存储系统负载平衡机制的评价与分析

罗香玉 汪 芸 陈笑梅

(东南大学计算机学院计算机网络和信息集成教育部重点实验室 南京 211189)

摘 要 负载平衡是提高大规模存储系统资源和能源使用效率,进而降低系统建设成本和运行成本的重要手段。然而,该领域相关工作多侧重于具体技术方法的研制和改进,缺乏不同方法间的比较和评价。通过对存储系统负载平衡方法的仿真与分析,揭示了现有负载平衡方法所面临的主要挑战,也为工程上各方法的选择提供了参考依据。

关键词 存储系统,负载平衡,数据迁移,数据放置

中图法分类号 TP393 **文献标识码** A

Evaluation and Analysis of Load Balancing Mechanisms in Storage Systems

LUO Xiang-yu WANG Yun CHEN Xiao-mei

(School of Computer Science and Engineering, Key Lab of CNII MoE, Southeast University, Nanjing 211189, China)

Abstract For a large-scale storage system, load balancing is an important way to improve the utilization ratio of system resources and the energy efficiency. It could greatly reduce the provisioning and operational cost. However, most existing studies are focused on the design or improvement of a specific method while mading no evaluation or comparison among different methods. In this paper, extensive simulations were conducted to evaluate the existing load balancing methods. The results reveal the main challenge and unsettled problem in the area. Besides, theoretical basis was founded for selecting the appropriate load balancing method in practice.

Keywords Storage system, Load balance, Data migration, Data placement

1 引言

随着存储系统规模的增长,其建设成本和运行成本日益攀升。单就能耗成本而言,存储设备所产生的能耗已占据数据中心总能耗的37%~40%[1],并且还在持续增加。有效降低存储系统成本的途径之一是提高资源和能源的使用效率。

负载平衡可提高大规模存储系统对所持有资源的使用效率;在相同性能需求下,可有效减少设备配置需求量,进而降低系统总能耗。所面临的主要挑战是分布极不均衡和动态变化的文件访问热度。例如,Web文件访问热度呈现出类似Zipf的分布[2],即文件访问频率和$1/i^{\alpha}$成正比(i是文件的访问热度排名,α是大于0的常数,称为文件访问偏斜度。α越大,文件间访问热度的差异性越显著)。另外,各文件的访问热度随时间推移可能发生变化[3]。因此,系统各节点间即使在某段时间达到了负载平衡,后续仍然可能因为文件访问热度的变化而不再平衡。

存储系统负载平衡问题受到广泛关注,各种负载平衡方法不断被提出。其中包括:对数据存储位置进行优化的算法,如LSB_Placement[4]、SOR[5];基于多副本的动态请求分发算法,如Kinesis[6]、D-SPTF[7];动态改变数据存储位置的数据迁移算法[8,9]。但是,现有工作大多关注具体技术方法的研制和改进,对一些更为基本的理论问题缺乏深入研究。如:1)如何对负载平衡效果进行评价?2)节点规模增长如何影响负载平衡效果?3)动态改变数据存储位置的迁移技术对解决负载平衡问题是否必需?4)如何针对特定应用选择适用的负载平衡方法?这些基本问题同负载平衡技术的研究和工程应用存在紧密联系。它们的有效解决依赖对存储系统负载平衡机制的评价与分析。

围绕上述问题,本文提出平衡指数的概念,用以表征系统各节点间的负载均衡程度。和常见指标(如吞吐量)不同,平衡指数独立于应用负载的具体数值及单个存储节点的硬件参数,更能反映负载平衡机制本身的性质,且可通过仿真实验求取。本文通过大量仿真实验,对存储系统的负载平衡技术进行研究,分析了各因素(如节点数量n,文件访问偏斜度α)对平衡指数的影响。研究结果表明,多副本动态请求分发技术的负载平衡效果随系统规模增长急速下降;单位数据对访问负载量的平均贡献值A和单位存储空间对I/O带宽资源的平均占有量E越接近,系统的负载平衡难度越高,对相关技术的要求越严格。从而得出结论:1)设计高可扩展的适用于大规模存储系统的数据迁移技术具有现实必要性,这恰是当前负载平衡研究中尚未解决的重要问题;2)工程应用中可根据A和E的关系选择其适用的负载平衡技术。

本文第2节介绍主流的负载平衡方法及其技术特点;第3节针对balance-blind存储系统进行仿真,分析规模增长给

到稿日期:2012-11-11 返修日期:2013-03-27 本文受国家863高技术计划(2011AA040502),国家自然科学基金(60973122)资助。

罗香玉(1984—),女,博士生,主要研究方向为分布式存储系统,E-mail:luoxiangyu@seu.edu.cn;汪 芸(1967—),女,博士,教授,主要研究方向为分布式计算、容错理论;陈笑梅(1989—),女,硕士生,主要研究方向为分布式存储系统。

