内存数据库在高速缓存方面的应用*

杨　艳　北京邮电大学网络与交换技术国家重点实验室硕士研究生
李　炜　北京邮电大学网络与交换技术国家重点实验室副教授
王　纯　北京邮电大学网络与交换技术国家重点实验室高级工程师

摘要：随着Internet技术的不断发展，影响网络速度的瓶颈主要集中在访问距离和服务器承载负荷能力方面。扩展服务器或者镜像服务器作为基本解决方案在运营维护方面花费的代价较高，而Cache缓存技术作为一种补充方案，以其简单的设计、高效的存储性能得到了越来越广泛的应用。内存数据库作为缓存的一种补充方案，应用范围也在逐步扩展。文章针对两种内存数据库产品的特征进行了对比介绍，同时分析了他们的应用场景。

关键词：Cache，内存数据库，Memcached，Redis，分布式，复制

Abstract: With the continuous development of Internet technology, the bottleneck of network speed mainly concentrated in access distance and load carrying capacity of the server. Expand server or mirror server as a basic solution costs higher in the aspect of operation and maintenance, while the Cache technology as a supplementary method has been more widely used, because it has simple design and efficient storage performance. Main memory database's applications in cache are also gradually extended. In this paper, two main memory database products are referred, with a comparative introduction, and their scenarios are analyzed.

Key words: Cache, MDB, Memcached, Redis, distributed, replication

随着Internet技术的快速发展，接入网速度不断提高，主干带宽不断扩容，用户规模不断扩大，传统影响网络速度的因素不断减少，目前影响网络速度的主要瓶颈集中在访问距离和服务器承载负荷能力方面。同时，随着网站访问量日渐增大、内容不断丰富和用户期望值不断提高，用户应用需要提供更快的访问速度和承受更大的负荷量，所有这些都依赖于网站服务器的基础设施，扩展服务器或者镜像服务器是一个基本的解决方案，但扩展服务器会增加大量的运行维护工作，同时增加了运营成本。

Cache缓存服务器技术有效地解决了基于磁盘的数据库中CPU和磁盘I/O之间的主要矛盾[1]，并且作为一个解决访问距离和提高源服务器能力瓶颈的有效方法，得到了越来越广泛的应用。由于高速缓存服务器的设计简单，性能高效，可以减少网站服务器的内容传输负荷，提高对用户的响应速度，有效地提高了网站性能和可扩展性。

* 基金项目：国家自然科学基金(No. 61072057,60902051)；国家973计划项目 (No. 2012CB315802)；中央高校基本科研业务费专项资金(BUPT2009RC0505)；国家科技重大专项(No. 2011ZX03002-001-01，移动互联网总体架构研究)。

1 Cache 技术概述

Cache 即高速缓冲存储器，我们常称之为缓存。最早指位于 CPU 与主内存间的一种容量较小但速度很高的存储器。Cache 中保存着 CPU 刚用过或循环使用的一部分数据，当 CPU 再次使用该部分数据时可从 Cache 中直接调用，这样就减少了 CPU 的等待时间，提高了系统的效率。传统的 Cache 实际是一种缓存介质，而按照缓存和应用的耦合程度可以将 Cache 划分为本地缓存（Local Cache）和远端缓存（Remote Cache）。

将 Cache 安装在 Web 服务器前端或数据库服务器的前端，作为“前置机”直接接受用户请求，且能与 Web 服务器保持自动同步更新，免维护的这种缓存方式往往称为 Local Cache，即 Local Cache 指包含在应用之中的缓存组件。Remote Cache 是对应于 Local Cache 来说的，指与应用解耦，在应用之外的缓存组件。Local Cache 由于与应用紧密耦合，多个应用程序无法共享缓存，容易造成内存的极度浪费，因此目前应用多采用 Remote Cache 来缓存访问数据。按照缓存方式的不同 Remote Cache 又可分为 Center Cache 和 Cluster Cache。Center Cache 指 Cache 作为缓存中心，应用作为客户端的一种缓存方式。Cluster Cache 指 Cache 分布在不同终端，这些 Cache 互相进行信息共享与同步的缓存方式。

利用 Cache 缓存数据应该认识到因为服务器上数据不断更新，所以必须考虑缓存与服务器之间的同步，保持缓存与服务器之间的数据一致[2]，不同的缓存解决方案具有不同的同步方式，本文不再赘述。

实际应用中，作为需要保证高度可用性的电信级业务，可以采取两级 Cache 结构，业务在两个执行平台中执行，有自己的 Cache 系统，同时在系统的局域网中，引入 Cache 前置机，从而构成一个两层的 Cache 系统[3]。

