并行调试中的若干关键问题*)

王 巍 方滨兴 张宏莉
(哈尔滨工业大学计算机学院 哈尔滨150001)

Several Key Problems in Parallel Debugging

WANG Wei FANG Bin-Xing ZHANG Hong-Li
(School of Computer Science, Harbin Institute of Technology, Haerbin 150001)

Abstract With the development of technique of parallel computing, parallel debugging becomes more and more important. Some new products and technology appear. In this paper, we introduce some key problems concerned in parallel debugging and some methods to solve them.

Keywords Parallel computing, Parallel debugging

1. 引言

程序的调试技术对于计算来说是很重要的,因为抛开性能不谈,程序的正确性是编程者追求的最终目标。然而随着科学的发展,需要解决的问题的规模越来越大,传统的串行计算模式已经难以满足需要,于是并行计算模式被提出。那么调试这一必需的功能也要被放到并行计算环境中重新考虑,从而提出了并行调试的概念。

由于并行计算本身所具有的特点,使得并行调试面临着很多挑战。其一,并行计算机的计算模式主要分为共享存储和分布存储两大类,因此需要针对每种机器的特点设计不同的并行调试器;其二,并行程序是在多个结点机上运行,即使在单结点机上也是以多进程或多线程的方式运行,这种计算模式突破了传统的串行计算概念,那么对于串行程序调试中的一些技术和手段将不再适用于并行调试。以上原因都要求并行调试器从设计到实现都要引入新的方法,才能更好地解决并行程序的调试问题。完善并行调试需要多方技术人员的努力,因此在1997年成立了高性能调试论坛(High Performance Debugging Forum, HPDF),为并行调试器的设计提出了标准,研讨并行调试中的问题。

本文将具体介绍并行调试中面临的几个关键问题及其已有的解决方案。

2. 并行调试技术简介

并行调试技术总是在具体的调试器中应用的,因此本文以下讨论的都是基于具体的并行调试器进行的。

一般意义上的调试器是一个为用户提供观察和控制执行中的程序,即目标程序功能的工具,一个并行调试器则是针对并行程序完成相应的功能[1],其在并行计算系统中所处的位置可由图1描述。然而并行程序的多机、多进程的运行方式为并行调试器完成上述功能造成很多困难。

通常的并行调试技术可分为事后(post-mortem)和运行时(on-the-fly)两类。事后分析主要采用记录/重放技术,而运行时调试则分为基于状态的调试和基于事件的调试。

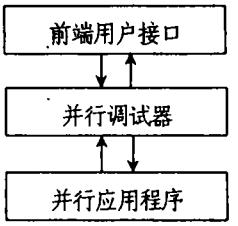

图1 并行调试器所在的层次

记录/重放技术的策略是先将程序的执行轨迹记录下来,在调试期对记录的数据在相同运行环境中进行重放,从中发现程序中潜在的错误。该技术在分布式多机系统上实现较多,因为重放要求对所有的同步通信进行记录,而分布式多机系统的计算和同步是分开的,需要记录的信息较少;而在共享存储的多处理机系统中,计算和通信都表现为对共享变量的访问,需要记录的信息量太大。目前有很多并行调试器采用记录/重放技术,如DCDB[2]、XPVM[3]等。

基于状态的调试是传统的串行调试器的主要策略,因为串行程序是以单进程方式运行的,只有一个状态空间。将这个概念扩展到并行程序时,就会出现问题,每个进程都有自己的状态,那么从宏观上来看,整个并行程序是一个若干异步运行着的进程的集合。为了进行状态调试,则必须引入全局一致状态的概念,这就将调试问题转移为如何识别运行中的全局一致状态的问题。Etnus公司的TotalView[4]就是基于状态调试的。

基于事件的调试则需要建立建模语言,用户通过建模语言定义需要监测的事件。前面介绍了并行程序中各个进程间是异步的,那么如何确定事件间的发生顺序是该调试策略的关键。Lamport提出的happened Before关系[5]定义了事件间的先后顺序,该关系为基于事件的调试打下了理论基础。

从以上介绍可以看出,基于状态的调试可以达到很细的粒度,但是比较复杂,需要识别全局一致状态;而基于事件的调试策略比较灵活,允许用户参与,但是粒度较粗。Uoregon大学提出了将程序的状态和事件结合到一起的并行调试器

