第34卷 第7期 计 算 机 工 程 2008年4月
Vol.34 No.7 Computer Engineering April 2008
·博士论文· 文章编号：1000—3428(2008)07—0004—03 文献标识码：A 中图分类号：TP393

蓝鲸分布式文件系统元数据服务

杨德志[1,2]，许 鲁[1]，张建刚[1]
(1. 中国科学院计算技术研究所，北京 100080；2. 中国科学院研究生院，北京 100039)

摘 要： 文件系统元数据请求占据了所有请求50%以上的比例，文件系统元数据服务对整个文件系统有着重要的影响。该文介绍了蓝鲸分布式文件系统的元数据服务器集群(BWMMS)的设计方法。BWMMS通过高扩展的系统结构和简单灵活的元数据请求处理协议，完成元数据服务的协同处理过程。初步评估表明，BWMMS的元数据管理机制能够提供较好的元数据处理性能，具有较高的扩展能力。

关键词： 网络存储系统；分布式文件系统元数据服务；元数据请求原子性保证

Blue Whale Distributed File System Metadata Service

YANG De-zhi[1,2], XU Lu[1], ZHANG Jian-gang[1]
(1. Institute of Computing Technology, Chinese Academy of Sciences, Beijing 100080;
2. Graduate University of Chinese Academy of Sciences, Beijing 100039)

【Abstract】 File system metadata requests may make up over 50% of all file system operations, which should be of critical importance in large scale distributed file system. This paper designs a metadata server cluster, named BWMMS, to provide metadata service in Blue Whale1000 network storage system. Due to its dynamic, logical and impartial rules, BWMMS can provide low latency and high scalability metadata services to system uses.

【Key words】 network storage system; distributed file system metadata service; metadata request operations' atomicity

已有的研究成果表明，文件系统中用户访问文件数据需要的文件元数据请求数量所占比例非常大。文献[1]描述了6个系统的结果，其中元数据请求所占比例分别为15.65%, 50.09%, 70.24%, 88.31%, 61.21%和38.89%，文件系统的元数据服务是分布式文件系统研究的重要内容，集群方式的元数据服务器结构成为其主流。但如何组织管理服务器、分布元数据请求负载和协同各个服务器完成原子性的元数据请求处理是其目前要解决的主要问题。本文介绍了蓝鲸分布式文件系统(BWFS)[2]的元数据服务器集群研究结果。

1 相关研究

文件系统元数据服务的研究是分布式文件系统研究的重要内容。从提供文件系统元数据服务的服务器系统结构来看，现有的研究主要分为没有集中服务器[3]和存在集中服务器[4]两大类。“是否有集中服务器”是指是否存在单一集中点决策元数据请求负载的分布。没有集中服务器的系统要求提供文件元数据服务的服务器间相互协同，共同完成元数据的分布决策。在有集中服务器的系统中，关键信息由集中点控制，所有元数据服务器仅与集中点交互，完成元数据的分布。没有集中点的系统需要服务器间相互通信，分布的任何变动都可能需要广播给所有的元数据服务器。从结构上看不存在系统瓶颈，系统采用的某些机制等仍然可能限制系统的扩展。采用集中点的系统结构中的集中点可能成为系统的瓶颈，但通过优化元数据请求的处理路径，集中点的瓶颈问题将大大降低，能够满足系统一定规模的扩展需要。

从元数据请求负载分布策略来看，这些系统可以分为根据文件系统的目录树结构的“目录子树分区策略”[3]和以文件路径名、索引节点号等因素为参数进行哈希的“哈希策略”[4]完成元数据请求负载的分布管理两大类。目录子树分区策略仅仅以文件系统目录树的静态结构为参数，没有考虑用户访问的动态变化，使得它既不能支持文件系统目录树结构的深度扩展问题，还限制了某些操作不能跨服务器进行。哈希分布策略能够快速定位元数据服务器，能够提高不需要根据目录树结构遍历的应用的访问速度，但是它对于需要根据目录树结构进行遍历的应用支持不够。静态设计的哈希函数同样不能很好地适应用户访问的动态性。哈希参数的任何改变，不仅影响被哈希的文件，而且哈希结果还可能通过文件系统的目录树结构放大出去，导致大范围的元数据迁移，引起系统突发的性能下降。

