软件学报 ISSN 1000-9825, CODEN RUXUEW
Journal of Software,2011,22(9):2206−2221 [doi: 10.3724/SP.J.1001.2011.03906]

E-mail: jos@iscas.ac.cn
http://www.jos.org.cn
Tel/Fax: +86-10-62562563

面向并行文件系统的性能评估及相对预测模型*

赵铁柱[1,2], 董守斌[1+], Verdi MARCH[3,4], Simon SEE[3,5]

[1](华南理工大学 广东省计算机网络重点实验室,广东 广州 510641)

[2](五邑大学 计算机学院,广东 江门 529020)

[3](Technical and Cloud Computing Center, Oracle Corporation, Singapore)

[4](Department of Computer Science, National University of Singapore, Singapore)

[5](Department of Mechanical & Aerospace Engineering, Nanyang Technological University, Singapore)

Performance Evaluation and Relative Predictive Model of Parallel File System

ZHAO Tie-Zhu[1,2], DONG Shou-Bin[1+], Verdi MARCH[3,4], Simon SEE[3,5]

[1](Guangdong Key Laboratory of Computer Network, South China University of Technology, Guangzhou 510641, China)

[2](School of Computer Science, Wuyi University, Jiangmen 529020, China)

[3](Technical and Cloud Computing Center, Oracle Corporation, Singapore)

[4](Department of Computer Science, National University of Singapore, Singapore)

[5](Department of Mechanical & Aerospace Engineering, Nanyang Technological University, Singapore)

+ Corresponding author: E-mail: zhao.kevin2008@gmail.com

Zhao TZ, Dong SB, March V, See S. Performance evaluation and relative predictive model of parallel file system. *Journal of Software*, 2011,22(9):2206−2221 . http://www.jos.org.cn/1000-9825/3906.htm

Abstract: In this paper, the performance evaluation and modeling of parallel file system based on Lustre file system is studied. After performing a survey on performance factors, a series of performance evaluations via experimental approaches and propose a performance relational model (PRModel). In the experimental and PRModel analysis, it is found that different performance factors have closed performance correlations. In order to mime the relational information, a novel relative performance predictive model (RPPModel) is proposed. This model can be used to predict the overhead over different performance factors. The model is validated through a series of experiments over a variety of performance factors. The experimental results show that the average relative errors results can be controlled within 17%~28%. This model is easy to use and can obtain better prediction accuracy.
Key words: parallel file system; performance evaluation; performance model; Lustre file system

摘 要: 基于 Lustre 文件系统,对并行文件系统的性能评估和性能建模进行了研究.通过对性能因子的调研,进行了一系列性能评估实验,并提出性能相关性模型(PRModel).在实验评估和 PRModel 分析中发现,在不同的性能因子之间存在着紧密的性能相关性.为了挖掘并利用这种相关性信息,提出了一种相对性能预测模型(RPPModel)来预测

* 基金项目: 国家自然科学基金(10805019); 国家重点基础研究发展计划(973)(2009CB320505); 国家发改委 CNGI 项目(CNGI2008-109/122)

收稿时间: 2010-03-25; 定稿时间: 2010-06-29

CNKI 网络优先出版: 2010-12-22 17:11, http://www.cnki.net/kcms/detail/11.2560.TP.20101222.1711.000.html

不同性能因子条件下的性能.为了验证 RPPModel 的有效性,设计了大量实验用例.结果表明,预测结果的平均相对误差能够控制在 17%~28%的范围内,易于使用且具有较好的预测准确度.

关键词: 并行文件系统;性能评估;性能模型;Lustre 文件系统

中图法分类号: TP316 **文献标识码**: A

并行文件系统具有高吞吐量、高带宽以及高可扩展性等特点,是高性能计算平台存储系统的首选.作为高性能计算(HPC)系统的核心组成部分,并行文件系统的性能直接影响着整个系统的性能.然而,构建一个并行文件系统是非常复杂且昂贵的,尤其当系统没有合适的配置时,系统的性能不能充分地挖掘出来,成本就更高了.如何评估性能,优化设计和预测性能成为当前存储工业界和学术界的研究热点.

目前,并行文件系统的研究主要集中在以下几个方面:

(1) 元数据的管理和查询优化,如 Wang 等人提出二级元数据管理方法以提高并行文件系统的可用性[1];

(2) 性能参数分析和调优,如 Yu 等人通过实验的方法对并行 I/O 进行了参数分析、配置和优化[2],同时提出写操作分块和层次条带化的方法来提高并行文件系统的 I/O 性能[3];Li 等人利用 Fork-Join 排队模型对整个存储系统框架进行模拟,同时提出一种参数分析的方法[4];

(3) 优化数据分布策略,Yu 等人发现,文件分布模型对聚合 I/O 带宽有着显著的影响,提出一种基于用户视角的数据分布策略[5];

(4) 优化数据访问路径,如 Piemas 等人提出了基于用户空间的 Lustre 并文件系统活动存储实现方案,优化数据访问路径提高系统的 I/O 性能[6];

(5) 可用性和可扩展性,如 Zhang 等人设计了一种逻辑镜像环(LMR)的部署机制,可以显著地提高并行文件系统的可靠性和可扩展性[7];

(6) 性能评估和建模,并行文件系统的性能评估大都是通过实验的方法进行,如 Yu 等人对 Cay XT 的并行 I/O 性能进行了有效的评估[8],Shan 等人通过 IOR 基准对 Lustre 文件系统的性能进行了评估和分析[9].

由于并行文件系统结构复杂且影响性能的因素错综复杂,目前还没有一个适合的性能预测模型.通过对大量的数据库模型、存储网络模型和 RAID 模型进行分析和借鉴,如文献[4,10−16],可以将目前的存储模型大致可以分为两种:白盒模型和黑盒模型.白盒模型通常需要找出应用程序负载特征(输入)和性能(输出)之间的定量函数关系,常见的负载特征包括读/写请求大小、读/写请求比率、请示队列长度、顺序-随机请求比率、I/O 到达时间隔、访问的突发性、时空相关性等,常见性能指标包括带宽、吞吐量、延迟等.然而,负载特征和性能输出之间的关系很复杂的,很难给出明确的函数表达.黑盒模型把整个系统看成一个黑盒子,不考虑系统的内在逻辑,避开去寻找负载特征和性能输出之间的定量关系,而是通过概率统计、机器学习等方法去预测性能输出.目前,黑盒模型因自身的特点,越来越受到人们的关注,尤其是应用在一些复杂存储系统中.现有模型主要有以下几类:分析模型[10,11,17−19]、基于机器学习的模型[20,21]、相对适合度模型[22,23]、基于表的模型[24]等.先前的主要研究工作如文献[18,20,21,24],是在一个给定的存储系统中预测具体工作负载的性能,以负载特征作为输入,性能指标作为输出.这些模型主要存在以下 3 方面的问题:

(1) 不能捕捉到应用程序和存储系统之间的反馈信息.通常来说,应用程序的性能依赖于存储性能,当存储性能升高或降低时,应用程序的 I/O 到达率也会相应产生变化;

(2) 应用程序的负载特征很难获得或精确的表示;

(3) 仅依赖少数的负载特征来描述一个复杂的负载是很难的,通常会带来一些重要信息的丢失.

