

ImportNew

- [首页](#)
- [所有文章](#)
- [资讯](#)
- [Web](#)
- [架构](#)
- [基础技术](#)
- [书籍](#)
- [教程](#)
- [Java小组](#)
- [工具资源](#)

- 导航条 - ▾

JVM (6) : JVM调优-从eclipse开始

2017/03/06 | 分类: [基础技术](#) | [2 条评论](#) | 标签: [Eclipse](#), [JVM](#)

分享到:

原文出处: [纯洁的微笑](#)



概述



什么是jvm调优呢? jvm调优就是根据gc日志分析jvm内存分配、回收的情况来调整各区域内存比例或者gc回收的策略;更深一层就是根据dump出来的内存结构和线程栈来分析代码中不合理的地方给予改进。eclipse优化主要涉及的是前者,通过gc日志来分析。本文主要是通过分析eclipse gc日志为例来示例如何根据gc日志来分析jvm内存而进行调优,像根据关闭eclipse启动项、关闭各种校验等措施来优化eclipse本文不再阐述,网上有很多,本次测试的eclipse已经进行了配置上面的优化。

准备环境

eclipse版本: Release 4.5.0

eclipse 默认配置: eclipse.ini

```
1 -startup
2 plugins/org.eclipse.equinox.launcher_1.3.100.v20150511-1540.jar
3 --launcher.library
4 plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.300.v20150602-1417
5 -product
6 org.eclipse.epp.package.jee.product
7 --launcher.defaultAction
8 openFile
9 --launcher.XXMaxPermSize
10 256M
11 -showsplash
12 org.eclipse.platform
13 --launcher.XXMaxPermSize
14 256m
15 --launcher.defaultAction
16 openFile
17 --launcher.appendVmargs
18 -vmargs
19 -Dosgi.requiredJavaVersion=1.7
20 -Xms256m
21 -Xmx1024m
```

在配置的末尾处添加如下配置文件:

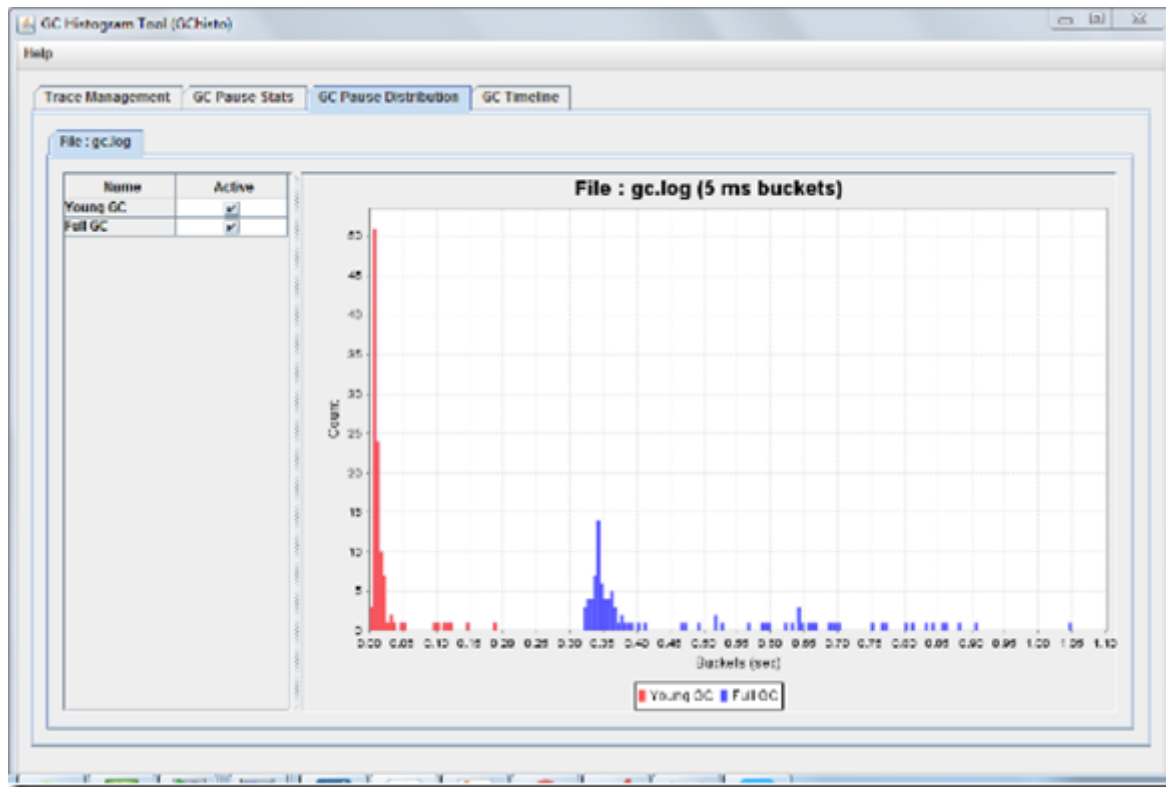
```
1 -XX:+PrintGCDetails // 输出GC的详细日志
2 -XX:+PrintGCDateStamps // 输出GC的时间戳 (以日期的形式)
3 -Xloggc:gc.log // 输出GC的详细日志
```

