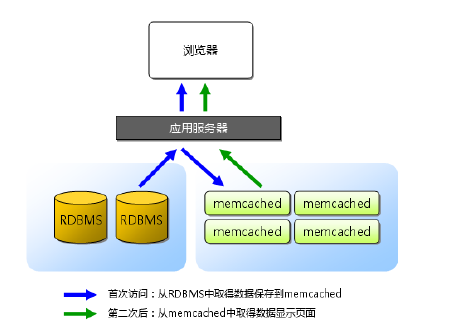
# memcached的基础

Memcached是高性能的分布式内存缓存服务器。通过缓存数据查询的结果，减少数据库的访问次数，以提高动态web应用的速度，提高可扩展性。

Memcache自身的设计就是分布式的，可以加入多个memcache服务器响应客户端的请求。他们之间在memcache的分布式算法下协同合作（不会将一份数据同时存储在多处）



1.2memcached的特征

协议简单：服务端和客户端基于文本行的协议

set foo 0 0 3

bar

STORED

get foo

VALUE foo 0 3

bar

END

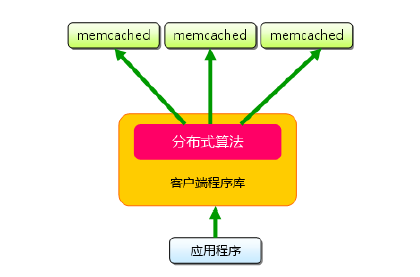
基于libevent的事件处理

memcached实现了一套内存存储方式。

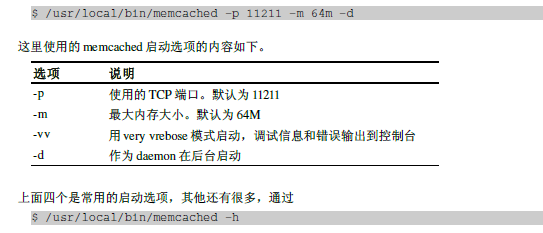
内置内存存储方式：因此重启memcacahed、重启操作系统会导致全部数据消失，而且当内存容量达到一定值后，会基于LRU算法（近期最少使用）自动删除不使用的缓存。Memcached本身是为缓存而设计的服务器，因此并没有考虑过多的数据的永久性问题。

Memcached的服务器端不互相通信，它实现分布式;

memcached的服务端并没有分布式的功能，而且各个服务器端之间是不互相通信的，分布式是通过memcached的客户端实现的，他将数据分别存储在不同的与他相连的服务器上。



linux下memcached启动选项的内容如下：



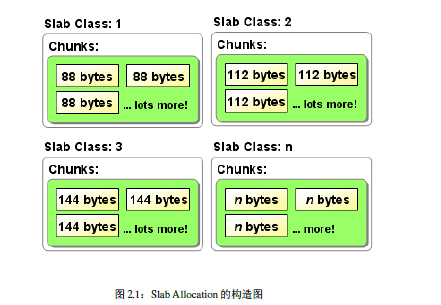
复杂数据序列化后在保存，因此散列、数组和对象等都可以直接保存到memcached中。

一些扩展如php-memcache会自动将数组等复杂数据在存储之前进行扩展，然后get的时候在反序列化，他们存储到memcache中的值实际上是序列化后的。

保存数据的方法：

1. add 仅当存储空间中不存在键相同的数据时才保存
2. Replace 仅当存储空间中存在键相同的数据是才保存
3. Set 与add和replace不同，无论何时都保存
4. 理解memcached的内存存储

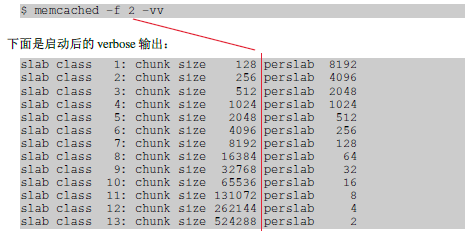
Memcached默认采用了slab allocator的内存分配机制。它的基本原理是按照预先规定的大小将分配的内存分割成特定长度的块，并把尺寸相同的块分成组。以解决内存碎片问题。



尽管它减少了内存碎片问题，但是他导致了内存的浪费问题，如100字节的数据存储到128字节的chunk中，剩余的28字节就是浪费。解决办法是根据存储数据的大小设定chunk的大小，但是chunk是运行开始就设定好的，因此这中办法并不能解决。

2.4 使用Growth Factor进行调优

Memcached在启动时指定Growth Factor因此来控制不同slab之间的差异。



部署时最好是预计存储数据的大小来调整growth factor以获得最恰当的设置。

2.5 查看memcached的内部状态

stats

STAT pid 4068

STAT uptime 3054540019

STAT time 236786705

STAT version 1.4.4-14-g9c660c0

STAT pointer\_size 64

STAT curr\_connections 10

STAT total\_connections 11

STAT connection\_structures 11

STAT cmd\_get 2

STAT cmd\_set 1

STAT cmd\_flush 0

STAT get\_hits 2

STAT get\_misses 0

STAT delete\_misses 0

STAT delete\_hits 0

STAT incr\_misses 0

STAT incr\_hits 0

STAT decr\_misses 0

STAT decr\_hits 0

STAT cas\_misses 0

STAT cas\_hits 0

STAT cas\_badval 0

STAT auth\_cmds 0

STAT auth\_errors 0

STAT bytes\_read 265

STAT bytes\_written 169

STAT limit\_maxbytes 67108864

STAT accepting\_conns 1

STAT listen\_disabled\_num 0

STAT threads 4

STAT conn\_yields 0

STAT bytes 71

STAT curr\_items 1

STAT total\_items 1

STAT evictions 0

END

而且通过输入stats slabs还可以获得关于缓存记录的信息;;

