网格环境中检查点技术的研究与实现

Research and Implementaion of the Checkpoint Technology in Grid Environment

梁 鸿 曾科宏 （中国石油大学(华东)计算机通信与工程学院 257061）

摘要:检查点机制作为一种软件容错机制,将其与网格环境相结合,提高网格计算的服务质量,更好地满足网格系统的要求。本文研究了如何面向网格应用实现检查点设置,使网格环境能够在某个计算结点发生故障后,将相关进程恢复到故障前的检查状态,从该检查点处继续执行,避免重新执行整个任务,节省了大量重复计算时间,实现了容错服务。

关键词:检查点 网格计算 进程恢复 容错 线程

1 引言

现代航天、航空、邮电、石油等应用领域对计算机系统的性能和可靠性提出了越来越高的要求。计算网格是提高计算能力、满足不断增长的应用需求的有效途径。而容错技术是提高计算可靠性的重要保证。计算网格[1]是一个充满吸引力的广域分布式高性能计算平台。计算网格中的计算资源地理上分布、普遍异构,网格结点失效非常频繁。网格结点在发生下列异常事件时会导致本次计算的彻底失败,此前的大量计算不再可用,如:① 运行不同应用程序的其他用户发生的异常事件;② 某机器所有者需要独占 CPU 资源;③ 异常关机;④ 结点(瞬时/间歇/永久)故障;⑤ 系统软件升级,结点更换等维护操作等等。

大规模科学工程计算任务执行时间都较长,一旦某计算结点发生上述异常事件,将导致系统运行失败,程序不得不从头开始执行。为了避免系统在发生上述事件后由于从头开始执行而引起计算上的大量浪费,充分提高网格系统的可用性,在系统正常运行的适当时刻设置检查点(Checkpoint),保存系统当时的规范状态,并对各进程进行相关性跟踪和记录。系统发生故障后,将相关进程回卷(Rollback)到故障前系统一致性状态(检查点),经过状态恢复后从该检查点处重新执行,而不是从程序开始执行,从而节省了大量重复计算时间,充分提高网格应用的可用性。因此利用检查点实现网格环境软件容错是非常必要的。检查点算法是当前计算机界研究的热点和难点,它在许多领域如进程迁移、回卷恢复及 Cactus、DateGrid、Globus 等计算网格中都有重要的作用。

2 网格检查点的发展现状

在网格环境下需要提供检查点服务,这已经成为网格研究人员的共识。在全球网格论坛(GGF)中成立了网格检查点/恢复工作组(GridCPR Workgroup)。在 2003 年 3 月召开的全球网格论坛(GGF)第 7 次会议上网格检查点/恢复工作组讨论了网格检查点/恢复服务的体系结构和 API,分析了网格环境下检查点实现的特殊性,明确了计算网格检查点/恢复服务[2]应该满足如下要求:

(1) 提供在各种计算资源间的互操作能力。在不同平台上进程检查点操作的接口必须互相兼容且通过公共的协议通信。因此计算网格的检查点服务应该提供标准的 API,使用户能编写可以在不同的网格资源间移动的与资源无关的代码来进行检查点和恢复操作。

(2) 保证检查点数据的可用性。检查点数据要保存在独立的、容错的存储设备上,并且能够灵活地适用于不同计算资源的能力和结构。检查点数据管理方法要能够满足应用的需求,并且能够在不修改检查点服务接口的情况下适应基础设施结构的改变。

(3) 提供与相关的网格中间件和基础设施服务互操作的能力。通过检查点服务提供的管理和交互接

口,网格中的调度、记账、安全、作业监测、性能度量等服务能够使用作业检查点的信息。

检查点机制对于长时间运行的大规模科学工程计算任务非常重要,一些网格计算项目在不同程度上提供了检查点机制,但是这些检查点机制通用性不够,而且还存在很多不足。在 Cactus[3] 环境中实现了一种非通用的检查点机制,主要侧重于对模拟参数的保存和恢复。欧盟的 DateGrid[4] 项目侧重于解决数据密集的计算问题。

