2、垃圾回收器和内存分配策略

T A H N K Y O U F O R W A T C H I N G

主讲老师Mark：446106311

课程咨询安生老师：669100976

垃圾收集器与内存分配策略

- GC要做的事

1、Where/Which ?

2、When ?

3、How ?

- 为什么我们要去了解GC和内存分配？

- 谁需要GC？

TO BE OR NOT TO BE-判断对象的存活

- **引用计数算法**
- **可达性分析**

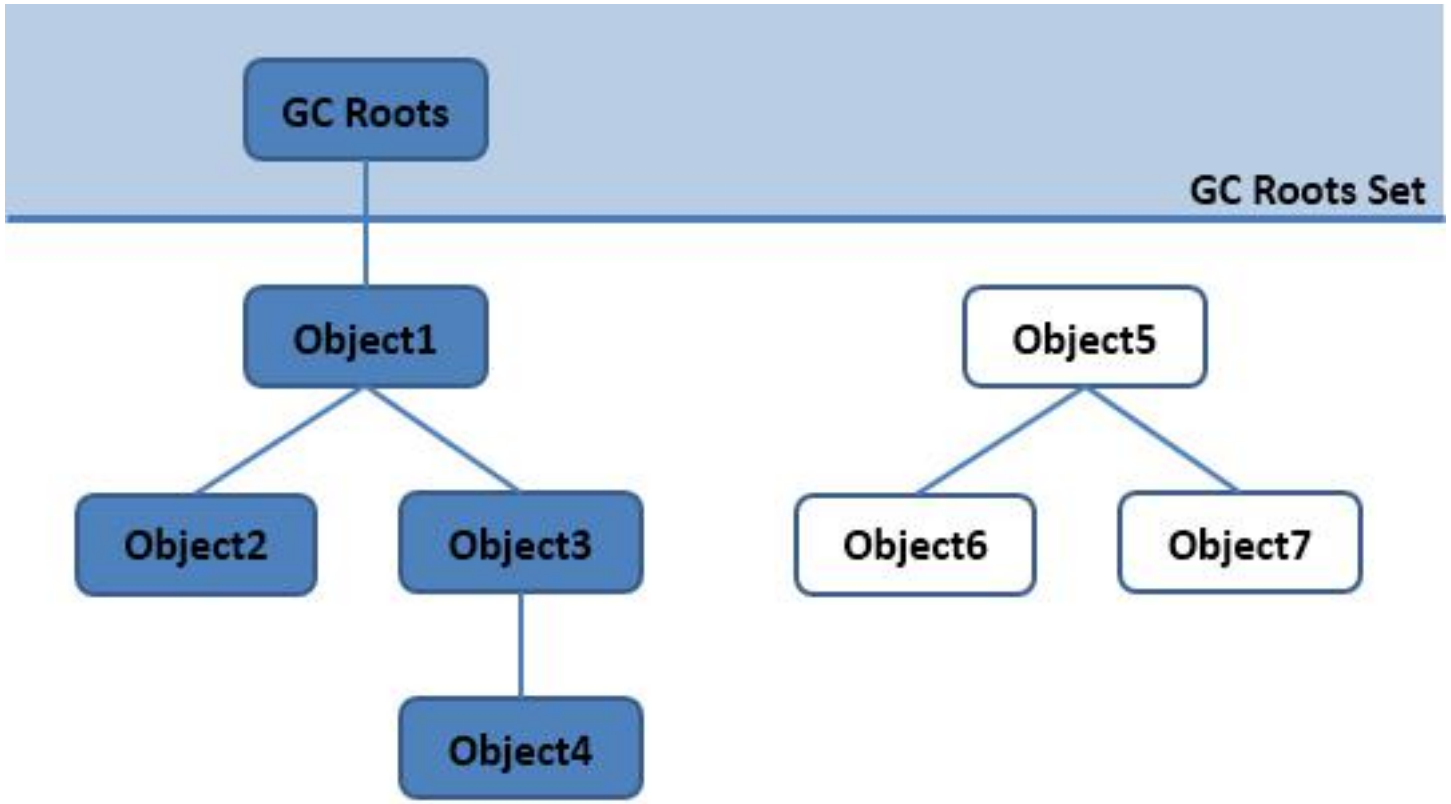
在Java, 可作为**GC Roots**的对象包括:

1.方法区: 类静态属性引用的对象;

2.方法区: 常量引用的对象;

3.虚拟机栈(本地变量表)中引用的对象.