Lustre 作为典型的基于对象的并行文件系统,已被广泛应用于 TOP500 的超级计算机中.世界排名前 100 的 HPC 集群中,就有超过 40%使用 Lustre 作为全局文件系统.Lustre 能够同时拥有上万个客户端、Peta-scale 的存储容量、几百 MB 的 I/O 吞吐量[25].Lustre 同当前其他主流的并行/分布式文件系统如 NFS,GFS,PVFS,GPFS,XFS,SFS,Coda 等相比,在设计上有以下 3 个特点[26−28]:

(1) 数据独立存储.Lustre 将文件的元数据(文件属性和状态信息)和文件的真实数据分别存储,真实数据

被条带化成数据块存储在多个 I/O 节点,元数据服务器提供单一的命名空间和目录结构,从而提高系统的聚合 I/O 带宽,提高并发度,扩大文件的空间,并使系统具有较好的扩展性;

(2) 基于对象的存储,将文件内容存储在对象中,单个文件内容可以划分到多个对象中,存储在一个或多少对象存储目标(OST)中.这种方式能够显著地提高系统的灵活性、可靠性和可扩展性;

(3) 针对大文件读写进行优化,可以提供高性能 I/O、服务和网络失效的快速恢复和分布式意向锁管理.

基于以上的分析可知,并行文件系统的性能评估和性能建模是一个具有挑战性且富有研究价值的工作,同时,Lustre 是一个典型的基于对象的并行文件系统,已广泛部署在许多 HPC 系统中.Lustre 的性能研究极具代表性,本文展开了 Lustre 并行文件系统的性能评估和性能建模的研究,主要工作有 3 个方面:

(1) 对 Lustre 的性能影响因子进行了全面的调研,并在此基础上进行了一系列实验性能评估,发现不同因子下的性能有着相似的发展趋势;

(2) 提出性能相关性模型(PRModel),对不同性能影响因子下的性能进行相关性分析,发现不同因子下的性能具有紧密的相关性;

(3) 为了挖掘不同因子下的性能相关性信息,提出了相对性能预测模型(RPPModel),通过对两个不同配置的 Lustre 系统进行训练来预测系统性能,保持较好的预测精度.本文提出的性能相关性模型(PRModel)和相对性能预测模型(RPPModel)对其他大中型分布式/并行文件系统如 PVFS,pNFS 等的性能研究与预测有较好的适用性和借鉴价值.

本文第 1 节对 Lustre 的基本架构和性能因子进行详细的调研,并在此基础上进行了模型设计及分析,提出性能相关性模型(PRModel)和相对性能预测模型(RPPModel).第 2 节进行性能评估实验、性能相关性分析和相对性能预测分析,进一步验证模型的可行性.第 3 节对本文的工作进行总结.

1 性能模型设计及分析

1.1 Lustre架构及性能因子

在性能评估实验和引出模型之前,首先对 Lustre 的基本构架和影响 Lustre 文件系统的性能因子进行了详细调研,为后续的性能评估和模型的构建打下基础.

Lustre 是一个基于对象的并行文件系统,通过将文件条带化存储到多个存储设备来获取高的聚合 I/O 带宽,并提供通过分布式锁管理来提供细粒度的并行文件服务.Lustre 主要包括 5 个主要部分:元数据服务器(MDS)管理元数据,并提供全局一致的命名空间;元数据目标(MDT)存储元数据(如文件名、目录、权限和文件层次结构);对象存储服务器(OSS)提供文件 I/O 服务,处理网络请求;对象存储目标(OST)存储文件实际数据,为了优化性能,通常将一个文件分散到多个 OST 中,逻辑对象卷(LOV)管理多个 OST 上的文件条带;客户端(client)是计算节点,它包括多个 Linux 虚拟文件系统(LVFS)和 Lustre 服务器之间的接口如元数据客户端(MDC)、对象存储客户端(OSC)、管理客户端(MGC).Lustre 的详细介绍可见文献[25,28].

通过参阅大量的文献[2,5, 8,25,29,30−32],将影响性能的因子归类为:OSS 数目、MDS 数量、OSS 服务线程数量、MDS 服务线程数量、日志类型(OST)、磁盘类型(OST)、存储连接方式、条带化模式(包括条带大小、条带数量、条带化初始位置等)、条带化算法、读/写 Cache 大小、inode 的数量和大小(OST/MDT)、数据分布策略等因素.图 1 表示了 Lustre 的基本架构及每个组成部件上的详细性能因子,包括 OSS,OST,MDS,MDT,客户端,高速互联网络,存储连接网络上的性能因子.

Lustre 通过将文件条带化存储到多个 OST 获取高的聚合 I/O 带宽,Lustre 的文件 I/O 操作的流程如图 2 所示,详细介绍见文献[25–27].

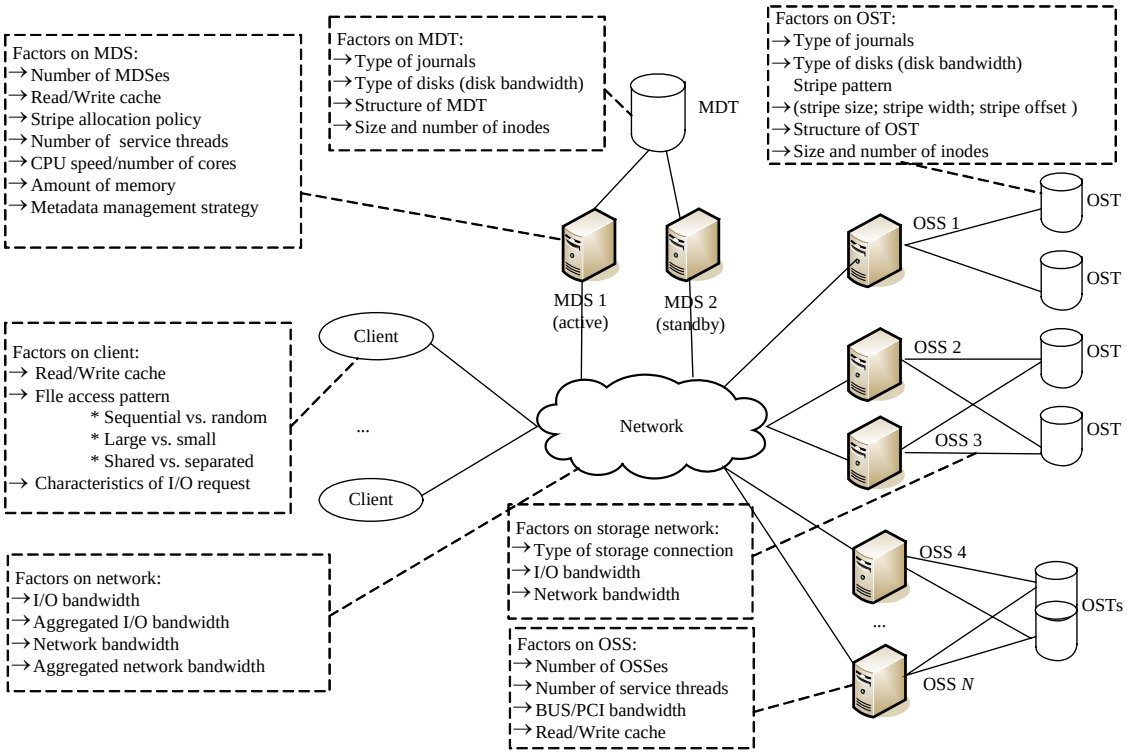

Fig.1 Performance factors and basic architecture of Lustre system

图 1 Lustre 系统的性能因子和基本架构

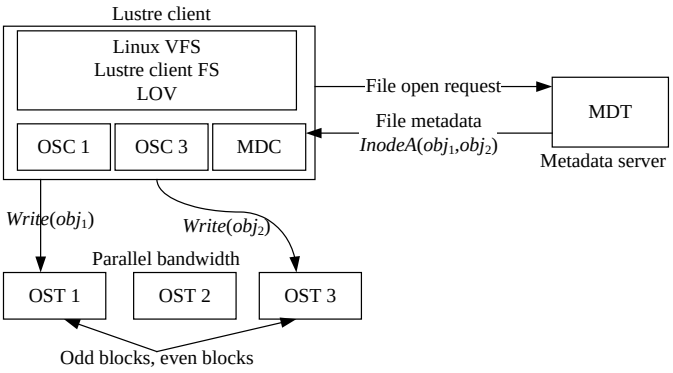

Fig.2 File open and file I/O in Lustre file system

图 2 Lustre 文件系统中文件打开和文件 I/O

在后续的性能评估和模型中,我们主要考虑几个重要性能因子,包括 OSS 数量、日志类型、磁盘类型、OSS 服务线程数量及存储连接方式.