eclipse启动计时插件:

<http://www.chendd.cn/information/viewInformation/experienceShare/148.a>

GChisto.jar:gc日志分析工具jar包一个





Visual GC: java自带的内存监控工具，通过visual gc可以实时的监控到各个内存区域的变化。





如何分析GC日志

摘录GC日志一部分（绿色为年轻代gc回收；蓝色为full gc回收）：

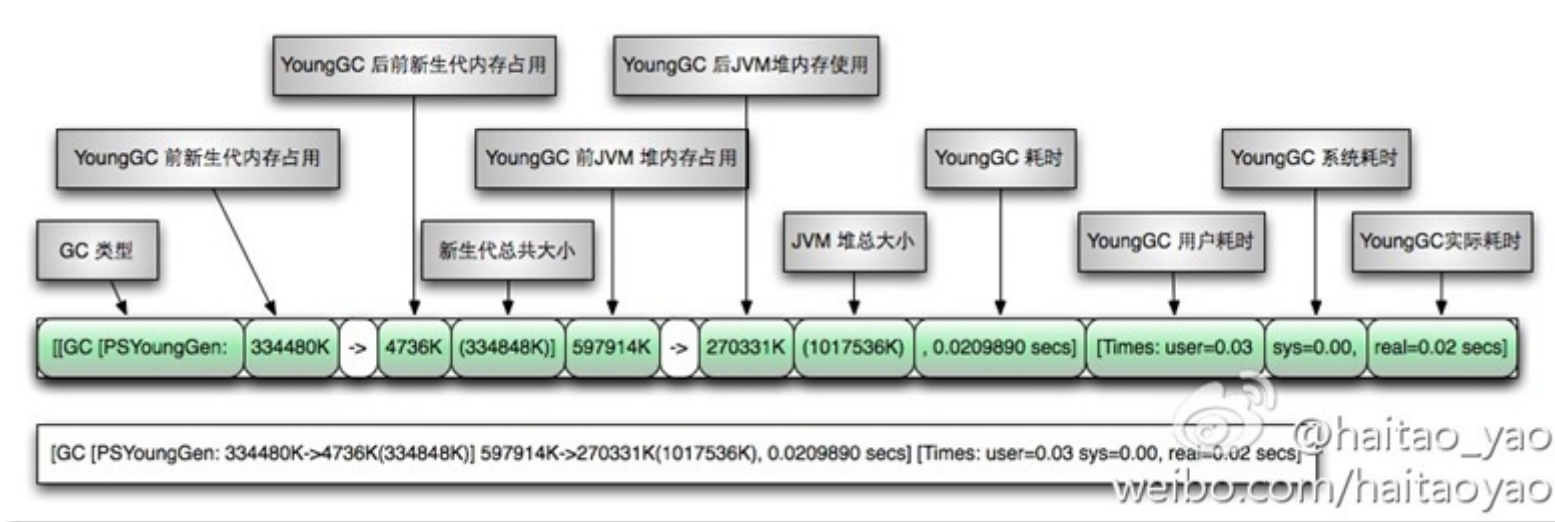
2016-07-05T10:43:18.093+0800: 25.395: [GC [PSYoungGen: 274931K->10738K(274944K)] 371093K->147186K(450048K), 0.0668480 secs] [Times: user=0.17 sys=0.08, real=0.07 secs]

