高性能计算的海量存储系统新型访问策略分析

朱 平[1,2], 李全龙[1], 徐晓飞[1], 朱建涛[2], 黄永勤[2]

(1. 哈尔滨工业大学 计算机科学与技术学院, 150001 哈尔滨; 2. 江南计算技术研究所, 214083 江苏 无锡)

摘 要: 为解决海量信息处理中实时访问的"I/O 墙"问题,提高海量信息分布式存储系统的性能,提出了一种基于 HPC 的存储部件新型访问策略. 首先分析了传统访问模型存在的问题;其次研究了存储部件直通路模式的工作机理, 建立了存储系统的多层次、分布式模型,根据不同层次和映射策略实现存储空间物理地址、缓存地址、存储系统逻辑空间地址的连续映射;继而分析了直通路访问模式下的存储路径时间开销;最后在模拟环境下进行存储部件访问的性能测试,并在实际应用系统中对该策略进行验证. 验证测试结果表明,该方法能够有效提高存储系统性能,满足海量信息处理的实时性需要.

关键词: 高性能计算;海量存储系统;存储部件直通路;存储层次映射

中图分类号: TP333 **文献标志码:** A **文章编号:** 0367 - 6234(2012)11 - 0059 - 06

Research on the new access policy of storage unit under HPC mass storage system

ZHU Ping[1,2], LI Quan-Long[1], XU Xiao-fei[1], ZHU Jian-tao[2], HUANG Yong-qin[2]

(1. School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China;
2. Jiangnan Institute of Computing Technology, 214083 Wuxi, China)

Abstract: To solve the "I/O wall" problem in the case of real-time accessing about mass information processing and to improve performance of distributed mass storage systems, an access policy based on storage unit pass-through is proposed and the problem of traditional access models is analyzed. Then the mechanism of pass-through pattern is studied, and a multi-level and distributed model is built up. Next, the continuous mapping of physical address, cache address of storage space and logical space address of storage system are realized depend on the different levels and mapping strategies. The time consuming of pass-through storage path in pass-through pattern is analyzed. Last, the performance of the storage unit in the simulated environment is tested. The results show that the method can improve the performance of storage system effectively, and can meet the needs of real-time accessing about massive information processing.

Key words: HPC; mass storage system; storage unit pass-through; map of storage hierarchical structure

最新公布的第37届国际高性能计算 TOP500 最快计算机是以 8 162 Tflops 的持续性能指标而荣登榜首,它是 1946 年第 1 台计算机 ENIAC 的 16 324 亿倍. 超性能计算不断以增加节点来增加系统性能,未来 E 级 HPC 系统规模将变得异常庞大,其存储系统带来了扩展性、I/O 性能和可用性等诸多严峻挑战[1].

1 问题提出

信息存储系统对 UC-HPC 至关重要,其性能优劣会严重影响系统的总体性能,包括 I/O 操作以及处理器间通信等. 图 1 反映了计算性能与I/O 性能存在难以弥合的"I/O 墙". 因此,本文基于此背景下研究了 HPC 海量存储系统的存储策略.

"I/O 墙"产生的原因主要有:CPU 性能每年增长超过 60%,而磁盘性能每年仅有 4% ~7% 的

收稿日期: 2011 - 10 - 12.
基金项目: 国家高技术研究发展计划资助项目(2009AA01A402).
作者简介: 朱 平(1965—),男,高级工程师;
徐晓飞(1962—),男,教授,博士生导师;
黄永勤(1955—),女,高级工程师,博士生导师.
通信作者: 朱 平,fendicmm@ sina. com.

速度增长;在并行分布共享多机系统中加重了主机与I/O速度的失配性;多处理器与多核系统使其整体性能以每年80%以上的速度增长;网络、多媒体以及巨型复杂课题等一些新应用领域产生了日益增长的I/O要求. 这些因素都加大了计算和存储系统的性能差距,加厚了"I/O墙"[7-8]. 外存储器与高性能计算的发展存在明显的"间隙",如图2所示.

