ISSN 1000-9825, CODEN RUXUEW
Journal of Software, Vol.21, No.12, December 2010, pp.3175−3185
doi: 10.3724/SP.J.1001.2010.03706

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

基于聚类和一致Hash的数据布局算法*

陈 涛+, 肖 侬, 刘 芳, 付长胜

(国防科学技术大学 计算机学院,湖南 长沙 410073)

Clustering-Based and Consistent Hashing-Aware Data Placement Algorithm

CHEN Tao+, XIAO Nong, LIU Fang, FU Chang-Sheng

(School of Computer, National University of Defense Technology, Changsha 410073, China)

+ Corresponding author: E-mail: lovely696521@163.com

Chen T, Xiao N, Liu F, Fu CS. Clustering-Based and consistent hashing-aware data placement algorithm. *Journal of Software*, 2010,21(12):3175−3185. http://www.jos.org.cn/1000-9825/3706.htm

Abstract: Large-Scale network storage systems are confronted with the big challenge of efficiently distributing data among storage devices. It's necessary to design an efficient, fair and adaptive data placement algorithm. This paper has developed an algorithm CCHDP (clustering-based and consistent hashing-aware data placement) to distribute data over heterogeneous devices in the systems. It combines clustering algorithm and consistent hashing, saving much memory space by avoiding extra virtual devices. The analysis and experiments show that CCHDP can notonly assign data evenly among devices and adapt well with the additions or departures of devices for the number of data moved is nearly equal to the optimal amount in the events of devices changes. Moreover, CCHDP is time efficient with little memory overhead.

Key words: data placement; clustering algorithm; consistent hashing; fair; self-adaptive

摘 要: 如何有效地对数据进行布局是大规模网络存储系统面临的重大挑战,需要一种能够自适应存储规模变化、公平有效的数据布局算法.提出的 CCHDP(clustering-based and consistent hashing-aware data placement)算法将聚类算法与一致 hash 方法相结合,引入少量的虚拟设备,极大地减少了存储空间.理论和实验证明,CCHDP 算法可以按照设备的权重公平地分布数据,自适应存储设备的增加和删除,在存储规模发生变化时迁移最少的数据量,并且可以快速地定位数据,对存储空间的消耗较少.

关键词: 数据布局;聚类算法;一致 hash;公平;自适应

中图法分类号: TP315 **文献标识码**: A

图灵奖获得者Jim Gray提出了一个新的经验定律:网络环境下,每 18 个月产生的数据量等于有史以来数据量之和[1].面对如此浩瀚的数据量,包含成百上千甚至成千上万个设备的大规模网络存储系统应运而生.如何在

* Supported by the National Natural Science Foundation of China under Grant Nos.60736013, 60903040 (国家自然科学基金); the National High-Tech Research and Development Plan of China under Grant Nos.2006AA01A106, 2006AA01A118 (国家高技术研究发展计划(863)); the Program for New Century Excellent Talents in University of China under Grant No.NCET-08-0145 (新世纪优秀人才支持计划)

Received 2008-11-06; Accepted 2009-07-17; Published online 2010-04-12

这些设备中有效地存放PB规模的数据,是一个具有挑战性的问题.不合理的数据布局使得大量的存储空间白白浪费,而且用户需要花费大量的时间寻找所请求的数据,因而数据布局算法在大规模网络存储系统中非常重要.

均衡地使用设备容量可以平衡I/O负载[2].为了尽可能地发挥每个设备的存储能力,使得存储系统的整体性能最大化,需要根据设备的容量以及带宽公平地分配数据.而且,大规模网络存储系统会定时添加新的设备以及删除旧的设备,或者旧的设备发生失效等.为了让存储系统的性能最大化,需要重新公平地分布数据.存储系统的某些应用是 24 小时运转的,迁移大量的数据必然会消耗大量的带宽,降低存储系统对应用的服务质量,甚至影响整个系统的可用性.为了不影响系统的存储服务,再次分布的数据量要尽可能地少.因而,存储系统的数据布局算法需要动态自适应存储规模的变化,在系统规模变化时迁移尽可能少的数据.同时,必须保证布局算法能够利用很少的时间和空间信息计算出某个数据的存放位置.

本文提出的 CCHDP(clustering-based and consistent hashing-aware data placement)算法可以在异构的设备集合中按照设备的权重公平地分布数据,自适应存储规模的变化,在存储规模变化时迁移最少的数据量,而且能够在较短的时间内计算数据的位置,仅需要引入少量的虚拟设备,大大减少了存储空间.

本文第 1 节描述相关工作.第 2 节给出问题描述以及相关概念.第 3 节提出基于聚类和一致 hash 的数据布局算法 CCHDP.第 4 节对算法进行理论分析,证明算法是公平和自适应的,同时定性地分析了算法定位数据的时间复杂度和空间复杂度.第 5 节给出实验与结果分析.第 6 节总结全文.

1 相关工作

目前,大部分的布局算法面向同构的存储系统,即所有的设备性能相同.最具代表性的同构布局算法来源于 1997 年Karger等人提出的一致hash机制[3].该机制为每个设备虚拟出$k\log|N|$个设备,其中,k为常数,N为设备空间的设备总数.虚拟设备的引入,使得一致hash机制可以满足数据布局的公平性.当存储系统规模发生变化时,其迁移的数据量是最优数据量的 1 倍,具有很好的动态自适应特性.一致hash机制在Sorrento系统以及Chord协议中得到了广泛的应用[4,5].

一致hash机制只适用于同构的存储系统,而实际的大规模网络存储环境通常是异构的,存在不同设备之间容量和带宽差异很大的情况.同构的一致hash方法在异构环境下使用时,无法按照设备的权重公平地分配数据.设备的权重可以表示为其容量、带宽或者是两者的结合.为了适应异构的存储环境,一个简单的方法是对同构的一致hash算法进行改进.按照设备的权重分配虚拟设备,权重多的设备包含更多的虚拟设备.但是,一致hash的简单改进需要大量的存储开销.设系统中设备的最小权重为$w_{\min}$,分配给该设备的虚拟设备数是$k\log|N|$.那么对于权重为w_i的设备需要分配$\frac{w_i}{w_{\min}}k\log|N|$个虚拟设备.设备集合$D_0$中的设备总共需要$\sum_{i=1}^{n}\frac{w_i}{w_{\min}}k\log|N|$个虚拟设备,其中,$n$为$D_0$中的设备总数.因而,区分开所有的虚拟设备需要$\log\left(\sum_{i=1}^{n}\frac{w_i}{w_{\min}}k\log|N|\right)$位.在权重差异较大的异构存储系统中,该方法需要引入大量的虚拟设备,占用大量的存储空间,这是存储系统无法容忍的.

