基于 MongoDB 的分布式缓存①

王 胜[1], 杨 超[1], 崔 蔚[1], 黄高攀[2], 张明明[2]

[1](国网信息通信产业集团有限公司, 北京 100029)
[2](国网江苏省电力公司信息通信分公司, 南京 210029)

摘 要: 电力信息化的发展对传统电力信息系统的数据处理、并发请求及响应能力提出诸多挑战. 针对电力信息系统数据处理的特点, 提出一种基于 MongoDB 数据库的分布式缓存, 并对该分布式缓存的运行机制、服务端架构和客户端功能模块的设计进行了分析与阐述. 基于 MongoDB 的分布式缓存能够有效地降低电力信息系统数据库层的访问负载量, 提高系统的整体性能, 它采用分布式文件存储缓存数据, 支持数据冗余备份和故障恢复功能, 具有较高的可靠性和扩展性. 基于 MongoDB 的分布式缓存已成功应用到电力某企业的项目管理系统中.

关键词: 数据缓存; MongoDB; 分布式缓存; 电力信息系统; NoSQL

Distributed Cache Based on MongoDB

WANG Sheng[1], YANG Chao[1], CUI Wei[1], HUANG Gao-Pan[2], ZHANG Ming-Ming[2]

[1](State Grid Information Communication Industry Group Co. Ltd., Beijing 100029, China)
[2](State Grid Jiangsu Electric Power Company Information &Telecommunication branch, Nanjing 210029, China)

Abstract: With the development of State Grid's information technology, new challenges are presented to the data processing, concurrent requests and responsiveness of conventional power information systems. According to the characteristics of power information system's data processing, a distributed cache based on MongoDB is proposed in this paper. The operational mechanism, server-side architecture and the design of client's function modules in distributed cache are analyzed and discussed. The distributed cache based on MongoDB can effectively reduce the payload on access to the database layer of power information system, and then improve the performance of power system as a whole. It adopts the distributed files to store the cache data with good reliability and scalability to support the function of data redundancy backup and failure recovery. Now the distributed cache based on MongoDB has been successfully applied to the Project Management System of one power corporation.

Key words: data cache; MongoDB; distributed cache; power information system; NoSQL

随着云计算技术的发展和电力企业信息化程度的不断提高, 电力信息系统的规模越来越庞大, 积累的数据越来越多, 系统面对海量数据的并发处理能力越来越重要, 在信息系统架构中, 数据库层的访问速度逐渐成为一大瓶颈. 数据缓存是解决这一问题的关键技术, 它能够有效地降低数据库的访问负载, 提高系统性能. 但面对着海量的数据, 简单的数据缓存已无法满足要求, 这就需要使用分布式缓存技术, 让服务器更快地响应用户的需求.

分布式缓存技术伴随着互联网、云计算的发展浪潮出现了不少优秀的产品, 如 Memcached、JBoss Cache、OSCache、Ehcache、Redis 等, 覆盖了数据存储方面的各个方面, 包含普通关系数据库、非关系数据库, 内存数据库, 内存 cache 等多种实现方案. 但这些产品在数据分布式存储、冗余备份及故障恢复方面存在有一定的不足[1]. Memcached 不支持冗余备份, 某一节点出现故障, 则该节点上的数据将全部丢失, 因此 Memcached 存在单点失效的问题. JBoss Cache 支持两种冗余策略:

① 基金项目:国家电网公司科技项目(SGJSXT00YWJS1400072)
收稿时间:2015-07-27;收到修改稿时间:2015-10-22

全局复制和 Buddy 复制, 全局复制将数据复制给集群中所有节点, 这样能够确保数据完整性, 但它限制了系统的伸缩性; Buddy 复制则挑选特定节点担当备份数据节点, 在备份节点失效时, 无法启动新的节点来替代. OSCache 提供的集群功能比较局限, 无法让缓存中数据在各个节点间复制. Ehcache 是一个 Java 语言开发的缓存系统, 没有提供冗余备份和故障恢复功能. Redis 只使用单核, 扩展性有限.

