ehcache memcache redis 三大緩存男高音

blog.csdn.net/jationxiaozi/article/details/8509732

最近項目組有用到這三個緩存，去各自的官方看了下,覺得還真的各有千秋！今天特意歸納下各個緩存的優缺點，僅供參考！

Ehcache

在Java項目廣泛的使用。它是一個開源的、設計於提高在數據從RDBMS中取出來的高花費、高延遲採取的一種緩存方案。正因為Ehcache具有健壯性（基於java開發）、被認證（具有apache 2.0 license）、充滿特色（稍後會詳細介紹），所以被用於大型複雜分佈式web application的各個節點中。

什麼特色？

1. 夠快

Ehcache的發行有一段時長了，經過幾年的努力和不計其數的性能測試，Ehcache終被設計於large, high concurrency systems.

2. 夠簡單

開發者提供的接口非常簡單明了，從Ehcache的搭建到運用運行僅僅需要的是你寶貴的幾分鐘。其實很多開發者都不知道自己用在用Ehcache，Ehcache被廣泛的運用於其他的開源項目

比如：hibernate

3.夠袖珍

關於這點的特性，官方給了一個很可愛的名字small foot print ，一般Ehcache的發布版本不會到2M，V 2.2.3 才668KB。

4. 夠輕量

核心程序僅僅依賴slf4j這一個包，沒有之一！

5.好擴展

Ehcache提供了對大數據的內存和硬盤的存儲，最近版本允許多實例、保存對象高靈活性、提供LRU、LFU、FIFO淘汰算法，基礎屬性支持熱配置、支持的插件多

6.監聽器

緩存管理器監聽器 （CacheManagerListener）和 緩存監聽器（CacheEvenListener）,做一些統計或數據一致性廣播挺好用的

如何使用？

夠簡單就是Ehcache的一大特色，自然用起來just so easy!

貼一段基本使用代碼

```
CacheManager manager = CacheManager.newInstance("src/config/ehcache.xml");
Ehcache cache = new Cache("testCache", 5000, false, false, 5, 2);
cacheManager.addCache(cache);
```

代碼中有個`ehcache.xml`文件，現在來介紹一下這個文件中的一些屬性

1. name:緩存名稱。
2. maxElementsInMemory：緩存最大個數。
3. eternal:對象是否永久有效，一但設置了，timeout將不起作用。
4. timeToIdleSeconds：設置對象在失效前的允許閒置時間（單位：秒）。僅當eternal=false對象不是永久有效時使用，可選屬性，默認值是0，也就是可閒置時間無窮大。
5. timeToLiveSeconds：設置對象在失效前允許存活時間,最大時間介於創建時間和失效時間之間。僅當eternal=false對象不是永久有效時使用，默認是0.，也就是對象存活時 間無窮大。
6. overflowToDisk：當內存中對象數量達到maxElementsInMemory時，Ehcache將會對象寫到磁盤中。
7. diskSpoolBufferSizeMB：這個參數設置DiskStore（磁盤緩存）的緩存區大小。默認是30MB。每個Cache都應該有自己的一個緩衝區。
8. maxElementsOnDisk：硬盤最大緩存個數。
9. diskPersistent：是否緩存虛擬機重啟期數
據 Whether the disk store persists between restarts of the Virtual Machine. The default value is false.
10. diskExpiryThreadIntervalSeconds：磁盤失效線程運行時間間隔，默認是120秒。
11. memoryStoreEvictionPolicy：當達到maxElementsInMemory限制時，Ehcache將會根據指定的策略去清理內存。默認策略是LRU。你可以設置為 FIFO或是LFU。
12. clearOnFlush：內存數量最大時是否清除。

memcache

memcache 是一種高性能、分佈式對象緩存系統，最初設計於緩解動態網站數據庫加載數據的延遲性，你可以把它想像成一個大的內存HashTable，就是一個key-value鍵值緩存。Danga Interactive為了LiveJournal所發展的，以BSD license釋放的一套開放源代碼軟件。

1.依賴

memcache C語言所編寫，依賴於最近版本的GCC和libevent。GCC是它的編譯器，同事基於libevent做socket io。在安裝memcache時保證你的系統同事具備有這兩個環境。

2.多線程支持

memcache支持多個cpu同時工作，在memcache安裝文件下有個叫threads.txt中特別說明，By default, memcached is compiled as a single-threaded application.默認是單線程編譯安裝，如果你需要多線程則需要修改./configure --enable-threads，為了支持多核系統，前提是你的系統必須具有多線程工作模式。開啟多線程工作的線程數默認是4，如果線程數超過cpu數容易發生操作死鎖的概率。結合自己業務模式選擇才能做到物盡其用。