Brinkmann等人[6]首次将异构环境下的数据布局问题转换到同构环境下.按照设备的剩余容量来分层,每层中所有设备的容量是相同的.从 0 层开始,在每层中使用同构布局策略,公平地将数据分布到设备上.当存储规模发生变化时,该方法迁移数据的过程非常复杂,而且定位数据的时间长,自适应很差.后来,Brinkmann等人[7]再次提出了两种数据布局算法:SHARE和SIEVE.SHARE方法公平性差,仍然需要引入虚拟设备;SIEVE方法具有良好的公平性,但是使用了大量的区间,存储规模大幅度变化时需要重新分配大量区间.两种方法迁移的数据量均是最优数据量的 2 倍,自适应性比较差.

Schindelhauer等人[8]提出的线性方法和对数方法将设备权重引入到一致hash的距离函数中.线性方法和对数方法具有良好的自适应性特性,迁移的数据量是最优数据量的 1 倍.但是,线性方法为了保证公平性,仍然需要引入虚拟设备.而且,两种方法均需要在较长的时间计算数据的位置,占用大量的区间.

Honicky等人[9]提出了在多个异构的子集群间公平地分布数据.算法可以保证数据在集群的设备间按照其

权重均匀分布.在集群增加时,迁移的数据量是最优的.但是,当集群删除时,系统的数据需要全部重新进行组织.在一个集群内部添加和删除单个设备时,集群内所有的数据需要重新组织.因而,算法在删除子集群、增加或者删除单个设备时自适应性非常差.后来,Honicky等人[10]对其在文献[9]中提出的算法进行改进,设计了一个算法族RUSH,但整个算法族在增加和删除单个存储设备时自适应性非常差.Weil等人[11]在RUSH算法[10]的基础上进行改进,提出了一个用户可以控制的数据布局算法CRUSH,已经应用于著名的存储系统Ceph[12],用户可以根据存储系统的存储规模对 4 类不同的设备对象进行组合来定制布局策略.但是,布局策略在最底层必须使用标准hash机制,当设备数发生变化时需要重新组织所有数据,因而算法不能自适应单个设备添加或者删除的情况.

某些著名的系统如NASD[13],Lustre[14],Panasas[2,15],GPFS[16]等使用伪随机的方法或者基于设备可用容量的启发式方法来分配数据.这些系统在设备集合变化时,一般不会迁移数据来保证数据再次分布的公平性,因而它们的自适应性比较差.Panasas[2,15]使用了主动的平衡机制,通过迁移数据以及修改匹配表来重新分布数据.这些系统将数据与设备的匹配信息保存在表中,因而在定位数据时需要查询匹配表,消耗了大量的时间和空间.

国防科技大学的刘仲等人[17]、清华大学的王迪等人[18]、华中科技大学的王芳等人[19]、中国科学院计算技术研究所的谈华芳等人[20]也在数据布局方面做了相关工作.文献[17]提出了基于动态区间映射的数据布局算法,该方法定位数据的时间以及区间数均与存储设备的数量相关.该方法不是时间和空间高效的.文献[18]通过查询表定位数据,需要很长的时间.文献[19]没有考虑数据布局的自适应性,而且对定位数据的时间没有进行任何分析.文献[20]提出了一个有效的数据布局算法,但是其自适应性以及再次分布的公平性较差.

本文提出的 CCHDP 算法将聚类算法和一致 hash 结合起来.首先将设备集合按照权重的大小进行聚类,使得同一类别中的设备权重差别小于预设的值,避免在类的内部使用一致 hash 时引入大量的虚拟设备,节省了大量的存储空间.聚类完成后,按照类的权重将[0,1]区间划分为多个小区间,为每个类分配小区间,将落入区间的数据分配给其对应的类,从而保证数据公平地分配给每个类.而且,区间数只与类数相关,与文献[7,8,17]相比,大大减少了存储空间.每个类内部使用一致 hash 方法将数据公平地分配到类中的设备上.由于类内设备权重差异很小,因而仅需要引入很少的虚拟设备,解决了对一致 hash 进行简单改进所带来的空间浪费问题.存储规模变化时,迁移的数据量等于理论上应该迁移的数据量,比文献[6,9–11,20]迁移的数据量少.同时,CCHDP 的聚类算法可以适合任意异构的设备集合.文献[9–11]要求每个集群内部的设备必须是相同的,因而,CCHDP 比文献[9–11]的设备管理更加灵活.而且,CCHDP 在每个类内部使用一致 hash 机制,支持单个设备的增加和删除.定位数据的时间只与类数相关,远远小于文献[6–8,13–16,18]定位数据的时间.

2 问题描述

在异构的海量存储系统中存放大规模的数据可以抽象为一个数学模型.我们将设备和数据分别作为一个集合,问题抽象为如何建立两个集合之间的映射关系.设设备空间D为$\{d_1,\ldots,d_N\}$,其中,N为设备的总数,d_i表示一个设备元素,其权重可以表示为容量、带宽或者是两者的结合.hash函数$h_1:D\to[0,1]$将D中的设备均匀地分布到单位区间上.设存储系统中的当前设备集合为$D_0=\{d_1,\ldots,d_n\}$,其中,$n\leqslant N$,$D_0\subseteq D$.设存储设备$d_i\in D_0$的相对权重为w_i,那么,n个设备的权重分别为$\{w_1,\ldots,w_n\}$,且$\sum_{i=1}^{n}w_i=1$.设存储系统中的数据集合为$X_0=\{x_1,\ldots,x_m\}$,其中,m表示数据的总数,$x_i\in X_0$表示一个数据元素.函数$f_0:X_0\to D_0$将数据集合X_0映射到设备集合D_0.数据布局算法A可以表示为一个二元函数$A(x,f)$,参数$x\in X_0$,参数f表示一个匹配函数,且$A(x,f)=f(x)$.数据x分配给由$A(x,f)$值标识的设备.

定义 1(公平性). 设存储系统的设备集合$D_0=\{d_1,\ldots,d_n\}$,设备$d_i\in D_0$的相对权重为w_i,数据集合$X_0=\{x_1,\ldots,x_m\}$,函数$f_0:X_0\to D_0$将数据集合X_0映射到设备集合D_0,$\forall x\in X_0$,布局算法A将数据x分配给$d_i\in D_0$的概率为$p(A(x,f_0)=d_i)$.$\forall\varepsilon>0$,若$|p(A(x,f_0)=d_i)-w_i|<\varepsilon$,则算法$A$是公平的[7].