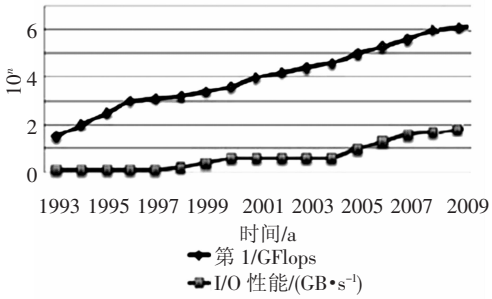


图1 TOP 500计算性能和I/O性能的趋势

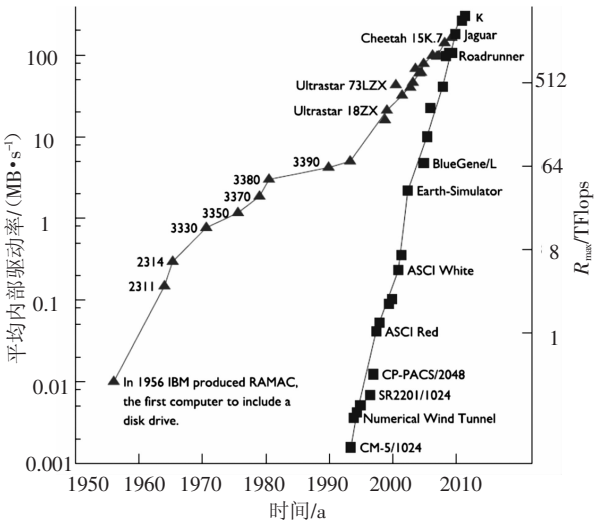


图2 磁盘存储和TOP500高性能计算的发展历程

缓解"I/O墙"的方法有:研究新型高速存储载体、体系结构、多级存储模式、缓存技术、外存系统调度策略、RAID技术、文件分条、并行I/O技术、网络存储技术等[2-3]. HPC的I/O硬件并行度远低于计算节点的并行度,使I/O性能与计算性能不匹配问题变得更加严重[4-5]. 系统规模越大越是严重地阻碍了下一代HPC计算能力的发挥[6-8]. 目前,全球已公开在研万万亿次级超级计算机系统如表1所示.

I/O及存储技术在高性能计算机的发展中始终是一个十分重要的关键技术,由其构成的系统是高性能计算机系统中的重要组成部分. 其技术特性决定了计算机I/O的处理能力,进而决定了计算机的整体性能以及应用环境. 为了提高存储系统的性能,人们对存储系统访问策略进行了研究并取得进展. HPC对存储系统的迫切要求有:超大系统规模和高性能,需要支持超过100 000颗处理器、数十万个节点的并发访问;支持系统的可扩展性,根据用户需求,HPC每4年并行I/O能力增长10倍,而实际用户需求每2年增加10倍,目前系统需支持GB/s乃至TB/s的I/O聚合带宽、数据高可靠和高可用、多核下存储系统软件研究等等.

表1 全球已公开的在研万万亿次级超级计算机系统

制造商	系统名称	目标性能/PFlogs	预计完成时间/a
	Sequoia(红衫)	20	2012
IBM	Blue Waters(蓝水)	10	2011
	Mira(米拉)	10	2012
富士通	K(京速)	10	2012
Cray	Titan(泰坦)	20	2012

这对下一代的高性能计算机存储系统结构、高性能存储载体部件、I/O通路、高性能存储网络、网络存储技术等都提出了新的挑战. 研究高性能计算机中存储系统关键技术,主要集中在:高性能存储网络研究、网络存储研究、高性能分布式文件系统、分布式多级缓存管理、分布式数据布局策略研究、网络存储虚拟化研究、网络存储系统高可用、分布式存储系统可扩展性研究、高性能存储载体研究、云存储技术研究等等. 本文重点研究提高存储系统性能的策略,如存储部件直通路访问策略,提出基于直通路方式的查找策略与系统实现.

2 新型存储部件访问策略

HPC海量存储系统需要对成千上万的存储对象进行随机访问,如何有效地实现系统的负载均衡,对提高系统的整体性能,充分有效利用系统资源至关重要. 在实际课题中有计算密集型、通信密集型、数据密集型及这3种组合型的应用课题. 其中本文需要重点关注与后两类有关的课题.

存储部件一般是由存储控制器和磁盘组成,其性能影响的关键是磁盘存储部件,由于近来新型的存储载体的性价比还存在问题,虽然有可能替代磁盘存储载体,但高性能存储系统大规模应用不能完全取代磁盘. 高性能计算存储系统将数据和元数据分离,但元数据是一个非大块交换量访问方式的数据,随机性强,根据I/O强度,有时访问密度很高,数据的局部性不是那么的好. 本文结合软、硬件条件提出改善这一状况的新思路.

