doi:10.11887/j.cn.202002008 http://journal.nudt.edu.cn

异构混合存储的软硬件协同数据放置策略*

李鸿飞[1,2],杜溢墨[2],曾 熠[2],王 磊[2]
(1. 信息工程大学 地理空间信息学院,河南 郑州 450001;2. 中国人民解放军31008部队,北京 100091)

摘 要:分析比较当前大数据中心典型混合存储架构,针对其不能综合运用存储管理系统和存储设备优势的问题,提出软硬件协同的数据放置策略,同时考虑软件层混合存储管理系统和硬件层混合存储设备的特点,根据应用特性为数据选择合适的存储管理系统和设备。面向不同应用场景,提出运行前规划存储路径的静态放置模式和运行中规划存储路径的动态放置模式。基于存储管理系统和设备性能参数建模,采用仿真方法实现各数据放置策略,并运行实际应用中的三类负载进行实验测试,结果表明采用软硬件协同的数据放置策略相比只考虑存储管理系统和存储设备的数据放置策略,性能可以提高近30%。

关键词:大数据;混合存储;非关系型数据库;数据放置
中图分类号:TP311.13 **文献标志码**:A **文章编号**:1001-2486(2020)02-064-08

Software and hardware co-design of data placement in heterogeneous hybrid store

LI Hongfei[1,2], *DU Yimo*[2], *ZENG Yi*[2], *WANG Lei*[2]
(1. Institute of Surveying and Mapping, Information Engineering University, Zhengzhou 450001, China;
2. The PLA Unit 31008, Beijing 100091, China)

Abstract: The analysis of the existing typical hybrid store architectures in big data center found that they cannot fully take advantage of both hybrid storage management systems and hybrid storage devices. Thus a hardware and software co-design data placement strategy was proposed, which simultaneously considers the hybrid storage management systems at software level and the hybrid storage devices at hardware level, and figures out the trail of data placement on both storage systems and devices regarding application characteristics. Moreover, the static placement pattern before running and the dynamic placement pattern during running were proposed on the basis of different application scenes. An experiment was implemented by running three kinds of workloads on simulated data placement strategies based on the model according to the performance parameters of storage management systems and storage devices. The results show that the proposed design outperforms traditional ones that either consider storage management systems or storage devices separately by up to 30%.

Keywords: big data; hybrid store; not only structured query language; data placement

随着网络信息技术的发展,特别是近年来,智能移动终端和多媒体技术深入社会生活各个领域,数据爆炸增长的趋势愈加显著。根据国际数据公司IDC发布的研究报告,互联网数据总量已达40 ZB[1]。除了体量大以外,数据还呈现出类型多、价值密度低、实时性强等"大数据"特性。

为了高效存储管理"大数据",学术界和产业界面向不同应用特性提出了各类存储管理系统和存储设备:存储管理系统主要有关系型数据库、非关系型数据库(Not only Structured Query Language, NoSQL)存储和以HDFS(Hadoop distributed file system)为代表的分布式文件系统等;存储设备主要有传统磁盘、闪存固态盘和相变存储器(Phase Change Memory, PCM)、自旋扭矩转换随机存储器(Shared Transistor Technology Random Access Memory, STT-RAM)、阻变式存储器(Resistive Random Access Memory, RRAM)等新型存储器件。这些存储管理系统和存储设备特点各异,适用于不同应用类型,但大数据环境下的应用复杂多样,批处理、流处理、交互式应用可能同时存在,而这些应用在数据类型、访问特性、时效要求上都有很大不同,单一存储管理系统和存储设备难以满足存储需求。混合存储,因能综合利用各类存储系统和设备的优点,成为一种较好选择,被企业、数据中心等广泛采用[2]。

当前的混合存储设计或只考虑了软件层面存

* 收稿日期:2019-03-02
作者简介:李鸿飞(1974—),男,黑龙江哈尔滨人,高级工程师,硕士,E-mail:lihongfei18188@126.com

储系统的混合,或只考虑了硬件层面存储设备的混合,无法综合利用混合存储系统和混合存储设备的优势。一种简单的软硬混合存储设计是将两个层面的混合进行堆叠,软件层面采用关系型数据库、非关系型数据库以及分布式文件系统混合的存储管理模式,硬件层面采用机械硬盘与固态盘混合的模式。但该设计需要进行两次数据特征识别和数据分配,既造成时间和资源的浪费,又由于数据分配没有同时考虑存储系统和存储设备,无法从全局角度为要存储的数据规划最合适的存储系统和存储设备。

本文提出的软硬件协同的数据放置策略,虽然位于存储系统之上,但除了能感知混合存储系统外,也能通过配置文件感知底层存储设备,相比软件层混合与硬件层混合简单堆叠的方法,只需通过一次特征识别和数据分配就能为数据选择合适的存储管理系统和存储设备,提高了存储执行效率。为适应不同需求,在软硬协同数据放置基础上,又提出了静态放置和动态放置两种放置模式。静态放置是在数据分配前,根据应用、存储系统、存储设备特性为不同应用事先规划好数据存放路径。动态放置是根据应用访问历史记录预测下一时刻应用特征,然后为数据选择适合存放的存储系统和存储设备,相比静态数据分配策略,其对应用识别的粒度更细,合理性更好,缺点是记录数据存放位置的元数据信息多,增加了时间和空间开销。