1.2 性能相关性模型PRModel

本节描述性能相关性模型(PRModel),我们用 PRModel 来分析不同性能因子下的性能相关性,为后面的性能评估和相对性能预测模型的构建奠定基础.

假设 $Lustre_i(i=0,1,2,\ldots,m)$:具有系统配置 i 的 Lustre 系统.给定应用的负载,假设 $X_0=(x_0(1),x_0(2),\ldots,x_0(n))$是

$Lustre_0$ 性能指标的特征序列,$X_1=(x_1(1),x_1(2),\ldots,x_1(n))$,$X_2=(x_2(1),x_2(2),\ldots,x_2(n))$,$X_m=(x_m(1),x_m(2),\ldots,x_m(n))$是分别对应 $Lustre_1,Lustre_2,\ldots,Lustre_m$ 性能指标的特征序列,对于给定的实数$\gamma(x_0(k),x_i(k))$,定义:

$$\gamma(X_0,X_i)=\frac{1}{n}\sum_{k=1}^{n}\gamma(x_0(k),x_i(k)) \tag{1}$$

其中, $\gamma(x_0(k),x_i(k))=\dfrac{\min\limits_i\min\limits_k|x_0(k)-x_i(k)|+\rho\max\limits_i\max\limits_k|x_0(k)-x_i(k)|}{|x_0(k)-x_i(k)|+\rho\max\limits_i\max\limits_k|x_0(k)-x_i(k)|}$.

如果$\gamma(X_0,X_i)$满足规范性、整体性、偶对称性、接近性,我们称$\gamma(X_0,X_i)$是特征序列 X_0 和 X_i 的相关度,$\gamma(x_0(k),x_i(k))$是特征序列 X_0 和 X_i 在点 k 的相关性系数,$\rho\in[0,1]$称为分辨系数.关于灰色关联 4 个公理及关联分析的详细介绍见文献[30,33−35].

PRModel 借鉴了灰色关联度分析的主要思想,是一种因素分析方法,通过对系统性能指标的特征序列几何关系的比较来分析多性能指标的特征序列间的关联程度.即认为刻画性能指标所表示的曲线的几何形状越接近,则性能指标发展变化态势越接近,因而他们之间的关联程度越大.如果两个不同配置的 Lustre 系统的性能指标有着较大的相关度值,从模型的几何意义可以看出,这两个不同配置的 Lustre 系统的性能指标具有相似的变化趋势,从而为后面的相对性能模型的引出打定基础.

1.3 相对性能预测模型RPPModel

1.3.1 模型假设

根据 Lustre 并行文件系统的特点,给出以下两点假设:

假设 1. 对于给定的应用负载,不同的 Lustre 系统会产生不同的负载特征.

假设 2. 综合资源使用率、服务器和存储目标的使用率和性能指标能够提高性能预测的准确度.

通常来说,应用程序和存储系统之间存在着信息的反馈,应用程序的性能依赖于存储性能.例如,当存储性能升高或降低时,应用程序的 I/O 到达率也会相应产生变化.所以,同一个应用负载在不同的 Lustre 系统中会产生不同的负载特征.由于应用程序的负载特征很难获得或精确地表示,即使用少数的负载特征来描述一个复杂的负载也会带来一些重要信息的丢失.在 Lustre 文件系统中,资源的使用率尤其是服务器和存储目标的使用率对 Lustre 系统性能有着很大的影响.如果把资源使用率、服务器和存储目标的使用率、性能指标等特征和负载特征融合在一起,就能捕获更多的有用信息,从而提高性能预测的准确度.从分析可以看出,以上两点假设是合理的.

1.3.2 模型参数

以下给出模型中几个重要参数的定义:

- P_i:$Lustre_i$ 的性能指标向量,性能指标包括带宽,吞吐量和延迟.

$$P_i=\left[Bandwidth \quad Throughput \quad Latency\right]^T.$$

- W_i:$Lustre_i$ 的负载特征向量,主要包括 *Threads*/*OST*(*THREAD*),*Objects*/*OST*(*OBJ*),读/写请求大小(*RS*),读-写请求比率(*RWR*),请示队列长度,顺序-随机请求比率,I/O 到达时间隔,访问的突发性,时空相关性等[36,37].

$$W_i=\left[THREAD_i \quad OBJ_i \quad RS_i \quad RWR_i \quad ...\right]^T.$$

- $RUtil_i$:$Lustre_i$ 的资源使用率,包括 CPU 使用率(*C*)、内存使用率(*M*)、存储空间使用率(*D*)、cache 命中率等.

$$RUtil_i=\left[C_i \quad M_i \quad D_i \quad ...\right]^T.$$

- $STUtil_i$:$Lustre_i$ 的服务器&存储目标使用率,包括 OSS 使用率(*OSS*)、OST 使用率(*OST*)、MDS 使用率(*MDS*)、MDT 使用率(*MDT*).

$$STUtil_i=\left[OSS_i \quad OST_i \quad MDS_i \quad MDT_i\right]^T.$$

1.3.3 相对性能预测模型构建

(1) 构造相对性能模型

构建相对性能模型的主要思想是通过对一对不同配置的 Lustre 系统进行训练,如采用一些机器学习的算法进行训练,去捕捉不同的 Lustre 系统的潜在相关性信息,并学习预测不同 Lustre 系统的性能比例因子,如图 3 所示.

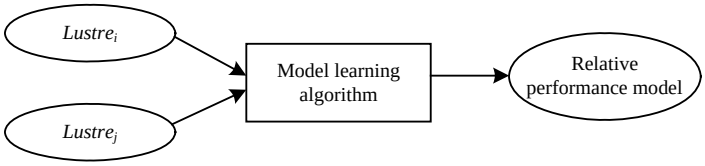

Fig.3 Construct relative performance model

图 3 构建相对性能模型

对于给定的 $Lustre_j$,我们采集应用程序在 $Lustre_j$ 上的数据,训练出负载特征 W_j 和性能指标 P_j 之间的函数关系 ϕ_j:

$$P_j=\phi_j(W_j) \tag{2}$$

为了捕捉到从 $Lustre_i$ 到 $Lustre_j$ 的负载特征信息的变化,定义函数 $\varphi_{i\to j}$ 来表示 W_i 和 W_j 之间的关系:

$$W_j=\varphi_{i\to j}(W_i) \tag{3}$$

组合公式(2)和公式(3),可得:

$$P_j=\phi_j(\varphi_{i\to j}(W_i)) \tag{4}$$

除了负载特征,性能指标、资源利用率、服务器和存储目标的使用率能够提高预测准确度(假设 2).也就是说, $Lustre_i$ 上的性能指标、资源利用率、服务器和存储目标的使用率能够用于对 $Lustre_j$ 的性能预测.

$$P_j=\phi_j(\varphi_{i\to j}(W_i,P_i,RUtil_i,STUtil_i)) \tag{5}$$

把 ϕ_j 和 $\varphi_{i\to j}$ 组合函数表示成单一的函数 $CF_{i\to j}$:

$$P_j=CF_{i\to j}(W_i,P_i,RUtil_i,STUtil_i) \tag{6}$$

为了提高模型的适用性,使用模型对一些新的应用负载也能进行预测,将预测 P_j 改成预测两个不同 *Lustre* 系统的性能比 P_j/P_i 可能显著提高模型的可用性.

$$P_j/P_i=RF_{i\to j}(W_i,P_i,RUtil_i,STUtil_i) \tag{7}$$

(2) 应用模型预测性能

通过训练相对性能模型,就可能通过 $Lustre_i$ 上的性能指标 P_i、资源利用率 $RUtil_i$、服务器和存储目标的使用率 $STUtil_i$ 对 $Lustre_j$ 的性能 P_j 进行预测,如图 4 所示.

