Netty高性能编程备忘录(上)

08月 14, 2016 | Filed under 技术 (http://calvin1978.blogcn.com/articles/category/%e6%8a%80%e6%9c%af)

网上赞扬Netty高性能的文章不要太多，但如何利用Netty写出高性能网络应用的文章却甚少，此文权当抛砖引玉。

估计很快就要被拍砖然后修改，因此转载请保持原文链接，否则视为侵权...

http://calvin1978.blogcn.com/articles/netty-performance.html
(http://calvin1978.blogcn.com/articles/netty-performance.html)

参考资料：

- Netty Best Practices a.k.a Faster == Better (http://normanmaurer.me/presentations/2014-facebook-eng-netty/slides.html) by Norman Maurer，Netty核心开发
- Netty高性能之道 (http://www.infoq.com/cn/articles/netty-high-performance) by 李林锋，《Netty权威指南》作者
- Netty系列之Netty百万级推送服务设计要点 (http://www.infoq.com/cn/articles/netty-million-level-push-service-design-points) by 李林锋，但和本文针对的SOA场景不太一样
- 用Netty开发中间件：高并发性能优化 (http://blog.csdn.net/dc_726/article/details/48978891)

1. 连接篇

1.1 Netty Native

Netty Native (http://github.com/netty/netty-tcnative)用C++编写JNI调用的Socket Transport，是由Twitter将Tomcat Native的移植过来，现在还时不时和汤姆家同步一下代码。

经测试，的确比JDK NIO更省CPU。

也许有人问，JDK的NIO也用EPOLL啊，大家有什么不同？ Norman Maurer这么说的：

- Netty的 epoll transport使用 edge-triggered 而 JDK NIO 使用 level-triggered
- C代码，更少GC，更少synchronized
- 暴露了更多的Socket配置参数

第一条没看懂，反正测试结果的确更快更省CPU。

用法倒是简单，只要几个类名替换一下，详见Netty的官方文档1 (http://netty.io/wiki/native-transports.html)，文档2 (http://netty.io/wiki/forked-tomcat-native.html)

但要注意，它跟OS相关且基于GLIBC2.10编译，而CentOS 5.8就只有GLIBC2.5（别问为什么，厂大怪事多，我厂就是还有些CentOS5.8的机器），所以最好还是不要狠狠的直接全文搜索替换，而是用 *System.getProperty(“os.name”)*和 *System.getProperty(“os.version”)*取出操作系统名称与版本，做成一个开关。

另外，Netty很多版本都有修复Netty Native相关的bug，看得人心里发毛，好在最近的版本终于不再说了，所以要用就用Netty的新版。

最后，Netty Native还包含了Google的boringssl(A fork of OpenSSL)，JDK的原生SSL实现比OpenSSL慢很多很多，而大家把SSL Provider配置成OpenSSL时，又要担心操作系统有没装OpenSSL，或者版本会不会太旧。现在好了。

1.2 异步连接，异步传输，告别Commons Pool

异步化最牛头不对马嘴的事情就是，给它配一个类似Commons Pool这样，有借有还的连接池。

在很多异步化的场景里，都用channel.writeAndFlush()原子的发送数据，发完不用同步等response，这时其实不需要独占一条Channel，不需要把它借出去，再还回池里。一来连接池出入之间有并发锁，二来并发请求一多就要狂建连接，到了连接池上限时还要傻傻的等待别人释放连接，而这可能毫无必要。

此时，建议直接建一个连接数组，随机到哪个连接就直接用它发送数据。如果那个连接还没建立或者已经失效，那就建立连接。

顺便说一句，异步的世界里，连建立连接的过程也是异步的，主线程不要等在建连接上，而是把发送的动作封成一个ChannelCallback，等连接建立了，再回调它发送数据，避免因为连接建立的缓慢或网络根本不通，把线程都堵塞了。

Netty4.0.28 (http://netty.io/news/2015/05/07/4-0-28-Final.html)开始也有ChannelPool了，供需要独占Channel的场景如HTTP1.1，比之Commons Pool的特色之一也是这个异步的建连接过程。