算法的公平性使得数据$x\in X_0$分配给设备$d_i\in D_0$的概率无限接近d_i的相对权重,保证数据公平地分配给每个设备.

定义 2(自适应性). 设存储系统的设备集合$D_0=\{d_1,\ldots,d_n\}$,设备$d_i\in D_0$的相对权重为w_i,数据集合

$X_0=\{x_1,\ldots,x_m\}$,函数$f_0:X_0\to D_0$将数据集合X_0映射到设备集合D_0.若布局算法A满足以下两个条件,则布局算法A是自适应的[21]:

(1) 当前设备集合D_0变为$D^+=\{d_1,\ldots,d_{n+1}\}$,设函数$f_+:X_0\to D^+$将数据集合$X_0$映射到当前设备集合$D^+$.$\forall x\in X_0$,若$A(x,f_+)=f_+(x)\in D_0$,则$A(x,f_+)=A(x,f_0)$;

(2) 当前设备集合D_0变为$D^-=\{d_1,\ldots,d_{i-1},d_{i+1},\ldots,d_n\}$,设函数$f_-:X_0\to D^-$将数据集合$X_0$映射到当前设备集合$D^-$.$\forall x\in X_0$,若$A(x,f_0)=f_0(x)\in D^-$,则$A(x,f_0)=A(x,f^-)$.

当存储系统规模发生变化时,布局算法的自适应性使得迁移的数据只从未变化的设备移到新增的设备上或者从被删除的设备上移到未变化的设备上,而未变化的设备之间没有数据的迁移,从而保证迁移的数据量是最少的.如增加设备d_{n+1}时,设备集合D_0变为$D^+=\{d_1,\ldots,d_{n+1}\}$时,数据只从$D_0$的设备迁移到$d_{n+1}$上,而$D_0$的设备之间没有数据的迁移.当删除设备$d_i$时,设备集合$D_0$变为$D^-=\{d_1,\ldots,d_{i-1},d_{i+1},\ldots,d_n\}$时,数据只在$d_i$和$D^-$中的设备之间迁移,而$D^-$中的设备之间没有数据的迁移.这样的方法使得数据的迁移量等于增加或者删除的设备上的数据量,而且在不增加新规则的情况下,数据迁移之后系统仍然可以使用算法A计算数据的位置.因而,算法A可以自适应存储系统的规模变化.存储系统的规模变化对用户来说是透明的,用户只需要修改系统规模的配置文件,然后根据算法A可以正确定位数据.

3 CCHDP 算法

在同构的一致 hash 机制中,为了解决异构的数据布局问题,需要根据设备的权重引入相应的虚拟设备.若设备之间的权重差异很大,则需要引入大量的虚拟设备.在最坏的情况下,如果系统中某个设备的权重非常小、与其他设备的权重差异非常大,那么引入的虚拟设备数是系统的内存空间无法容忍的.本文提出的 CCHDP 算法由 3 步组成:

(1) 首先采用聚类算法对设备集合进行分类,使得每个类中设备的权重差异在预设的范围内;

(2) 聚类完成后,类间的布局机制按照类的权重将[0,1]区间划分为多个子区间,为每个类分配一个子区间,将落入某个子区间的数据分配给相应的类;

(3) 每个类的内部布局机制使用一致 hash 方法进行数据的再次分配,将数据布局到具体的设备上.

下面分别介绍设备的聚类算法、类间的分配机制以及类内的布局机制.

3.1 设备的聚类算法

定义 3(设备之间的距离). 设备集合D_0中的两个元素d_i和d_j的权重分别为w_i和w_j,则d_i和d_j的距离为$w_{ij}=|w_i-w_j|$.

定义 4(设备到类的距离). 设设备类为S,其聚类中心为d_i,其中$d_i\in S$,则设备d_j到类S的距离等于设备d_j到d_i的距离.

聚类算法的目标是使得每个类中的设备到其聚类中心的距离小于一个预设的值.按照定义 4 中的距离公式计算设备到聚类中心的距离.设类内距离阈值为T,将设备到聚类中心的距离与T进行比较,判断设备所属的类或者将设备作为一个新类的中心.首先取任意的一个设备d_i作为第 1 个类S_1的聚类中心,如取d_1作为类S_1的中心,根据定义 4 计算下一个设备d_2到D_1的距离.如果该值小于等于T,则将d_2归到类S_1中;否则,将d_2作为新类S_2的中心.设已有k个聚类中心,计算未分类的设备d_i到k个聚类中心的最小距离,如果该值大于T,则将设备d_i作为新类S_{k+1}的中心;否则,将设备分配给k类中距离其最近的类.重复上述过程,直到将所有的设备划分到各个类中.

3.2 类间分配机制

聚类算法将设备集合$D_0=\{d_1,\ldots,d_n\}$划分为g个类的集合$C=\{D_1,\ldots,D_g\}$.设类D_j为$\{d_i^j;i=1,2,...,n_j\}$,其中$1\le j\le g$,则$\sum_{j=1}^{g}n_j=n$.设类D_j的权重为w_j,则$w_j=\sum_{i=1}^{n_j}w_i$,g个类总的权重$\sum_{i=1}^{g}w_j=1$.问题转化为在异构的类集合C中分布数据.

3.2.1 类间分配算法

为了在类集合$C=\{D_1,\ldots,D_g\}$中公平地分配数据,将[0,1]区间按照C中各个类的权重划分为多个子区间,类D_j所在的子区间为$I_j=\left[\sum_{i=1}^{j-1}w_i,\sum_{i=1}^{j}w_i\right)$.类间分配算法BCDP(data placement between classes)使用hash函数$h_2:X\to[0,1]$将数据$x\in X_0$映射到[0,1]区间.若$h_2(x)\in I_j$,则将x分配给子区间I_j.由概率论可知,落入每个区间的数据量与区间大小成正比,因而数据可以在类集合$C=\{D_1,\ldots,D_g\}$中均匀地分布.设函数$f_c:X_0\to C$表示将数据集合X_0映射到类的集合C,则BCDP可以表示为函数$BCDP(x,f_c)$,其中,$BCDP(x,f_c)=f_c(x)$.f_c的流程如图 1 所示.

```
Input: x, {D1,...,Dg}, {X1,...,Xg};
Process:
For (j=1; j<=g; j++)
//Determine the sub interval Ij to which x belongs.
//If there are multiple sub intervals related with Ij,
//then traverse all of these sub intervals.
   if (h2(x)∈Ij)
   then add x to Xj related with Ij;
   End if
End for
Output: class Dj related with Ij;
```

Fig.1 Function of data assignment between classes

图 1 类间的数据分配函数

3.2.2 类间数据迁移

在扩展存储系统时,可能一次性批量增加设备.通常来说,这些设备的容量以及性能很相似,可以将其归为一个新类,因而类间的分配机制需要自适应存储规模批量增加的情况.由于类中的设备不是按照物理位置来分配的,因而将整个类中的设备同时删除的情况不具有一般性,我们在这里不作考虑.单个设备增加和删除的情况将会在类内数据迁移中描述.下面讨论设备批量增加时的数据迁移过程.

