JVM初探

体系结构

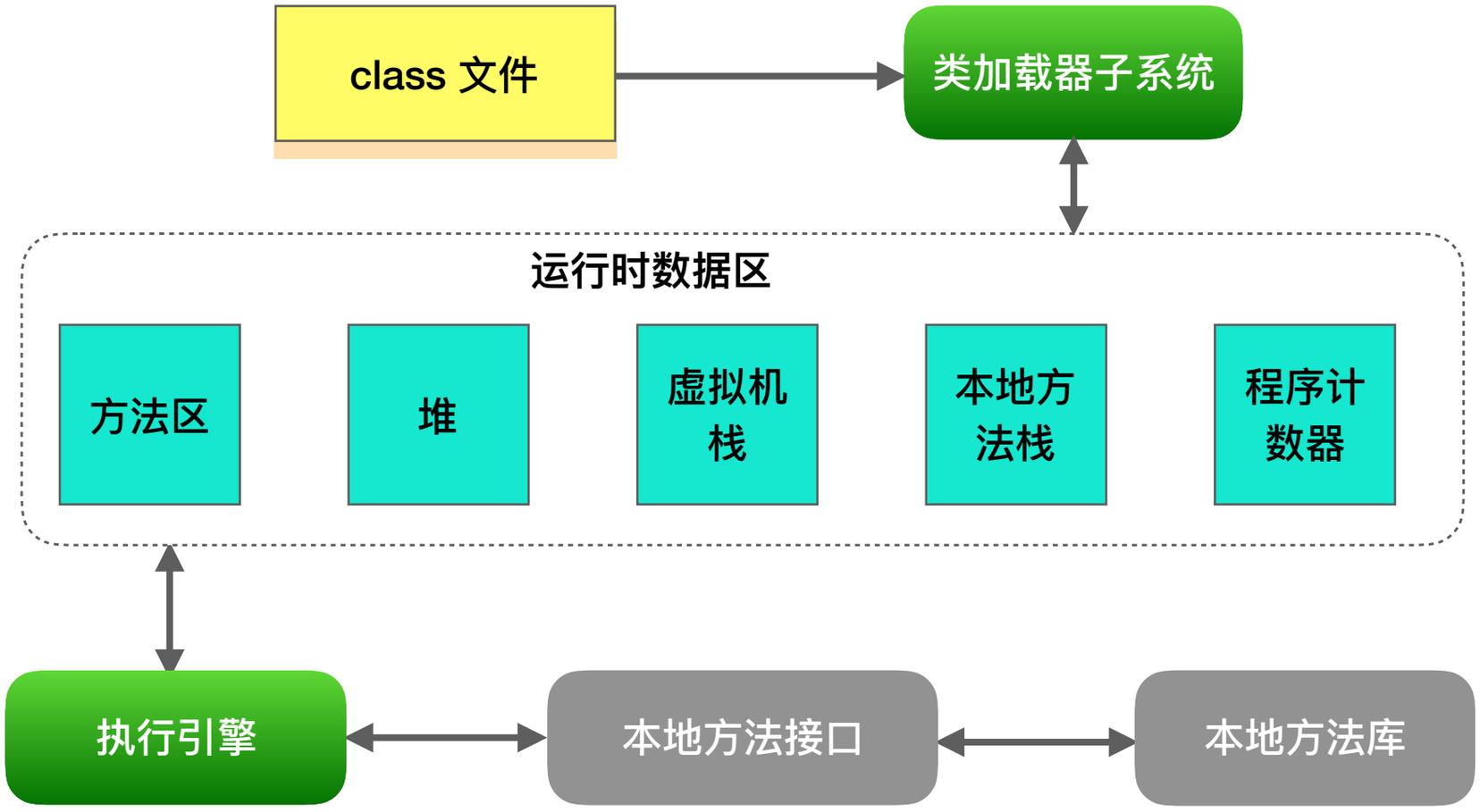
类加载子系统

内存模型

GC

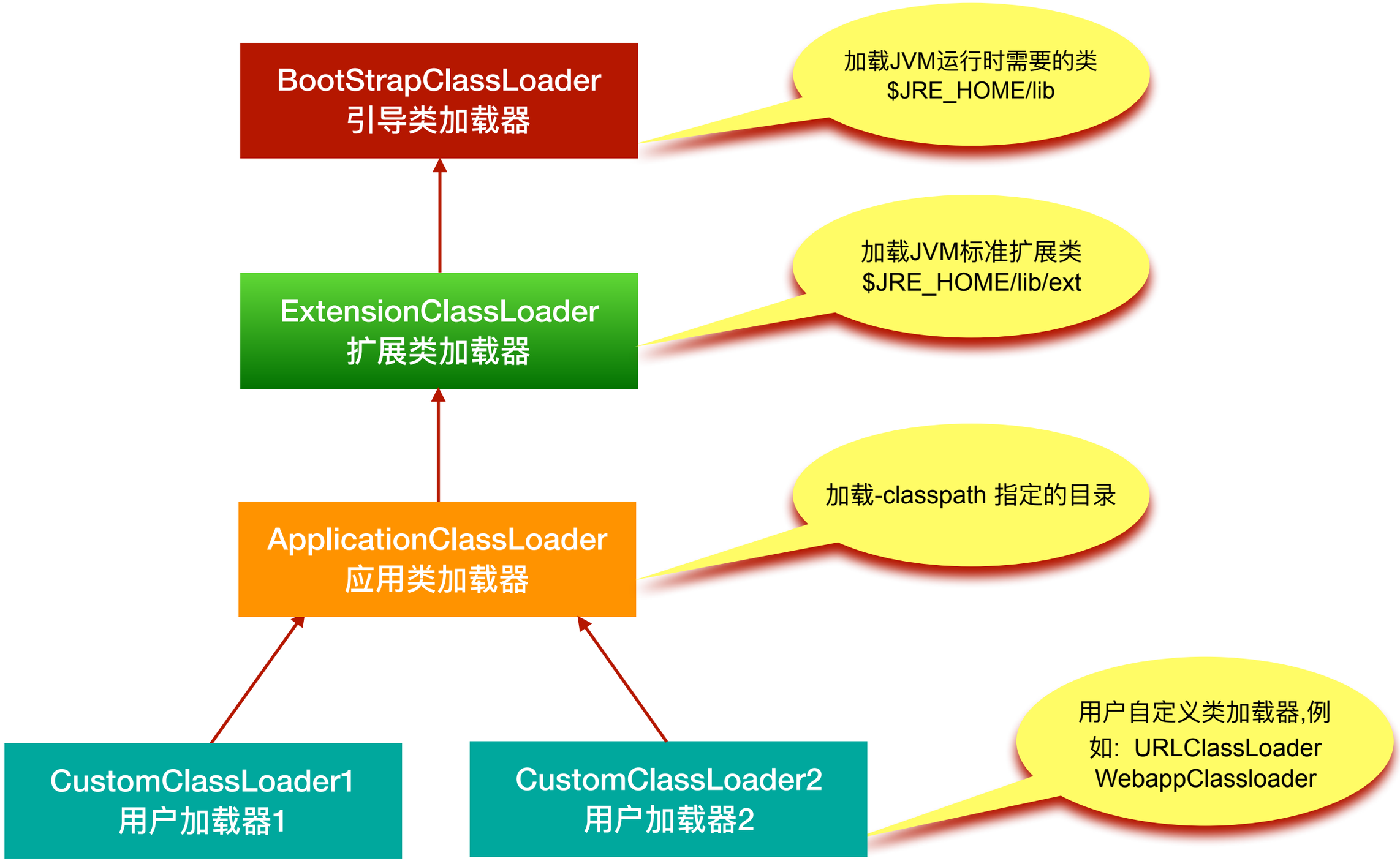
常用工具

体系结构

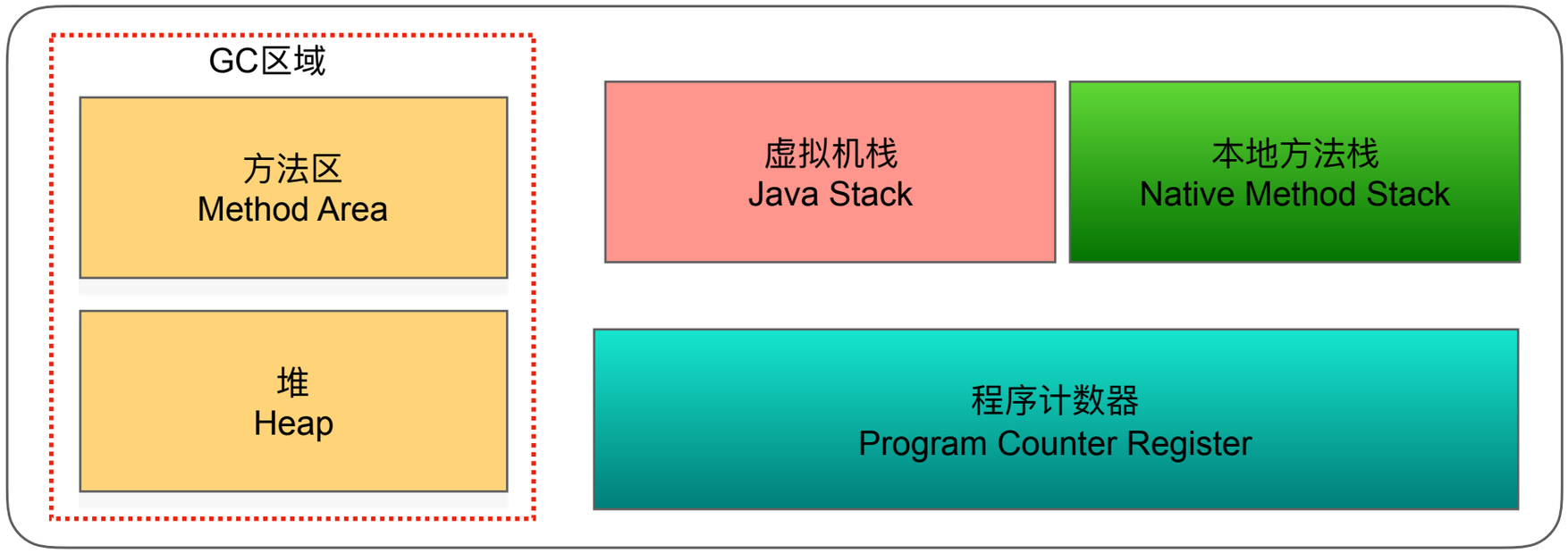


- 类加载器子系统: JVM启动时或者是在运行时将需要的class文件加载到JVM中
- 运行时数据区：JVM运行时内存空间的组织
- 执行引擎：用于执行JVM字节码指令

类加载器


BootStrapClassLoader
引导类加载器
加载JVM运行时需要的类
$JRE_HOME/lib
ExtensionClassLoader
扩展类加载器
加载JVM标准扩展类
$JRE_HOME/lib/ext
ApplicationClassLoader
应用类加载器
加载-classpath 指定的目录
CustomClassLoader1
用户加载器1
CustomClassLoader2
用户加载器2
用户自定义类加载器,例
如：URLClassLoader
WebappClassloader