负载平衡问题带来的挑战；第 4 节针对采用多副本动态请求分发机制的存储系统进行仿真，分析该类技术的局限性；第 5 节讨论工程应用中负载平衡技术的选择问题；最后总结全文。

2 现有负载平衡技术介绍

现有的存储系统负载平衡技术大致包括如下 4 类：数据分片、数据放置优化、多副本下动态请求分发以及数据迁移。

2.1 数据分片

数据分片是最简单的负载平衡技术。通过将文件分割到多个磁盘进行存储，可有效屏蔽文件间的访问热度差异，保证各磁盘的负载平衡。

数据分片对中小规模磁盘阵列系统非常适用。但是，对大规模磁盘阵列系统或网络存储系统而言，若一个文件被分散到各个磁盘或节点上存储，则访问该文件所付出的系统代价将增加。比如，进行更多次磁盘寻道，建立更多网络连接等。因此，在大规模存储系统中，单个文件的分片数量远少于节点总数。数据分片仅作为其它技术的补充手段。如 Google 的分布式文件系统 GFS[10] 在采用数据分片提高并行访问速度的同时，配合数据迁移实现负载平衡。

2.2 数据放置优化

如上所述，数据分片不适于作为大规模存储系统负载平衡问题的完整解决方案。一些研究者关注数据放置算法，其通过对数据存储位置的优化实现负载平衡。

数据放置问题即经典的 FAP(File Assignment Problem)：存储系统资源集合用 D 表示，$D=\{d_1,\cdots,d_j,\cdots,d_n\}$；待存储的文件用集合 F 表示，$F=\{f_1,\cdots,f_i,\cdots,f_m\}$。$d_j$ 表示一个磁盘或存储节点，用一组参数表征：$d_j=(c_j,t_j,l_j)$，其中 c_j 表示存储容量；t_j 表示数据读取速度；l_j 表示 d_j 所存储文件的访问占空比之和(访问占空比的定义见下文)。f_i 表示单个文件，也用一组参数表征：$f_i=(s_i,\lambda_i,es_i,h_i)$，其中 s_i 表示文件大小；λ_i 是文件上的请求到达率；es_i 表示文件上单个请求的执行时间，假设该文件被分配到 d_j 上，则 $es_i=s_i/t_j$；定义 $h_i=\lambda_i\cdot es_i$，h_i 称为文件的访问占空比，用来表征该文件上的访问请求对磁盘 I/O 所造成的压力。d_j 存储文件的访问占空比之和 l_j 表示 d_j 处于工作状态的比例。用 ρ 表示单个磁盘或存储节点的平均访问占空比，则 $\rho=\frac{1}{n}\sum_{j=1}^{n}l_j=\frac{1}{n}\sum_{i=1}^{m}h_i$；以 S_j 表示分配到 d_j 上的所有文件所需占用的存储空间。FAP 就是求解 F 到 D 的映射，满足约束条件 $S_j\leqslant c_j$，且使 l_j 尽可能接近 ρ，即各节点忙于数据访问的时间大致相当[5]。FAP 已被证明是 NP 难问题[11]，很多近似算法被提出，如 LSB_Placemen[4] 和 SOR[5]。

LSB_Placement 巧妙地综合运用了 MMPacking[12] 和 Bin packing 两个算法。MMPacking 在文件大小一致的假设下，可实现各磁盘所分配文件的访问负载量之和近似相等，且文件数量大致相当。Bin packing 在文件大小不一致的条件下，可实现各磁盘空间占用量近似相等。LSB_Placement 主要思想如下：首先，使用 Bin packing 算法将大小不一的文件分成尽可能多的文件组，且各文件组所含文件大小之和近似相等，每一个文件组可抽象成一个“大文件”。然后，使用 MMPacking完成“大文件”的放置，实现各磁盘上“大文件”数量近似相等，且其上的访问负载量也大致均衡。

SOR 主要思想是大小相近的文件集中放置且避免热点文件集中放置。首先计算出节点的平均访问负载量；然后找出一些访问频率最高的文件，将它们分配到不同的节点上以分散访问热点；最后将剩余文件按照大小作升序排列，顺序将其分配到各节点上，仅当一个节点超出平均访问负载量时才开始将文件分往下一个节点。SOR 尽可能减少同一节点上文件间的大小差异，以减少系统的平均响应时间。

数据放置优化要求应用提供每个文件的访问频率信息，无法适应文件访问频率未知或动态变化的情况。于是，另外两类负载平衡方法被提出。一类效仿 Web 服务器的负载平衡思想，为数据存储多个副本，在副本间根据负载状况实时进行请求分发，以平衡不同节点上的负载；另一类随着文件访问热度变化，动态调整数据存储位置，即进行数据迁移。

2.3 多副本动态请求分发

该类方法为数据创建多个副本。访问请求到来时，将其调度到存有相应数据副本且当前负载比较轻的节点执行，以平衡节点间的负载。代表性工作有 D-SPTF[7] 和 Kinesis[6]。