本文基于这一现状研究并设计一种能够支持数据分布式存储、冗余备份及故障恢复的分布式缓存.

1 MongoDB简介

MongoDB 是 NoSQL 数据库的典型代表. 它是一个高性能、可扩展、模式自由、面向文档、基于分布式文件存储的开源数据库, 采用 C++语言编写, 旨在为 Web 应用提供可扩展的高性能数据存储解决方案.

1.1 MongoDB 的主要特性

MongoDB 的特点是高性能、易部署、易使用, 存储数据非常方便, 它的主要特性有: 面向文档存储、支持完全索引、快速就地更新、支持动态查询及查询记录分析、具备数据自动分片功能以支持云级扩展性、具备数据复制与故障切换功能以支撑高可靠性. 除此之外, MongoDB 还具有模式自由、支持 Map/Reduce、提供分布式文件系统 GridFS、支持多种编程语言、使用高效的二进制数据存储、文件存储格式为 BSON[2].

1.2 MongoDB 的适用场景

MongoDB 在键/值存储与 RDBMS 系统间架起了一座桥梁, 集两者优势于一身. MongoDB 适合用于处理网站数据: 它支持数据的实时插入、更新与查询, 并具备实时数据存储所需的可靠复制及高度伸缩性; MongoDB 适合用于数据缓存: 它具有极高的数据处理性能, 适合作为信息基础设施的缓存层, 由于数据持久化存储在分布式文件中, 在系统重启之后可以避免底层数据源的过载访问; MongoDB 适合用于高伸缩性的场景, 支持由数十台、数百台甚至更多的服务器来架构数据存储服务, MongoDB 内置 MapReduce[3]引擎, 可以将大批量数据分解, 然后通过大量机器进行并行计算处理.

MongoDB 的使用也存在一些限制, 它并不适合用于高度事务性的场景, 例如电力调度控制、电费在线缴纳等. 传统的关系型数据库更适用于需要大量原子性复杂事务的应用系统. MongoDB 不适用于传统的商业智能应用, 针对特定问题的 BI 数据库会产生高度优化的查询方式, 对于此类应用, 数据仓库可能是更合适的选择.

2 基于MongoDB的分布式缓存

当文件系统遭遇大规模数据访问时, IO 读取往往成为性能瓶颈, 导致过高的数据延迟. 分布式缓存通过一定的规则及策略, 采用键/值方式将数据分散存储于不同的物理机器上, 它是云存储技术的一种, 满足云计算技术的高性能、大容量、高可靠及可扩展的特点, 可以通过动态增加或者减少节点应对变化的数据负载, 提供可预测的性能. 分布式缓存的目标是降低数据源访问次数, 减免数据源访问瓶颈的发生, 进而提高应用系统的性能, 同时用以支撑云环境下的海量数据缓存.

传统基于内存数据库的缓存组件通常只支持简单的键值存储, 对缓存文件的类型、文件之间的关系的存储和查询不够方便. MongoDB 作为文档数据库, 其自动分片机制对大规模分布式存储的支持较好, 尤其适合作为大规模非结构化文件的存储缓存.

本文研究的分布式缓存采用 MongoDB 数据库作为底层存储, 并结合数据缓存的特性扩展 MongoDB 驱动设计实现一套分布式缓存套件.