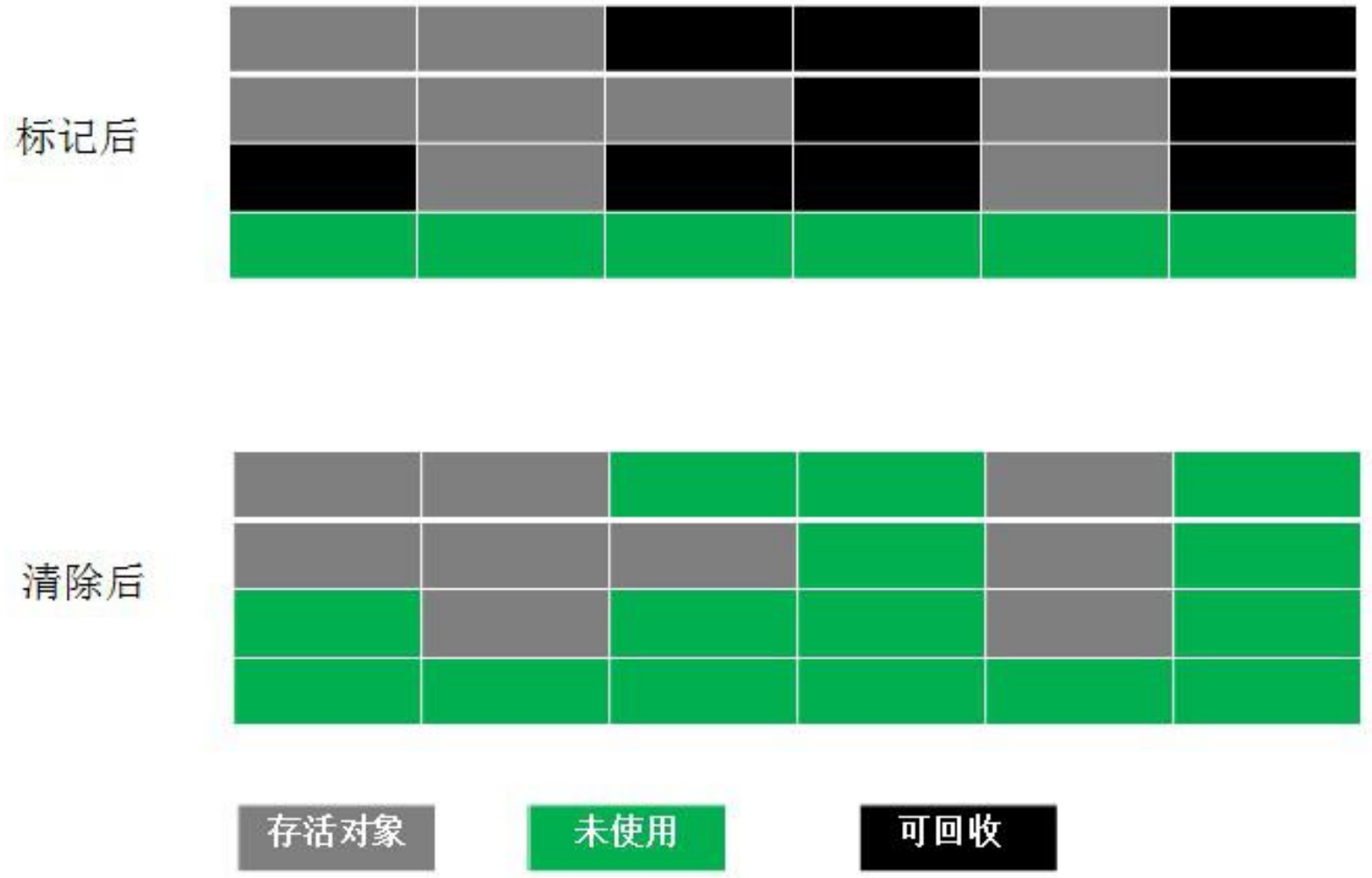
4.本地方法栈JNI(Native方法)中引用的对象。



各种引用

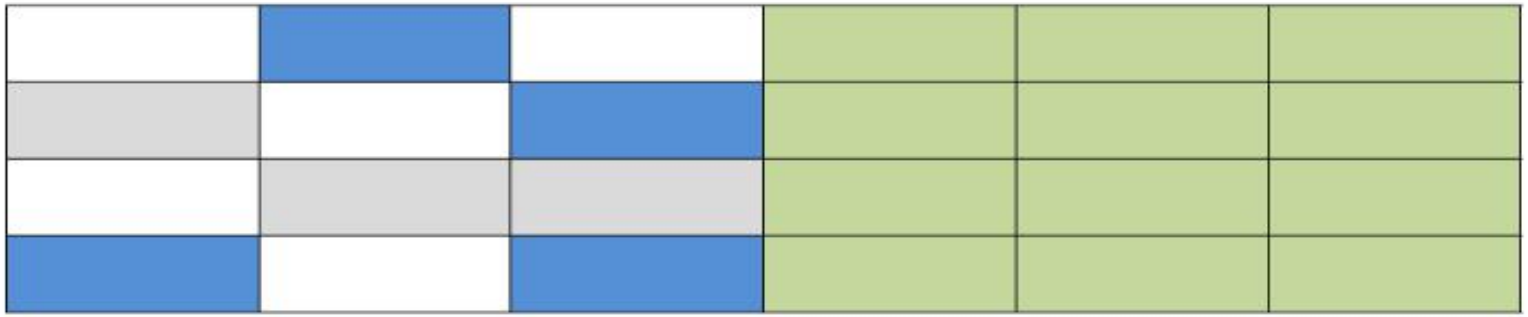
- 强引用
- 软引用 SoftReference
- 弱引用 WeakReference
- 虚引用 PhantomReference

