基于对象存储设备上的服务质量研究*)

赵水清　冯　丹

(华中科技大学信息存储系统教育部重点实验室　武汉 430074)

摘　要　基于对象存储(Object-based Storage，OBS)作为下一代互联网存储协议标准逐渐被人们所接受，基于对象存储设备(Object-based Storage Device，OSD)所具有的可扩展性和智能性能很好地支持应用程序的服务质量(Quality of Service，QoS)需求。本文分析了基于对象存储系统的QoS需求，讨论了如何把应用客户的QoS需求转化为QoS属性，扩展了OSD SCSI协议集标准，以支持QoS，接着采取量化分析方法对QoS进行了分析，详细分析应用客户和OSD之间QoS信息交互的工作过程，最后，紧密结合OSD的特性，给出OSD上的QoS三层模型Q-Model和一优化算法BRP。

关键词　基于对象存储设备，服务质量，智能存储设备

The Study of Quality of Service in Object-based Storage Device

ZHAO Shui-Qing　FENG Dan

(Data Storage System Key Laboratory of Ministry of Education, Huazhong University of Science and Technology, Wuhan 430074)

Abstract　Object-based storage(OBS), considered as the next Internet storage protocol standard, has been accepting gradually. Object-based storage device(OSD)can give more support on applications' Quality of Service(QoS)requirements. This paper firstly analyses object-based storage system's QoS requirements, discusses how to translate QoS requirements to QoS attributes and extends the OSD SCSI protocol standard to support QoS, then using quantify analysis method, analyses QoS and the process of communicating QoS information between application and OSD. finally, integrating the OSD's characters, the paper proposes a QoS three-layer model called Q-Model and an optimization arithmetic called BRP in OSD.

Keywords　OSD, QoS, Intelligent Storage Device

1　引言

信息的爆炸式增长和网络的飞速发展促使网络存储技术的发展，而人们对存储系统的新要求，如安全机制、QoS支持、智能化的存储设备等，使得OBS存储协议标准逐渐被人们所关注和认可，被认为是下一代互联网存储新标准。存储网络工业协会(Storage Networking Industry Association，SNIA)提交给T10的一个基于OSD的SCSI命令集[1]，是SCSI标准命令集的一部分。美国国家实验室和惠普公司联合开发的Lustre项目[2]则构建了一个基于OSD协议标准的文件系统，与对象存储设备的交互系采用全新的对象接口。

对传统的网络存储系统而言，由于存储设备对外提供接口时存在缺陷，比如NAS的文件级接口、SAN的块级接口，它们都缺乏对QoS语义上的支持，不能理解应用程序的QoS需求，故需要向用户提供QoS支持时显得力不从心。从应用程序的角度来看，NAS和SAN都仅仅是一个尽力服务模型。然而，在基于OSD的存储系统，OSD向用户提供的是一个富有表现力的对象级接口，这给人们带来了很多新的机会和挑战。对象的属性具有高可扩展性，用户可以随意定义自己的属性，当然也就包括QoS相关属性。因此，OSD可以很好地接受和理解QoS需求；另外，OSD还是一个具有智能性的存储设备，通过不断的自我学习，在满足用户的QoS前提下，使得存储设备拥有更优的存储性能。

文[3]提出了基于OSD存储系统上的一个QoS框架，扩展了由T10的OSD SCSI命令协议集和iSCSI协议集以支持这三级的QoS，也给出了OSD上的一个简单的软件模块框架。文[4]从用户和OSD之间的QoS需求交互过程出发，给出了在具体的应用中，应用程序的QoS需求如何转换为对象的QoS属性，对OSD SCSI命令集也进行了扩展。上面两篇论文虽然对OSD上的QoS都进行了描述，但都很简单而粗略，不具体。而OSD是整个存储系统上实现QoS支持的核心部件，因此需要把OSD单独拿出来，结合其特性——可扩展性和智能性，进行具体而详尽的研究。

