堆外内存排查

工具

被章耿添加，被章耿最后更新于九月 21, 2016

现象

有人反应线上JSF服务端有一个进程的占用内存特别高，具体如下。

top命令如下：

PID USER PR NI VIRT **RES** SHR S %CPU %MEM TIME+ COMMAND
33801 admin 20 0 **41.4g 14g** 10m S 5.0 5.8 4015:27 java

可以看到这个java进程占用的资源特别高。

分析内存分布

启动命令为：-server -Xms4096m -Xmx4096m -XX:MaxPermSize=256m

1.查看JVM占用内存大小

```
[admin@host-xxxx ~]$ jmap -heap 33801
Attaching to process ID 33801, please wait...
Debugger attached successfully.
Server compiler detected.
JVM version is 20.0-b11
using thread-local object allocation.
Parallel GC with 43 thread(s)
Heap Configuration:
   MinHeapFreeRatio = 40
   MaxHeapFreeRatio = 70
   MaxHeapSize      = 4294967296 (4096.0MB)
   NewSize          = 1310720 (1.25MB)
   MaxNewSize       = 17592186044415 MB
   OldSize          = 5439488 (5.1875MB)
   NewRatio         = 2
   SurvivorRatio    = 8
   PermSize         = 21757952 (20.75MB)
   MaxPermSize      = 268435456 (256.0MB)
Heap Usage:
PS Young Generation
Eden Space:
   capacity = 1402273792 (1337.3125MB)
   used     = 1305741520 (1245.2521514892578MB)
   free     = 96532272 (92.06034851074219MB)
   93.11601824474518% used
From Space:
   capacity = 14680064 (14.0MB)
   used     = 0 (0.0MB)
   free     = 14680064 (14.0MB)
   0.0% used
To Space:
   capacity = 14155776 (13.5MB)
   used     = 0 (0.0MB)
   free     = 14155776 (13.5MB)
   0.0% used
PS Old Generation
   capacity = 2863333376 (2730.6875MB)
   used     = 78102656 (74.4844970703125MB)
   free     = 2785230720 (2656.2030029296875MB)
   2.7276829395642124% used
PS Perm Generation
   capacity = 63176704 (60.25MB)
   used     = 62522104 (59.62572479248047MB)
   free     = 654600 (0.6242752075195312MB)
   98.96385857673107% used
```

可以看到堆内内存正常，没有超过4G，说明大部分内存都是堆外内存。

2. 查看堆外内存情况

网上都推荐使用google-perftools来跟踪，但是需要重启应用。我们先不重启，直接dump内存分析下。

执行 pmap

```
[admin@host-xxx~]$ pmap 33801 > 33801.txt
[admin@host-xxx~]$ cat 33801 | more
Address           Kbytes     RSS   Dirty Mode   Mapping
0000000040000000      36      36       0 r-x--  java
0000000040108000       8       8       8 rwx--  java
0000000040b6d000     132       8       8 rwx--    [ anon ]
……
00007f2c74000000  131072  131072  131072 rwx--    [ anon ]
00007f2c7c000000  131072  131072  131072 rwx--    [ anon ]
00007f2c84000000  131072  131072  131072 rwx--    [ anon ]
00007f2c8c000000  131072  131072  131072 rwx--    [ anon ]
00007f2c94000000  131072  131072  131072 rwx--    [ anon ]
……
……
```

其中看见不少 128M的内存块。 比较可疑

3. dump内存块

使用 gdb进行内存dump。如果没有，则需要安装 yum install gdb。

```
[admin@host-xxx ~]$ gdb --pid 33801
……
Reading symbols from /export/Domains/follow.soa.jd.com/server1/temp/libnetty-transport-native-
epoll4107761723280840577.so...(no debugging symbols found)...done.
Loaded symbols for /export/Domains/follow.soa.jd.com/server1/temp/libnetty-transport-native-
epoll4107761723280840577.so
0x00007f3206fa522d in pthread_join () from /lib64/libpthread.so.0
Missing separate debuginfos, use: debuginfo-install glibc-2.12-1.149.el6.x86_64
(gdb)
看到这个代表可以进行调试
然后外面找刚才的一段地址（前面加上0x）进行dump
(gdb) dump binary memory ./33801_1.dat 0x00007f2c74000000 0x00007f2c7c000000


dump完 按q 退出
(gdb) q
A debugging session is active.
    Inferior 1 [process 33801] will be detached.
Quit anyway? (y or n) y
```