系统存储性能的提高可以用多方面的解决方案,但是性能改善的源头还是存储部件,在使用一定的存储介质后,系统的输入性能已经基本确定.

增加带宽、采用并发并行的方法能够有效的缓解其性能不匹配的瓶颈. 本文在此基础上提出另外一种提高系统效率的思路，主要针对存储部件控制器经常需要在缓存中“转存 - 转运”数据而影响系统 I/O 性能的特点. 数据流是通过主机发出请求给存储部件，存储部件控制器解析，形成对缓存空间的请求，如果经过查找，不满足条件，需要从设备空间获取，这就需要分析和多次的传送数据. 如何利用数据流在设备间的连续传送，在协议基础上拓展以提高其传输效率是一个问题.

本文把整个空间及其数据流的转换限定为物理层、操作层、映射层、策略层和应用层等，如图 3 所示.

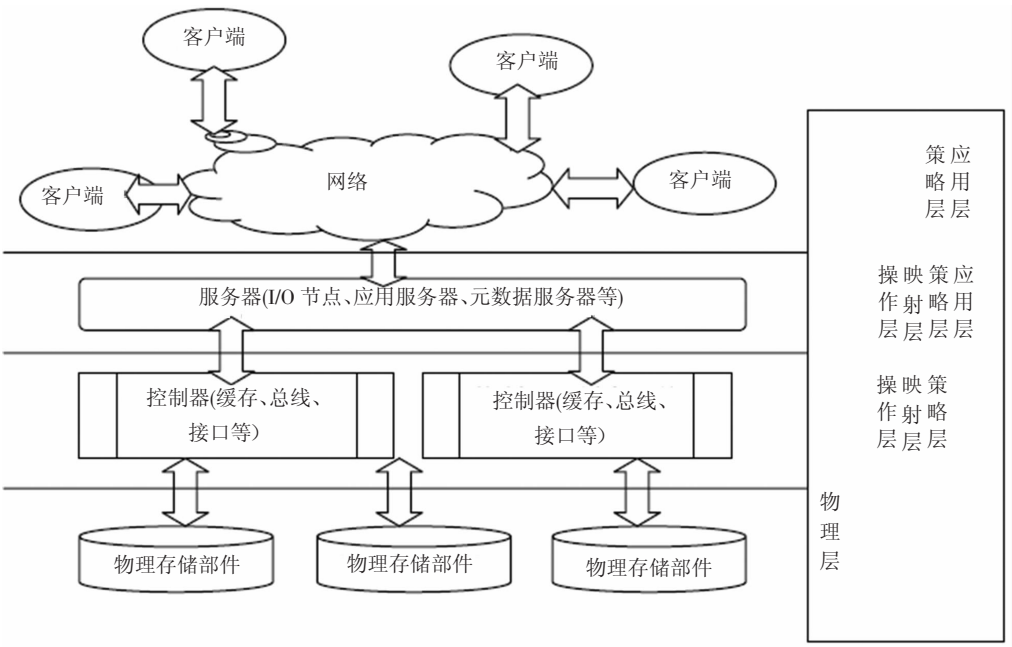


图 3　数据流及处理策略

设整个 HPC 的空间为 S_{HPC}，物理部件存储空间为 P_{AP_i}，控制器缓存空间 BufC_{AC_j}，元数据定义文件系统的空间 $\mathrm{MDS}_{\mathrm{AMDS}_k}$，每个 I/O 访问空间为 $\mathrm{IO}_{\mathrm{AIO}_l}$. 策略层有 P_a 个参数因子，映射层有 M_b 个参数因子，操作层有 O_c 个参数因子，它们将共同作用，完成系统的空间管理和数据流的方向. 用户请求的地址为 Addr_x 交换量为 Count_y，则有用户的请求和最终物理空间之间的关系有：

$$HY_{\mathrm{CIO}}(\mathrm{Addr}_{cx},\mathrm{Count}_{cy}) = \alpha(\mathrm{IO}_{\mathrm{AIO}_l},\mathrm{MDS}_{\mathrm{AMDS}_k},S_{\mathrm{HPC}},P_a,M_b,O_c). \quad (1)$$