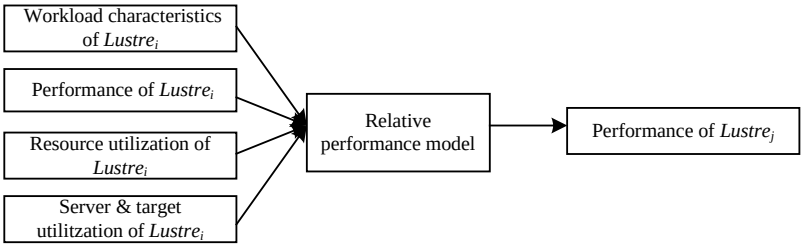

Fig.4 Predict performance

图 4 预测性能

通过解公式(7),可以得到性能预测公式:

$$P_j=RF_{i\to j}(W_i,P_i,RUtil_i,STUtil_i)\times P_i \tag{8}$$

1.3.4 模型传递性质

在模型的实际应用中不难发现以下应用场景:当需要通过训练 $Lustre_i$ 到 $Lustre_j$ 的相对性能预测模型 RPPModel 时,由于缺乏实验条件或其他因素,很难直接训练出从 $Lustre_i$ 到 $Lustre_j$ 的 RPPModel,而 $Lustre_i$, $Lustre_{k1}$,…,$Lustre_{kn}$ 间则相对容易训练,这时可以通过对 $Lustre_i$,$Lustre_{k1}$,…,$Lustre_{kn}$,$Lustre_j$ 进行一系列训练间接地获得从 $Lustre_i$ 到 $Lustre_j$ 的性能比例因子.

以下我们定义两个算子:串行传递算子⊗和并行传递算子⊕.为了简化表示,定义 $P_i^j = P_j / P_i$.

定义 1(串行传递算子⊗). 假设目标是获得从 $Lustre_i$ 到 $Lustre_j$ 的性能比例因子,可以通过对 $Lustre_{k1}$,…, $Lustre_{kn}$ 的训练间接实现,如图 5 所示,$Lustre_i$,$Lustre_{k1}$,…,$Lustre_{kn}$,$Lustre_j$ 间存在着串行传递关系.

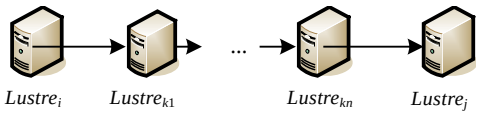

Fig.5 Serial transmission

图 5 串行传递

定义串行传递算子⊗如下:

$$P_i^j = P_i^{k1} \otimes P_{k1}^{k2} \otimes ... \otimes P_{kn}^j = P_i^{k1} \times P_{k1}^{k2} \times ... \times P_{kn}^j .$$

算子⊗的合理性,我们很容易证明.

证明:由定义可知 $P_i^j = P_j / P_i$,同时,

$$P_i^j = P_i^{k1} \otimes P_{k1}^{k2} \otimes ... \otimes P_{kn}^j = P_i^{k1} \times P_{k1}^{k2} \times ... \times P_{kn}^j = \frac{P_{k1}}{P_i} \times \frac{P_{k2}}{P_{k1}} \times ... \times \frac{P_j}{P_{kn}} = \frac{P_j}{P_i}.$$ □

定义 2(并行传递算子⊕). 同样也存在这样的情形,当想获得从 $Lustre_i$ 到 $Lustre_j$ 的性能比例因子时,我们可以从多个中间训练集中获得 $Lustre_i$ 到 $Lustre_j$ 的性能比例因子,即有多条路径能够得到 $Lustre_i$ 到 $Lustre_j$ 的性能比例因子,如图 6 所示.

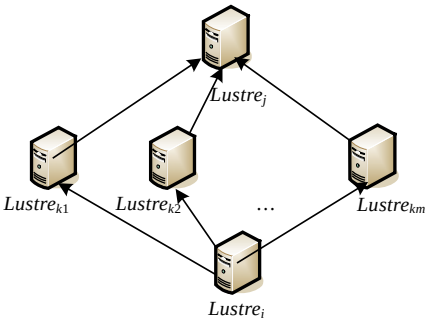

Fig.6 Parallel transmission

图 6 并行传递

定义并行传递算子⊕如下:

$$P_i^j = P_i^{j \leftarrow k1} \oplus P_i^{j \leftarrow k2} \oplus ... \oplus P_i^{j \leftarrow km} = \sum_{l=1}^{m} \frac{\alpha_l}{\sum_{r=1}^{m} \alpha_r} \times P_i^{j \leftarrow kl},\ 0 \leqslant \alpha_l \leqslant 1,\ \sum_{l=1}^{n} \alpha_l = 1,$$

其中, $P_i^{j \leftarrow ki}$ 表示经过 $Lustre_{ki}$ 传递路径,从 $Lustre_i$ 到 $Lustre_j$ 的性能比例因子.由上面的串行传递算子⊗可得:

$$P_i^{j \leftarrow ki} = P_i^{ki} \otimes P_{ki}^j .$$

2 性能评估及模型分析

2.1 性能评估实验

(1) 实验环境(见表 1)

Table 1 Configuration of experiment environment

表 1 实验环境配置

Component	Details
Lustre version	1.6.6
Operating system	Redhat enterprise linux 5.2
Linux kernel	Lustre supported kernel 2.6.18
OSS nodes	2x sun fire X4240
CPU& memory	2 AMD dual core opteron at 3 Ghz with 8 GB memory
Network	10 gigabit TCP/IP ethernet
SAS disks	24x 300GB disks
SATA disks	48x 1TB disks
Benchmark	OBDFilter-Survey

(2) 测试基准工具

OBDFilter-Survey 是 Lustre I/O Kit 里一个重要的测试基准,它能够模拟 Lustre 服务器的对象 I/O 协议. OBDFilter 是 Lustre 对象存储服务器(OSS) I/O 堆栈里重要的组成部分,OBDFilter-Survey 能够模拟创建、删除、读取、写入对象并显示 OSS 性能输出等功能.由于它能够正确地模拟 OSS 上的标准 Lustre I/O,所以被选为实验的基准测试工具.

(3) 实验用例设计

在实验中,主要考查以下 5 个重要的性能因子:OSS 的数量(1OSS vs. 2 OSS)、日志类型(internal journal vs. external journal)、磁盘类型(SAS disk vs. SATA disk)、存储连接方式(directly connected vs. daisy-chain connected)、Threads/OST 数量.我们设计了 4 组实验用例,每组实验用例中考查一个性能因子对输出性能的影响(见表 2).

Table 2 Design of experimental cases

表 2 实验用例设计

Group	Case	No. of OSSes	Type of journals	Storage connection	Type of disks	Test Intent
1	1.1	1	External	Direct	SAS	Test performance under different numbers of OSSes
	1.2	2	External	Direct	SAS	
2	2.1	1	Internal	Direct	SAS	Test performance under different types of journals
	2.2	1	External	Direct	SAS	
3	3.1	2	External	Direct	SAS	Test performance under different types of disks
	3.2	2	External	Direct	SATA	
4	4.1	2	External	Direct	SATA	Test performance under different storage connection approaches
	4.2	2	External	Daisy-Chain	SATA	

在实验中,性能指标选择吞吐量(MB/s).对于每个实验用例,用 OBDFilter-Survey 分别进行了写(W)、重写(ReW)、读(R)等 3 种操作的测试.为了便于观察性能变化的相关性,我们将实验结果进行了规一化处理.实验结果如图 7~图 10 所示,它们分别对应于第 1 组~第 4 组的实验用例测试结果.

