

CSDN新首页上线啦，邀请你来立即体验！(http://feed.csdn.net/)

立即体验

CSDN

博客 (http://feed.csdn.net/?ref=toolbar)学院 (http://edu.csdn.net/?ref=toolbar)

下载 (http://download.csdn.net/?ref=toolbar)更多 ▾

3

JVM能够开启多少线程

原创2015年11月11日 21:21:01

标签：Java (http://so.csdn.net/so/search/s.do?q=Java&t=blog) / 线程 (http://so.csdn.net/so/search/s.do?q=线程&t=blog) / jvm (http://so.csdn.net/so/search/s.do?q=jvm&t=blog) / Linux内核 (http://so.csdn.net/so/search/s.do?q=Linux内核&t=blog)

2798

CanYellWang (http://my.csdn.net/?ref=toolbar)

清欢君 (http://blog.csdn.net/lovewithbeauty)

12

0

0

未开通 (https://github.com/lovewithbeauty)

最近在看<<Java并发编程实战>>一书过程中，在Task Execution一节中讲到,针对每个任务都启动一个线程来处理,尤其在大量创建线程的场景下,会给工程实践带来很多问题。

- 1)线程的创建和销毁都是有开销的。线程的创建需要时间，如果针对每个任务都启动线程处理，则无疑会造成请求不能得到尽快处理。如果请求比较频繁而且轻量级，为每个请求创建新线程会消耗大量资源。
- 2)资源消耗.线程会消耗系统资源，尤其是内存。如果可运行的线程(Runnable状态)数量大于可用处理器数量，那么有些线程会得不到调度。这些线程一旦数量太多就会占用大量内存，给GC带来回收压力。同时大量的线程对CPU的竞争带来的上下文切换也会影响系统性能。如果已经有足够的线程来让CPU保持忙碌，那么再创建新的线程非但不能提高CPU利用率，反而会降低性能。
- 3)稳定性。能够创建的线程数量是有上限的。这个数量与平台相关，也受限于多个因素:JVM启动参数(-Xss),创建Thread时在构造函数中指定的线程栈大小(stack size)和底层操作系统对线程数的限制。

从第三点可见，线程数量和线程栈大小(通过-Xss或stack size指定，一般为0.5M-1M)、可用内存和操作系统限制有关系。因此本文讨论一下，究竟JVM最多能够开启多少线程。

不考虑OS限制的情况下，RAM越大,线程栈越小，可以创建的线程就越多。一个理论上的计算公式如下：
Max number of threads(最大线程数)=(进程最大可用内存-JVM分配内存-OS Reserved Memory(JNI,本地方法栈))/thread stack size(线程栈大小)
对于32位操作系统，允许创建的线程数量一个重要的制约因素便是内存RAM。由于寻址空间仅仅有2^32byte(4GB)，按照线程栈大小为0.5M来计算，最大线程数为2^13,即不到1万。如果按照线程栈大小为1M来计算，最大线程数不过数千。
对于64为操作系统，内存不再是最大制约因素，最大线程数受限于Linux内核.以我的Ubuntu 64-bit 操作系统，16GB RAM为例，执行如下代码：

```
[java]
1. package snow.rabbit.java.thread;
2. /**
3.  *
4.  * @author lawrence
5.  *
6.  */
7. public class TestMaxSupportThreads {
8.
9.     private static final class TestThread extends Thread
10.    {
11.        private final int number;
12.
13.        public TestThread(int number) {
14.            this.number = number;
15.        }
16.    }
```

他的最新文章

更多文章

(http://blog.csdn.net/lovewithbeauty)

为何需要定制化的Thread Factory?