式(1)表示用户申请访问的全局系统存储空间需要通过 Client 发出请求传给 MDS，经过 I/O 节点传送，使用多种策略和映射方法获取数据.

$$\mathrm{HIO}_{\mathrm{AIO}_l}(\mathrm{Addr}_{\mathrm{IO}x},\mathrm{Count}_{\mathrm{IO}y}) = \theta(\mathrm{IO}_{\mathrm{CIO}_l},\mathrm{MDS}_{\mathrm{AMDS}_k},\mathrm{BufC}_{AC_j},P_a,M_b,O_c). \quad (2)$$

$$\mathrm{HMDS}_{\mathrm{AIO}_l}(\mathrm{Addr}_{\mathrm{MDS}x},\mathrm{Count}_{\mathrm{MDS}y}) = \lambda(\mathrm{IO}_{\mathrm{CIO}_l},\mathrm{MDS}_{\mathrm{AMDS}_k},\mathrm{BufC}_{AC_j},P_a,M_b,O_c). \quad (3)$$

式(2)、式(3)表示不论 I/O 节点还是 MDS 获取数据需要通过存储部件的缓存及其控制器相应的映射策略等可以获取数据.

$$\mathrm{HBufC}_{\mathrm{AIO}_l}(\mathrm{Addr}_{\mathrm{Bufx}},\mathrm{Count}_{\mathrm{Bufy}}) = \lambda(P_{AP_i},\mathrm{IO}_{\mathrm{CIO}_l},\mathrm{MDS}_{\mathrm{AMDS}_k},\mathrm{BufC}_{AC_j},P_a,M_b,O_c).$$

而控制器的缓存获取数据则需要通过 P_{AP_i}、$\mathrm{IO}_{\mathrm{CIO}_l}$、$\mathrm{MDS}_{\mathrm{AMDS}_k}$、$\mathrm{BufC}_{AC_j}$、$P_a$、$M_b$、$O_c$ 等共同作用.

在 HPC 中一般应用集成的存储部件，接口间不同协议割裂了不同层之间的映射关系. 本文可以通过空间映射的方法，直接在 $\mathrm{IO}_{\mathrm{AIO}_l}$ 与 P_{AP_i} 以及 $\mathrm{MDS}_{\mathrm{AMDS}_k}$ 与 P_{AP_i} 间建立关系，可以形成一个或多个物理磁盘通过存储部件控制器和 I/O 服务器的存储空间或者 MDS 的存储服务空间的直接映射，产生直接的数据流动.

$$\mathrm{HIO}_{\mathrm{AIO}_l}(\mathrm{Addr}_{\mathrm{IO}x},\mathrm{Count}_{\mathrm{IO}y}) = f(P_{AP_i},\mathrm{IO}_{\mathrm{CIO}_l},\mathrm{MDS}_{\mathrm{AMDS}_k},\mathrm{BufC}_{AC_j},P_a,M_b,O_c).$$

$$\mathrm{HMDS}_{\mathrm{AIO}_l}(\mathrm{Addr}_{\mathrm{MDS}x},\mathrm{Count}_{\mathrm{MDS}y}) = g(P_{AP_i},\mathrm{IO}_{\mathrm{CIO}_l},\mathrm{MDS}_{\mathrm{AMDS}_k},\mathrm{BufC}_{AC_j},P_a,M_b,O_c).$$

具体的实现方法是把控制器中设备地址映像到缓存地址改变为映像到连接主机接口的地址空间，建立数据流的直接流向. 这样可以把设备接口和主机接口的数据空间通过数据链连接起来，实现了物理存储设备和服务器(包括 MDS 和 I/O 服务器)间的联系. 直通路是提高存储系统性能的方法之一. 图 4 针对系统结构特点，建立存储系统数据流和控制流模型，描述了 I/O 请求路径和I/O

响应数据传输路径.