1.3 最佳连接数：一条连接打天下？还有传说中的海量连接？

NIO这么神奇，有一种做法是只建一条连接，如Memcached的客户端SpyMemcached。还有一种是既然你能支持海量连接，几千几万的，那我就无节制的可劲的建了。

测试表明，一条连接有瓶颈，毕竟只用到了一个CPU核。 海量连接，CPU和内存在燃烧。。。。

那最佳连接数是传说中的CPU核数么？依然不是。

一切还是看你的场景，连接数在满足传输吞吐量的情况下，越少越好。

举个例子，在我的Proxy测试场景里：

- 2条连接时，只能有40k QPS。
- 48条连接，升到62k QPS，CPU烧了28%
- 4条连接，QPS反而上升到68k ，而且CPU降到20%。

1.4 Channel参数设定

TCP/Socket的大路设置，无非 SO_REUSEADDR， TCP_NODELAY， SO_KEEPALIVE 。另外还有 SO_LINGER ， SO_TIMEOUT, SO_BACKLOG, SO_SNDBUF, SO_RCVBUF。

而用了Native后又加了TCP_CORK和KeepAlive包发送的时间间隔(默认2小时)，详见 EpoolSocketChannelConfig的JavaDoc (http://netty.io/4.0/api/io/netty/channel/epoll/EpollSocketChannelConfig.html)。

所有这些参数的含义，不一一描述了，自己搜索，比如Linux下高性能网络编程中的几个TCP/IP选项 (http://blog.csdn.net/zlxfogger/article/details/44922993)。

而Netty自己的参数CONNECT_TIMEOUT_MILLIS，是Netty自己起一个定时任务来监控建立连接是否超时，默认30秒太长谁也受不了，一般会弄短它。

2. 线程篇

基本知识：《Netty in Action》中文版—第七章 EventLoop和线程模型 (http://ifeve.com/netty-in-action-7/)

2.1 WorkerGroup 与 Boss Group

大家都知道，Boss Group用于服务端处理建立连接的请求，WorkGroup用于处理I/O。

EventLoopGroup的默认大小都是是2倍的CPU核数，但这并不是一个恒定的最佳数量，为了避免线程上下文切换，只要能满足要求，这个值其实越少越好。

Boss Group每个端口平时好像就只占1条线程，无论配了多少。

2.2 上下游线程的绑定

在服务化的应用里，一般处理上游请求的同时，也会向多个下游的服务集群发送请求，但调优指南里都说，尽量，全部重用同一个EventLoop。否则，处理上游请求的线程，就要把后续任务以Runnable的方式，提交到下游Channel的处理线程。

但，一个EventLoop线程可以处理多个Channel的信息，而一个Channel只能注册一个EventLoop线程。所以没办法保证处理上游的Channel，与下游多个连接的Channel，刚好是属于一个EventLoop？

因此，追求极致的Proxy型应用，可能会放弃前面的固定连接池的做法，而是为每个处理上游请求的线程，对应每一台下游服务器创建一条Channel，而且设定它的工作线程就是本上游线程，然后存到threadLocal里。这样的做法连接数可能会增多，但减少了切换，要自行测试权衡。

2.2 业务线程池

Netty线程的数量一般固定且较少，所以很怕线程被堵塞，比如同步的数据库查询，比如下游的服务调用（又来罗嗦，future.get()式的异步在执行future.get()时还是堵住当前线程的啊）。

所以，此时就要把处理放到一个业务线程池里操作，即使要付出线程上下文切换的代价，甚至还有些ThreadLocal需要复制。

2.3 定时任务

像发送超时控制之类的一次性任务，不要使用JDK的ScheduledExecutorService，而是如下：

```
ctx.executor().schedule(new MyTimeoutTask(p), 30, TimeUnit.SECONDS)
```

首先，JDK的ScheduledExecutorService是一个大池子，多线程争抢并发锁。而上面的写法，TimeoutTask只属于当前的EventLoop，没有任何锁。

其次，如果发送成功，需要从长长Queue里找回任务来取消掉它。现在每个EventLoop一条Queue，明显长度只有原来的N分之一。

2.4 快速复习一下Netty的高性能线程池

Netty的线程池理念有点像ForkJoinPool，都不是一个线程大池子并发等待一条任务队列，而是每条线程自己一个任务队列。
不过Netty4的方法是建了N个只有一条线程的线程池，然后用前面说的选择器去选择。而曾经的Netty5 Alpha好像直接就用了ForkJoinPool。