从图 7~图 10 不难发现:

(1) 读的性能要明显好于写和重写的性能,尤其当 Threads/OST 的数量是 2,3 和 4 时特别明显,这与文献[8,9]的结果是一致的;

(2) 对于每组实验用例,读-读的性能、写-写的性能和重写-重写的性能都具有相似的发展趋势;同时,写-重写的性能也有相似的发展趋势.这种相似的发展趋势映射到性能相关性模型(PRModel)就表现为较强的性能相关性,这一点可以从性能相关性模型(PRModel)的几何意义中清晰地看到.PRModel 认为,刻画性能指标所表示的曲线几何形状越接近,则性能指标发展变化态势越接近,因而它们之间的

关联程度越大.

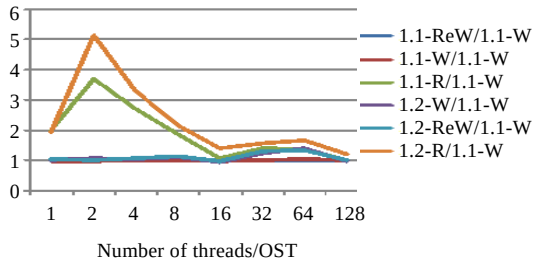

Fig.7 Performance trend under different numbers of OSSes

图 7 不同 OSS 数量下的性能变化趋势

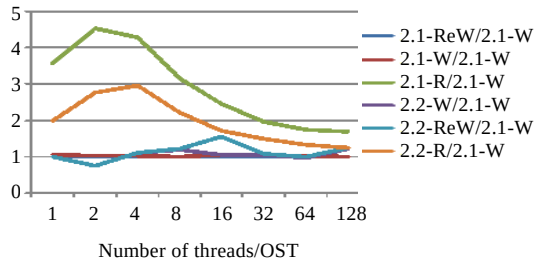

Fig.8 Performance trend under different types of journals

图 8 不同日志类型下的性能变化趋势

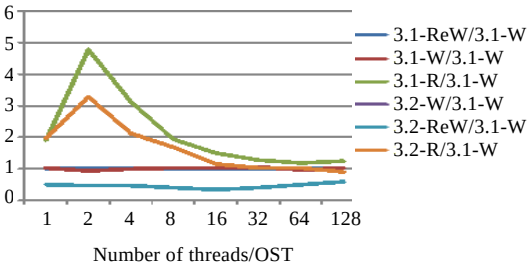

Fig.9 Performance trend under different types of disks

图 9 不同磁盘类型下的性能变化趋势

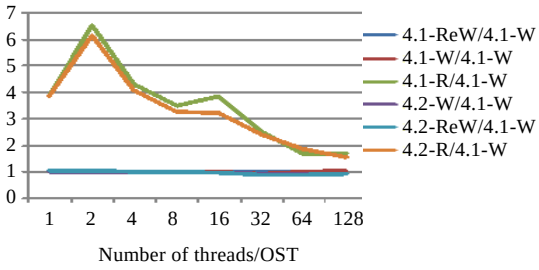

Fig.10 Performance trend under different storage connection approaches

图 10 不同存储连接方式下的性能变化趋势

2.2 性能相关性分析

从上述实验结果可知,读-读的性能、写-写的性能和重写-重写的性能都具有相似的发展趋势.而性能相关性模型(PRModel)是通过对系统性能指标的特征序列几何关系的比较来分析多性能指标的特征序列间的关联程度.下面将使用 PRModel 对实验结果进行相关性分析.其中,表 3 代表在一种不同性能因子下的性能相关性分析,表 4 代表在两种不同性能因子下的性能相关性分析.

Table 3 Relational degree under one different factor

表 3 一种不同性能因子下的性能相关度

(a) Relational degree of case1.1-case1.2					(b) Relational degree of case2.1-case2.2				
γ		Case 1.2			γ		Case 2.2		
		W	ReW	R			W	ReW	R
Case 1.1	W	0.928	0.93	0.874	Case 2.1	W	0.879	0.906	0.86
	ReW	0.94	0.942	0.87		ReW	0.903	0.892	0.872
	R	0.865	0.867	0.869		R	0.77	0.77	0.879
(c) Relational degree of case3.1-case3.2					(d) Relational degree of case4.1-case4.2				
γ		Case 3.2			γ		Case 4.2		
		W	ReW	R			W	ReW	R
Case 3.1	W	0.824	0.836	0.872	Case 4.1	W	0.963	0.957	0.76
	ReW	0.826	0.838	0.873		ReW	0.983	0.976	0.765
	R	0.808	0.799	0.844		R	0.769	0.772	0.96

Table 4 Relational degree under two different factors

表 4 两种不同性能因子下的相关度

(a) Relational degree of case1.2-case2.1

γ		Case 2.1		
		W	ReW	R
Case 1.2	W	0.858	0.802	0.782
	ReW	0.859	0.846	0.782
	R	0.834	0.82	0.849

(b) Relational degree of case2.2-case3.2

γ		Case 3.2		
		W	ReW	R
Case 2.2	W	0.877	0.87	0.859
	ReW	0.867	0.875	0.858
	R	0.76	0.755	0.969

(c) Relational degree of case3.2-case4.2

γ		Case 4.2		
		W	ReW	R
Case 3.2	W	0.868	0.864	0.704
	ReW	0.859	0.855	0.699
	R	0.861	0.864	0.869

从表 3 可以看出,单个不同因子下的写-写、重写-重写和读-读操作性能对存在紧密的相关性,相关度的值比较大.从 PRModel 的几何意义得知,如果两个不同配置的 Lustre 系统的性能指标有紧密的相关度值,则这两个不同配置的 Lustre 系统的性能指标具有相似的变化趋势.这一点从性能评估实验曲线(图 2~图 5)同样可以发现,读-读的性能、写-写的性能和重写-重写的性能都具有相似的发展趋势,这与 PRModel 认为的“刻画性能指标所表示曲线的几何形状越接近,则性能指标发展变化态势越接近,因而他们之间的关联程度越大”是一致的.

表 4 表示在两个不同因子下的性能相关性,其中,(a)表示在不同 OSS 数量和不同日志类型下的性能相关性,(b)表示在磁盘类型和 OSS 数下的性能相关性,(c)表示在不同存储连接类型和不同磁盘类型下的性能相关性.从表 4 发现,读-读的性能、写-写的性能和重写-重写的性能同样具有较强的相关性.

从上面性能相关性模型 PRModel 的分析可知,不同性能影响因子(不同配置)下的 Lustre 系统的性能存在较强的相关性,性能表现为有相似的发展趋势,从而辩证地给相对性能预测模型 RPPModel 打下了理论基础.

2.3 相对性能预测分析

为了验证相对性能模型 RPPModel 的有效性,我们扩充了第 2.1 节的性能评估实验.同样选用第 2.1 节的实验用例和 OBDFilter-survey 基准工具进行测试,但我们在实验同时变化 *Threads*/*OST*(取值分别为 8,16,32,64 和 128,即 2^m(*m*=3,4,5,6,7))和 *Objects*/*OST*(取值分别为 1,2,4 和 8,即 2^n(*n*=0,1,2,3))的情况下测试不同因子(不同配置)下的性能.同时,在配置中将记录大小(record size)设置成 1024KB,I/O 请求大小(I/O request size)设置成 16GB(足够大的 I/O 请求大小可以避免 Cache 效应).对于每个实验用例,同样用 OBDFilter-Survey 分别进行了写(简称 W)、重写(简称 ReW)和读(简称 R)这 3 种操作的测试.其中,所有用例的写操作(W)、重写操作(ReW)和读操作(R)的测试数据分别见表 5~表 7.