1　现状综述

1.1　NoSQL 存储

NoSQL 存储相比关系型数据库,具有可扩展性好、存储模式灵活、数据模型与应用层数据结构接近等优势,越来越多地用于大数据存储环境中,包括 Key-Value 存储、列家族存储、文档存储、图存储等。Sadalage 等[3]全面介绍了 NoSQL 存储,包括起源、特性、典型应用等。Catell[4]比较了结构化查询语言(Structured Query Language, SQL)和几种典型的 NoSQL 数据库,并给出了不同类型数据库的适用场景。综合分析几种典型的 NoSQL 存储,其主要特点如表 1 所示。

1.2　混合存储管理系统

Nance 等[5]认为虽然 NoSQL 在大数据时代应用广泛,但关系型数据库仍有生存空间和应用场景,涵盖 NoSQL 与关系型数据库的混合存储管理系统就被很多数据中心采用。联想采用混合存储架构为智慧城市提供数据存储,其中,12 PB 的非结构化数据采用分布式 NoSQL 存储,33 TB 的结构化数据采用关系型存储[6]。淘宝推出的数据魔方产品主要提供行业数据分析、店铺数据分析功能,其存储层采用 OldSQL + NoSQL 混合模式,由基于 MySQL 的分布式关系型数据库集群 MyFOX 和基于 HBase 的 NoSQL 存储集群 Prom 组成[7]。安洋等[8]提出了行列混合的存储布局,可实现实体的行式存储或列式存储,同时在一个实体内部,可针对不同字段的稀疏、稠密等数据特征,选择适合的行存储或列存储格式。Huang 等[9]提出了一种基于 NoSQL 与 SQL 混合数据库管理系统的负载均衡机制,动态检测系统中的热点,并将热点上的数据迁移到冷节点上,从而提高混合存储系统的吞吐量。Grolinger 等[10]分析比较了 NoSQL 和 NewSQL 数据库以及它们各自适用的场景,并为数据库系统设计人员选择合适的数据存储提供指导和借鉴。Lawrence[11]针对 NoSQL 数据库访问接口各异且不支持 SQL 的问题,提出了一种统一架构,将 NoSQL 存储与 SQL 存储集中统一管理,通过设置虚拟层,能将 SQL 和 Java 数据库连接(Java DataBase Connectivity, JDBC)访问转换成对 NoSQL 的访问。

表 1　典型的 NoSQL 存储及其特征

Tab. 1　Typical NoSQL stores and characteristics

数据模型	典型产品	主要特性
Key-Value 存储	Berkeley DB, Level DB, Memecached, Project Voldemort, Redis, Riak	聚合模型,聚合结构可以任意组织,且对用户透明
文档存储	CouchDB, MongoDB, RavenDB, Terrastore	聚合模型,聚合结构要按照设定要求组织,用户可以看到其内部结构
列家族存储	HBase, Cassandra, Amazon SimpleDB, Hypertable	双层聚合模型,有基于行和基于列两种组织方式,多个列聚合成一个列家族,并且可以自由添加列
图存储	FlockDB, Neo4J, HyperGraphDB, Infinit Grapy	非聚合模型,记录小,但记录间的连接关系复杂

1.3　混合存储设备

闪存、PCM 等新型存储器件的引入,改变了

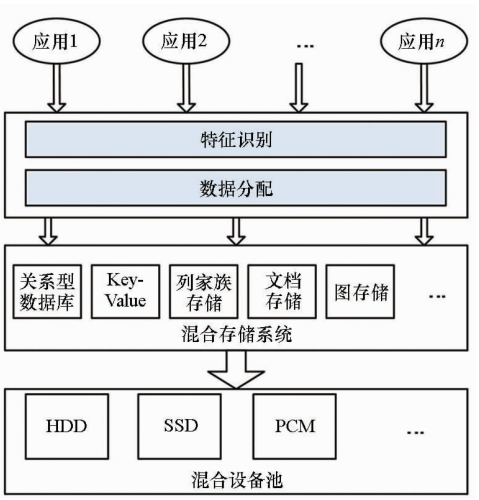
以往基于动态随机存取存储器(Dynamic Random Access Memory, DRAM)、磁盘的存储架构。由于存储器件特性各异,而大数据应用需求千差万别,综合应用各类存储设备的混合存储架构为解决大数据存储瓶颈提供了新思路。Ouyang 等[12]指出,百度采用混合存储设备,将经常访问的网页和图片等数据存储在定制化的固态盘(Solid State Drive, SSD)上,不经常访问的数据存储在硬盘(Hard Disk Drive, HDD)上,提升了整体数据访问性能。金培权[13]针对大数据存储与管理中的高效存储、实时处理等挑战,研究了通过引入闪存、PCM 等新型存储器件构建混合大数据存储架构的解决思路,包括基于 DRAM 与 PCM 的混合主存架构,DRAM、闪存与磁盘共存的混合层次架构,冷热数据分离的混合分布式存储等。Sang 等[14]提出采用 SSD 与 HDD 混合的存储架构,利用 SSD 读和大块写性能比 HDD 要好的特点,用 SSD 代替部分 HDD 存储事务日志、回滚段、临时表空间等,可以显著提高数据库应用性能。Canim 等[15]提出使用 SSD 代替部分 HDD,作为数据系统缓存区的扩展,可以有效减少 HDD 上的 I/O 访问,显著提高数据库性能。

