

- 不同周期检查点设置时间开销

在作业执行过程中,自动周期性检查点设置的时间间隔越小,作业执行检查点设置的次数越多,所需要的额外时间开销就越大.本文还对不同检查点周期时间设置下,各测试程序的整体运行时间进行了测试,测试结果如图 6 所示,不同检查点周期时间设置情况下,作业检查点设置带来的额外时间开销符合理论值:单次检查点设置时间开销×检查点设置次数.

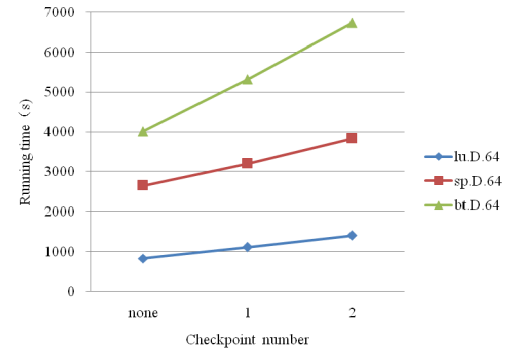

Fig.6 Different cycle checkpoint time overhead

图 6 不同周期检查点设置时间开销

3.2 计算节点应用层

计算节点应用层以在深腾 7000 上完成的小麦基因数据引物扩增比对的并行程序为例来进行测试.该程序实现了作业的应用级容错,使用 MPI 并行编程模型,采用主从通信模式进行任务分配和管理,针对任务分割的情况,对每个引物文件建立一个进度状态文件,文件中存放该引物文件与每个子模板数据文件的匹配计算任务的完成情况.程序每次启动时,主进程读取每个引物文件的进度文件,以确定哪些任务已完成,哪些任务仍待分配到各个从进程处理.程序执行过程中,主进程每收到一个子进程发送的任务完成信息,即立刻更新相应引物文件的进度信息,并写入对应的进度文件.这样,所有任务是否完成的信息都会及时地写到磁盘上,程序在任何时刻被中止后,再次启动时仍然可以获知哪些任务已经完成,而无需重新计算这些任务.这些基于应用层的进度控制功能为长时间大规模的计算提供了良好的容错保障.

在深腾 7000 上使用 x64_blades 队列进行测试,测试分别在 64 核、128 核、256 核、512 核的不同规模下进行,记录计算任务所需时间,并求出相应的处理速度、加速比和并行效率,测试结果见表 3 及图 7 所示[25].测试中还发现,即便进度控制的功能关闭,程序的处理速度仍然基本上不会发生变化,说明应用级容错开销相对于系统级容错开销是微不足道的.

© 中国科学院软件研究所 http://www.jos.org.cn

Table 3 The results of parallel sequence test in Deepcomp 7000
表 3 并行序列比对程序在深腾 7000 上的测试结果

Processor cores	64	128	256	512
Running time(s)	4 105	2 086	1 083	553
Processing speed (num/min)	0.733	1.442	2.777	5.439
Parallel speedup	1.00	1.97	3.79	7.42
Parallel efficiency (%)	100.00	98.39	94.76	92.79

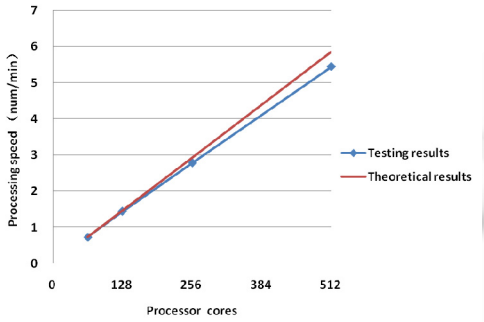

Fig.7 Comparision diagram for parallel sequence test
图 7 并行序列比对程序测试结果对比图

3.3 FLUENT软件

FLUENT 是世界领先的 CFD(computational fluid dynamics)软件,在流体建模中被广泛应用.LSF 作业系统提供对 FLUENT 软件的支持.通过对深腾 7000 上 FLUENT 软件队列的配置和软件提交程序的修改,使用户在提交 FLUENT 作业的时候可以设定自动 checkpoint 的周期,定期保存任务的中间结果,或者通过 bchkpnt 命令随时触发 checkpoint 操作,及通过 brestart 命令恢复作业的执行.FLUENT 的容错解决方案对有保留恢复功能的商业软件具有很好的参考价值.深腾 7000 上 FLUENT 软件容错开销测试结果如表 4 及图 8、图 9 所示.由测试结果可以看出,Fluent 软件检查点设置的时间开销和检查点文件存储开销不大,并且不随作业规模的扩大而增加,检查点恢复开销随着作业规模的扩大而增大主要是因为较大规模的 MPI 作业启动时间较长.