Cache 主要可应用在：

·降低数据库负载，对数据库数据进行缓存；

·提高 Web 页面生成速度，对于重复使用的 Web 页面缓存部分或者全部；

·提高计算速度，对于复杂计算结果缓存。

内存数据库（MDB：Main Memory Database）产品的出现给目前的诸多应用提供了有效方便的缓存解决方案。MDB 通过将数据完全加载到内存，在内存中实现对数据的管理。MDB 有效地解决了基于磁盘的数据库中 CPU 和磁盘 I／O 之间的主要矛盾，内存中数据读写的速度比磁盘要高出几个数量级，将数据保存在内存中相比从磁盘上访问数据能够极大地提高应用的性能[1]。内存数据库主要应用于一些实时响应度要求比较高和性能要求比较高的系统，如电信的业务支撑系统（BSS：Business Support System），计费系统（BOSS：Business & Operation Support System）等等。同时由于内存数据库受到物理内存的限制，在常见的缓存解决方案中往往是磁盘数据库和内存数据库并存，磁盘数据库用于存储可异步处理的数据，内存数据库用于存储必须快速处理、及时响应的数据。

常见的开源内存数据库包括 Memcached，Redis 等。

2 Memcached

2.1 概述

Memcached 是一个高性能的分布式的内存对象缓存系统[4]，通过在内存里维护一个统一的巨大的 hash 表，能够存储各种格式的数据[5]。一般使用目的是通过缓存数据库查询结果，减少数据库访问次数，以提高动态 Web 等应用速度和提高扩展性。

通常的 Web 应用会将数据保存到关系数据库管理系统（RDBMS：relational database management system）中，应用服务器从中读取数据并在浏览器中显示。当数据量增大、访问集中时，会出现 RDBMS 的负担加重、数据库响应恶化、网站显示延迟等重大影响。图 1 为 Memcached 运行图。

如图 1 所示，增加类似 Memcached 的处理方案可以提高数据库的访问速度，有点类似内存数据库，

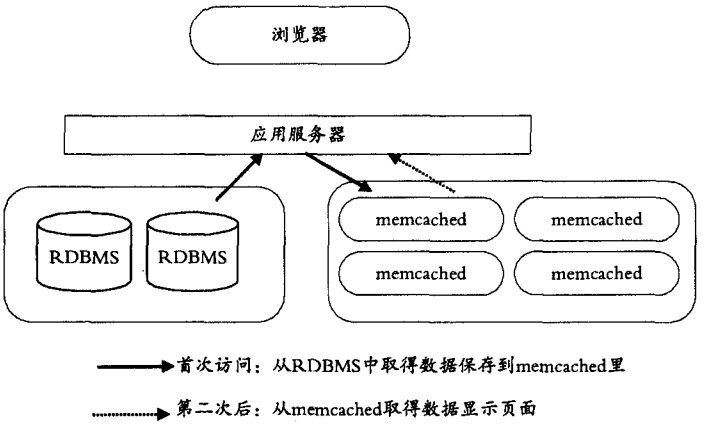

图 1 Memcached 运行图

但其方案简单，成本很低，并支持分布式部署。

2.2 基本特征

Memcached 作为高速运行的分布式缓存服务器，具有以下的特点：

·协议简单；

·基于 libevent 的事件处理；

·内置内存存储方式；

·不互相通信的分布式 。

2.2.1 C/S 结构协议简单

Memcached 的服务器客户端通信不使用复杂的 XML 等格式，而使用简单、基于文本行的协议。因此，通过 telnet 也能在 Memcached 上保存数据和取得数据：

```
$ telnet localhost 11211
Trying 127.0.0.1...
Connected to localhost.localdomain (127.0.0.1).
Escape character is '^]'.
set foo 0 0 3   （保存命令）
bar             （数据）
STORED          （结果）
get foo         （取得命令）
VALUE foo 0 3   （数据）
bar             （数据）
```

2.2.2 基于 libevent 的事件处理

libevent 是个程序库，它将 Linux 的 epoll、BSD（berkeley software distribution）类操作系统的 kqueue 等事件处理功能封装成统一的接口（跨平台的事件处理接口的封装）。即使对服务器的连接数增加，也能发挥 O(1)的性能。Memcached 使用这个 libevent 库来进行网络并发处理的连接，因此能在 Linux、BSD、Solaris 等操作系统上发挥其高性能。