2016-07-05T10:43:18.160+0800: 25.462: [Full GC [PSYoungGen: 10738K->0K(274944K)] [ParOldGen: 136447K->140379K(302592K)] 147186K->140379K(577536K) [PSPermGen: 85411K->85376K(171008K)], 0.6763541 secs] [Times: user=1.75 sys=0.02, real=0.68 secs]

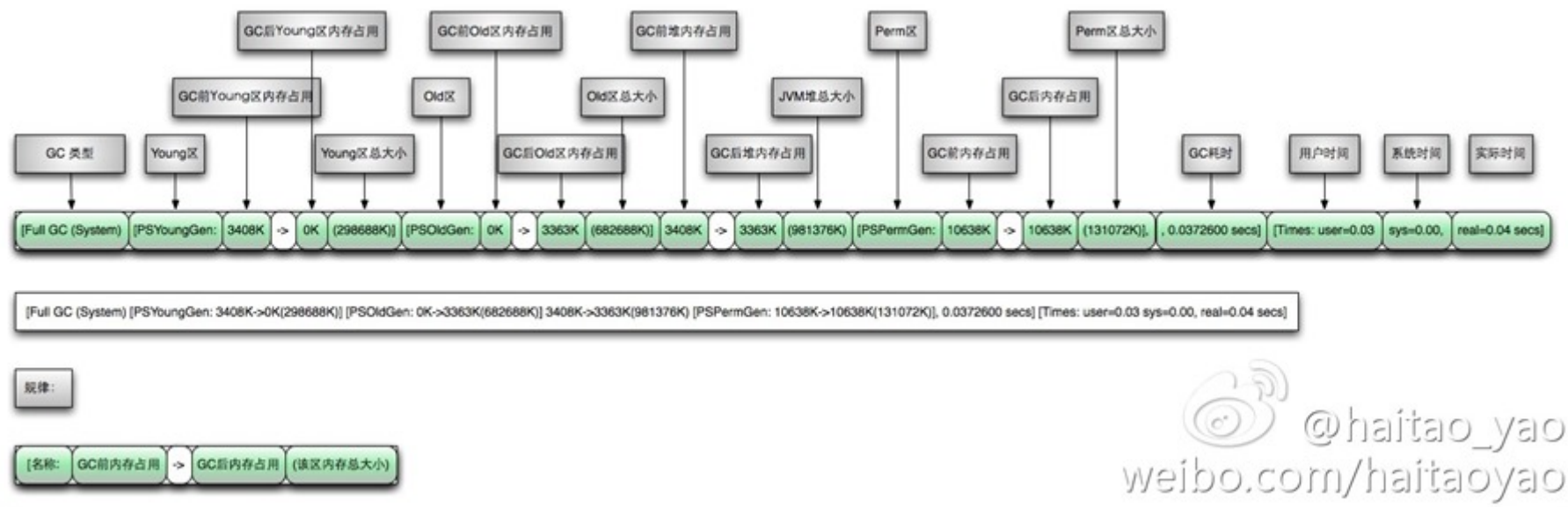
通过上面日志分析得出，PSYoungGen、ParOldGen、PSPermGen属于Parallel收集器。其中PSYoungGen表示gc回收前后年轻代的内存变化；ParOldGen表示gc回收前后老年代的内存变化；PSPermGen表示gc回收前后永久区的内存变化。young gc 主要是针对年轻代进行内存回收比较频繁，耗时短；full gc 会对整个堆内存进行回收，耗时长，因此一般尽量减少full gc的次数

