**Memcached**

一个高性能的分布式内存对象缓存系统，用于动态Web应用以减轻数据库负载。在内存中缓存数据和对象，为动态、数据库驱动网站提供更快的运行速度。



**基本特征**

使用自己的页块分配器

使用基于存储“键-值”对的hashmap哈希表。

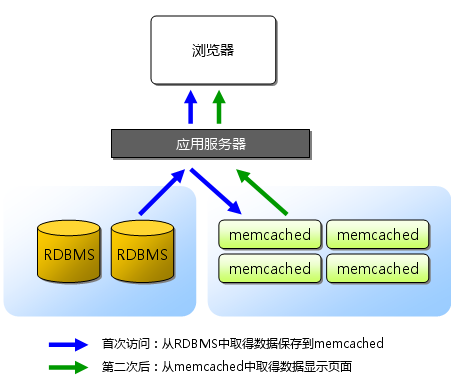
虚拟内存不会产生碎片，虚拟内存分配的时间复杂度可以保证为O(1)

**数据的逻辑结构**

memcached中存放两种形式的数据   
 (1) result of SQL query；

(2) 普通变量(variable)。   
 两种数据的key的组合方式不同，由get\_key进行判断和完成。

**Memcached运行原理**

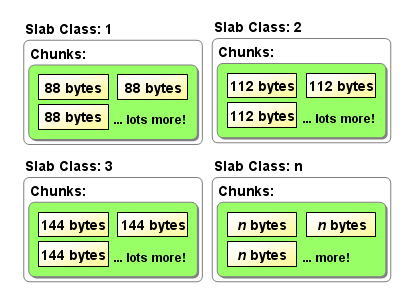
****

**Memcached内存分配**

目前memcached默认情况下采用Slab Allocator机制分配、管理内存。

Chunk——按照预先规定的大小，将分配的内存分割成各种特定长度的块。

slab class——尺寸相同的块分成组（chunk的集合）。

****

**Memcached数据传输**

**使用二进制协议，无需文本协议的解析处理**

**Memcached协议包.emf**

**Memcached的分布式处理**

Memcached的标准的分布式方法对键的存储根据服务器台数的余数进行分散：

1）求得键的整数哈希值；

2）除以服务器台数，根据其余数来选择服务器。

3）当选择的服务器无法连接时，rehash——将连接次数添加到键之后再次计算哈希值并尝试连接。

优点：方法简单，数据的分散性一般较好。

缺点：当添加或移除服务器时，缓存重组的代价大。

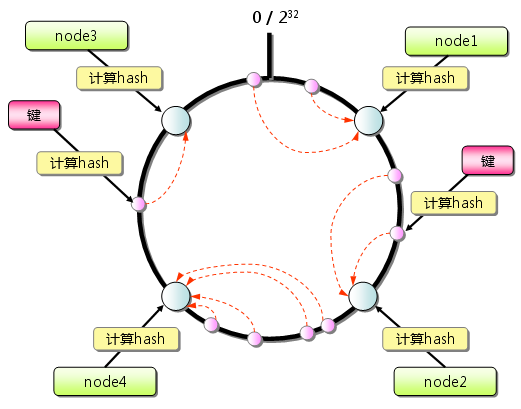
**改进的分布式方法——Consistent Hashing：**

1）求出服务器节点的哈希值， 将其配置到0～232的圆上；

2）用同样的方法求出存储数据的键的哈希值并映射到圆上；

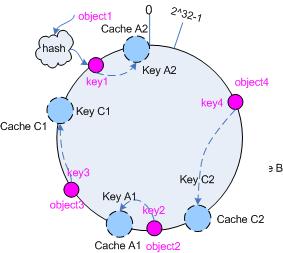
3）从数据映射到的位置开始顺时针查找，将数据保存到找到的第一台服务器上；

4）如果超过232仍然找不到服务器，就保存到第一台服务器上。

****

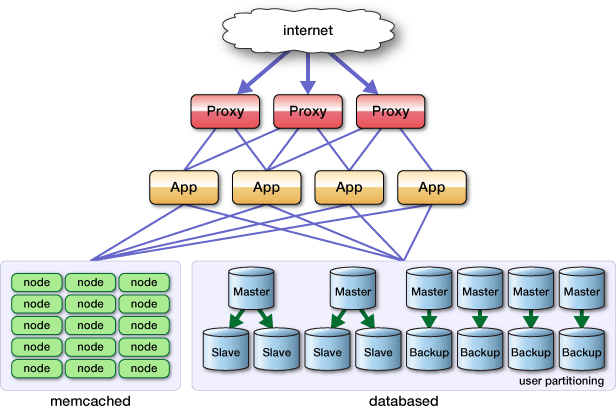
使用一般的hash函数，服务器的映射地点的分布可能出现不均匀的情况。

使用虚拟节点：

**为每个物理节点（服务器）在圆环上分配100～200个点，从而抑制分布不均匀，最大限度地减小服务器增减时的缓存重**

**Memcached架构举例**

200台左右的memcached服务器, 每台服务器的容量为3GB，则系统就有了将近600GB的巨大的内存数据库。

****