2 软硬协同混合存储

2.1 问题描述

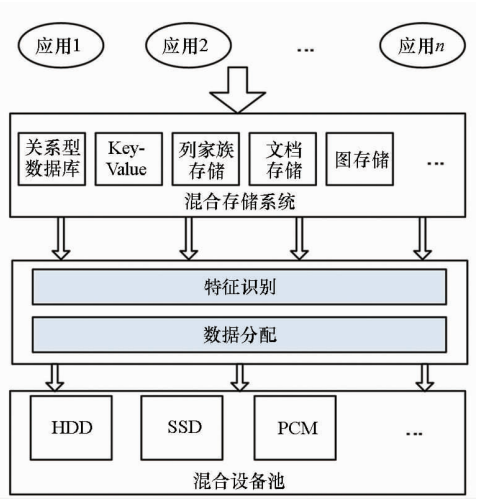
当前的大数据中心具有以下几个特点:一是服务的应用类型多样,包括事务处理、在线分析、实时交互等;二是存储管理系统多样,包括关系型数据库、Key-Value 存储、列家族存储等;三是存储设备多样,包括 HDD、SATA SSD、NVMe SSD 等。为了同时服务不同类型的应用,综合利用不同存储系统和存储设备的优势,数据中心通常采用混合存储体系架构:软件层包含各类存储管理系统的混合,硬件层包含各类存储设备的混合。对于混合存储架构,需要考虑的一个重要问题是数据放置,即为数据选择合适的存储管理系统和存储设备。通常,数据放置可分为两步:第一步是特征识别,识别对象包括应用、存储系统、存储设备;第二步是数据分配,根据第一步识别的特征为数据选择合适的存放位置。

图 1 给出了混合存储架构中各类数据放置策略设计,其中图 1(a) ~ (b)是当前已有的,图 1(c)是图 1(a)和图 1(b)的结合,图 1(d)是本文提出的。图 1(a)的数据放置模块位于应用层与存储管理系统层之间,数据放置时只考虑应用和存储管理系统的特征。图 1(b)的数据放置模



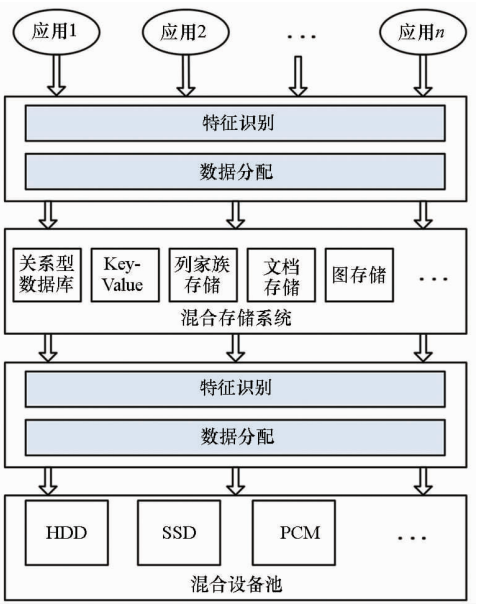
(a) 只考虑混合存储系统

(a) Only considering hybrid storage management systems



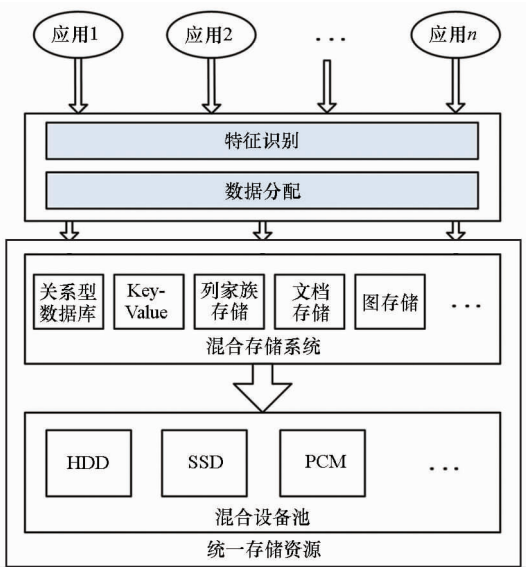
(b) 只考虑混合存储设备

(b) Only considering hybrid storage devices



(c) 分别考虑混合存储系统和混合存储设备

(c) Considering hybrid storage management systems and hybrid storage devices respectively



(d) 同时考虑混合存储系统和混合存储设备

(d) Considering hybrid storage management systems and hybrid storage devices simultaneously