*)本文得到航天创新基金资助。王 巍 博士研究生,研究方向:并行与分布式计算。方滨兴 博士生导师,研究方向:并行计算技术、Web计算。张宏莉 博士,研究方向:并行计算、群机计算。

Ariadne[6]，将两者的优势结合起来。

以上介绍了主要的并行调试技术，与并行调试相关的还有调试器的体系结构、前端界面显示问题，这里所涉及到的一些具体问题将在下一节中详细讨论。

3. 并行调试中的几个关键问题

上一节简要地介绍了并行调试中的一些概念和主要技术，本节将对其中的一些关键问题进行讨论。

3.1 不确定性

并行程序中的各个进程是异步执行的，尤其是分布式多机系统中，消息传递的次序在每一次运行时可能都不同，这就使得程序的运行结果不一致，为程序的调试带来困难，这就是并行程序的运行不确定性，简称为不确定性。

导致不确定性的一个根本原因是竞争的存在。竞争对于共享存储和分布式存储来讲，其含义不同。对于共享存储，对于共享变量的访问冲突成为竞争；而对分布式存储，一个进程接收其它进程发来的消息的次序发生变化，造成程序执行结果不确定，这种现象称为竞争，图2给出了分布式系统中的竞争的一个例子。

图2表示一个分布式系统中运行着3个进程 p1，p2和 p3，进程 p1和 p3向 p2发送消息 m_1 和 m_2，p2在接收消息完毕后继续计算。在图2中实箭头表示消息 m_2 比 m_1 先到，如果实际运行中出现了如图2虚箭头的情况，即 m_1 先到，并且 p3的后续计算与两个消息传送的数据有关的话，那么必然就会产生运行结果的不确定性。因此，为了消除并行程序的不确定性，就要检测并消除竞争。

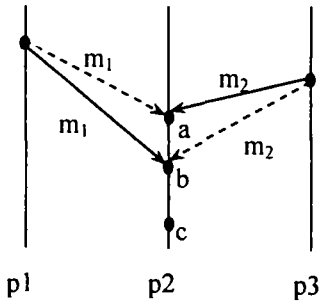

图2 分布式系统中的竞争

文[7]给出了解决共享存储中的竞争问题的一个方案。其基本思想是检测到运行中的第一次竞争(the)，因为如果将第一次竞争消除掉，就有可能将其后的竞争也随着消除，这样可以大大节省时间、减小开销。它提出了一种两阶段方法，首先是收集可能产生竞争的访问集合，然后对该集合作二次操作，从而发现造成第一次竞争的访问。这两次操作都是以运行时方式进行的。

文[8]针对分布式系统提出了竞争检测机制。该方法是基于 Lamport 提出的事件间的 Happen Before 关系，将竞争分为人工和非人工两类，它主要解决非人工竞争的检测问题。该方法同样也是分为两个阶段，第一步是判断运行中是否存在竞争，若存在，则进行第二步，重新执行程序并对其进行跟踪，从而发现竞争的具体位置。第一阶段是运行时方式完成的，而第二阶段则是进行事后(post-mortem)分析。采取该策略的原因是纯粹的运行时分析需要监测整个程序的运行，对于长时间运行的程序是不适合的；而纯粹的事后分析则需要记录大量的事件(消息的收发)，造成过大的空间开销。该策略是两者的一个折衷。

这里需要说明的是，事后分析方法对共享存储系统是不适合的，因为共享存储系统的计算和通信是分不开的。这样就需要记录的信息量极大，是系统承担不起的。

3.2 可移植性

早期的并行计算机都是多处理器的大型机，无论是共享存储还是分布存储，所有结点都是同构的，在这种环境下运行的并行程序比较易于调试，只要关心程序本身，而无需关心其运行环境。但是随着 NOW 技术的发展，目前群机计算逐渐成为主流，而构成群机的结点可以是同构的，也可以是异构的。另外就是 Web 计算概念的提出，由互联网上的若干台机器构成的分布式计算环境则更为复杂。以上两点对并行调试提出了新的挑战。

为了解决对异构环境下的并行调试，就要提高并行调试器的可移植性，摆脱其对具体机器的依赖。NASA 开发的 p2d2并行调试器[9]对可移植性进行了研究，主要的思想是将用户接口与后端运行环境进行隔离，从而实现可移植性，p2d2的总体结构如图3所示。