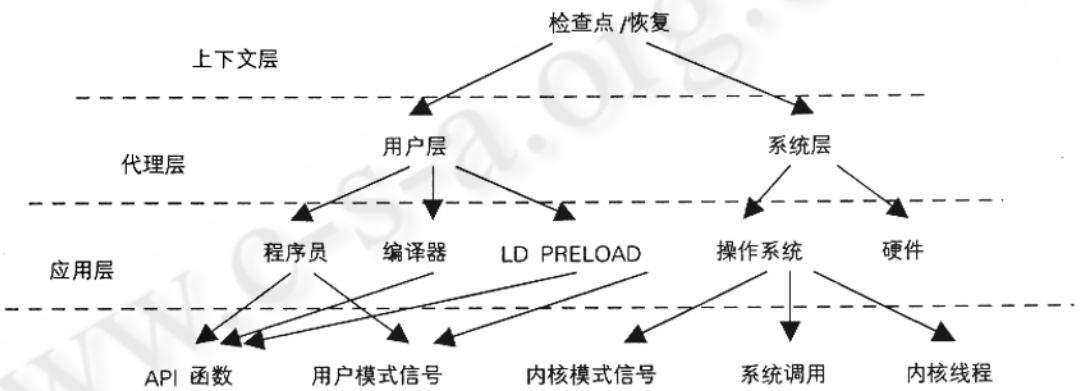

图 1　检查点技术的分类[5]

3　检查点机制

3.1　检查点实现层次

检查点是一种允许进程在正常运行中每隔一定时间间隔保存其状态以使失效后恢复时减小丢失工作量的技术。检查点的作用对象是进程,在某个时间点进程所拥有的各种资源和登记信息在这一时刻的状态。

检查点恢复可以从三个层次加以分类:上下文层、代理层和实现层。在上下文层上,检查点可以分为用户级和系统级两类。图 1 详细描述了各个层上的分类。

用户级实现的常用方式是程序员在代码加入设置检查点的代码或者由编译器在预编译阶段自动插入设置检查点的语句。系统级的实现则分为操作系统及硬件两大类。在操作系统端实现的方式大致分为:内核信号处理、系统调用和系统级线程。

3.2　进程的状态表示

进程的状态是指,为了保证进程在回卷恢复之后能够继续正确地执行所必需的信息。进程状态的保存是进程迁移和回卷恢复的关键。它要包括以下几点:

(1) 进程数据段、用户栈内容。

(2) 与上下文切换有关的项、包括程序计数器 PC、处理机状态字寄存器 PSW、栈指针 SP 等。

(3) 活动文件信息,包括文件描述符、访问方式、文件大小、读写指针等。

(4) 有关信号信息,包括信号量屏蔽码、信号量栈指针、信号量处理函数句柄,以及被挂起的信号量。

(5) 与进程相关的用户文件内容。

3.3　设置检查点

如何来设置检查点过程[6],如图 2。主要分为两个阶段:

(1) 应用程序注册一个线程回调函数,在设置检查点之前这个回调线程被阻塞在内核中,其它线程都处于正常运行状态。当需要对用户进程保存检查点时,就会启用检查点设置机制。首先在/proc 目录下打开一个特殊文件,以检查点的类型、进程编号以及需要保存文件的文件句柄作为参数,调用 ioctl 函数。使用 ioctl 函数来生成检查点文件。在返回用户空间运行线程回调函数后,ioctl 函数放弃对回调线程的阻塞,并且利用周期性的内核函数判断检查点保存进程是否发生了异常中断,或者检查点保存是否超过设定的时间上限,如果超过限制则取消这次检查点保存并清除数据。当所有的线程回调函数进入 cr_checkpoint 阶段后,线程回调函数将重新进入内核态。