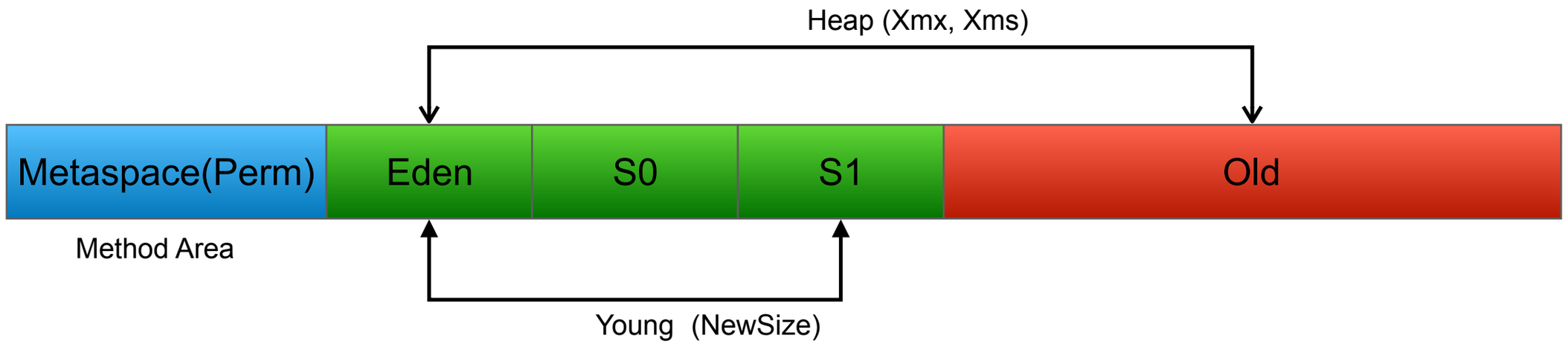
内存模型



- 方法区: 所有线程共享,
- 堆：所有线程共享，存放对象实例, GC的主要区域
- 虚拟机栈: 线程私有, 主要存储局部变量表 , 方法入口等
- 本地方法栈：线程私有, 与虚拟机栈类似, 只是为native方法服务
- 程序计数器: 线程私有, 当前线程所执行的字节码指令的行号指示器