通过两张图非常明显看出gc日志构成：

young gc 日志

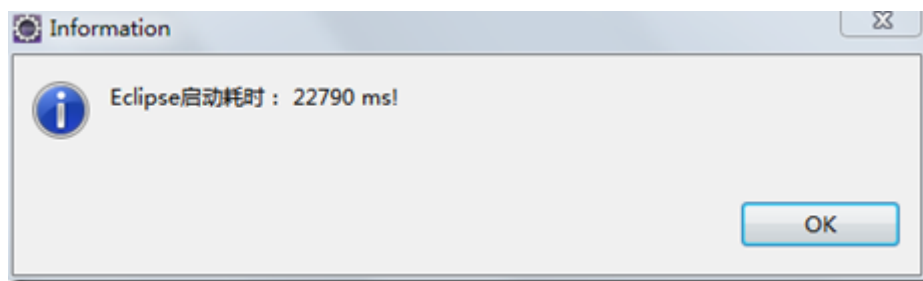


Full GC日志

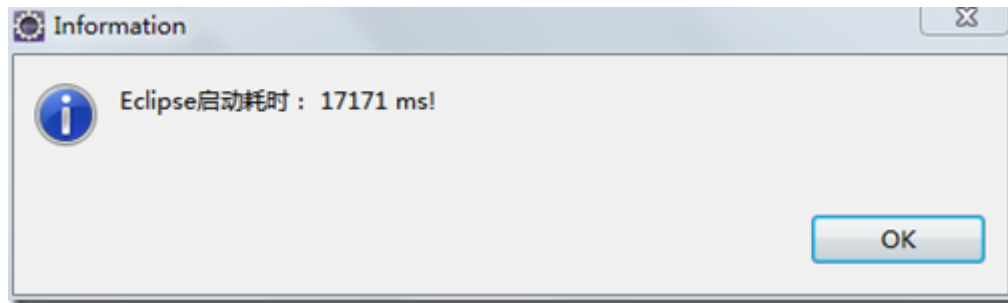


启动调优

启动eclipse查看默认配置下启动时间大概是22秒。



根据GChisto分析gc日志看出来, 启动过程中进行了一次full gc,19次minor gc;full gc和young gc的时间差不多都是0.65秒左右。



第一步优化:

为了避免内存频繁的动态扩展, 直接把-Xms配置和-Xmx一致, 修改如下:

-Xms1024m

修改完毕, 重新启动:

All GC Stats	Chart: Num	Chart: Total GC (sec)	Chart: Avg (ms)	Chart: Siz
	Num	Num (%)	Total GC (sec)	Total GC (%)
All	22	100.00%	0.915	100.00%
Young GC	22	100.00%	0.915	100.00%
Full GC	0	0.00%	0.000	0.00%

	Num	Num (%)	Total GC (sec)	Total GC (%)
All	7	100.00%	0.738	100.00%
Young GC	7	100.00%	0.738	100.00%
Full GC	0	0.00%	0.000	0.00%

启动时间缩小到17秒, 分析gc日志得出young gc22次, full gc没有了! 但是young gc增加了两次。

第二步优化:

