Active Block Layout：一种高性能磁盘布局机制*)

卢 军 卢显良 罗光春 韩 宏 魏青松
（电子科技大学计算机学院 成都 610054）

Active Block Layout: A High Performance Block Layout Mechanism

LU Jun LU Xian-Liang LUO Guang-Chun HAN Hong WEI Qing-Song
(Department of Computer Science of UEST of China, Chengdu 610054, China)

Abstract The access frequency of different files in file system is dissimilar. If file system can optimize the block layout of these hot files which are frequently accessed, the performance of file system will be improved. This paper presents a high performance block layout mechanism: Active Block Layout (ABL). ABL can record the access frequency of every file in file system and actively optimize the block layout of these hot files by block duplicating. The duplicated blocks can be placed in the special zone of track, which is called "Cooling Zone". ABL can automatically determine the placing position and the copy count of the blocks which need to be duplicated. In order to reduce the overhead of block duplication, this paper also presents a mechanism which uses the potential disk bandwidth to realize the block duplication, and does not obviously degrade the performance of file system.

Keywords Disk, Performance, Layout, Active, Bandwidth

1. 引言

随着计算机技术的发展，CPU 的运算速度越来越快，但磁盘存储系统速度的提高却远远跟不上 CPU 速度的提高。这种差别导致在许多计算机系统中，存储系统成为系统性能的瓶颈。磁盘的存储容量和存取速度的发展是不平衡的。磁盘存储容量以每年一倍的速度发展，而磁盘的存取时间仅仅每年提高10%左右[1]。为了提高计算机系统的性能，出现了许多技术来提高磁盘系统的性能。一种重要的技术就是使用增加磁盘的数量来换取更高的性能，例如数据条列化、数据镜像和 RAID 等等[1,2]。

除了使用增加磁盘数量的方式来提高存储系统性能以外，也可以在单个磁盘上通过冗余存储来提高性能[3]。本文将提出一种新型的高性能磁盘块布局机制：Active Block Layout（ABL）。ABL 不是使用增加磁盘数量的方法来提高磁盘的性能，而是利用磁盘上的空间来冗余存储数据以提高数据的访问速度。

本文的工作主要在于：(1)提出了一种新型的数据块布局机制。使用这种机制，可以使磁盘上访问频繁的磁盘块进行自动复制并优化分布，从而提高对这些磁盘块的访问性能，从而提高存储系统的性能。(2)提出使用磁盘的潜在带宽来进行数据复制和同步。使用潜在带宽可以在不明显降低磁盘响应时间的情况下，高效地进行磁盘块复制和同步。

2. 磁盘的潜在带宽

一次磁盘访问时间 T_{access} 由三个部分组成[4]：$T_{access}=T_{seek}+T_{rotate}+T_{transfer}$。$T_{seek}$ 代表磁头寻道的时间，T_{rotate} 代表磁头旋转定位的时间，$T_{transfer}$ 代表磁头实际传输数据的时间。在这三部分时间中，只有 $T_{transfer}$ 是真正从磁盘传输数据的有效时间。但是在 T_{access} 的三个组成部分中，T_{seek} 和 T_{rotate} 通常占据大部分时间，其中 T_{seek} 通常占据80%以上的时间[4]。因此，提高磁盘性能的两种途径是：(1)减少 T_{seek}，例如各种磁头调度算法。(2)通过优化数据块的放置来提高 $T_{transfer}$ 在整个磁盘访问中所占的份额。

现代磁盘除了上述特性以外，还具有一种新的特性："Zero Latency Access(ZLA)"[5]。通常磁盘在访问第 T 号磁道上的第 M 号到 N 号连续扇区的时候，首先必须将磁头定位到 T 磁道，然后等待磁头旋转到 M 扇区，再按照 Logical Block Number(LBN)顺序依次读出每一扇区的数据。在 ZLA 方式下，磁头寻道到 T 磁道后，可以马上以任何顺序将 M-N 间的扇区读入磁盘驱动器的内部 Cache 中，然后再将 Cache 中的扇区数据按照 M 到 N 的顺序重新排序组装，返回给用户。如果磁盘驱动器需要读取一个磁道的所有扇区，那么 ZLA 方式可以在磁盘寻道结束后，马上开始读取一个磁道的所有扇区，而不需要任何的旋转延迟。当所有的扇区都读入到磁盘驱动器的内部 Cache 后，磁盘驱动器的硬件将这些扇区的数据重新按照 LBN 的顺序组装，然后返回给用户。同样的方法，也可以使用在写操作中。写操作和读操作不同的是，使用 ZLA 写，必须将全部的写扇区的数据从主机拷贝到磁盘驱动器的 Cache 中之后，数据才能开始写入磁盘。