而且Netty的线程，并不只是简单的阻塞地拉取任务，而是非常辛苦命的在每个循环同时做三件事情：

- 先处理NIO的事件
- 然后获取2.3里提到的本线程的定时任务，放到本线程的任务队列里
- 再然后混合2.2里提到的其他线程提交给本线程的任务，一起执行

每个循环里处理NIO事件与其他任务的时间消耗比例，还能通过ioRatio变量来控制，默认是各占50%。

可见，Netty的线程根本没有阻塞等待任务的清闲日子，所以也不使用有锁的BlockingQueue如ArrayBlockingQueue来做任务队列了，而是直接使用下篇里提到的JCTools提供的无锁的MpscLinkedQueue(Mpsc 是Multiple Producer, Single Consumer的缩写)。

文章太长没人看，写到这里就停笔了。剩下内容请看 Netty高性能编程备忘录（下） (http://calvin1978.blogcn.com/articles/netty-performance2.html)

(http://files.blogcn.com/wp01/M00/06/7B/CpAZrVevSwgAAAAAAAcWyBMEGcA880.jpg)

有关的...

- 2016-08-14 -- Netty高性能编程备忘录（下） (http://calvin1978.blogcn.com/articles/netty-performance2.html)
- 2016-10-29 -- Java性能优化指南1.8版，及唯品会的实战 (http://calvin1978.blogcn.com/articles/javatuning.html)
- 2016-10-26 -- 关键业务系统的JVM参数推荐(2016热冬版) (http://calvin1978.blogcn.com/articles/jvmoption-2.html)
- 2016-09-14 -- Btrace入门到熟练小工完全指南 (http://calvin1978.blogcn.com/articles/btrace1.html)

by calvin | tags : Netty (http://calvin1978.blogcn.com/articles/tag/netty), 调优 (http://calvin1978.blogcn.com/articles/tag/%e8%b0%83%e4%bc%98) | 4 (http://calvin1978.blogcn.com/articles/netty-performance.html#comments)

Netty高性能编程备忘录（下） (http://calvin1978.blogcn.com/articles/netty-performance2.html) »
« 从dstat理解Linux性能监控体系 (http://calvin1978.blogcn.com/articles/dstat.html)

You can leave a response (#respond) , or trackback (http://calvin1978.blogcn.com/articles/netty-performance.html/trackback) from your own site.

4 Comments

yancoco
08月 14th, 2016 at 12:19 (#comment-302)

netty官方不提供netty5下载了,理由是增加了代码复杂性并没带来显著的性能提升

回复 (/articles/netty-performance.html?replytocom=302#respond)

匿名
08月 18th, 2016 at 16:44 (#comment-311)

第一条说一个用水平触发，一个用边界触发

回复 (/articles/netty-performance.html?replytocom=311#respond)

匿名
08月 27th, 2016 at 23:50 (#comment-321)

对于业务线程池个人认为也是一个比较纠结的设计，如果是大量的IO等待型场景，会迫使你将消息分发到一个散状的线程池（如JDK的线程池）而不是直接使用netty的线程池，这又不得不带来锁竞争和线程切换的消耗。不知道业务线程池方面，除了使用jdk标准的线程池，是否有更好的解决方案？

回复 (/articles/netty-performance.html?replytocom=321#respond)

匿名
03月 30th, 2017 at 23:10 (#comment-450)

为何不用talent-aio

回复 (/articles/netty-performance.html?replytocom=450#respond)

发表评论

您的电子邮箱不会被公开。

名称

电子邮箱

电子邮箱

网址

评论

发表评论

分类

工作 (http://calvin1978.blogcn.com/articles/category/%e5%b7%a5%e4%bd%9c)
技术 (http://calvin1978.blogcn.com/articles/category/%e6%8a%80%e6%9c%af)
文艺 (http://calvin1978.blogcn.com/articles/category/%e6%96%87%e8%89%ba)
日常 (http://calvin1978.blogcn.com/articles/category/%e6%97%a5%e5%b8%b8)
议论 (http://calvin1978.blogcn.com/articles/category/%e8%ae%ae%e8%ae%ba)

