计算机研究与发展
Journal of Computer Research and Development
ISSN 1000-1239/CN 11-1777/TP
46(3): 492-497, 2009

一种面向大规模存储系统的数据副本映射算法

穆 飞 薛 巍 舒继武 郑纬民
(清华大学计算机科学与技术系 北京 100084)
(清华信息科学与技术国家实验室(筹) 北京 100084)
(mufei@mails.tsinghua.edu.cn)

A Mapping Algorithm for Replicated Data in Large-Scale Storage System

Mu Fei, Xue Wei, Shu Jiwu, and Zheng Weimin
(*Department of Computer Science and Technology, Tsinghua University, Beijing* 100084)
(*Tsinghua National Laboratory for Information Science and Technology (TNList), Beijing* 100084)

Abstract Data mapping is a critical problem in large scale storage systems. Besides high performance and scalability, excellent data mapping algorithm should also provide minimum amount of data migration for keeping balance under dynamic storage environment. Proposed in this paper is a decentralized mapping algorithm for replicated data in large scale storage system. By adopting storage node weighting and consistent hash mechanism, this algorithm could distribute mass storage objects among tens or hundreds of thousands storage devices according to their serving abilities. When the number of storage devices changes, the amount of data migration for data rebalance nearly reaches theoretical lower band. By maintaining only little amount of information, front applications could calculate the location of every data object without consulting the conventional centralized data object mapping table, which greatly improves system scalability. At the same time, the complexity of the data mapping process is quite low. Testing results show that the deviation of the allocated data objects quantity for each storage device from the theoretical value is less than 5%. Testing results also show that the deviation of migrated data objects quantity needed for data rebalance from the theoretical lower band is less than 1%.

Key words dynamic mapping; data replica; scalability; node weight; data migration

摘 要 提出一种适应动态环境、无需数据映射表的多副本数据对象映射算法.该算法引入节点权重,借鉴一致性 Hash 技术,使得海量的数据对象按照节点服务能力在各存储节点间均匀分布;当存储节点个数发生变化时,数据依然在节点间均匀分布,且数据迁移量接近理论下限;只需维护少量数据即可计算得到数据布局,从而有效提高了系统的可扩展性.测试结果表明,算法可使所有节点分配对象个数与理论值偏差小于 5%,节点个数变化时移动数据数量与理论下限偏差小于 1%.

关键词 动态映射; 数据副本; 可扩展性; 节点权重; 数据迁移

中图法分类号 TP302.1

人类社会进入信息时代,每年产生的数据量都以很高的速度持续增长.根据 2007 年的统计结果[1],2006 年产生的新数据总量高达 161 EB,预计 2010 年将产生 988 EB 数据,数据量增长速度达到

收稿日期:2007-09-18;**修回日期**:2008-10-06
基金项目:国家自然科学基金项目(90612018,60473101);国家"九七三"重点基础研究发展规划基金项目(2004CB318205);科技部"十一五"国家科技支撑计划基金重大项目(2006BAA02A17)

每年57%.在这一形势推动下,对大规模存储系统的研究得以广泛开展.我们用数据对象指代存储系统中某一层次的数据单元,它可以理解为传统文件的分片,可以理解为数据块组,也可以理解为面向对象存储系统中的对象.当存储系统的容量达到PB(10^{15}B)规模时,系统中维护的数据对象个数将达到10^9级别,包含数以万计的具有不同服务能力的存储节点,在这种规模下如何将数据对象按照存储节点的服务能力映射到各节点上成为一个很复杂的问题.

随着存储系统规模的增长和存储节点数量的增加,系统内出现数据丢失的概率将会大大增加,数据副本技术成为保障数据可靠性的流行技术,因此数据映射算法必须有效支持数据副本.此外,数据映射算法必须对于频繁变化的存储节点视图提供数据迁移策略,在依然维持数据平衡的前提下最小化数据迁移量.