2　基于OSD存储系统

2.1　基于OSD存储系统体系结构

基于OSD的存储系统主要由3部分组成：元数据服务器(Metedata Server，MDS)、基于对象存储设备(OSD)和客户端(Client)。其体系结构如图1所示。

(1)元数据服务器(MDS)。管理与文件系统上层有关的元数据，负责维护目录树和文件到对象的映射关系，其功能还包括元数据一致性维护和安全策略，比如身份验证、授权证书管理、访问控制等。

(2)基于对象存储设备(OSD)。负责文件系统下层的管

*)国家"九七三"重点基础研究发展规划资助项目(2004CB318201)。**赵水清**　硕士研究生，主要研究方向：网络存储系统、磁盘陈列等；**冯　丹**　博士生导师，主要研究方向：计算机系统结构、磁盘阵列技术、海量存储、网络存储等。

理，如存储空间的分配。基于对象存储设备对外提供一个标准的对象级接口，具有跨平台和可扩展性的特点，其集合带宽(Aggregate Bandwidth)随着基于对象存储设备的增加而接近呈线性增加。另外，基于对象存储设备还是一个智能化的存储设备，可进行自我学习、自我管理、自我优化等。

(3)客户端(Client)。其上运行一个基于对象存储文件系统(Object-based Storage File System，OBFS)，上层仍然符合VFS接口，对文件的操作仍然是调用Open()，Read()，Write()等。对应用客户来说，他看不到下层的实现细节，其表现的界面跟操作方法和传统的文件系统没有差别。

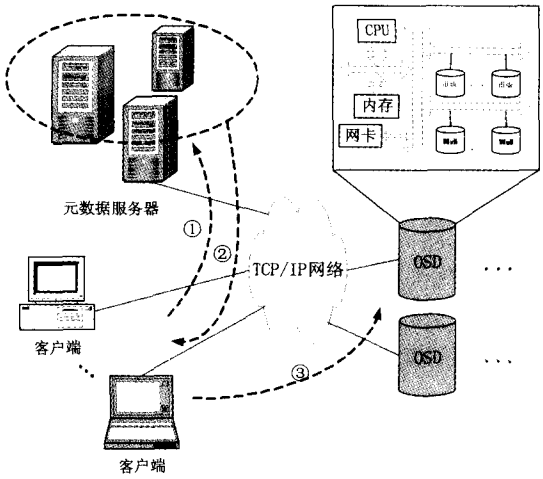

图1 基于OSD存储系统体系结构

3个部分之间的通讯流程简单描述为：

① 客户端向元数据服务器发送请求，如对某一文件的Open，Read，Create操作。

② 元数据服务器对客户端进行身份验证，并向客户端发送证书和与文件有关的元数据，证书描述了客户端可进行的操作。

③ 客户端向OSD发送请求，请求包含有命令和元数据服务器发给的证书，建立与OSD之间连接；OSD对证书的真伪和完整性进行检查，通过验证后，OSD与客户端之间直接进行数据传送。

2.2 对象属性

对象被看作是存放数据的容器，在OSD SCSI协议标准里提出了4种类型的对象：根对象(Root Object)、分区对象(Partition Object)、集合对象(Collection Object)、用户对象(User Object)。对象有3个基本要素：数据、对象属性和对象操作。对象属性是对象数据的元数据，对象操作定义了对该对象所允许的所有操作，例如Read，Write，GetAttribute等。外部用户对对象的所有访问都是通过对象操作这个接口进行的。另外，对象属性和对象操作还是可扩展的。

按照对象属性存在时间的相对长短，我们将它简单地分为2类：永久属性和临时属性。

· 永久属性 是随着对象的创建而建立，随着对象的删除才消亡，比如对象的创建时间属性、对象的UID属性。

· 临时属性 是在一段特定的时间内存在和有效，这类属性主要是用户的自定义属性，是临时创建的，不需要的时候就加以删除。临时属性充分体现了对象属性的可扩展特性。这样，用户要定义一个属于自己的属性，比如QoS属性，就非常简单了。