因为本机的内存8G,给eclipse分配1g还是有点小了, 简单粗暴直接所有内存配置加倍。

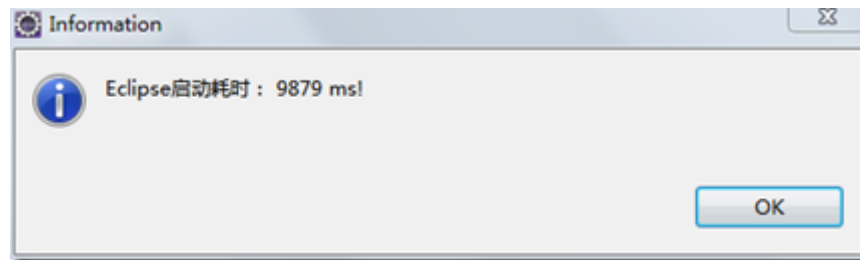
配置如下:

```
1 --launcher.XXMaxPermSize
2 512M
3 --launcher.XXMaxPermSize
4 512m
5 -Xms2048m
6 -Xmx2048m
```

启动时间缩小到15秒, 但是 young gc已经缩短到只有7次, 说明因为gc回收导致eclipse 启动慢的问题已经初步解决



第三步优化:



通过Visual GC看到在eclipse启动的时候classloader加载class的时间有一些，关闭字节码可能会优化一部分启动时间，加入如下参数：

-Xverify:none（关闭Java字节码验证，从而加快了类装入的速度）

重新启动测试,启动时间已经优化到了9秒！

查看启动日志，young gc 的次数仅仅只有了一次！

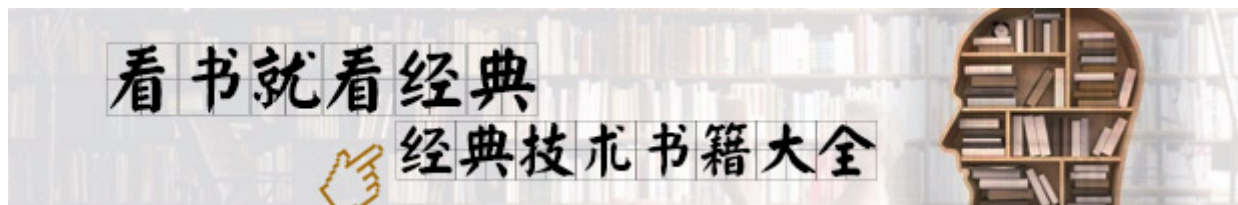
至此优化结束，附最终的eclipse.ini文件

```
1 -startup
2 plugins/org.eclipse.equinox.launcher_1.3.100.v20150511-1540.jar
3 --launcher.library
4 plugins/org.eclipse.equinox.launcher.win32.win32.x86_64_1.1.300.v20150602-1417
5 -product
6 org.eclipse.epp.package.jee.product
7 --launcher.defaultAction
8 openFile
9 --launcher.XXMaxPermSize
10 512M
11 -showsplash
12 org.eclipse.platform
13 --launcher.XXMaxPermSize
14 512m
15 --launcher.defaultAction
16 openFile
17 --launcher.appendVmargs
18 -vmargs
19 -Dosgi.requiredJavaVersion=1.7
20 -Xms2048m
21 -Xmx2048m
22 -Xverify:none
23 -XX:+PrintGCDetails
24 -XX:+PrintGCDateStamps
25 -Xloggc:gc.log
```



本系列:

- [JVM \(1\) : Java 类的加载机制](#)
- [JVM \(2\) : JVM内存结构](#)
- [JVM \(3\) : Java GC算法 垃圾收集器](#)
- [JVM \(4\) : Jvm调优-命令篇](#)
- [JVM \(5\) : tomcat性能调优和性能监控 \(visualvm\)](#)
- [JVM \(6\) : JVM调优-从eclipse开始](#)



相关文章

- [Java虚拟机 \(JVM\) 概述](#)
- [从JVM heap dump里查找没有关闭文件的引用](#)
- [使用 JITWatch 查看 JVM 的 JIT 编译代码](#)
- [JVM堆内存使用率持续上升的一种排查思路](#)
- [直播一次问题排查过程](#)
- [Java 虚拟机16: Metaspace](#)
- [Java 虚拟机 13: 互斥同步、锁优化及synchronized和volatile](#)
- [Java 虚拟机 12 : Java 内存模型](#)
- [Java 虚拟机 11 : 运行期优化](#)
- [Java 虚拟机10: 类加载器](#)