2.1 分布式缓存运行机制

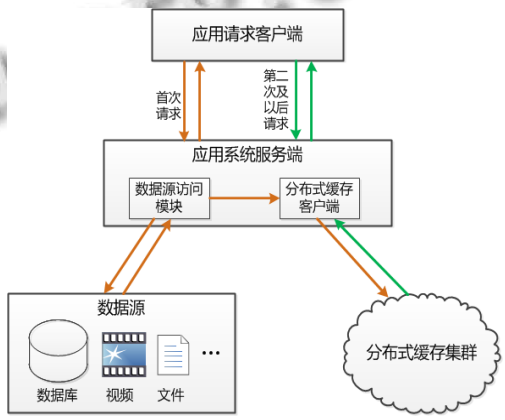

图 1　分布式缓存运行机制

首次请求应用数据时, 应用系统调用数据源访问模块获取数据, 同时通过缓存客户端将数据包装并存储至分布式缓存集群; 之后再请求同一应用数据时, 应用系统直接通过缓存客户端快速地查找分布式缓存

集群中对应的数据并返回, 而不必再请求数据源, 从而缓解数据源访问压力, 提高数据访问性能.

2.2 分布式缓存服务端架构

本文研究的分布式缓存服务端采用MongoDB数据库的自动分片机制结合副本集部署[4]方式进行架构, 以提高扩展性与可靠性. 分布式缓存服务端架构见图 2.

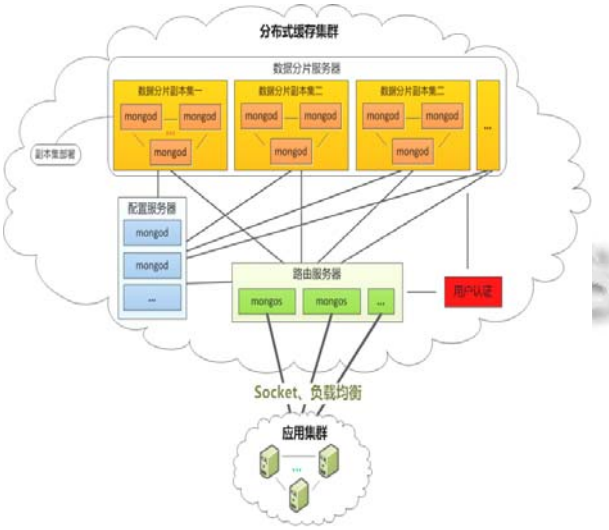

图 2 分布式缓存服务端架构

2.2.1 缓存数据分片机制

一个数据分片副本集由多个数据分片服务器构成, 一个数据分片服务器对应一个 mongod 进程. mongod 是 MongoDB 的核心进程, 它对应 MongoDB 的一个具体实例, 用于缓存数据的存储与处理, 完成缓存数据的持久化[5]. 一个 mongod 实例可以承载多个数据库, 每个数据库之间是完全独立的, 一个数据库下包含有多个数据集合, 数据集合下存放着具体的缓存数据. 在自动分片机制下, 缓存数据以 chunk 为单位进行存储[6]. 可通过动态删除和增加服务器节点来提升缓存的数目和效率.

2.2.2 数据分片策略配置

配置服务器用于存储分布式缓存集群中的元数据, 包括每个数据分片服务器的信息及每个数据块(chunk)的信息. 集群中每一个配置服务器节点都保存了分布式缓存上所有的元数据信息[7].

2.2.3 缓存一致性策略

该组件采用路由服务器作为连接应用集群与分布式缓存集群的门户, 它对用户屏蔽了分布式缓存集群的内部细节, 使分布式缓存集群看起来像是一个单一的应用, 使得系统无需考虑缓存一致性问题.

2.2.4 用户认证

用户认证用于确保客户端与服务器端的数据安全访问, 它支持简单的用户/密码认证方式, 避免系统外用户非法获取缓存服务器数据, 从而提高缓存服务器的安全性.

2.3 分布式缓存客户端设计

本文研究的分布式缓存客户端采用模块化思想和面向接口的方式设计, 具有较好的可扩展性, 主要功能是接收并包装应用系统的缓存请求, 将包装后的缓存请求命令(包含应用数据等)发送至分布式缓存集群以处理, 同时对应用数据的生命周期进行统一管理. 分布式缓存客户端由以下三大模块构成: 缓存管理模块、缓存模块、缓存注解模块.