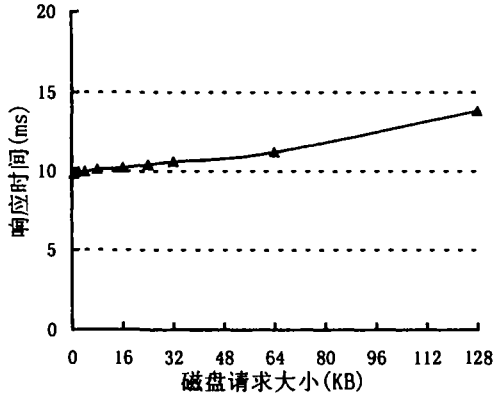

图1 利用 ZLA 加入数据后的写延迟情况

*)本文受国家95重点攻关项目支持。卢 军 博士研究生。卢显良 教授，博士生导师。罗光春 博士研究生。韩 宏 博士研究生。魏青松 博士研究生。

当磁盘驱动器的磁头在一个磁道进行读写的时候，磁头通常需要进行旋转定位，在这段时间内，可以利用 ZLA 技术，将附加的读写信息添加到读写队列中。而磁盘驱动器可以利用 ZLA 技术将附加的读写扇区插入到原来读写操作的队列中，对原来的读写不会造成明显的影响。图1是一个对 ZLA 方式性能的测试试验，在测试中，对一个磁道读取不同数量的扇区，来测试 ZLA 的性能。从图中可以看到，磁盘请求大小从8kB 增加到64kB 后，访问的响应时间仅仅增加10%左右[6]。

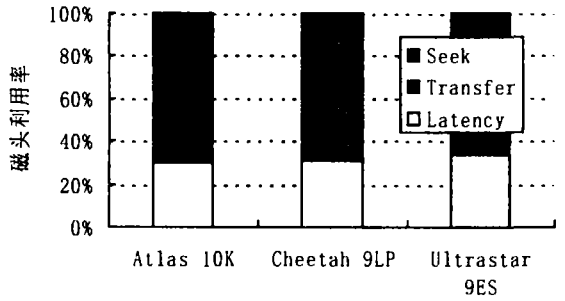

图2　几种现代磁盘的磁头利用情况

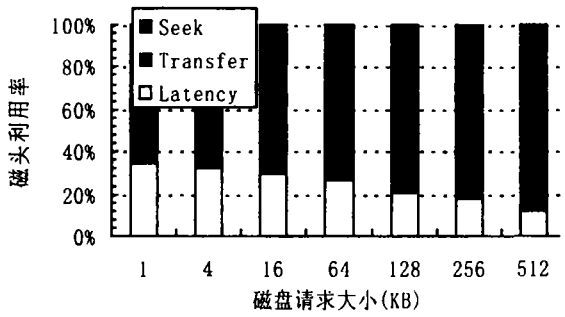

图3　磁头利用率和磁盘请求大小的关系

图2是几种现代磁盘的磁头利用率统计[6]。从图中可以看到，大多数现在磁盘有30-40%的磁盘带宽是潜在带宽，可以使用 ZLA 进行利用。图3是磁盘访问大小和磁头利用率的关系，从图中可以看到大的请求尺寸使磁盘的寻道时间和旋转时间在整个磁头利用率中的比例降低了，这是因为大的请求要求较长的磁头读写时间。文[7]指出在 UNIX 环境中，80%的文件访问是小于10kB。此次，在文件系统中，潜在带宽通常都有30%-40%。

3. 磁盘散热机制

3.1　文件温度监控

在现代文件系统中，文件的访问频率是不一样的。整个文件系统中的不同文件，可能具有非常悬殊的文件访问频率[8]。因此，为了提高文件系统的效率，一种有效的方法是将访问频率高的文件放置在磁盘上优化的位置。这样当对这些文件进行访问的时候，可以减少寻道和旋转的时延。因此，可以在文件系统中统计文件的访问次数，假设文件的访问次数称为文件的热度，单位时间内文件的访问次数称为文件的温度，温度较高的文件称为"热文件"。因此，可以修改文件系统的实现，将热文件所在磁盘块进行复制，并均匀分布在磁盘上。这样，可以提高这些磁盘块的访问速度。

为了确定在文件系统中哪些文件是"热"文件，必须对文件系统进行修改。当文件系统中对一个文件进行访问时，必须记录文件的读写次数。可以在文件系统的 i 节点中添加两个记录项，一个用以记录文件的访问次数，即热度；一个用以记录单位时间内文件的访问次数，即温度。每次文件访问时，系统都要将文件的热度加1，然后依据热度计算出文件的温度。当文件的温度达到一个阈值时，系统将触发文件的降温操作，对文件进行复制，实现文件的降温。