初始情况下,聚类算法将系统中的设备集合分为g个类,[0,1]区间被划分为g个子区间.批量增加的存储设备作为一个新的类D_{g+1}.由于引入了类D_{g+1},系统中所有类的权重均发生变化,因而按照新的权重为类重新分配区间.设类D_j的新权重为w'_j,其中,$1\le j\le g$.为了满足数据分布的公平性,则类D_j需要迁移$(w_j-w'_j)m$个数据到类D_{g+1}上,其中,m表示数据总量.将类D_j的区间I_j分割成$\left[\sum_{i=1}^{j-1}w_i,\sum_{i=1}^{j-1}w_i+w'_j\right)$和$\left[\sum_{i=1}^{j-1}w_i+w'_j,\sum_{i=1}^{j}w_i\right)$,其中$\left[\sum_{i=1}^{j-1}w_i+w'_j,\sum_{i=1}^{j}w_i\right)$区间分配给类$D_{g+1}$,则类$D_{g+1}$对应的子区间$I_{g+1}$为$\bigcup_{j=1}^{g}\left[\sum_{i=1}^{j-1}w_i+w'_j,\sum_{i=1}^{j}w_i\right)$,将落入这些区间的数据都迁移到类$D_{g+1}$上.这样,数据重新公平地分布在$g+1$个类上.从上述迁移的过程可以看出,只有旧类的数据迁移到新类上,旧类之间没有数据的迁移,迁移的数据量等于新增类应该得到的数据量.因而,类间分配算法可以有效保证公平性,具有很好的自适应性.依次增加新的类时,按照权重分割现有类的区间,将每个类多余的区间分配给新的类,同时将落入这些区间的数据迁移到新增类上.

3.3 类内布局机制

类间的分配机制将数据公平地分布到各个类中,而且在增加新类时,迁移最优的数据量让数据分布再次达到公平.类的内部同样需要一种布局机制将分配给各个类的数据公平地分布到类的各个设备中,而且在设备规模变化时迁移最少的数据量.

3.3.1 类内布局算法

对存储系统中的设备集合进行聚类后,每个类的设备容量相差不大,因而在类内使用一致hash机制不会引入太多的虚拟设备.类内布局算法ICDP(data placement in class)采用一致hash机制的标准[3].设类D_j的区间为I_j,根据BCDP算法得到落入类D_j的数据集合为X_j.X_j中数据元素的hash值聚集于[0,1]上的区间I_j,处于局部聚集状

态,而不是均匀分布在[0,1]上.为了保证公平性,使用$h_3:X_j\to[0,1]$重新计算X_j内数据元素的hash值,使得X_j中的数据元素在[0,1]区间上公平地分布.$\forall d_i^j\in D_j$,使用$h_1:D_j\to[0,1]$将d_i^j映射到[0,1]区间的某个点.由于D_j的元素在[0,1]区间上分布不均匀,因而需要为每个设备引入相应的虚拟设备.根据聚类算法可知,D_j中任意两个设备的权重差值小于$2T$.我们忽略设备权重的差异,则每个设备引入的虚拟设备数是相同的.由文献[3]可知,为了保证公平性,每个设备引入的虚拟设备数为$k\log|N|$,其中,N为设备空间$D=\{d_1,\ldots,d_N\}$的设备数,k为一个常数.将每个设备复制$k\log|N|$次,然后使用h_1将每个虚拟设备映射到[0,1]区间.$\forall x\in X_j$,设x到某个设备d_i^j的距离为$|h_3(x)-h_1(d_i^j)|$.计算x到每个设备包括虚拟设备的距离,将其分配给离其最近的设备.设函数$f_j:X_j\to D_j$表示将数据集合X_j映射到设备集合D_j,则ICDP可以表示为函数$ICDP(x,f_j)$,其中,$ICDP(x,f_j)=f_j(x)$.f_j的流程如图2所示.

```
Input: x∈X_j,  D_j{d_i^j; i = 1,2,...,n_j};
Process:
temp=1;
num=0;
for (i=1; i<=n_j; i++)
   for (m=1; m<= klog|N|; m++)
     if (|h_3(x) − h_1(d_i^j)| <temp)
//temp is the temporary nearest distance between x and the device in D_j
          then temp=|h_3(x) − h_1(d_i^j)|;
               num=i;   //num is the temporary nearest number of device
     End if
   End for
End for
Output: device  d_num^j
```

Fig.2 Function of data placement within a class

图2 类内的数据布局函数

3.3.2 类内数据迁移

当存储系统规模发生变化时,例如增加新的存储设备$d_{n_j+1}^j$,使用$h_1:D_j\to[0,1]$将$d_{n_j+1}^j$映射到[0,1]区间的某个点.对于该点的左邻居以及右邻居上的数据x,如果x到该点的距离更近,则将其迁移到新设备上.这样迁移的结果与布局算法的计算结果是一致的.同理,当删除旧的存储设备d_i^j时,对于该设备上的数据,计算其到左邻居以及右邻居的距离,将该数据迁移到较近的邻居上.对于新增设备或者被删除设备的虚拟设备,重复同样的过程.对虚拟设备的处理,使得迁移后数据的分布可以再次达到公平.从上述迁移过程可知,数据只在增加或者删除的设备与未变化的设备之间迁移,而未变化的设备之间没有引入额外的数据迁移.

4 算法分析

4.1 公平性

引理1. 类间算法BCDP是公平的.