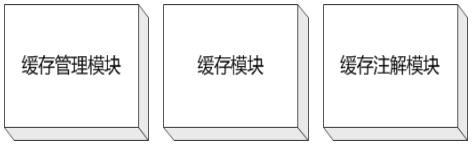

图 3 分布式缓存客户端功能模块

2.3.1 缓存管理模块

缓存管理模块负责初始化客户端配置信息, 建立、管理缓存对象以及协调各模块间的工作. 缓存管理模块抽象一个统一的缓存管理接口 CacheManager, 在该接口中定义配置信息初始化、缓存对象加载、添加、移除等方法. 分布式缓存管理器实现统一缓存管理接口, 封装 MongoDB 数据库的 DB 对象, 采用工厂模式实现一个分布式缓存管理器创建工厂, 该工厂负责管理分布式缓存管理器对象的实例化过程, 具有较高的可扩展性.

2.3.2 缓存模块

缓存模块是整个分布式缓存客户端的核心, 它负责将业务应用数据包装成缓存数据, 并往分布式缓存集群发送缓存数据添加、移除、清空等操作指令, 同时管理缓存数据的生命周期. 缓存模块针对各应用系统关注点的不同抽象出两个接口: 缓存数据存取接口 CacheAccess、缓存运行监视接口 CacheMonitor, 同时扩展这两个接口提供一个统一缓存接口 Cache. 分布式缓存实现统一缓存接口, 封装 MongoDB 数据库的数据集对象, 借助该数据集对象完成缓存数据的管理功能.

分布式缓存数据包装了业务应用数据，同时赋予该应用数据一定的生命期限，其生命周期管理结合了TTI(Time To Idle)与 TTL(Time To Live)两种机制[8]，以满足缓存数据与数据源数据的一致性要求[9]. 缓存数据生命周期管理采用客户端延迟失效模式：当分布式缓存客户端获取到缓存数据后，立即判断该数据是否过期，如果已经过期，则删除缓存数据并发布数据过期事件，如果未过期，则将缓存数据返回给用户. 通过该模式可以避免分布式缓存服务端上失效检测线程的开销，提高整个分布式缓存集群的运行性能.

2.3.3 缓存注解模块

分布式缓存客户端基于 AOP(Aspect Oriented Programming)思想提供一套缓存注解功能[10]. 通过该功能可以使开发人员更专注于业务逻辑处理，当需要缓存功能时，仅需配置一个拦截器并在业务方法上配置缓存注解即可. 缓存注解包括两个：缓存数据加载@CacheFetch、缓存数据刷新@CacheRefresh.

(1)@CacheFetch 标注在获取业务数据的方法上. 当应用系统调用具有此注解的业务方法时，会自动先从分布式缓存中获取数据，如果取到数据就直接返回，如果未取到数据则执行业务方法，把执行结果返回，同时放入缓存.

(2)@CacheRefresh 标注在业务数据更新的方法上. 当应用系统调用具有此注解的业务方法时，会自动根据缓存注解配置信息刷新缓存中相应的数据.

3　应用案例

基于 MongoDB 的分布式缓存方案已在电力某企业的 EIC(Electricity Infrastructure of Common)公共平台中实现，并应用于该企业的项目管理系统(基于 EIC 公共平台开发)当中. EIC 公共平台在系统启动时将运行时相对不易变动的数据库数据放入分布式缓存中，以提高系统响应能力，这些数据包括数据字典、组织机构信息、用户信息、模型配置信息等. 项目管理系统的分布式缓存服务器使用 2 个数据分片节点来缓存数据，每个分片节点对应一个副本集，每个副本集由 3 台数据分片服务器组成，目前该系统运行良好.

项目管理系统的"项目"模型关联较多的配置信息，此处以该模型信息的获取作为验证点，使用 Apache 的 JMeter2.9 软件进行多用户并发访问的性能测试. 分别在不使用分布式缓存和使用分布式缓存两种情况下对系统响应时间进行测试，针对每一种情况对应某一指定用户数进行 10 次重复测试，取其平均时间作为系统响应时间，得到如图 4 所示的验证结果.