2.2.3 内置内存存储方式

为提高性能，Memcached 中保存的数据都存储在 Memcached 内置的内存存储空间中。由于数据仅存在于内存中，因此重启 Memcached、重启操作系统会导致全部数据消失。另外，内容容量达到指定值之后，就基于 LRU(least recently used）算法自动删除不使用的缓存。 Memcached 本身是为缓存而设计的服务器，因此并没有过多考虑数据的永久性问题。

数据存储方式：Slab Allocation，按照规定的大小，将分配的内存分割成各种特定长度的块（chunk），以完全解决内存碎片问题。特定大小的 chunk 的组成为 Slab class。因此会存在无法有效利用分配的内存的问题。

数据过期方式包括：

·Lazy Expiration：Memcached 内部不主动监视记录是否过期。而通过 get 操作时查看记录的时间戳，检查是否过期。

·LRU：Memcached 会优先使用已过期记录的空间，但也会发生追加新记录时空间不足的情况，此时 Memcached 使用“最近最少使用”机制 LRU 来分配空间。因此当 Memcached 内存空间不足时，将搜索最近未被使用的记录，将其空间分配给新的记录。

2.2.4 不互相通信的分布式 / 基于客户端的分布式

Memcached 尽管是“分布式”缓存服务器，但服务器端并没有分布式功能。各个 Memcached 不会互相通信以共享信息，如何进行分布式完全取决于客户端（严格地说是取决于分布式算法如何选择 Memcached 服务器）的实现，因此也称为基于客户端

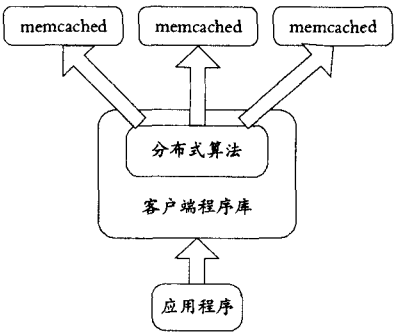

图2 基于客户端的 Memcached 的分布式

的分布式。如图2所示。

应用程序要实现分布式,需要读取操作之前,根据键值 key,采用相同的分布式算法获取服务器 ID(nodeID),从而命中 Memcached 服务器;Memcached 服务器再根据键值搜索到相应的值 Value,对 Value 进行指定的操作后,返回结果。

3 Redis

3.1 概述

Redis 同 Memcached 一样,也是一个 C/S 结构(应用作为客户端,Cache 提供服务)的远程 Cache 实现,Redis 在很多方面与 Memcached 具有相同的特征,唯一的不同在于 Redis 增加了持久化的功能,Redis 把整个数据库系统加载在内存当中进行操作的同时,定期通过异步操作把数据库数据 flush 到硬盘上进行保存,因此可以把 Redis 看做一个可持久化的内存数据库。

3.2 基本特征

由于 Redis 在很多方面与 Memcached 具有相似的特征,为了更好的对比,这里重点讲解 Redis 区别于 Memcached 的特征。

·持久化;

·多数据结构的支持;

·支持主从复制;

·Virtual Memory 功能。

3.2.1 持久化

Redis 虽然是一个内存数据库,但它可以根据数据的更新量和间隔时间定期通过异步操作将数据刷新到硬盘中进行保存(默认保存为 dump.rdb),即使 server 重启,数据仍不会丢失,这是与 Memcached 的一大区别。Redis 配置文件 redis.conf 中有这样的配置 save <seconds> <changes> 意义为每隔 seconds 秒的时间如果至少有 changs 数量的数据变化 (可以理解为写操作发生的次数),则将保存数据库到磁盘。

以上方式是一种不完全可靠的方式,比如在 redis down 或系统断电情况下,写到 redis 中的数据将会丢失,为了提供高可靠的持久化机制,Redis 提供了一种附加档案功能 AOF(append only file)。

AOF 方式也可以称为写日志的方式,Redis 提供了一种名为 append only file 的文件。使用这种方式的前提是需要打开 redis 配置文件中的 appendonly 开关,即只需配置"appendonly yes"。append only file 的工作方式为:每次 Redis 接收到改变数据库的命令时,都将此命令添加到 append only file 的文件中,当重新启动 Redis 时,这些命令将重新执行以恢复 Redis 数据库的状态。当 append log 文件过大时,redis 提供了一种命令 BGREWRITEAOF,可以重写文件(重写文件的目的是合并之前的操作)而不用停止客户端命令的执行。