这样我们就拿到了一个内存dump文件。可以通过16进制大概看看里面的内容。

4. 分析内存块。

使用hexdump或者 下载下来拿UltraEdit EditPlus都可以看16进制文件。

```
[admin@host-xxxx ~]$ hexdump -e '16/1 "%02X " " | "' -e '16/1 "%_p" "\n"' ./33801_1.dat  | more
20 00 00 98 2D 7F 00 00 00 00 00 80 2C 7F 00 00 |  ...-.......,...
00 00 00 04 00 00 00 00 00 00 00 04 00 00 00 00 | ................
00 00 00 00 00 00 00 00 45 00 00 00 00 00 00 00 | ........E.......
45 72 72 6F 72 20 77 68 69 6C 65 20 72 65 61 64 | Error while read
28 2E 2E 2E 29 3A 20 43 6F 6E 6E 65 63 74 69 6F | (...): Connectio
6E 20 72 65 73 65 74 20 62 79 20 70 65 65 72 00 | n reset by peer.
00 00 00 00 00 00 00 00 45 00 00 00 00 00 00 00 | ........E.......
45 72 72 6F 72 20 77 68 69 6C 65 20 72 65 61 64 | Error while read
28 2E 2E 2E 29 3A 20 43 6F 6E 6E 65 63 74 69 6F | (...): Connectio
6E 20 72 65 73 65 74 20 62 79 20 70 65 65 72 00 | n reset by peer.
00 00 00 00 00 00 00 00 45 00 00 00 00 00 00 00 | ........E.......
45 72 72 6F 72 20 77 68 69 6C 65 20 72 65 61 64 | Error while read
28 2E 2E 2E 29 3A 20 43 6F 6E 6E 65 63 74 69 6F | (...): Connectio
6E 20 72 65 73 65 74 20 62 79 20 70 65 65 72 00 | n reset by peer.
00 00 00 00 00 00 00 00 45 00 00 00 00 00 00 00 | ........E.......
```

里面全是 ....E....... Error while read(...): Connection reset by peer.

好像是一个对象，里面的字符串是 Error while read(...): Connection reset by peer.

分析数据来源

程序有JSF，JMQ使用了Netty4，zookeeper使用了Netty3。 这几个都有可能申请堆外内存。

搜索字符串“Error while read”

查询JSF和JMQ源码 都未发现该字符串。

搜索netty4.0.24的源码 发现了该字符串：

transport-native-epoll/src/main/c/io_netty_channel_epoll_Native.c

```c
jint read0(JNIEnv * env, jclass clazz, jint fd, void *buffer, jint pos, jint limit) {
    ssize_t res;
    int err;
    do {
        res = read(fd, buffer + pos, (size_t) (limit - pos));
        // Keep on reading if we was interrupted
    } while (res == -1 && ((err = errno) == EINTR));

    if (res < 0) {
        if (err == EAGAIN || err == EWOULDBLOCK) {
            // Nothing left to read
            return 0;
        }
        if (err == EBADF) {
            throwClosedChannelException(env);
            return -1;
        }
        throwIOException(env, exceptionMessage("Error while read(...): ", err));
        return -1;
    }

    if (res == 0) {
        // end-of-stream
        return -1;
    }
    return (jint) res;
}
```

分析源码：如果使用了epoll，客户端连上服务端又断开，就会触发这个异常。

继续查看Netty源码和ReleaseNote，发现在4.0.25版本 已经优化掉了这个东西。

参见：http://netty.io/news/2014/12/31/4-0-25-Final.html

https://github.com/netty/netty/pull/3227

https://source.jboss.org/changelog/Netty?cs=c0e889ae543e291f03f003485ee54dd281945ec6&_sscc=t

测试：

1.测试关闭epoll是否会重现此问题。-------关闭后正常

2.更新netty版本，看是否解决。---------解决

解决：

1.单独升级Netty4的版本。

2.或者使用JSF1.6.0，默认依赖已经提高到Netty4.0.33。

无