一种分布式缓存系统的关键技术及应用

屠要峰[1] 刘 辉[2] 张国良[2] 刘 春[2]
(南京理工大学计算机科学与工程学院 南京 210012)[1]
(河南大学计算机与信息工程学院 河南 开封 475000)[2]

摘 要 分布式缓存作为处理海量数据的关键技术方案,近年来被广泛关注和应用。通过分析业界分布式缓存的现状和缺陷,提出了一种分布式缓存系统架构。在此基础上,深入阐述了其关键技术原理,研发并实现了新一代的分布式缓存系统DCACHE。最后,在融合通信(RCS)业务应用中对DCACHE进行了分析和验证。

关键词 数据库,分布式缓存,NoSQL,云计算
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2018.05.026

Key Techniques of a Kind of Distributed Cache Systems and Their Applications

TU Yao-feng[1] LIU Hui[2] ZHANG Guo-liang[2] LIU Chun[2]
(School of Computer Science and Engineering, Nanjing University of Science and Technology, Nanjing 210012, China)[1]
(School of Computer and Information Engineering, Henan University, Kaifeng, He'nan 475000, China)[2]

Abstract As a key technique for mass data processing, distributed cache has received much attention and has been widely applied recently. By analyzing the current situation and the deficiencies of distributed cache, this paper proposed an architecture for a kind of distributed cache system. Further, the principles of the key techniques in this architecture were addressed. Based on this, the principles of its key technologies were deeply explained, and a new generation of distributed cache system DCACHE was developed and implemented. Finally, DCACHE was analyzed and validated in RCS (Rich Communication Suite) application.

Keywords Database, Distributed cache, NoSQL, Cloud computing

随着互联网和云计算[1]的高速发展,数据量呈爆炸式增长,许多新型应用对海量数据的处理提出了更高、更复杂的需求。例如,铁路售票12306系统要求百万级的并发用户访问、每秒数以万计的并发事务处理、在线的弹性伸缩、多种数据类型、低延时及7×24×365可用性等[2]。使廉价的、标准化的硬件和软件平台给大容量、业务关键型的应用提供良好的支撑,是当前应用开发者面临的挑战。分布式缓存[3]和NoSQL[4]的产生为解决该难题提供了高并发、低时延、多种类数据、海量数据存储和处理的技术解决方案。

1 现状

分布式缓存是NoSQL的一种重要实现方式,与其他NoSQL系统间并无清晰的界限[5]。它们作为替代传统关系型数据库的核心技术,具有广阔的应用场景。根据很多行业分析师的观点,如果没有分布式缓存这一关键技术的支撑,云计算的潜能,特别是数据密集型的应用,将会受到很大局限[6]。在Forrester研究公司2010年2季度的技术报告中,Gualtieri等人首次提出了弹性缓存平台(Elastic Caching Platforms)[3]的概念,并强调其关键特性是动态扩展性(Scalability)、故障容忍(Fault Tolerance)和高可用性(High Availability)[3]。

NoSQL通常泛指非关系型数据库[4]。它经历了Non-relational阶段和Not Only SQL阶段,从否定关系型数据库逐步向关系型数据库靠拢,其技术架构和功能在不断演进。当前主流的NoSQL数据库有四大类[4]:列存储数据库、文档型数据库、键值存储数据库、图数据库。图数据库不常见,在此不赘述。

常见的列存储数据库有Bigtable,Cassandra,Hbase等,通常用于稀疏的、分布式的、持久化存储的海量数据。列式存储具有较高的压缩率,但是功能相对局限。文档型数据库的典型代表是MongoDB,它的数据模型是版本化的文档,多个键及其关联的值有序地放在一起就构成了文档。MongoDB最大的特点是具有较强的数据表达能力和查询能力以及丰富的索引,但它也存在资源消耗大、磁盘IO负荷不可控及访问缓慢等缺陷。

键值存储数据库主要使用一个哈希表,这个表中有一个特定的键和一个指针指向特定的数据,其典型代表是Dyna-

到稿日期:2017-04-06 返修日期:2017-07-09 本文受深圳市科技创新委员会科技应用示范项目资金(sf20170036)资助。
屠要峰(1972-),男,教授,CCF会员,主要研究方向为云计算、大数据及人工智能,E-mail:13605151819@qq.com(通信作者);刘 辉(1979-),男,副教授,主要研究方向为大数据、人工智能及信息安全,E-mail:xdliuhui@163.com;张国良(1993-),男,硕士生,主要研究方向为大数据;刘 春(1982-),男,副教授,主要研究方向为软件工程、大数据及人工智能。

mo 系统。我们的第一代分布式缓存[8]就属于这种架构,它采用改良的一致性哈希算法+NRW 副本策略,适用于高并发、低时延的场景,具有良好的可靠性和扩展性,可灵活制定策略以实现对 CAP 理论 3 个维度的取舍;其缺点是不支持数据类型,且对于只更新或查询 Value 中部分值的场景效率较低。

当前应用热度最高的分布式缓存系统是 Redis,它支持丰富的数据类型,性能优越,适用于数据变化快且数据规模与内存容量匹配的场景。然而,由于只支持数据全部驻留内存、服务端单线程等架构,其应用范围受到限制。