(/lovewithbeauty/article/details/5005746)

在线服务的内存泄露问题

(/lovewithbeauty/article/details/4986846)

JVM能够开启多少线程

(/lovewithbeauty/article/details/4978655)

Ubuntu 12.10 中增加Eclipse图标到启动器

(/lovewithbeauty/article/details/1464419)

在线课程

神经网络的原

原理及结构设计

免费直播

神经网络

的进阶之路

移动研发

(http://edu.csdn.net/huiyiCourse/detail/596?utm_source=blog9)

(http://edu.csdn.net/huiyiCourse/detail/602?utm_source=blog9)

热门文章

JVM能够开启多少线程 (/lovewithbeauty/article/details/49786551)

2797

为何需要定制化的Thread Factory? (/lovewithbeauty/article/details/50057469)

502

Process vs Thread (/lovewithbeauty/article/details/9220357)

497

Java基本特性(1)-Polymorphism: 论多态 (/lovewithbeauty/article/details/9091461)

```
17. public TestThread(int number,int stackSize) {
18.     super(null, null, "Thread-Hi-" +number,stackSize);
19.     this.number = number;
20. }
21.
22. @Override
23. public void run() {
24.     if (number%100 ==0)
25.     {
26.         System.out.println("Thread "+number+" started.");
27.     }
28.     try {
29.         Thread.sleep(Long.MAX_VALUE);
30.     } catch (InterruptedException e) {
31.         e.printStackTrace();
32.     }
33. }
34.
35. /**
36.  * @param args
37.  */
38. public static void main(String[] args) {
39.     int i=0;
40.     try {
41.         for (i=0; i < 200; i++)
42.         {
43.             Thread t=new TestThread(i,2<<19);//512K
44.             t.start();
45.         }
46.     } catch (Throwable e) {
47.         System.out.println("Error occured when creating thread "+i);
48.         e.printStackTrace();
49.     }
50. }
51. }
```

471

在线服务的内存泄露问题 (/lovewithbeauty/article/details/49868465)

459

输出结果如下:

```
[plain]
1. Thread 32100 started.
2. Error occured when creating thread 32129
3. java.lang.OutOfMemoryError: unable to create new native thread
4. at java.lang.Thread.start0(Native Method)
5. at java.lang.Thread.start(Thread.java:640)
6. at TestMaxSupportThreads.main(TestMaxSupportThreads.java:37)
```

此时内存还有很大剩余,但是线程创建到32129时便出现异常,这是由于OS的限制导致,和kernal/ulimit有关.具体见如下设置:

1)/proc/sys/kernel/pid_max :系统允许的最大pid

与用户态不同,对于Linux内核而言,进程和线程之间的区别并不大,线程也不过是共享内存空间的进程.每个线程都是一个轻量级进程(Light Weight Process),都有自己的唯一PID(或许叫TID更合适一些)和一个TGID(Thread group ID),TGID是启动整个进程的thread的PID.

简单来说,当一个进程被创建的时候,它其实是一个PID和TGID数值相同线程.当线程A启动线程B时,线程B会有自己的唯一PID,但它的TGID会从A继承而来.这样通过PID线程可以独立得到调度,而相同的TGID可以知道哪些线程属于同一个进程,这样可以共享资源(RAM,虚拟内存、文件等)。

```
[plain]
1. 通过ps -fL可以看到LWP轻量级进程信息:
2. lizhong@lizhongpc:~$ ps -fL
3.  UID      PID  PPID  LWP   C  NLWP  STIME  TTY          TIME CMD
4.  lizhong   4167  2252  4167   0    1  14:49 pts/2    00:00:00 -bash
5.  lizhong   6592  4167  6592   0    1  16:40 pts/2    00:00:00 ps -fL
6.
7. 我们启动200个线程后,可以看到LWP显著增加.
8. lizhong@lizhongpc:~$ ps -fL
9.  UID      PID  PPID  LWP   C  NLWP  STIME  TTY          TIME CMD
10. lizhong   4167  2252  4167   0    1  14:49 pts/2    00:00:00 -bash
11. lizhong   6828  4167  6828   0    1  16:40 pts/2    00:00:00 ps -fL
```

- 2)/proc/sys/kernel/threads-max :系统支持的最大线程数
- 3)max_user_process (ulimit -u) :每个用户允许的最大进程数

4)/proc/sys/vm/max_map_count: Linux支持虚拟内存，也就是交换空间,可以把磁盘的一部分作为RAM的扩展，逻辑存储和物理存储的映射就要保存在地址映射表中。max_map_count限制了线程可以拥有的VMAs (Virtual Memory Areas)。

