17 | 高性能缓存架构

2018-06-05 李运华

17 | 高性能缓存架构
朗读人：黄洲君　10′33″ | 4.84M

虽然我们可以通过各种手段来提升存储系统的性能，但在某些复杂的业务场景下，单纯依靠存储系统的性能提升不够的，典型的场景有：

- 需要经过复杂运算后得出的数据，存储系统无能为力

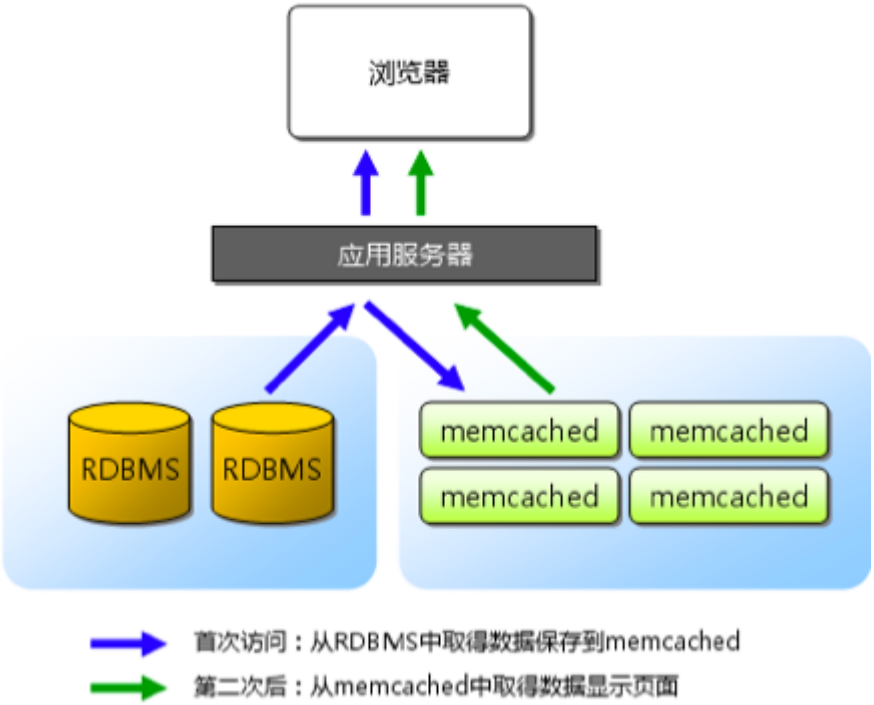
例如，一个论坛需要在首页展示当前有多少用户同时在线，如果使用 MySQL 来存储当前用户状态，则每次获取这个总数都要“count(*)”大量数据，这样的操作无论怎么优化 MySQL，性能都不会太高。如果要实时展示用户同时在线数，则 MySQL 性能无法支撑。

- 读多写少的数据，存储系统有心无力

绝大部分在线业务都是读多写少。例如，微博、淘宝、微信这类互联网业务，读业务占了整体业务量的 90% 以上。以微博为例：一个明星发一条微博，可能几千万人来浏览。如果使用 MySQL 来存储微博，用户写微博只有一条 insert 语句，但每个用户浏览时都要 select 一次，即使有索引，几千万条 select 语句对 MySQL 数据库的压力也会非常大。

缓存就是为了弥补存储系统在这些复杂业务场景下的不足，其基本原理是将可能重复使用的数据放到内存中，一次生成、多次使用，避免每次使用都去访问存储系统。

缓存能够带来性能的大幅提升，以 Memcache 为例，单台 Memcache 服务器简单的 key-value 查询能够达到 TPS 50000 以上，其基本的架构是：



（http://pic001.cnblogs.com/img/dudu/200809/2008092816494460.png ）

缓存虽然能够大大减轻存储系统的压力，但同时也给架构引入了更多复杂性。架构设计时如果没有针对缓存的复杂性进行处理，某些场景下甚至会导致整个系统崩溃。今天，我来逐一分析缓存的架构设计要点。

缓存穿透

缓存穿透是指缓存没有发挥作用，业务系统虽然去缓存查询数据，但缓存中没有数据，业务系统需要再次去存储系统查询数据。通常情况下有两种情况：

1. 存储数据不存在

第一种情况是被访问的数据确实不存在。一般情况下，如果存储系统中没有某个数据，则不会在缓存中存储相应的数据，这样就导致用户查询的时候，在缓存中找不到对应的数据，每次都要去存储系统中再查询一遍，然后返回数据不存在。缓存在这个场景中并没有起到分担存储系统访问压力的作用。

