jdk1.7 默认垃圾收集器Parallel Scavenge（新生代）+Parallel Old（老年代）

jdk1.8 默认垃圾收集器Parallel Scavenge（新生代）+Parallel Old（老年代）

jdk1.9 默认垃圾收集器G1

-XX:+PrintCommandLineFlagsjvm参数可查看默认设置收集器类型

-XX:+PrintGCDetails亦可通过打印的GC日志的新生代、老年代名称判断

JDK1.8新特性：

速度更快 – 红黑树  
代码更少 – Lambda  
强大的Stream API – Stream  
便于并行 – Parallel  
最大化减少空指针异常 – Optional

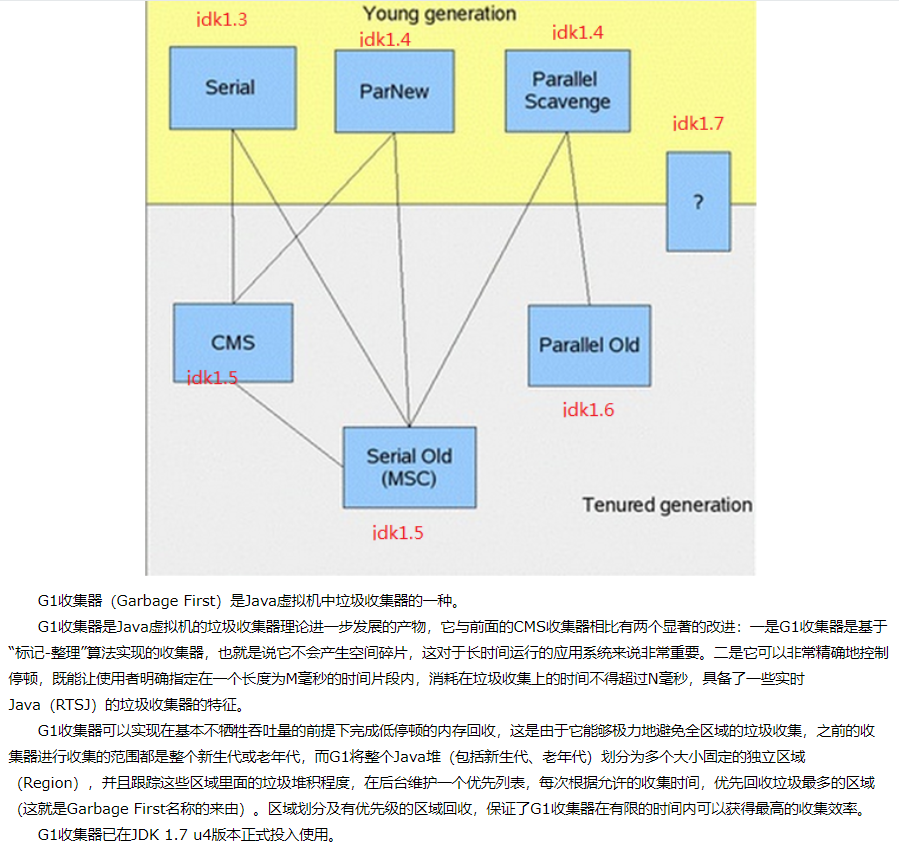
堆  
-Xms starting堆初始大小  
-Xmx max堆最大容量  
-Xmn new新生代大小

栈  
-Xss 设置每个线程的栈大小  
线程栈的大小是个双刃剑，如果设置过小，可能会出现栈溢出，特别是在该线程内有递归、大的循环时出现溢出的可能性更大，如果该值设置过大，就有影响到创建栈的数量，如果是多线程的应用，就会出现内存溢出的错误。  
JVM可创建的最大线程数限制因素：线程栈大小——》进程的最大内存——》操作系统位数

JVM种类：  
Hotspot, JRockit(Oracle)  
J9, JikesRVM(IBM)  
Zulu, Zing (Azul)

各版本默认垃圾收集器  
JDK7 Parallel Scavenge（新生代）+Parallel Old（老年代）  
JDK8 Parallel Scavenge（新生代）+Parallel Old（老年代）  
JDK9 G1

作者：晚歌y  
链接：https://www.jianshu.com/p/f25ec35f2c49  
来源：简书  
著作权归作者所有。商业转载请联系作者获得授权，非商业转载请注明出处。



ZGC JDK11实验性加入，不支持类卸载，JDk12支持支持类卸载，优化缩短暂停时间，控制在10MS，Zing VM的媲美C4垃圾收集器

在jdk 12 中实验性的引入了Shenandoah 垃圾收集器(openJDk，可选，默认还是G1)

**ZGC**

听说你是 zerpo paused GC?  
Java 11引入了ZGC，宣称暂停时间不超过10ms,支持 4TB,JDK13 到了 16TB!

和内存无关,TB 级也只停顿 1-10ms

Z Garbage Collector（ZGC）是在JAVA 11中引入的垃圾回收器，但一直都是实验版本，在JDK 15中，终于要上线了。

ZGC是一个重新设计的并发的垃圾回收器，可以极大的提升GC的性能。

# JEP 379: 新的垃圾回收器Shenandoah上线

Shenandoah和ZGC一样，是以实验特性在JAVA 12中引入的JEP 189。现在终于要在JAVA 15中转正了

JDK9-JDK14 默认都是G1

C:\Users\15164

λ java -XX:+PrintGCDetails -version

[0.004s][warning][gc] -XX:+PrintGCDetails is deprecated. Will use -Xlog:gc\* instead.

[0.029s][info ][gc,heap] Heap region size: 1M

[0.038s][info ][gc ] Using G1

[0.041s][info ][gc,heap,coops] Heap address: 0x0000000081c00000, size: 2020 MB, Compressed Oops m

java version "9.0.4"

Java(TM) SE Runtime Environment (build 9.0.4+11)

Java HotSpot(TM) 64-Bit Server VM (build 9.0.4+11, mixed mode)

[0.159s][info ][gc,heap,exit ] Heap

[0.160s][info ][gc,heap,exit ] garbage-first heap total 131072K, used 2048K [0x0000000081c0000

0400, 0x0000000100000000)

[0.161s][info ][gc,heap,exit ] region size 1024K, 3 young (3072K), 0 survivors (0K)

-Xlog:gc\*

今天面试问道JDK8默认使用的垃圾收集器是什么，然后回来第一时间CMD命令查看了一下：

java -XX:+PrintCommandLineFlags -version

-XX:InitialHeapSize=266390080 -XX:MaxHeapSize=4262241280 -XX:+PrintCommandLineFlags

-XX:+UseCompressedClassPointers -XX:+UseCompressedOops

-XX:-UseLargePagesIndividualAllocation -XX:+UseParallelGC

java version "1.8.0\_191"

Java(TM) SE Runtime Environment (build 1.8.0\_191-b12)

Java HotSpot(TM) 64-Bit Server VM (build 25.191-b12, mixed mode)