标记-清除算法（Mark-Sweep）

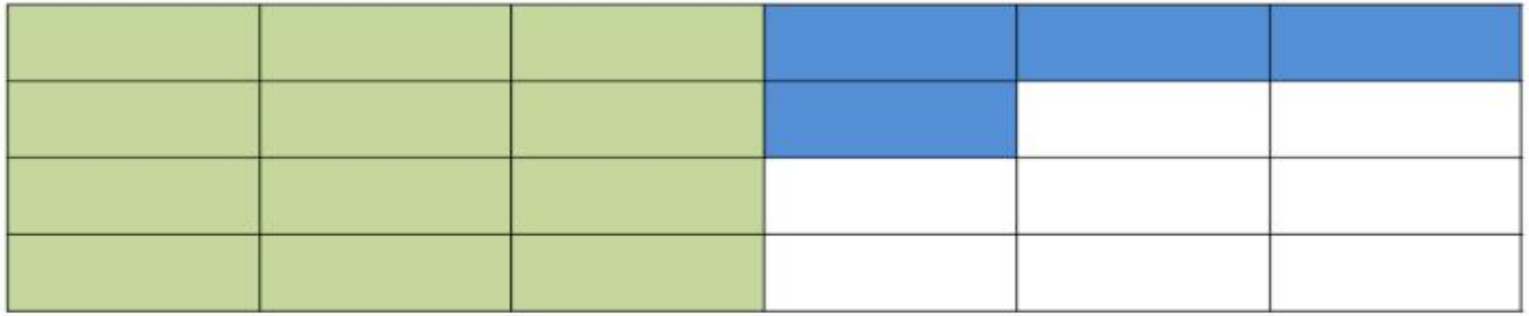


复制算法（Copying）

内存整理前



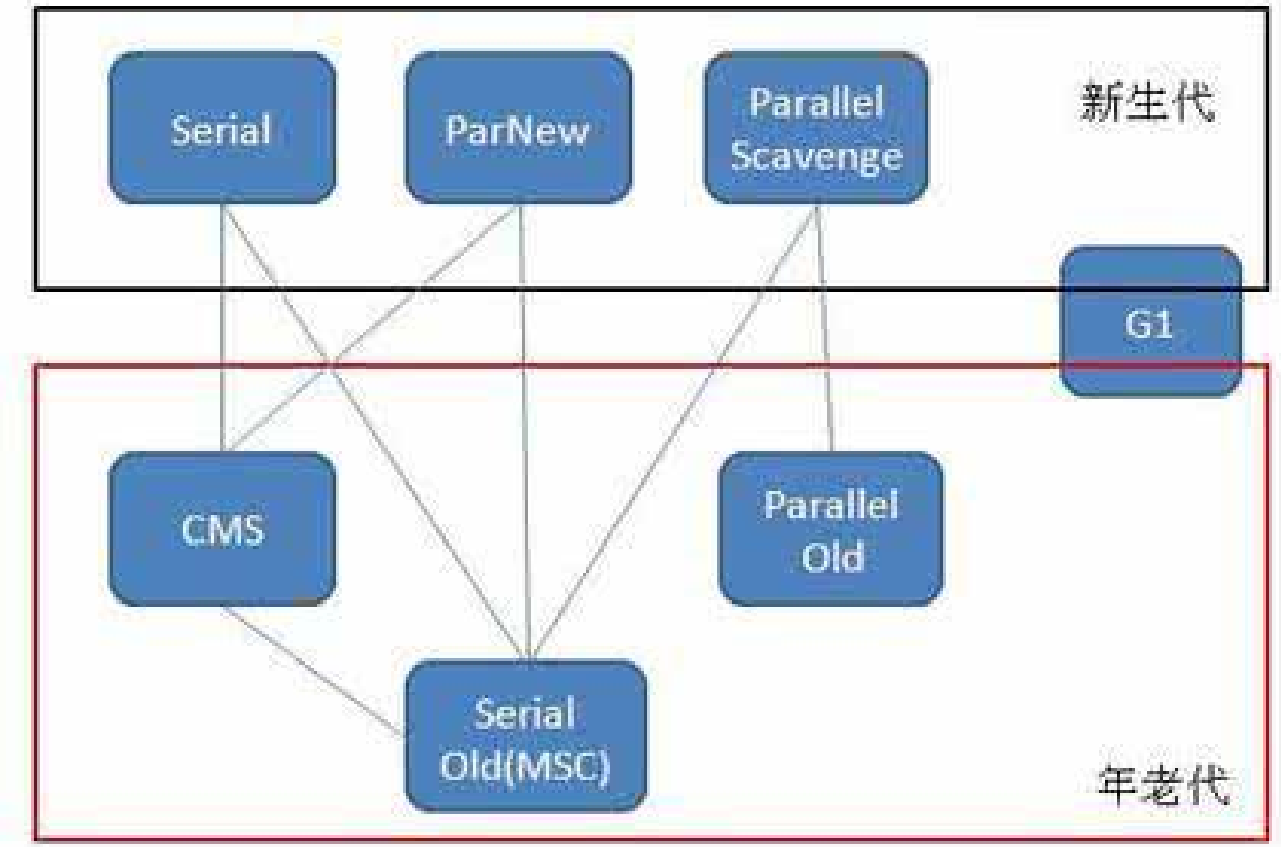
内存整理后



标记-整理算法（Mark-Compact）

把算法们都用上

- **分代收集**



垃圾回收器列表

收集器	收集对象和算法	收集器类型	说明	适用场景
Serial	新生代，复制算法	单线程	进行垃圾收集时，必须暂停所有工作线程，直到完成；(stop the world)	简单高效； 适合内存不大的情况；
ParNew	新生代，复制算法	并行的多线程收集器	ParNew垃圾收集器是Serial收集器的多线程版本	搭配CMS垃圾回收器的首选
Parallel Scavenge 吞吐量优先收集器	新生代，复制算法	并行的多线程收集器	类似ParNew，更加关注吞吐量，达到一个可控制的吞吐量；	本身是Server级别多CPU机器上的默认GC方式，主要适合后台运算不需要太多交互的任务；