图 1　混合存储架构中各类数据放置策略设计

Fig. 1　Data placement strategies for hybrid store architecture

块位于存储管理系统与存储设备之间,数据放置时只考虑存储管理系统和存储设备的特征。这两类数据放置策略都只考虑了应用、存储管理系统、存储设备三个要素中的两个,显然不是全局最优。如果综合考虑三个要素,简单直观的设计是将图 1(a)、(b)结合,设置两个数据放置模块,如图 1(c)所示。然而,该设计虽然考虑了三个要素,但还存在两个问题:一是需要设置两层数据放置模块,开销较大;二是每一层的数据放置只考虑了其上下层,即两个要素的特性,虽然总体上涵盖了三个要素,但并没有将三个要素综合考虑,也难以达到全局最优。针对这两个问题,本文提出了软硬件协同混合存储数据放置策略,如图 1(d)所示,只有一层数据放置模块,位于应用层与存储系统层之间,但进行数据放置时不仅考虑这两层的特征,也能通过读取存储设备配置文件同时感知存储设备的特征。相比图 1(c),该设计有两个优势:一是只有一层数据放置策略,但包含了图 1(c)中两层数据放置的功能,减少了开销;二是数据放置策略一次性综合考虑应用特征、混合存储管理系统、混合存储设备三个要素,提高了数据放置的合理性。

图 1 四种策略的形式化描述如下:假设数据放置策略用 P 表示,影响放置策略的三个要素——应用、存储管理系统、存储设备分别用 A、S、D 表示,各种放置策略其实是 P 关于 A、S、D 的函数,分别用 F_1、F_2、F_3、F_4 表示,如表 2 所示。图 1(c)的放置策略使用了三个函数进行表示,F_3 是在 F_1、F_2 各自最优解基础上的最优解,明显优于 F_2;图 1(d)的放置策略用 F_4 表示,综合考虑了 A、S、D,是(A,S,D)三维空间上的最优解,要优于 F_3 的最优解。具体证明过程此处不再赘述。

表 2　图 1 中各放置策略的数学表示和含义

Tab. 2　Mathematic expression and meaning of data placement strategies in the Fig. 1

体系架构	放置策略数学表示	含义
图 1(a)	$P=F_1(A,S)$	放置策略与应用和存储管理系统有关
图 1(b)	$P=F_2(S,D)$	放置策略与存储管理系统和存储设备有关
图 1(c)	$P=F_3(P_1,P_2)$ $P_1=F_1(A,S)$ $P_2=F_2(S,D)$	放置策略分为两步:第一步的放置与应用和存储管理系统有关;第二步的放置与存储管理系统和存储设备有关
图 1(d)	$P=F_4(A,S,D)$	放置策略一步完成,综合考虑应用、存储管理系统、存储设备

2.2　总体设计

图 1(d)展示了软硬件协同数据放置策略概要架构,图 2 进一步描述了数据放置策略的详细设计。图 2 中的数据放置模块是数据放置策略的核心,主要包括应用特征识别器、放置策略设置器、放置策略执行器、数据打包器和数据分流器五个部件。其中,应用特征识别器对所有应用的访问进行识别分类,生成应用特征参数,作为放置策略的一个输入。放置策略设置器读入应用特征参数、存储管理系统参数文件、存储设备参数文件,根据设定的算法生成存储路径信息,包括存储数据的存储管理系统和存储设备。放置策略执行器用于执行放置策略,将数据的存储路径作为元数据信息记入元数据记录表中。数据打包器根据元数据记录的存储路径信息给数据加上存储设备标签,用于混合存储管理系统选择存储设备位置。数据分流器根据元数据记录的存储路径信息,将打包好的数据分配到相应的存储系统。

图 2 还展示了软硬协同数据放置策略的数据存储流程。首先,各应用的数据访问请求发送到

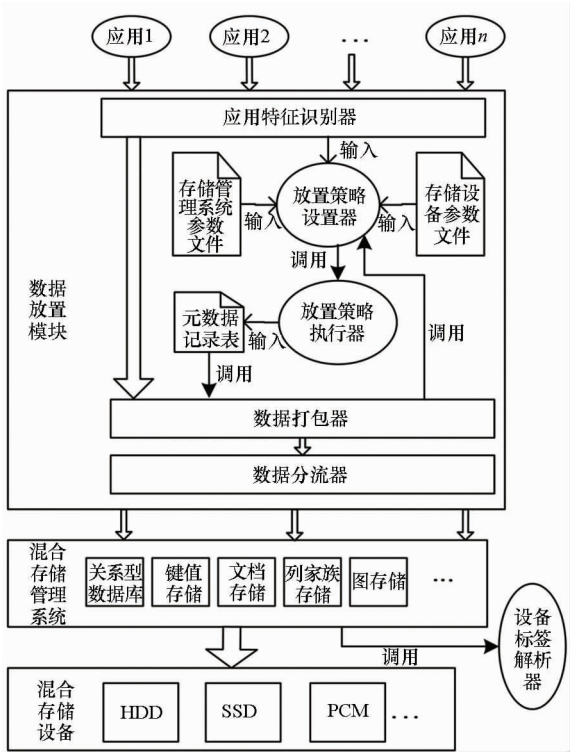


图2 软硬协同混合存储体系架构设计

Fig. 2 Software and hardware co-design of hybrid store architecture

应用特征识别器,经分析后生成应用特征结果,同时,访问请求发送到数据打包器。数据打包器接收请求后,判断请求类型,若是读请求,读取元数据记录表,给请求加上存储设备标签后发送到数据分流器。若是写请求,需三步完成:①调用放置策略设置器,放置策略设置器读取应用特征识别器生成的应用特征参数、存储管理系统参数文件、存储设备参数文件,生成数据存储路径;②调用放置策略执行器,将数据存储路径登记在元数据记录表中;③根据元数据记录的存储路径信息给请求加上存储设备标签,明确请求要存放的存储设备,然后将请求发送到数据分流器。数据分流器对接收的读写请求,读取元数据记录表,根据存储路径信息将请求分配到相应的存储管理系统。