通常情况下，业务上读取不存在的数据的请求量并不会太大，但如果出现一些异常情况，例如被黑客攻击，故意大量访问某些读取不存在数据的业务，有可能会将存储系统拖垮。

这种情况的解决办法比较简单，如果查询存储系统的数据没有找到，则直接设置一个默认值（可以是空值，也可以是具体的值）存到缓存中，这样第二次读取缓存时就会获取到默认值，而不会继续访问存储系统。

2. 缓存数据生成耗费大量时间或者资源

第二种情况是存储系统中存在数据，但生成缓存数据需要耗费较长时间或者耗费大量资源。如果刚好在业务访问的时候缓存失效了，那么也会出现缓存没有发挥作用，访问压力全部集中在存储系统上的情况。

典型的就是电商的商品分页，假设我们在某个电商平台上选择“手机”这个类别查看，由于数据巨大，不能把所有数据都缓存起来，只能按照分页来进行缓存，由于难以预测用户到底会访问哪些分页，因此业务上最简单的就是每次点击分页的时候按分页计算和生成缓存。通常情况下这样实现是基本满足要求的，但是如果被竞争对手用爬虫来遍历的时候，系统性能就可能出现问题。

具体的场景有：

- 分页缓存的有效期设置为 1 天，因为设置太长时间的话，缓存不能反应真实的数据。
- 通常情况下，用户不会从第 1 页到最后 1 页全部看完，一般用户访问集中在前 10 页，因此第 10 页以后的缓存过期失效的可能性很大。
- 竞争对手每周来爬取数据，爬虫会将所有分类的所有数据全部遍历，从第 1 页到最后 1 页全部都会读取，此时很多分页缓存可能都失效了。
- 由于很多分页都没有缓存数据，从数据库中生成缓存数据又非常耗费性能（order by limit 操作），因此爬虫会将整个数据库全部拖慢。

这种情况并没有太好的解决方案，因为爬虫会遍历所有的数据，而且什么时候来爬取也是不确定的，可能是每天都来，也可能是每周，也可能是一个月来一次，我们也不可能为了应对爬虫而将所有数据永久缓存。通常的应对方案要么就是识别爬虫然后禁止访问，但这可能会影响 SEO 和推广；要么就是做好监控，发现问题后及时处理，因为爬虫不是攻击，不会进行暴力破坏，对系统的影响是逐步的，监控发现问题后有时间进行处理。

缓存雪崩

缓存雪崩是指当缓存失效（过期）后引起系统性能急剧下降的情况。当缓存过期被清除后，业务系统需要重新生成缓存，因此需要再次访问存储系统，再次进行运算，这个处理步骤耗时几十毫秒甚至上百毫秒。而对于一个高并发的业务系统来说，几百毫秒内可能会接到几百上千个请求。由于旧的缓存已经被清除，新的缓存还未生成，并且处理这些请求的线程都不知道另外有一个线程正在生成缓存，因此所有的请求都会去重新生成缓存，都会去访问存储系统，从而对存储系统造成巨大的性能压力。这些压力又会拖慢整个系统，严重的会造成数据库宕机，从而形成一系列连锁反应，造成整个系统崩溃。

缓存雪崩的常见解决方法有两种：更新锁机制和后台更新机制。

1. 更新锁

对缓存更新操作进行加锁保护，保证只有一个线程能够进行缓存更新，未能获取更新锁的线程要么等待锁释放后重新读取缓存，要么就返回空值或者默认值。

对于采用分布式集群的业务系统，由于存在几十上百台服务器，即使单台服务器只有一个线程更新缓存，但几十上百台服务器一起算下来也会有几十上百个线程同时来更新缓存，同样存在雪崩的问题。因此分布式集群的业务系统要实现更新锁机制，需要用到分布式锁，如 ZooKeeper。

2. 后台更新

由后台线程来更新缓存，而不是由业务线程来更新缓存，缓存本身的有效期设置为永久，后台线程定时更新缓存。

后台定时机制需要考虑一种特殊的场景，当缓存系统内存不够时，会“踢掉”一些缓存数据，从缓存被“踢掉”到下一次定时更新缓存的这段时间内，业务线程读取缓存返回空值，而业务线程本身又不会去更新缓存，因此业务上看到的现象就是数据丢了。解决的方式有两种：