内存管理

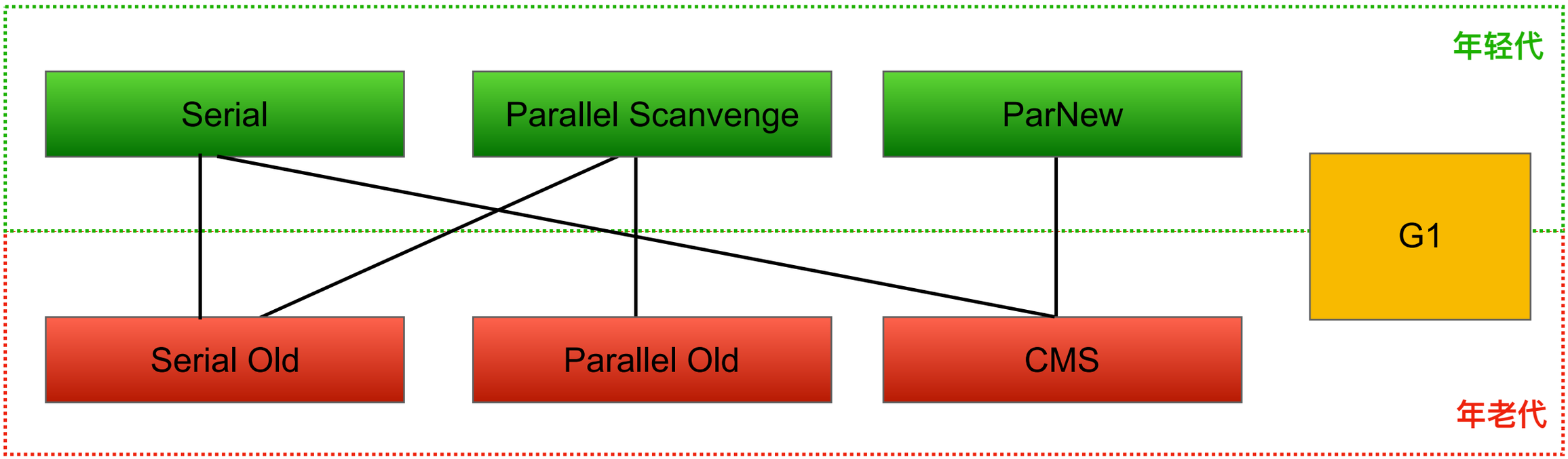
分代管理



1. 为什么?
   - 不同的对象生命周期不同, 大部份是临时对象, 朝生夕死
   - 根据不同的代, 采用不同的GC算法, 提高GC效率

2. GC 类型
   - Minor GC
   - Major/Full GC

GC



1. 年轻代使用复制算法, 分配对象时, Eden 空间不足时触发
   - 复制Eden和From(S0,S1)中存活对象到To(S1,S0)
   - 部分对象会晋升到Old区
   - 清空Eden和From; From和To交换，直到下次GC
2. Serial Old: 使用标记-压缩算法,串行，独占式垃圾回收器, STW 时间长
3. Parallel Old: 使用标记-压缩算法，关注系统吞吐率
4. CMS：并发-标记-清除算法，获取最短停顿时间

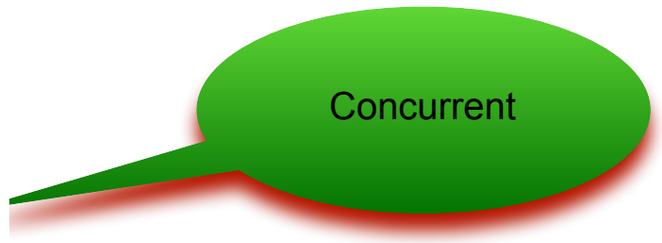
CMS(concurrent mark sweep)

Initial Mark

标记Root可以直达的对象，耗时短

Concurrent Mark

从第一步标记的对象出发，并发标记可达的对象

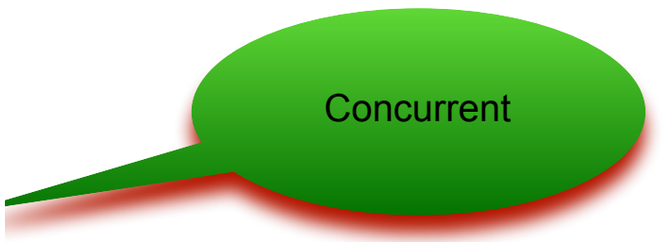


Remark

重新标记, 修正并发标记期间因用户程序继续运行导致的对象关系间变化及新创建的对象

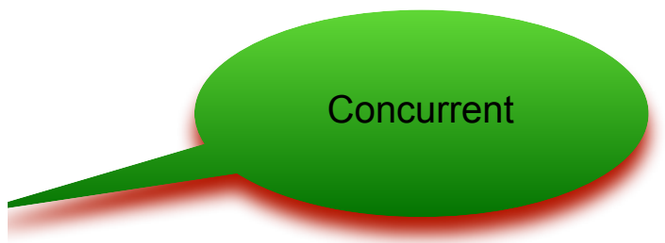
Concurrent Sweep

并行的进行无用对象的回收



Concurrent Reset

并发重置,为下一次gc做好准备



CMS日志

JVM启动后经历的时间(秒)