stats slabs

STAT 1:chunk\_size 96

STAT 1:chunks\_per\_page 10922

STAT 1:total\_pages 1

STAT 1:total\_chunks 10922

STAT 1:used\_chunks 1

STAT 1:free\_chunks 0

STAT 1:free\_chunks\_end 10921

STAT 1:mem\_requested 71

STAT 1:get\_hits 2

STAT 1:cmd\_set 1

STAT 1:delete\_hits 0

STAT 1:incr\_hits 0

STAT 1:decr\_hits 0

STAT 1:cas\_hits 0

STAT 1:cas\_badval 0

STAT active\_slabs 1

STAT total\_malloced 1048512

END

完全的命令参考可以参照memcached软件包中的protocol.txt文档。

1. memcached 的删除机制和发展方向

3.1

数据不会真正从memcached中消失，记录超时后客户端就无法看见该记录，其存储空间即可重复使用。

Lazy Expiration:memcached内部不会监视记录是否过期，而是在get时查看记录的时间戳，检查记录是否过期。

3.2 LRU：从缓存中有效删除数据的原理，删除最近最少使用，将其空间分配给新的数据。

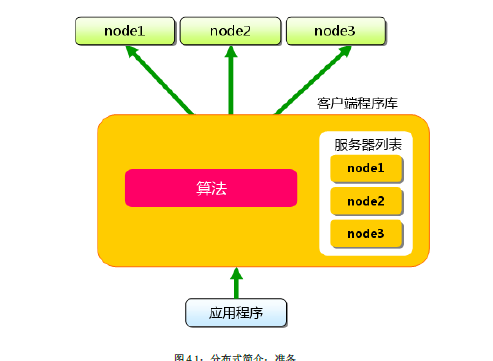
-M启动参数可以禁用LRU

Memcached不是存储器而是缓存

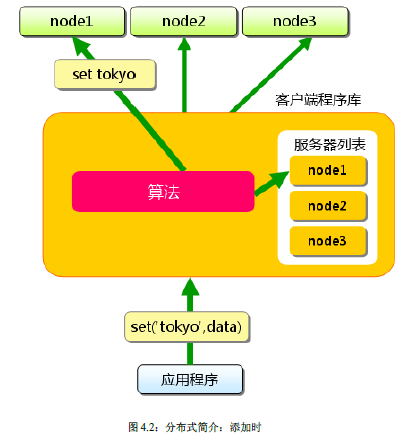
1. memcached的分布式算法

4.1 memcached的分布式

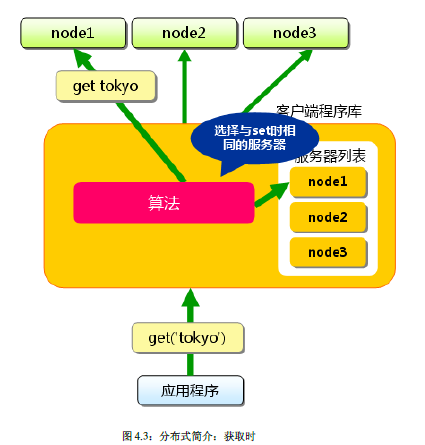
Memcached的分布式是完全有客户端实现的。



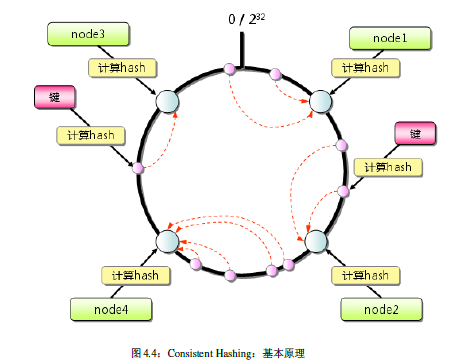
假设要保存“tokyo”、“kanagawa”、“chiba”、“saitama”、“gunma”数据，首先向memcached中添加tokyo，将tokyo传给客户端程序库后，客户端实现的算法就会根据键来决定保存数据库的服务器。

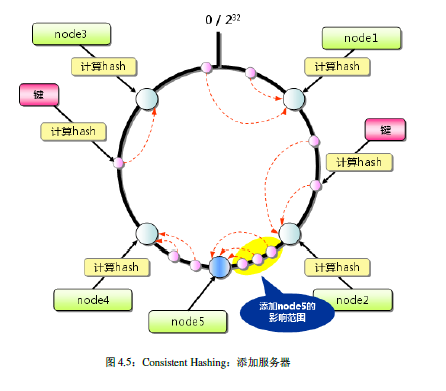


获取时也是根据相同的算法找到存储改键的数据库获取数据。



4.3 consistent hashing 一致性哈希





使用一致性哈希后，在添加新的memcached节点时，只有这个新的节点和逆时针临近节点之间的数据会受到影响，其他的数据还是正常的。

当然了还有更进一步的优化方案。