可见,尽管在实际应用中已经部署了大量的 NoSQL 系统,但这些解决方案都是针对不同的应用场景和需求而提出的,都存在明显的局限。部分主流 NoSQL 系统正尝试着实现所有的特性,使得明确的边界消失,但这些系统的能力参差不齐,缺乏清晰的规划和质量保障,使得确定使用哪种系统比以前更加复杂。

本文针对当前云计算中分布式缓存系统的困境和不足,总结了第一代分布式缓存系统在实际应用中的经验教训,借鉴主流开源软件的优点,结合最新技术的发展趋势,提出并研制了新一代分布式缓存系统 DCACHE。该系统兼顾多样化业务需求,支持多种数据类型和检索方式以及类 SQL 语法,同时具备分布式、高并发、低时延、易扩展等特点和优势,在全球近百个业务局点中被部署和使用,其设计目标和优势得到验证。

2　DCACHE:新一代分布式缓存系统

DCACHE 系统的关键设计目标有:1)为大规模数据应用提供 TB 级数据缓存平台,支持 Integer,String,List,MAP 等常用数据类型,满足多样化的业务需求;2)支持内存、SSD 间数据的智能调度和分布,可在线动态扩展;3)高并发、低时延、高可靠、高可用;4)智能运维,最小人工介入。

DCACHE 采用完全无中心化的系统架构(见图 1),集群中各个数据节点完全对等,且不存在任何 Master 节点,属于名副其实的无共享体系架构[8]。它具有高度可扩展性和可用性,通过简单增加/移除节点即可线性扩展/缩减存储能力,消除了单点故障,拥有较强的 Failover 特性[9]。

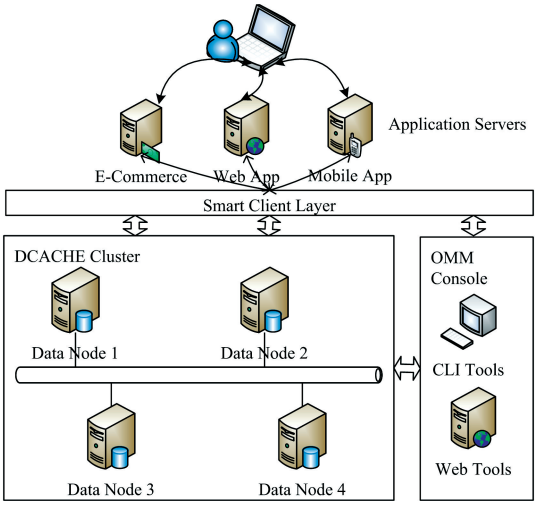

图 1　DCACHE 的系统架构

Fig. 1　System architecture of DCACHE

DCACHE 系统由智能客户端、运维管理控制台和 DCACHE 服务器集群 3 个模块组成。

智能客户端实现了访问 DCACHE 集群的 API 接口,支持 C,C#,Java,PHP,Python,Go,Perl 等编程语言的动态链接库或驱动程序包,是应用程序访问 DCACHE 集群进行数据存取的主要方式。智能客户端能够自动感知整个集群的变化,跟踪各个节点状态,缓存整个集群所有名字空间的路由信息表,知晓数据在集群中的具体存储位置,即根据 Key 值可以自动计算出管理该记录的数据节点,并将请求直接发送到正确节点上,减少了事务处理时延。

运维管理控制台负责整个集群的运维管理与监控,提供了两种操作方式:基于命令行的客户端工具和基于 Web 页面的图形化管理界面。二者均能接入到系统中,实现对集群状态的监控和管理,可以动态查看 CPU、内存、磁盘空间、副本状态、各个数据节点的负荷、存储的记录数等统计信息,并实现对异常状态的告警管理功能。

DCACHE 服务器集群是 DCACHE 的核心模块,在软件架构上分为集群与数据分布层和数据存储层两个层面。

集群与数据分布层实现管理集群内部节点间通信、自动故障切换、副本复制与数据同步、智能数据再平衡与数据迁移等功能。它基于无共享架构,实现了线性扩展能力和许多 ACID 事务特性以及集群管理功能的自动化,包括集群管理模块、数据迁移模块和事务处理模块 3 个模块。集群管理模块跟踪集群内节点间的连通性,其关键算法是基于 Paxos 的一致性投票过程,决定哪些节点属于集群的一部分。当添加或者删除一个节点时,DCACHE 集群重新确定成员关系。每个节点使用分布式哈希算法来划分数据分区,并为每个分区指定属主。数据迁移模块智能地在集群的所有节点间对数据分布进行再均衡,以确保每一位数据都被有效管理,且依据副本策略在节点间进行复制。整个数据的分区再划分过程是纯算法的,不需要专门的 Master 节点来协调,因此也就不需要其他共享架构所要求的额外设施。事务处理模块负责根据用户的请求进行读写操作,并提供一致性和隔离性保证。它负责同步/异步复制副本数据、重定向用户请求并解决重复记录版本冲突。为了避免一些糟糕的用户请求(比如批量操作、超大结果集聚合运算)干扰整个集群的总体读写性能,DCACHE 采用小步分批快速处理技术来处理严重消耗资源的用户请求,以非侵入式框架设计数据聚合运算流程,使得整个处理过程对 CPU、内存和 I/O 的消耗完全可控。而这一点恰恰是 MongoDB 被广为诟病的缺陷。