归档

2017年10月 (http://calvin1978.blogcn.com/articles/2017/10)
2016年11月 (http://calvin1978.blogcn.com/articles/2016/11)
2016年10月 (http://calvin1978.blogcn.com/articles/2016/10)
2016年09月 (http://calvin1978.blogcn.com/articles/2016/09)
2016年08月 (http://calvin1978.blogcn.com/articles/2016/08)
2016年07月 (http://calvin1978.blogcn.com/articles/2016/07)
2016年05月 (http://calvin1978.blogcn.com/articles/2016/05)
2016年04月 (http://calvin1978.blogcn.com/articles/2016/04)
2016年01月 (http://calvin1978.blogcn.com/articles/2016/01)
2015年12月 (http://calvin1978.blogcn.com/articles/2015/12)
2015年11月 (http://calvin1978.blogcn.com/articles/2015/11)
2015年09月 (http://calvin1978.blogcn.com/articles/2015/09)

链接

RSS (/feed)
我的微博 (http://www.weibo.com/calvin1978)
春天的旁边 (http://springside.io/)

标签云

AboutMe (http://calvin1978.blogcn.com/articles/tag/aboutme)
bigdata (http://calvin1978.blogcn.com/articles/tag/bigdata)
Dolphin (http://calvin1978.blogcn.com/articles/tag/dolphin)
Football (http://calvin1978.blogcn.com/articles/tag/football)
java (http://calvin1978.blogcn.com/articles/tag/java)
Netty (http://calvin1978.blogcn.com/articles/tag/netty)
Redis (http://calvin1978.blogcn.com/articles/tag/redis)
SpringSide (http://calvin1978.blogcn.com/articles/tag/springside)
SpringSide AboutMe (http://calvin1978.blogcn.com/articles/tag/springside-aboutme)
云门舞集 (http://calvin1978.blogcn.com/articles/tag/%e4%ba%91%e9%97%a8%e8%88%9e%e9%9b%86)
人山人海 (http://calvin1978.blogcn.com/articles/tag/%e4%ba%ba%e5%b1%b1%e4%ba%ba%e6%b5%b7)
伍迪艾伦 (http://calvin1978.blogcn.com/articles/tag/%e4%bc%8d%e8%bf%aa%e8%89%be%e4%bc%a6)
卡夫卡 (http://calvin1978.blogcn.com/articles/tag/%e5%8d%a1%e5%a4%ab%e5%8d%a1)
古诗 (http://calvin1978.blogcn.com/articles/tag/%e5%8f%a4%e8%af%97)
在节假日 (http://calvin1978.blogcn.com/articles/tag/%e5%9c%a8%e8%8a%82%e5%81%87%
技术会 (http://calvin1978.blogcn.com/articles/tag/%e6%8a%80%e6%9c%af%e4%bc%9a)
方所 (http://calvin1978.blogcn.com/articles/tag/%e6%96%b9%e6%89%80)
服务化 (http://calvin1978.blogcn.com/articles/tag/%e6%9c%8d%e5%8a%a1%e5%8c%96)
现代舞 (http://calvin1978.blogcn.com/articles/tag/%e7%8e%b0%e4%bb%a3%e8%88%9e)
现代诗 (http://calvin1978.blogcn.com/articles/tag/%e7%8e%b0%e4%bb%a3
知识分子

(http://calvin1978.blogcn.com/articles/tag/%e7%9f%a5%e8%af%86%e5%88%86%e5%ad%90)
码农
(http://calvin1978.blogcn.com/articles/tag/%e7%a0%81%e5%86%9c)
窦唯
(http://calvin1978.blogcn.com/articles/tag/%e7%aa%a6%e5%94%af)
调优
(http://calvin1978.blogcn.com/articles/tag/%e8%b0%83%e4%bc%98)
音乐现场
(http://calvin1978.blogcn.com/articles/tag/%e9%9f%b3%e4%b9%90%e7%8e%b0%e5%9c%l
黄耀明
(http://calvin1978.blogcn.com/articles/tag/%e9%bb%84%e8%80%80%e6%98%8e)