Table 4 Checkpoint and recovery time overhead of FLUNET
表 4 FLUENT 检查点设置与恢复时间开销

Processor cores	Running time (s)	Checkpoint file size (MB)	Checkpoint			Recovery	
			Time interval (s)	Time overhead (s)	Percentage (%)	Time overhead (s)	Percentage (%)
8	7 810	374	600	60	0.77	120	1.54
			300	60	0.77		
16	3 908	374	600	60	1.54	256	6.55
			300	60	1.54		

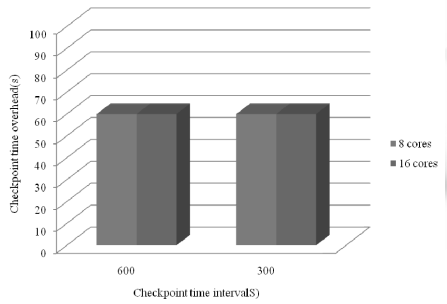

Fig.8 Checkpoint time overhead of FLUENT
图 8 FLUENT 检查点设置时间开销

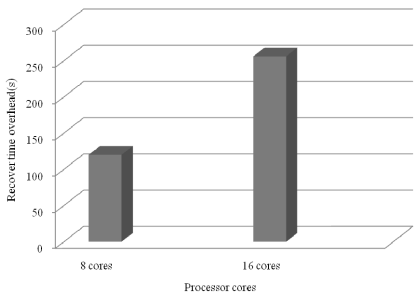

Fig.9 Recovery time overhead of FLUENT
图 9 FLUENT 检查点恢复时间开销

© 中国科学院软件研究所 http://www.jos.org.cn

3.4 容错开销的分析与比较

在系统级容错中,检查点设置的时间开销主要包括写检查点文件的开销和进程协同的开销.写检查点文件的开销受任务进程物理内存消耗和文件系统I/O性能的影响,而进程协同的开销与并行程序的通信特性及规模相关.检查点恢复的时间开销主要是读取检查点文件的开销,与检查点文件的大小和文件系统的 I/O 性能相关.检查点文件的存储开销受应用问题规模和任务进程物理内存消耗的影响.在本文给出的系统级容错测试中,使用的 NPB 测试程序在执行时将所有的数据都读入内存,而有些应用在不同执行阶段操作的数据是变化的,内存需求不同,在不同阶段执行检查点设置操作的存储开销也会不同.

应用级容错的检查点操作是在用户编写应用程序的过程中实现的,检查点的设置和恢复都是应用程序的一部分,而且只需要保存少量必要的信息至检查点文件,因此,应用级容错的检查点开销往往很小.虽然从用户角度出发,在超级计算环境容错框架中将商业软件容错划归系统层,但从容错实现上看,诸如 FLUENT 等软件的容错实际上也是应用级容错.在上述小麦基因数据引物扩增比对程序和 FLUENT 软件容错中,容错开销均不大,并且不随作业规模的扩大而增加,未对应用性能造成影响.

如上分析,系统级容错通常带来较大的系统开销,当并行程序规模扩大时,内存消耗增加,容错带来的系统开销也会随之增大.但是,系统级容错具有完全的用户透明性,无需对用户程序进行任何修改,用户的负担小.应用级容错的容错开销通常较小,但需要用户了解应用程序的结构及功能等,在编写应用程序的过程中实现容错功能,对用户的要求高,完全不具备用户透明性.

4 结论语

本文在高性能计算机常用的容错技术的基础上提出了一种适用于超级计算环境的容错框架,并给出了系统层、应用层和商业软件的容错方案和性能测试,对不同层次的容错开销进行了分析比较,为其他应用容错提供了借鉴和参考依据.通过对超级计算环境容错机制的研究和实践,提高了超级计算环境的可靠性,提升了超级计算服务质量,为下一步超级计算环境建设积累了经验.但是,超级计算环境的容错需要硬件层、操作系统层、系统软件层、用户应用层和网格层等多层次的努力,综合利用各个层次的容错技术,从故障发现到故障的仲裁和处理构建完整、可靠、易用的超级计算环境是一个需要不断探索和研究的问题.

References:

[1] Li HC. Study of computer fault-tolerant technology. Microcontrollers & Embedded Systems, 2010,11:19−21 (in Chinese with English abstract).

[2] Huang YQ, Jin LF, Liu Y. Current situation and trend of reliability technology in high performance computers. Journal of Computer Research and Development, 2010,47(4):589−594 (in Chinese with English abstract).

[3] Wan GW. Research on fault-tolerance technology for message-passing system [MS. Thesis]. Changsha: National University of Defense Technology, 2006 (in Chinese with English abstract).

[4] Berkeley Lab. Checkpoint/Restart Homepage. http://crd.lbl.gov/groups-depts/ftg/projects/current-projects/BLCR/

[5] Tannenbaum T, Wright D, Miller K, Livny M. Condor—A distributed job scheduler. In: Sterling T, ed. Beowulf Cluster Computing with Linux. The MIT Press, 2002.

[6] Plank JS, Beck M, Kingsley G, Li K. Libckpt: Transparent checkpointing under Unix. Technical Report, UT-CS-94-242, University of Tennessee Knoxville, 1994.