Table 5 Test data of cases (W)

表 5 用例测试数据(W)

Case 1.1-W					Case 1.2-W				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	381	404	407	389	8	430	492	441	425
16	659	688	671	646	16	622	702	657	643
32	590	820	837	782	32	728	803	816	754
64	598	884	827	848	64	846	942	922	869
128	837	915	888	893	128	817	937	1 116	1 006
Case 2.1-W					Case 2.2-W				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	319	298	300	304	8	381	404	407	389
16	419	428	410	438	16	659	688	671	646
32	557	555	547	558	32	590	820	837	782
64	624	626	627	626	64	598	884	827	848
128	692	693	695	676	128	837	915	888	893

Table 5 Test data of cases (W) (Continue)

表 5 用例测试数据(W)(续)

Case 3.1-W					Case 3.2-W				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	430	492	441	425	8	179	180	183	179
16	622	702	657	643	16	217	222	219	211
32	728	803	816	754	32	304	314	312	290
64	846	942	922	869	64	424	430	411	398
128	817	937	1 116	1 006	128	484	481	490	488
Case 4.1-W					Case 4.2-W				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	348	363	358	358	8	349	363	362	356
16	419	428	430	416	16	408	422	426	413
32	597	613	624	578	32	548	574	578	537
64	834	831	833	805	64	783	770	749	717
128	933	959	962	973	128	876	889	892	893

Table 6 Test data of cases (ReW)

表 6 用例测试数据(ReW)

Case 1.1-ReW					Case 1.2-ReW				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	386	412	412	397	8	433	446	459	458
16	652	678	653	620	16	635	703	658	620
32	604	837	814	776	32	764	814	781	736
64	621	878	874	853	64	805	952	922	886
128	853	888	910	885	128	829	946	1 022	981
case 2.1-ReW					Case 2.2-ReW				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	313	314	316	314	8	386	412	412	397
16	435	428	432	463	16	652	678	653	620
32	566	550	559	563	32	604	837	814	776
64	627	618	619	605	64	621	878	874	853
128	689	690	690	686	128	853	888	910	885
Case 3.1-ReW					Case 3.2-ReW				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	433	446	459	458	8	180	183	185	185
16	635	703	658	620	16	216	229	225	217
32	764	814	781	736	32	300	323	326	306
64	805	952	922	886	64	427	423	418	414
128	829	946	1 022	981	128	492	493	496	489
Case 4.1-ReW					Case 4.2-ReW				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	352	364	366	365	8	354	361	358	358
16	419	436	446	432	16	410	422	430	417
32	554	602	630	595	32	543	602	579	558
64	830	845	825	816	64	763	769	765	754
128	965	975	980	953	128	892	899	898	878

从以上性能评估实验和 PRModel 的性能相关性分析发现,不同性能影响因子(不同配置)下的性能有着相似的发展趋势,性能之间存在较强的相关性.这种性能的相关性为下一步的相对性能预测模型 RPPModel 的建立打下了理论基础.RPPModel 通过对一对不同配置的 Lustre 系统进行训练,如采用一些机器学习的算法进行训练,去捕捉不同 Lustre 系统的潜在相关性信息,并学习预测不同 Lustre 系统的性能比例因子.

Table 7 Test data of cases (R)

表 7 用例测试数据(R)

Case 1.1-R					Case 1.2-R				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	707	680	603	576	8	823	680	622	615
16	716	791	718	725	16	920	816	755	736
32	829	840	840	761	32	928	944	876	828
Case 1.1-R					Case 1.2-R				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
64	823	813	869	852	64	991	963	965	911
128	849	902	779	914	128	1 002	990	993	949
Case 2.1-R					Case 2.2-R				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	1 001	960	958	948	8	707	680	603	576
16	1 034	1 024	1 000	1 019	16	716	791	718	725
32	1 087	1 037	1 031	1 034	32	829	840	840	761
64	1 084	1 042	1 032	1 061	64	823	813	869	852
128	1 169	1 175	1 158	1 128	128	849	902	779	914
Case 3.1-R					Case 3.2-R				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	823	680	622	615	8	722	534	461	456
16	920	816	755	736	16	716	626	518	507
32	928	944	876	828	32	752	655	584	543
64	991	963	965	911	64	860	693	668	689
128	1 002	990	993	949	128	738	768	807	760
Case 4.1-R					Case 4.2-R				
Throughput (MB/s)	*Objects*/*OST*				Throughput (MB/s)	*Objects*/*OST*			
Threads/*OST*	1	2	4	8	*Threads*/*OST*	1	2	4	8
8	1 373	975	871	876	8	1 134	916	862	837
16	1 469	1 156	985	959	16	1 346	1 120	965	931
32	1 501	1 273	1 092	1 083	32	1 444	1 173	1 095	1 014
64	1 396	1 335	1 313	1 228	64	1 540	1 247	1 202	1 197
128	1 610	1 498	1 477	1 387	128	1 448	1 463	1 418	1 328

在这部分的相对性能预测分析中,我们选用分类回归树算法(CART)作为模型的学习算法.分类回归树的基本算法是贪心算法,它以自顶向下递归的划分-控制方式构造分类回归树.它包括分裂节点和叶节点或起始节点,分裂节点指明分割条件,叶子节点给出分类结果.输入数据通过与分裂节点和分割条件作比较,决定其属于左节点还是右节点.不断重复上述过程,就可以得到该输入数据的分类结果.从根节点到叶节点的所有分割条件则指明了输入变量对应输出结果之间的规则.CART 算法分为最大分类树构造和回归剪枝两个步骤,最大分类树构造是通过将历史数据逐步分解成不相连的子集来完成的,它从包括所有训练数据的根结点开始,通过穷尽搜索,寻找最小化分类误差的分割点.该点产生后,根节点相应地被分为两个子节点,然后对这两个子节点进行同样的划分,直至某一个子节点的划分误差的减少小于某一个确定值.因为复杂的树结构会出现对训练数据的过度拟合,需要对生成的复杂分类树进行有选择的回归剪枝,将一些分裂节点用叶节点代替,从而生成一系列嵌套的子树.最优分类树选择即确定分裂条件,从嵌套的子树序列中选择误分类代价估算值最小的子树作为最优分类回归树.分类回归树可以用来提取规则、输入和输出变量之间的映射关系.关于分类回归树构造的详细步骤可参考文献[21,38].

为加深对 RPPModel 的了解,下面将给出一个简单的示例,演示相对性能预测的基本步骤.通过对表 5 中的测试数据进行子集提取,构造模型的训练样本示例(见表 8).表 8 有两个预测变量,即 *Threads*/*OST* 和 *Objects*/*OST*,被预测的变量是从 $Lustre_i$ 到 $Lustre_j$ 的性能(吞吐量)比例因子.

Table 8 An example of training samples for CART model

表 8 CART 模型的训练样本示例

Threads/*OST*	*Objects*/*OST*	$RF_{i\to j}$
8	1	1.13
16	2	1.02
32	1	1.23
64	2	1.06

按照表 8 模型的训练样本,图 11 展示了分类回归树的构建步骤.在图 11(a)中,将 *Threads*/*OST* 划分为两个类:*Threads*/*OST*≤16 和 *Threads*/*OST*>16. 图 11(b)将 *Objects*/*OST* 划分为两个类:*Objects*/*OST*≤1 和 *Objects*/*OST*>1.采用叶子节点最小均方根误差作为分裂条件可知,*Objects*/*OST* 被优先划分,然后对每个子树进行递归划分,即对每个子树进行 *Threads*/*OST* 划分分类,如图 11(c)所示.