2.3 多级别QoS

根据对象的类型和OSD SCSI命令处理模型，我们定义了同一应用上的多级别QoS服务，依次分为4个级别：分区对象级、用户对象级、会话级和操作级。对多级别QoS服务的支持是基于对象存储系统的一个显著的特点，应用客户因此可以更加自由、更加灵活地给出自己的QoS要求。

· 分区对象级(Partition Level)。在分区对象上建立的QoS服务要求，分区对象级的QoS服务对所有的会话、每次会话内的所有操作都有效。

· 用户对象级(User Level)。在用户对象上建立的QoS服务要求，用户对象级的QoS服务对所有的会话、每次会话内的所有操作都有效。

· 会话级(Session Level)。在会话开始前建立的QoS服务要求，只对当前的会话成立。

· 操作级(Operation Level)。对某个具体操作定义的QoS服务要求，只对当前的操作成立。

值得注意的是，在分区对象上建立的QoS服务请求对其中所有的用户对象都有效。但是，若某个用户对象不需要QoS支持时，可以通过设置该用户对象的QoS属性页中的请求类型属性项为00h，详见3.2节。另外，这4个级别并不是彼此互斥的，一个应用可以同时具有多个级别的QoS服务请求。当同一应用同时具有多个级别的QoS属性定义的时候，就应该考虑这些QoS服务的优先级别了。用P_p，P_u，P_s，P_a分别表示分区对象级、用户对象级、会话级、操作级的优先级，我们规定：

$$P_p<P_u,\ P_s<P_a$$

以在线视频点播应用为例，播放的视频对象对应于某个用户对象，该用户对象又属于某一个分区对象，若在该分区对象上定义了一个分区级别的QoS属性，如要求最低带宽是300kb/s；同时在该用户对象也定义了一个用户级别的QoS属性，如要求最低带宽是600kb/s。此时，该在线视频点播应用对应了两个级别的QoS定义，即同时拥有两个级别的QoS服务请求。因为$P_p<P_u$，最终OSD给应用提供的最低带宽要求为600kb/s。

3 QoS量化分析

3.1 QoS需求

Ravi Wijayaratne把存储系统的应用需求分成3类：周期性需求、交互式需求和非周期性需求[5]。周期性需求要求它发出的服务请求在一定的时间间隔内要得到满足；交互式需求要求它发出的服务请求得到最快的响应；非周期性需求主要指那些普通的文件请求。

用存储系统的性能指标来说，周期性需求可以理解为有一定的带宽要求，典型的例子就是在线视频点播，它的每个流媒体数据包必须在规定的最大时间间隔内到达，不然就会出现抖动的现象；交互式需求认为是有最大响应延迟的要求，在线的网络游戏是该类型需求的最好应用；非周期性需求则是对整个存储系统的可靠性的要求，要求存储系统能提供可持续服务。因此，我们把应用请求分成3类：基于带宽要求的请求、基于延时要求的请求和基于可靠性要求的请求(也叫非QoS要求的请求)。

3.2 QoS属性

在OSD SCSI协议标准里，对象属性用许多个属性页来描述，每个属性页又由许多具体的属性项组成，属性页由属性页号*Page Number*来确定，属性项对应有属性号*Attribute*

Number。这样，具体一个属性项用二元组(*Page Number*，*Attribute Number*)来索引。利用对象的属性的可扩展性，我们对OSD SCSI协议标准的属性页进行扩展，自定义一个QoS属性页。根据应用请求的分类，预定义请求类型、带宽、时延和可靠性这4个属性项，该属性页的结构如表1。

(1)带宽(*BW*)。应用客户和OSD之间每秒钟传输的字节数，若在时间$(T-t)$内传输的字节数为N，则$BW = N/(T-t)$，故BW是一段时间内的平均值而不是即时值。