由于数据规模增长非常迅速,在设计大规模存储系统时必须关注系统的可扩展性.若采用30 b表示数据对象ID,10 b表示存储节点ID,则在采用多副本技术的PB级存储系统中,在不考虑数据压缩的情况下,数据对象到存储位置的映射表所占用的存储空间将达到GB级别.已有研究指出[2],传统的采用映射表结构来记录数据分布情况的做法已经成为限制系统可扩展性的主要瓶颈.理想的映射方式应该无需存储所有数据对象的映射信息,只需保存少量必需信息即可计算出数据对象的布局,从而很大程度提高系统的可扩展性.国内外都在积极开展这种无中心的数据映射机制.

本文提出了一种基于一致性Hash[3]的数据对象映射算法,只需维护少量信息(10 MB左右)即可计算出数据对象的映射信息,是一种无中心的数据映射算法.该算法支持数据副本,能够在异构存储节点环境中达到映射负载平衡,并且在节点加入和退出时数据迁移量接近理论下限.模拟数据表明,该算法对于大规模数据对象环境效果良好,和已有算法相比具有维护简便、定位迅速的优点.

1　相关工作

最原始的数据映射方法是对数据对象唯一标识进行Hash等运算进行映射.这一做法对于静态节点环境简单有效,但发生旧节点退出、新节点加入等情况时将会导致大量数据迁移.SCADDAR方法[4]设计了一种改进的Hash映射算法,目标是在节点数量扩展后只转移尽量少的数据,局限性在于只适应同构的存储节点环境.resource-based striping (RBS)[5]方法考虑到不同规格磁盘所组成的异构存储系统,引入了区分权重的分片方法.但该方法无法保证空间使用均匀,也不适用于节点更新频繁或访问频率变化较大的存储环境.LH* 算法[6]采用分裂存储节点已有数据的方法来适应存储系统规模的扩大,但其划分手段较为机械,导致数据迁移量较大,而且难以适应异构环境.以上算法均没有摆脱数据映射表的限制.

近年来存储系统容量急剧增长,需要能够摆脱数据映射表的映射算法以提高系统的可扩展性.针对面向对象存储系统的CRUSH方法[7]综合了以RUSH为基础的一系列算法,支持副本和节点分组,支持异构环境的数据映射.局限性在于每次确定新数据的映射位置需要在节点组之间递归查找才能确定目的节点,效率较低.在进行数据迁移时容易产生热点区域,而且数据迁移量是理论下限的h倍,h是存储节点树形组织的高度.基于动态区间映射的映射算法[8]将对象ID散列到[0,1]区间,每个存储节点对应一个子区间,以此为依据进行数据映射.局限性在于随着存储节点增减次数的增长,定位所需维护的区间个数将迅速增长,对于大规模系统中节点频频失效的情况区间维护将非常困难.

一致性Hash技术脱胎于网络环境的内存维护领域,在P2P领域中也有一定的应用.Serrento[9]系统将该技术应用到数据对象映射中,但面向环境为小规模同构存储系统,其数据迁移机制开销很大,而且不支持带有节点权重的数据布局.

2　机制描述

2.1　节点分组和权重

对于数量巨大的存储节点,在搭建存储系统时将面临电源布线方案、交换机拓扑结构以及摆放位置布局等多种选择,于是数量众多的存储节点会自然划分成多个分组.对于使用同一电源接线的节点,一旦电源出现故障,这些节点将同时失效,称组内节点的失效相关性强于组外节点的失效相关性.因此在数据映射时,同一数据对象的多个副本应该尽量放置不同的节点分组.为叙述方便,本文后续章节中采用OSD(object storage device)来表示存储节点.定义同组关系g为存储节点集合O上的二元关系.

若节点 OSD_i 与 OSD_j 属于同一节点分组,则有 $OSD_i\ g\ OSD_j$.在实际情况中,节点分组具有不同的层次:例如对所有数据节点分组形成不同的数据中心;在同一数据中心内部,位于不同房间的存储节点形成另一层次的分组.同组关系的组合可以表达丰富的分组含义.限定同一数据对象的 k 个副本必须两两满足不同的分组关系,可以形成不同的数据对象映射策略,在这些策略的作用下系统将处于不同的可靠性级别.

