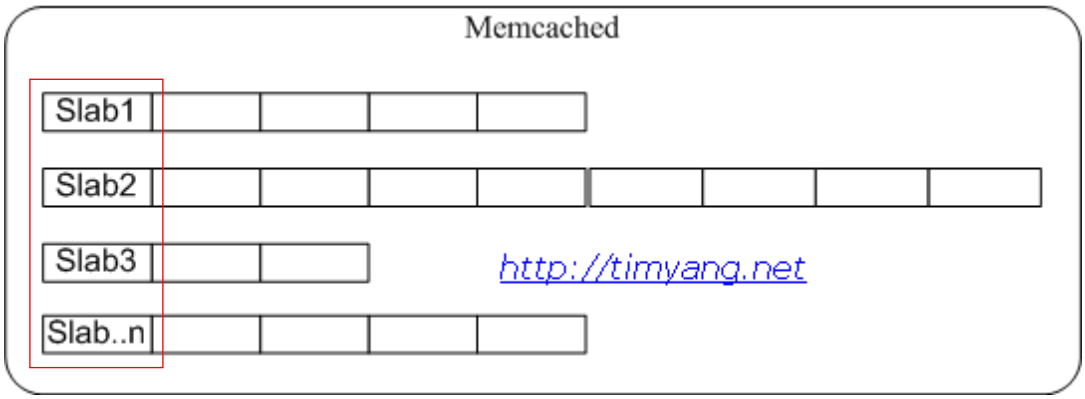
Memcached数据被踢(evictions>0)现象分析

Monday, Sep 7th, 2009 by Tim | Tags: evictions, LRU, memcached

很多同学可能熟知Memcached的LRU淘汰算法，它是在slab内部进行的，如果所有空间都被slabs分配，即使另外一个slab里面有空位，仍然存在踢数据可能。你可以把slab理解为教室，如果你的教室满了，即使别的教室有空位你的教室也只能踢人才能进人。

例子举得很赞！！！



本文介绍的却是另外一种现象。今天监控发现线上一memcached发生数据被踢现象，用stats命令看evictions>0,因为以前也出现过此问题，后来对这个参数增加了一个监控，所以这次主动就发现了。由于给memcached分配的内存远大于业务存储数据所需内存，因此初步判断是“灵异现象”。

第一步，netstat查看所有连接，排除是否被一些未规划的client使用，经排查后断定无此可能。

第二步，用tcpdump抽样检查set的指令，排除是否有忘记设cache过期时间的client，初步检查所有典型的业务都有expire time。

第三步，Google，未果

第四步，看源代码，了解evictions计数器增加时的具体细节，oh, no…

in items.c, memcached-1.2.8,

```
125           for (search = tails[id]; tries > 0 && search != NULL;
tries--, search=search->prev) {
126               if (search->refcount == 0) {
127                   if (search->exptime == 0 || search->exptime >
current_time) {
128                       itemstats[id].evicted++;
129                       itemstats[id].evicted_time = current_time -
search->time;
130                       STATS_LOCK();
131                       stats.evictions++;
132                       STATS_UNLOCK();
133                   }
134                   do_item_unlink(search);
135                   break;
```

```
136                 }
137             }
```

从源代码发现踢数据只判断一个条件，if (search->refcount == 0)，这个refcount是多线程版本计数用，在当前服务器未启用多线程情况下，refcount应该始终为0,因此初步判断memcached是从访问队列尾部直接踢数据。

为了证实想法，设计以下场景：

1. 部署一个memcached测试环境，分配比较小的内存，比如8M
2. 设置1条永远不过期的数据到memcached中，然后再get一次，这条数据后续应该存在LRU队尾。
3. 每隔1S向memcached set(并get一次) 1,000条数据，过期时间设为3秒。
4. 一段时间后，stats命令显示evictions=1

按我以前的理解，第2步的数据是永远不会被踢的，因为有足够过期的数据空间可以给新来的数据用，LRU淘汰算法应该跳过没过期的数据，但结果证实这种判断是错误的。以上业务的服务器发生被踢的现象是由于保存了大量存活期短的key/value,且key是不重复的。另外又有一业务保存了小量不过期的数据，因此导致不过期的数据惨遭被挤到队列踢出。

本来这个问题就告一段落了，但在写完这篇文章后，顺便又看了新一代memcached 1.4.1的源代码，很惊喜发现以下代码被增加。

items.c, memcached 1.4.1

```
107     /* do a quick check if we have any expired items in the tail.. */
108     int tries = 50;
109     item *search;
110
111     for (search = tails[id];
112          tries > 0 && search != NULL;
113          tries--, search=search->prev) {
114         if (search->refcount == 0 &&
115             (search->exptime != 0 && search->exptime < current_time)) {
116             it = search;
117             /* I don't want to actually free the object, just steal
118              * the item to avoid to grab the slab mutex twice ;-)
119              */
120             it->refcount = 1;
121             do_item_unlink(it);
122             /* Initialize the item block: */
123             it->slabs_clsid = 0;
124             it->refcount = 0;
125             break;
126         }
127     }
```

重复进行上述测试，未发生evictions。

9/8 Update: 注意到L108的tries=50没有？试想把测试第2步设置51条不过期数据到cache

中，情况会怎样？因此新版的Memcached也同样存在本文描述问题。

几条总结：

- 过期的数据如果没被显式调用get，则也要占用空间。
- 过期的不要和不过期的数据存在一起，否则不过期的可能被踢。
- 从节约内存的角度考虑，即使数据会过期，也不要轻易使用随机字符串作为key，尽量使用定值如uid，这样占用空间的大小相对固定。
- 估算空间大小时请用slab size计算，不要按value长度去计算。
- 不要把cache当作更快的key value store来用，cache不是storage。

« | »

37 Comments »

xLight 09-09-08 12:24

业务

如果每个应用都有自己独立的memcache集群就不会触发这个问题。
不过运维复杂了。

sky 09-09-09 16:36

晕。memcached文档中都有写，lazy expiration logic。expires与LRU无关。

dennis 09-10-14 15:06

事实上你后面的补充是错误的吧，哪怕把测试第2步设置51条不过期数据到cache中，也不会发生数据被踢现象的，测试证明。原因在于do_item_unlink(it);
调用的条件原来只判断refcount，现在还加上了(search->exptime != 0 && search->exptime < current_time)，变成了同一个if语句，而非原来的两个。

dennis 09-10-14 15:31

咳咳，原因说错了，是因为没有统计
stats.evictions++;
才对。

Tom 10-08-19 17:16

请问下楼主有没有遇到过，memcache 的连接不断开的，就是请求完数据，连接不断开，导致httpd进程不断增多

Liexusong 11-02-20 13:28

楼上的原因是因为客户端没有调用quit命令吧。看过memcached的源码好像没有自动断开连接的。

hills 12-05-18 16:44

是啊 我就设置了永不过期， 一天以后就失效了！ 原来是这样子！

pesiwang 12-09-29 19:20

memcached-1.4.15 变成了只检查一个了，注意最后边的break

```
/* We walk up *only* for locked items. Never searching for expired.
* Waste of CPU for almost all deployments */
for (; tries > 0 && search != NULL; tries–, search=search->prev) {
uint32_t hv = hash(ITEM_key(search), search->nkey, 0);
…
break;
}
```

所以不过期的数据还是不要跟过期数据放一起好

simon 14-04-18 10:25

这个refcount是多线程版本计数用，在当前服务器未启用多线程情况下，refcount应该始终为0。这个理解有问题吧！！！
refcount表示引用计数，即使是单线程也不可能永远为0的。