(2)时延(*D*)。从应用客户发出应用请求时刻$t1$到收到响应时刻$t2$所化的时间，包括请求服务时间和网络延迟，$D = t2-t1$。

(3)可靠性(*DP*)。OSD能够正常提供服务的能力，可靠性的描述比较抽象，我们用一个概率值来表示，$DP = MTTF/(MTTF + MTTR)$，其中*MTTF*(mean time to failure)表示平均无故障时间，*MTTR*(mean time to repair)表示平均修复时间。

请求类型是为了区分应用请求的类型，其值的定义见表2。应该注意，应用客户只能设置带宽、时延和可靠性这3个属性项，而没有权限设置请求类型属性项，它必须由OSD来设置。

表1 扩展的QoS属性页结构

属性号	长度(字节)	属性项	客户是否可设置
0h	40	属性页标志	否
1h	1	请求类型	否
2h	2	带宽	是
3h	20	时延	是
4h	4	可靠性	是
5h-FFFF FFFEh		保留	

表2 请求类型属性项的值定义

属性值	请求类型
0h	无QoS要求请求
1h	基于带宽要求请求
2h	基于时延要求请求
3h-FFh	保留

3.3 QoS的形式化描述

在系统的实现过程中，应用请求的QoS需求最终必须进行量化，在量化的基础上才能进行QoS的协商等工作。下面给出量化的形式化描述，首先列出后面所用的一些符号的定义和说明，见表3。

表3 符号定义说明

符号	说明
A	QoS属性集合
a	QoS属性，$a\in A$
V	QoS属性量化值的集合
v	QoS属性量化值，$v\in V$
M	匹配关系的集合
m	匹配关系，$m\in M$
R	约束值集合
r	约束值，$r\in R$
B	QoS请求响应，B取值0(拒绝)或者1(接受)

定义1 匹配关系$m(q1,q2)$为一个二元运算符，表示$q1$和$q2$匹配。

说明：在应用客户和OSD进行QoS参数协商过程中，只存在两种匹配关系，即代数意义上的“大于等于≥”和“小于等于≤”。

定理1(准入定理) $B = 1$成立的条件是当且仅当：

$\forall v\in V$，对相应的$m\in M, r\in R$，有$m(v,r)$。

说明：在应用客户和OSD进行QoS参数协商过程中，OSD要根据当时系统的负载情况，对应用客户的QoS需求进行判断，从而决定是接受还是拒绝应用可以的QoS请求。只有当应用客户所有的QoS需求都得到满足的情况下才被OSD所接纳。

推论1 若$\exists v\in V$，对相应的$m\in M, r\in R, m(v,r)$不成立，则$B=0$。

说明：该推论是定理1的逆否命题，说明OSD系统在当前负载情况下，只要有一个QoS需求不能得到满足，则不再需要继续判断其它的QoS需求，OSD系统就可以做出拒绝该QoS请求的响应。

3.4 工作过程

应用程序的QoS需求如何表达并告诉存储系统，存储系统又如何接收并理解客户的QoS请求，这是应用客户和存储设备之间的QoS信息交互问题。在传统的存储系统中，正是由于缺乏对这种QoS信息交互的支持，从而对QoS的支持力不从心，只能做到尽力服务而已。所以，人们在研究存储系统的QoS的时候，不得不在传统的存储系统模型上引入一个新的机制来支持QoS。例如，有人提出了的基于语法的QoS[6]，即事先定义一种QoS语法，语法规定了QoS的相关参数和QoS信息的交互规则。

OSD SCSI命令处理模型是一个请求-应答(request-response)模型，即任何一个操作都必须是先发送请求(这里的请求指OSD SCSI命令)，然后才有应答。从上面已经知道，在基于OSD存储系统中，QoS需求信息最终都表现为QoS属性。而*SET/GET ATTRIBUTES*命令是专门用来设置、获取对象属性的，因此用*SET ATTRIBUTES*命令可以实现QoS属性设置。具体的工作过程分为以下4步：