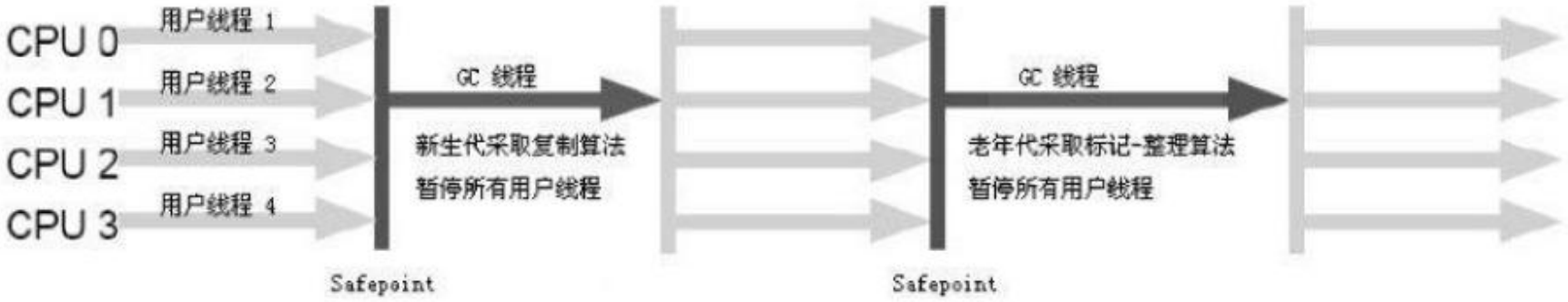
注： 吞吐量=运行用户代码时间/(运行用户代码时间+ 垃圾收集时间)
垃圾收集时间= 垃圾回收频率 * 单次垃圾回收时间

垃圾回收器列表

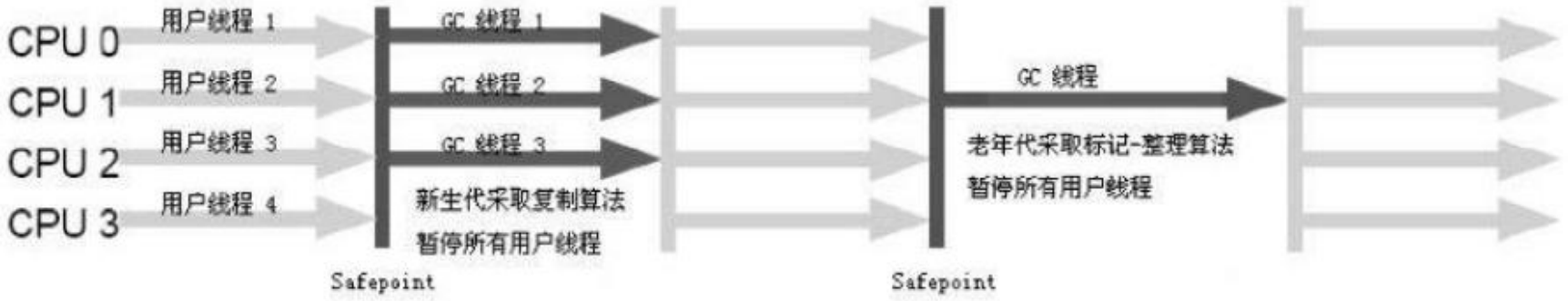
收集器	收集对象和算法	收集器类型	说明	适用场景
Serial Old	老年代，标记整理算法	单线程	jdk7/8默认的老生代垃圾回收器	Client模式下虚拟机使用
Parallel Old	老年代，标记整理算法	并行的多线程收集器	Parallel Scavenge收集器的老年代版本，为了配合Parallel Scavenge的面向吞吐量的特性而开发的对应组合；	在注重吞吐量以及CPU资源敏感的场合采用
CMS	老年代，标记清除算法	并行与并发收集器	尽可能的缩短垃圾收集时用户线程停止时间；缺点在于： 1.内存碎片 2.需要更多cpu资源 3.浮动垃圾问题，需要更大的堆空间	重视服务的响应速度、系统停顿时间和用户体验的互联网网站或者B/S系统。互联网后端目前cms是主流的垃圾回收器；
G1	跨新生代和老年代；标记整理 + 化整为零	并行与并发收集器	JDK1.7才正式引入，采用分区回收的思维，基本不牺牲吞吐量的前提下完成低停顿的内存回收；可预测的停顿是其最大的优势；	面向服务端应用的垃圾回收器，目标为取代CMS