3.高性能

通過libevent完成socket 的通訊，理論上性能的瓶頸落在網卡上。

簡單安裝：

1.分別把memcached和libevent下載回來，放到 /tmp 目錄下：

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

```
-rw-r--r-- 1 root root 413811 Oct 16  2006 libevent-1.2.tar.gz
-rw-r--r-- 1 root root  40020 Jan 14 22:01 memcached-1.2.0.tar.gz
[root@localhost tool]# _
```

2.先安裝libevent：

tar zxvf libevent-1.2.tar.gz

cd libevent-1.2

./configure -prefix=/usr

make （如果遇到提示gcc 沒有安裝則先安裝gcc)

make install

3.測試libevent是否安裝成功：

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

還不錯，都安裝上了。

4.安裝memcached，同時需要安裝中指定libevent的安裝位置：

cd /tmp

tar zxvf memcached-1.2.0.tar.gz

cd memcached-1.2.0

./configure -with-libevent=/usr

make

make install

如果中間出現報錯，請仔細檢查錯誤信息，按照錯誤信息來配置或者增加相應的庫或者路徑。

安裝完成後會把memcached放到 /usr/local/bin/memcached ，

5.測試是否成功安裝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

啟動memcache服務

啟動Memcached服務：

1.啟動Memcache的服務器端：

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

-d選項是啟動一個守護進程，

-m是分配給Memcache使用的內存數量，單位是MB，我這裡是8096MB，

-u是運行Memcache的用戶，我這裡是root，

-l是監聽的服務器IP地址，如果有多個地址的話，我這裡指定了服務器的IP地址192.168.77.105，

-p是設置Memcache監聽的端口，我這裡設置了12000，最好是1024以上的端口，

-c選項是最大運行的並發連接數，默認是1024，我這裡設置了256，按照你服務器的負載量來設定，

-P是設置保存Memcache的pid文件，我這裡是保存在 /tmp/memcached.pid，

2.如果要結束Memcache進程，執行：

cat /tmp/memcached.pid 或者 ps -aux | grep memcache　（找到對應的進程id號）

kill 進程id號

也可以啟動多個守護進程，不過端口不能重複。

memcache 的連接

telnet ip port

注意連接之前需要再memcache服務端把memcache的防火牆規則加上

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

重新加載防火牆規則

service iptables restart

OK ,現在應該就可以連上memcache了

在客戶端輸入stats 查看memcache的狀態信息

```
stats
STAT pid 3349
STAT uptime 1318
STAT time 1358243023
STAT version 1.2.2
STAT pointer_size 32
STAT rusage_user 0.000999
STAT rusage_system 0.016997
STAT curr_items 0
STAT total_items 0
STAT bytes 0
STAT curr_connections 1
STAT total_connections 2
STAT connection_structures 2
STAT cmd_get 0
STAT cmd_set 0
STAT get_hits 0
STAT get_misses 0
STAT evictions 0
STAT bytes_read 65
STAT bytes_written 63
STAT limit_maxbytes 10485760
STAT threads 1
```

pid　　memcache服務器的進程ID

uptime　　服務器已經運行的秒數

time　　服務器當前的unix時間戳

version　　memcache版本

pointer_size　　當前操作系統的指針大小（32位系統一般是32bit）

rusage_user　　進程的累計用戶時間

rusage_system　　進程的累計系統時間

curr_items　　服務器當前存儲的items數量

total_items　　從服務器啟動以後存儲的items總數量

bytes　　當前服務器存儲items佔用的字節數

curr_connections　　當前打開著的連接數

total_connections　　從服務器啟動以後曾經打開過的連接數

connection_structures　　服務器分配的連接構造數

cmd_get get命令　　（獲取）總請求次數

cmd_set set命令　　（保存）總請求次數

get_hits　　總命中次數

get_misses　　總未命中次數

evictions　為獲取空閒內存而刪除的items數（分配給memcache的空間用滿後需要刪除舊的items來得到空間分配給新的items）

bytes_read　讀取字節數（請求字節數）

bytes_written　總發送字節數（結果字節數）

limit_maxbytes　分配給memcache的內存大小（字節）

threads　　當前線程數

Redis是在memcache之後編寫的，大家經常把這兩者做比較，如果說它是個key-value store 的話但是它具有豐富的數據類型，我想暫時把它叫做緩存數據流中心，就像現在物流中心那樣，order、package、store、classification、distribute、end。現在還很流行的LAMP PHP架構 不知道和 redis+MySQL 或者 redis + mongodb的性能比較（聽群裡的人說mongodb分片不穩定）。