在数据对象映射时还需要考虑存储节点的权重,以此来决定各存储节点所分配的数据量的多少.可以综合评价存储节点的数据处理能力,包括节点存储容量、处理器性能、缓存大小、可提供带宽以及访问延迟等.我们的目标是使系统中各存储节点所包含的数据对象个数与其权重成比例.虽然在面向对象存储系统中数据对象具有可变大小,但考虑到系统中数据对象个数一般非常庞大,因此采用对象个数来近似度量数据量的映射情况是可以接受的.

2.2 基本映射算法

首先定义以下集合.

数据对象集合 OB:所有数据对象组成的集合.每个元素用对象 ID 来表示.

存储节点集合 O:$O=\{OSD_i\mid 1\leqslant i\leqslant ON\}$,其中 OSD_i 是节点标识,ON 是系统中包含的 OSD 总数.

分组关系集合 G:多级同组关系的各种组合结果组成分组关系集合 G.

数据对象映射过程可以定义为函数 $map(OID,k,rule)$,其定义域为 $OB\times\mathbb{N}\times G$($\mathbb{N}$ 为自然数集),值域为 O 的幂集 2^O.其中 OID 为对象 ID,k 为副本阶数,$rule$ 为给定的分组关系.输出结果为 O 的子集 OR,其中的元素两两满足关系 $rule$,且 $\|OR\|=k$.

进一步定义以下集合和函数.

样本空间 Ω:$\Omega=\{n:1\leqslant n\leqslant M,n\in\mathbb{N}\}$.其中 M 是一个很大的自然数,需要大于系统中允许的数据对象个数的上限.Ω 中的元素按升序首尾相接可看作一个圆周 C.

对象散列函数 $H(OID)$:该函数的定义域为对象集合 OB,值域为样本空间 Ω,作用是将对象 ID 随机分散到 Ω 上.

单节点定位种子集合 S_i:对象节点 OSD_i 维护定位种子集合 S_i,S_i 是 Ω 的子集且 $\|S_i\|=\mu\omega_i$,其中 μ 为调节因子,ω_i 为该 OSD 的权重.S_i 中的元素在 Ω 中随机选取,而且对于任意 $i\neq j$,有 $S_i\cap S_j=\varnothing$.

定位种子集合 S:$S=S_1\cup S_2\cup\cdots\cup S_{\|O\|}$,易知 $\|S\|=\mu\sum\omega_i$.

寻找函数 $Lookup(k,c)$:该函数定义域为 $\mathbb{N}\times\Omega$,值域为定位种子集合 S.其中 k 是一个自然数,c 为样本空间 Ω 上的任意元素,也可以理解为圆 C 上的一个点.该函数的值域是定位种子集合 S.功能可以描述为:从点 c 出发,按顺时针方向沿圆周 C 遇到的第 k 个属于集合 S 的元素.

定位函数 $Locate(s)$:该函数定义域为定位种子集合 S,值域为对象节点集合 O.该函数的作用是返回某一定位种子所对应的 OSD.对 $s\in S_i$,有 $Locate(s)=OSD_i$.

基于以上定义,可以给出副本数据对象映射算法的描述:

$map(OID,k,rule)$

1) $Oresult=\varnothing$,$count=0$,$step=0$;

2) While($count<k$) {

$step++$;

$Otmp=Locate(Lookup(step,H(OID)))$;

若 $Otmp$ 与 $Oresult$ 中所有元素满足关系 $rule$,则 $count++$,$Oresult=Oresult\cup\{Otmp\}$; }

3) Return $Oresult$.

2.3 分布均衡机制

定位种子集合 S 中的点都是在 Ω 上随机选取得到,理想的随机函数可使 S 中的点在 Ω 上均匀分布.因此对于某个特定数据对象 Ob,设其在圆周 C 上的映像为点 c,有 $P(Lookup(1,c)\in S_i)=\|S_i\|/\|S\|$,进而有

$$\frac{P(Ob\in OSD_i)}{P(Ob\in OSD_j)}=\frac{\|S_i\|}{\|S_j\|}=\frac{\omega_i}{\omega_j}.$$

当数据对象个数和存储节点规模都很庞大时,依据大数定理,映射结果将更接近理论值.因此可以认为该数据映射算法在大规模存储系统中,使各 OSD 所分配的对象个数与该 OSD 的权重成正比.