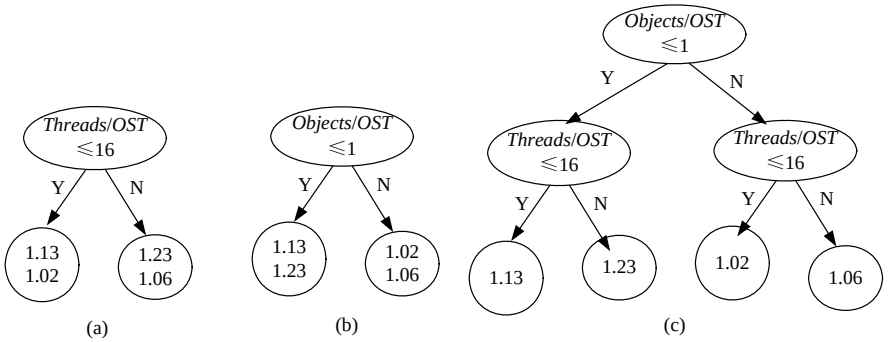

Fig.11 An example of the steps taken by CART

图 11 CART 模型步骤示例

沿着由根节点到树叶节点的路径,可以将图 11(c)的分类回归树转换为 IF-THEN 分类规则,由图 11(c)提取的规则是:

- IF *Objects*/*OST*≤1 and *Threads*/*OST*≤16 THEN $RF_{i\to j}$=1.13;
- IF *Objects*/*OST*≤1 and *Threads*/*OST*>16 THEN $RF_{i\to j}$=1.23;
- IF *Objects*/*OST*>1 and *Threads*/*OST*≤16 THEN $RF_{i\to j}$=1.02;
- IF *Objects*/*OST*>1 and *Threads*/*OST*>16 THEN $RF_{i\to j}$=1.06.

RPPModel 能够对一些新应用的性能进行预测,假设新应用的 *Threads*/*OST* 和 *Objects*/*OST* 分别是 8 和 2.为了对新应用的性能进行预测,将这个新应用分别运行在 $Lustre_i$ 和 $Lustre_j$,且运行在 $Lustre_i$ 的实际吞吐量是 414MB/s,运行在 $Lustre_j$ 的实际吞吐量是 446MB/s.由于新应用的 *Objects*/*OST*>1 且 *Threads*/*OST*≤16,从图 11(c)可知,新应用从 $Lustre_i$ 到 $Lustre_j$ 的性能比例因子 $RF_{i\to j}$=1.02.从运行在 $Lustre_i$ 的实际吞吐量和性能比例因子,可以得到运行在 $Lustre_j$ 的性能预测值 414×1.02=422.28MB/s,相对误差是(446–422.28)/464×100%=8.99%.

通过同样的步骤,对 case1.1~case1.2,case2.1~case2.2,case3.1~case3.2 和 case4.1~case4.2 分别进行了写-写(W-W)、重写-重写(ReW-ReW)、读-读(R-R)和写-重写(W-ReW)这 4 种不同预测情景的性能预测,预测结果的平均相对误差见表 9(注意,表中的 A.R.E 是平均相对误差的简称).

从表 9 可以看出,写操作(W-W)的性能预测结果的平均相对误差能控制在 17%~28%的范围内,重写操作(ReW-ReW)的性能预测结果能够控制在 17%~26%的范围内,读操作(R-R)和写-重写(W-ReW)的预测结果的平均相对误差分别控制 20%~28%和 17%~28%的范围内.RPPModel 的性能预测结果的平均相对误差能控制在 17%~ 28%的范围内,保持一个较好的预测准确度.

Table 9 Average relative errors of prediction results

表 9 预测结果平均相对误差

(a) W-W prediction			(b) ReW-ReW prediction		
Predictor variable	Predicted variable	A.R.E. (%)	Predictor variable	Predicted variable	A.R.E. (%)
Case1.1-W	Case1.2-W	18.52	Case1.1-ReW	Case1.2-ReW	21.40
Case2.1-W	Case2.2-W	27.31	Case2.1-ReW	Case2.2-ReW	25.21
Case3.1-W	Case3.2-W	19.09	Case3.1-ReW	Case3.2-ReW	17.15
Case4.1-W	Case 4.2-W	23.45	Case4.1-ReW	Case4.2-ReW	23.38
(c) R-R prediction			(d) W-ReW prediction		
Predictor variable	Predicted variable	A.R.E. (%)	Predictor variable	Predicted variable	A.R.E. (%)
Case1.1-R	Case1.2-R	24.16	Case1.1-W	Case1.2-ReW	25.36
Case2.1-R	Case2.2-R	27.18	Case2.1-W	Case2.2-ReW	27.88
Case3.1-R	Case3.2-R	21.75	Case3.1-W	Case3.2-ReW	19.42
Case4.1-R	Case4.2-R	19.49	Case4.1-W	Case4.2-ReW	17.11

3 结束语

并行文件系统的性能对整个 HPC 系统的性能有着重大的影响,其性能评估和建模是一个潜在研究热点和难点.本文选用目前非常流行的 Lustre 文件系统作为研究对象,设计并进行了一系列的性能评估实验,同时提出了性能相关性模型(PRModel),对不同性能影响因子下性能进行了相关性研究.从实验和 PRModel 分析中发现,不同因子下的性能之间存在着相似的性能发展趋势,不同因子下的性能之间存在较强的相关性.这种性能相关性的发现,为相对性能预测模型(RPPModel)的构建打下理论基础.为了挖掘出不同配置的 Lustre 系统间性能相关性信息,提出了 RPPModel 模型.通过对一对不同配置的 Lustre 系统进行训练,如采用一些机器学习的算法进行训练,去捕捉不同 Lustre 系统的潜在相关性信息,并学习预测不同 Lustre 系统的性能比例因子.在实验中我们发现,RPPModel 在写-写(W-W)、重写-重写(ReW-ReW)、读-读(R-R)和写-重写(W-ReW)这 4 种不同预测情景下,性能预测结果的平均相对误差分别能够控制在 17%~28%,17%~26%,20%~28%和 17%~ 28%的范围内.RPPModel 的性能预测结果的平均相对误差能够控制在 17%~28%的范围内,保持一个较好的预测准确度且易于使用.本文提出的 PRModel 模型和 RPPModel 模型对其他大中型分布式/并行文件系统的性能研究具有较好的借鉴价值.如果其他大中型分布式/并行文件系统的不同性能因子间也具有较好的性能相关性,那么 PRModel 模型和 RPPModel 模型对这些大中型分布式/并行文件系统的性能研究同样具有较好的适用性和有效性.

致谢 非常感谢来自 Sun Microsystems Inc.的 Atul Vidwansa 给予我们真诚的帮助,尤其在实验搭建方面.

References:

[1] Wang F, Yue YL, Feng D, Wang J, Xia P. High availability storage system based on two-level metadata management. In: Proc. of the 2007 Japan-China Joint Workshop on Frontier of Computer Science and Technology (FCST 2007). 2007. 41−48. [doi: 10.1109/FCST.2007.19]

[2] Yu WK, Vetter JS, Canon RS, Jiang S. Exploiting Lustre file joining for effective collective IO. In: Proc. of the 7th IEEE Int'l Symp. on Cluster Computing and the Grid (CCGrid 2007). 2007. 267−274. [doi: 10.1109/CCGRID.2007.51]

[3] Yu WK, Vetter JS, Oral HS. Performance characterization and optimization of parallel I/O on the Cray XT. In: Proc. of the 22nd IEEE Int'l Parallel and Distributed Processing Symp. (IPDPS 2008). 2008. 1−11. http://dx.doi.org/10.1109/IPDPS.2008.4536277

[4] Li HY, Liu Y, Cao Q. Approximate parameters analysis of a closed fork-join queue model in an object-based storage system. In: Proc. of the 8th Int'l Symp. on Optical Storage and 2008 Int'l Workshop on Information Data Storage. 2008. 1−6. [doi: 10.1117/12.822592]

[5] Yu WK, Oral HS, Canon RS, Vetter JS, Sankaran R. Empirical analysis of a large-scale hierarchical storage system. In: Proc. of the 14th Int'l Euro-Par Conf. on Parallel Processing. LNCS 5168, 2008. 130−140. [doi: 10.1007/978-3-540-85451-7_15]

[6] Piernas J, Nieplocha J, Felix EJ. Evaluation of active storage strategies for the Lustre parallel file system. In: Proc. of the 2007 ACM/IEEE Conf. on Supercomputing (SC 2007). 2007. 1−10. [doi: 10.1145/1362622.1362660]

[7] Zhang H, Wu WG, Dong XS, Qian DP. A high availability mechanism for parallel file system. In: Proc. of the 6th Int'l Workshop on Advanced Parallel Processing Technologies (APPT 2005). LNCS 3756, 2005. 194−203. [doi: 10.1007/11573937_22]

[8] Yu WK, Oral HS, Vetter J, Barrett R. Efficiency evaluation of Cray XT parallel IO stack. In: Proc. of Cray User Group Meeting (CUG 2007). 2007. 1−9.