(2) 设置检查点的信号将会发给该程序中的每一个线程,并且调用所有的信号回调函数。当程序中所有的线程完成各自的回调函数,进入内核态。当进程

中的线程将自己的状态完全保存下来以后，就进入最后一个阻塞处。所有线程都执行到该处后，保存检查点的工作也就完成了。如果用户设定需要在保存完成检查点后，终止程序，则通过系统调用结束当前进程。如果需要继续运行，则所有的线程回到用户空间，继续执行剩下的任务。

件中读取基本信息，并且恢复各个线程的编号以及它们之间的关系。在通过最后一个阻塞点后，所有的线程离开内核空间，进入用户空间，并且它们的回调函数继续执行进程恢复后的代码。如果需要恢复的进程的编号已经被其它进程占用了，则恢复工作失败。利用检查点恢复进程[6]的流程如图3。

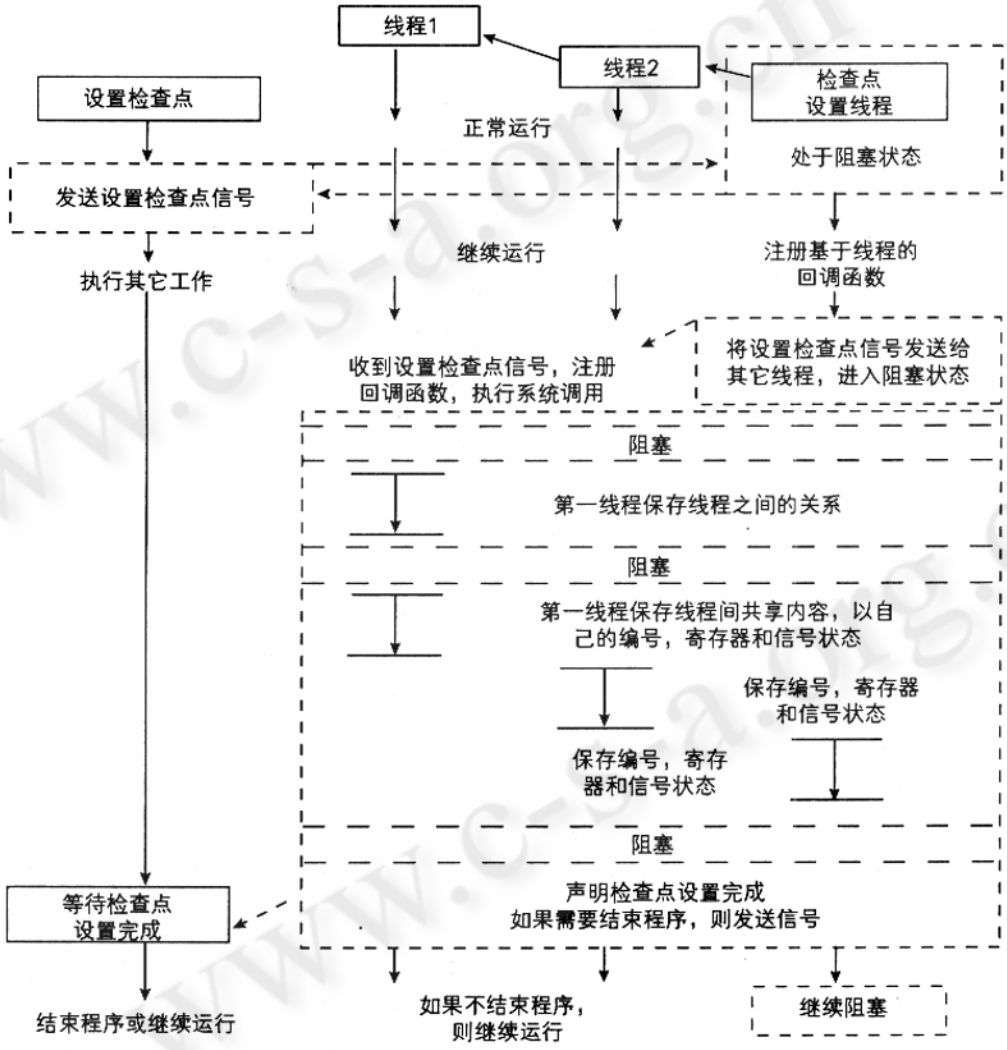

图 2　设置检查点过程

3.4 进程恢复

恢复进程的主要过程就是从检查点文件中恢复内存镜像。恢复机制利用 ioctl 调用，读取检查点文件。创建完子进程后，父进程返回用户空间，一直等到恢复任务完成，结束退出。子进程则利用 clone 函数，产生多个线程重新恢复原任务。其中一个线程从检查点文