D-SPTF 将文件读请求发送给所有副本节点。若某副本节点 cache 命中，则通知其它副本节点将相应请求从队列中删除；节点从队列中取出一个请求开始执行时，也通知其它节点将相应请求删除。D-SPTF 保证请求总是由响应最快的副本调度执行，且均衡了节点间的负载。

D-SPTF 没有指定文件副本的放置方法，Kinesis 则给出了一种副本放置方法。和 Chain[13] 等方式相比，Kinesis 使副本的放置更加“随机化”，即任意两个节点所存储的文件副本都可能存在交集，如此提升了请求动态分发机制调整节点间负载平衡的能力。Kinesis 将节点分成 k 个规模大致相当的组，每个组对应一个哈希函数，这 k 个哈希函数相互独立。数据放置时，Kinesis 根据文件标识的哈希值确定 k 个节点，从中选取 r 个最空闲的节点存放数据副本($r<k$)。数据访问时，Kinesis 同样根据文件标识的哈希值确定 k 个节点，将访问请求发送给它们。k 个节点分别返回是否存有相应数据以及自身当前的负载值，Kinesis 选择其中最轻载且存有数据副本的节点进行数据访问。

多副本动态请求分发技术中数据的存储位置静态不变，仅动态指定为请求服务的副本，故称其为半动态技术。

2.4 数据迁移

数据迁移技术同样借助数据存储位置的优化平衡节点间负载。然而，数据放置优化仅需确定初始时文件和节点间的映射关系，数据迁移则复杂很多，至少需解决 3 方面问题：1)何时触发迁移；2)如何确定迁移目标状态；3)如何规划迁移到目标状态的步骤。代表性工作有文献[8,9]，前者研究目标状态的选择，即如何在迁移收益和代价间进行权衡，后者研究迁移步骤的规划方法。此外，Q. Wei 等提出基于数据迁移的动态副本管理算法 CDRM[14]，C. Xie 等提出基于随机算法的分布式迁移机制 RMSA[15]，W. Wang 等提出基于 FARP 的可支持副本位置在线灵活调整的算法 NDSC[16]。

数据迁移会增加系统复杂性，需解决数据迁移后文件的重定位问题、迁移过程中用户访问请求和迁移本身的资源占用冲突问题等。数据存储位置及访问请求服务的节点均动态变化，因此，称数据迁移为全动态技术。

2.5 各类负载平衡技术的比较

各类负载平衡技术的综合比较见表 1。该表从适用存储

环境、对应用负载的假设以及技术复杂度3个方面，对前述4类技术进行比较。

表1 各类负载平衡技术的比较

技术分类	适用存储环境	应用负载假设	技术特征
数据分片	中小规模 RAID	未知、变化	全静态
数据放置优化	RAID 和网络存储	已知、固定	全静态
多副本请求分发	网络存储	未知、变化	半动态
数据迁移	网络存储	未知、变化	全动态

从表中可以看出：数据分片的缺陷在于无法适用于大规模存储系统；数据放置优化要求各文件访问热度可知且不随时间变化，应用范围比较受限；多副本请求分发和数据迁移对存储环境和应用负载限制较少，且前者较后者机制简单，便于工程实现。此外，数据迁移技术可导致文件副本分布的“无序化”，扼杀了系统的成比例能耗能力[17]。随着能耗成本问题日益突出，要求各类技术在实现负载平衡的同时保持对成比例能耗的支持能力。现有保证系统成比例能耗的技术大多采用多副本请求分发机制[18,19]，而避免采用数据迁移技术。然而，多副本请求分发的负载平衡能力缺乏相关评价。

尽管现有研究中存在大量同时考虑读性能及写性能的算法，如 MyCassandra[20]，但本文主要关注 Worm(Write-once-read-many，即一次写多次读)类型的负载，对技术的相关分析和实验研究均以该负载类型为前提。

3 原始平衡指数分析

3.1 平衡指数

为比较和分析各种负载平衡技术，需要建立负载平衡能力的评价指标。本文提出平衡指数的概念，用以表征各种技术的负载平衡能力。平衡指数是指系统所能承受的最大负载量 L_{max} 与系统总 I/O 能力 T 的比值，用 θ 表示，θ 介于 0 和 1 之间。系统所承受的负载量 L 用单位时间所有文件上的访问请求所访问的数据量之和表示，即 $L=\sum_{i=1}^{m}\lambda_i s_i$，$\lambda_i$ 表示文件 f_i 的访问频率，s_i 表示文件 f_i 的大小；而系统所能承受的最大负载量 L_{max} 是指负载最重的节点恰好满负荷时系统所承受的负载量 L。系统总 I/O 能力 T 用所有节点的数据读取速度之和表示，即 $T=\sum_{j=1}^{n}t_j$，t_j 表示节点 d_j 所能提供的数据访问带宽。可见，节点间负载越均衡，L_{max} 越接近于系统总 I/O 能力 T，θ 越接近于 1。