3.2.2 多数据结构的支持

像大多数的 key-value 数据库一样,key 值为 string 类型,Redis 的不同之处就在于它的 value 值除了 String 之外可以有多种类型,目前 Redis 支持以下几种类型的 value:String,list,set,zset(sorted set),hash(Redis2.0 中新增的数据类型)。针对每种数据类型,Redis 提供了相应的命令对其中的元素进行操作,如 PUSH/POP,ADD/REMOVE,对于集合类型在 server 侧提供并、交、差等集合操作命令,此外还提供了针对链表和集合的排序能力支持。

3.2.3 主从复制(master-slave Replication)

Redis 支持快速简单的主从复制,Redis 主从复制具有以下几个特点:

·支持一个主服务器连接多个从服务器;

·从服务器能够接受来自其他从服务器的连接;

·在主服务器 master 一侧，数据的复制是非阻塞的，即主服务器在接受从服务器同步命令的同时仍可以接受来自客户端的查询操作；

·在从服务器 slave 一侧，数据的复制是阻塞的，即从服务器执行同步过程不能接受来自客户端的查询。

利用 Redis 的主从复制特性，我们可以实现以下功能：

·可以把主服务器 master 改造成一个完全的内存服务器，修改主服务器配置，保证"永远"不会往磁盘写数据，这样所有操作都可以在内存中完成，而持久化由从服务器完成，这样操作主服务器效率会更高(不过这种方式不适合保存重要数据)。

·可以利用主从服务器的方便性来备份，专门做一台从服务器用于备份功能。

3.2.4 Virtual Memory 功能

Redis 的主要缺点是数据库容量受到物理内存的限制，不能用作海量数据的高性能读写，而 Virtual Memory 则是针对这一缺点提出的改进方案。

Redis 的 Virtual Memory 功能是 Redis2.0 新增的一个非常稳定和可靠的功能，目的是为了提高 Redis 的性能，也就是把很少使用的 value 保存到 disk，而 key 保存在内存中。假设有 10 万个 key 在内存中，而只有仅仅 10%左右的 key 经常使用，那么 Redis 可以通过开启 Virtual Memory 尝试将不经常使用的 Value 转换到磁盘上保存。

4 Memcached 与 Redis 应用分析

通过以上 Memcached 与 Redis 的特征分析，可以看到作为同种中心 Cache 工具，Redis 具有比 Memcached 更加完善的功能，对应特征的不同，两者应用场景也有所区别。

4.1 Memcached 应用举例

在下一代网(NGN)的业务运行过程中，需要处理呼叫的业务逻辑，包括鉴权、计费等，因而需要访问大量的数据库资源[6]，把能提供会话状态共享的服务器称为状态服务器。会话状态等信息通常改动较为频繁，在数据库上进行读写操作，一是数据库压力会较大，性能较低；二是这些数据(尤其是临时状态)持久化的意义不大。而 Cache 目的就是要加速对资源的访问和临时存储信息。因此用户可以考虑利用内存数据库来实现状态服务器。利用 Memcached 实现状态服务器组网如图 3 所示。

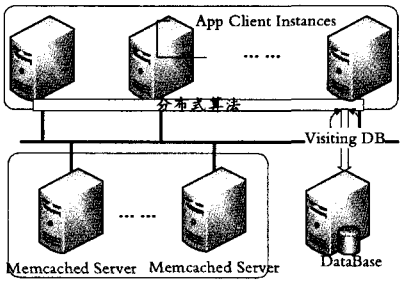

图 3 基于 Memcached 的状态服务器

根据前面的介绍，Memecached 作为缓存服务器端，如何分布式部署取决于客户端。换句话说，运行 Memcached 的节点属于单点实例方式——只有缓存能力。若希望提供集群缓存能力，需要设计人员额外的工作量——通过改造客户端或服务端的方式来满足需求。当然此种应用下将 Memecached 替换为 Redis 同样能满足应用要求。如果应用在缓存数据的同时要求数据做永久存储，在 Redis 无疑是合适的选择。

4.2 Redis 应用举例

Redis 由于增加了诸多新的特性，其应用领域逐渐增加，以东信北邮短信业务进行说明。

如图 4 所示，其中应用服务器(AS:Application Server)指东信北邮应用服务器，短信前置用来路由转发短信消息。短信前置收到上行短信消息将消息送到 Redis 服务器，AS 定时从 Redis 取消息上报给业务。Redis 在此过程中缓存短信消息。当业务下发消息时，将下发的短信消息保存到 Redis 服务器，短信前置也通过定时器定时取下行消息，若下行消息要求回复状态报告，则可能造成状态报告和响应逆序，业务逻辑会出现问题。因此如果状态报告先于响应到达，利用 Redis 缓存报告，等响应到时再按序上报业务，这样可保证业务逻辑的正确性，利用 Redis

