**Redis 简介**

Redis 是完全开源免费的，遵守BSD协议，是一个高性能的key-value数据库。

Redis 与其他 key - value 缓存产品有以下三个特点：

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

**Redis 优势**

* 性能极高 – Redis能读的速度是110000次/s,写的速度是81000次/s 。
* 丰富的数据类型 – Redis支持二进制案例的 Strings, Lists, Hashes, Sets 及 Ordered Sets 数据类型操作。
* 原子 – Redis的所有操作都是原子性的，意思就是要么成功执行要么失败完全不执行。单个操作是原子性的。多个操作也支持事务，即原子性，通过MULTI和EXEC指令包起来。
* 丰富的特性 – Redis还支持 publish/subscribe, 通知, key 过期等等特性。

**Redis与其他key-value存储有什么不同？**

* Redis有着更为复杂的数据结构并且提供对他们的原子性操作，这是一个不同于其他数据库的进化路径。Redis的数据类型都是基于基本数据结构的同时对程序员透明，无需进行额外的抽象。
* Redis运行在内存中但是可以持久化到磁盘，所以在对不同数据集进行高速读写时需要权衡内存，因为数据量不能大于硬件内存。在内存数据库方面的另一个优点是，相比在磁盘上相同的复杂的数据结构，在内存中操作起来非常简单，这样Redis可以做很多内部复杂性很强的事情。同时，在磁盘格式方面他们是紧凑的以追加的方式产生的，因为他们并不需要进行随机访问。

## Redis介绍

介绍Redis之前，先了解下NoSQL （Not noly SQL）不仅仅是SQL属于非关系型数据库；Redis就属于非关系型数据库，传统的Mysql ,oracle ,sql server 等 都是关系型数据库。

为什么需要NoSQL，主要应对以下问题，传统关系型数据库力不从心：   
（1）High performance -高并发读写   
（2）Huge Storage-海量数据的高效率存储和访问   
（3）High Scalablility && High Availability 高可扩展性和高可用性

NoSQL 产品有MongoDB Redis等等 ，但是Redis是目前主流。

## NoSQL的特点

易扩展   
灵活的数据模型   
大数据量，高性能   
高可用   
高性能键值对数据库，支持的键值数据类型：字符串类型，列表类型，有序集合类型，散列类型，集合类型。

## Redis的应用场景

缓存   
任务队列   
网站访问统计   
数据过期处理   
应用排行榜   
分布式集群架构中的session分离

首先，分布式缓存框架 可以 看成是nosql的一种

## （1）什么是redis?

redis 是一个基于内存的高性能key-value数据库。

## （2）Reids的特点

Redis本质上是一个Key-Value类型的内存数据库，很像memcached，整个数据库统统加载在内存当中进行操作，定期通过异步操作把数据库数据flush到硬盘上进行保存。因为是纯内存操作，Redis的性能非常出色，每秒可以处理超过 10万次读写操作，是已知性能最快的Key-Value DB。  
Redis的出色之处不仅仅是性能，Redis最大的魅力是支持保存多种数据结构，此外单个value的最大限制是1GB，不像 memcached只能保存1MB的数据，因此Redis可以用来实现很多有用的功能，比方说用他的List来做FIFO双向链表，实现一个轻量级的高性 能消息队列服务，用他的Set可以做高性能的tag系统等等。另外Redis也可以对存入的Key-Value设置expire时间，因此也可以被当作一 个功能加强版的memcached来用。  
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，因此Redis适合的场景主要局限在较小数据量的高性能操作和运算上。

## （3）Redis支持的数据类型

Redis通过Key-Value的单值不同类型来区分, 以下是支持的类型:  
Strings  
Lists  
Sets 求交集、并集  
Sorted Set   
hashes

## （4）为什么redis需要把所有数据放到内存中？

Redis为了达到最快的读写速度将数据都读到内存中，并通过异步的方式将数据写入磁盘。所以redis具有快速和数据持久化的特征。如果不将数据放在内存中，磁盘I/O速度为严重影响redis的性能。在内存越来越便宜的今天，redis将会越来越受欢迎。  
如果设置了最大使用的内存，则数据已有记录数达到内存限值后不能继续插入新值。

## （5）Redis是单进程单线程的

redis利用队列技术将并发访问变为串行访问，消除了传统数据库串行控制的开销

## （6）虚拟内存

当你的key很小而value很大时,使用VM的效果会比较好.因为这样节约的内存比较大.  
当你的key不小时,可以考虑使用一些非常方法将很大的key变成很大的value,比如你可以考虑将key,value组合成一个新的value.  
vm-max-threads这个参数,可以设置访问swap文件的线程数,设置最好不要超过机器的核数,如果设置为0,那么所有对swap文件的操作都是串行的.可能会造成比较长时间的延迟,但是对数据完整性有很好的保证.

自己测试的时候发现用虚拟内存性能也不错。如果数据量很大，可以考虑分布式或者其他数据库

## （7）分布式

redis支持主从的模式。原则：Master会将数据同步到slave，而slave不会将数据同步到master。Slave启动时会连接master来同步数据。

**这是一个典型的分布式读写分离模型。我们可以利用master来插入数据，slave提供检索服务。这样可以有效减少单个机器的并发访问数量**

### （8）读写分离模型

通过增加Slave DB的数量，读的性能可以线性增长。为了避免Master DB的单点故障，集群一般都会采用两台Master DB做双机热备，所以整个集群的读和写的可用性都非常高。  
读写分离架构的缺陷在于，不管是Master还是Slave，每个节点都必须保存完整的数据，如果在数据量很大的情况下，集群的扩展能力还是受限于单个节点的存储能力，而且对于Write-intensive类型的应用，读写分离架构并不适合。

### （9）数据分片模型

为了解决读写分离模型的缺陷，可以将数据分片模型应用进来。

可以将每个节点看成都是独立的master，然后通过业务实现数据分片。

结合上面两种模型，可以将每个master设计成由一个master和多个slave组成的模型。

**（10）Redis的回收策略**

* volatile-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