3.2　文件的降温

图4　热文件的散热示意图

如图4所示，设文件 F 是一个热文件，文件的降温过程就是将文件 F 所包含的磁盘块进行复制，放置在磁盘的合理位置上，以实现文件的降温。如图4中所示的 F_1 和 F_2。当文件 F 被复制了以后，如果用户需要访问文件 F，那么可以任意地选择 F 的一个拷贝进行访问，这样可以降低文件寻道的次数。文件在磁盘上的复制个数，和文件 F 的温度有关。F 温度越高，复制的个数越多。

复制的磁盘块在磁盘中的放置位置，可以选择放置在原来磁道的对称位置。设文件 F 的磁盘块在 T 磁道，文件 F 将复制 N 份，那么第 n 个复制的磁盘块在磁盘上的磁道数 T_n 可以依据式(1)进行计算。

$$T_n = T + n \times [T_{MAX}/(N+1)] Mod\ T_{MAX} \quad (1)$$

在式(1)中，T_{MAX} 是磁盘的磁道总数，Mod 代表取余运算。

当一个文件经历了"降温"处理以后，在文件的 i 节点中，应该同时记录每个拷贝的磁盘块在磁盘上的位置。当系统访问这个文件的时候，磁盘驱动程序可以依据当前磁头的位置和运动方向，选择一个最近的拷贝进行访问，这样可以提高文件的访问速度。

要完成磁盘块的复制，必须对数据进行复制。当磁盘比较繁忙的时候，文件复制会加重磁盘负荷。避免这个问题的一种方法是在磁盘空闲的时候才进行数据的复制。这种方法的不足是：①如果是一个任务比较繁忙的服务器，例如交易事务服务器，那么可能在每周7×24小时的时间中，磁盘始终处于较重负荷状态；②这种方法不能及时地使热文件进行降温处理，缺乏处理的及时性。

因此，本文提出了一种新颖的数据复制方式。如图4所示，文件的降温区占据磁道的固定位置，例如降温区可以占据1/2、1/3或1/4的磁道。现代磁盘通常每个磁道有数百个扇区，因此每个磁道的1/2、1/3或1/4大小的容量可以达到64k-128kB。文[9]指出，在 UNIX 环境中文件系统中的绝大部分文件大小小于32k，在 Windows 环境小于64k。因此，降温区的大小足够容纳一个完整的文件，可以使一次文件读写在一次磁盘访问中完成。降温区的具体大小可以依据文件系统中文件的大小和磁道的扇区数而定。

当文件系统访问热文件 F 后，F 的内容被放置在内存中。然后，当磁头寻道到 F 的降温区磁道进行读写时，就可以

将内存中F的数据插入到原来的读/写任务中，从而组合为一次较大的磁盘读/写访问。因为现代磁盘都具有ZLA特征，因而插入额外的读写数据后，不会为原来的读/写访问带来明显的延迟。

在图4所示的散热示意图中，文件 F 数据块的磁盘分布是连续的。在很多情况下，文件 F 的数据块可能是不连续分布的，例如分布在不同的磁道上。在这种情况下，通过降温处理后，原来不连续的数据块分布在降温区中可以转变为连续的数据块分布，这种转变也导致文件访问性能的提高。

3.3 文件的写同步

当文件 F 完成降温处理后，一个重要的问题是如何保持 F 的多个拷贝之间的数据一致。在ABL中，在每个文件的i节点中增设了拷贝的同步属性位。当需要对文件 F 进行写的时候，驱动程序可以依据磁头的当前位置和移动方向，选择一个最近的拷贝进行修改。然后，在 F 的i节点中将其他所有拷贝的同步属性设置为“disabled”，表示需要进行同步操作。ABL中的文件写同步有两种方式，一种使用ZLA方式，一种使用插入同步方式。当磁头下一次在标记为“disabled”的拷贝所在的磁道进行读写的时候，可以使用ZLA方式进行写同步，也即将该拷贝的内容用内存中保存的修改内容进行更新，然后可以将该拷贝的同步属性修改为“enabled”。如果某个拷贝所在的磁道长时间没有被访问到，那么系统可以使用插入同步方式，也即在磁盘队列中插入一个同步访问来专门进行同步操作。使用ZLA方式对磁盘性能不会带来明显的降低，而使用插入同步方式会对磁盘性能带来一些降低。但是，在重负荷的磁盘中，各磁道访问频繁，因此使用ZLA方式的概率比使用插入同步方式的概率大得多，因此对磁盘性能的总体损失很小。

