[网站缓存技术总结（ ehcache memcache redis）](http://www.xuetimes.com/archives/325)

[知识点](http://www.xuetimes.com/archives/category/%e7%bc%96%e7%a8%8b%e4%b9%90%e5%9b%ad/web%e5%bc%80%e5%8f%91/%e7%9f%a5%e8%af%86%e7%82%b9) [water](http://www.xuetimes.com/archives/author/lipeng) 2年前 (2014-12-04) 8598℃ [0评论](http://www.xuetimes.com/archives/325#comments)

网站技术高速发展的今天，缓存技术已经成为大型网站的一个关键技术，缓存设计好坏直接关系的一个网站访问的速度，以及购置服务器的数量，甚至影响到用户的体验。

网站缓存按照存放的地点不同，可以分为客户端缓存、服务端缓存。  


**客户端缓存**  
客户端缓存又可分为：浏览器缓存、网关或代理服务器缓存

网关或代理服务器缓存是将网页缓存中网关服务器上，多用户访问同一个页面时，将直接从网关服务器把页面传送给用户。

浏览器缓存是最靠近用户的缓存，如果启用缓存，用户在访问同一个页面时，将不再从服务器下载页面，而是从本机的缓存目录中读取页面，然后再浏览器中展现这个页面。

浏览器缓存的控制，可以设置meta标签，可以设置数字，也可以设置时间，如下：

<Meta http-equiv=”Expires” Content=”3600″>

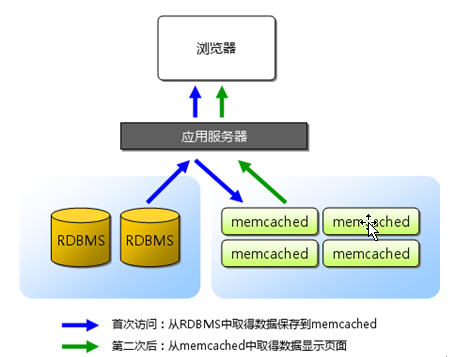
<Meta http-equiv=”Expires” Content=”Wed, 26 Feb 1997 08:21:57 GMT”>

HTTP头信息如下：

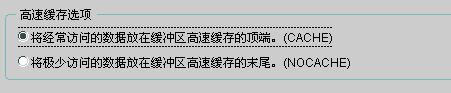
HTTP/1.1 200 OK  
Date: Fri, 30 Oct 1998 13:19:41 GMT  
Server: Apache/1.3.3 (Unix)  
Cache-Control: max-age=3600, must-revalidate  
Expires: Fri, 30 Oct 1998 14:19:41 GMT  
Last-Modified: Mon, 29 Jun 1998 02:28:12 GMT

不过现在的网站为了保证用户访问到最新的内容，一般很少采用浏览器缓存，取而代之的是更加灵活的服务器缓存。  
**服务端缓存**  
服务端缓存分为：页面缓存、数据缓存、数据库缓存  
**1、页面缓存**  
页面缓存是将动态页面直接生成静态的页面放在服务器端，用户调取相同页面时，静态页面将直接下载到客户端，不再需要通过程序的运行和数据库的访问，大大节约了服务器的负载。  
早期的网站很多使用发布系统来完成这个功能，在后台发布时将数据和页面模板整合成静态页面，存放在硬盘中。但这样的缺陷很明显，一是后台的程序的编写很 复杂，二是缓存的控制只能通过人为的方式来控制，这对一些更新十分频繁的网站就是一个噩梦，网站可能在不停的做缓存的删除和重建。当然后来出现了一些自动 更新这些缓存的框架，比如PHP的SMARTY模板技术，可以定义缓存过期的时间，自动去更新这些缓存。这对一些信息发布类网站已经确实适用了。  
除了整个页面的缓存技术，还有一种技术叫做“网页片段缓存技术”，将页面的部分而不是全部进行缓存。代表作有ESI cache。