垃圾回收器工作示意图

- Serial/Serial Old收集器运行示意图

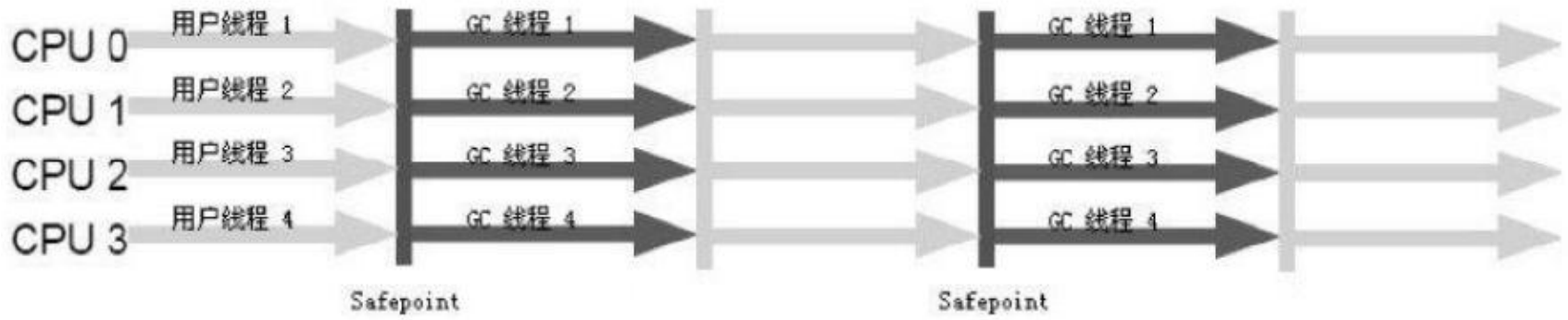


- ParNew/Serial Old收集器运行示意图

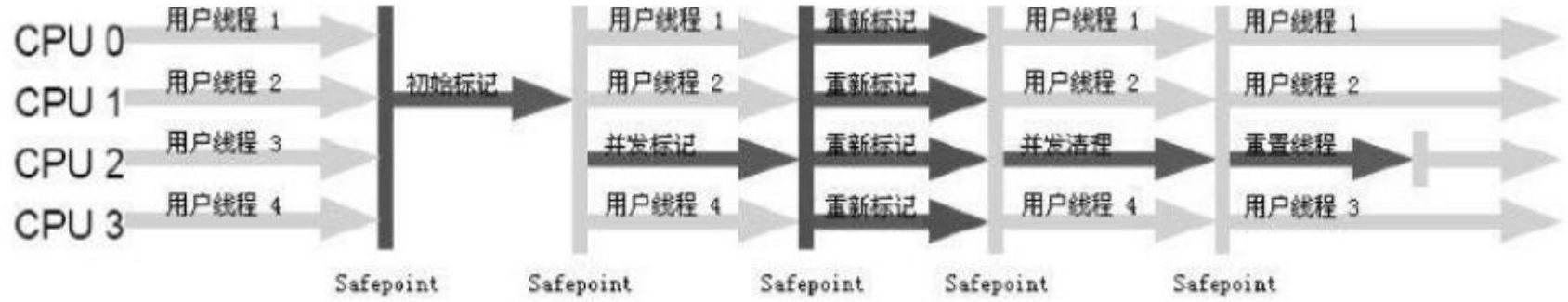


垃圾回收器工作示意图

- Parallel Scavenge/Parallel Old收集器运行示意图



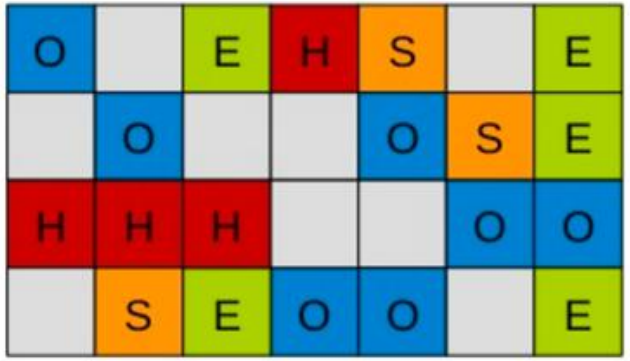
- Concurrent Mark Sweep收集器运行示意图

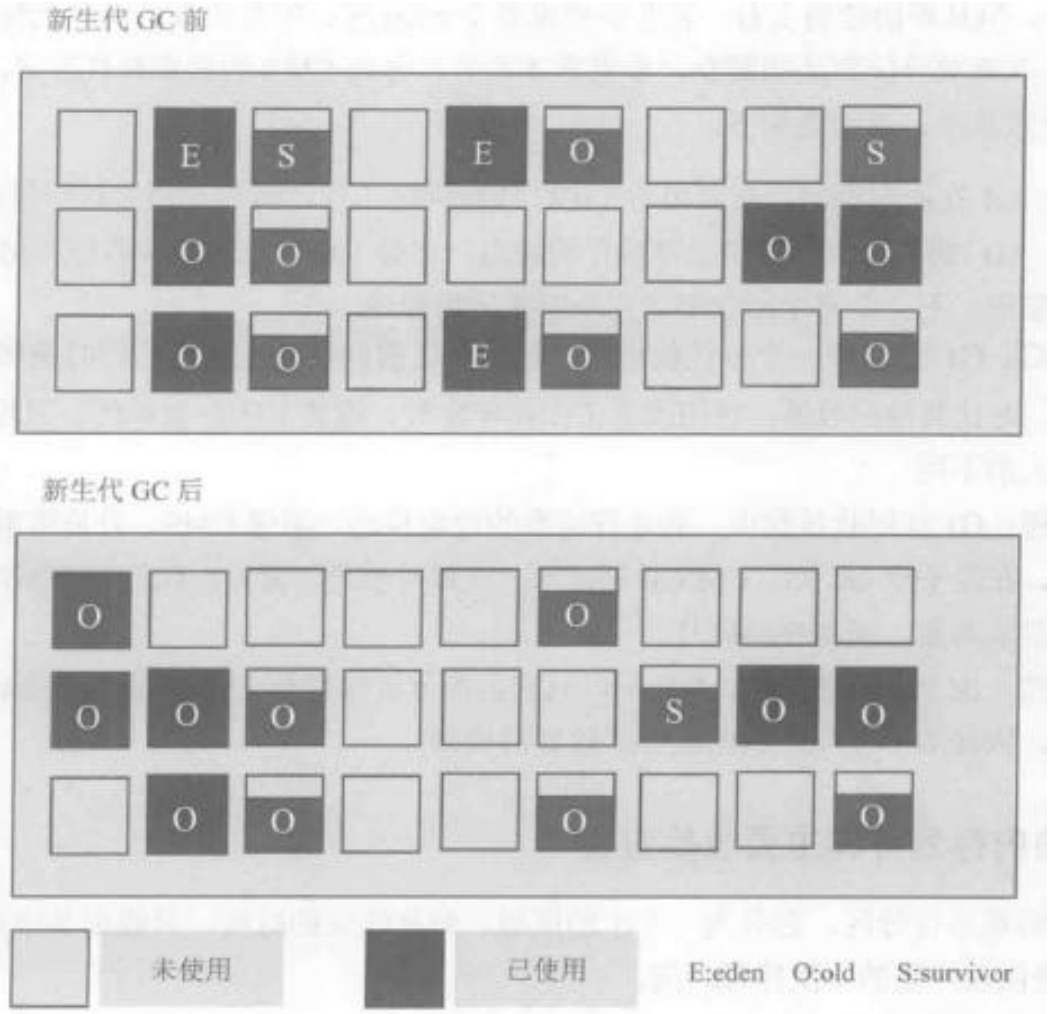


G1图示

G1收集的几个阶段