证明:设函数$f_c:X_0\to C$表示在类间机制中将数据集合X_0映射到类的集合$C=\{D_1,\ldots,D_g\}$,则类间算法BCDP可以表示为函数$BCDP(x,f_c)$,其中,$BCDP(x,f_c)=f_c(x)$.分配给类$D_j\in C\{D_1,\ldots,D_g\}$的区间为$I_j$.设区间$I_j$的长度为$|I_j|$,类$D_j$的相对权重为$w_j$.BCDP按照类的相对权重为类分配区间,因而$|I_j|=w_j$.由概率论可知,$\forall x\in X_0$,$x$落入区间$I_j$的概率为$|I_j|$,即$x$分配给类$D_j$的概率为$|I_j|$,因而$P(BCDP(x,f_c)=f_c(x)=D_j)=|I_j|=w_j$,即$|P(BCDP(x,f_c)=D_j)-w_j|=0$.故$\forall\varepsilon>0,|P(BCDP(x,f_c)=D_j)-w_j|<\varepsilon$.根据定义1,算法BCDP是公平的,$X_0$在类集合$C$中公平地分布. □

引理2. 类内算法ICDP是公平的.

证明:对于任意$D_j\in C$,其中$1\leqslant j\leqslant g$,设函数$f_j:X_j\to D_j$表示在类内机制中将数据集合X_j映射到设备集合D_j,则类间算法ICDP可以表示为函数$ICDP(x,f_j)$,其中,$ICDP(x,f_j)=f_j(x)$.由文献[3]可知,$P(f_j(x)=d_i^j)=\frac{1}{n_j}$,其

中,$d_i^j \in D_j$.由于 d_i^j 在D_j中的相对权重 $w_i^j = \frac{1}{n_j}$,且$ICDP(x,f_j)=f_j(x)$,因而 $P(ICDP(x, f_j) = f_j(x) = d_i^j) = \frac{1}{n_j} = w_i^j$.即 $|P(ICDP(x, f_j) = d_i^j) - w_i^j| = 0$.故$\forall \varepsilon>0, |P(ICDP(x, f_j) = d_i^j) - w_i^j| < \varepsilon$.由定义 1 可知,算法ICDP是公平的,$X_j$在设备集合$D_j$中公平地分布. □

定理 1. CCHDP 算法是公平的.

证明:设函数$f_0:X_0 \to D_0$表示在本文提出的布局机制中将数据集合X_0映射到设备集合D_0,CCHDP算法可以表示为函数$CCHDP(x,f_0)$,其中,$CCHDP(x,f_0)=f_0(x)$.CCHDP算法由BCDP算法和ICDP算法组成.因而$\forall x \in X_0$,设$CCHDP(x,f_0)=f_0(x)=d_i$,则存在$D_j \in C$,使$BCDP(x,f_c)=D_j$, $ICDP(x, f_j) = d_i^j$ 和 $d_i^j = d_i$.那么

$$P(CCHDP(x,f_0)=d_i)=P(BCDP(x,f_c)=D_j)\cdot P(ICDP(x,f_j)=d_i) \tag{1}$$

根据引理 1,$\forall \varepsilon_1>0$,有

$$|P(BCDP(x,f_c)=D_j)-w_j|<\varepsilon_1 \tag{2}$$

由引理 2 可知,$\forall \varepsilon_2>0$,有

$$|P(ICDP(x,f_j)=d_i)-w_i|<\varepsilon_2 \tag{3}$$

其中,w_j表示D_j相对于D_0的权重,w_i表示d_i在D_j中占的比重.则d_i在D_0中的权重为$w_j \cdot w_i$,那么

$$\begin{aligned}
|P(CCHDP(x, f_0) = d_i) - w_j \cdot w_i| &= |P(BCDP(x, f_c) = D_j) \cdot P(ICDP(x, f_j) = d_i) - w_j \cdot w_i| && \text{由公式(1)}\\
&= |P(ICDP(x, f_j) = d_i) \cdot (P(BCDP(x, f_c) = D_j) - w_j) + \\
&\quad w_j \cdot (P(ICDP(x, f_j) = d_i) - w_i)| \\
&\leqslant |P(ICDP(x, f_j) = d_i) \cdot (P(BCDP(x, f_c) = D_j) - w_j)| + \\
&\quad |w_j \cdot (P(ICDP(x, f_j) = d_i) - w_i)| \\
&< P(ICDP(x, f_j) = d_i) \cdot \varepsilon_1 + w_j \cdot \varepsilon_2 && \text{由公式(2)和公式(3)}
\end{aligned}$$

因$P(ICDP(x,f_j)=d_i) \in [0,1]$且$w_j \in [0,1]$,则根据$\varepsilon_1>0$ 和$\varepsilon_2>0$ 的任意性以及定义 1 可知CCHDP算法是公平的. □

4.2 自适应性

在聚类算法中,设备集合按其权重聚集到各个类中.由于类中的设备不是按照物理位置来分配的,因而将整个类中的设备同时删除的情况不具有一般性,这里不作考虑.在扩展存储系统时会批量增加设备,这些设备的容量和性能是相近的,因而可以设置一个新类容纳大量的新设备.对于添加单个设备的情况,首先判断该设备所属的类,在类的内部进行数据的迁移.在删除单个设备时,同样在内部进行调整.虽然仅仅在类的内部迁移数据会损失整体的公平性,但是由于类中的设备数很多,增加的设备几乎不会改变该类在类集合中的权重.因而在某类中增加或者删除设备后,各个类的权重几乎没有变化,数据仍然是公平地分布在各个类上.下面分两种情况讨论CCHDP 算法的自适应性:

(1) 当批量增加容量和性能相近的设备时,设类的集合$C=\{D_1,\ldots,D_g\}$变为$C_1=\{D_1,\ldots,D_g,D_{g+1}\}$,当前设备集合由$D_0$变为 D_0'.对于每个类$D_j \in C$,根据D_{g+1}的权重重新计算类D_j的权重.类D_j的权重会变小,按照权重的差值将类D_j的区间分割,将多出的区间分给D_{g+1},落在多余区间的数据迁移给D_{g+1}.数据迁移后,设函数 $f_0': X_0 \to D_0'$ 表示数据集合X_0到新设备集合 D_0' 的映射.从迁移过程可以看出,D_j只是迁出数据到D_{g+1},没有数据迁入D_j,则D_j内部的设备 d_i^j 上没有数据迁入.在迁移后某数据放在D_j的设备 d_i^j 上,那么该数据在迁移前也放在D_j的 d_i^j 上.即若 $CCHDP(x, f_0') = d_i^j$ 且 $d_i^j \in D_j$,则 $CCHDP(x, f_0) = d_i^j$.由于 $d_i^j \in D_j$ 且 $D_j \subset D_0$,因而 $d_i^j \in D_0$.于是,若 $CCHDP(x, f_0') = d_i^j$ 且 $d_i^j \in D_0$,则 $CCHDP(x, f_0) = d_i^j = CCHDP(x, f_0')$.批量删除设备的情况不具有一般性,我们不予考虑.由定义 2 可知,算法 CCHDP 是自适应的.