数据存储层实现数据在内存和磁盘中的可靠存储和快速检索。DCACHE 数据存储层采用特殊设计,兼顾了访问速度和硬件成本两个相互矛盾的设计目标,既支持数据全部驻留内存的存储模式,又支持数据在内存和 SSD 间智能分布的存储模式。它突破了数据存储规模受限于物理内存大小的局限,可支持 TB 级数据的存储,这是 Redis 和 Memcached 等纯内存缓存系统所不能比拟的。DCACHE 数据存储层还针对 SSD 等 Flash 设备做了专门的优化,充分利用 Flash 设备随机高速读写的特性,简化存储组织形式,提高流程处理效率;同时采用多线程运行模式,充分利用 CPU 多核并行处理 I/O 请

求来提高系统的并发度。

DCACHE 数据存储层采用无模式的数据模型，表名和字段名都在运行时动态指定，无需遵守某个特定且严格的模式，每个字段存储的都是强类型的值，既可以是内置的各种数据类型(Integer，String，Blob，Double 等)，也可以是复合数据类型(如 List，Map 等)；并且不同记录的相同字段可以存储不同类型的数据。DCACHE 支持的数据类型涵盖了 MongoDB 和 Redis 所支持的数据类型，为多样化的应用提供了丰富的数据类型和灵活的存储方式。

索引是快速获取数据的最有效的手段。传统关系数据库最大的优势在于它能够支持各种类型的索引，从而适应各种各样的复杂业务逻辑。DCACHE 吸收了传统关系数据库的优点，不仅支持主键索引，还支持二级索引，从而有效地满足了各种复杂应用场景的需要。为了实现对记录的极快访问，所有索引全部驻留在内存中。

为了保证内存与磁盘空间的可用性，数据存储层启用了智能碎片整理器和智能记录清除器两个后台例程。智能碎片整理器跟踪各个数据块内部的活跃记录数，当该值低于某个阈值时自动回收该数据块。智能记录清除器则根据配置的策略回收过期记录，并释放相应的内存空间和磁盘空间。

先进的架构和关键环节的设计优化，使得 DCACHE 有较广的适用范围。其典型的使用场景有如下 4 种。

1)大容量高速缓存：可以缓存只读配置数据、业务过程数据、中间计算结果和临时数据，还可以缓存页面数据等。

2)小文件持久化存储：对性能和存储容量要求较高的应用可以使用 DCACHE 替换分布式文件系统来存储小于 1MB 的小文件。比如，在彩信系统中存储彩信内容，在网盘系统中存储小图片文件。

3)高性能数据存储：作为 NoSQL 数据库使用，可以存储任意数据类型的 Key-Value 值，并提供持久化、可靠性和一致性保障。

4)关系数据库结果集缓存：可以缓存关系数据库的查询结果集，避免频繁、重复访问关系数据库系统，减轻其负荷并提高二次访问的性能。

3 关键技术

3.1 动态无模式的数据模型

DCACHE 采用无模式数据模型，存储数据记录之前无须预先定义表结构和字段类型，与互联网＋时代各类跨界融合应用的开发潮流相契合。

为了更好地管理数据，DCACHE 采用多级容器来组织数据记录，如图 2 所示。将整个分布式缓存系统看成一个海量数据的存储仓库，并根据不同业务场景创建多个名字空间。名字空间是最顶层的物理容器，每个名字空间可以单独配置存储属性，如存储介质、副本个数、缺省 TTL 等。名字空间与传统关系数据库的 database 概念相当。在名字空间之下还可以创建不同的数据表，这是一个逻辑容器，无须显式创建。数据表中可以存储任意多条记录，记录是 DCACHE 的最小存储单位，每一条记录都有一个唯一的主键 ID 以及一段元数据信息，如该记录的版本号、TTL 等。

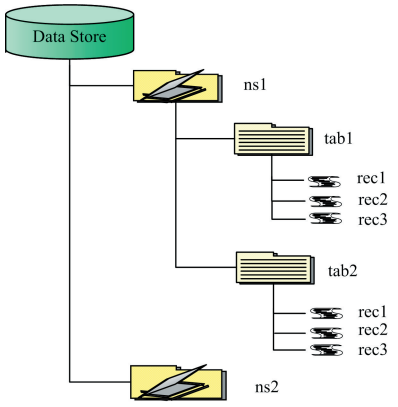

图 2 数据模型

Fig. 2 Data model

记录也是一个逻辑容器，真正的数据被存储在它的字段中。一条记录可以包含多个字段，每个字段可以存储不同类型的键值数据。Integer，String，Blob，Double 是内嵌的基本数据类型。Map 和 List 为复合数据类型，它们的内部可以包含多个值，每个值的类型可以是基本数据类型，也可以是复合数据类型。复合数据类型是支持嵌套的，可以通过复合数据类型和基本数据类型的任意组合来描述现实世界的各种复杂对象关系。DCACHE 复合数据类型可以与 MongoDB 的文档类型媲美。

