Table of Contents

Eclipse optimization

java, eclipse, install
One size doesn't fit all :) Submitted by Andrew Sowerby on Mon, 2016-02-15 13:44 at discussion of Eclipse optimizer plugin

Even if this isn't just a way of harvesting user details, which some reviewers seem to think it is, I can't see the point of this plugin.

You can achieve the same results by adjusting your -vmargs in the eclipse.ini, or in your shortcut or command line you use to start the application.

1) Just add adequate heap settings: e.g. **Xms1024m** and **Xmx1024m**, or more, depending on how hard you push eclipse (setting them to the same size will avoid unnecessary heap resizing and garbage collection cycles)

In my main dev environment, I use **-Xms4096m** and **-Xmx4096m** (the machine has 16GB physical memory) because I work with some fairly large files that would occasionally trigger an OOM otherwise.

You should always start fairly small, and only increase it when you face OOM errors, so that you get to know how much memory you typically need. There's no point having a 4G or 8G heap if you're only using 1GB of it on average!! When it eventually grows and a GC cycle is triggered, it will take quite some time to complete.

2) disable class verification with **-Xverify:none**

Note: this can backfire if you are loading plugins from the marketplace that aren't very solid - if you find Eclipse crashes, you should start by removing (or commenting out) this parameter from your eclipse.ini

3) adjust perm sizes (Oracle JVM - not relevant to the IBM JVM) - however, if you have a 64-bit JVM, you should see they are already set this way by default, anyway :)

```
-XX:PermSize=256m
-XX:MaxPermSize=256m
```

If you DO have a 64-bit JVM, you should probably consider increasing them both to 512m.

If you are running eclipse on an IBM JVM, you can use **-Xcompressedrefs** (this is pretty nice, and is enabled by default in all IBM WebSphere based products since WAS 8, perhaps older).

Don't set it for an Oracle or OpenJDK JVM, because it won't be understood, and might get misinterpreted as -Xcomp!

There are also some things this plugin does that I wouldn't agree with:

4) it switches GC policy (Oracle JVM - not relevant to IBM JVM) -XX:+UseParallelGC

This is not usually helpful unless you have really huge heapsizes (e.g. 10G, according to Oracle themselves).

I think that it would be better to just disable System.gc() with -XX:+DisableExplicitGC and leave it at that, in most cases. Otherwise, you could be doing more harm than good.

5) -server

this sets the server version of the Java HotSpot VM. Even though this setting causes the JVM to perform more slowly at first, performance improves over time. However, I'm not sure this will ever be the case for a typical user of Eclipse, who will not be running the eclipse JVM for long enough to see the that “performance improves over time”. I suspect that the default “-client” JVM setting would be just as good in most cases.

TLDR;

This tool makes changes that probably aren't necessary for most eclipse environments, and some of which may actually do more harm than good.

Of course, it is better to enable verbose GC logging, collect data and analyze it with good tools such as IBM's PMAT or GCMV (verbosegc log troubleshooting tools) to see what your bottleneck actually is. Then, make the suggested changes, or just look at Oracle's or IBM's (depending on which JVM you use) performance tuning guides to make the correct changes.

These “one size fits all” tools are rarely a precise fit for anyone!

См. также

- Заметки по Java