(1)应用客户调用*QosRequest*()操作向OSD发出QoS请求。具体的操作接口如下：

QoSRequest(u_int64 Partition_ID, u_int64 User_ID)

其中，参数*Partition_ID*为请求对象的分区ID，*User_ID*为请求对象的用户ID，由二元组(*Partition_ID*, *User_ID*)可以唯一确定OSD上被请求的对象。

(2)应用客户和OSD之间进行QoS参数协商操作*Negotiate*()，其实，*Negotiate*操作最终调用的是多次*SetAttribute*()操作。*SetAttribute*的操作接口为：

SetAttribute (u_int64 *PageNumber*, u_int64 *AttributeNumber*, void **Qos*, int *Length*)

其中，参数*PageNumber*是QoS属性页号，我们定义为8000h；*AttributeNumber*代表要设置的QoS属性的属性号，*Qos*为QoS属性的量化值，*Length*为*Qos*所占的空间大小，用Byte数来表示。*SetAttribute*操作完毕之后，OSD将对刚才的*Qos*进行匹配操作m。按照定理1(准入定理)，得到最终的匹配结果，将以QoS响应的形式返回给应用客户。

(3)OSD调用*QosResponse*()操作返回QoS响应给应用客户，其操作接口定义为：

Bool *QosResponse*(u_int64 *Partition_ID*, u_int64 *User_ID*)

其参数意义和(1)中的相同。返回值是一个布尔类型的值，是OSD进行一系列匹配操作后的最终结果。

(4)应用客户在请求完毕的时候，调用 *FeedBack*()操作把实际的QoS的量化值反馈给OSD。*FeedBack* 操作接口定义为：

FeedBack(u_int64 *PageNumber*, u_int64 *AttributeNumber*, void **Qos*, *int Length*)

其参数意义和(2)中的相同，这里不赘述。

4 QoS模型

存储系统QoS的实现目前有2种方式：综合服务和区分服务。要求的两者的主要区别在于是否为单个的应用请求预留资源。但是，由于Linux操作系统内核本身的原因，使得不能够很好地实现综合服务，所以下面将重点讨论区分服务模型。

4.1 OSD上的QoS三层模型

在详细分析基于对象存储系统的QoS需求、QoS属性的量化方法以及应用客户和OSD之间QoS信息交互的工作过程后，结合OSD特性，我们提出了一个OSD上的QoS三层模型Q-Model，如图2。

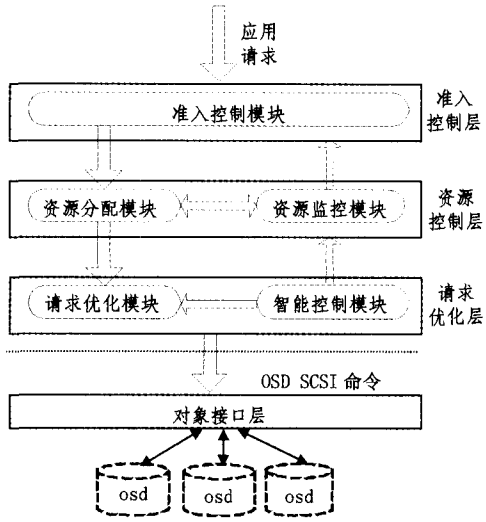

图2 Q-Model层次图

Q-Model由上到下依次为：准入控制层、资源控制层、请求优化层。那些QoS得到满足的并且经过系统优化了的应用请求，发送给下面的对象接口层，由它进行命令的解析并最终操作物理设备。注意，对象接口层并不是QoS模型里面的一部分。

(1)准入控制层(Admission Control Layer, ACL)是所有有着QoS需求的应用请求进入存储设备的第一道门，也叫准入控制器。它从下层资源控制层的资源监控模块那里得到实时的系统和网络信息，据此判断应用请求的QoS需求是否可以得到满足。若满足，则该应用请求可以进入系统，为其分配相应的资源，应用客户也可以继续发送后面的操作命令；若不满足，则给应用客户发QoS服务被拒绝的响应。