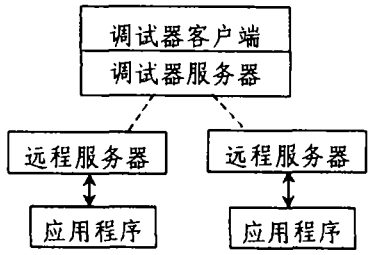

图3 并行调试器 p2d2的总体结构

由图3可以看出 p2d2是按照 client-server 形式构建的。在本地的调试服务器中定义了与调试会话有关的 C++对象的集合，如进程和堆栈等。与目标程序的分布式特性以及用户界面相关的部分都由客户端来实现。本地相关的部分都由客户端来实现。本地服务器将机器相关的代码与客户端进行了隔离，即对远端程序的调试对用户来说是不可见的，从而使得客户端可以以很高的移植性的方式进行实现。然而 p2d2也存在着问题，远端服务器仍然依赖于具体目标机器的类型。因此，p2d2选择了 gdb 作为目标机器的调试服务器，因为 gdb 的可移植性比较好，而且在绝大多数类 unix 的系统上都得到了实现。

清华大学在开发并行调试器 BUSTER(Debugger for Cluster)[10]的过程中也考虑了可移植性问题。BUSTER 同样采取了层次式的结构，将与系统相关的特性隐藏于 BUSTER 的底层。该调试器对移植性的支持主要体现在3个方面：(1)将调试命令控制与底层调试分割开来，与 p2d2的思想相近；(2)对调试系统进行了抽象，把对具体系统(包括硬件和软件)的操作进行了模块封装，并提供了接口。这样在移植时只需替换相应的模块即可。(3)充分利用了现有的标准的系统调用或库函数，从而使得实现标准化，能够得到更多底层系统的支持。与该问题相关的研究还有 IDCD(IBM's Distributed Computing Debugger)[11]等。

3.3 运行状态显示

调试器的基本功能就是让用户及时了解程序的运行状

(下转第167页)

验中，热文件和其他文件分布与图6、7相同，同时有客户A和客户B，A、B以每秒不同的次数分别访问热文件和其他文件，试验结果如图所示。从图中看到，随着访问频率的增加，SCAN方式的访问延迟迅速增大，而SCAN+ABL的访问延迟却增加较慢，表明ABL机制可以在文件访问频繁的时候显著地提高文件访问性能。

结束语 本文提出了一种新型高性能磁盘布局机制ABL。ABL可以统计磁盘上的文件的访问频度，并根据访问频度将频繁访问的磁盘文件进行复制。复制的文件将放置在磁盘的"降温区"里面，系统可以根据文件的访问频度自动地确定文件的复制位置和次数，使少数高频率访问的文件在磁盘上合理地备份，这样可以显著地提高磁盘的性能。在进行热文件降温复制时，本文利用磁盘的潜在带宽来完成磁盘块的复制。潜在带宽充分利用了现代磁盘的"Zero-Latency Access"特性，可以在不增加系统文件系统延迟的情况下，实现热文件的复制和同步，从而提高磁盘系统的性能。试验表明，ABL可以有效地降低热文件的访问延迟，显著地提高了文件系统的性能。

参考文献

1 Chen P M, Lee E K, Gibson G A, Katz P H, Patterson D A. RAID: High-Performance, Reliable Secondary Storage. ACM Computing Surveys, 1994, 26(2): 145～185

2 Gibson G. Redundant Disk Arrays: Reliable, Parallel Secondary Storage. MIT Press, 1992

3 Yu Xiang, Gum B, et al. An-derson. Trading capacity for performance in a disk array. Symposium on Operating Systems Design and Implementation (San Diego, CA, 23-25 October 2000), USENIX Association, 2000. 243～258

4 Ruemmler C, Wilkes J. An introduction to disk drive modeling. IEEE Computer, 1994, 27(3): 17～28

5 Jiri S, John L, Lumb C P, et al. Track-Aligned Extents: Matching Access Patterns to Disk Drive Characteristics. In: Proc. of the FAST '02 Conf. on File and Storage Technologies, Monterey, California, Jan. 2002. 259～274

6 Lumb C, Schindler J, Ganger G R, Riedel E, et al. D. F. Towards Higher Disk Head Utilization: Extracting Free Bandwidth from Busy Disk Drives. In: Proc. of the Fourth Symposium on Operating Systems Design and Implementation, San Diego, CA, Oct. 2000