根据平衡指数的定义，如果节点同构，则有：

$$\theta=L_{max}/T=\sum_{i=1}^{m}\lambda_i(\max)s_i/\sum_{j=1}^{n}t_j=(1/n)\sum_{i=1}^{m}\lambda_i(\max)es_i$$

$$\theta=(1/n)\sum_{i=1}^{m}h_i(\max)=\rho(L_{max}) \qquad (1)$$

即 θ 等于最大可承受负载下节点的平均访问占空比。其中，$\lambda_i(\max)$ 是系统在负载量 L_{max} 下文件 f_i 的访问频率。

t_j 越大，系统可承受的最大负载量也成比例增加，即 $\lambda_i(\max)$ 成比例增加。所以，θ 独立于 t_j 和 $\lambda_i(\max)$ 的具体数值，而主要决定于两方面：一是存储系统本身的负载平衡机制；二是应用负载在文件间分布的均匀程度，其可由文件访问偏斜度 α 表征。不同的平衡机制，其负载平衡能力的差异会导致 θ 不同；相同机制下，文件访问偏斜度的不同也会影响 I/O 资源的利用情况，导致 θ 不同。同系统吞吐量或请求平均响应时间相比，平衡指数独立于节点的 I/O 能力以及应用负载的具体数值，更能反映负载平衡机制本身的性质。

3.2 原始平衡指数

原始平衡指数是指系统采用 balance-blind 策略对数据进行完全随机放置时所能获得的平衡指数，用 θ_0 表示。为便于分析，本文暂不考虑文件大小差异以及节点异构性，重点分析文件访问不均程度和系统规模对平衡指数的影响。在单个节点平均存储的文件数目 u($u=m/n$，m 是文件总数)给定的条件下，原始平衡指数仅和两个因素有关：节点数量 n 和文件访问偏斜度 α。

原始平衡指数可以反映系统对负载平衡技术的需求程度。若该指数较高，则负载平衡技术可发挥作用的空间较有限，系统对其需求程度较弱；反之，则对技术的负载平衡能力有较强要求。另外，分析原始平衡指数的影响因素，可获知系统负载不均程度随各参量(如节点规模 n)变化的规律。

3.3 仿真实验及结果分析

由同构系统下 θ 的计算式(1)可知，求解 θ 即求解系统最大可承受负载下各节点的 I/O 占空比均值。所谓最大可承受负载，是指系统最重载节点的 I/O 占空比为 1 时的系统负载。最大可承受负载下各节点 I/O 占空比的均值的求解仅需计算最重载节点所承受的负载占系统总负载的比例 ω 即可，θ 可由式(2)求得。

$$\theta=1/(n*\omega) \qquad (2)$$

上述问题可抽象描述如下：假定有 n 个盒子和 m 个球($m=n*u$)，球的重量依次为 $(1/i^{\alpha})/(\sum_{j=1}^{m}1/j^{\alpha})$，$i=1,\cdots,m$；将球依次放入盒子中，每次均以等概率从 n 个盒子中随机选择目标容器。求各盒子所装球的重量和的最大值 ω。

针对上述抽象问题，直接进行数学推导求解十分困难。本文采用蒙特卡罗仿真方法求解，并设计如下实验来分析参数 α 和 n 对原始平衡指数的影响规律。

仿真实验1 访问偏斜度 α 对原始平衡指数的影响

假定 u 取 1000，分别在 $n=200$，400 和 800 这 3 种情况下观察访问偏斜度 α 对原始平衡指数 θ_0 的影响。实验结果如图 1 所示。可见，负载本身在文件间的分布越不均匀，即 α 越大，则节点间的负载不均程度就越严重，即 θ_0 越小。

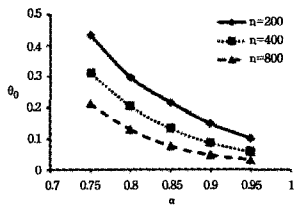

图1 θ_0 和访问偏斜度 α 之间的关系

仿真实验2 节点数 n 对原始平衡指数的影响

假定 u 取 1000，分别在 $\alpha=0.8$，0.85 和 0.90 这 3 种情况下观察自变量 n 对原始指数 θ_0 的影响。仿真结果如图 2 所示。可见，系统的平衡指数随 n 的增大而急速下降。

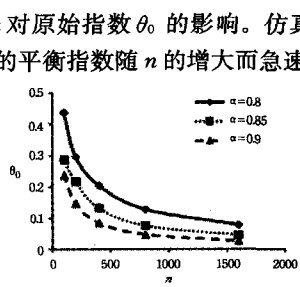

图2 θ_0 和节点数量 n 之间的关系

3.4 相关结论

即使负载在文件间的分布偏斜程度不变，节点间的负载不均程度也会随着系统规模的增加而增加。研究具有高扩展性、适用于大规模存储系统的负载平衡技术具有现实必要性。

4 DRDAR 平衡指数分析

4.1 影响因素

DRDAR 平衡指数是指系统采用多副本动态请求分发机制时所获得的平衡指数(DRDAR 是 Dynamic Request Distribution Among Replicas 的缩写)。