(2)资源控制层(Resource Control Layer, RCL)也叫资源监控器，是所有要提供QoS服务的系统的必要部件之一。这里的资源包括两个方面：一是系统资源，包括CPU计算资源，内存资源等；一是网络资源，主要是指网络带宽、网络利用率等。

(3)请求优化层(Request Optimize Layer, ROL)是该模型中的一个重要部件，也是OSD的一个特色。OSD负责对象空间的分配，加上可扩展的属性，是OSD能具有智能的两个关键因素。该层充分利用了OSD所具有的智能性，对已经满足了QoS需求的应用请求再进行优化，使得设备的性能得到充分的发挥，提供设备资源的利用率，然后其信息又将反馈RCL，最终影响到ACL，使之能接受更多的应用请求，从而形成一个良性循环。

4.2 准入控制器

准入控制器的主要作用是控制应用请求进入存储设备服务系统，只有当应用请求的QoS需求能够得到系统的满足时才能进入请求队列。准入控制器由两大功能部件构成：请求类型设别部件和准入子控制部件——带宽准入子控制器、最少时延准入子控制器和可靠性准入子控制器。准入控制器主要完成两大功能：

(1)应用请求类型识别。

在3.1节中对应用请求进行了分析，把应用请求分成了3大类，即基于带宽要求的请求、基于最少时延限制的请求和一般的普通请求(或者说非QoS要求的请求)。3类请求用QoS属性项—请求类型来区分。应用请求的类型可以由OSD自动识别并设置其请求类型属性项。例如，某网络游戏应用请求在QoS协商阶段只对时延(D)属性项进行了设置，那么OSD就可以知道该应用请求的类型是基于时延要求的，然后再自动设置其请求类型属性值为02h，从而完成应用请求类型的识别。

(2)把QoS请求得到满足的应用请求放入系统请求队列。

在OSD完成了对应用请求类型的识别之后，根据请求类型属性把请求置入相应的准入子控制器的请求队列。准入子控制器负责自己的请求队列，按照FIFO的顺序依次判断请求队列中的每个请求的QoS需求是否可以被满足。如果满足，则把它置入系统的请求队列中。应用请求的准入控制处理如图3所示。

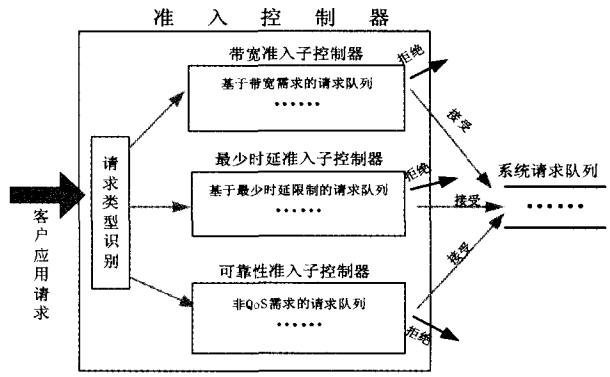

图3 准入控制处理示意图

4.3 智能控制器

从资源控制层(RCL)下来的应用请求队列是没有经过任何优化的。尽管OSD已经能够保证队列中所有应用请求的QoS需求，但是不能保证发挥出系统的最佳性能，提高资源的有效利用率。这样，OSD也就没能体现出它比传统存储设备的优越性。而OSD是一个智能设备，利用其智能性来优化应用请求是OSD的特色，也是将来很具有挑战性的一个领域，这正是智能控制器所要完成的任务。

(下转第127页)

觉系统)构造自适应算法是未来具有挑战性的水印研究方向。

(4)现有图像水印算法向多媒体领域的移植。基于 VQ 的水印算法多集中在图像水印研究,基于音频等领域的水印还较欠缺。现有水印算法向多媒体领域的移植,将是今后的发展方向。