先說說reidis的特性

1. 支持持久化

redis的本地持久化支持兩種方式：RDB和AOF。RDB 在redis.conf配置文件裡配置持久化觸發器，AOF指的是redis沒增加一條記錄都會保存到持久化文件中（保存的是這條記錄的生成命令），如果不是用redis做DB用的話還會不要開AOF ，數據太龐大了，重啟恢復的時候是一個巨大的工程！

2.豐富的數據類型

redis 支持 String 、Lists、sets、sorted sets、hashes 多種數據類型,新浪微博會使用redis做nosql主要也是它具有這些類型，時間排序、職能排序、我的微博、發給我的這些功能List 和 sorted set

的強大操作功能息息相關

3.高性能

這點跟memcache很想像，內存操作的級別是毫秒級的比硬盤操作秒級操作自然高效不少，較少了磁頭尋道、數據讀取、頁面交換這些高開銷的操作！這也是NOSQL冒出來的原因吧，應該是高性能

是基於RDBMS的衍生產品，雖然RDBMS也具有緩存結構，但是始終在app層面不是我們想要的那麼操控的。

4.replication

redis提供主從複製方案，跟mysql一樣增量複製而且複製的實現都很相似，這個複製跟AOF有點類似複製的是新增記錄命令，主庫新增記錄將新增腳本發送給從庫，從庫根據腳本生成記錄，這個過程非常快，就看網絡了，一般主從都是在同一個局域網，所以可以說redis的主從近似及時同步，同事它還支持一主多從，動態添加從庫，從庫數量沒有限制。 主從庫搭建，我覺得還是採用網狀模式，如果使用鏈式（master-slave-slave-slave-slave……）如果第一個

slave出現宕機重啟，首先從master 接收 數據恢復腳本，這個是阻塞的，如果主庫數據幾TB的情況恢復過程得花上一段時間，在這個過程中其他的slave就無法和主庫同步了。

5.更新快

這點好像從我接觸到redis到目前為止 已經發了大版本就4個，小版本沒算過。redis作者是個非常積極的人，無論是郵件提問還是論壇發帖，他都能及時耐心的為你解答，維護度很高。有人維護的話，讓我們用的也省心和放心。目前作者對redis 的主導開發方向是redis的集群方向。

redis的安裝

redis的安裝其實還是挺簡單的，總的來說就三步：下載tar包，解壓tar包，安裝。

不過最近我在2.6.7後用centos 5.5 32bit 時碰到一個安裝問題，下面我就用圖片分享下安裝過程碰到的問題，在redis文件夾內執行make時有個如下的錯 undefined reference to '__sync_add_and_fetch_4'

```
    LINK redis-server
zmalloc.o: In function `zmalloc_used_memory':
/usr/tools/redis-2.6.7/src/zmalloc.c:223: undefined reference to `__sync_add_and_fetch_4'
collect2: ld returned 1 exit status
make[1]: *** [redis-server] Error 1
make[1]: Leaving directory `/usr/tools/redis-2.6.7/src'
make: *** [all] Error 2
[root@bogon redis-2.6.7]#
```

上網找了了好多最後在 https://github.com/antirez/redis/issues/736 找到解決方案，write CFLAGS= -march=i686 on src/Makefile head!

```
# Just use 'make dep', but this is only needed by developers.

CFLAGS= -march=i686

release_hdr := $(shell sh -c './mkreleasehdr.sh')
uname_S := $(shell sh -c 'uname -s 2>/dev/null || echo not')
```