在单节点平均存储文件数目 u 确定的条件下，影响 DRDAR 平衡指数的因素除节点数量 n 和文件访问偏斜度 α 外，还包括副本数目 r。r 越大，则请求分发时可选择节点越多，调整节点间负载平衡的能力也越强(原始平衡指数可看成 $r=1$ 情况下的 DRDAR 平衡指数)。

4.2 仿真实验及结果分析

对典型的多副本动态请求分发机制 Kinesis 进行仿真，分析该机制下的负载平衡能力。主要分为数据放置过程仿真和数据访问过程仿真。

数据放置过程中，首先将 n 个节点划分成 k 个相等的组(为便于分析，假设 k 整除 n)，各组有其独立的哈希函数。以文件标识(仿真中用随机产生的字符串表示)作输入，计算出 k 个哈希值；根据哈希值在各组中确定一个候选节点；将 k 个候选节点根据 I/O 请求队列长度进行升序排列(若队列长度相同，则按空间占用量升序排列)；选取前 $r(r<k)$ 个节点完成文件副本放置。数据放置结束后，将所存储的文件随机编号为 1 到 m 间不重复的整数。

针对数据访问过程，构造一个离散型随机变量，变量的取值范围是 1 到 m 之间的所有整数。整数 $i(1\leqslant i\leqslant m)$ 的取值概率为 $(1/i^{\alpha})/(\sum_{j=1}^{m}1/j^{\alpha})$。每次访问请求按此概率分布选择文件编号。为请求指定副本节点的依据是访问队列长度最短优先。请求所需执行的时间由相对时间表征。将应用负载的请求平均到达时间定义为单位时间，应用负载量占系统总 I/O 能力的比率为 θ' 时，单位时间系统能够处理的请求数量为 $1/\theta'$，单个节点的处理能力为 $1/(n\theta')$，即单个请求的执行需要 $n\theta'$ 个单位时间。逐步增加 θ'，直至某个节点的访问队列发生溢出，此时的 θ' 即等于系统的平衡指数。

仿真实验 3　访问偏斜度 α 对 DRDAR 平衡指数的影响

假定 u 取 1000，分别在($n=200,r=3$)，($n=200,r=4$)和($n=400,r=4$) 3 种情况下做仿真。分析自变量 α 对 DRDAR 平衡指数的影响。仿真结果见图 3。可见，访问负载在文件间的分布越不均匀，即 α 越大，则 DRDAR 平衡指数越小。

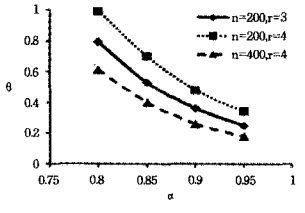

图 3　θ 和访问偏斜度 α 之间的关系

仿真实验 4　节点数 n 对 DRDAR 平衡指数的影响

假定 u 取 1000，分别在($r=3,\alpha=0.90$)，($r=3,\alpha=0.80$)和($r=4,\alpha=0.90$) 3 种情况下做仿真，分析自变量 n 对 DRDAR 平衡指数的影响。仿真结果见图 4。可见，多副本动态请求分发技术下，即使副本数目和负载在文件间的分布不均程度保持不变，随着系统规模的增长，平衡指数仍然急剧下降。

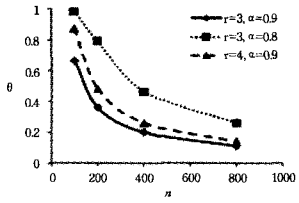

图 4　θ 和节点数量 n 之间的关系

仿真实验 5　访问偏斜度 α 对 r 的要求

假定 u 取 1000，分别在 $n=100$，200 和 400 这 3 种情况下做仿真，观察自变量 α 和满足 DRDAR 平衡指数不低于 0.80 的最小副本数 $r_{\min}$ 之间的关系。仿真结果见图 5。可见，随着访问偏斜度 α 的增加，满足给定平衡指数要求的最小副本数 $r_{\min}$ 快速增长。因此，对文件访问偏斜度较大的系统而言，仅依靠多副本下动态请求分发来实现负载平衡，代价高昂。

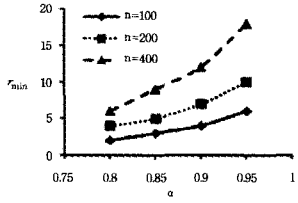

图 5　$r_{\min}$ 和访问偏斜度 α 之间的关系

仿真实验 6　节点数 n 对 r 的要求

假定 u 取 1000，分别在 $\alpha=0.80$，0.85 和 0.90 这 3 种情况下做仿真，观察自变量 n 和满足 DRDAR 平衡指数不低于 0.80 的最小副本数 $r_{\min}$ 之间的关系。仿真结果见图 6。可见，满足给定平衡指数要求的最小副本数 $r_{\min}$ 随节点规模增长而迅速增加，Kinesis 无法满足大规模系统对负载平衡的要求。