我们引入最小分散距离机制使 S 中的点在 Ω 上分散尽量均匀.具体做法是,选取合适的最小距离 d,在生成 S 中的元素时,首先在区间 $[1,M/d]$ 上随机选取元素 s',然后将 $d\times s'$ 作为 S 中的元素,于是 S 中的元素最小间距为 d,在一定程度上分布更为平均.选取数值时应综合考虑 μ 和 d 的关系,需要满足 $\mu\sum\omega_i\ll M/d$.当 d 取值较大时,可以只记录

s',从而记录 S 所需的存储空间减少 $\log_2 d\ /\log_2 M$. 具体取值对算法的影响将在第 3 节讨论.

2.4　动态调整策略

针对系统中存储节点组织情况变化较为频繁的情况,算法需要提供完备的动态调整策略. 设系统中存储节点发生变化前总权重为 W_{old},发生变化后总权重为 W_{new},数据对象总数为 n,则需要迁移的数据对象个数的理论下限为

$$migration_{\text{theory}} = \frac{|\ W_{\text{new}} - W_{\text{old}}\ |}{\max\ (W_{\text{new}}, W_{\text{old}})} \times n. \quad (1)$$

在本文算法框架下,增删存储节点的处理机制非常简便易行. 当系统中增加新节点 OSD_{new} 时,确定其权重 ω_{new},生成它对应的单节点定位种子集合 S_{new},其中元素个数为 $\mu\omega_{\text{new}}$. 在进行数据迁移时,考察某一特定数据对象 Ob,副本阶数为 k,设其在圆周 C 上所对应的点 c 顺时针前向符合分组规则的 k 个点为 $\{s_1, s_2, \cdots, s_k\}$,与之对应保存有 Ob 的对象节点组成集合 $\{OSD_1, OSD_2, \cdots, OSD_k\}$. 若 S_{new} 中存在位于 s_1 到 s_k 之间的点,则 Ob 的第 k 个副本需要由节点 OSD_k 迁移至 OSD_{new}. 若 OSD_{new} 中不存在位于 s_1 到 s_k 之间的点,则该数据对象无需进行数据迁移. 在这一机制作用下,当加入新节点时,数据迁移的目的节点只可能是新加入的节点 OSD_{new},在旧节点之间互相没有数据流动. 调整完毕后,由上节讨论结果可知,各对象节点含有的数据对象个数与其权重成正比,达到数据分布均衡,而且数据迁移量为理论最小.

对于节点失效和主动移除节点的情况处理方法类似. 设 OSD_f 发生失效,它包含的某一数据对象所对应的点 c 在圆周 C 上顺时针前向的 k 个符合分组规则的点为 $\{s_1, s_2, \cdots, s_k\}$,其中 $s_f \in S_f$. 进行数据修复时,从集合 $\{s_1, s_2, \cdots, s_k\}/\{s_f\}$ 所确定的节点中随机选取一个节点作为数据复制源节点,数据复制目的节点为从 c 出发,顺时针前向第 $k+1$ 个符合映射规则的点所确定的节点. 由 S 中的节点生成规则可知,数据复制的目的节点和源节点都可能有很多个,从而避免了数据修复时出现热点节点. 该方法对于一些系统采用的直到加入新替代节点后才进行数据修复的机制也同样适用. 由于新节点一般会具有较高的容量和性能,因此新节点对应的单节点定位种子集合也会包含较多的元素,最终将导致新节点在数据复制时获得更多的数据对象.

数据迁移会占用较多的存储节点读写带宽和网络带宽,为了尽量减少对前端应用的影响,可以采用惰性迁移机制. 该机制不主动进行数据迁移,而是当数据对象被访问到时才根据数据映射策略决定是否需要迁移. 这种机制的好处是隐藏了主动数据迁移的数据定位时间,在存储节点频繁加入系统的情况下可以减少数据迁移次数,从而减少数据迁移数量.

