为什么使用 Redis及其产品定位

作者 田琪 发布于 2011年7月19日 | 11 讨论

传统MySQL+ Memcached架构遇到的问题

实际MySQL是适合进行海量数据存储的，通过Memcached将热点数据加载到cache，加速访问。很多公司都曾经使用过这样的架构，但随着业务数据量的不断增加和访问量的持续增长，我们遇到了很多问题：

1. MySQL需要不断进行拆库拆表，Memcached也需不断跟着扩容，扩容和维护工作占据大量开发时间。
2. Memcached与MySQL数据库数据一致性问题。
3. Memcached数据命中率低或down机，大量访问直接穿透到DB，MySQL无法支撑。
4. 跨机房cache同步问题。

众多NoSQL百花齐放，如何选择

最近几年，业界不断涌现出很多各种各样的NoSQL产品，那么如何才能正确地使用好这些产品，最大化地发挥其长处，是我们需要深入研究和思考的问题。实际归根结底最重要的是了解这些产品的定位，并且了解每款产品的tradeoffs，在实际应用中做到扬长避短。总体上这些NoSQL主要用于解决以下几种问题：

1. 少量数据存储，高速读写访问。此类产品通过数据全部in-memory 的方式来保证高速访问，同时提供数据落地的功能，实际这正是Redis最主要的适用场景。
2. 海量数据存储，分布式系统支持，数据一致性保证，方便的集群节点添加/删除。 （Cassandra、HBase）
3. 这方面最具代表性的是dynamo和bigtable 2篇论文所阐述的思路。前者是一个完全无中心的设计，节点之间通过gossip方式传递集群信息，数据保证最终一致性；后者是一个中心化的方案设计，通过类似一个分布式锁服务来保证强一致性。数据写入先写内存和redo log，然后定期compat归并到磁盘上，将随机写优化为顺序写，提高写入性能。
4. Schema free，auto-sharding等。比如目前常见的一些文档数据库都是支持schema-free的，直接存储json格式数据，并且支持auto-sharding等功能，比如mongodb。

面对这些不同类型的NoSQL产品,我们需要根据我们的业务场景选择最合适的产品。

Redis适用场景，如何正确的使用

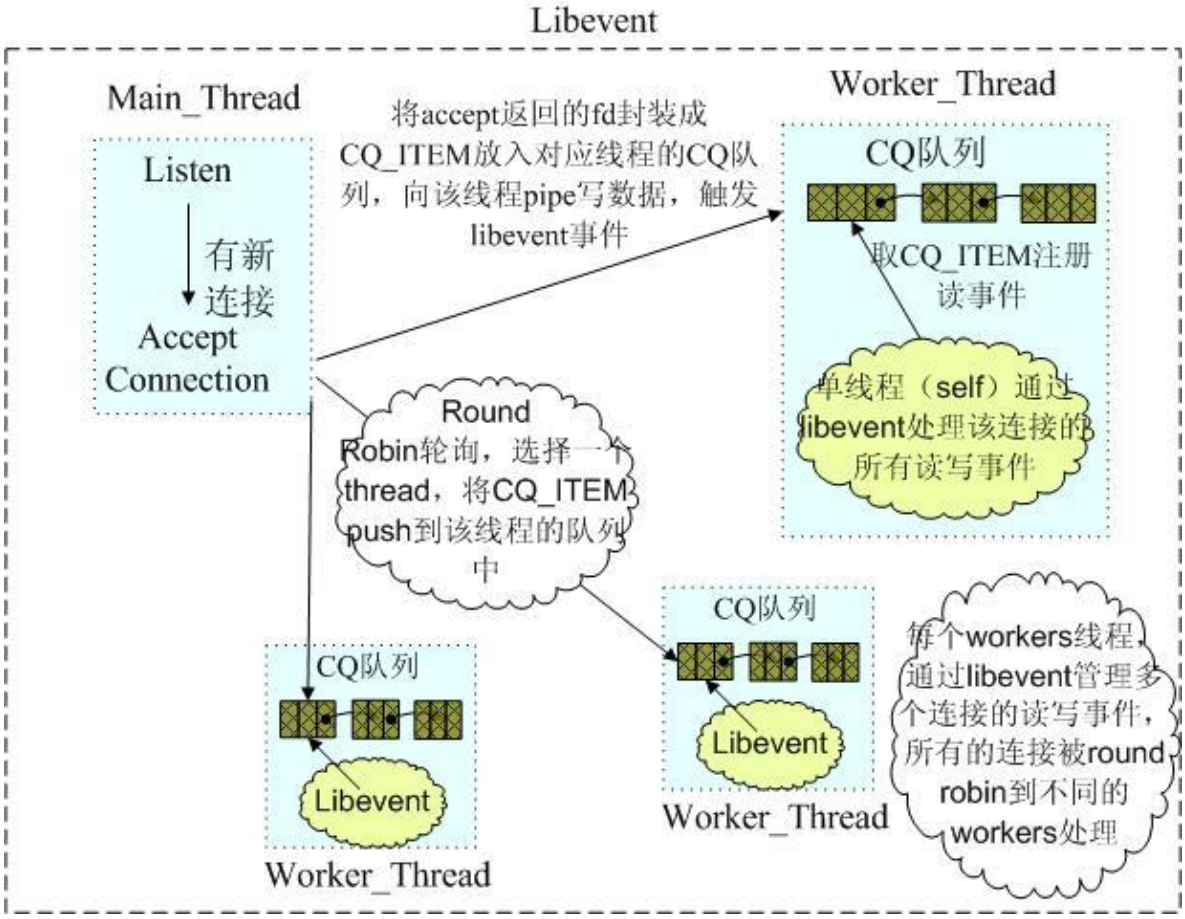
前面已经分析过，Redis最适合所有数据in-memory的场景，虽然Redis也提供持久化功能，但实际更多的是一个disk-backed的功能，跟传统意义上的持久化有比较大的差别，那么可能大家就会有疑问，似乎Redis更像一个加强版的Memcached，那么何时使用Memcached，何时使用Redis呢？