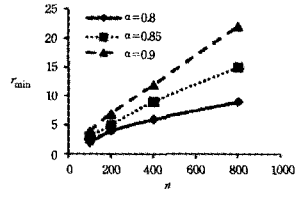

图 6　$r_{\min}$ 和节点数量 n 之间的关系

4.3 相关结论

多副本动态请求分发技术并不能解决所有存储系统的负载平衡问题，数据迁移等复杂技术的研究有其必要性。

5 目标平衡指数代价分析

原始平衡指数和 DRDAR 平衡指数，独立于单个节点的 I/O 能力以及应用负载的具体数值，是特定机制下系统的一种固有属性。本节将面向具体应用需求，研究如何利用平衡指数选择适用的负载平衡机制。

5.1 相关定义

系统所采用的负载平衡机制不同，可获得的平衡指数也可能不同。本节用 θ 泛指一个系统获得的平衡指数。例如，balance-blind 机制下对应于原始平衡指数，多副本动态请求分发机制下对应于 DRDAR 平衡指数。由 $\theta=L_{\max}/T$ 可知，

在系统 I/O 资源 T 给定条件下，若应用负载 L 可被系统承受，则 $L \leqslant \theta T$，即 $\theta \geqslant L/T$。令 $\theta_{\min}=L/T$，称 $\theta_{\min}$ 为目标平衡指数。在给定 I/O 资源量 T 和应用负载量 L 条件下，一种负载平衡机制可被接受的必要条件是该机制下系统的平衡指数不低于目标平衡指数。

对给定应用负载 L，若某种机制下目标平衡指数不可达，根据 $\theta_{\min}=L/T$，唯有提高资源供给量 T，才能降低目标平衡指数。不同机制实现的目标平衡指数所需的资源供给量 T 可能不同。满足给定负载量所需资源供给量的大小可用来评价不同机制的优劣。本文采用实际配置节点数量超出最低需求量的比例 R_{over} 来衡量一种机制实现目标平衡指数所付出的资源代价。设单节点存储能力为 s，I/O 能力为 l；应用待存储数据总量为 S，单位时间系统的 I/O 负载量为 L。节点最低需求量 $n_{\min}=\max\{S/s, L/l\}$，资源代价 $R_{over}=(n_{req}-n_{\min})/n_{\min}$，$n_{req}$ 为实现目标平衡指数所需配置的节点数量。

除资源代价外，还需考虑软件代价，如软件开发代价、运行时占用系统资源的代价以及机制复杂性对系统可靠性和可维护性的影响等。工程上优先选择简单机制，仅当 R_{over} 超出容忍范围时，才考虑复杂机制。

数据分片和数据放置优化两类机制均存在限制条件，前者限制存储系统的规模；后者要求文件访问频率固定且可知。二者均不适用于大规模的网络存储系统。因此，本文重点关注多副本动态请求分发和数据迁移两类机制之间取舍的一般性规律。

5.2 仿真实验及结果分析

根据 $n_{\min}=\max\{S/s, L/l\}$，可求出 $n_{\min}$；根据 $r_0=n_{\min}/(S/s)$，可求出 r_0。为计算方便，限定 S/s 整除 $n_{\min}$。令 $r=r_0$，根据 4.2 节计算 DRDAR 平衡指数的方法，求得副本数为 r 时的平衡指数 θ。观察其能否达到目标平衡指数 $\theta_{\min}$，$\theta_{\min}$ 计算方式如下：$\theta_{\min}=L/(r*(S/s)*l)=(L/S)/(r*(l/s))$。若 $\theta \geqslant \theta_{\min}$，则求出资源代价 $R_{over}=(r*(S/s)-n_{\min})/n_{\min}$。否则，$r$ 加 1，重复上述步骤。

由上述计算过程可知，在 α 和 u 给定条件下，r，n_{req} 仅依赖比值 L/S（记为 A，表示单位数据对访问负载量的平均贡献值）以及 l/s（记为 E，表示单位存储空间对 I/O 带宽资源的平均占有量）。将 A 和 E 的比值记为 R_{AE}，即 $R_{AE}=(L/S)/(l/s)=(L/l)/(S/s)$，则 S/s 给定时，R_{AE} 决定了资源代价的高低，从而决定了负载平衡机制的选择。本小节通过仿真实验研究 R_{AE} 和多副本动态请求分发机制资源代价 R_{over} 之间的关系。

假定 α 取值 0.8，分别在（$s/S=0.01$，$m=100000$），（$s/S=0.005$，$m=200000$）和（$s/S=0.0025$，$m=400000$）3 种情况下做仿真。观察自变量 R_{AE} 和资源代价 R_{over} 之间的关系。仿真结果见图 7。可见，R_{AE} 越接近 1（对数值接近 0），资源代价越高。