**2、数据缓存**  
但是当WEB2.0兴起的今天，信息的发布已经不再是管理员统一发布的了，而是所有的用户都在发布信息，用户发布完信息后当然是想看到这些信息，而不是等到缓存时间到刷新后才看到这些数据，于是数据缓存的相关技术也就应运而生了。  
比较有名的数据缓存框架有ehcache和 memcached。  
ehcache有很多缓存的分支（包括页面缓存的模块），但最核心的模块还是它的数据缓存部分，比如，当ehcache和hibernate进行整合 时，能将查询出的对象集合放入内存中，下次如果再查询这个查询，将直接从内存中返回这个数据集合，不需要再进行数据库的查询，同时，你可以配置缓存的刷新 模式，有read-only，nonstrict-read-write，read-write 几种模式，其中read-only表示缓存是不刷新的（要刷新就只有重启了），nonstrict-read-write表示刷新是不及时的，你可以设置 超时的时间去刷新，read-write表示在数据发生变化时缓存都会发生刷新，具体怎么配置可能就要根据具体业务了。  
Memcached大 致的原理也和ehcache 相同，将数据采用键值的形式存放在内存中，使用时可以将查询的md5作为键，查询的结果作为值。相对ehcache而言，memcached是一个工 具，ehcache是一个框架，memcached更加底层更加灵活，当然你也要写相应的代码去使用它。  
这是一张网上的memcached图，说明了memcached在系统中的位置。

  
近几年兴起的NOSQL技术，虽然现在归于数据库的一种，但其本质也是缓存技术和数据库技术的一种融合产物。

目前缓存的做法分为两种模式:  
内存缓存：缓存数据存放在服务器的内存空间中。  
优点：速度快 缺点：资源有限

文件缓存：缓存数据存放在服务器的硬盘空间中。  
优点：容量大 缺点：速度偏慢，尤其在缓存数量巨大时  
**数据库缓存**  
数据库的缓存一般由数据库提供，比如ORACLE，可以对表建立高速缓存，提高对经常访问的数据的访问速度。  


**总结**  
究竟怎样去使用缓存，使用哪一层次的缓存，是由网站本身的具体业务来决定的。缓存技术的一个原则是：让数据更靠近用户。缓存技术是一门博大精深的艺术，我也是只知些皮毛。

**最近项目组有用到这三个缓存，去各自的官方看了下,觉得还真的各有千秋！今天特意归纳下各个缓存的优缺点，仅供参考！**

**Ehcache**

在java项目广泛的使用。它是一个开源的、设计于提高在数据从RDBMS中取出来的高花费、高延迟采取的一种缓存方案。正因为Ehcache具有健壮性（基于java开发）、被认证（具有apache 2.0  license）、充满特色（稍后会详细介绍），所以被用于大型复杂分布式web application的各个节点中。

什么特色？

1.  够快

Ehcache的发行有一段时长了，经过几年的努力和不计其数的性能测试，Ehcache终被设计于large, high concurrency systems.

2. 够简单

开发者提供的接口非常简单明了，从Ehcache的搭建到运用运行仅仅需要的是你宝贵的几分钟。其实很多开发者都不知道自己用在用Ehcache，Ehcache被广泛的运用于其他的开源项目

比如：hibernate

3.够袖珍

关于这点的特性，官方给了一个很可爱的名字small foot print ，一般Ehcache的发布版本不会到2M，V 2.2.3  才 668KB。

4. 够轻量

核心程序仅仅依赖slf4j这一个包，没有之一！

5.好扩展

Ehcache提供了对大数据的内存和硬盘的存储，最近版本允许多实例、保存对象高灵活性、提供LRU、LFU、FIFO淘汰算法，基础属性支持热配置、支持的插件多

6.监听器

缓存管理器监听器 （CacheManagerListener）和 缓存监听器（CacheEvenListener）,做一些统计或数据一致性广播挺好用的

如何使用？

够简单就是Ehcache的一大特色，自然用起来just so easy!

贴一段基本使用代码

CacheManager manager = CacheManager.newInstance("src/config/ehcache.xml");

Ehcache cache = new Cache("testCache", 5000, false, false, 5, 2);

cacheManager.addCache(cache);