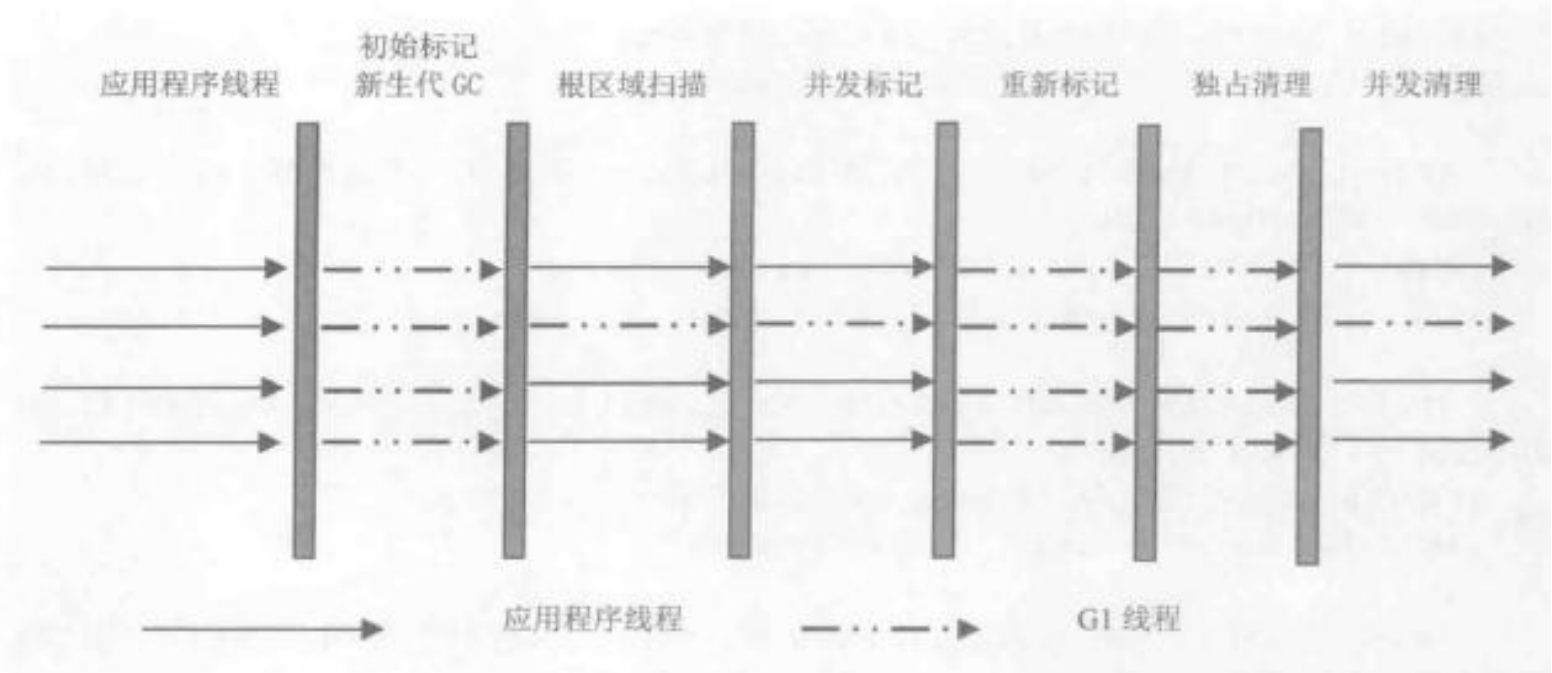
- 新生代GC
- 并发标记周期
- 混合收集
- 可能的FullGC





G1图示

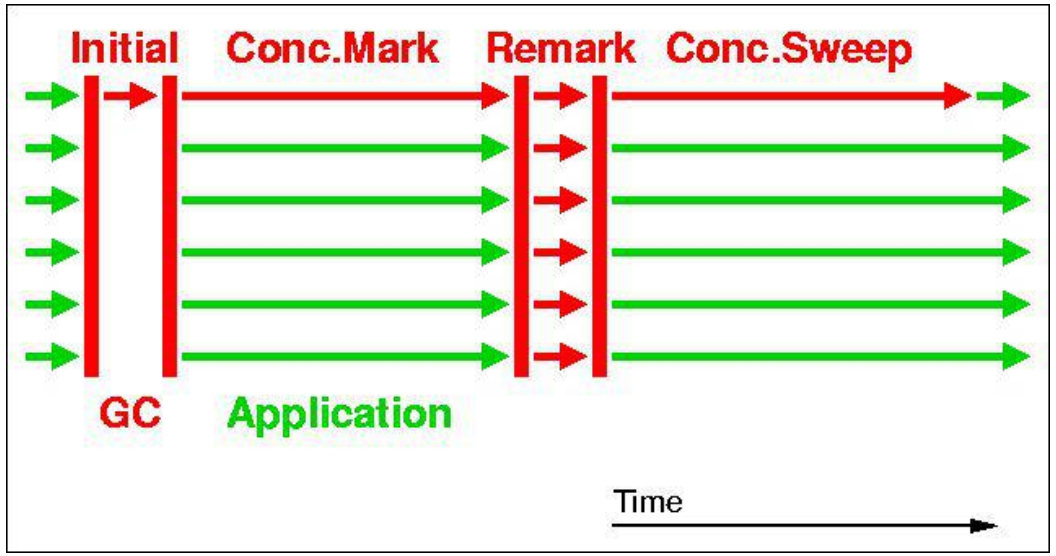
- G1收集器并发标记运行示意图

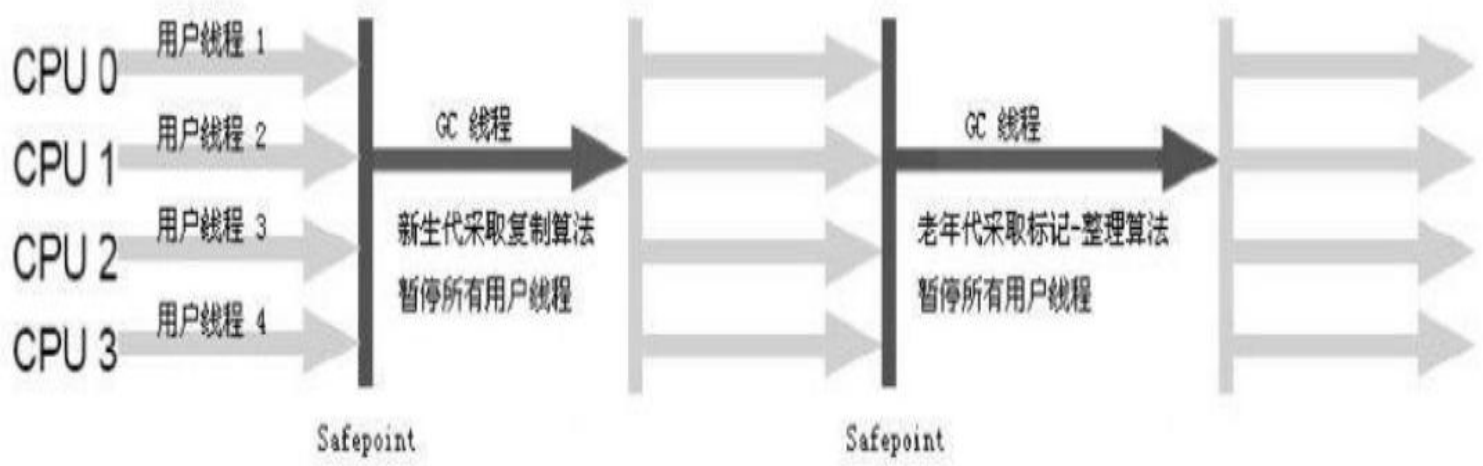


未来的垃圾回收

- JDK 11中的ZGC-一种可扩展的低延迟垃圾收集器
- 处理TB量级的堆
- GC时间不超过10ms
- 与使用G1相比，应用吞吐量的降低不超过15%

Stop The World现象





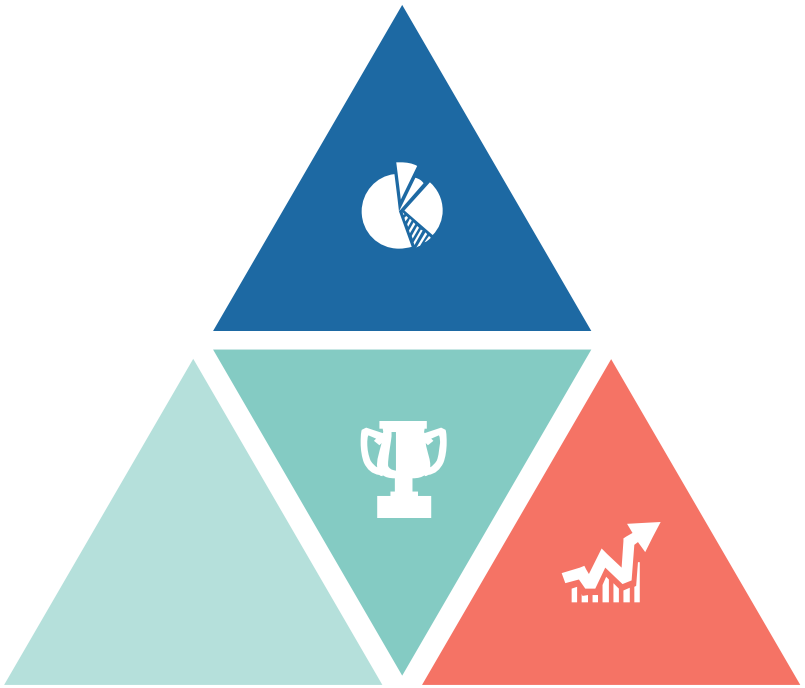
内存分配与回收策略

- 对象优先在Eden分配