两阶段提交协议[5]或其变种是请求原子性保证的常用协议，但它的协议开销很大，服务器需要永久存储请求的重要状态，系统的故障恢复协议较复杂。

2 蓝鲸元数据服务器集群设计

2.1 研究背景

BWFS[2]是国家高性能计算机工程技术研究中心(NRCHPC)自主设计用于海量网络存储系统的分布式文件系统，为系统用户提供文件级存储服务。BWFS采用专用服务器提供数据和元数据服务。系统用户通过元数据访问协议从元数据服务器获得文件属性后，再通过网络设备访问协议从存储服务器获取文件数据。文件访问的数据流与控制流的有效分离机制为系统客户提供高并发和高扩展的数据能力。

本文下面将介绍蓝鲸元数据服务器集群(BWMMS)的系

基金项目： 国家自然科学基金资助项目(60373045)
作者简介： 杨德志(1977－)，男，博士研究生，主研方向：网络存储，分布式文件系统；许 鲁，研究员、博士、博士生导师；张建刚，副研究员、博士
收稿日期： 2007-04-30 **E-mail：** yangdezhi@nrhpc.ac.cn

统结构、元数据分布管理和文件系统一致性协议。

2.2 集群系统结构

集中控制的元数据分布机制能够简化系统，提高系统的管理能力。但是为降低集中点成为系统瓶颈的可能性，元数据请求处理路径需要合理安排，BWMMS 将负责元数据分布决策的集中点绑定服务器(BS)放置在文件系统元数据请求处理路径的尾端，如图 1 所示。

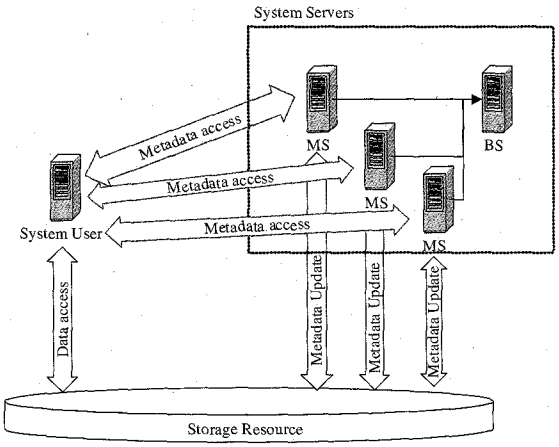

图 1 BWFS 元数据服务器集群结构

BWMMS 由若干元数据服务器(Metadata Server, MS)和一个绑定服务器(Binding Server, BS)组成。所有的元数据服务器通过网络块设备访问协议在集中共享存储空间中进行分布式文件系统元数据的存取访问。

BS 主要负责元数据服务器集群节点的管理和元数据处理负载在各个服务器的分布。在正常情况下，MS 独立对等地为系统用户(AS)提供元数据服务，在必要时才与 BS 进行通信，并通过 BS 完成多个元数据服务器的协同。

2.3 元数据分布管理机制和策略

目录子树分区法和哈希法以文件系统静态属性为决策参数，以整个系统的所有文件为决策对象，这与已有研究结果不符。事实上，系统用户的文件系统元数据访问在一定时间内表现出局部性和动态性的特征，请求集中在占很小比例的部分元数据集上。

BWMMS 的元数据分布策略以文件系统静态结构和用户元数据的动态访问为参数，仅针对用户访问到的小部分文件系统元数据进行分布管理。

为描述 BWMMS 的元数据分布策略，需要定义活跃元数据、正交映射和元数据宿主等。

定义 1 活跃元数据。活跃元数据是指正在被用户访问的文件系统元数据，访问包括读和写。当一个文件元数据为活跃元数据时，其文件路径名上的前缀目录的元数据都是活跃元数据。

定义 2 正交映射。元数据的读写权限分配给元数据服务器的过程称为“元数据分布映射过程”。“正交映射”指的是满足这种条件的元数据分布映射：一个元数据被映射后，有且仅有一个元数据服务器负责其读写请求的处理。