DCACHE 的数据模型与传统关系数据库非常相似，然而它们在本质上却是截然不同的。传统关系数据库的表、记录及其字段的构成和数据类型都是在第一条数据记录存储之前预先定义好的，属于有模式的数据模型，这些定义构成了传统数据库数据字典的一部分。DCACHE 的表和字段是在数据存储的过程中动态创建的，无须定义，因此属于无模式数据模型。DCACHE 的数据模型比较灵活，同一张表的不同记录可以包含不同名称的字段，不同记录的同名字段可以存储不同类型的数据。

存储的本质功能是为了有效检索数据。为了帮助用户快速检索数据记录，DCACHE 不仅支持主键索引，还引入了传统关系数据库才有的二级索引功能，使得数据查询能力有了本质的提升。除了简单条件查询，用户往往需要汇聚或统计某些数据集合，为此 DCACHE 还支持用户自定义函数，使得用户可以自己来设定数据聚合过程的各个步骤和动作。

灵活的数据模型和数据类型的表达方式，使得 DCACHE 可以轻松实现结构化数据与半结构化数据的存储与查询。

3.2 高可靠、在线动态扩展

DCACHE 使用一种一致性哈希算法来对数据进行水平分片，对每个名字空间单独进行数据切片，如图 3 所示。要求每一条记录都要有一个全局唯一的主键，主键可以是任意值。使用 RIPEMD160 哈希算法，计算得到一个 160 比特的哈希主键，取其开头的 12 比特形成一个整数值，其取值范围是 0～4095，即得到一个数据分区的 ID。这是一个纯算法的计算过程，对于一个给定的主键值，其计算结果将保持不变，因此任何一条记录所归属的数据分区的 ID 永远不变。

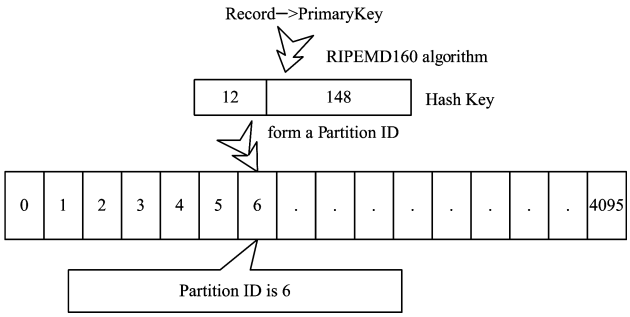

图 3　数据分区

Fig. 3　Data partition

对于确定每个数据分区归属哪个数据节点来管理的问题,DCACHE 使用的分配策略非常简单,即将所有节点均分,如果有余数,个别节点会多分 1 个分区。数据分区的分配过程完全自动进行,无需人工介入,全部由各个数据节点之间相互协商,分区随机归属于节点。

上述过程暂未考虑到多副本情形。在配置多副本的情况下,每个数据分区都会被分配给一个唯一的节点,该节点成为该数据分区的 Master,负责它的读写操作,其他节点则存储其副本。主数据分区的分配过程即为上述均分过程,这样对于一个含 n 个节点集群的名字空间,每个节点大约均分到$1/n$的主数据分区。一个节点管理的所有主数据分区都需要在其他节点上创建副本。以两副本配置为例,这些主数据分区被均匀地分配到其他 $n-1$ 个节点上去创建一个副本,最后,每个节点都存储了大约 $1/n$ 的副本数据分区。

为了增加数据的可靠性,节点之间的数据复制一般被配置成同步方式,主节点接收到客户端的写请求时,会把这个写操作传播到其他副本节点上,待其他副本节点返回成功且本地也写入成功时,才给客户端返回成功。这样,节点间的数据便达到即时一致性状态,单节点故障时不会丢失任何数据。

当集群增加或删除一个节点时,数据分布需要再均衡,这个过程也是完全自动化的。当增加一个节点时,会从其他现有节点上分割出一部分数据分区给新增节点,均衡完毕后所有节点管理的数据分区大约是均等的。当删除一个节点时,该节点分管的所有主数据分区将被其他节点均匀地接管。从上述副本数据分区分布算法可知,这个过程几乎不影响系统目前的运行。由于该节点分管的所有主数据分区的副本已均匀地分布在其他节点上,只需要让这些副本分区转变成主数据分区即可,然后再重新构造它们的新副本分区。

如上所述,DCACHE 的数据分布过程完全智能化,无需人工干预,当出现单点故障时,系统会启动智能再均衡过程,直到数据自愈;而当故障节点恢复之后,集群又会自动把它重新纳入集群成员关系中,再次启动数据再均衡过程。整个过程无需人工介入,也几乎不影响在线业务的访问,因此降低了人工维护的开销,并提高了系统的可用性。

3.3　高性能、低成本极速存储

高并发与高吞吐是 DCACHE 的设计目标之一。单个节点每秒需要处理和存储几万至几十万条记录,面对如此高并发量和数据存储速度的要求,除了硬件层面的支持外,更主要的是需要软件层面在存储机制上进行巧妙设计。在硬件存储设备上,DCACHE 推荐优先使用 Flash 存储介质,如 SSD,其次再分别考虑 SAS 和 SATA 等传统机械硬盘。不过,后者的随机 IOPS 能力明显很低,会大大降低系统的并发吞吐量并增加事务处理的时延。