(2) 同理可以证明,增加和删除单个设备时,CCHDP 算法是自适应的.限于篇幅,该证明过程略.

4.3 性能分析

算法CCHDP可以将聚类过程进行预处理,一次聚类的结果可以在数据布局的时候重复使用,因而在分析CCHDP的时间复杂度时不考虑聚类的时间.在聚类算法中,根据类内距离阈值将设备集合分为若干类.将[0,1]区间按照类的权重分配给各个类,数据x在类间进行布局时,首先遍历所有的区间,根据$h(x)$的值判断x落在哪个区间内.设区间总数为I,初始分类时,每个类有一个区间,$I=g$.随着类的添加,I值不断扩大.采用树数据结构保存区间,遍历I个值可以在$O(\log I)$时间内完成.类第 1 次增加时,$I_1=g+g=2g$.类第 2 次增加时,$I_2\leqslant 2g+(g+1)$.类第s次增加时,$I_s\leqslant I_{s-1}+(g+s-1)$.采用聚类算法后,类数与设备数相比大大减少,而且PB规模的系统运转后,批量购入上百台设备的次数有限,因而I值不会变得很大.在类的内部使用一致hash的定位机制,由文献[3]可知,我们可以在$O(1)$内计算数据的位置.因而,使用CCHDP算法定位一个数据需要$O(\log I)$时间,其中,I表示当前系统的区间数.

接下来,我们讨论 CCHDP 算法的存储需求.根据聚类算法可知,类中设备的权重差异在预设的范围内.在类内进行布局时,忽略设备权重差异,每个设备的虚拟设备数目是相同的.与一致 hash 的简单改进方法相比,其使用的虚拟设备数极大地减少.在 CCHDP 算法中,为所有类分配的区间总数是 I,每个区间需要 $\log I$ 位来表示.对于任意类 $D_j\{d_i^j;i=1,2,...,n_j\}\in C\{D_1,...,D_g\}$,设设备 d_i^j 的权重为 w_j^i,权重最小的设备为 $w_j^{\min}$,则类D_j共需要 $\sum_{i=1}^{n_j}\frac{w_j^i}{w_j^{\min}}k\log|N|$ 个虚拟设备.由于类中设备的权重相差不大,$\sum_{i=1}^{n_j}\frac{w_j^i}{w_j^{\min}}k\log|N|$ 近似等于$n_jk\log|N|$.类中设备数最多为 $n_{\max}=\max_j n_j$,其中,$1\leqslant j\leqslant g$.设备数最多的类需要$\log(n_{\max}k\log|N|)$位来表示一个虚拟设备.整个设备集合中,表示一个虚拟设备需要$(\log I+\log(n_{\max}k\log|N|))$位.由第 1 节的讨论可知,在一致hash的简单改进中,区分开所有的虚拟设备需要 $\log\left(\sum_{i=1}^{n}\frac{w_i}{w_{\min}}k\log|N|\right)$ 位的存储开销.当存储系统中设备的性能和容量差异较大时,则对于 $1\leqslant i\leqslant n$来说,大部分 $\frac{w_i}{w_{\min}}$ 值很大.由于大规模存储系统中的设备数n成百上千或者成千上万,并且从上述讨论得知I值不会变得很大,因而$(\log I+\log(n_{\max}k\log|N|))$远远小于 $\log\left(\sum_{i=1}^{n}\frac{w_i}{w_{\min}}k\log|N|\right)$.由此可知,CCHDP算法减少了大量的存储空间.

5 实验与结果分析

为了测试 CCHDP 算法的公平性和自适应性,本文在模拟环境下实现了 CCHDP.测试实验取 200 个设备,40 个设备的权重为 1 000,30 个设备的权重为 2 000,30 个设备的权重为 4 000,40 个设备的权重为 6 000,30 个设备的权重为 8 000,30 个设备的权重为 9 000.设备的权重以存储的数据量为单位,如设备的权重为 5 000,则可以存储 5 000 个数据.设类内设备权重的阈值为 1,则聚类算法将设备分为 6 类,每个类的权重为类内设备的权重之和.

向这些设备分别发送 10 万、20 万、30 万、40 万以及 60 万个数据,测试数据在类间以及类内的分布情况.图 3 为类间的数据分布图,横轴表示发送不同的数据量,纵轴表示每个类的占用率.类的占用率等于分配给类的数据量与其权重之比.从图 3 可以看出,发送 10 万个数据时,每个类的占用率均为 10%左右,数据在各个类之间的分布是公平的.分别发送 20 万、30 万、40 万以及 60 万个数据时,每个类的占用率几乎是相同的.由于数据量的限制,每个类的占用率不是完全相同,只是非常地接近,图 3 显示这个误差非常的小,从统计意义上来讲每个类的占用率是相等的.接下来测试类内的数据分布的公平性,测试实验以第 2 个类为例.发送不同的数据量时,第 2 个类得到的数据量分别为 6 144,12 217,18 614,24 665 以及 37 081,这些数据按照类内布局机制分配给类 2 中的每个设备.图 4 为类 2 的每个设备的占用率,设备的占用率等于分配给设备的数据量与其权重之比.从图 4 可以看出,发送 10 万个数据时,类 2 中每个设备的占用率在 10%左右,偏离 10%的幅度比类间要大,因为数据量级变小.发送其他不同的数据量时,类 2 中每个设备的占用率也在平均值的上下浮动.从统计意义上来讲,每个设备的

占用率是相等的.由图 3 以及图 4 可知,数据在设备上的分布是公平的,因而 CCHDP 算法是公平的.