定义 3 元数据宿主。元数据当前映射到的元数据服务器称为该元数据的宿主。

BWMMS 的各个 MS 以一定时间间隔向 BS 汇报自己的负载情况，供 BS 元数据分布决策参考。目前 MS 收集的参数包括 CPU、内存、网络和已经分布在自己的元数据的个数等 4 个参数。BS 将这 4 个参数进行加权平均，计算出每个 MS 的负载情况，供决策使用。

元数据分布策略需要考虑 3 种情况：(1)系统用户可能要求限定元数据宿主的范围；(2)从系统角度看，对文件系统元数据的访问表现出局部性，并且某些元数据间表现出访问上的相关性，如一个目录和其下所有孩子文件的元数据；(3)服务器集群需要考虑服务器负载的相对公平性，避免某些元数据服务器负载过大而成为系统的性能瓶颈。根据以上策略考虑，BWMMS 的元数据宿主选择算法如下：

```
/* MS_limit 表示 MS 选择范围，rMS 是访问上有逻辑关系的元数据的宿主。
*/ if (用户设定宿主范围) {
        MS_limit <----符合设定条件的 MS;
   } else {
        MS_limit <----所有 MS;
   }
   if  (需要考虑访问的关联性)  {
        rMS <---  参考元数据的宿主 MSS;
        检查 rMS 的负载。
          if (rMS 负载小于其负载上限) {
           将 rMS 设置为该元数据宿主;
           返回结果;              }
   }
   将 MS_limit 中负载最小的 MS 设置为该元数据的宿主。
   返回结果。
```

当活跃元数据因为没有访问变得不活跃后，元数据宿主通知 BS 释放对该元数据的分布信息管理。

2.4 元数据请求事务性保障机制

在元数据服务器集群的环境中，用户的元数据请求不可避免地需要多个 MS 协同完成，如何保证这些请求处理不会影响系统的完整性成为重要的问题。现有研究将元数据存放和请求处理紧耦合在一起，导致请求处理需要两阶段提交协议等分布式事务协议。同时，为保证请求处理的事务性和操作协议的容错性，每个元数据服务器必须保存操作过程中的重要状态信息，为系统的故障恢复提供支持。系统的故障恢复需要根据 MS 记录的信息，通过复杂的协议完成。

BWFS 使用集中共享虚拟存储来集中存储文件系统的元数据和数据，分布式分层完成存储资源的管理[6]。

文件系统元数据的存储和访问的分离，使得 BWMMS 可以通过简单的请求驱动的元数据宿主改变将分布式元数据请求处理变成集中式处理。从而消除复杂的处理协议，影响性能的状态记录和复杂的错误恢复协议。其基本思想是通过 BS 的协同完成元数据宿主的改变，将请求涉及到的所有元数据集中到一个元数据服务器，然后完成元数据请求的处理。本地文件系统日志技术被用来记录请求的处理情况，以提高错误恢复的速度。宿主改变的对象是单个元数据，包括 inode 和目录数据块的内容。宿主改变过程仅涉及到发迁移请求的 MS(主动 MS)、BS 和接收迁移请求的 MS(被动 MS)。

(1)基于请求驱动的元数据宿主改变协议

BWMMS 采用分层的机制管理元数据映射信息。BS 记录当前系统活跃元数据的映射信息。各个元数据服务器和客户机根据访问缓存 BS 记录的信息。当出现冲突时，以 BS 记录的信息为准。

元数据宿主的改变过程需要元数据内容在 2 个 MS 间的迁移和元数据分布信息的更改 2 个步骤。分布信息的更改需更改 BS 记录的元数据映射信息和 2 个 MS 缓存的分布信息。为简化过程，主动 MS 通过 BS 要求被动 MS 将元数据写回存

储设备，然后从存储设备读回元数据。图 2 给出了元数据宿主改变中所有服务器的操作时序，其过程描述如下：

1)AS 将元数据请求发送给 MS1。

2)MS1 查找元数据分布记录，获得(obj, not-on-me)。MS1 向 BS 请求获得该元数据请求涉及到的元数据的宿主权限。

3)BS 检查请求的元数据当前的宿主服务器信息，获得(obj, MS2)。BS 通知 MS2 释放管理权限。

4)MS2 在能够完成权限更改的条件下，将元数据更新到 SN。更新完成后，更改自己维护的元数据分布记录(obj, MS2)-→ (obj, MS1)，并返回结果 BS。