[9] Shan HZ, Shalf J. Using IOR to analyze the I/O performance for HPC platforms. In: Proc. of Cray User Group Conf. 2007. 2007. 1−15.

[10] Kim MY, Tantawi AN. Asynchronous disk interleaving: Approximating access delays. IEEE Trans. on Computers, 1991,40(7): 801−810. [doi: 10.1109/12.83618]

[11] Kim MY. Synchronized disk interleaving. IEEE Trans. on Computers, 1986,35(11):978−988. [doi: 10.1109/TC.1986.1676699]

[12] Ruemmler C, Wilkes J. An introduction to disk drive modeling. IEEE Computer, 1994,27(3):17−28. [doi: 10.1109/2.268881]

[13] Varki W, Merchant A, Xu J, Qiu X. Issues and challenges in the performance analysis of real disk arrays. IEEE Trans. on Parallel and Distributed Systems, 2004,15(6):559−574. [doi: 10.1109/TPDS.2004.9]

[14] Feng D, Qin LJ. Adaptive object placement in object-based storage systems with mining blocking probability. In: Proc. of the 20th Int'l Conf. on Advanced Information Networking and Application. 2006. 611−616.

[15] Bokhari S, Rutt B, Wyckoff P, Buerger P. Experimental analysis of a mass storage system. Concurrency and Computation: Practice and Experience, 2006,18(4):1929−1950. [doi: 10.1002/cpe.1038]

[16] Thomasian A, Liu C. Comment on "issues and challenges in the performance analysis of real disk arrays". IEEE Trans. on Parallel and Distributed Systems, 2005,16(11):1103−1104. [doi: 10.1109/TPDS.2005.132]

[17] Verma A, Anand A. General store placement for response time minimization in parallel disks. Journal of Parallel and Distributed Computing, 2007,67(12):1286−1300. [doi: 10.1016/j.jpdc.2007.07.012]

[18] Uysal M, Alvarez GA, Merchant A. A modular, analytical throughput model for modern disk arrays. In: Proc. of the Int'l Workshop on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems. 2001. 183−192.

[19] Lebrecht AS, Dingle NJ, Knottenbelt WJ. A response time distribution model for zoned RAID. In: Proc. of the 15th Int'l Conf. on Analytical and Stochastic Modeling Techniques and Applications (ASMTA 2008). LNCS 5055, 2008. 144−157. [doi: 10.1007/978-3-540-68982-9_11]

[20] Kelly T, Cohen I, Goldszmidt M, Keeton K. Inducing models of black-box storage arrays. Technical Report, HPL-2004-108, 2004. 1−14.

[21] Wang MZ, Au K, Ailamaki A, Brockwell A, Faloutsos C, Ganger GR. Storage device performance prediction with CART models. In: Proc. of the 12th Annual Int'l Symp. on Modeling, Analysis, and Simulation of Computer and Telecommunications Systems (MASCOTS 2004). 2004. 588−595.

[22] Mesnier MP, Wachs M, Sambasivan RR, Zhang AX, Ganger GR. Relative fitness modeling. Communications of the ACM, 2009,52(4):91−96. [doi: 10.1145/1498765.1498789]

[23] Mesnier MP, Wachs M, Sambasivan RR, Zheng AX, Ganger GR. Modeling the relative fitness of storage. Performance Evaluation Review, 2007,35(1):37−48. [doi: 10.1145/1269899.1254887]

[24] Anderson E. Simple table-based modeling of storage devices. SSP Technical Report, HPL-SSP-2001-4, 2001. 1−4.

[25] Sun Microsystems, Inc. Lustre™ File System. 2008. 1−15.

[26] Sun Microsystems, Inc. Solving the HPC I/O Bottleneck: Sun™ Lustre™ Storage System. 2009. 1−27.

[27] Sun Microsystems, Inc., Oak Ridge National Laboratory (ORNL). Peta-Scale IO with the Lustre File System. 2008. 1−18.

[28] DataDirect Networks, Inc. Best Practices for Architecting a Lustre-Based Storage Environment. 2008. 4−26.

[29] Xie NM, Liu SF. Research on evaluations of several grey relational models adapt to grey relational axiom. Journal of Systems Engineering and Electronics, 2009,20(2):304−309.

[30] Yu WK, Vetter JS. Xen-Based HPC: A parallel I/O perspective. In: Proc. of the 8th IEEE Int'l Symp. on Cluster Computer and the Grid (CCGrid 2008). 2008. 154−161. [doi: 10.1109/CCGRID.2008.119]

[31] Yu WK, Vetter J. ParColl: Partitioned collective I/O on the Cray XT. In: Proc. of the Int'l Conf. on Parallel Processing (ICPP 2008). 2008. 562−569. [doi: 10.1109/ICPP.2008.76]

[32] Logan J, Dickens P. Towards an understanding of the performance of MPI-IO in Lustre file systems. In: Proc. of the 2008 IEEE Int'l Conf. on Cluster Computing. 2008. 330−335. [doi: 10.1109/CLUSTR.2008.4663791]

[33] Liu SF, Dang YG, Fang ZK. Grey System Theory and Applications. 3rd ed., Beijing: Science Press, 2004. 21−112 (in Chinese).

[34] Kayacan E, Ulutas B, Kaynak O. Grey system theory-based models in time series prediction. Expert Systems with Application, 2010,37(2):1784−1789. [doi: 10.1016/j.eswa.2009.07.064]

[35] Lu M, Wevers K. Grey system theory and applications: A way forward. Journal of Grey System, 2007,10(1):47−54.

[36] Wang MZ, Ailamaki A, FAaloutsos C. Capturing the spatio-temporal behavior of real traffic data. Performance Evaluation, 2002,49(4):147−163. [doi: 10.1016/S0166-5316(02)00108-6]

[37] Wang M, Madhyastha T, Chan NH. Data mining meets performance evaluation: Fast algorithms for modeling bursty traffic. In: Proc. of the Int'l Conf. on Data Engineering. 2002. 507−516.

[38] Razi MA, Athappilly K. A comparative predictive analysis of neural networks (NNs), nonlinear regression and classification and regression tree (CART) models. Expert Systems with Applications, 2005,29(1):65−74. [doi: 10.1016/j.eswa.2005.01.006]

附中文参考文献:

[33] 刘思峰,党耀国,方志耕.灰色系统理论及其应用.第 3 版,北京:科学出版社,2004.21−112.

赵铁柱(1983－),男,湖南娄底人,博士,主要研究领域为高性能计算,虚拟化技术,分布式计算.

Verdi MARCH(1978－),男,博士,主要研究领域为并行与分布式计算.

董守斌(1967－),女,博士,教授,博士生导师,主要研究领域为高性能计算,信息检索,下一代互联网.

Simon SEE(1965－),男,博士,教授,主要研究领域为高性能计算,虚拟化技术,应用数学及模拟方法.