Memcached是一个高性能(可以理解为一个巨大的哈希表，性能当然高)的分布式内存缓存服务器(也可以理解为系统，数据库)。

Memcached是C/S架构，由C语言实现，它虽然是分布式的，但其实多台服务器的Memcached之间没有通信，它的分布式是通过客户端实现的。Memcached的两个核心组件是服务器(ms)和客户端(mc)，一个分布式的Memcached系统，通常会有多个ms(也可能有多个mc)，一次Memcached查询，mc客户端先通过计算key的hash值来确定它在哪个ms服务器上，然后会朝这个ms上发送一个网络请求，ms收到请求后也会根据一个内部哈希算法(计算key在哪个chunk中)，查找到真正的数据，这就是Memcached的两阶段哈希。

例：，假设有3个客户端1, 2, 3，3台memcached A, B, C：

Client 1想把数据”barbaz”以key “foo”存储。Client 1首先参考节点列表（A, B, C），计算key “foo”的哈希值，假设memcached B被选中。接着，Client 1直接connect到memcached B，通过key “foo”把数据”barbaz”存储进去。

查询时，Client 2使用与Client 1相同的客户端库（意味着阶段一的哈希算法相同），也拥有同样的memcached列表（A, B, C）。于是，经过相同的哈希计算（阶段一），Client 2计算出key “foo”在memcached B上，然后它直接请求memcached B，得到数据”barbaz”。

各种客户端在memcached中数据的存储形式是不同的（perl Storable, php serialize, java hibernate, JSON等）。一些客户端实现的哈希算法也不一样。但是，memcached服务器端的行为总是一致的。

Memcached的数据都存储在内存中，且不支持持久化，服务器重启后，所有数据都会丢失，当内存中的缓存的数据容量达到设定的内存值时，会采用LRU算法删除过期的数据。

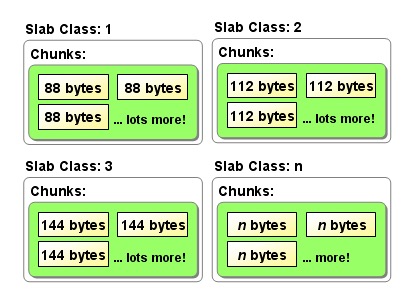
Memcached的内存管理机制：采用Slab Allocation来分配和管理内存，这是一种预分配机制。传统的内存机制是通过malloc来分配内存，然后通过free来回收内存，这容易造成内存碎片。

内存碎片：简单来说就是内存中用户无法使用的小且不连续的内存分区，分为内部碎片和外部碎片。

内部碎片：通过C++类的长度就可以知道，内存大小相对于类中的变量大小而言，往往会多分配一些，多出的这些字节也不会使用，就成了内存碎片。

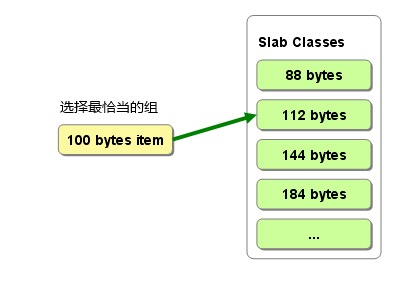
外部碎片：频繁的内存分配和回收会导致外部内存碎片产生。例：对于长度是100单位的连续空闲内存空间，先申请10单位的内存空间，从0-9，再申请5单位的，10-14，如果此时释放了0-9，重新变成空闲内存，但之后分配的内存都大于10，这样0-9就无法再被使用，成了外部碎片。

为了解决这些内存碎片问题，Memcached采用Slab Allocation机制，按照预先规定的大小，将分配的内存以page为单位，默认的page大小为1M(可以自己设定)，分割成各种尺寸的块(chunk)，并把尺寸相同的块分成组。page一旦被分配，重启前不会回收或重新分配，这样就可以大大减少内存碎片。



默认情况下，下一个slab的最大值是上一个的1.25倍(也可自己设置)。

在Memcached中申请内存时，Memcached首先会选择合适的slab，然后看看是否有空闲的chunk，如果有就直接使用，如果没有就会进行申请，申请时以page为单位，无论使用多少，都会有1page的内存分配给该slab，如果没有空闲的page了，即整个Memcached已经达到了设定的最大内存，就会对该slab进行LRU，而不是整个Memcached进行LRU。



Memcached的最大优势就是可扩展性高，因为客户端进行了一次哈希计算，所以可以在系统中增加大量的Memcached，并且由于各个Memcached服务器没有通信，也不会增加Memcached的负载，

Memcached没有数据冗余机制，如果某个结点失去了所有数据，那么需要重新从数据源(例如数据库)中取获取数据。

Memcached没有容错机制，如果某个结点down掉了，需要应用程序自己处理失效的问题，有以下几种方式：

1. 忽略它，等待其恢复，在此之前用其它结点代替
2. 把失效结点移除掉，做这个操作要非常谨慎，在默认情况下，客户端添加或移除结点，会导致所有的缓存不可用
3. 启动热备结点，替换到失效结点的IP，防止哈希紊乱
4. 如果希望添加或删除结点不影响之前的缓存内容，可用采用一致性哈希算法，目前有很多客户端支持这种算法。

Memcached没有身份验证机制，如果你希望进行访问限制，可以使用防火墙。

Memcached接受的最大长度的key是250字节。

Memcached数据的最长失效时间是30天，也可以自己手动设置，但不能超过30天。

Memcached能存储的最大单个数据item是1M的，超过了需要进行拆分。

不同的Memcached的结点上的缓存空间可以是不同的，但一般都是相同的。

所有的被发送到memcached的单个命令是完全原子的。如果您针对同一份数据同时发送了一个set命令和一个get命令，它们不会影响对方。它们将被串行化、先后执行。即使在多线程模式，所有的命令都是原子的，也就是说执行单个命令时，不会被打断掉。

命令序列不是原子的。如果您通过get命令获取了一个item，修改了它，然后想把它set回memcached，我们不保证这个item没有被其他进程（process，未必是操作系统中的进程）操作过。在并发的情况下，您也可能覆写了一个被其他进程set的item。

memcached 1.2.5以及更高版本，提供了gets和cas命令，它们可以解决上面的问题。如果您使用gets命令查询某个key的item，memcached会给您返回该item当前值的唯一标识。如果您覆写了这个item并想把它写回到memcached中，您可以通过cas命令把那个唯一标识一起发送给 memcached。如果该item存放在memcached中的唯一标识与您提供的一致，您的写操作将会成功。如果另一个进程在这期间也修改了这个item，那么该item存放在memcached中的唯一标识将会改变，您的写操作就会失败。