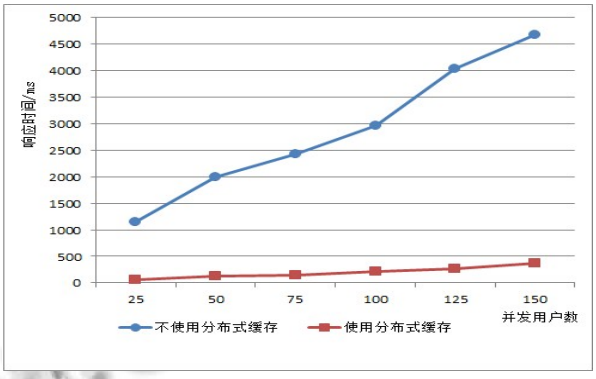

图 4　不同情况下项目管理系统响应时间对比

通过对上述的验证结果分析，可以发现在相同并发用户数条件下，使用分布式缓存比不使用分布式缓存系统响应时间明显减少，而且当并发用户数增加时，使用分布式缓存的系统响应时间曲线增长较为平缓. 这说明基于 MongoDB 的分布式缓存对于项目管理系统的多用户并发访问系统性能的提高有较为显著的作用.

同时，在上述验证测试过程中发现，当缓存数据量较为庞大(超过分片节点间数据量差异阈值)时，缓存服务器能够自动根据指定规则将数据分散存储于两个分片节点中，当分片节点数据量差异达到指定阈值时，缓存服务器能够自动重新均衡数据分布；验证过程中尝试手动停止副本集中的主节点，发现缓存服务器能够自动选择出一台副节点接管主节点工作. 这说明基于 MongoDB 的分布式缓存能够有效地支持数据分布式存储、冗余备份及故障恢复功能.

4　结语

本文研究并设计了一种基于 MongoDB 的分布式缓存，简要地介绍了 MongoDB 数据库，从分布式缓存的运行机制、服务端架构、客户端功能设计等方面进行了阐述. 基于 MongoDB 的分布式缓存具有极高的数据存取速度，支持数据复制与故障切换功能，支持数据自动分片功能，能够支撑海量数据缓存，有效地降低数据库的访问负载，提高系统性能. 本文所提出的分布式缓存可以为电力企业中需要海量数据高性能并发访问的信息化系统的缓存层设计提供参考.

参考文献

1 戚伟强.分布式缓存模式研究及其在金融系统中的应用[硕士学位论文].杭州:浙江大学, 2008.

2 王光磊.MongoDB 数据库的应用研究和方案优化.中国科技信息,2011,(20):93–94,96.

3 Dean J, Ghemawat S. MapReduce: Simplified Data Processing on Large Clusters. Communications of The ACM, 2008,51(1):107–113.

4 红丸.MongoDB 管理与开发精要.北京:机械工业出版社, 2012.

5 李文逍,杨小虎.基于分布式缓存的消息中间件存储模型.计算机工程,2010,36(13):93–95.

6 詹道楠.统一数据服务层框架缓存模块的设计和实现[硕士学位论文].南京:南京大学,2012.

7 李文中,陈道蓄,陆桑璐.分布式缓存系统中一种优化缓存部署的图算法.软件学报 2010,21(7):1524–1535.

8 黄世能,奚建清.分布访问环境中的数据缓存体系研究.计算机工程与科学,2000,22(6):88–91.

9 Gwertzman J, Seltzer M. World wide web cache consistency. USENIX 1996 Annual Technical Conference, 1996. California: Harvard University, 1996: 22–26.

10 Pawlak R, RetaillÉ J, Seinturier L. Foundations of AOP for J2EE Development. New York: A Press, 2005: 53–60.