存储管理系统采用各自存储管理模式对数据请求进行处理,然后调用设备标签解析器提取出数据请求中的存储设备信息,再将请求发送到相应的存储设备,具体在设备上的位置仍由存储管理系统自身管理。除了应用本身需要存储的数据,存储管理系统自身还会产生一些数据,用于存储的管理,比如索引、日志等,由于这些数据经常访问,通常存放在访问速度较快的存储设备上[11,14]。

2.3 数据放置策略设计

放置策略设置器根据应用、存储管理系统、存储设备类型及特性,为数据选择尽可能"最优"的存储路径。"最优"通过应用执行时间的长短来衡量,所选的存储路径使应用的执行时间越短,则路径越优。最优存储路径寻找问题和方法的形式化描述如下:

假设数据中心有 l 个应用,分别表示为 $A_1, A_2, \cdots, A_l$;m 个存储系统,分别表示为 $S_1, S_2, \cdots, S_m$;n 个存储设备,分别表示为 $D_1, D_2, \cdots, D_n$。为数据选择最优存储路径时,首先明确该数据所在的应用,假设为 $A_k(1 \leqslant k \leqslant l)$,为应用 A_k 的数据选择存储系统 $S_p(1 \leqslant p \leqslant m)$、$D_q(1 \leqslant q \leqslant n)$,使得应用执行时间 T 最短,T 可以表示成 $T(A_k, S_p, D_q)$,S_p、D_q 即为应用 A_k 中数据的最优存储路径。通过求解方程式(1)可得 S_p、D_q。

$$T(A_k, S_p, D_q) = \min(T(A_k, S_1, D_1), \cdots, T(A_k, S_1, D_n), T(A_k, S_2, D_1), \cdots, T(A_k, S_2, D_n), T(A_k, S_m, D_1), \cdots, T(A_k, S_m, D_n)) \quad (1)$$

方程左边表示应用 A_k 的数据在最优存储路径下的执行时间,右边表示应用 A_k 的数据在所有存储路径下最短的执行时间。应用执行时间与应用、存储管理系统、存储设备的部分特征有关,选取重要特征作为计算执行时间的依据,用式(2)表示。

$$T(A_k, S_g, D_h) = \boldsymbol{FA}_k \cdot \boldsymbol{I}_a + \boldsymbol{FS}_g \cdot \boldsymbol{I}_g + \boldsymbol{FD}_h \cdot \boldsymbol{I}_h \quad (2)$$

式中:$\boldsymbol{FA}_k$、$\boldsymbol{FS}_g$、$\boldsymbol{FD}_h$ 分别表示应用 A_k、存储管理系统 S_g、存储设备 D_h 的特征值向量;$\boldsymbol{I}_a$, $\boldsymbol{I}_g$, $\boldsymbol{I}_h$ 分别表示各特征值向量对执行时间的影响因子向量。应用特征向量 $\boldsymbol{FA}_k$ 包括读写比例、读写块大小、局部性强弱等要素;存储管理系统特征向量 $\boldsymbol{FS}_g$ 包括读写访问延迟、存储容量、一致性强弱等要素;存储设备特征向量 $\boldsymbol{FD}_h$ 包括设备读写访问延迟、吞吐量、存储容量等要素。$\boldsymbol{I}_a$、$\boldsymbol{I}_g$、$\boldsymbol{I}_h$ 可以通过在不同存储管理系统和存储设备组合的环境下运行负载得到应用执行时间后,根据式(2)进行估算。

根据不同应用场景,提出两种数据放置模式:静态放置和动态放置。

1)静态放置是在系统运行前就规划好数据存储路径的放置策略。采用静态放置策略时,应用、存储管理系统、存储设备的特征都根据经验和参数文件确定,对应用特征进行识别时,将各应用视为一个整体,一旦为整个应用规划好了存储路径,该应用产生的所有数据都按规划的途径存储。每种应用在各类存储路径下的执行时间在运行前根据式(2)计算,然后根据式(1),在系统运行前为应用选择出存储路径。例如:应用甲、应用乙分

别是在线事务处理和批处理应用,基于以往对此类应用特征的认识以及各类存储管理系统和存储设备的特点,为应用甲规划以下存储路径——关系型数据库、SATA SSD,为应用乙规划以下存储路径——Key-Value 存储、HDD。