- 后台线程除了定时更新缓存，还要频繁地去读取缓存（例如，1 秒或者 100 毫秒读取一次），如果发现缓存被“踢了”就立刻更新缓存，这种方式实现简单，但读取时间间隔不能设置太长，因为如果缓存被踢了，缓存读取间隔时间又太长，这段时间内业务访问都拿不到真正的数据而是一个空的缓存值，用户体验一般。
- 业务线程发现缓存失效后，通过消息队列发送一条消息通知后台线程更新缓存。可能会出现多个业务线程都发送了缓存更新消息，但其实对后台线程没有影响，后台线程收到消息后更新缓存前可以判断缓存是否存在，存在就不执行更新操作。这种方式实现依赖消息队列，复杂度会高一些，但缓存更新更及时，用户体验更好。

后台更新既适应单机多线程的场景，也适合分布式集群的场景，相比更新锁机制要简单一些。

后台更新机制还适合业务刚上线的时候进行缓存预热。缓存预热指系统上线后，将相关的缓存数据直接加载到缓存系统，而不是等待用户访问才来触发缓存加载。

缓存热点

虽然缓存系统本身的性能比较高，但对于一些特别热点的数据，如果大部分甚至所有的业务请求都命中同一份缓存数据，则这份数据所在的缓存服务器的压力也很大。例如，某明星微博发布“我们”来宣告恋爱了，短时间内上千万的用户都会来围观。

缓存热点的解决方案就是复制多份缓存副本，将请求分散到多个缓存服务器上，减轻缓存热点导致的单台缓存服务器压力。以微博为例，对于粉丝数超过 100 万的明星，每条微博都可以生成

100 份缓存，缓存的数据是一样的，通过在缓存的 key 里面加上编号进行区分，每次读缓存时都随机读取其中某份缓存。

缓存副本设计有一个细节需要注意，就是不同的缓存副本不要设置统一的过期时间，否则就会出现所有缓存副本同时生成同时失效的情况，从而引发缓存雪崩效应。正确的做法是设定一个过期时间范围，不同的缓存副本的过期时间是指定范围内的随机值。

实现方式

由于缓存的各种访问策略和存储的访问策略是相关的，因此上面的各种缓存设计方案通常情况下都是集成在存储访问方案中，可以采用“程序代码实现”的中间层方式，也可以采用独立的中间件来实现。

小结

今天我为你讲了高性能架构设计中缓存设计需要注意的几个关键点，这些关键点本身在技术上都不复杂，但可能对业务产生很大的影响，轻则系统响应变慢，重则全站宕机，架构师在设计架构的时候要特别注意这些细节，希望这些设计关键点和技术方案对你有所帮助。

这就是今天的全部内容，留一道思考题给你吧，分享一下你所在的业务发生过哪些因为缓存导致的线上问题？采取了什么样的解决方案？效果如何？

欢迎你把答案写到留言区，和我一起讨论。相信经过深度思考的回答，也会让你对知识的理解更加深刻。（编辑乱入：精彩的留言有机会获得丰厚福利哦！）

精选留言

bluefantasy 60

我们的系统就出现过类似的问题，开始的时候没有缓存，每次做活动访问量大的时候就会导致反应特别慢。后来通过加redis缓存解决了问题。

对于缓存雪崩问题，我们采取了双key策略：要缓存的key过期时间是t，key1没有过期时间。每次缓存读取不到key时就返回key1的内容，然后触发一个事件。这个事件会同时更新key和key1。

2018-06-05

作者回复

很有创意👍👍

2018-06-05

mapping 13

计算机界两大难题：命名和缓存过期。没用缓存的时候，想着怎么用缓存提升性能，用了缓存又担心数据更新不及时。技术上希望所有的请求都能命中缓存，业务上又恨不得数据实时最新。所以就会引入各种缓存过期策略，如设置过期时间，按规则删除，打版本。这些应该在前期设计缓存系统时规划好，我们最早是将 sql md5 作 key 查询结果存入缓存，结果业务系统数据不一致，要清除缓存简直是噩梦，只能祭出绝招重启 memcache，后面改成按规则删除，在 key 中加上业务和用户的前缀，可以很方便删除某个业务或某个用户的缓存。以上过期策略在前端浏览器也是这样，最简单就是 web 服务器设置静态资源缓存过期时间，如果业务频繁发新版本，过期时间不宜设置太长，但其实每次变动的文件很少，这种策略会导致大部分缓存命中率不高。按规则删除，早期很多网站上会有诊断助手类的东西，页面加载错误点下诊断助手就帮你清除缓存，原理就是对静态文件逐一带上 no-cache 请求头发送 ajax 请求强制覆盖缓存（跟 DevTools 中 disable cache 原理一样）。打版本其实就相当于让浏览器请求一个新版本文件，对于老版本文件就让它在缓存中自生自灭。