代码中有个ehcache.xml文件，现在来介绍一下这个文件中的一些属性

1. name:缓存名称。
2. maxElementsInMemory：缓存最大个数。
3. eternal:对象是否永久有效，一但设置了，timeout将不起作用。
4. timeToIdleSeconds：设置对象在失效前的允许闲置时间（单位：秒）。仅当eternal=false对象不是永久有效时使用，可选属性，默认值是0，也就是可闲置时间无穷大。
5. timeToLiveSeconds：设置对象在失效前允许存活时间,最大时间介于创建时间和失效时间之间。仅当eternal=false对象不是永久有效时使用，默认是0.，也就是对象存活时 间无穷大。
6. overflowToDisk：当内存中对象数量达到maxElementsInMemory时，Ehcache将会对象写到磁盘中。
7. diskSpoolBufferSizeMB：这个参数设置DiskStore（磁盘缓存）的缓存区大小。默认是30MB。每个Cache都应该有自己的一个缓冲区。
8. maxElementsOnDisk：硬盘最大缓存个数。
9. diskPersistent：是否缓存虚拟机重启期数据 Whether the disk store persists between restarts of the Virtual Machine. The default value is false.
10. diskExpiryThreadIntervalSeconds：磁盘失效线程运行时间间隔，默认是120秒。
11. memoryStoreEvictionPolicy：当达到maxElementsInMemory限制时，Ehcache将会根据指定的策略去清理内存。默认策略是LRU。你可以设置为 FIFO或是LFU。
12. clearOnFlush：内存数量最大时是否清除。

**memcache**

memcache 是一种高性能、分布式对象缓存系统，最初设计于缓解动态网站数据库加载数据的延迟性，你可以把它想象成一个大的内存HashTable，就是一个key-value键值缓存。Danga Interactive为了LiveJournal所发展的，以BSD license释放的一套开放源代码软件。

1.依赖

memcache C语言所编写，依赖于最近版本的GCC和libevent。GCC是它的编译器，同事基于libevent做socket io。在安装memcache时保证你的系统同事具备有这两个环境。

2.多线程支持

memcache支持多个cpu同时工作，在memcache安装文件下有个叫threads.txt中特别说明，By default, memcached is compiled as a single-threaded application.默认是单线程编译安装，如果你需要多线程则需要修改./configure –enable-threads，为了支持多核系统，前提是你的系统必须具有多线程工作模式。开启多线程工作的线程数默认是4，如果线程数超过cpu数容易发生操作死锁的概率。结合自己业务模式选择才能做到物尽其用。

3.高性能

通过libevent完成socket 的通讯，理论上性能的瓶颈落在网卡上。

简单安装：

1.分别把memcached和libevent下载回来，放到 /tmp 目录下：

# cd /tmp

# wget http://www.danga.com/memcached/dist/memcached-1.2.0.tar.gz

# wget http://www.monkey.org/~provos/libevent-1.2.tar.gz

http://img.my.csdn.net/uploads/201301/17/1358386757_5640.jpg

2.先安装libevent：

# tar zxvf libevent-1.2.tar.gz

# cd libevent-1.2

# ./configure -prefix=/usr

# make （如果遇到提示gcc 没有安装则先安装gcc)

# make install

3.测试libevent是否安装成功：

# ls -al /usr/lib | grep libevent

lrwxrwxrwx 1 root root 21 11?? 12 17:38 libevent-1.2.so.1 -> libevent-1.2.so.1.0.3