将pid_max,threads-max,max_user_process和max_map_count都改为4194000后，分别将Thread构造函数中stackSize指定为1M,512K,64K,达到的最大线程数分别是423100,626400,626400。

- 从中我们可以看出两点：
- 1)进程可用内存不仅仅局限于RAM，还包括Swap分区(虚拟内存)，在线程不断创建的过程中，首先空闲的RAM不断减少，然后是空闲Swap不断减少。
 - 2)我们在线程构造函数中指定了stackSize分别为512K和64K，但达到的最大线程数是相差无几的，说明JVM并非绝对按照我们指定的大小为线程分配内存。

综上所述，JVM最大开启线程数取决于可用内存(包括虚拟内存)和stack size，在OS层面又和pid_max、threads-max、max_user_process和max_map_count相关。

下面是在16G内存下，stack size 设置为512K,各项OS内核参数调整到最大，创建线程的过程中不断跟踪内存情况和轻量级线程情况的截图：

[plain]						
1.	yangy@yangy:~/lizhongspace\$ free -m					
2.	total	used	free	shared	buffers	cached
3.	Mem:	15935	579	15355	41	50 245
4.	-/+ buffers/cache:	282	15652			
5.	Swap:	19070	0	19070		
6.	yangy@yangy:~/lizhongspace\$ free -m					
7.	total	used	free	shared	buffers	cached
8.	Mem:	15935	4805	11130	41	50 245
9.	-/+ buffers/cache:	4508	11426			
10.	Swap:	19070	0	19070		
11.	yangy@yangy:~/lizhongspace\$ free -m					
12.	total	used	free	shared	buffers	cached
13.	Mem:	15935	7013	8921	41	50 245
14.	-/+ buffers/cache:	6717	9217			
15.	Swap:	19070	0	19070		
16.	yangy@yangy:~/lizhongspace\$ free -m					
17.	total	used	free	shared	buffers	cached
18.	Mem:	15935	8394	7540	41	50 245
19.	-/+ buffers/cache:	8098	7836			
20.	Swap:	19070	0	19070		
21.	yangy@yangy:~/lizhongspace\$ free -m					
22.	total	used	free	shared	buffers	cached
23.	Mem:	15935	11192	4742	41	50 245
24.	-/+ buffers/cache:	10895	5039			
25.	Swap:	19070	0	19070		
26.	yangy@yangy:~/lizhongspace\$ free -m					
27.	total	used	free	shared	buffers	cached
28.	Mem:	15935	12630	3304	41	50 245
29.	-/+ buffers/cache:	12334	3600			
30.	Swap:	19070	0	19070		
31.	yangy@yangy:~/lizhongspace\$ free -m					
32.	total	used	free	shared	buffers	cached
33.	Mem:	15935	14248	1686	41	50 245
34.	-/+ buffers/cache:	13951	1983			
35.	Swap:	19070	0	19070		
36.	yangy@yangy:~/lizhongspace\$ free -m					
37.	total	used	free	shared	buffers	cached
38.	Mem:	15935	15091	843	41	50 245
39.	-/+ buffers/cache:	14794	1140			
40.	Swap:	19070	0	19070		
41.	yangy@yangy:~/lizhongspace\$ free -m					
42.	total	used	free	shared	buffers	cached
43.	Mem:	15935	15758	176	27	0 32
44.	-/+ buffers/cache:	15725	209			
45.	Swap:	19070	635	18435		
46.	yangy@yangy:~/lizhongspace\$ free -m					
47.	total	used	free	shared	buffers	cached
48.	Mem:	15935	15787	147	2	0 13
49.	-/+ buffers/cache:	15773	161			
50.	Swap:	19070	8706	10364		
51.	yangy@yangy:~/lizhongspace\$ free -m					
52.	total	used	free	shared	buffers	cached
53.	Mem:	15935	15788	146	7	0 26
54.	-/+ buffers/cache:	15761	173			
55.	Swap:	19070	10723	8347		

```
56. yangy@yangy:~/lizhongspace$ ps -fL
57.  UID      PID      PPID      LWP    C  NLWP  STIME  TTY          TIME CMD
58.  yangy    2237    2236    2237    0    1  21:20  pts/11    00:00:00 -bash
59.  yangy    631902  2237    631902  0    1  21:31  pts/11    00:00:00 ps -fL
```