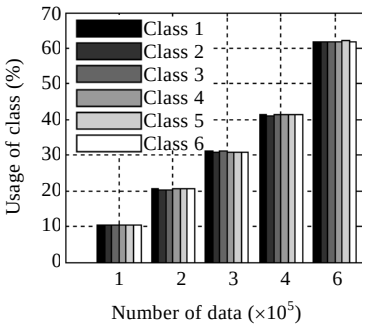


Fig.3 Distribution of data between classes

图 3 类间的数据分布图

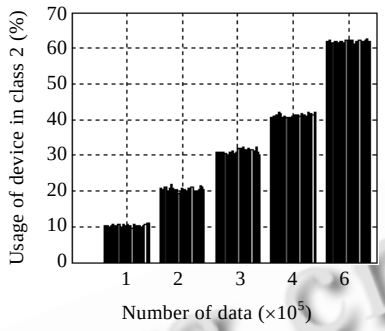


Fig.4 Distribution of data within a class

图 4 类内的数据分布图

为了测试 CCHDP 算法的自适应性,我们在改变存储设备规模的情况下计算数据迁移量.连续 4 次批量增加 30 个设备,每次批量增加的设备权重分别为 10 000,12 000,14 000 以及 16 000.按照聚类算法,每次批量增加的设备分在一个类中.测试实验以发送 40 万个数据为前提.图 5 显示了依次增加新类后的数据迁移量与理论上的迁移量.横轴表示增加的类,纵轴表示在增加某个类时迁移的数据量.增加第 7 个类以及第 10 个类时,迁移数据量的实验值略高于理论值.增加第 8 个类时,实验值与理论值非常接近.增加第 9 个类时,实验值略低于理论值.原因在于增加新类后,为新类分配区间,将落入这些区间的数据迁移到新类上.由于实验时落入这些区间的数据量小于理论上的数据量,导致迁移的数据小于理论值,因而出现了实验值略低于理论值的情况.从图 5 可以看出,增加新类时,迁移数据量的实验值与理论值非常接近.

为了测试单个设备增加或删除时算法的自适应性,我们连续 3 次增加权重为 2 000 的设备,然后连续 3 次删除该类中的设备.根据聚类算法,增加的设备落入第 2 个类中.图 6 显示了依次增加权重为 2 000 的设备以及依次删除权重为 2 000 的设备时数据的迁移量与理论上的迁移量.随着设备的增加,新增设备的相对权重变小,因而数据迁移量依次减小.依次减少设备时,被删设备的相对权重变大,数据迁移量依次增加.当设备数增加到 33 以及设备数减少到 30 的时候,迁移数据量的实验值略低于理论值,原因同类间的情况是一样的.从图 6 可以看出,数据迁移的实验值与理论值非常地接近.由图 5 以及图 6 可知,CCHDP 算法具有良好的自适应性.

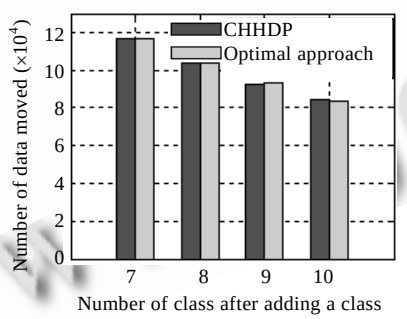


Fig.5 Movements of data between classes

图 5 类间数据的迁移图

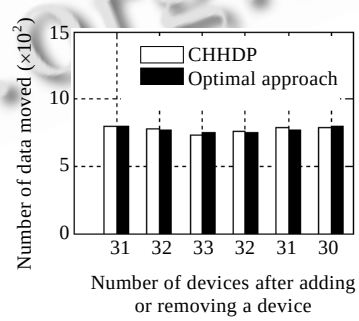


Fig.6 Movements of data within a class

图 6 类内数据的迁移图

图 7 和图 8 显示了存储设备规模改变后的数据重新分布情况.测试在上述实验的基础上进行,仍然以发送 40 万个数据量为前提.连续 4 次批量增加 300 个设备后,类间的数据重新分布情况如图 7 所示.横轴表示系统中的类数,纵轴表示每个类的占用率.随着类的增加,每个类的占用率变小.从图 7 可以看出,增加新类后,每个类的占用率几乎是相同的.依次增加权重为 2 000 的设备以及删除设备后,数据的重新分布如图 8 所示.横轴表示类中的设备数,纵轴表示每个设备的占用率.依次增加设备时,其占用率逐渐变小;连续删除设备时,其占用率逐渐

变大.从图 8 可以看出,增加单个设备或者删除单个设备后,每个设备的占用率几乎是相同的.由图 7 和图 8 可知,存储规模改变后,CCHDP 算法可以再次满足数据分布的公平性.

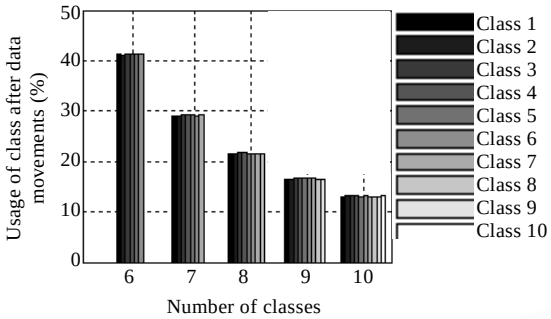


Fig.7 Redistribution of data between classes after adding a new class

图 7 增加类后数据的重新分布图

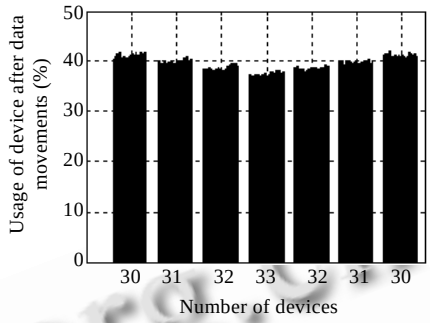


Fig.8 Redistribution of data within a class after adding or deleting a single device

图 8 增加或者删除设备后数据的重新分布图

6 结 论

近年来,大量的数据存储需求使得大规模网络存储系统发展迅猛.如何有效地对数据进行布局是系统面临的重大挑战.本文提出的 CCHDP 算法将设备集合按照权重的大小进行聚类,使得同一类别中的设备权重差别小于预设的值,避免在类的内部使用一致 hash 时引入大量的虚拟设备,同时按照类别分配区间,使用的区间量少,节省了大量的存储空间.CCHDP 可以在 $O(\log I)$内快速地定位数据,其中,I 表示与类相关的区间数.CCHDP 算法具有良好的公平性以及自适应性,可以快速地定位数据,消耗少量的存储空间,是大规模网络存储系统的一个好的选择.