3　测试结果

我们模拟生成海量数据对象,按照文中算法将其映射到大量存储节点上,统计映射结果以达到评价映射算法的目的.

首先测试算法基本功能. 取 $M=2^{40}$,存储节点权重 ω 的取值范围为 1～32,系统中的节点随机分配权重. 随机生成数据对象集合 OB 将其中元素映射在存储节点上. 对 OSD_i,统计其分配到的数据对象个数 n_i. 定义 $\eta_i = (n_i/\|OB\|)/(\omega_i/\sum\omega_i)$. 理想情况下所有 η_i 均为 1,但算法的实现过程会带来一定的随机误差. 我们通过统计 η_i 值落在[0.9,1.1]之上的存储节点个数百分比 p 来度量负载均衡程度. p 越接近 1,说明在映射算法作用下各节点负载越平衡. 取 $\mu=8, d=2^{20}$. 考察系统的可扩展性,固定节点个数为 1024 个,当系统数据对象总数由 10^5 增长至 10^7 负载情况如图 1 所示:

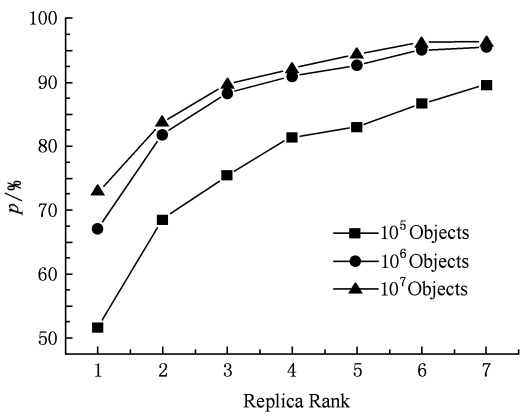

Fig. 1　Map result with varied object number.

图 1　系统包含对象个数对映射结果的影响

如图 1 所示,随着系统中对象个数增多,映射结果越来越均衡,说明本文算法对大规模存储系统效果良好. 图 1 还表明算法在数据对象个数较少时依然保持着较好的映射结果,可见算法对于不同规模的存储系统适应性良好. 此外,图 1 中显示随着数据副本个数的增长,映射结果表现得越为平衡. 这表明算法非常适合多副本存储系统.

接下来考察存储设备个数变化时，维持数据始终平均分布所需迁移的数据量. 取 $\mu=32$，$d=2^{20}$，固定对象总数为 10^7，存储节点初始值为 1024，每步增加 128 个存储节点. 统计每次节点数量改变引起的实际数据对象迁移个数 $migration_{real}$，通过式(1)计算数据迁移理论最小值 $migration_{theory}$，计算 $|1-migration_{real}/migration_{theory}|$ 来衡量算法的有效性. 测试结果表明，实际数据迁移量与理论下限 $\Delta\omega/W$ 相比偏差不超过 1%. 与 CRUSH 机制作用下的迁移量 $h\Delta\omega/W$ 相比明显减少. 如图 2 所示：

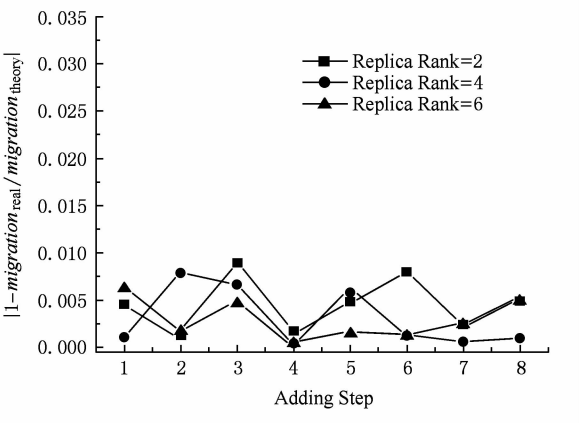

Fig. 2 Deviation of objects migration quantity from theoretical lower band.