5)BS 更改元数据分布记录(obj, MS2)-→(obj, MS1)，赋予 MS1 宿主权限。

6)MS1 更改元数据分布记录(obj, not-on-me)-→((obj, MS1)。然后从 SN 获得元数据信息，完成相应的请求。

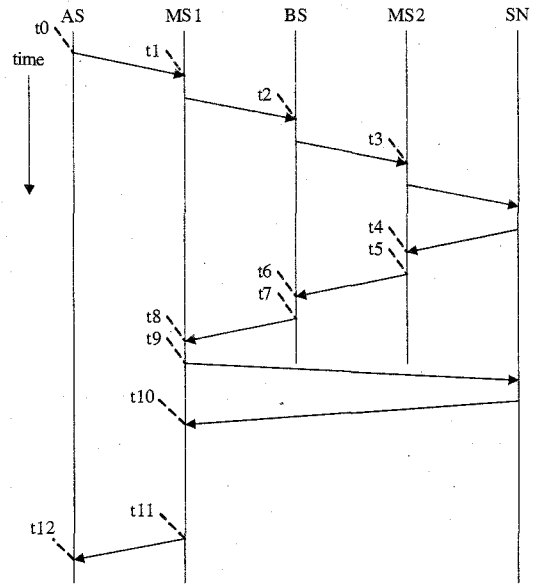

图 2　元数据宿主改变协议时序

由于 MS 记录的映射信息是 BS 记录的全局信息的缓存，因此在 BS 正常工作情况下，任意 MS 宕机，系统均能正常工作。同样，在 MS 都正常工作的前提下，即使 BS 宕机，系统也能够恢复。但是，如果 BS 和任一 MS 同时出现故障，系统将崩溃。

(2)分布策略效率对比

由于实现成本问题，本文没有实现目录分区子树法和哈希法，分布策略的效率对比通过逻辑分析进行。

目录子树分区法以子目录为单位进行分布管理。在绝大多数情况下，元数据请求由单个服务器完成，不需要与其他服务器的通信，也不需要写设备。故其元数据请求处理延迟较低，能获得较高的吞吐率。但是，由于其存在目录不能跨区迁移的问题，在功能上存在缺陷，并且存在元数据请求性能的深度扩展问题。哈希法按照哈希函数计算元数据分布，不需要与集中服务器进行通信。但是，如果其哈希参数发生变化，则其结果将被文件系统目录树无限放大。假定目录下有 10 级子目录，每级目录下有 10 000 文件，哈希函数以文件名为参数。当目录名字发生改变，系统需为(10×10 000 + 1)个元数据分布重新哈希，并有可能导致突发的相同数目的元数据迁移。在 BWMMS 环境中，目录的迁移不会导致任何其他元数据的迁移，其带来的影响是将单个服务器能够完成的操作变成分布式的操作。但是，BWMMS 的动态影射机制可以大幅度降度迁移带来的负面影响。

BWMMS 与两阶段锁加两阶段提交的事务保障协议对比可以用删除文件进行。后者首先需要锁住父目录，在父目录中将被删除文件的目录项删除。然后，需要在锁住父目录的前提下，要求文件所在服务器完成删除文件需要的操作。在文件删除后，父目录服务器发起同步更新的请求，得到肯定回答后，2 个服务器将更新写回到设备，文件服务器更新完毕后，向父目录服务器返回结果。这个过程需要服务器间的 3 次通信和 2 次同步写设备。而 BWMMS 则首先通过迁移协议将元数据迁移到父目录的宿主服务器，再完成文件的删除。迁移过程需要服务器间 2 次通信，MS2 的 1 次写设备和 MS1 的 1 次读设备。并且，在要求文件服务器删除文件时，两阶段将父目录锁住，阻塞对父目录其他文件的访问，而 BWMMS 仅仅对父目录设置标志，供请求迁移父目录的过程决策使用，它并不阻塞对父目录下文件的访问。