[7] Zhong H, Nieh J. CRAK: Linux checkpoint/restart as a kernel module. Technical Report, CUCS-014-01, Department of Computer Science, Columbia University, 2001.

[8] Stellner G. Cocheck: Checkpointing and process migration for MPI. In: Proc. of the 10th Intel. Par. Proc. Symp. 1996.

[9] Niu HB. Research and implementation of MPI parallel fault tolerant technology [MS. Thesis]. Changsha: National University of Defense Technology, 2011 (in Chinese with English abstract).

[10] Du YF. The study and analysis on fault-tolerant parallel algorithm [Ph.D. Thesis]. Changsha: National University of Defense Technology, 2008 (in Chinese with English abstract).

© 中国科学院软件研究所 http://www.jos.org.cn

[11] MPI Standard. http://www.mcs.anl.gov/research/projects/mpi/

[12] MPICH and MPICH2, Argonne. http://www-unix.mcs.anl.gov/mpi/mpich2/

[13] Fagg GE, Dongarra JJ. FT-MPI: Fault tolerant MPI, supporting dynamic applications in a dynamic world. In: Proc. of the EuroPVM-MPI. 2000.

[14] Fagg GE, Gabriel E, Bosilca G, Angskun T, Chen ZZ, Pjesivac-Grbovic J, London K, Dongarra JJ. Extending the MPI specification for process fault tolerance on high performance computing systems. In: Proc. of the ISC. 2004.

[15] Fagg GE, Gabriel E, Chen ZZ, Angskun T, Bosilca G, Pjesivac-Grbovic J, Dongarra JJ. Process fault-tolerance: Semantics, design and application for high performance computing. Int'l Journal of High Performance Computing Applications, 2005,19(4).

[16] MVAPICH and MVAPICH2 Project. http://mvapich.cse.ohio-state.edu

[17] Liu JX, Wu JS, Panda DK. High performance RDMA-based MPI implementation over infiniband. Int'l Journal of Parallel Programming, 2004.

[18] InfiniBand Trade Association. http://www.infinibandta.org

[19] LAM/MPI Parallel Computing. http://www.lam-mpi.org/

[20] Open MPI: Open Source High Performance Computing. http://www.open-mpi.org

[21] Garbriel E, *et al*. Open MPI: Goals, concept, and design of a next generation MPI implementation. In: Proc. of the 11th European PVM/MPI Users' Group Meeting. 2004.

[22] Huang WY. Research on key problems of grid fault-tolerant [MS. Thesis]. Beijing: Tsinghua University, 2008 (in Chinese with English abstract).

[23] Dai ZH. Optimized design and implementation of three layer supercomputing environment [Ph.D. Thesis]. Beijing: Graduate University of Chinese Academy of Sciences, 2011 (in Chinese with English abstract).

[24] Deepcomp 7000. http://www.sccas.cas.cn/yhfw/yjzy/st7000/

[25] Cao ZY, Lang XY, Liu X, Chi XB. Parallelization and optimization of huge scale sequence alignment computation. Journal of Computer Applications, 2011,S2:32−35.

附中文参考文献:

[1] 李洪超.计算机系统的容错技术方法.单片机与嵌入式系统应用,2010,11:19−21.

[2] 黄永勤,金利峰,刘耀.高性能计算机的可靠性技术现状与趋势.计算机研究与发展,2010,47(4):589−594.

[3] 万国伟.消息传递系统容错技术研究[硕士学位论文].长沙:国防科学技术大学,2006.

[9] 牛海波.基于 MPI 的并行容错技术研究与实现[硕士学位论文].长沙:国防科学技术大学,2011.

[10] 杜云飞.容错并行算法的研究与分析[博士学位论文].长沙:国防科学技术大学,2008.

[22] 黄炜元.网格容错关键技术的研究[硕士学位论文].北京:清华大学,2008.

[23] 戴志辉.三层架构超级计算环境优化设计与实现研究[博士学位论文].北京:中国科学院研究生院,2011.

[25] 曹宗雁,郎显宇,刘昕,迟学斌.超大规模序列比对计算的并行优化.计算机应用,2011,S2:32−35.

赵毅(1976－),女,河南洛阳人,高级工程师,主要研究领域为高性能计算机系统管理,容错技术.
E-mail: zhaoyi@sccas.cn

朱鹏(1971－),男,高级工程师,主要研究领域为高性能计算机系统管理,计算数学.
E-mail: zhupeng@sccas.cn

曹宗雁(1981－),男,博士,副研究员,CCF 会员,主要研究领域为高性能计算机系统管理与优化技术.
E-mail: zycao@sccas.cn

迟学斌(1963－),男,博士,研究员,博士生导师,CCF 高级会员,主要研究领域为高性能计算方法与软件,并行计算.
E-mail: chi@sccas.cn

© 中国科学院软件研究所 http://www.jos.org.cn