-rwxr-xr-x 1 root root 263546 11?? 12 17:38 libevent-1.2.so.1.0.3

-rw-r-r- 1 root root 454156 11?? 12 17:38 libevent.a

-rwxr-xr-x 1 root root 811 11?? 12 17:38 libevent.la

lrwxrwxrwx 1 root root 21 11?? 12 17:38 libevent.so -> libevent-1.2.so.1.0.3

还不错，都安装上了。

4.安装memcached，同时需要安装中指定libevent的安装位置：

# cd /tmp

# tar zxvf memcached-1.2.0.tar.gz

# cd memcached-1.2.0

# ./configure -with-libevent=/usr

# make

# make install

如果中间出现报错，请仔细检查错误信息，按照错误信息来配置或者增加相应的库或者路径。

安装完成后会把memcached放到 /usr/local/bin/memcached ，

5.测试是否成功安装memcached：

# ls -al /usr/local/bin/mem\*

-rwxr-xr-x 1 root root 137986 11?? 12 17:39 /usr/local/bin/memcached

-rwxr-xr-x 1 root root 140179 11?? 12 17:39 /usr/local/bin/memcached-debug

启动memcache服务

启动Memcached服务：

1.启动Memcache的服务器端：

# /usr/local/bin/memcached -d -m 8096 -u root -l 192.168.77.105 -p 12000 -c 256 -P /tmp/memcached.pid

-d选项是启动一个守护进程，

-m是分配给Memcache使用的内存数量，单位是MB，我这里是8096MB，

-u是运行Memcache的用户，我这里是root，

-l是监听的服务器IP地址，如果有多个地址的话，我这里指定了服务器的IP地址192.168.77.105，

-p是设置Memcache监听的端口，我这里设置了12000，最好是1024以上的端口，

-c选项是最大运行的并发连接数，默认是1024，我这里设置了256，按照你服务器的负载量来设定，

-P是设置保存Memcache的pid文件，我这里是保存在 /tmp/memcached.pid，

2.如果要结束Memcache进程，执行：

# cat /tmp/memcached.pid 或者 ps -aux | grep memcache   （找到对应的进程id号）

# kill 进程id号

也可以启动多个守护进程，不过端口不能重复。

memcache 的连接

telnet  ip   port

注意连接之前需要再memcache服务端把memcache的防火墙规则加上

-A RH-Firewall-1-INPUT -m state –state NEW -m tcp -p tcp –dport 3306 -j ACCEPT

重新加载防火墙规则

service iptables restart

OK ,现在应该就可以连上memcache了

在客户端输入stats 查看memcache的状态信息



pid              memcache服务器的进程ID

uptime      服务器已经运行的秒数

time           服务器当前的unix时间戳

version     memcache版本

pointer\_size         当前操作系统的指针大小（32位系统一般是32bit）

rusage\_user          进程的累计用户时间

rusage\_system    进程的累计系统时间

curr\_items            服务器当前存储的items数量

total\_items           从服务器启动以后存储的items总数量

bytes                       当前服务器存储items占用的字节数

curr\_connections        当前打开着的连接数

total\_connections        从服务器启动以后曾经打开过的连接数

connection\_structures          服务器分配的连接构造数

cmd\_get get命令          （获取）总请求次数

cmd\_set set命令          （保存）总请求次数

get\_hits          总命中次数

get\_misses        总未命中次数

evictions     为获取空闲内存而删除的items数（分配给memcache的空间用满后需要删除旧的items来得到空间分配给新的items）

bytes\_read    读取字节数（请求字节数）

bytes\_written     总发送字节数（结果字节数）

limit\_maxbytes     分配给memcache的内存大小（字节）

threads         当前线程数

**redis**

redis是在memcache之后编写的，大家经常把这两者做比较，如果说它是个key-value store 的话但是它具有丰富的数据类型，我想暂时把它叫做缓存数据流中心，就像现在物流中心那样，order、package、store、classification、distribute、end。现在还很流行的LAMP PHP架构 不知道和 redis+mysql 或者 redis + mongodb的性能比较（听群里的人说mongodb分片不稳定）。

先说说reidis的特性

1. 支持持久化

redis的本地持久化支持两种方式：RDB和AOF。RDB 在redis.conf配置文件里配置持久化触发器，AOF指的是redis没增加一条记录都会保存到持久化文件中（保存的是这条记录的生成命令），如果不是用redis做DB用的话还会不要开AOF ，数据太庞大了，重启恢复的时候是一个巨大的工程！

2.丰富的数据类型

redis 支持 String 、Lists、sets、sorted sets、hashes 多种数据类型,新浪微博会使用redis做nosql主要也是它具有这些类型，时间排序、职能排序、我的微博、发给我的这些功能List 和 sorted set

的强大操作功能息息相关

3.高性能

这点跟memcache很想象，内存操作的级别是毫秒级的比硬盘操作秒级操作自然高效不少，较少了磁头寻道、数据读取、页面交换这些高开销的操作！这也是NOSQL冒出来的原因吧，应该是高性能

是基于RDBMS的衍生产品，虽然RDBMS也具有缓存结构，但是始终在app层面不是我们想要的那么操控的。

4.replication

redis提供主从复制方案，跟mysql一样增量复制而且复制的实现都很相似，这个复制跟AOF有点类似复制的是新增记录命令，主库新增记录将新增脚本发送给从库，从库根据脚本生成记录，这个过程非常快，就看网络了，一般主从都是在同一个局域网，所以可以说redis的主从近似及时同步，同事它还支持一主多从，动态添加从库，从库数量没有限制。 主从库搭建，我觉得还是采用网状模式，如果使用链式（master-slave-slave-slave-slave·····）如果第一个slave出现宕机重启，首先从master 接收 数据恢复脚本，这个是阻塞的，如果主库数据几TB的情况恢复过程得花上一段时间，在这个过程中其他的slave就无法和主库同步了。