为了消除硬件差异带来的性能落差,需要在软件机制上进行优化,其中最主要的手段就是缓存。众所周知,传统机械硬盘的随机 IOPS 能力很低,这是由机械磁头的特性决定的,但其顺序读写的吞吐量却相当可观,因为磁头几乎不用换道或者换道时间很短。如何利用这一特性,成为存储机制设计的关键。DCACHE 首先把记录缓存到 1 MB 大小的内存块中,写满之后以追加方式(Append)将其刷写到 DCACHE 日志结构文件系统(Log Structured File System)中,这巧妙地把随机写操作转换成了顺序写操作,从而获得可观的性能。

通过缓存机制,DCACHE 在一定程度上缩小了 Flash 设备与机械硬盘在性能方面的巨大差异,使得 DCACHE 具有更好的部署通用性。然而,根据摩尔定律,硬件存储设备的单位成本快速下降,这使得曾经非常昂贵的 Flash 存储设备变成大众普及品,其价格虽略高于机械硬盘,但却远远低于内存价格。尽管 Flash 存储设备与内存条相比堪称价格低廉,但是其随机读写速度却比较接近于内存存取速度,其性价比极高。因此,DCACHE 默认推荐使用 Flash 存储设备,如 SSD,并专门为它做了存储机制上的优化,使数据存取直接绕过底层文件系统,以充分利用底层 SSD 读写模式的优势,即 DCACHE 以自己独有的数据组织方式来管理 SSD 裸盘设备。每个命名空间可以配置多块 SSD 裸盘,这些裸盘在数据存储过程中可以并行 I/O,并发吞吐量极高。可见,在不增加内存设备的情况下,可以通过添加 SSD 设备将整个分布式缓存系统的数据存储规模大幅提高至 TB 级,而数据读写性能与纯内存存储模式大致相当。

下面介绍基于 Flash 设备的极速存储技术。

DCACHE 日志结构文件系统将整个磁盘空间划分成规整的大小为 1MB 的存储块,除了第一个块用于记录文件系统的全局控制信息外,其他都是数据块。日志结构文件系统可以是常规文件系统下的一个普通文件,也可以是配置在一起的一组裸盘,默认是后者。日志结构文件系统是一个循环的空间,当写到最后一个块时,DCACHE 会调头从第一个块开始写入。

DCACHE 设置一个全局数据块描述符表来描述日志结构文件系统中的所有数据块,每个描述符项对应一个数据块,其中记录了该数据块是否已经加载到内存及其内存地址等信息,并开启了两个数据块缓冲区,一个是写缓冲区(称之为 Write Block List),另一个是读缓冲区(称之为 Read Block List),如图 4 所示。

写缓冲区用于写入新数据。事实上,除了 INSERT 之外,对于 UPDATE 和 DELETE,DCACHE 也都将其转换成某种形式的写入操作。INSERT 直接追加新记录到写缓冲区的当前 Block 缓存中即可;而 UPDATE 较为复杂,其采用 Copy-On-Write 技术,先从旧 Block 读取旧记录值,然后再把

更新后的值写入到当前 Block 缓存中，旧 Block 上的记录不会被立即删除，它们等待后台维护例程定期扫描回收。DELETE 可转换成一个特殊的 UPDATE 操作，但无须读取旧记录值，只记录主键和删除标记等信息。写缓冲区是一个 FIFO 队列，DCACHE 总是尽可能及时地把每一个 Block 缓存块写入到磁盘中。DCACHE 将缓存块的数据刷写到磁盘后，就会将缓存块从写缓冲区队列转移到读缓存区队列中。

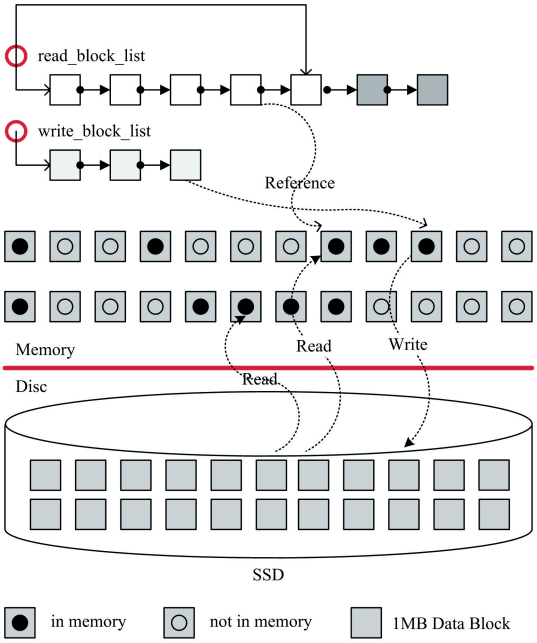

图 4 数据存储组织

Fig. 4 Data storage and organization

当接收到读请求操作时，DCACHE 首先查看全局数据块描述符表的对应表项，判断该 Block 数据是否已经加载到内存中。若已被加载到内存中，则直接读取出相关记录并返回；否则需要先从磁盘上加载该 Block 数据到内存中，将之添加到读缓冲区中，然后再读取相关记录。读缓冲区主要用于存放热点数据，被设计成一个特殊的 LRU 队列，当队列长度超过配置的阈值时释放最老的内存块。为了避免表扫描冲刷掉热点数据，在读缓冲区中设置两个头指针，第一个指针指向真正的热点数据的子队列头部，第二个指针则指向上述子队列的尾部。当一个数据块第一次被加载到内存时，它不会立刻插入到第一个指针所指的子队列头部，而是插入到第二个指针之后的位置，这样第二个指针实际上就是另一个子队列的头指针，该子队列用于存放新数据块和老数据块。只有当数据块第二次被访问时，该数据块才从第二子队列移动到第一子队列的头部。DCACHE 通过这种方式避免了热点数据的抖动。