7 Baker M, Hartman J, Kupfer K, Shirriff K, Ousterhout J. Measurements of a Distributed File System. ACMSymposiumon Operating SystemsPrinciples, 1991. 198～212

8 Ruemmler, Chris, Wilkes J. UNIX disk access patterns. Usenix Conference Proceedings, Jan. 1993. 405～420

9 Ganger G R, Kaashoek M F. Embedded inodes and explicit grouping: exploiting disk bandwidth for small files. Annual USENIX Technical Conference (Anaheim, CA), Jan. 1997. 1～17

(上接第133页)

态，从而发现其中的问题。为了实现这一点，调试器就要将运行状态展现给用户。对于串行程序来讲，这是很容易的，只需提供单独一个进程的信息就可以描述整个程序的运行状态。但是一个并行程序中包含了若干个同时运行的进程，这就给如何显示运行进程的信息带来了问题。如果要将所有的进程运行信息都显示出来，必然造成前端的信息过于拥挤，而且可扩展性也不好；如果提供的信息过少，也无法全面地描述并行程序的运行状态。这也是一个需要折衷的问题。

高性能调试(HPD)标准提出了输出聚集(output aggregation)的概念，来缩减输出信息量以及重复的显示。为了用户方便，当跨多进程的输出内容相同时，调试器负责将这些内容合并，只显示其中的一个副本。举一个例子，假设一个100个元素的数组中刚刚被初始化，并复制到多个进程中，用户此时想查看其中的内容，即使进程数量不太大，那么要对每个进程单独进行显示，这个量也是很大的，而且很难判断数组中的值是否已经发生了变化。聚集化的输出提供了统计所含内容相同的进程功能。如果所有进程中的数组的内容都相同，则只显示其中的一个副本；若其中有不同的，则将其单独显示。采用这种方法可以大大压缩需要显示的信息量。

p2d2在调试器主界面中设计了进程网格(process grid)来表示计算中所有的进程[12]，从而可以在相对小的空间内显示比较多的进程信息。它是用二维网格中每一个单元表示一个进程，每个单元可以根据进程的不同状态加以不同的表示。如可以用绿色表示运行中的进程；红色表示停止运行的进程。从宏观上看，整个计算中进程的状态可以浓缩到这个二维网格内。

为了能够使用户可以细致地看到进程，该网格提供了两个选择器。一个是进程选择器，用来选定单个进程；另一个是组选择器，可以选择某一个进程组。这两个选择器都是通过操作两个特殊的网格元素来实现的。

小结 随着并行计算技术的发展，并行调试成为该领域的一个主要研究方向，有很多问题有待解决，很多国内外的商业和研究机构致力于此。本文介绍了并行调试中的几个关键问题，并给出了目前已经存在的解决方案。相信随着并行计算技术在各个领域的不断推广，必将使并行调试技术更为成熟和完善。

参考文献

1 http://www.ptools.org/hpd

2 Wang Feng, et al. Completely Debugging Indeterminate MPI/PVM Programs. Journal of Software. 2001, 12(3)

3 http://www.netlib.org/pvm3/xpvm

4 http://www.etnus.com/TotalView

5 Lamport L. Time, clocks, and the ordering of events in a distributed system. CACM, 1978, 21(7): 558～565

6 Shende S, Cuny J. Event and state-based debugging in TAU--a prototype. In: Proc. of SPDT'96: SIGMETRICS Symposium on Parallel and Distributed Tools, May 1996. 21～30

7 Park Hee-Dong, Jun Yong-Kee. Detecting the first races in parallel programs with ordered synchronization Parallel and Distributed Systems. In: Proc. 1998 Intl. Conf. 1998. 201～208

8 Netzer T W, et al. Debugging race conditions in message-passing programs. In: Proc. of the SIGMETRICS symposium on parallel and distributed tools, 1996. 31～40

9 Hood R, Jost G. Heterogeneous Computing. A debugger for computational grid applications Workshop, 2000. (HCW 2000) Proceedings. 9th, 2000. 262～270

10 杜术，王东升，余宏亮，郑纬民. 并行调试器BUSTER可移植性的设计与实现. 高技术通讯增刊，2000. 8

11 Meier M S, et al. Experiences with building distributed debuggers. In: Proc. of the SIGMETRICS symposium on parallel and distributed tools, 1996. 70～79

12 Hood R. The p2d2 project: building a portable distributed debugger. In: Proc. of the SIGMETRICS symposium on parallel and distributed tools, 1996. 127～136