5.更新快

这点好像从我接触到redis到目前为止 已经发了大版本就4个，小版本没算过。redis作者是个非常积极的人，无论是邮件提问还是论坛发帖，他都能及时耐心的为你解答，维护度很高。有人维护的话，让我们用的也省心和放心。目前作者对redis 的主导开发方向是redis的集群方向。

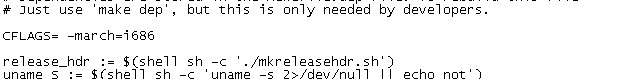
redis的安装

redis的安装其实还是挺简单的，总的来说就三步：下载tar包，解压tar包，安装。

不过最近我在2.6.7后用centos 5.5 32bit 时碰到一个安装问题，下面我就用图片分享下安装过程碰到的问题，在redis 文件夹内执行make时有个如下的错 undefined reference to ‘\_\_sync\_add\_and\_fetch\_4′



上网找了了好多最后在  https://github.com/antirez/redis/issues/736 找到解决方案，write CFLAGS= -march=i686 on src/Makefile head!

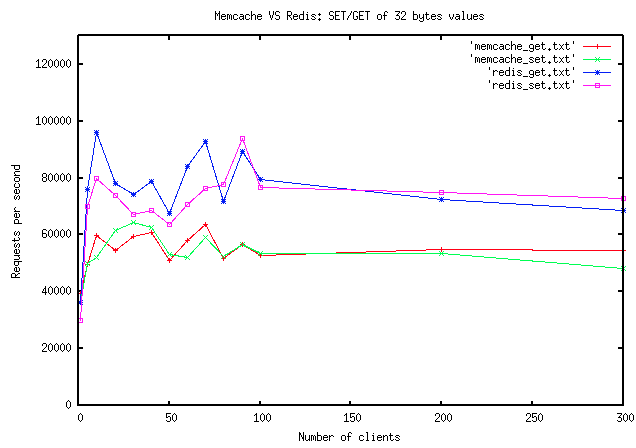


记得要把刚安装失败的文件删除，重新解压新的安装文件，修改Makefile文件，再make安装。就不会发现原来那个错误了

关于redis的一些属性注释和基本类型操作在上一篇redis 的开胃菜有详细的说明，这里就不再重复累赘了（实质是想偷懒 ，哈哈！）

最后，把memcache和redis放在一起不得不会让人想到两者的比较，谁快谁好用啊，群里面已经为这个事打架很久了，我就把我看到的在这里跟大家分享下。

在别人发了一个memcache性能比redis好很多后，redis 作者 antirez 发表了一篇博文，主要是说到如何给redis 和 memcache 做压力测试，文中讲到有个人说许多开源软件都应该丢进厕所，因为他们的压力测试脚本太2了，作者对这个说明了一番。redis  vs  memcache is  definitely an apple to apple comparison。 呵呵，很明确吧，两者的比较是不是有点鸡蛋挑骨头的效果，作者在相同的运行环境做了三次测试取多好的值，得到的结果如下图:



需要申明的是此次测试在单核心处理的过程的数据，memcache是支持多核心多线程操作的（默认没开）所以在默认情况下上图具有参考意义，若然则memcache快于redis。那为什么redis不支持多线程多核心处理呢？作者也发表了一下自己的看法，首先是多线程不变于bug的修复，其实是不易软件的扩展，还有数据一致性问题因为redis所有的操作都是原子操作，作者用到一个词nightmare 噩梦，呵呵！  当然不支持多线程操作，肯定也有他的弊端的比如性能想必必然差，作者从2.2版本后专注redis cluster的方向开发来缓解其性能上的弊端，说白了就是纵向不行，横向提高。

转载请注明：[学时网](http://www.xuetimes.com/) » [网站缓存技术总结（ ehcache memcache redis）](http://www.xuetimes.com/archives/325)