3.4 失效记录的智能回收

所有的记录都有过期时间(TTL)，它们将在过期后的某个时刻被后台线程自动回收。还有一些记录是被显式删除的，它们也将被回收。将过期记录和被删除记录统称为失效记录。为了使回收过程尽可能不影响正常的服务请求，DCACHE 使用了一种智能触发策略。

每个数据块的大小是 1MB，它可以存储 1～n 条记录。同一数据块内的记录不一定会同时过期或者同时被删除，因此需要一种机制来实时反馈块内空间的占用情况。3.3 节提到的全局数据块描述符表就是用来登记这个信息的数据结构的，该表的一个表项对应一个数据块的描述符，描述符中有一个被称作 used_count 的字段，专门用来登记该数据块内部有效记录占用的字节空间大小。当一条记录过期或者被删除时，将该记录所在数据块的描述符 used_count 减去该记录占用的空间，当 used_count 低于某个配置的阈值时，启动智能碎片整理流程。如图 5 所示，智能碎片整理器读取该数据块的所有有效记录，并把它们重新写入到新数据块中，就如同插入新记录一样(区别是它们的 TTL 保存不变)，因此它们可能和其他刚插入或更改的记录混合存储在一起。此时，原有数据块 used_count 变成 0，整个数据块被回收。

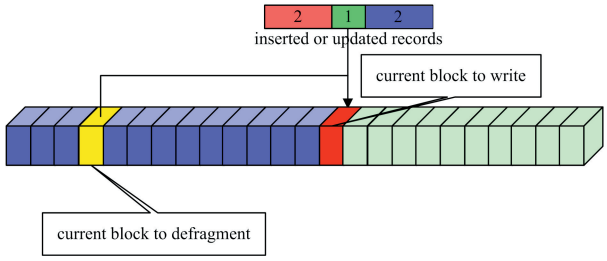

图 5 碎片整理

Fig. 5 Defragmentation

DCACHE 日志结构文件系统是循环写入的，所有数据块按照块号顺序组成一个环，Begin(图 5 中的 current block to defragment)表示当前日志结构文件系统最老的有效块号，End(图 5 中的 current block to write)表示下一个可写入的块号。当 End 追上 Begin 时，表示磁盘空间已满，无法再写入任何数据，此时系统进入只读模式，只能进行读操作，所有的写操作被禁用。

为了避免发生磁盘空间满的现象，DCACHE 在磁盘水位达到配置的阈值时启动强制碎片整理流程，比如阈值设置为 50%，则智能碎片整理器在每个扫描周期整理 n 个数据块，把这些数据块内的有效记录搬运到 End 所指的新数据块内，此时系统的吞吐量会下降，以避免新记录(包括 Insert 和 Update)过快增长而导致磁盘空间爆满。如果新记录增长的速度仍然快于碎片整理的速度，则智能碎片整理器根据磁盘水位每递增 1%就在每个周期增加扫描 10 个数据块，以进一步压制系统吞吐量。

综上所述，智能碎片整理器根据系统资源的占用情况动态调节失效记录空间回收的速度，以保证系统平稳运行。

4 DCACHE 在 RCS 业务中的应用

融合通信(RCS)业务融合了语音、视频、呈现技术、社区网络等多种通信方式及功能，为用户提供丰富的通信体验，与微信类似但功能更多。以 DCACHE 在某运营商 RCS 业务系统的应用为例，该系统支持 1 亿注册用户，其中手机端日活跃用户 1600 万以上，PC 端日活跃用户 400 万以上，起呼消息达到 2.6 万 tps 以上，时延要求在 1 s 以内，是典型的海量数据高并发低时延的应用场景。

在 RCS 老架构中，用户注册信息、消息体本身、控制信息等业务中间过程数据被存放在 DCACHE 第一代的 KV 存储集群中，共有 25 台存储服务器（256G DDR3 内存/8 * 900G SAS 盘）提供服务，单节点性能为 3 万 tps。该配置使用 3 副本机制来实现高可靠性，因此实际有效存储规模不超过 60TB（即 8×900G×25÷3）。用户消息队列等 List 数据则被存放在 Redis 集群中，由 4 台存储服务器提供 NoSQL 服务，单节点性能在 10 万 tps 左右。Redis 由于受到数据必须全部驻留在内存的实现机制的限制，无法使用高性能 SSD 盘降低成本。

如图 6 所示，采用新一代 DCACHE 系统后，DCACHE 既被当作一个分布式高速缓存来存储中间过程数据，又可作为 NoSQL 数据库来存储重要业务数据和用户资料；不需要再维护 DCACHE 第一代和 Redis 两个系统，使得系统的运维和复杂度得到简化；同时，系统增加了 SSD 盘配置，充分利用了 DCACHE 在内存和高性能 SSD 上交换数据的功能。在相同的话务模型下，机器数量减少了 1/3，而且随着业务量的递增，在一定规模下只须扩充 SSD 盘即可提升系统性能，在显著降低配置成本的同时，提升了系统的扩展能力。