图 2 迁移数据量与理论下限偏差

接下来考察算法参数. 对于节点的定位种子集合，如果所含元素个数太少将会引发较大随机误差，随着元素个数增加可以在一定程度上降低随机误差，但会增加定位种子集合维护难度，影响数据定位速度. 考察参数改变对映射结果的影响，取副本阶数 $k=5$，ω 取值范围为 1～16，1024 个存储节点，10^7 个数据对象，取不同的 μ 值和 d 值，度量对应的 p 值. 测试结果如图 3 所示.

由图 3 可以看出，增大 μ 值和 d 值对于减小随机误差都具有良好的作用. 特别地，在 $M=2^{40}$，$d=2^{20}$，当 μ 值取到 32 时，分配对象个数与理论值偏差小于 10%的节点达到 100%；μ 值取到 64 时，分配对象个数与理论值偏差小于 5%的节点也达到 100%.

根据测试结果，在记录定位种子集合 S 时，可取 $\mu=32$，$M/d=2^{40}$，节点权重取值范围为 1～64 之间. 这样对于数千节点的存储系统，维护定位信息约十几兆字节，完全可以置于服务器内存中. 与动态区间映射算法相比，其定位信息存储量与系统发生的

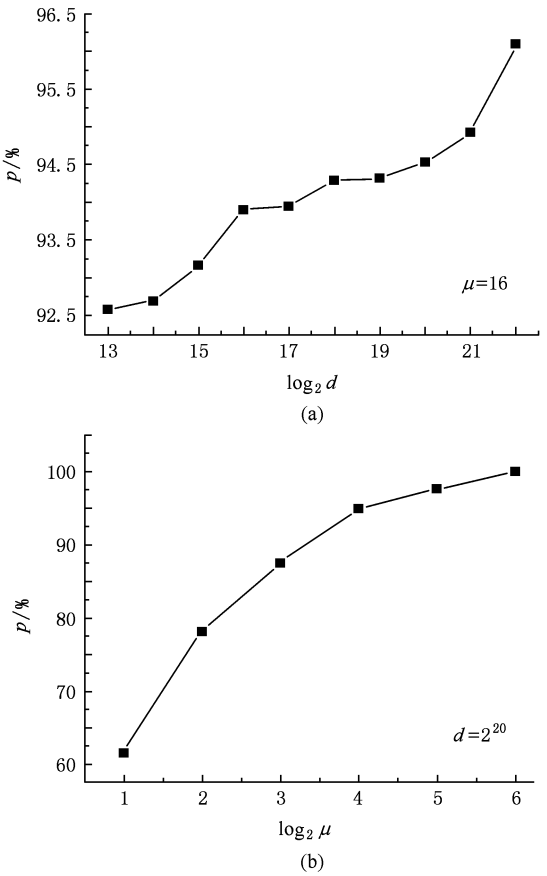

Fig. 3 Map result with varied algorithm parameters. (a) Map result with fixed μ and varied d and (b) Map result with fixed d and varied μ.

图 3 算法参数取值对映射结果的影响. (a) 固定 μ 值，变化 d 值对映射结果的影响；(b) 固定 d 值，变化 μ 值对映射结果的影响

节点视图变化次数和存储节点个数乘积成正比. 于是随着系统使用时间的增长，其所需存储空间将超过本文算法.

通过有效组织定位目录的节点，每次定位操作运算次数为 $O(\log(\|S\|))$次，在一般情况下通过几十次内存操作可以完成. 对于 RUSH 族算法，需要遍历存储节点，根据每次生成的随机数来确定数据的放置位置，显然本文算法具备更高的性能. 而对于其他需要映射表来定位数据的算法需要在对象映射表中查找定位信息，系统中的数据对象数量规模相比要庞大的多，因此查找耗时也会增大许多.

4 总 结

在大规模存储系统中，存储节点组织具有异构性和动态性. 本文提出了一种无中心的数据对象映

射算法.该算法支持数据副本、节点权重以及节点分组,能够使数据节点分配到的数据对象个数和其服务能力成正比.在存储节点视图发生动态变化时,需要迁移的数据量接近理论下限.