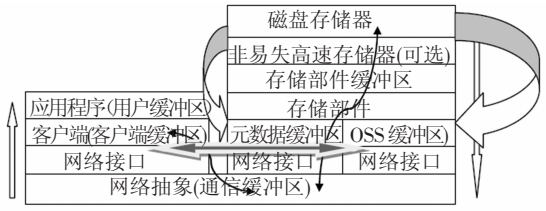


图 4 处理数据流

图 4 中左侧为计算节点(PN,Client)发出 I/O 请求,右侧为 n 个 OSS 的 I/O 节点(存储服务器 OSS)和 MDS 及所挂存储部件,执行并行 I/O 服务请求. 以一次文件读请求为例. 一次文件读请求服务时间 T 包括:I/O 请求在 Client 和服务器端的传输及服务延时,分别表示为 T_{Client} 和 T_{OSS}、响应数据传输延时 $T_{\text{IO}_{\text{data}}}$、并行 IO 处理开销 T_{Parallel} 及 I/O 请求的网络传输服务时间 T_{Network}. 从图 4 中可看出采用直通路技术前后数据传输路径有较大差异. 采用直通路技术前分布文件系统服务一次文件读请求,未命中缓存情况下读响应数据要经过下列传输路径:

1)以 DMA 方式从磁盘组传输到非易失高速存储器可选,再到磁盘阵列控制器的缓存,记为 $T_{\text{DMA}_{\text{disk}}}$;

2)以 DMA 方式从磁盘阵列控制器的缓存传输到存储服务器的缓存,记为 $T_{\text{DMA}_{\text{IO}}}$;

3)从存储服务器缓存拷贝到通讯缓冲区,记为 T_{COPY_1};

4)以 RDMA 方式从存储服务器的通讯缓冲区,经过存储互连网,传输到发出请求的 Client 的通讯缓冲区,记为 T_{RDMA};

5)从 Client 通讯缓冲区拷贝到缓存,记为 T_{COPY_2};

6)从 Client 缓存拷贝到用户缓冲区,记为 T_{COPY_3}. 共计 3 次内存拷贝,读响应数据传输延时 $T_{\text{IO}_{\text{data}}}$ 为

$$T_{\text{IO}_{\text{data}}} = T_{\text{DMA}_{\text{disk}}} + T_{\text{DMA}_{\text{IO}}} + T_{\text{COPY}_1} + T_{\text{RDMA}} + T_{\text{COPY}_2} + T_{\text{COPY}_3}.$$

不考虑 Client 发送 I/O 请求的开销,则 1 次文件读请求服务时间 T_1 为

$$T_1 = T_{\text{DMA}_{\text{disk}}} + T_{\text{DMA}_{\text{IO}}} + T_{\text{COPY}_1} + T_{\text{RDMA}} + T_{\text{COPY}_2} + T_{\text{COPY}_3} + T_{\text{Network}} + T_{\text{Parallel}}.$$

而整个直通路过程应包含:以直通路方式从磁盘组,经过磁盘阵列控制器到存储服务器记为 $T_{\text{PDMA}_{\text{disk}}}$;其余相同.

读响应数据传输延时 $T_{\text{IO}_{\text{data}}}$ 为

$$T_{\text{IO}_{\text{data}}} = T_{\text{PDMA}_{\text{disk}}} + T_{\text{COPY}_1} + T_{\text{RDMA}} + T_{\text{COPY}_2} + T_{\text{COPY}_3}.$$

相应的一次文件读请求服务时间 T_2 为

$$T_2 = T_{\text{PDMA}_{\text{disk}}} + T_{\text{COPY}_1} + T_{\text{RDMA}} + T_{\text{COPY}_2} + T_{\text{COPY}_3} + T_{\text{Network}} + T_{\text{Parallel}}.$$

因此采用直通路策略后,1 次文件读请求减少的服务时间为

$$T_{\text{PDMA}_{\text{disk}}} - (T_{\text{DMA}_{\text{disk}}} + T_{\text{DMA}_{\text{IO}}}).$$

文件读请求的处理涉及从磁盘阵列控制器到本地 I/O 内存的 DMA 写和从本地内存到 Client 内存的 RDMA 写过程.

元数据的过程类似,可参照不重复比较.

经过抽象,可以把上述问题简化为如图 5 所示 2 张图,表示在一个控制器内部数据的流动方向.