Redis与Memcached的比较

1. 网络IO模型

   Memcached是多线程、非阻塞IO复用的网络模型，分为监听主线程和worker子线程，监听线程监听网络连接，接受请求后，将连接描述字pipe 传递给worker线程，进行读写IO, 网络层使用libevent封装的事件库。多线程模型可以发挥多核作用，但是引入了cache coherency和锁的问题，比如，Memcached最常用的stats 命令，实际Memcached所有操作都要对这个全局变量加锁，进行计数等工作，带来了性能损耗。

   【问题】



（Memcached网络IO模型）

Redis使用单线程的IO复用模型，自己封装了一个简单的AeEvent事件处理框架，主要实现了epoll、kqueue和select。对于单纯只有IO操作来说，单线程可以将速度优势发挥到最大，但是Redis也提供了一些简单的计算功能，比如排序、聚合等，对于这些操作，单线程模型实际会严重影响整体吞吐量，CPU计算过程中，整个IO调度都是被阻塞住的。 【问题】

2. 内存管理方面

Memcached使用预分配的内存池的方式，使用slab和大小不同的chunk来管理内存，Item根据大小选择合适的chunk存储，内存池的方式可以省去申请/释放内存的开销，并且能减小内存碎片产生，但这种方式也会带来一定程度上的空间浪费，并且在内存仍然有很大空间时，新的数据也可能会被剔除，原因可以参考Timyang的文章：http://timyang.net/data/Memcached-lru-evictions/ 【问题】

Redis使用现场申请内存的方式来存储数据，并且很少使用free-list等方式来优化内存分配，会在一定程度上存在内存碎片，Redis根据存储命令参数，会把带过期时间的数据单独存放在一起，并把它们称为临时数据，非临时数据是永远不会被剔除的，即便物理内存不够，导致swap也不会剔除任何非临时数据（但会尝试剔除部分临时数据），这点上Redis更适合作为存储而不是cache。 最新版的Redis中，非临时数据也是可以被剔除的。在redis.conf中配置maxmemory-policy allkeys-lru

3. 数据一致性问题 复合操作

Memcached提供了cas命令，可以保证多个并发访问操作同一份数据的一致性问题。 Redis没有提供cas 命令，并不能保证这点，不过Redis提供了事务的功能，可以保证一串命令的原子性，中间不会被任何操作打断。

4. 存储方式及其它方面

Memcached基本只支持简单的key-value存储，不支持枚举，不支持持久化和复制等功能

Redis除key/value之外，还支持list,set,sorted set,hash等众多数据结构，提供了KEYS

进行枚举操作，但不能在线上使用，如果需要枚举线上数据，Redis提供了工具可以直接扫描其dump文件，枚举出所有数据，Redis还同时提供了持久化和复制等功能。

5. 关于不同语言的客户端支持

   在不同语言的客户端方面，Memcached和Redis都有丰富的第三方客户端可供选择，不过因为Memcached发展的时间更久一些，目前看在客户端支持方面，Memcached的很多客户端更加成熟稳定，而Redis由于其协议本身就比Memcached复杂，加上作者不断增加新的功能等，对应第三方客户端跟进速度可能会赶不上，有时可能需要自己在第三方客户端基础上做些修改才能更好的使用。

【适用场景】

根据以上比较不难看出，当我们不希望数据被踢出，或者需要除key/value之外的更多数据类型时，或者需要落地功能时，使用Redis比使用Memcached更合适。

关于Redis的一些周边功能

Redis除了作为存储之外还提供了一些其它方面的功能，比如聚合计算、pubsub、scripting等，对于此类功能需要了解其实现原理，清楚地了解到它的局限性后，才能正确的使用，比如pubsub功能，这个实际是没有任何持久化支持的，消费方连接闪断或重连之间过来的消息是会全部丢失的，又比如聚合计算和scripting等功能受Redis单线程模型所限，是不可能达到很高的吞吐量的，需要谨慎使用。

总的来说Redis作者是一位非常勤奋的开发者，可以经常看到作者在尝试着各种不同的新鲜想法和思路，针对这些方面的功能就要求我们需要深入了解后再使用。

总结：

1. Redis使用最佳方式是全部数据in-memory。
2. Redis更多场景是作为Memcached的替代者来使用。
3. 当需要除key/value之外的更多数据类型支持时，使用Redis更合适。
4. 当存储的数据不能被剔除时，使用Redis更合适。

后续关于Redis文章计划：

1. Redis数据类型与容量规划。
2. 如何根据业务场景搭建稳定、可靠、可扩展的Redis集群。
3. Redis参数，代码优化及二次开发基础实践。

关于作者

田琪，目前负责新浪微博平台底层架构与研发工作，之前曾担任搜狐白社会实时游戏平台核心架构工作，主要关注webgame, 分布式存储，nosql 和 erlang 等领域，目前主要从事mysql源代码的一些深入研究工作，浪微博：http://weibo.com/bachmozart。