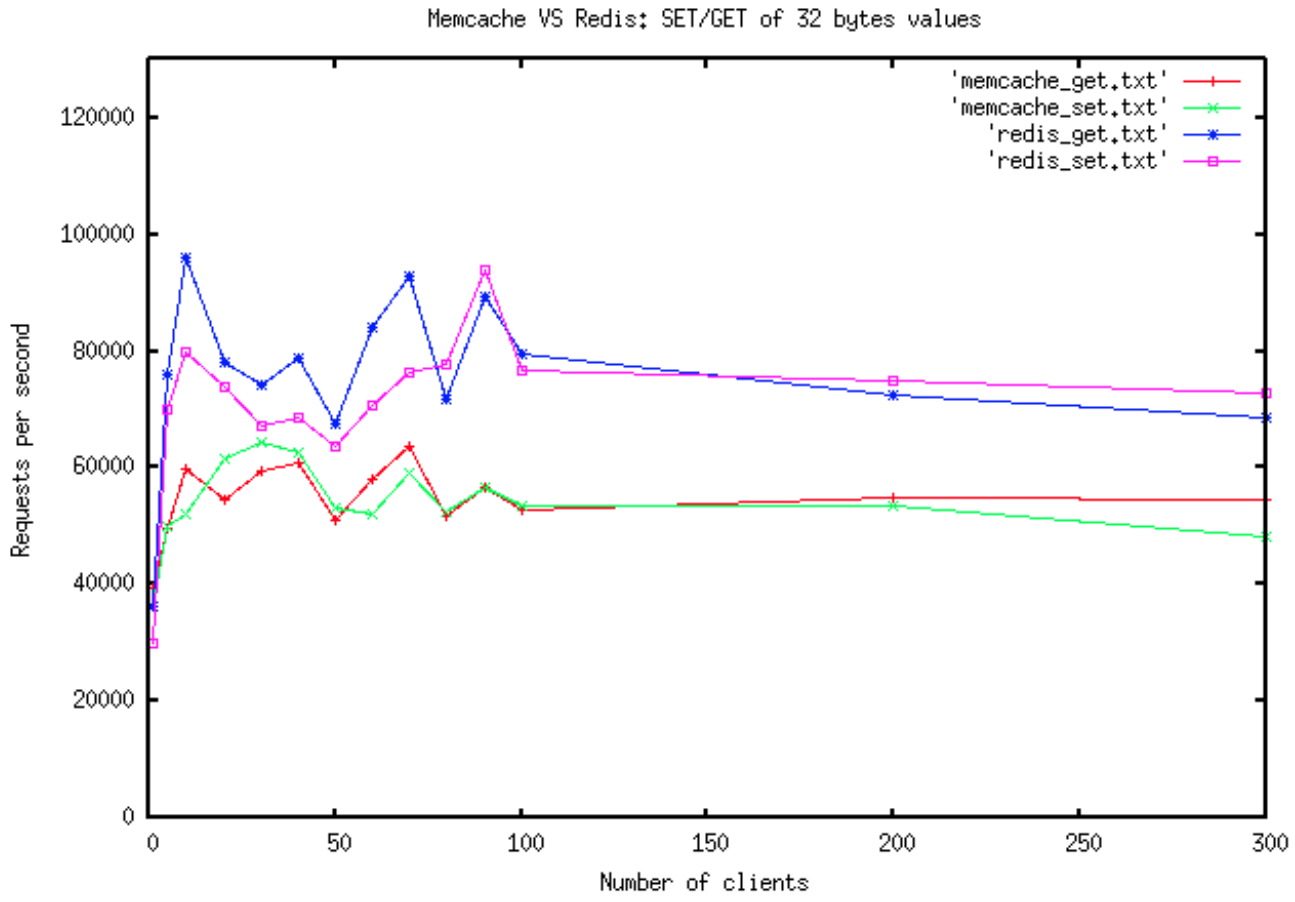
記得要把剛安裝失敗的文件刪除，重新解壓新的安裝文件，修改Makefile文件，再make安裝。就不會發現原來那個錯誤了

關於redis的一些屬性註釋和基本類型操作在上一篇redis 的開胃菜有詳細的說明，這裡就不再重複累贅了（實質是想偷懶 ，哈哈！）

最後，把memcache和redis放在一起不得不會讓人想到兩者的比較，誰快誰好用啊，群裡面已經為這個事打架很久了，我就把我看到的在這裡跟大家分享下。

在別人發了一個memcache性能比redis好很多後，redis 作者 antirez 發表了一篇博文，主要是說到如何給redis 和 memcache 做壓力測試，文中講到有個人說許多開源軟件都應該丟進廁所，因為他們的壓力測試腳本太2了，作者對這個說明了一番。redis vs memcache is definitely an apple to apple comparison。 呵呵，很明確吧，兩者的比較是

不是有點雞蛋挑骨頭的效果，作者在相同的運行環境做了三次測試取多好的值，得到的結果如下圖:



需要申明的是此次測試在單核心處理的過程的數據，memcache是支持多核心多線程操作的（默認沒開）所以在默認情況下上圖具有參考意義，若然則memcache快於redis。那為什麼redis不支持多線程多核心處理呢？作者也發表了一下自己的看法，首先是多線程不變於bug的修復，其實是不易軟件的擴展，還有數據一致性問題因為redis所有的操作都是原子操作，作者用到一個詞nightmare 噩夢，呵呵！ 當然不支持多線程操作，肯定也有他的弊端的比如性能想必必然差，作者從2.2版本後專注redis cluster的方向開發來緩解其性能上的弊端，說白了就是縱向不行，橫向提高。