Redis 常见问题


人生走马灯 (/u/dc3b69a5b174) + 关注
2016.11.16 16:49* 字数 2538 阅读 60658 评论 3 喜欢 4
(/u/dc3b69a5b174)

1. 使用redis有哪些好处?

- 速度快，因为数据存在内存中，类似于HashMap，HashMap的优势就是查找和操作的时间复杂度都是O(1)
- 支持丰富数据类型，支持string，list，set，sorted set，hash
- 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行
- 丰富的特性：可用于缓存，消息，按key设置过期时间，过期后将会自动删除

2. redis相比memcached有哪些优势?

- memcached所有的值均是简单的字符串，redis作为其替代者，支持更为丰富的数据类型
- redis的速度比memcached快很多
- redis可以持久化其数据
- 值有512MB（memcached最大是1MB），

3. redis常见性能问题和解决方案：

- Master最好不要做任何持久化工作，如RDB内存快照和AOF日志文件
- 如果数据比较重要，某个Slave开启AOF备份数据，策略设置为每秒同步一次
- 为了主从复制的速度和连接的稳定性，Master和Slave最好在同一个局域网内
- 尽量避免在压力很大的主库上增加从库
- 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...

这样的结构方便解决单点故障问题，实现Slave对Master的替换。如果Master挂了，可以立刻启用Slave1做Master，其他不变。

4. mySQL里有2000w数据，redis中只存20w的数据，如何保证redis中的数据都是热点数据

相关知识：redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。redis 提供 6种数据淘汰策略：

- voltile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

- volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰
- volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰
- allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰
- allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰
- no-enviction（驱逐）：禁止驱逐数据

  redis 确定驱逐某个键值对后，会删除这个数据并，并将这个数据变更消息发布到本地（AOF 持久化）和从机（主从连接）。

5. Memcache与Redis的区别都有哪些?

- 存储方式

  Memecache把数据全部存在内存之中，断电后会挂掉，数据不能超过内存大小。

  Redis有部份存在硬盘上，这样能保证数据的持久性。

- 数据支持类型

  Memcache对数据类型支持相对简单。

  Redis有复杂的数据类型。

- 使用底层模型不同

  它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。

  Redis直接自己构建了VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

- value大小

  redis最大可以达到1GB，而memcache只有1MB

- 线程

  redis目前是单线程,memcached 是多线程的

- Libevent。

  和Memcached不同，Redis并没有选择libevent。Libevent为了迎合通用性造成代码庞大(目前Redis代码还不到libevent的1/3)及牺牲了在特定平台的不少性能。Redis用libevent中两个文件修改实现了自己的epoll event loop。

扩展 (https://link.jianshu.com?t=http://blog.csdn.net/ketonfly/article/details/39434819)

6. Redis 常见的性能问题都有哪些？如何解决?

- Master写内存快照，save命令调度rdbSave函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以Master最好不要写内存快照。

- Master
  AOF持久化，如果不重写AOF文件，这个持久化方式对性能的影响是最小的，但是AOF文件会不断增大，AOF文件过大会影响Master重启的恢复速度。Master最好不要做任何持久化工作，包括内存快照和AOF日志文件，特别是不要启用内存快照做持久化,如果数据比较关键，某个Slave开启AOF备份数据，策略为每秒同步一次。

- Master调用BGREWRITEAOF重写AOF文件，AOF在重写的时候会占大量的CPU和内存资源，导致服务load过高，出现短暂服务暂停现象。

- Redis主从复制的性能问题，为了主从复制的速度和连接的稳定性，Slave和Master最好在同一个局域网内

7, redis 最适合的场景

Redis最适合所有数据in-momory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，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?

- 如果简单地比较Redis与Memcached的区别，大多数都会得到以下观点：

  1 、Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，zset，hash等数据结构的存储。
  2 、Redis支持数据的备份，即master-slave模式的数据备份。
  3 、Redis支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。

- 会话缓存（Session Cache）

  最常用的一种使用Redis的情景是会话缓存（session cache）。用Redis缓存会话比其他存储（如Memcached）的优势在于：Redis提供持久化。当维护一个不是严格要求一致性的缓存时，如果用户的购物车信息全部丢失，大部分人都会不高兴的，现在，他们还会这样吗?

  幸运的是，随着 Redis 这些年的改进，很容易找到怎么恰当的使用Redis来缓存会话的文档。甚至广为人知的商业平台Magento也提供Redis的插件。

- 全页缓存（FPC）

  除基本的会话token之外，Redis还提供很简便的FPC平台。回到一致性问题，即使重启了Redis实例，因为有磁盘的持久化，用户也不会看到页面加载速度的下降，这是一个极大改进，类似PHP本地FPC。

  再次以Magento为例，Magento提供一个插件来使用Redis作为全页缓存后端。

  此外，对WordPress的用户来说，Pantheon有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

- 队列

  Reids在内存存储引擎领域的一大优点是提供 list 和 set 操作，这使得Redis能作为一个很好的消息队列平台来使用。Redis作为队列使用的操作，就类似于本地程序语言（如Python）对 list 的 push/pop 操作。

如果你快速的在Google中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用Redis创建非常好的后端工具，以满足各种队列需求。例如，Celery有一个后台就是使用Redis作为broker，你可以从这里去查看。

- 排行榜/计数器

  Redis在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，Redis只是正好提供了这两种数据结构。所以，我们要从排序集合中获取到排名最靠前的10个用户–我们称之为“user_scores”，我们只需要像下面一样执行即可：

  当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行：

  ZRANGE user_scores 0 10 WITHSCORES

  Agora Games就是一个很好的例子，用Ruby实现的，它的排行榜就是使用Redis来存储数据的，你可以在这里看到。

- 发布/订阅

  最后（但肯定不是最不重要的）是Redis的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用Redis的发布/订阅功能来建立聊天系统！

  Redis提供的所有特性中，我感觉这个是喜欢的人最少的一个，虽然它为用户提供如果此多功能。

来源 (https://link.jianshu.com?t=http://www.cnblogs.com/hwisecn/p/5912374.html)

redis有时候会请求超时，已知都是固定的一个时间

请求redis超时，如果时间较长，比如60s或者75s这样的，可能是redis的timeout配置。
目前是时间较短，200ms就超时，一般来说redis没道理在能处理请求的时候报超时错误，会否是现在超过了redis设置的最大连接数maxclients，导致拒绝服务；会否是client自身的连接超时设置。
无论如何，CS模式的两端都有可能是原因。所以根据具体业务和网络环境需要进行详细的分析 。

来源 (https://link.jianshu.com?t=https://www.zhihu.com/question/28812501)