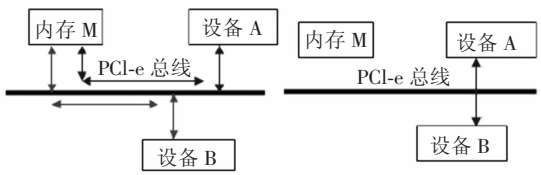


图 5 两种传输模式

其工作原理是内存 M 与设备 A 或 B 存在映射为

$$y_{\text{M}}(\text{Addr}_{\text{device\_Buffer}}, \text{Count}_{\text{device\_Buffer}}) = f(\text{Addr}_{\text{device}}, \text{Count}_{\text{device}}, x), (x \in \{\text{A}, \text{B}\}).$$

本文可以通过硬件方式进行地址空间(如存储器空间或 I/O 空间)映射,产生另外一个映射,能够使设备 A 与设备 B 之间存在映射为

$$y_x(\text{Addr}_{\text{Host\_Mem}}, \text{Count}_{\text{Hoste\_Memr}}) = f(\text{Addr}_{\text{device}}, \text{Count}_{\text{device}}, \text{A}), (x \in \{\text{A}, \text{B}\}).$$

3 新型存储部件访问策略(Pass-through)性能分析测试

在直通路(Pass-through)传输模式下,设备 A 和设备 B 的数据交换可以通过 PCI-e 总线直接进行,此时存储器已被旁路;通过逻辑设置设备的主/从方和正确的寻址方式完成以上操作. 这种传输方式提高了数据的传输效率,以不占用系统内部总线为前提,减少系统开销.

假设设备 i 传到设备 j 的数据量为 Data_{ij},耗时为 T_{ij},其平均传输率为 v_{ij},则

$$T_{ij} = T_{ij_1} + T_{ij_2} + T_{ij_3} + T_{ij_4} + T_{ij_5}.$$

假设 T_{ij_1}、T_{ij_2}、T_{ij_3}、T_{ij_4}、T_{ij_5} 中每次交易中 T_{ij_1}、T_{ij_2}、T_{ij_3}、T_{ij_5} 是常量,且随每次数据交换量的不同而不同,设其突发数传率为 C,则

$$C = \text{Data}_{ij}/T_{ij3},$$

$$v_{ij} = \text{Data}_{ij}/T_{ij}.$$

式中:T_{ij_1} 为逻辑地址映射分配;T_{ij_2} 为仲裁选择;

T_{ij_3} 为建立连接；T_{ij_4} 为数据传输；T_{ij_5} 为撤消连接. 直通路方式，其设备 A 到设备 B 传送交换量为 D 数据量的数传率为

$$v_{\text{Pass}=\text{AB}} = \frac{\text{Data}_{\text{AB}}}{T_{\text{AB}}} = \frac{D}{T_{\text{AB}}}.$$

而传统方式下数传率为

$$v_{t-\text{AB}} = \frac{\text{Data}_{\text{AB}}}{T_{\text{AM}} + T_{\text{MB}}}.$$

又设 T_{ij_1} 为 $10t$，T_{ij_2} 为 $5t$，T_{ij_3} 为 $5t$，T_{ij_5} 为 $5t$，其直通路 $v_{\text{pass-AB}}$ 和传统方式下的 $v_{t-\text{AB}}$ 加速比 A 为

$$A = \frac{v_{\text{pass-AB}}}{v_{t-\text{AB}}}. \tag{4}$$

则将对应参数代入式(4)中，得到

$$A = \frac{2D + 50tC}{D + 25tC} = 2.$$

但是由于存储器与磁盘介质传送数据，不确定的是寻道时间和缓存策略处理时间，因此其加速比仅为

$$A = x + 1, 0 < x \leqslant 1.$$

存储系统性能评测程序也可获取一部分 I/O 的特征信息[9-10]. 存储系统测试通常采用一些基准应用，如 Postmark、IOzone、IOmeter、Bonnie 等工具[11-17]. 本文使用 IOmeter、Bonnie 基准程序以及用 Linux 的基本命令 dd 等编写的标准测试脚本来测试系统的性能.

3.1　系统实验环境

根据存储部件直通路(Passthrough)研究的设计思路和具体实现完成关于直通路方式和正常传输方式下的性能测试. 以服务器(Linux 平台)挂接存储部件为例主机接口协议为 IBA 的 SRP 协议，底层是 SCSI 协议. 设备接口是 SAS 接口，挂接 15 000 r/min 的企业级 300 GB SAS 磁盘.

3.2　性能测试

用标准测试及其脚本对上述环境进行读写测试. 图 6 为存储部件以直通路方式、传统方式 0 命中和传统方式全命中进行数据传送，交换量从 512 Byte/(MB · s^{-1}) ~1 024 KB/(MB · s^{-1}). 如图 6 及表 2 所示.