2)动态放置是在系统运行过程中规划数据存储路径的放置策略。采用动态放置策略时,应用特征根据应用的历史访问记录动态分析识别,存储管理系统和存储设备特征根据系统实时状态确定,然后通过式(2)和式(1)计算出存储路径。应用、存储管理系统和存储设备的特征值在不同时段可能发生变化,根据式(1)和式(2)计算出的数据存储路径也相应跟着变化。相比静态放置,动态放置实时感知系统运行状态,对应用的感知粒度更细,能提高存储路径规划的合理性,但其实现有一定的时空开销,在实验部分将对其进行比较。

2.4　元数据设计

元数据记录表由一系列访问请求存储路径的记录组成,每条记录包含三部分,分别是逻辑地址、存储管理系统号和存储设备号,其设计如图 3 所示。

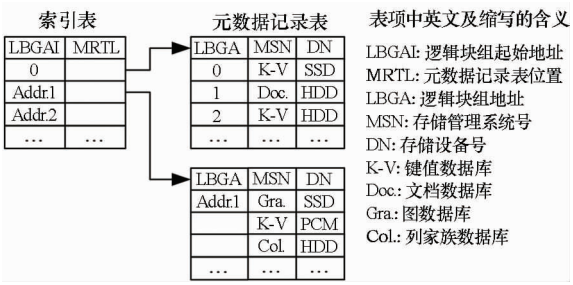


图 3　元数据设计

Fig. 3　Meta-data design

为减少元数据开销,采用两种优化方法:一是采用逻辑块组减少元数据记录表条目。在元数据记录表中记录逻辑块组地址,而非各逻辑块地址。根据设定的组大小,取逻辑块地址的前 x 位(x = 逻辑地址位数 $-\log_2$(组大小))作为逻辑块组地址。当逻辑块组确定好存储路径后,该组下的所有逻辑块均采用该存储路径。二是采用索引加速元数据搜索。将元数据记录表分为若干大小相同的子表,每个子表第一个表项的逻辑块组地址作为索引表中的逻辑块组起始地址,该子表自身地址作为元数据记录表位置,将它们依次记录在索引表中。根据逻辑块组地址查找元数据记录表时先查询索引,确定元数据记录子表后,再在子表中查询。子表大小根据索引表可用空间大小确定,索引表可用空间越大,子表越小,查询效率也越高。

3　实验测试及结果分析

3.1　实验配置

数据放置策略需要感知存储管理系统和存储设备的状态,部分存储管理系统和设备没有相关接口,故采用仿真方法实现各类数据放置策略并进行测试。对于存储管理系统和存储设备,根据官方参数文件对其性能进行建模。测试负载来源于实际场景,主要包括查询、更新、归档应用,根据系统运行特点,选取三个时间段,分别截取负载记录,并分析存储设备访问轨迹,特点如表 3 所示。

表 3　测试负载类型及特点

Tab. 3　Test workloads` types and characteristics

负载	请求类型	特点
负载 1	读:70% 更新:30%	08:00—13:00 截取的负载,以读为主,有部分更新
负载 2	读:95% 更新:5%	15:00—19:00 截取的负载,几乎都是读,只有少量更新
负载 3	更新: 100%	24:00—05:00 截取的负载,系统仅有更新

存储管理系统从关系型数据库和三类 NoSQL 存储中分别选取了典型的一种,关系型数据库性能来源于 Oracle 官方性能文档[16];NoSQL 存储性能来源于试验测试报告[17-19],对其在相同配置、负载量下的性能进行了分析,如表 4 所示。

表 4　存储管理系统性能参数

Tab. 4　Storage management systems performance data　　ms

存储管理系统	平均每条记录加载时间	平均读响应时间	平均写响应时间
键值数据库(Redis)	7.9E-2	1.1E-2	1.2E-2
文档数据库(MongoDB)	10.4E-2	1.5E-2	1.9E-2
列家族数据库(Cassandra)	13.5E-2	5.5E-2	5.3E-2

存储设备选取了 SAS HDD、SAS SSD 和 NVMe SSD 三类,根据官方参数文件[20-22],对其在相同配置、负载下的性能进行了分析,如表 5 所示。

表5 存储设备性能参数

Tab. 5 Storage devices performance data

存储设备	顺序读/(MB/s)	顺序写/(MB/s)	4 KB随机读/IOPS	4 KB随机写/IOPS
SAS HDD	400	300	200	150
SAS SSD	540	520	70 000	80 000
NVMe SSD	2500	1500	300 000	110 000

实验测试比较了五种放置策略:Str. a、Str. b、Str. c 是基准测试策略,Str. d. static 和 Str. d. dynamic 是本文提出的软硬件协同数据放置策略,分别表示静态放置和动态放置。对于 Str. d. static 和 Str. d. dynamic,从三种应用、四种存储管理系统和三种存储设备中选取可量化的典型特征值,如表6所示。

表6 应用、存储管理系统和存储设备所用的典型特征值

Tab. 6 Selected typical feature factors of applications, storage systems and devices