参 考 文 献

1 Swanson M D, Kobayashi M, Tewfik A H. Multimedia Data Embedding and Water marking Technologies [J]. Proceedings of the IEEE, 1998, 86(6):1064~1087
2 Miller M L, Cox I J, Bloom J A. Informed Embedding: Exploiting Image and Detector Information During Watermark Insertion [A]. In: Proc. of IEEE Intl. Conf. on Image Processing [C]. Vancouver: IEEE Press, 2000. 1~4
3 Tirkel A Z, Rankin G A, Van Schyndel R G, et al. Electronic watermark [C]. Proceedings DICTA-93, Macquarie University, Sydney, 1993, 1:666~673
4 Cox I J, Kilian J, Leighton F T, Shamoon T. Secure spread spectrum watermarking for multimedia [J]. IEEE Trans. Image Processing, 1997, 6(12): 1673~1687
5 Linde Y, Buzo A, Gray R M. An algorithm for vector quantizer design [J]. IEEE Trans. Commun, 1980, 28(1):84~95
6 Lu Z M, Sun S H. Digital image watermarking technique based on vector quantisation [J]. Electronics Letters, 2000, 36(4):303~305
7 Jo M, Kim H D. A digital image watermarking scheme based on vector quantization [J]. IEICE Trans. Inf. & Syst., 2002, E85-D(6):1054~1056
8 Lu Z M, Pan J S, Sun S H. VQ-based digital image watermarking method [J]. Electron. Lett., 2000, 36(14):1201~1202
9 Wu Hsien-Chu, Chang Chin-Chen. A novel digital image watermarking scheme based on the vector quantization technique [J]. Computers & Security, 2005, 24(6): 460~471
10 Pan J S, Sung M T, Hsiang C H, et al. Robust VQ-based Digital Watermarking for Memoryless Binary Symmetric Channel [J]. ISCAS, 2004(5):580~583
11 Lu Z M, Xing W, Xu D G, et al. Digital Image Watermarking Method Based on Vector Quantization with Labeled Codewords [J]. IEICE Trans. Inf. & Syst., 2003, E86-D(12):2786~2789
12 Lin Y C, Huang Z K, Pong T T, et al. A Robust Watermarking Scheme Combined with the FSQV for Images [J]. ICITA'05, 2005
13 Makur A, Selvi S S. Variable dimension vector quantization based image watermarking [J]. Signal Processing, 2001, 81(4): 889~893
14 Wang F H, Jain L C, Pan J S. VQ-based gray watermark embedding scheme with genetic index assignment [J]. Int. Journal of Computational Intelligence and Applications (IJCIA), 2004, 4(2):165~181
15 Wang F H, Jain L C, Pan J S, et al. A VQ-Based Image Data Hiding Scheme [J]. ICME, 2004. 2191~2194
16 Wang F H, Jain L C, Pan J S. Hiding Watermark in Watermark [J]. In: IEEE Intl. Symposium on Circuits and Systems, 2005. 4018~4021
17 Huang H C, Wang F H, Pan J S. Efficient and robust watermarking algorithm with vector quantisation [J]. Electronics Letters, 2001, 37(13):826~828
18 Huang H C, Wang F H, Pan J S. A VQ-based robust multi-watermarking algorithm [J]. IEICE Trans. Fundamentals, 2002, E85-A(7):1719~1726
19 Lu Z M, Liu C H, Xu D G. Semi-fragile Image Watermarking Method Based on Index Constrained Vector Quantization [J]. Electron. Letter, 2003, 39(1):35~36
20 Pan J S, Hsin Y C, Huang H C, et al. Robust Image Watermarking Based on Multiple Description Vector Quantisation [J]. Electronics Letters, 2004, 40(22):1409~1410
21 Pan J S, Hsin Y C, Huang H C, et al. Multiple Description Watermarking for Lossy Network [J]. In: 2005 IEEE Int'l Symposium on Circuits and Systems, 2005. 3990~3993
22 Lu Z M, Sun Z. An Image Watermarking Method Based on Mean-Removed Vector Quantization for Multiple Purpose [J]. MVA2005, 2005. 558~561
23 Lu Z M, Xu D G, Sun S H. Multipurpose Image Watermarking Algorithm Based on Multistage Vector Quantization [J]. IEEE Transactions on Image Processing, 2005, 14(6):822~831
24 Shieh C S, Huang H C, Wang F H, et al. An Embedding Algorithm for Multiple Watermarks [J]. J. Inf. Sci. Eng., 2003, 19(2):381~395