总之，与现有系统比较，BWMMS 的动态性为简单、有效的文件系统元数据服务提供了必要和足够的支持。

3　系统原型实现与评估

为验证设计，本文实现一个由 2 个元数据服务器构成的原型系统，BWMMS 的动态机制能够支持更多数量的服务器的加入。原型系统基于 Redhat8.0, EXT3, JBD, NBD 和 NFSv3 实现。MS 通过扩展 NFSv3 协议提供元数据服务。BS 根据访问动态决定请求的分布。AS 和 MS 缓存元数据映射信息。

原型系统的评估采用在客户端运行 1~16 个进程的 DBench[7]对比测试单个元数据服务器的 BWFS，2 个元数据服务器构成的 BWMMS 和标准的 NFSV3 的扩展能力。所有机器硬件为 Celeron®1 GHz, 256 MB 内存，千兆以太网连接，软件为 Redhat®8.0，用作 NFS Server 的节点使用 ext3 文件系统，BWMMS 由 2 个元数据服务器构成。NFS Server 为 8 个进程。结果如图 3 所示。

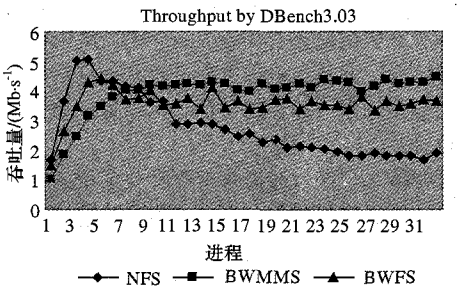

图 3　NFS,BWFS 和 BWMMS 的吞吐率对比

从图 3 可以看出，由于 BWMMS 要求额外的请求映射通信，在进程数小于 8 时，NFS 的处理性能最好。当用户进程大于系统服务进程数(NFS 和 BWFS 均为 8，BWMMS 为 16)时，吞吐率出现一定程度的下降。但是，BWMMS 针对元数据的活跃性进行元数据请求负载管理管理的动态灵活的机制和策略为系统吞吐率提供了保证，使得系统聚合吞吐率的下降幅度并不明显，并呈现出较平稳的趋势，体现了系统的扩展能力。

4　结束语

本文介绍了 BWFS 的元数据服务，并通过初步的评估验证其扩展能力。在未来的工作中，将着重于系统的评估和优化，并深化系统可靠性和可用性的研究。

(下转第 9 页)

先进行列表调度得到调度长度 c，再在 c 个周期的功能单元资源约束下做调度。设指令数目为 n，簇数目为 N_c，外层循环次数为 n，内存循环次数最大值为 cnN_c，即二维作用力的最多可能个数，对 OP_k 能否被调度到(CS_i,C_j)上的尝试算法复杂度为 $O(n^2)$，因此，总的算法复杂度为 $O(cn^4)$。

4 实验结果

在多簇处理器的调度算法中，UAS 算法的效果较好。列表调度算法也可直接用于 RFCC-VLIW 处理器。本文对二维力量引导调度算法、UAS 算法和列表调度算法进行比较。如第 3 节所述，对于 RFCC-VLIW 结构处理器，由于其簇间数据传输无延时也无须额外拷贝操作，且指令调度的首要目标是调度长度最短，因此各种算法所得的调度长度都与列表调度长度相同。对于算法优劣的评估，应该对互连寄存器堆的访问次数和使用数目进行比较。

TGFF(Task Graphs For Free)是一种随机任务流图生成器[6]，可以通过对参数的设置灵活地产生各种不同的随机任务流图。本文以四簇八发射 RFCC-VLIW 处理器为目标，分别用二维力量引导调度算法、UAS 算法和列表调度算法，对 TGFF 生成的指令数由 50 变化到 1 000 的 20 个随机调度图进行调度。图 3 为 3 种算法对互连寄存器堆的访问次数的比较，图 4 为使用互连寄存器数的比较。