4　检查点的性能评价

为了评估检查点用于容错的开销，在实验室搭建了六个结点的网格环境。六个结点均采用操作系统为 Red Hat 9.0，CPU 为 P4 2.4GHz，内存为 512M 的 PC 机。在这个平台下选择三个运行时间长的计算密集型

应用程序来运行，得出在没有设置检查点（容错）情况下的运行时间及设置检查点的运行时间。

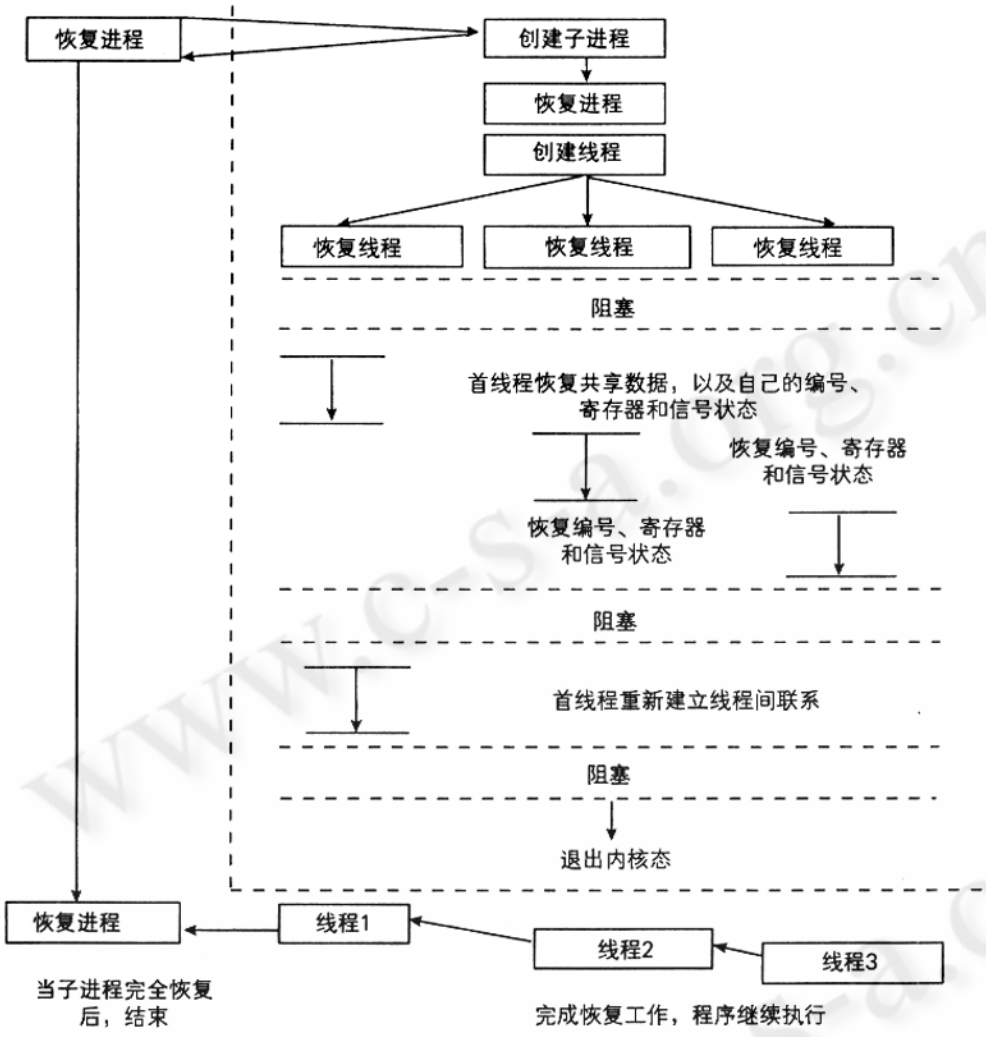

图 3　利用检查点恢复进程

表 1　性能评价

应用程序	运行时间（秒）		储器（KB）
	没有设置检查点	设置检查点	
Guass	301	337	1735
Matmul	679	725	2742
fft	1013	1096	1217

（1）高斯（Gauss）应用程序。它在 1024 * 1024 的矩阵上进行列选主元高斯消去计算，矩阵被划分在几个进程上，在归约的每一次迭代中，拥有主元的进程将主元列发送给所有其他进程。

（2）Matmul 的乘法应用程序。它计算两个 1024 * 1024 方阵的乘积，计算被在被在几个进程之间划分开来，除了对计算进行汇总之外，不需要进程之间的通信。

（3）傅氏变换（fft）应用程序。它计算 32768 点的快速傅里叶变换，每个进程被分配的数据点范围是相同的。

每个应用程序被分布在 6 个主机上；1 个主机执行主进程，其他 5 个主机执行计算进程。

根据测试，表 1 说明这 Gauss，matmul 和 fft 需要较大的数据量来实现检查点和状态恢复机制。这三个计算密集型应用程序在容错管理（设置检查点）的情况下与没有设置检查点的运行时间比较，程序运行时间稍微有所变大。但是相对于这些程序出现异常情况，重新运行程序。设置检查点还是很有效的。利用检查点实现了网格系统的容错机制，提高了应用程序的可靠性。

5　结语

文章以检查点软件容错为出发点，将网格检查点/恢复服务（GridCPR）引入网格环境，为网格平台提供容错功能，提升计算任务的效率。通过设置检查点，根据不同的任务类型以及当前的网格的负载状况，采取不同的任务执行策略，在网格系统中更加灵活的调度当前运行的任务，提高使用效率。另外，利用检查点机制提高了系统计算任务的可靠性，为网格计算的进一步发展提供可靠的性能保证。因此研究检查点技术在网格环境中的应用具有重要的实用价值。

（下转第 53 页）

(上接第 49 页)

参考文献

1 Foster I. The Grid: A New Infrastructure for 21 Century Science Physics Today, 2002, 54(2).