2018-06-05

作者回复

你们的缓存设计有点复杂，还不如调整业务，越复杂的方案越容易出错，参考架构设计原则

2018-06-05

loveluckystar 12

讲一个头两天发生的事情，我们的一个业务背后是es做db，之前是通过redis做缓存，缓存一段时间后失效再从es读取，是业务访问加载缓存的方式。有一天线上es集群机器单台出现问题，返回慢，由于分布式的缘故，渐渐拖满了所有请求，缓存失效来查询es发生了超时，加载失败，于是下次访问还是直接访问es。最终缓存全部失效，qps翻了好多倍，直接雪崩，es集群彻底没有响应了。。。之后我们只好先下线这个缓存加载功能，让集群活过来，最终改造缓存加载方式，用后台进程去更新缓存，而不用业务访问加载。

2018-06-05

作者回复

现学现用的案例，很赞👍👍

2018-06-05

三月沙@wecatch 9

好的缓存方案应该从这几个方面入手设计：
1.什么数据应该缓存
2.什么时机触发缓存和以及触发方式是什么
3.缓存的层次和粒度（网关缓存如 nginx，本地缓存如单机文件，分布式缓存如redis cluster，进程内缓存如全局变量）
4.缓存的命名规则和失效规则
5.缓存的监控指标和故障应对方案
6.可视化缓存数据如 redis 具体 key 内容和大小

2018-06-05

作者回复

确实，细节不少，可以写本书了😃

2018-06-05

夏天的味道 4

线上遇到的一个错误：业务查询的结果序列化后放到redis，下次从redis取出来时报错。原来是结果类虽然实现了Serializable接口，但是没有重写serialVersionUID，导致不能成功反序列化。

2018-06-06

鹅米豆发 4

1、最早也是采用后台用数据库，前台用关系型数据库+被动缓存的模式。结果是经常的性能抖动，且缓存一致性问题很难解决。后来我们的多数系统，都采用了前后台分离的模式——后台原始数据仍然是关系型数据库，前台使用缓存作为数据源，两者之间数据实时同步+定时同步+人工触发结合。
这个模式，基本根除了穿透，雪崩，不一致，性能抖动这些。但带来了新的问题，比如数据丢失且不可恢复。我们的做法是，让缓存具备相对可靠的持久化机制+运维体系。
2、遇到过几次热点问题，感觉这个更加棘手些。第一种情况，单Key数据结构本身过大，单个分片出现热点，单次访问的复杂度变大。这个相对容易，可以对key进行拆分，使用hashtag机制分片。第二种情况，数据分片普遍不均衡，较少遇到，遇到就比较棘手。第三种情况，数据分片均衡，但访问不均衡，可以增加副本数量。

2018-06-05

作者回复

缓存持久化是一个不错的方法

2018-06-05

公号-Java大后端 4

上缓存架构的时候，结合以前的实际经历，会有几个值得注意的地方:

1 哪些数据才真正的需要缓存？缓存也并非银弹。既然允许数据缓存，那么在你是可以接受在一定时间区间内的数据不一致性的。(当然可以做到最终一致性)

2 确定好1后，就需要会数据类型进行分类，比如业务数据缓存，http缓存等

3 根据数据类型及访问特点的不同选择不同缓存类型的技术方案。

请问华仔，热点数据存在相当的突发性，临时的扩容似乎也来不及，能否从缓存架构角度如何避免类似微博宕机的事件？

2018-06-05

作者回复

1. 限流

2. 容器化+动态化

3. 业务降级，例如限制评论

2018-06-05

倔强小小 4

我们系统是做美术馆的3d展示的，后台配置的数据有点多，画框数据有几m，单个模型数据有几百k，最早直接mysql直接读取，后来用redis但是感觉效果不理想，又引入ehcache，发现不经过网络传输性能很好，经过网络传输后网络一般还是卡的很，后来又在前端用local storage进行缓存，后端通过rabbitmq进行消息通知前端清除本地缓存，缓存设置有效期都是一天，请问下我们这种情况有更好的方案吗