参 考 文 献

[1] Gantz J F. The expanding digital universe: A forecast of worldwide information growth through 2010 [R/OL]. [2007-08-06]. http://www.emc.com/about/destination/digital_universe/

[2] Weil S A, Brandt S A, Miller E L, *et al*. Ceph: A scalable, high-performance distributed file system [C] //Proc of OSDI '06. Berkeley, CA: USENIX Association, 2006: 307-320

[3] Kager D, Lehman E, Leighton F, *et al*. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the world wide Web [C] //Proc of the 29th Annual ACM Symp on Theory of Computing. New York: ACM, 1997: 654-663

[4] Goel A, Shahabi C, Yao D S, *et al*. SCADDAR: An efficient randomized technique to reorganize continuous media blocks [C] //Proc of the 18th Int Conf on Data Engineering (ICDE'02). Piscataway, NJ: IEEE, 2002: 473-482

[5] Ding Jenwen, Huang Yuehmin. Resource-based striping: An efficient striping strategy for video servers using heterogeneous disk-subsystems [J]. Multimedia Tools and Applications, 2003, 19(1): 29-51

[6] Litwin W, Schwarz T. LH*RS: A high-availability scalable distributed data structure using reed solomon codes [C] //Proc of the 2000 ACM SIGMOD Int Conf on Management of data. New York: ACM, 2000: 237-248

[7] Weil S A, Brandt S A, Miller E L, *et al*. CRUSH: Controlled, scalable, decentralized placement of replicated data [C] //Proc of Super Computing'06. New York: ACM, 2006: 31-42

[8] Liu Zhong, Zhou Xingming. A data object placement algorithm based on dynamic interval mapping [J]. Journal of Software, 2005, 16(11): 1886-1893 (in Chinese)
(刘仲,周兴铭. 基于动态区间映射的数据对象布局算法 [J]. 软件学报, 2005, 16(11): 1886-1893)

[9] Tang Hong, Gulbeden A, Zhou Jingyu, *et al*. A self-organizing storage cluster for parallel data-intensive applications [C] //Proc of the 2004 ACM/IEEE Conf on Supercomputing. New York: ACM, 2004: 52-64

Mu Fei, born in 1980. Ph. D. candidate. His main research interests include storage network, object storage and high performance computing.

穆　飞,1980 年生,博士研究生,主要研究方向为存储网络、面向对象存储、高性能计算等.

Xue Wei, born in 1974. Ph. D., and associate professor. Member of China Computer Federation. His main research interests include numerical parallel algorithm, large scale parallel and distributed system, and network storage.

薛　巍,1974 年生,博士,副研究员,中国计算机学会会员,主要研究方向为数值并行算法、大规模并行/分布式系统和网络存储.

Shu Jiwu, born in 1968. Professor, Ph. D. supervisor. Senior member of China Computer Federation. His main research interests include storage networks, parallel and distributed computing, and cluster systems.

舒继武,1968 年生,教授,博士生导师,中国计算机学会高级会员,主要研究方向为存储网络、并行与分布式计算、集群系统等.

Zheng Weimin, born in 1946. Professor, Ph. D. supervisor. Fellow member of China Computer Federation. His main research interests include computer architecture, parallel computing, compiler technology, grid computing and network storage.

郑纬民,1946 年生,教授,博士生导师,中国计算机学会副理事长,主要研究方向为计算机体系结构、并行计算、编译器技术、网格计算和网络存储.

Research Background

In large-scale storage systems, mass data objects need to be mapped onto tens or hundreds of thousands of storage devices. High performance, scalability, availability and maintenance are key factors for data mapping algorithms. Supported by the National Nutural Science Foundation of China (Nos. 90612018 and 60473101), the National 973 Basic Research Program of China (No. 2004CB318205), and the R&D Infrastructure and Facility Development Program in the 11th Five-year Plan (No. 2006BAA02A17), we proposed a decentralized mapping algorithm for replicated data in large scale storage systems, which could distribute mass storage objects onto numbers of storage devices according to their serving abilities by maintaining only little amount of information.