1.6 HyperLogLog: 用于粗略统计UV/PV;

* 基本指令：pfadd/pfcount/pfmerge
* 基本原理：随机数N与地位连续0的位数之间的关系。

1.7 布隆过滤器：检查集合里面的元素是否存在，简单的去重功能；

* 基本指令：bf.add/bf.exists/bf.madd/bf.mexists;
* 初始化：bf.reserve 参数：key，error rate，initial size
* 基本原理：底层存储的是位数组，使用多个hash对value计算位置并设置为1，检测是否存在使用hash并检测位置是否都是1；所以初始大小的设置很重要。

1.10 GeoHash：经纬度位置标示，（未完待续）

2.1 鞭辟入里-线程IO模型：

* Redis是单线程程序；
* 非阻塞IO，读写不会阻塞，直接返回，事件轮询（多路复用），select函数。
* 指令队列，每个连接保持一个指令队列，按照时间到来的先后执行；
* 响应队列，与指令队列类似；
* 定时任务，定时任务维持一个最小堆结构，最上面的任务的等待时间就是selec函数的timeout时间。

2.2 交头接耳，通讯协议：

* Redis通信的数据的协议很简单，具体协议的内荣可以看书主要是就是几个简单的要素，并使用\r\n结束；

2.3 持久化：

* Snapshot：一次全量备份；COW(copy on write)，备份的时候产生子进程，共享同样的数据，由于linux系统得机制，主进程修改数据时，会将数据所在的内存页复制出来一份做修改。
* AOF： 命令日志；只记录变更日志，bgrewriteaof完成瘦身工作（子进程一次生成当前数据的所有的写操作）替换旧的aof文件；因为是日志写入到缓存器，所以可能是会丢失，一般1s调用fsync同步缓存器的内容到aof日志；redis的主节点不进行持久化，从节点进行；混合持久化，使用snapshot记录快照，使用aof记录快照开始的增量日志，然后恢复时，一起恢复，避免单独使用快照造成数据丢失，单独使用aof造成恢复事件过长的问题。
* Redis只适合用来做缓存，用来做数据库不合适？
* Redis为什么先执行指令，之后在记录aof日志？

2.4 管道

* 管道是客户端提供的，不是服务端提供的；与HTTP的管道功能类似；

2.5 事务

* Redis也有事务，但并没有实现事务的全部特点，只是部分的；
* 指令的常用命令begin、commit、rollback；redis事务的命令multi、exec、discard；开始multi后，后续的指令都会进入服务器的指令队列；
* 中间指令执行失败不会影响后续指令执行；
* Multi特别适合使用管道执行，节省网络的传输时间；
* Watch，乐观锁，检查变量在事务之前是否被修改了；

2.6 pub/sub

* 消息多播：同样的消息被发到不同的队列，由不同的消费者处理，完成不同的逻辑，如果不放到不同的消息队列，那么不同额消费者就要串行处理；
* Redis使用pubsub支持消息多播（发布订阅模式），操作的方法是非阻塞的，可以使用listen方法阻塞消费，直到消息到来，可以及时进行处理；
* 可以一次订阅多个key，也可以订阅key的模式；
* 消息输出的格式：data，channel（key），type（类型），pattern=None；
* 缺点，断连期间消息丢失，停机期间，消息丢失；5.0版本使用Stream代替；

2.7 小对象压缩

* 使用32bit内存地址编译，适用于使用内存不超过4BG的情况，因为地址限度；内存地址空间占用小一半；
* 集合数据结构中数据少时，使用紧凑型数据结构存储；比如ziplist，intset；
* 删除key，内存不会被回收，可以继续复用，如果页面内没有有效的key了，则删除；
* Redis的内存分配是使用第三方库做的。

3.1 主从同步

* CAP原理：C， Consistent 一致性；A，Availability 可用性；P Partition Tolerance 分区容忍性，网络分区也就是网络断开，分布式系统无法同步变更信息；这时，一致性与可用性难以2全；
* Redis采用最终一致性；
* 增量同步（通过指令同步），维持一个环形队列缓存主节点指令，如果断开时间太长，同步的指令可能会被覆盖，此时系统会发起快照同步；
* 快照同步，直接数据同步，从节点全量加载；加载完后进行增量同步；
* Wait指令同步进行复制；
* Redis只做缓存，不需要同步，如果有持久化，需要同步保证数据安全。