发表评论

Comment form

Name*

邮箱*

网站 (请以 http://开头)

评论内容*

(*) 表示必填项

[提交评论](#)

2 条评论

1. 郑敏说道:

[2017/07/18 上午 11:37](#)

写的不错! 看不到图

 0  0

[回复](#)

◦ 唐尤华说道:

[2017/07/24 下午 2:46](#)

感谢提醒, 已更新

 0  0

[回复](#)

[« JVM \(5 \) : Tomcat 性能调优和性能监控 \(visualvm \)](#)
[JVM \(7\) : JVM调优-工具篇 »](#)

Search for:



- [本周热门文章](#)
- [本月热门](#)
- [热门标签](#)



0 [内存屏障和 volatile 语义](#)

1 [SpringBoot | 第十七章: web ...](#)

2 [SpringBoot | 第十八章: web 应用开...](#)

3 [Java 线程池详解](#)

4 [JDK 源码阅读 : DirectByteBu...](#)

5 [Map 大家族的那点事儿 \(5 \) : We...](#)

6 [Map 大家族的那点事儿 \(6 \) : Lin...](#)





7 [Map 大家族的那点事儿 \(7 \) : Concu...](#)

8 [如果非得了解下 git 系统.....](#)

9 [SpringBoot | 第十九章: web 应用开发...](#)



最新评论

-  Re: [内存屏障和 volatile 语义](#)
会思考的作者 小宇宙
-  Re: [SpringBoot | 第十五章: 基于Pos...](#)
一直用postman www.wuliaokankan.cn
-  Re: [探究 Java 虚拟机栈](#)
不错 aa
- 



Re: [Java并发编程: CountdownLatch、CyclicB...](#)

> \"release()用来释放许可。注意, 在释放许可之前, 必须先获获得许可。\"Semapho... 苍穆



Re: [HashMap的工作原理](#)

那为什么不使用HashMap也要说清楚呀, 要不然稀里糊涂的 渔夫



Re: [并发编程 – Concurrency](#)

总结的很细致, 感谢作者! 落雨无声



Re: [做一次面向对象的体操: 将JSO...](#)

大侠, TransferUtil 和 Order 类没有, 能否贴出来, 学习学习。谢谢。 sailor




Re: [Map大家族的那点事儿\(1\): M...](#)

可以的 李红波



关于ImportNew



ImportNew 专注于 Java 技术分享。于2012年11月11日 11:11正式上线。是的, 这是一个很特别的时刻 :)

ImportNew 由两个 Java 关键字 import 和 new 组成, 意指: Java 开发者学习新知识的网站。import 可认为是学习和吸收, new 则可认为是新知识、新技术圈子和新朋友.....





联系我们

Email: ImportNew.com@gmail.com

新浪微博: [@ImportNew](https://weibo.com/ImportNew)

推荐微信号



反馈建议: ImportNew.com@gmail.com

广告与商务合作QQ: 2302462408

推荐关注

[小组](#) – 好的话题、有启发的回复、值得信赖的圈子

[头条](#) – 写了文章? 看干货? 去头条!

[相亲](#) – 为IT单身男女服务的征婚传播平台

[资源](#) – 优秀的工具资源导航

[翻译](#) – 活跃 & 专业的翻译小组

[博客](#) – 国内外的精选博客文章

[设计](#) – UI, 网页, 交互和用户体验

[前端](#) – JavaScript, HTML5, CSS

[安卓](#) – 专注Android技术分享

[iOS](#) – 专注iOS技术分享

[Java](#) – 专注Java技术分享

[Python](#) – 专注Python技术分享

© 2018 ImportNew