名称	特征值
应用	读写比例、块大小
存储管理系统	读延迟、写延迟
存储设备	容量、读延迟、写延迟

1)Str. a:基于混合存储管理系统的数据放置策略,数据放置只考虑存储管理系统类型和性能差异,对于存储设备,按比例轮流分配,即图1(a)所示策略。

2)Str. b:基于混合存储设备的数据放置策略,数据放置只考虑存储设备类型和性能差异,对于存储管理系统,按比例轮流分配,即图1(b)所示策略。

3)Str. c:基于混合存储管理系统和存储设备的软硬件独立数据放置策略,数据放置需分两次进行,分别针对混合存储管理系统和混合存储设备,即图1(c)所示策略。

4)Str. d. static:基于混合存储管理系统和存储设备的软硬件协同数据放置策略,数据放置一次性完成,同时考虑混合存储管理系统和混合存储设备特点,数据放置采用静态放置。

5)Str. d. dynamic:与 Str. d. static 一样,都是基于混合存储管理系统和存储设备的软硬件协同数据放置策略,区别是数据放置策略采用动态放置。

3.2 加载时间

在执行负载之前,需要对存储管理系统加载数据。为了比较不同数据放置策略下存储管理系统加载数据的性能,分别测试了10万、100万、500万条数据下,五种不同数据放置策略的数据加载时长,并以 Str. a 的加载时长为基础进行了标准化,如图4所示。在五种数据放置策略中,Str. b 的数据加载时间最长,Str. d. dynamic 相比 Str. c 可以缩短近25%的加载时间。数据加载量从10万条到100万条时,随着加载量变大,Str. d 的性能显著提升,但加载量从100万条到500万条时,性能提升不明显。这是由于在数据量为10万条时,带宽还有较大富余,而到100万条时,由于实验中网络带宽设置较低,Str. d 带来的性能改善受到了带宽瓶颈的限制。在真实数据中心,网络带宽较高,性能提升不会受带宽影响。

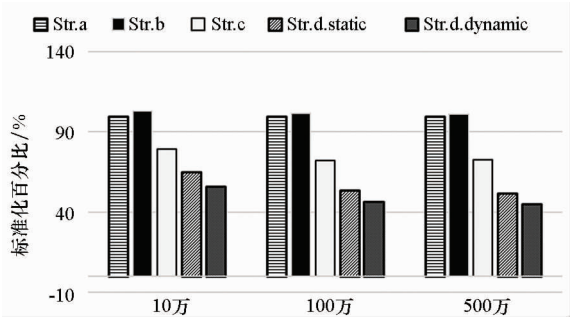


图4 不同数据量下五种数据放置策略数据加载时间

Fig. 4 Data loading time of 5 data placement strategies for different loading volume

3.3 执行时间

为了比较不同数据放置策略的性能,测试了五种不同数据放置策略在加载10万条数据记录后执行三类负载的执行时间,并以 Str. a 的执行时间为基础进行了标准化,如图5所示。三类负载下,Str. d. dynamic 都具有最短的执行时间,相比 Str. a,最大可缩短近30%;相比 Str. d. static,Str. d. dynamic 的执行时间可缩短约10%。

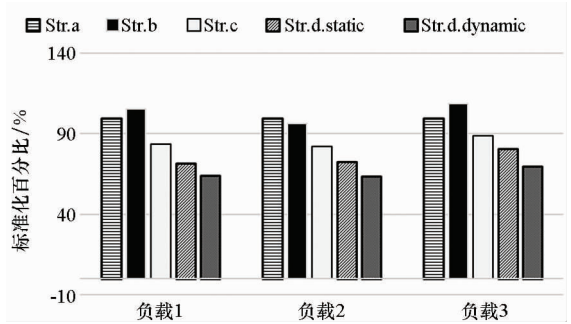


图5 五种数据放置策略执行三类负载的时间

Fig. 5 Execution time of 5 data placement strategies under 3 kinds of workloads

3.4 时空开销分析

本文提出的软硬件协同数据放置策略的空间开销主要是元数据所用空间,时间开销主要是查询元数据记录表所用时间。静态放置策略的元数据记录表大小只与应用个数、数据存储管理系统个数和存储设备个数相关,由于应用、存储管理系统和存储设备个数通常都不大,相比软硬独立设计的数据放置策略,其时空开销可以忽略。动态数据放置策略的元数据包括索引和元数据记录表两部分。元数据记录表每条记录大小为 10 B (LBGA:8 B,MSN:1 B,DSN:1 B),索引表每条记录的大小为 9 B(LBGAI:1 B,MRTL:1 B)。若逻辑块组大小设置为 16 KB,对于 1 TB 的数据量,所需元数据记录表大小为 640 MB,索引表大小为 6.4 MB。由于元数据存储在内存中,相比 I/O 访问,其时间开销可以忽略[23]。

4 结论

面向大数据应用,针对多存储系统和多存储设备并存的混合存储架构提出了软硬协同的数据放置策略,并根据应用需求设计了静态放置和动态放置两种模式。实验结果表明,软硬协同的数据放置策略可以提升数据访问性能,动态放置在有一定空间开销的情况下性能提升更为显著。

参考文献(References)

[1] Gantz J, Reinsel D. The digital universe in 2020: big data, bigger digital shadows, and biggest growth in the far east [R]. IDC: EMC, 2012: 1 - 7.

[2] Avram A. Hybrid SQL-NoSQL databases are gaining ground [EB/OL]. (2012 - 02 - 07)[2018 - 01 - 30]. https://www.infoq.com/news/2012/02/Hybrid-SQL-NoSQL.