- 大对象直接进入老年代
- 长期存活的对象将进入老年代
- 动态对象年龄判定
- 空间分配担保

内存泄漏和内存溢出辨析

- 相同与不同

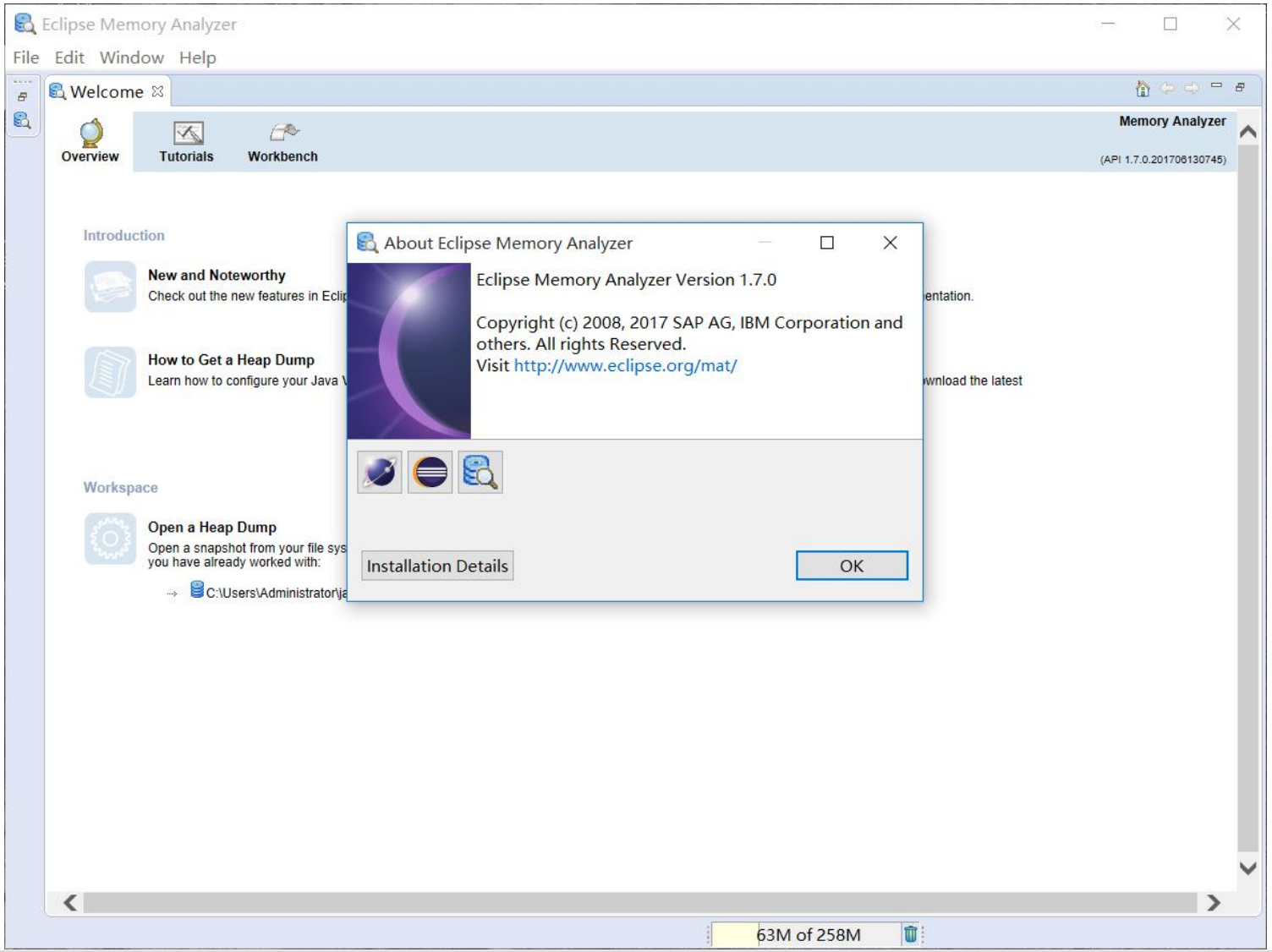
- 如何避免内存泄漏



JDK为我们提供的工具

名称	作用
jps	虚拟机进程状况工具
jstat	虚拟机统计信息监视工具
jinfo	Java配置信息工具
jmap	Java内存映像工具
~~jhat~~	虚拟机堆转储快照分析工具
jstack	Java堆栈跟踪工具
JConsole	Java监视与管理控制台
VisualVM	多合一故障处理工具

了解MAT


Eclipse Memory Analyzer
File Edit Window Help
Welcome
Overview
Tutorials
Workbench
Memory Analyzer
(API 1.7.0.201706130745)
Introduction
New and Noteworthy
How to Get a Heap Dump
Workspace
Open a Heap Dump
About Eclipse Memory Analyzer
Eclipse Memory Analyzer Version 1.7.0
Copyright (c) 2008, 2017 SAP AG, IBM Corporation and others. All rights Reserved.
Visit http://www.eclipse.org/mat/
Installation Details
OK
63M of 258M