2 GridCPR Working Group. An Architecture for Grid Checkpoint Recovery Services and a GridCPR API. http://www. gridforum. org/meetings/ggf7/drafts/GridCPR001. doc.

3 G. Allen T. Dramlitsch, I. Foster, N. Karonis, M. Rpieanu, I. Seidel, and B. Toonen. Supporting Efficient Exectution in Heterogeneous Distributed Computing Environments with Cactus and Globus. In Proceedings of the ACM/IEEE SC2001 Conference, November 10 – 16, 2001 Denver, Colorado.

4 DataGrid Project. WPI Document, Job description language. Howto. Version 0.1 September 2001. http://server11. infn. it/workload – grid/docs/Datagrid – 01 – TEN – 0102 – 02 – Document. pdf

5 Jos' e Carlos Sancho, Fabrizio Petrini, Kei Davis, Roberto Gioiosa, Song Jiang, "Current Pracitice and a Direction Forward in Checkpoint/Restart Implementations for Fault Tolerance," Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium (IPDPS'05).

6 J. Duell, P. argrove, and E. Roman, "The Design and Implementation of Berkeley Lab's Linux Checkpoint/Restart," 2002 10 – 13.

7 张琳、杨静，网格检查点恢复服务及其应用编程接口分析，计算机应用，Vol 24, NO. 7,, 2004. 7.

8 李睿江、肖侬、杨学军，基于作业进展描述的计算网格作业检查点，计算机工程，Vol. 31 NO. 10, 2005. 5.