2018-06-05

作者回复

前后端分离，在node上缓存和渲染试试

2018-06-05

刘磊 4

对缓存的key也需要进行valid，避免无效的key查询缓存

2018-06-05

醇梨子 2

华仔，请教一下，针对这种高并发缓存架构设计中，缓存和存储系统一致性问题怎么保证？比如说商品浏览人数，需要存库，然后又需要放缓存，需要频繁更新数据库。

2018-06-17

作者回复

没法保证，这类数据允许一定的不一致，一定范围内的对用户也没有影响，不要只从技术的角度考虑问题，结合业务考虑技术

2018-06-19

100kg 2

你好，我想请教下，如果两台mysql互为主从，其中一台主键自增步长设置为2，另一个为1，这样做会影响性能吗？比如索引的连贯性之类的

2018-06-06

作者回复

不会，索性是对已经存在的值建立索性数据结构，值的连续性和大小对数据结构本身没有影响，值的数量才会影响索性的大小和性能

2018-06-06

星火燎原 2

如果用redis做分布式锁 从业务上设置超时时间为1s 但是有一些逻辑单元确实需要执行比如3s才能释放掉锁。那么这种“正常的异常情况”应该怎么解决呢?

2018-06-05

作者回复

那为何不直接设置为3秒超时?

2018-06-06

王磊 2

关于缓存雪崩的定义，文中的例子是说单条缓存在生成的时间间隔内有大量相同请求查询存储系统，除此之外，记得还有一个更典型的场景，是大量缓存同时过期，导致大量不同请求去查询存储系统(数据库)。请点评。

2018-06-05

作者回复

一般不会出现这种情况，但缓存服务器宕机的场景和你说的类似，我们一般会要求线上进行模拟测试，假如一台缓存服务器宕机，系统是否能够顶住，如果顶不住，就要增加缓存服务器，或者优化某些key的设计

2018-06-06

武坤 2

在 缓存雪崩的2.后台更新中“缓存系统内存不够时，会‘踢掉’一些缓存数据”，既然是内存不够那么总会有不在缓存里面的数据。后面的两种解决方式怎么能解决这个问题呢?

2018-06-05

作者回复

一定程度上缓解，不能完全解决

2018-06-05

涛哥迷妹 1

大神们的智慧也提供一个互相交流学习的机会赞一个👍

2018-07-22

作者回复

读者卧虎藏龙👍👍

2018-07-24

Only U 1

我来说说我做过的一个项目设计的缓存策略吧，因为需要实时查询所有open状态的基础订单列表信息，所以有个后台进程每分钟刷新一次缓存。但因为open态order太多，导致redis序列化与反序列化过程太久而连接中断，后来我采用的一个方案就是根据每次查询必选的组合

查询条件分别分组然后以单个查询值作为key放到缓存中，这样既达到了按照查询条件过滤的情况，又缩减了每次存取数据的字节数。可能比较简单吧，感觉也没有什么特别的。哈哈😄

2018-06-11

作者回复

我们用过另外一种方式：将查询条件组合成字符串再计算md5，作为缓存的key，优点是简单灵活，缺点是浪费一部分缓存

2018-06-12

bigticket 1

为应对读写不平衡，高并发读取的业务场景，加入缓存机制；缓存又涉及到缓存穿透、缓存雪崩等问题，需要设计合理的过期时间，分页读取等

2018-06-08

万岁爷 1

要是能写些demo就完美了。demo可以课件形式，放到网盘

2018-06-06

作者回复

你可以自己写，不难

2018-06-07

大光头 1

我们遇到的问题是代码bug导致缓存没有过期时间，缓存很快满了，缓存命中率降低，造成对业务很大的压力

2018-06-06

作者回复

做好监控就可以及时发现

2018-06-06

何磊 1

遇到过一次缓存失效导致缓存穿透，很多请求的压力直接到了db。为了处理这种情况，我采用的方案就是：key永不过期，后台有个进程定时更新所有缓存。
文中提到的结合消息队列来更新更具有时效性，非常棒，看到评论中的有一个双key机制，设计很巧妙，不过成本太高了。相当于成本翻倍。

2018-06-06

作者回复

除非特殊场景，一般我还是建议尽量用简单直观甚至粗暴的方案😂

2018-06-06