以上内容同时发表在我的个人博客清欢de个人博客
(<http://mthinking.net/blog/tecnology/workexperience/b8e9d8d1-8bb2-4dbb-8287-b4e3b5027da2>)
中，欢迎大家访问。

版权声明：本文为博主原创文章，未经博主允许不得转载。

发表你的评论

(<http://my.csdn.net/>)

相关文章推荐

JVM可支持的最大线程数 (/xyls12345/article/details/26482387)

JVM最大线程数 (2012-07-04 23:20:15) 转载▼ 标签： jvm 最大线程数 it 分类： java分...
xyls12345 (<http://blog.csdn.net/xyls12345>) 2014-05-21 15:59 9469

JAVA最多支持多少个线程 (/xcwll_sina/article/details/47042661)

1. java的线程开启，默认的虚拟机会分配1M的内存，但是在4G的windows上线程最多也就开到300多，是因为windows本身的一些限制导致。2. 虚拟机给每个线程分配的内存（栈...
xcwll314 (<http://blog.csdn.net/xcwll314>) 2015-07-24 16:05 1720

BlockingQueue详解 (/ydj7501603/article/details/17246949)

本文转载自：<http://ws:majunfeng.iteye.com/blog/1629354> 前言： 在新增的Concurrent包中，BlockingQueue很好的...
ydj7501603 (<http://blog.csdn.net/ydj7501603>) 2013-12-10 17:00 5877

Cache和Buffer的区别 (/caoshuming_500/article/details/7332561)

1. Cache：缓存区，是高速缓存，是位于CPU和主内存之间的容量较小但速度很快的存储器，因为CPU的速度远远高于主内存的速度，CPU从内存中读取数据需等待很长的时间，而 Cache保存着CPU刚...
caoshuming_500 (http://blog.csdn.net/caoshuming_500) 2012-03-08 14:14 2864

JVM最大可支持线程数计算方法 (/xiaoqingyu999/article/details/52551625)

转自：http://blog.csdn.net/freebird_lb/article/details/8196743 JVM最大创建线程数量由JVM堆内存大小、线程的Stack内存大小、系...
xiaoqingyu999 (<http://blog.csdn.net/xiaoqingyu999>) 2016-09-15 23:28 489

关于jvm的线程调度在不同操作系统下的实现说明 (/destruction666/article/details/8627217)


java的线程概念与操作系统的线程概念是不同的，java的线程概念差不多与windows线程概念一致，但是java既然目标是跨平台语言，那么它的线程机制概念是在所有平台上都是一样的，但是实际实现又不是...
destruction666 (<http://blog.csdn.net/destruction666>) 2013-03-01 17:54 1414

<http://blog.csdn.net/lovewithbeauty/article/details/49786551>

4/6


java jvm 最大线程数设置 (/javascript_2011/article/details/40658941)


最近想测试下Openfire下的最大并发数，需要开大量线程来模拟客户端。对于一个JVM实例到底能开多少个线程一直心存疑惑，所以打算实际测试下，简单google了把，找到影响线程数量的因素有下面几个： ...

 javascript_2011 (http://blog.csdn.net/javascript_2011) 2014-10-31 18:39 5506

Tomcat 6.0.32中调整JVM大小及最大线程数 (/aovenus/article/details/6600789)

1、调整JVM大小调整前：JVM大小查看，如下图所示：【调整方法】编辑startup.bat，添加如下内容保存，并重启tomcat即可。调整后查看JVM大小： 2、调整最大线程数tomcat 6.0.3...

 aovenus (<http://blog.csdn.net/aovenus>) 2011-07-12 17:17 1611



java 查看JVM中所有的线程的活动状况 (<http://download.csdn.net/detail...>)

[/http://download.csdn.net/detail/...](#) 2010-06-21 15:45 59KB [下载](#)




Java分布式应用学习笔记03JVM对线程的资源同步和交互机制 (<http://dow...>)

[/http://download.csdn.net/detail/...](#) 2011-09-28 17:22 277KB [下载](#)


JProfiler的详细使用介绍（JVM对象内存线程监测工具） (/chenleixing/article/details/442...