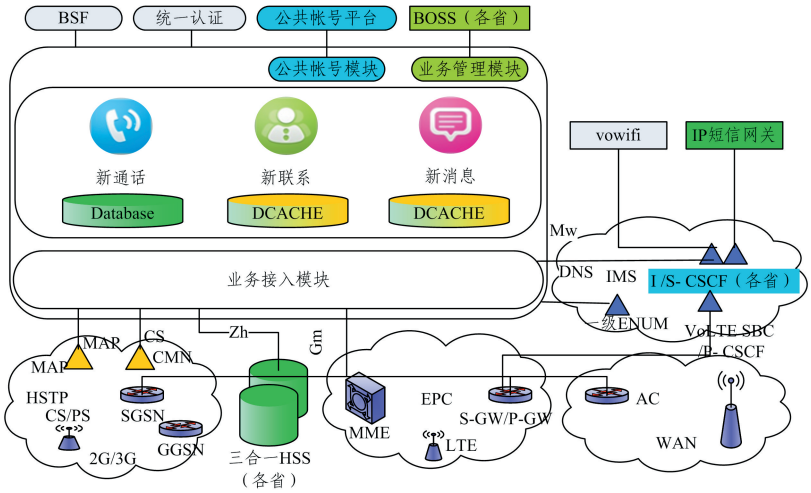

图 6　DCACHE 在 RCS 中的应用

Fig. 6　Application of DCACHE in RCS

基于 RCS 的业务模型，本文对 DCACHE 和 Redis 单台物理设备的处理能力进行对比测试。模拟双副本模式的测试结果如图 7 所示。

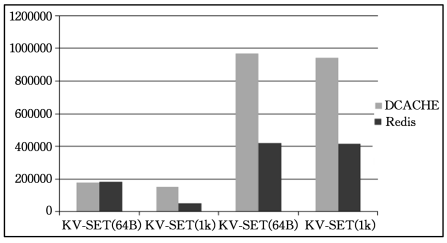

图 7　DCACHE 与 Redis 的性能对比测试

Fig. 7　Comparison test between DCACHE and Redis

从图 7 可知，对于 SET 操作（写操作），当 Value 长度较小（如 64Bytes）时，二者的性能相当；随着 Value 长度的增加（比如 1k Bytes）时，Redis 的性能下降较快，而 DCACHE 的性能比较恒定。对于 GET 操作（读操作），DCACHE 的性能领先于 Redis 一倍以上，其主要原因是，DCACHE 集群中每个节点都是完全对等的，副本互为备份，即每个节点既存储主副本数据分片，又存储备份副本数据分片，因此能够同时对外提供服务；而 Redis 只支持主备节点互为备份的模式，只有主节点才能对外提供服务。

借助于新一代 DCACHE 优良的特性，轻松打造了一个分布式、高可用性、高可伸缩性和高可靠性的大型电信服务系统，助力运营商提升 ARPU 值及 MOU。

结束语　分布式缓存技术的发展经历了迂回的、螺旋式的过程，从最初的刻意追求高性能、低时延的特性，到兼顾多样化的业务需求，支持多种数据类型和检索方式等，分布式缓存系统在技术架构上不断调整和进化。本文介绍了一种分布式缓存系统 DCACHE 的架构及其关键技术，并通过实际应用证明该系统具有强数据表达能力、高性能、高可靠性、低成本、易扩展等优势，解决了大规模数据应用面临的数据可靠性、成本及线性扩展、多样数据类型支持等难题。未来随着计算框架的发展以及新型存储介质的出现，分布式缓存技术将继续发展，为大数据时代丰富多彩的应用系统开发提供支撑和保障。

参 考 文 献

[1]　Wikipedia. Cloud computing[OL]. https://en. wikipedia. org/wiki/ Cloud_computing.

[2]　ZHU J S, WANG M Z, YANG L P, et al. Architecture optimization and evolution of 12306 Internet Ticketing and Reservation System[J]. Railway Computer Application, 2015, 24(11): 1-4. (in Chinese)
朱建生，王明哲，杨立鹏，等. 12306 互联网售票系统的架构优化及演进[J]. 铁路计算机应用，2015，24(11)：1-4.

[3]　GUALTIERI M, RYMER J R. The forrester wave: Elastic caching platforms[OL]. ftp://ftp. software. ibm. com/software/solutions/ soa/pdfs/wave_elastic_caching_platforms_q2_2010. pdf.

[4]　Wikipedia. NoSQL[OL]. https://en. wikipedia. org/wiki/NoSQL.

[5] QIN X L,ZHANG W B,WEI J,et al. Progress and challenges of distributed caching techniques in cloud computing[J]. Journal of Software,2013,24(1):50-66. (in Chinese)
秦秀磊,张文博,魏峻,等.云计算环境下分布式缓存技术的现状与挑战[J].软件学报,2013,24(1):50-66.

