### 作业一 redis与memcache的区别及应用场景

#### 不同点

1.redis在数据支持上要比memecache多的多，Redis除key/value之外，还支持list,set,sorted set,hash等众多数据结构，

2.Redis和Memcache都是将数据存放在内存中，都是内存数据库。不过memcache还可用于缓存其他东西，例如图片、视频等等。

3．Redis不仅仅支持简单的k/v类型的数据，同时还提供list，set，hash等数据结构的存储。

4.虚拟内存–Redis当物理内存用完时，可以将一些很久没用到的value 交换到磁盘 。

5.redis支持数据持久化，redis有部份存在硬盘上，这样能保证数据的持久性；memcache不支持数据持久化，断电就要重启。

#### 应用场景

1.对于key-value这样简单的数据储存，memcache的内存使用率更高。如果采用hash结构，redis的内存使用率会更高。

2.数据持久化和主从复制时，只有redis拥有这两个特性。如果目标是构建一个缓存在升级或者重启后之前的数据不会丢失的话，那也只能选择redis。

3.如果需要缓存的数据只是key-value这样简单的结构时，采用memcache，它也足够的稳定可靠。如果涉及到存储，排序等一系列复杂的操作时，毫无疑问选择redis。

4.由于Redis只使用单核，而Memcached可以使用多核，所以在比较上，平均每一个核上Redis在存储小数据时比Memcached性能更高。而在100k以上的数据中，Memcached性能要高于Redis。

因此，有持久化需求或者对数据结构和处理有高级要求的应用，选择redis，其他简单的key/value存储，选择memcache

#### Redis五种类型

一：字符串类型string

字符串类型是Redis的最基本类型，它可以存储任何形式的字符串。其它的四种类型都是字符串类型的不同形式。

最基本的命令：GET、SET

整数递增：INCR

增加指定的整数：INCRBY

整数递减：DECR

减少指定的整数：DECRBY

增加指定浮点数：INCRBYFLOAT

向尾部追加值：APPEND

获取字符串长度：STRLEN

获取多个键值：MGET

设置多个键值：MSET

二进制指定位置值：GETBIT

设置二进制位置值：SETBIT

二进制是1的个数：BITCOUNT

位运算：BITOP

偏移：BITPOS

二：散列类型hash

设置单个：HSET

设置多个：HMSET

读取单个：HGET

读取多个：HMGET

读取全部：HGETALL

判断字段是否存在：HEXISTS 语法：HEXISTS key field，存在返回1 ，不存在返回0

字段不存在时赋值：HSETNX 语法：HSETNX key field value，与hset命令不同，hsetnx是键不存在时设置值

增加数字：HINCRBY

删除字段：HDEL

只获取字段名：HKEYS

只获取字段值：HVALS

字段数量：HLEN

三：列表类型（list）

内部使用双向链表实现，所以获取越接近两端的元素速度越快，但通过索引访问时会比较慢

添加左边元素：LPUSH

添加右边元素：RPUSH

移除左边第一个元素：LPOP

移除右边第一个元素：RPOP

列表元素个数：LLEN

删除指定值：LREM

四：集合类型（set）

集合类型值具有唯一性，常用操作是向集合添加、删除、判断某个值是否存在，集合内部是使用值为空的散列表实现的。

添加元素：SADD

删除元素：SREM

获取全部元素：SMEMBERS

值是否存在：SISMEMBER

差运算：SDIFF

交运算：SINTER

并运算：SUNION

集合元素个数：SCARD

五：有序集合类型

添加集合元素：ZADD

获取元素分数：ZSCORE

元素小到大：ZRANGE

元素大到小：ZREVRANGE

指定分数范围元素：ZRANGEBYSCORE

增加分数：ZINCRBY

### 作业2 redis和MongoDB区别

#### 不同点

1.MongoDB与Redis都依赖内存，但redisTPS高于MongoDB

2.MongoDB有丰富的数据表达,索引;最类似于关系型数据库,支持丰富的查询语句，redis数据丰富,较少的IO

3.MongoDB适合大数据量存储,依赖系统虚拟内存,采用镜像文件存储;内存占用率比较高,官方建议独立部署在64位系统，Redis2.0后支持虚拟内存特性(VM) 突破物理内存限制;数据可以设置时效性,类似于memcache

4.MongoDB支持master-slave,replicatset(内部采用paxos选举算法,自动故障恢复),auto sharding机制,对客户端屏蔽了故障转移和切片机制，redis依赖客户端来实现分布式读写;主从复制时,每次从节点重新连接主节点都要依赖整个快照,无增量复制;不支持auto sharding,需要依赖程序设定一致性hash机制

5.MongoDB从1.8版本后,采用binlog方式(类似Mysql) 支持持久化，redis依赖快照进行持久化;AOF增强可靠性;增强性的同时,影响访问性能

6.MongoDB不支持事务,靠客户端保证，redis支持事务,比较脆,仅能保证事务中的操作按顺序执行

7.MongoDB内置数据分析功能，redis不支持数据分析

8.MongoDB海量数据的访问效率提升，相比之下redis较小数据量的性能和运算

#### 应用场景

1.当TPS(每秒传输的事物处理个数)非常高的时候，可以用到redis中数据持久化和分布式锁的内容，通过redis数据持久化，我们可以将缓存的数据保存到本地中来。此时使用redis较好。

2.当拥有海量数据但要求有限查询时间可以采用mongodb中的分片来实现，通过mongodb的分片机制，我们可以将海量的数据查询分别负载到不同的分片服务器上面，最后将数据查询的数据结果整合到一起。基于这种情况，不管数据量有多大，我们都可以实现快速的查询功能，查询时间约等于（数据量/分片数量）。分片其实本身就是一种高可用性的方案，因为每一个分片都保留着完整的一份数据，每次插入数据的时候，先插入一个主分片中，然后同步复制到所有从分片中，即使一个分片挂了，其余分片也能自动升级为主分片，继续工作。