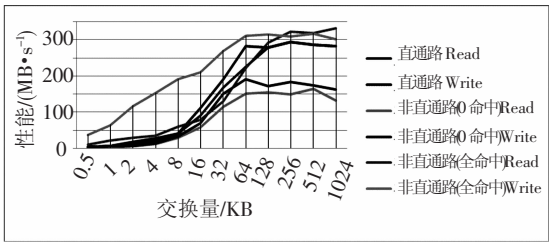


图 6　直通路方式与传统方式的对比

表 2　部分存储部件性能测试列表

交换量/KB	直通路		非直通路(0 命中)		非直通路(全命中)	
	Read (v_{Tread})	Write (v_{Twrite})	Read ($v_{\text{NTread}-0}$)	Write($v_{\text{NTwrite}-0}$)	Read($v_{\text{NTread}-100}$)	Write($v_{\text{NTwrite}-100}$)
0.5	12.93	13.75	0.97	1.92	10.25	37.29
1	26.51	36.49	3.54	5.21	22.16	64.62
2	53.49	67.25	7.01	6.49	29.93	117.35
4	70.17	85.96	12.62	13.65	35.52	152.21
8	100.77	110.13	30.08	32.94	60.41	190.70
16	160.32	180.40	57.58	69.17	78.70	210.37
32	217.50	200.28	113.41	150.13	130.39	269.05
64	284.53	280.91	150.60	190.48	220.85	309.22
128	357.32	295.32	153.52	170.53	290.32	313.71
256	292.48	292.48	147.90	182.25	320.57	307.39
512	285.27	285.27	163.27	173.93	317.36	315.73
1 024	281.35	281.35	130.40	160.96	330.92	301.14

假设直通路与非直通路(全命中)相比，因为数据全部在缓存或者暂时存放在缓存，后者性能比前者高，性能损失直通路读和非直通路(全命中)数据相近. 直通路(全命中)和非直通路(0 命中)是两个极端，两个理想状态便于测试，实际应用介于两者之间. 其他值可根据实际情况测试和验证，但是比较复杂，不能很好地收集缓存的真实情况，所以才利用最极端的两种情况说明问题.

在实际应用中计算节点两组系统采用不同策略进行性能比较.

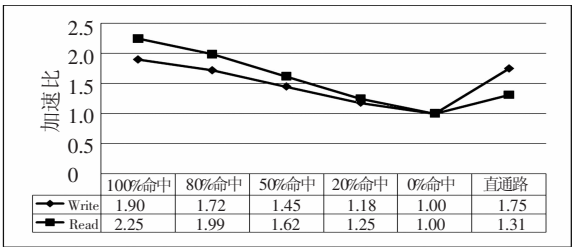
1)系统实验环境.

系统试验的目的是对比虚根文件系统和局部文件系统的性能测试情况. 本系统中以整机系统有 2 000 个计算节点为例，原始设计中考虑每个计算节点有自己的高速计算网络接口、千兆以太网口、维护接口、15 000 r/min 的企业级 300 GB SCSI 磁盘等. 硬盘用于本地局部 OS 启动以及装载系统 Client 自身的文件系统. 利用虚根文件系统管理虚拟空间再次生成的每个计算节点所需要的逻辑空间，作为其系统的 OS、交换区的 swap 空间及其局部空间.

2)性能测试.

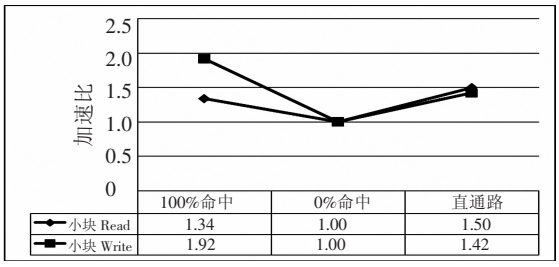
图 7 ~ 图 9 表示对虚拟逻辑磁盘和真实物理磁盘的性能测试比较，采用标准测试程序和脚本.

总体表明前者的性能是后者性能的3倍.