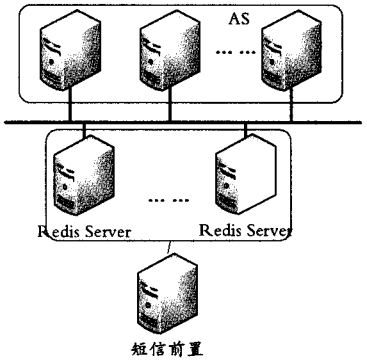

图 4 基于 Redis 的短信业务组网图

的缓存特性在保证业务逻辑正确性方面发挥了重要作用。同时由于 Redis 支持多种数据结构，在此应用中利用两对队列保存上下行短信消息，方便了 AS 和短信前置对消息的存取。

5 结语

为了满足缓存服务器应用的不同需求，同时为了更好地提出缓存解决方案，给出了 Cache 的定义及分类，对两种内存数据库 Memecached 和 Redis 的主要特征进行了分析，同时对他们作为缓存解决方案的应用场景进行了示例说明。Memecached 和 Redis 作为内存数据库的典型代表，可以方便满足高响应速度的缓存需求。随着 Cache 缓存技术的不断发展，这种内存数据库的应用会越来越多，同时我们也应该考虑到目前许多大规模数据还不可能完全置于主存，MemCached 和 Redis 作为内存数据库，都会受到物理内存的限制，这是所有内存数据库共同的限制。但还是可以将大量数据尤其是热点数据置于主存[7]，仍然可以充分发挥内存数据库的优势。MSTT

参考文献

[1] 徐海华. 面向应用的内存数据库研究 [D]. 上海：上海师范大学，2008.
[2] 王文林，廖建新，朱晓民.基于 VoiceXML 的语音平台缓存一致性控制算法[J].电子学报，2007,4(4):1-3.
[3] 郭志伟，武家春，廖建新等.应用于 VoiceXML 系统的缓冲机制的研究[J].信息网络，2005,(11):1-3.
[4] 俞华锋.Memcached 在大型网站中的应用 [J]. 科技信息，2008,(1):1-3.
[5] 宗小中.基于 Memcached 构建 Web 缓存服务器[J].电脑知识与技术，2011,7(5):1-4.
[6] 宋述燕，王锦程，尹建新.NGN 业务平台内存数据库的设计与实[J].现代计算机工程，2008,34(23):1-4.
[7] 王珊，肖艳芹，刘大为，等.内存数据库关键技术研究[J].计算机应用，2007,10(10):1-7.

（上接第 58 页）

2.4 结果

对于主机侧的软件来说，首先要有一个优秀的软件结构。用面向对象的思想，将职责合理分配到各个类上，通过依赖于接口来屏蔽具体的实现，并通过实现特定的接口来提高通信效率。“外设控制器”这个类是与硬件通信的关键，也是图 1 中连接上下层的核心。上下层的通信以“Observer”和“Observable”的方式来实现，使得通信简洁而优雅。再来回顾一下 3.2 中所提出的问题，发现棘手的问题都已经迎刃而解。这就是设计之美。

3 结语

笔者开发过多款外设的主机侧软件，在多年的实践中，对其间反复发生的问题，总结了有价值的设计经验，并编写了此设计模式。总之，一款软件仅仅实现了相关功能还远远不够，软件不仅要有良好的外在特性，更重要的是要拥有良好的内在特性。软件持续升级是不可避免的，软件的可维护性差，无法应对快速的变化，将逐渐被市场淘汰。因此，软件在开发之初，先要确定良好的结构，合理的分配职责，抽取出各部分间的软件接口，用接口隔离各个易于变化的部分，并用接口实现消息的传递，最终使软件具备持久的竞争力。MSTT

参考文献

[1] Erich Gamma,Richard Helm,Ralph Johnson,et al. 设计模式：可复用面向对象软件的基础.机械工业出版社，2005(6).
[2] 姜拓，张剑平.基于 C# 的数据采集系统上位机软件设计与实现 [J].电子测试，2009(9).
[3] 柯艳，李杰，孔祥磊.基于 USB2.0 的多路数据采集系统上位机软件设计[J].测试技术学报,2010(4).
[4] 童洪洁，李宝华. USB 上位机程序开发与设计 [J]. 仪器仪表用户，2005(1).
[5] 朱丽，陈钟荣，张秀再. 基于 VC 的 USB 接口通信程序设计[J].电子工程师. 2008(3).