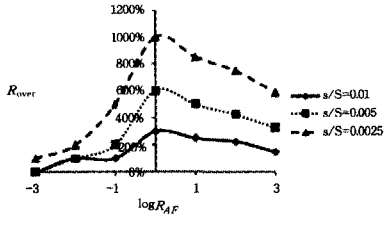

图 7 R_{over} 和 R_{AE} 之间的关系

5.3 相关结论

$R_{AE}>1$ 时，$n_{\min}=L/l$；且 R_{AE} 越大，按最低需求量配置下的存储空间冗余度越高，多副本动态请求分发可取得的平衡指数越高。$R_{AE}<1$ 时，$n_{\min}=S/s$；且 R_{AE} 越小，按最低需求量配置下的 I/O 资源冗余度越大，目标平衡指数越小，对负载平衡能力的要求越低。因此，仅当 R_{AE} 位于 1 附近的区间时，宜采用机制复杂的数据迁移技术；否则，采用机制较简单的多副本动态请求分发技术即可满足负载平衡需求。

结束语 本文将平衡指数作为负载平衡机制的评价指标。该指标独立于存储硬件参数，能够反映负载平衡机制本身的性质，并可通过仿真实验获得。通过大量仿真实验和数据分析，得出如下重要结论：1）系统的原始平衡指数随节点规模增长急剧下降。该结论明确了研究可扩展的适用于大规模存储系统的负载平衡方法的现实必要性。2）随着节点规模增长或文件访问偏斜度增加，保证相同 DRDAR 平衡指数所需副本数量急剧增长。该结论明确了迁移技术对解决负载平衡问题的必要性。3）应用特性和存储节点属性越接近，系统负载平衡难度越高，对负载平衡机制平衡能力的要求也越高。工程上可以据此进行负载平衡机制的选择。这里，应用特性用单位数据对访问负载量的平均贡献值表征，存储节点属性用单位存储空间对 I/O 带宽资源的平均占有量表征。

总之，通过对平衡指数影响因素的讨论以及对多副本动态请求分发机制平衡能力的分析，本文论证了研究较复杂的数据迁移技术的现实必要性；同时，本文为负载平衡方法的选择提供了重要依据，可用于指导工程上存储系统的设计。负载平衡将有力提高系统对资源的利用能力，降低系统对资源总量的需求；在节约设备成本的同时，减少了系统总体能源成本，这是实现绿色存储的重要保证之一。

参 考 文 献

[1] 葛雄资，冯丹，陆承涛，等. 绿色网络存储系统的动力学分析模型[J]. 计算机科学，2011，38(8)：291-296

[2] Breslau L，Cao P，Fan L，et al. Web Caching and Zip-like Distributions：Evidence and Implications [C]// IEEE International Conference on Computer Communications. 1999：126-134

[3] Hall J，Hartline J，Karlin A R，et al. On Algorithms for Efficient Data Migration [C]// The 12th Annual ACM-SIAM Symposium on Discrete Algorithms. 2001：620-629

[4] Ma Y C，Chiu J C，Chen T F，et al. Variable-Size Data Item Placement for Load and Storage Balancing [J]. The Journal of Systems and Software，2003，66(2)：157-166

[5] Xie T，Sun Y. A File Assignment Strategy Independent of Workload Characteristic Assumptions [J]. ACM Transactions on Storage，2009，5(3)：1-24

[6] Maccormick J，Murphy N，Ramasubramanian V，et al. Kinesis：a New Approach to Replica Placement in Distributed Storage Systems[J]. ACM Transactions on Storage，2009，4(4)：11：1-11：28

[7] Lumb C R，Golding R，Ganger G R. D-SPTF：Decentralized Request Distribution in Brick based Storage Systems [C]// The 11th International Conference on Architectural Support for Programming Languages and Operating. 2004：37-47

[8] Verma A，Sharma U，Jain R，et al. Compass：Optimizing the Migration Cost vs. Application Performance Tradeoff [J]. IEEE Transactions on Network and Service Management，2008，5(2)：118-131

[9] Kari C, Kim Y A, Russell A. Data Migration in Heterogeneous Storage Systems [C]// International Conference on Distributed Computing Systems. 2011:143-150

[10] Ghemawat S, Gobioff H, Leung S T. The Google File System [C]// The 19th ACM Symposium on Operating Systems Principles. 2003:29-43

[11] Dowdy W, Foster D. Comparative Models of the File Assignment Problem [J]. ACM Computing Surveys, 1982, 14(2):287-313

[12] Serpanos D N, Georgiadis L, Bouloutas T. Mmpacking: A Load and Storage Balancing Algorithm for Distributed Multimedia Servers [J]. IEEE Transactions on Circuits and Systems for Video Thechnology, 1998, 1(8):13-17

[13] Rowstron A, Druschel P. Storage Management and Caching in PAST, a Large-scale, Persistent Peer-To-Peer Storage Utility [C]// The 18th ACM Symposium on Operating Systems Principles. 2001:188-201