[6] EARLS A. Distributed data grids:Foundation for future cloud computing? [OL]. http://searchsoa. techtarget. com/news/1518647/Data-Grids-Foundation-for-future-cloud-computing.

[7] TU Y F. Cloud computing distributed cache with application and practice[J]. Telecom World,2012(10):69-71. (in Chinese)
屠要峰.云计算分布式缓存及其应用实践[J].通讯世界,2012(10):69-71.

[8] STONEBRAKER M. The case for shared nothing[J]. Database Engineering,1986,9(1):4-9.

[9] JAYASWAL K. Administering Data Centers:Servers,Storage,and Voice Over IP[M]. Chicago:John Wiley & Sons,2005.

(上接第 146 页)

出了改进,在选择假位置时既考虑到了历史查询概率,也使得选择的假位置尽可能分散,从而提高了用户的隐私保护程度。但是,这种方法存在时间开销大的问题,特别是在寻找假位置的过程中,会使得服务质量降低。未来将就降低该算法的时间开销和平衡隐私保护程度与用户服务质量问题进行进一步的深入研究。

参 考 文 献

[1] KATO R,IWATA M,HARA T,et al. A dummy-based anonymization method based on user trajectory with pauses[C]// International Conference on Advances in Geographic Information Systems. 2012:249-258.

[2] CHOW C Y,MOKBEL M F,LIU X. A peer-to-peer spatial cloaking algorithm for anonymous location-based service[C]// Proceedings of the 14th annual ACM international symposium on Advances in geographic information systems. ACM,2006:171-178.

[3] YIU M L,JENSEN C S,HUANG X,et al. Spacetwist:Managing the trade-offs among location privacy,query performance,and query accuracy in mobile services[C]// IEEE 24th International Conference on Data Engineering. IEEE,2008:366-375.

[4] HUA B,ZHONG C. The Privacy Preserving Technology in Data Mining[J]. Microelectronics & Computer,2009,26(8):38-41. (in Chinese)
华蓓,钟诚.数据挖掘中的隐私保护技术进展分析[J].微电子学与计算机,2009,26(8):38-41.

[5] PAN X,XIAO Z,MENG X F. Survey of location privacy-preserving[J]. Journal of Computer Science & Frontiers,2007,1(3):268-281. (in Chinese)
潘晓,肖珍,孟小峰.位置隐私研究综述[J].计算机科学与探索,2007,1(3):268-281.

[6] HUO Z,MENG X F. A Survey of Trajectory Privacy-preserving Techniques[J]. Chinese Journal of Computers,2011,34(10):1820-1830. (in Chinese)
霍峥,孟小峰.轨迹隐私保护技术研究[J].计算机学报,2011,34(10):1820-1830.

[7] BERESFORD A R,STAJANO F. Location Privacy in Pervasive Computing[J]. Pervasive Computing IEEE,2003,2(1):46-55.

[8] JIANG T,WANG H J,HU Y C. Preserving location privacy in wireless lans[C]// International Conference on Mobile Systems,Applications,and Services. 2007:246-257.

[9] GRUTESER M,GRUNWALD D. Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking[C]// International Conference on Mobile Systems,Applications,and Services. DBLP,2003:31-42.

[10] KIDO H,YANAGISAWA Y,SATOH T. An anonymous communication technique using dummies for location-based services [C]// International Conference on Pervasive Services(ICPS'05). IEEE,2005:88-97.

[11] BUĞRA G,LIU L. Location Privacy in Mobile Systems:A Personalized Anonymization Model[C]// IEEE International Conference on Distributed Computing Systems(ICDCS 2005). IEEE,2005:620-629.

[12] MA C Y,YAU D K,YIP N K,et al. Privacy vulnerability of published anonymous mobility traces[C]// Sixteenth International Conference on Mobilie Computing and Networking. ACM,2010:185-196.

[13] LIU X,LIU K,GUO L,et al. A game-theoretic approach for achieving k-anonymity in Location Based Services[C]// INFOCOM,2013 Proceedings IEEE. IEEE,2013:2985-2993.

[14] LU H,JENSEN C S,MAN L Y. PAD:privacy-area aware,dummy-based location privacy in mobile services[C]// ACM International Workshop on Data Engineering for Wireless and Mobile Access. DBLP,2008:16-23.

[15] WU Y N,ZHAO Z M,SUN C L. A anonymous Algorithm in Location Privacy Protection Based on Divide Sub Clocking Region[J]. Information Security and Technology,2014,5(10):33-37. (in Chinese)
武艳娜,赵泽茂,孙传林.划分子匿名区域的 k-匿名位置隐私保护方法[J].信息安全与技术,2014,5(10):33-37.

[16] ASHOURI-TALOUKI M,BARAANI-DASTJERDI A,SELÇUK A A. The Cloaked-Centroid protocol:location privacy protection for a group of users of location-based services[J]. Knowledge and Information Systems,2015,45(3):1-27.

[17] MOKBEL M F,CHOW C Y,AREF W G. The new Casper:query processing for location services without compromising privacy [C]// International Conference on Very Large Data Bases. VLDB Endowment,2006:763-774.

[18] NIU B,LI Q,ZHU X,et al. Achieving k-anonymity in privacy-aware location-based services[C]// INFOCOM,2014 Proceedings IEEE. IEEE,2014:754-762.