References:

[1] Gray J. What next? A few remaining problems in information technology. 1998. http://research.microsoft.com/~gray/talks/Gray_Turing_FCRC.pdf

[2] Welch B, Unangst M, Abbasi Z, Gibson G. Scalable performance of the Panasas parallel file system. In: Baker M, ed. Proc. of the 2008 Conf. on File and Storage Technologies (FAST 2008). San Jose: USENIX, 2008. 17−33.

[3] Karger D, Lehman E, Leighton T, Levine M, Lewin D. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In: Proc. of the 29th Annual ACM Symp. on Theory of Computing (STOC'97). El Paso: ACM Press, 1997. 654−663.

[4] Tang H, Gulbeden A, Zhou JY, Strathearn W, Yang T, Chu LK. A self-organizing storage cluster for parallel data-intensive applications. In: Huskamp JC, ed. Proc. of the 2004 ACM/IEEE Conf. on Supercomputing (SC 2004). Pittsburgh: IEEE Computer Society, 2004. 52−63.

[5] Stoica I, Morris R, Karger D, Kaashoek F, Balakrishnan H. Chord: A scalable peer-to-peer lookup service for internet applications. In: Cruz R, ed. Proc. of the Annual Conf. of the Special Interest Group on Data Communication (SIGCOMM 2001). San Diego: ACM Press, 2001. 149−160.

[6] Brinkmann A, Salzwedel K, Scheideler C. Efficient, distributed data placement strategies for storage area networks. In: Gary M, ed. Proc. of the 12th ACM Symp. on Parallel Algorithms and Architectures. Bar Harbor: ACM Press, 2000. 119−128.

[7] Brinkmann A, Salzwedel K, Scheideler C. Compact, adaptive placement schemes for non-uniform distribution requirements. In: Maggs B, ed. Proc. of the 14th ACM Symp. on Parallel Algorithms and Architectures. Winnipeg: ACM Press, 2002. 53−62.

[8] Schindelhauer C, Schomaker G. Weighted distributed hash tables. In: Paul S, ed. Proc. of the 17th ACM Symp. on Parallelism in Algorithms and Architectures (SPAA 2005). Las Vegas: ACM Press, 2005. 218−227.

[9] Honicky RJ, Miller EL. A fast algorithm for online placement and reorganization of replicated data. In: Dongarra J, ed. Proc. of the 17th Int'l Parallel & Distributed Processing Symp. (IPDPS 2003). Nice: IEEE Computer Society, 2003.

[10] Honicky RJ, Miller EL. Replication under scalable hashing: A family of algorithms for scalable decentralized data distribution. In: Bader DA, ed. Proc. of the 18th Int'l Parallel & Distributed Processing Symp. (IPDPS 2004). Santa Fe: IEEE Computer Society, 2004.

[11] Weil SA, Brandt SA, Miller EL, Maltzahn C. CRUSH: Controlled, scalable and decentralized placement of replicated data. In: Miller BH, ed. Proc. of the 2006 ACM/IEEE Conf. on Supercomputing. Tampa: IEEE Computer Society, 2006.

[12] Weil SA, Brandt SA, Miller EL, Long DDE, Maltzahn C. Ceph: A scalable, high-performance distributed file system. In: Bershad B, ed. Proc. of the 7th Symp. on Operating Systems Design and Implementation. Seattle: USENIX, 2006. 307−320.

[13] Gobioff H, Gibson G, Tygar D. Security for network attached storage devices. Technical Report, TRCMU-CS-97-185, Pittsburgh: Carniege Mellon University, 1997.

[14] Sun Microsystems, Inc. Lustre file system: High-Performance storage architecture and scalable cluster file system. 2007. https://www.sun.com/offers/docs/LustreFileSystem.pdf

[15] Nagle D, Serenyi D, Matthews A. The Panasas ActiveScale storage cluster—Delivering scalable high bandwidth storage. In: Huskamp JC, ed. Proc. of the 2004 ACM/IEEE Conf. on Supercomputing. Pittsburgh: IEEE Computer Society, 2004.

[16] Schmuck F, Haskin R. GPFS: A shareddisk file system for large computing clusters. In: Long D, ed. Proc. of the 2002 Conf. on File and Storage Technologies (FAST 2002). Monterey: USENIX, 2002. 231−244.

[17] Liu Z, Zhou XM. A data object placement algorithm based on dynamic interval mapping. Journal of Software, 2005,16(11): 1886−1893 (in Chinese with English abstract). http://www.jos.org.cn/1000-9825/16/1886.htm [doi: 10.1360/jos161886]

[18] Wang D, Shu JW, Xue W, Shen MM. Self-Adaptive hierarchical storage management in SAN based on block-level. Advanced Technology Communication, 2007,17(2):111−115 (in Chinese with English abstract).

[19] Wang F, Zhang SD, Feng D, Zeng LF. Hybrid object allocation policy for object storage systems. Journal of Huazhong University of Science & Technology (Nature Science Edition), 2007,35(3):46−48 (in Chinese with English abstract).

[20] Tan HF, Sun LL, Hou ZF. An effective algorithm of data placement in a mass storage system. Computer Engineering, 2006,32(10): 47−49 (in Chinese with English abstract).

[21] Brinkmann A, Effert S, Heide FMAD, Scheideler C. Dynamic and redundant data placement. In: Abdelrahman TS, ed. Proc. of the 27th Int'l Conf. on Distributed Computing Systems. Toronto: IEEE Computer Society, 2007.

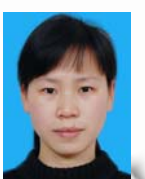
附中文参考文献:

[17] 刘仲,周兴铭.基于动态区间映射的数据对象布局算法.软件学报,2005,16(11):1886−1893. http://www.jos.org.cn/1000-9825/16/1886.htm [doi: 10.1360/jos161886]

[18] 王迪,舒继武,薛巍,沈美明.基于块级别的 SAN 系统自适应分级存储.高技术通讯,2007,17(2):111−115.

[19] 王芳,张顺达,冯丹,曾令仿.对象存储系统中的柔性对象分布策略.华中科技大学学报(自然科学版),2007,35(3):46−48.

[20] 谈华芳,孙丽丽,侯紫峰.大规模存储中的一个有效的数据放置算法.计算机工程,2006,32(10):47−49.



陈涛(1982－),女,湖北荆州人,博士生,主要研究领域为网络存储,网格监控.

刘芳(1976－),女,博士,讲师,主要研究领域为信息安全,网络存储.

肖侬(1969－),男,博士,教授,博士生导师,主要研究领域为分布计算,网格计算,网络存储.

付长胜(1982－),男,博士生,主要研究领域为网格计算,网络存储.