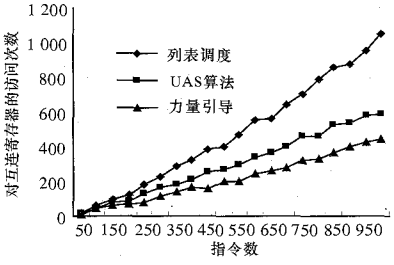

图 3 互连寄存器堆访问次数的对比

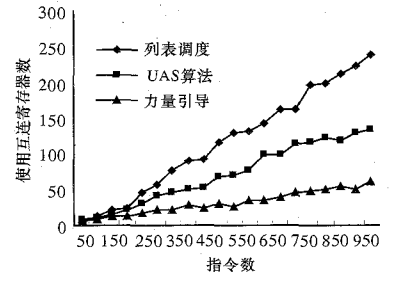

图 4 使用互连寄存器数的对比

可以看出，本文提出的二维力量引导调度算法效果最好。与 UAS 算法相比，二维力量引导调度算法对互连寄存器堆的访问次数减少了 20%~30%，对互连寄存器的使用数目减少了 30%~60%。

UAS 等调度算法都仅是根据前驱和后继指令来局部性地判断如何进行簇的分配，对每条指令进行调度时无法考虑到其他未调度指令的信息；而二维力量引导算法通过对力量的计算全局性地估计了所有指令可能对功能单元的占用情况，同时也考虑到了指令之间的相关性，因此，得到了更好的调度结果。

5 结束语

本文提出了一种用于 RFCC-VLIW 结构的二维力量引导调度算法，利用力量引导算法中均衡的思想，将指令分布到处理器的各个周期和簇中，同时又可将相关性较强的指令分布到同一个簇中。该算法既保证了调度长度最短，又可以减少功能单元对互连寄存器堆的访问次数和使用的互连寄存器数目，优于现有的其他多簇调度算法。

参考文献

[1] Ozer E, Banerjia S, Conte T M. Unified Assign and Schedule: A New Approach to Scheduling for Clustered Register File Microarchitectures[C]//Proc. of the 31st Int'l Symp. on Microarchitecture. Dallas, TX: [s. n.], 1998.

[2] Ellis J R. Bulldog: A Compiler for VLIW Architectures[M]. Cambridge, MA: MIT Press, 1986.

[3] Capitanio A, Dutt N, Nicolau A. Partitioned Register Files for VLIWs: A Preliminary Analysis of Tradeoffs[C]//Proc. of the 25th Int'l Symp. on Microarchitecture. Portland, Oregon, USA: [s. n.], 1992.

[4] Zhang Yanjun, He Hu, Sun Yihe. A New Register File Access Architecture for Software Pipelining in VLIW Processors[C]// Proceedings of ASP-DAC. Shanghai, China: [s. n.], 2005.

[5] Paulin P G, Kight J P. Force-directed Scheduling in Automatic Data Path Synthesis[C]//Proc. of the 25th Design Automation Conference. Anaheim, California, USA: [s. n.], 1988.

[6] Rhodes D, Dick R. TGFF[EB/OL]. [2007-04-29]. http://ziyang.ece.northwestern.edu/tgff.

（上接第 6 页）

参考文献

[1] Ellard D, Ledlie J, Malkani P, et al. Passive NFS Tracing of Email and Research Workloads[C]//Proc. of FAST'03. San Francisco, CA: [s. n.], 2003.

[2] Yang Dezhi, Huang Hua, Zhang Jiangang, et al. BWFS: A Distributed File System with Large Capacity, High Throughput and High Scalability[J]. Journal of Computer Research and Development, 2005, 42(6): 1028-1033.

[3] Menon J, Pease D A, Rees R, et al. IBM Storage Tank-A Heterogeneous Scalable SAN File System[J]. IBM System Journal, 2003, 42(2): 250-267.

[4] Anderson D, Chase J, Vahdat A. Interposed Request Routing for Scalable Network Storage[C]//Proc. of the 4th ACM/USENIX Symposium on Operating Systems Design and Implementation. San Diego, CA: [s. n.], 2000.

[5] Bernstein P A, Hadzilacos V, Goodman N. Concurrency Control and Recovery in Database Systems[M]. Boston, MA, USA: Addison-Wesley Publishing Company, 1987: 236-250.

[6] Huang Hua, Zhang Jiangang, Xu Lu. Distributed Layered Resource Management Model in Blue Whale Distributed File System[J]. Journal of Computer Research and Development, 2005, 42(6): 1034-1038.

[7] DBench[Z]. (2007-04-20). http://samba.org/ftp/tridge/dbench/.