[14] Wei Q, Veeravalli B, et al. CDRM: A Cost-effective Dynamic Replication Management Scheme for Cloud Storage Cluster [C]// IEEE International Conference on Cluster Computing. 2010: 188-196

[15] Xie C, Cai B. A Decentralized Storage Cluster with High Reliability and Flexibility [C]// The 14th Euromicro International Conference on Parallel, Distributed, and Network-Based Processing. 2006:116-123

[16] Wang W, Zhao Y. A Novel Network Storage Scheme: Intelligent Network Disk Storage Cluster [C]// IEEE International Conference on Networking, Sensing and Control. 2008:142-147

[17] Kim J, Chou J, Rotem D. Energy Proportionality and Performance in Data Parallel Computing Clusters [C]// International Conference on Scientific and Statistical Database Management, LNCS 6809. 2011:414-431

[18] Thereska E, Donnelly A, Narayanan D. Sierra: Practical Power-proportionality for Data Center Storage [C]// Eurosys. 2011: 169-182

[19] Amur H, Cipar J, et al. Robust and Flexible Power-proportional Storage [C]// The 1st ACM Symposium on Cloud Computing. 2010:217-228

[20] Nakamura S, Shudo K. MyCassandra: a Cloud Storage Supporting both Read Heavy and Write Heavy Workloads [C]// The 5th Annual International Systems and Storage Conference. 2012:14

(上接第 37 页)

库 Libgomp 版本，比较同一 OpenMP 并行应用使用处理器硬件锁和软件锁的性能差异。

实验平台为实现了硬件锁的 FT 处理器仿真器，该处理器有 16 核 64 线程(每核 4 线程)。使用的 OpenMP 并行程序为 NPB[6] 测试集中的 LU 和 MG，以及专门用于测试的 OpenMP 栅栏同步的测试用例 Syncbench[7]。表 1 列出了在不同线程数目、不同同步次数情况下，OpenMP 程序在使用硬件锁和使用软件锁情况下的性能差异。

表 1 使用 OS 管理的硬件锁时应用加速比

程序	并行线程数	Barrier 同步次数	加速比 (硬件锁/软件锁)
Lu. A. x	32	73791	0.8%
Mg. A. x	32	25087	0.2%
Syncbench	32	6057503	1.6%
Lu. A. x	64	147583	1.2%
Mg. A. x	64	50175	0.6%
Syncbench	64	12115007	2.4%

可见，对于科学计算类的应用程序，如 NPB 中的 Lu. A. x 和 Mg. A. x，使用处理器提供的硬件锁，加速效果非常不明显；如果把多次运行的误差考虑在内，使用硬件锁几乎没有加速效果。对于专门用来测试同步的 Syncbench 题目，使用硬件锁，性能稍有提高。

性能提高不明显的一个重要原因是硬件锁的申请和释放需要调用操作系统的系统调用来完成，这会给使用硬件锁带来比较大的系统开销。对于同步数非常多的高并发应用，如果可以考虑在库中固定硬件锁地址(固定使用几个硬件锁)，省去调用操作系统来分配硬件锁和释放硬件锁的过程，可以获得更进一步的性能提升。

表 2 针对 Syncbench 应用，采用使用固定地址的硬件锁方法进行了性能对比测试。可见，使用硬件锁的性能加速比有一些提高。

表 2 使用固定硬件锁的应用加速比

程序	并行线程数	Barrier 同步次数	加速比 (硬件锁/软件锁)
Syncbench	32	6057503	2.2%
Syncbench	64	12115007	3.2%

综合以上测试，可见针对 FT 处理器实现的这种硬件锁机制，应用程序必须在并发度非常大的情况下，才能获得明显的性能提升。

结束语 在多核多线程的高并行度处理器中，针对同步操作提供硬件实现的加速机制是处理器设计中的一个趋势。本文针对 FT 处理器提供的硬件锁机制，在多线程库中使用该处理器的硬件锁实现了栅栏同步机制，并对效果进行了评估和分析。本文的工作对处理器设计以及多线程并行库的设计有重要的参考价值。

参 考 文 献

[1] UltraSPARC Architecture 2007 [S]. Draft D0. 9. 1, 01 Aug 2007:423

[2] Intel 64 and IA-32 Architectures Software Developer's Manual, Combined Volumes: 1, 2A, 2B, 3A and 3B [S]. Intel Corporation, May 2011

[3] OpenMP Application Program Interface. Version 3.0 [S]. OpenMP Group, May 2008. http://www. open mp. org/

[4] The GNU OpenMP Implementation [OL]. http:/gcc. gnu. org

[5] 李春江，杜云飞，易会战，等. GCC 中内嵌函数实现剖析[J]. 计算机科学，2012(6A)

[6] NASA Parallel Benchmark 3. 3. 1[OL]. http://www. nas. nasa. gov/publications/npb. html

[7] Syncbench [OL]. http://trac. mcs. anl. gov/projects/perfor mance/browser/benchmarks/openmpbench-C-v2/syncbench. c? rev =666