Appropriate Tomcat 5.5 start-up parameters to tune JVM for extremely high demand, large heap web application?

Asked 16 years, 2 months ago Modified 12 years, 4 months ago

Viewed 23k times

11

We have recently migrated a large, high demand web application to Tomcat 5.5 from Tomcat 4 and have noticed some peculiar slowdown behavior that appears to be related to JVM pauses. In order to run our application and support increased load over time on Tomcat 4, many not so standard JVM parameters were set and tuned as per the below, and I am hoping someone with Tomcat JVM tuning experience can comment on anything that would likely be detrimental to a Tomcat 5.5 install. Note also that some of these could be carry over from previous versions of Java (we were running Tomcat 4 on Java 1.6 with these parameters successfully for some time, but some may have been introduced to help garbage collection on Java 1.4 which was the basis of our Tomcat 4 install for a long time, and may now doing more harm than good).

Some notes:

- Application memory footprint is around 1GB, probably slightly over.

- CPU is not an issue - all machines serving the app (load balanced) are < 30% CPU
- Lots of headroom on physical memory on the machines.
- -XX:MaxPermSize=512m was the only parameter added as part of the 5.5 upgrade and was reactive to an outofmemory permgen space issue (which it solved).
- Running on Java 1.6, Solaris OS

-server -Xms1280m -Xmx1280m -XX:MaxPermSize=512m -XX:ParallelGCThreads=20 -XX:+UseConcMarkSweepGC -XX:+UseParNewGC -XX:SurvivorRatio=8 -XX:TargetSurvivorRatio=75 -XX:MaxTenuringThreshold=0 -XX:+AggressiveOpts -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:-TraceClassUnloading -Dsun.io.useCanonCaches=false -Dsun.net.client.defaultConnectTimeout=60000 -Dsun.net.client.defaultReadTimeout=60000

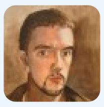
java tomcat jvm performance

Share

Improve this question

Follow

asked Oct 14, 2008 at 19:30



Peter

29.8k ● 22 ● 91 ● 126

5 Answers

Sorted by: Highest score (default)

7

One of the Java Champions, Kirk Pepperdine's blog : http://kirk.blog-city.com/how_to_cripple_gc_ergonomics.htm.
Quote 1 "GC documentation will tell you what the setting affects but often without telling what the effect will be. The biggest clue that you've taken the wrong fork in the road is when you explicitly set a value and then give a hint to GC ergonomics. Another clue is if you don't have a sound reason to adjust a setting. And just because some so called expert says this setting works best is only noise, not sound and certanly not a reason."

Quote 2 "As I've stated in a prevous blog entry , don't touch the knobs unless you have a very good reason to do so. If you must touch the knobs, tred lightly by only using those that help ergonomics and not those that pin things down crippling ergonomics ability to meet your pause time and throughput goals."

So, I would suggest that you go back to plain
-server -Xms1280m -Xmx1280m -XX:MaxPermSize=512m -XX:+UseConcMarkSweepGC -XX:+PrintGCDetails -XX:+PrintGCTimeStamps -XX:-TraceClassUnloading -Dsun.io.useCanonCaches=false -Dsun.net.client.defaultConnectTimeout=60000 -Dsun.net.client.defaultReadTimeout=60000

Find if that gives you better performance. If yes, stick to it
BTW, did -XX:MaxPermSize=378m have any issues ?
Java 1.6 has much better ergonomics than 1.4. You might want to tune it less than 1.4

BTW, did you try Tomcat 6 ? Tomcat 6 runs much better on Java 6 than Tomcat 5.5.

P.S : I've been using Tomcat for a while now and usually try to give sun's JDK free reign with little tuning here and there.

Share Improve this answer Follow

answered Oct 14, 2008 at 20:14

anjanb
13.8k ● 19 ● 80 ● 106

3

As someone who's in the midst of messing with this as well, I certainly don't have any definitive answers, especially given how application-specific this sort of thing is. A good reference, which you've likely seen, is here:

http://java.sun.com/javase/technologies/hotspot/gc/gc_tuning_6.html

However,it's a pretty long list of jvm parameters, which suggests that there's likely unnecessary parameters set, especially given that you have several debugging options on (PrintGCDetails, PrintGCTimeStamps, TraceClassUnloading) which can't be good on a production app. 60 second timeouts might also be eating up resources. "server" is default but won't do any harm.

How does the application run with minimal tuning parameters (jvm size, MaxPermSize)?

Share Improve this answer

answered Oct 14, 2008 at 19:54

Follow

Steve B.
57.2k ● 12 ● 97 ● 134

1 I have a lot of production JVMs running with PrintGCDetails, there is always *some* overhead, but negligible for the value they provide. – Jé Queue Oct 13, 2010 at 1:44

3

just found a webinar from tomcat developers on tuning tomcat : http://springsource.com/node/555. a follow-up to the webinar : http://blog.springsource.com/2008/10/14/optimising-and-tuning-apache-tomcat-part-2/

BR,
~A

Share Improve this answer

Follow

answered Oct 14, 2008 at 20:49
anjanb
13.8k ● 19 ● 80 ● 106

1 The link to that webinar is dead. I don't suppose you have a more recent link for it, do you? – Brad Larson Aug 9, 2012 at 20:29

Please avoid link-only answers, also to avoid the link-rot problem. – Valerio Bozz Apr 17, 2020 at 8:59

I guess I didn't know that 12 years ago -- especially about the link rot -- there was quite a bit of content in that link and I thought that link would do better justice than copy paste the content. There might be something in web archives and I'll try to see if the archives saved that content. – anjanb Apr 17, 2020 at 13:25

the link to the tuning article is still valid. the link to the webinar is lost, I guess for now. – anjanb Apr 17, 2020 at 13:52

1

You might also want to take a look at changing the min/max number of threads that Tomcat will use to handle requests in `conf/server.xml`:

```
<Connector port="8080" maxThreads="150" minSpareThread
maxSpareThreads="75" ...
```

One rule of thumb that I had heard in a previous job was that the maxThreads should be equal to the amount of simultaneous connections you expect to handle. I'm not sure how scientific that claim is though, although I certainly think it makes sense as you don't want clients to be blocked waiting for a thread to free up to handle their request..

Share Improve this answer Follow

answered Oct 15, 2008 at 13:47
matt b
140k ● 66 ● 284 ● 350

Thread count rule I've heard is to take the CPU% of the average thread and the # of execution cores. Threads should be Cores/Thread%. If you have 8 cores and each thread uses 25% CPU then 8/.25 = 32 threads. Better for a connection to wait a little than for it to slow down every other connection. – Zan Lynx Oct 12, 2010 at 23:35

I don't think there is any rule that you can use to calculate, it depends entirely on how many resources each thread consumes. I find adjust the max number until it reaches close to the thresh-holds under maximum load is a good option. What you do not want under any circumstance is that you start using the page file because the server will then slow down so much that it might as well have crashed! More threads != more throughput. If 1 Thread uses most of the Available Resources, then its best to stick to 1 thread. You'll be able to process 10 Threads ALOT quicker this way than 10 simultaneous. – Brian Apr 5, 2013 at 8:18

0

It may also be worth experimenting with an alternative JVM (like JRockit) to see if the differences in the garbage collection model are well suited to your application.

Share Improve this answer Follow

answered Oct 15, 2008 at 16:55

Tim Howland
7,970 ● 4 ● 29 ● 46