4. 性能测试结果

在进行实际的性能测试之前，本文首先使用一个模拟试验来检测ABL的有效性。假设有一个磁盘访问序列，访问的磁道号依次为：(53,98,2,183,37,122,14,1,124,65,67)，分别使用SCAN方式和SCAN+ABL方式对这个访问队列进行访问。设磁盘的磁道号为1-200，平均寻道时间为5ms。假设当前磁头在1道，并且正在向外部运动。在ABL中，假设磁道2和磁道3是访问频繁的热磁道，它们的数据被复制到磁道102和103上。试验结果如图5所示，图中左侧坐标是磁道号，右侧坐标是访问时间，单位ms。从图中可以看到，由于第2道和第3道被复制到第102和103道上，SCAN+ABL方式对2和3磁道的访问可以在进行了第98道以后就进行，而SCAN方式对2和3磁道的访问必须在访问了14磁道后进行。从图5中的访问时间可以看到，对第2道和第3道的访问，SCAN+ABL方式在第245ms和250ms就可以完成，而SCAN将在1720ms和1725ms才能进行访问，并且SCAN+ABL方式的总寻道数量也比SCAN方式更少。从图5可以看到，使用ABL方式后，可以大大地提高磁盘热磁道的访问速度，并且也使磁盘总体寻道数量减少了。

图6和图7是实际测试的结果。在试验中，总的文件访问次数为8000个，SCAN队列长度为20。硬盘为1024的磁道，每磁道128个扇区，设热文件数量为50，分布的磁道为：20×n，其中n=1,…,50。每个热文件的访问概率为2‰，其余文件均匀分布在其他各磁道，访问概率为(1.0-0.1)/(1024-50)=0.924‰。两图中，降温区的分布依据公式(1)来确定，1:1降温N=1，1:2降温N=2。图6是访问总体时间对比图，可以看到使用ABL后，磁盘访问总体时间降低了，但是降低的幅度不大，这是因为ABL主要是将热文件的访问提前了，并没有显著地减少访问次数。图7是热文件访问平均延迟的对比图，可以看到使用ABL后，热文件的访问延迟降低了50%以上，证明ABL可以非常显著地提高热文件访问性能。并且，文件的温度越大，ABL可以使用更多的降温区来进行降温处理，使热文件的访问延迟始终保持较低。

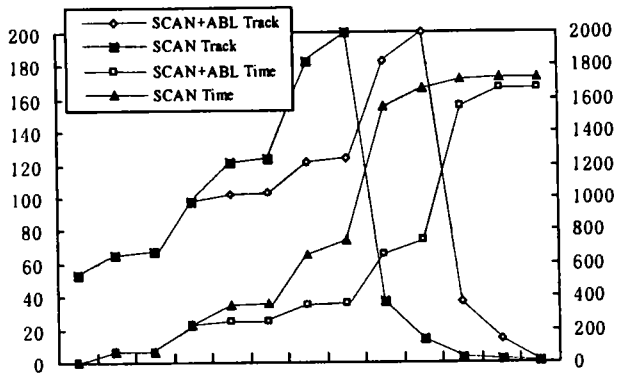

图5 ABL的寻道和延迟

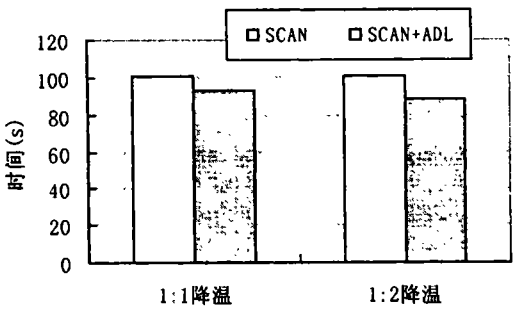

图6 总体访问时间

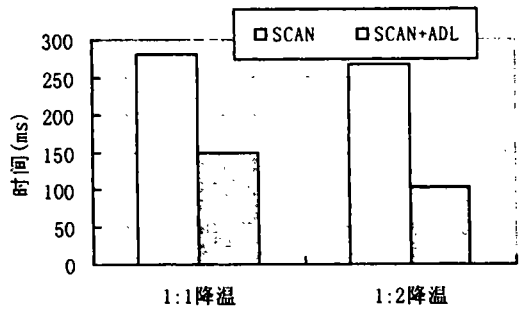

图7 热磁道访问延迟

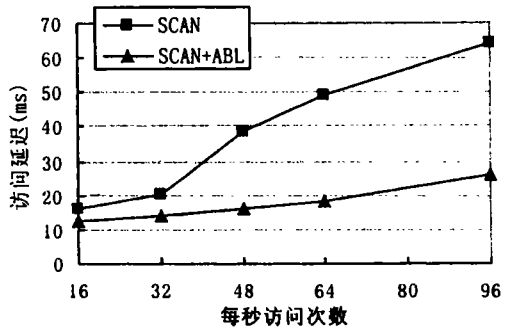

图8 访问时延随访问频率变化对比

图8是热文件的访问时延随访问频率变化的对比图。在试

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