[3] Sadalage P J, Fowler M. NoSQL distilled: a brief guide to the emerging world of polyglot persistence [M]. Pearson Education, 2013.

[4] Cattell R. Scalable SQL and NoSQL data stores [C]// SIGMOD Record, 2010, 39(4): 12 - 27.

[5] Nance C, Losser T, Iype R, et al. NoSQL vs RDBMS—why there is room for both [C]// Proceedings of the Southern Association for Information Systems Conference, 2013: 111 - 116.

[6] 毕巍. 开放共赢建设新型智慧城市[C]. 2017 世界电信和信息社会日, 2017.
BI Wei. Developing new smart city with open and win-win strategy[C]. 2017 World Telecommunication and Information Society Day (WTISD), 2017. (in Chinese)

[7] Wangshfa. NewSQL、NoSQL 与 OldSQL 之混合部署应用方案[EB/OL]. (2015 - 01 - 30)[2018 - 01 - 30]. http://blog.csdn.net/wangshfa/article/details/43308575.
Wangshfa. NewSQL、NoSQL and OldSQL: hybrid deployment [EB/OL]. (2015 - 01 - 30)[2018 - 01 - 30]. http://blog.csdn.net/wangshfa/article/details/43308575. (in Chinese)

[8] 安洋, 赵洪松. 基于行列混合存储的大数据存储方法研究与实现[J]. 通信管理与技术, 2014(1): 24 - 27.
AN Yang, ZHAO Hongsong. Big data store method research and implementation based on row and column hybrid store[J]. Communication Management and Technology, 2014(1): 24 - 27. (in Chinese)

[9] Huang H S, Hung S H, Yeh C W. Load balancing for hybrid NoSQL database management systems [C]// Proceedings of the Conference on Research in Adaptive and Convergent Systems (RACS), 2015: 80 - 85.

[10] Grolinger K, Higashino W A, Tiwari A, et al. Data management in cloud environments: NoSQL and NewSQL data stores[J]. Journal of Cloud Computing: Advances, Systems and Applications, 2013, 2(1): 5 - 22.

[11] Lawrence R. Integration and virtualization of relational SQL and NoSQL systems including MySQL and MongoDB [C]. International Conference on Computational Science and Computational Intelligence, 2014: 285 - 290.

[12] Ouyang J, Lin S D, Jiang S, et al. SDF: software-defined flash for web-scale internet storage systems[C]// Proceedings of the 19th International Conference on Architectural Support for Programming Languages and Operating Systems (ASPLOS), 2014, 49(4): 471 - 484.

[13] 金培权. 基于新型存储的大数据存储管理[J]. 大数据, 2017, 3(5): 70 - 82.
JIN Peiquan. Big data storage management based on new storage[J]. Big Data Research, 2017, 3(5): 70 - 82. (in Chinese)

[14] Sang W L, Bongki M, Chanik P, et al. A case for flash memory SSD in enterprise database applications [C]// Proceedings of the ACM SIGMOD International Conference on Management of Data, 2008: 1075 - 1086.

[15] Canim M, Mihaila G A, Bhattacharjee B, et al. SSD bufferpool extensions for database systems[C]// Proceedings of the Very Large Data Base (VLDB) Endowment, 2010, 3(2): 1435 - 1446.

[16] Oracle. Oracle extends performance leadership with x86 world record on TPC-C benchmark [R]. [2018 - 05 - 25]. http://www.oracle.com/us/corporate/press/1485280.

[17] Tang E Q, Fan Y S. Performance comparison between five NoSQL databases [C]// Proceedings of 7th International Conference on Cloud Computing and Big Data, 2016, 2016: 105 - 109.

[18] Veronika A, Jorge B, Pedro F. Experimental evaluation of NoSQL databases [J]. International Journal of Database Management Systems, 2014, 6(3): 1 - 16.

[19] Armstrong T G, Ponnekanti V, Borthakur D, et al. LinkBench: a database benchmark based on the Facebook social graph [C]// Proceedings of the ACM SIGMOD International Conference on Management of Data, 2013.

[20] Seagate. Desktop HDD data sheet[R]. [2018 - 05 - 25]. http://www.seagate.com/staticfiles/docs/pdf/datasheet/disc/desktop-hdd-data-sheet-ds1770 - 1 - 1212us.pdf.

[21] Samsung. 850 120G 2.5 inches SATAIII SSD[R]. [2018 - 05 - 25]. http://www.samsung.com/cn/memory-storage/850-pro-sata - 3 - 2 - 5-inch-ssd/MZ-7KE2T0BCN/.

[22] Samsung. 950 PRO NVMESSD [R]. [2018 - 05 - 25]. http://www.samsung.com/cn/memory-storage/950-pro-nvme-m-2-ssd/MZ-V5P512BW/.

[23] 肖利民, 霍志胜. 大数据存储系统 I/O 性能优化技术研究进展[J]. 大数据, 2017, 3(6): 65 - 84.
XIAO Limin, HUO Zhisheng. Review of I/O performance optimization technology for big data storage system[J]. Big Data Research, 2017, 3(6): 65 - 84. (in Chinese)