	100%命中	80%命中	50%命中	20%命中	0%命中	直通路
Write	1.90	1.72	1.45	1.18	1.00	1.75
Read	2.25	1.99	1.62	1.25	1.00	1.31

图7 几种方式的性能加速比



	100%命中	0%命中	直通路
小块 Read	1.34	1.00	1.50
小块 Write	1.92	1.00	1.42

图8 小块方式的性能加速比

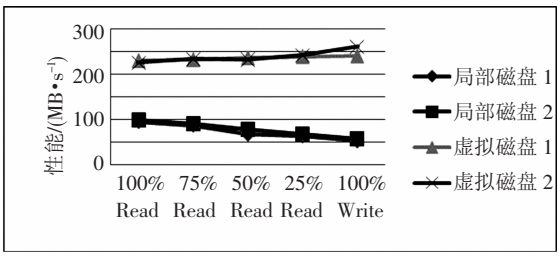


图9 实际系统验证对比

通过上述应用环境下的模拟可以得到系统具备有局部盘不可能有的优点:加载时间快、I/O性能提高、可靠性增强、系统易管理和利用空间充分等.

4 结 论

1)直通路策略很好地解决了设备间的传输效率问题,在虚拟存储文件系统的小块不命中的元数据存储过程中能够提高元数据的获取效率和处理能力.

2)由于还有软件开销和不同数据流在缓存算法中的应用延时不同,没有精确的对比.在主机接口性能一致的前提下,分析传统存储部件传输机制上的问题,研究数据流存储操作的方式,为最大限度满足系统存储性能要求,分析其对传统传输方式的加速比.

3)综合利用存储策略在实际系统平台上进行了测试和性能对比.该方法是提高分布式海量存储部件级性能的有效策略,可以结合其他方法综合提高系统性能.

参考文献:

[1] TOP 500 Supercomputer Sites. TOP500 List for June 2011 [EB/OL]. [2011-06-01]. http://www.top500.org.

[2] PATTERSON D A, GIBSON G, KATZ R. A case for redundant arrays inexpensive disks (RAID)[J]. ACM SIGMOD Conference, 1988, 17(3):109-116.

[3] PATTERSON D A, HENNESSEY J L. Computer Organization and Design: The Hardware/software Interface [M]. San Francisco, CA: Morgan Kaufmann,1998.

[4] RIPEANU M, IAMNITCHI A. S4: a simple storage service for sciences[C]//Proceedings of the 16th IEEE International Symposium on High Performance Distributed Computing (HPDC). Monterey Bay, CA: Hot Topics Track, 2007.

[5] HWANG Kai, XU Zhiwei. Scalable Parallel Computing Technology, Architecture Programming[M]. [S. n.]: McGraw-Hill, 1998.

[6] SALEM K, GARCIA-MOLINA H. Disk striping[C]//Proceedings of 2nd IEEE International Conference on Data Engineering. Washington, DC: IEEE, 1986: 336-342.

[7] CABRERA L, LONG D D E. Swift: using distributed disk striping to provide high i/o data rates[R]. Santa Cruz, CA: University of California at Santa Cruz, 1991.

[8] KIM M Y. Synchronized disk interleaving[J]. IEEE Transactions on Computers, 1986, 35(11): 978-988.

[9] KATCHER J. PostMark: A new file system benchmark. [EB/OL]. http://www.netapp.com//technology/level3/3022.html.

[10] BRYANT R, RADDAZ D, SUNSHINE R. PenguinoMeter: a new fileIO benchmark for linux®[C]//Proceedings of the 5th Annual Linux Showcase & Conference. Berkeley, CA: USENIX Association, 2001: 5-10.

[11] Network Appliance. PostMark: a new file system benchmark [EB/OL]. [1997-10-08]http://www.netapp.com.

[12] TIM Bray. The bonnie benchmark[EB/OL]. http://www.text-uality.com.

[13] NORCOTT W, CAPPS Don. IOzone filesystem benchmark[EB/OL]. http://www.iozone.org.

[14] Intel Corporation. Iometer[EB/OL]. http://www.iometer.org.

[15] I/O Performance Inc. Xdd[EB/OL]. http://www.ioperformance.com.

[16] TRAEGER A, ZADOK E, JOUKOV N, *et al.* A nine year study of file system and storage benchmarking[J]. ACM Transactions on Storage, 2008, 4(2): 5-56.

[17] Intel Corp. IOMETER user guide [EB/OL]. www.intel.Com/developer/iometer.etc.

(编辑 张 红)