一、安装JProfiler 从<http://www.ej-technologies.com/>下载5.1.2并申请试用序列号 二、主要功能简介 1. 内存剖析 Memory pr...

 chenleixing (<http://blog.csdn.net/chenleixing>) 2015-03-12 21:07 8437


JVM的线程状态及如何排查死锁原因 (/u014484649/article/details/27338819)

1、TIMED_WAITING 2、waiting 3、blocked 4、runnable

 u014484649 (<http://blog.csdn.net/u014484649>) 2014-05-28 16:56 3001


linux下JVM线程负载过高问题排查 (/hawkdown/article/details/38315931)

这月初，公司搜索服务发生了无响应

 hawkdown (<http://blog.csdn.net/hawkdown>) 2014-07-31 13:47 2238


jvm 内存模型与线程 & Volatile (/u012868901/article/details/51516313)

1.Java内存模型 CPU在运行的时候，不可能把所有的东西都放在寄存器里面，所有需要使用内存。这个内存就是我们知道的那个内存。但是实际情况是，内存的读写速度于CPU的指令操作差了几个数...

 u012868901 (<http://blog.csdn.net/u012868901>) 2016-05-27 14:10 203



JVM线程资源同步及交互机制 (/chen_fly2011/article/details/54923538)

1 JVM线程资源同步及交互机制 Java程序采用多线程的方式来支撑大量的并发请求处理，程序在多线程方式执行的情况下，复杂程度远高于单线程串行执行的程序。尤其是在多核或多 CPU系统中，多线程执行...

 chen_fly2011 (http://blog.csdn.net/chen_fly2011) 2017-02-08 10:11 98



JVM最多支持多少个线程 (/xinyuan_java/article/details/45502377)

Java虚拟机最多支持多少个线程？跟虚拟机开发商有关么？跟操作系统呢？还有其他的因素吗？ 这取决于你使用的CPU，操作系统，其他进程正在做的事情，你使用的Java的版本，还有其他的因素。 ...

 xinyuan_java (http://blog.csdn.net/xinyuan_java) 2015-05-05 14:47  374

JVM监控工具介绍jstack, jconsole, jinfo, jmap, jdb, jsta (Linux 如何查看进程的各线程的CP...

<http://dolphin-ygj.iteye.com/blog/366216> JVM监控工具介绍jstack, jconsole, jinfo, jmap, jdb, jsta ...

 caolaosanahnu (<http://blog.csdn.net/caolaosanahnu>) 2012-05-15 11:39  1190

3



JVM之线程实现 (/qq_33938256/article/details/52615257)

线程 1 实现线程的方式 11 使用内核线程实现 12 使用用户线程实现 13 用户线程加轻量级进程混合实现 2 Java的线程实现并发不一定要依赖多线程，PHP中有多进程并发。但是，Java里面的并...

 qq_33938256 (http://blog.csdn.net/qq_33938256) 2016-09-22 00:46  283


通过JVM堆栈分析线程出现大量异常的原因 (/jessysong/article/details/69218722)

转自：<http://rujingzhang.iteye.com/blog/2251792> 首先进入线上，使用ps -aux命令，查看jvm进程，可以得到运行tomcat的jdk的地址： /hom...

 jessysong (<http://blog.csdn.net/jessysong>) 2017-04-04 23:28  174

java获取JVM的CPU占用率、内存占用率、线程数及服务器的网口吞吐率、磁盘读写速率 (/qq...

运行环境：jdk 1.8 + tomcat 7 + Spring 4.3 tomcat 容器启动及进入init()进行JVM PID的获取及执行任务 可能会碰到的问题：无法引入com.sun.m...

 qq_28580453 (http://blog.csdn.net/qq_28580453) 2017-04-19 22:02  125