(上接第 92 页)

一方面,OSD 能够从 QoS 属性那里轻易地得知应用请求的服务模式(服务类型),这个信息对大部分传统的存储系统而言是很渴求却又很难得到的,因为服务模式对于存储设备的相关策略和算法都有很重要的参考价值。另一方面,OSD 掌握着存储空间的分配权,可以通过优化数据在存储设备上的布局以达到提供性能的目的。

综合以上两点,考虑到基于带宽需求的应用请求所具有的周期特性,结合 Buffer 的预取策略,我们可以给出一个基于带宽请求优先的应用请求优化(Bandwidth-based Request Priority, BRP)算法。BRP 算法通过优先考虑基于带宽需求的请求来达到整个系统性能优化的目的。对基于带宽需求的写请求,该算法尽量把该应用请求的数据放在一片连续的空间中;对读请求,则采用 Buffer 的预取策略,把数据提前读出。算法描述如下:

BRP 算法:

(1)系统服务是否中止。如果是,转(5);

(2)判断系统请求队列中是否有基于带宽需求的请求。如果没有,转(4);

(3)摘取系统请求队列中第一个基于带宽需求的请求,判断该请求是不是请求的类型:

如果是 Write 请求,则用一个系统唯一 ID 来标识该请求。接着遍历系统队列,找到同一应用的其它基于带宽需求的请求,并且用同一 ID 标识它,判断在该请求与紧邻的前一个同 ID 的请求合并后,两者间的其它应用请求需求是否仍然可以满足。如果满足,则把该请求提前并且完成合并。转(2);

如果是 Read 请求,则生成一预取数据请求。预取的数据与该请求对应的数据相邻。如果该预取请求对应的数据在 Buffer 中没有命中,则把预取请求附加在该请求上,否则撤销预取请求。转(2);

(4)是否有某 ID 对应的应用请求全部完成,有,则回收该 ID。转(1);

(5)结束

结束语 OSD 对外提供一个全新的、富有表达力的对象级接口,使得基于 OSD 存储系统能够很好地支持应用程序的 QoS 服务;智能性使 OSD 的系统性能得到优化,从而更进一步保证了 OSD 提供稳定的 QoS 服务。本文在分析基于对象存储系统的 QoS 需求之后,对 OSD SCSI 协议集标准进行扩展,接着采取量化的方法对 QoS 进行了分析,给出了应用客户和 OSD 之间 QoS 信息交互的工作过程。最后,紧密结合 OSD 的特性,提出 OSD 上的 QoS 三层模型 Q-Model 和请求优化算法 BRP。在未来的工作中,我们将研究多个 OSD 环境下的 QoS。

参 考 文 献

1 Satran J, et al. Object-based Storage Device Commands. http://www.t10.org/drafts/osd/, Oct. 2004
2 Lustre project, http://www.lustre.org
3 Lu Yingping, Du D H C, Ruwart T. QoS Provisioning Framework for an OSD-based Storage System. IEEE/NASA MSST 2005, Apr. 2005
4 KleinOsowski K, Ruwart T, Lilja J. Communicating Quality of Service Requirements to an Object-Based Storage Device. IEEE/NASA MSST 2005, Apr. 2005
5 Wijayaratne R, Reddy A L N. Integrated QOS management for disk I/O. IEEE, 1999
6 Henson V, Bonwick J, Ahrens M. Existential QoS for Storage. 1999