年老代当前使用情况

整个堆当前使用情况

```
954235.276: [GC (CMS Initial Mark) [1 CMS-initial-mark: 1048518K(1048576K)] 1900944K(1992320K), 0.1274726 secs] [Times: user=0.24 sys=
0.00, real=0.13 secs]
954235.404: [CMS-concurrent-mark-start]
954235.627: [CMS-concurrent-mark: 0.223/0.223 secs] [Times: user=0.42 sys=0.01, real=0.22 secs]
954235.627: [CMS-concurrent-preclean-start]
954235.722: [CMS-concurrent-preclean: 0.090/0.095 secs] [Times: user=0.17 sys=0.00, real=0.10 secs]
954235.722: [CMS-concurrent-abortable-preclean-start]
954235.722: [CMS-concurrent-abortable-preclean: 0.000/0.000 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]
954235.738: [GC (CMS Final Remark) [YG occupancy: 907298 K (943744 K)]954235.738: [Rescan (parallel) , 0.1801212 secs]954235.918: [wea
k refs processing, 0.0000435 secs]954235.918: [class unloading, 0.0174879 secs]954235.936: [scrub symbol table, 0.0096392 secs]954235.
945: [scrub string table, 0.0010738 secs][1 CMS-remark: 1048518K(1048576K)] 1955816K(1992320K), 0.2085769 secs] [Times: user=0.35 sys=
0.00, real=0.20 secs]
954235.947: [CMS-concurrent-sweep-start]
954235.992: [CMS-concurrent-sweep: 0.045/0.045 secs] [Times: user=0.09 sys=0.01, real=0.05 secs]
954235.992: [CMS-concurrent-reset-start]
954235.995: [CMS-concurrent-reset: 0.003/0.003 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]
```

年轻代当前使用情况

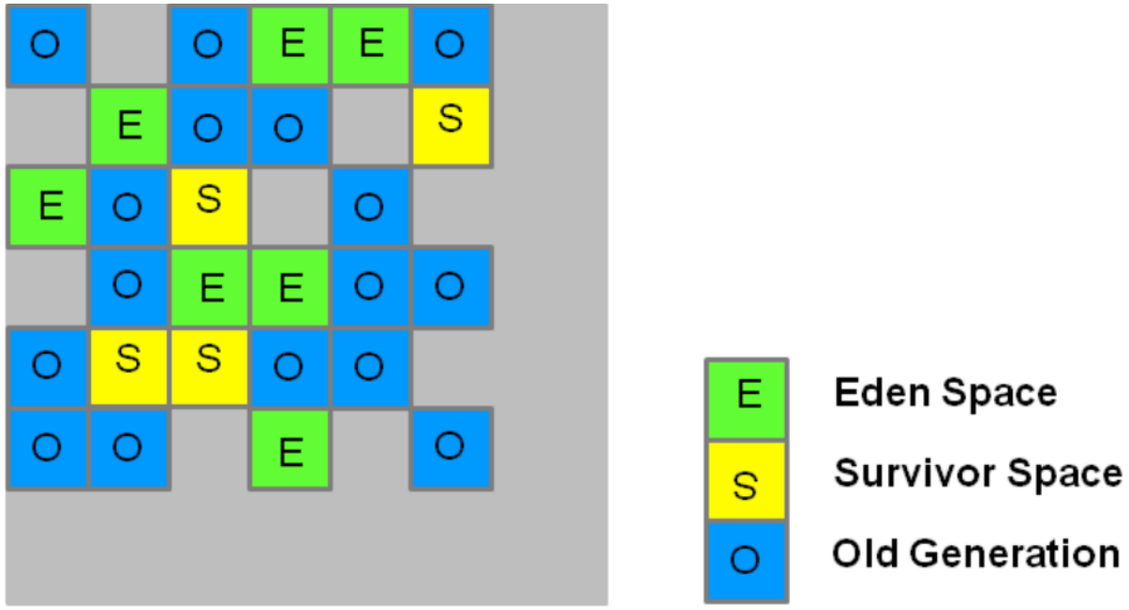
```
954232.983: [GC (Allocation Failure) 954232.983: [ParNew: 943743K->943743K(943744K), 0.0000203 secs]954232.983: [CMS954233.003: [CMS-c
oncurrent-mark: 0.160/0.167 secs] [Times: user=0.31 sys=0.00, real=0.17 secs]
 (concurrent mode failure): 1048518K->1048518K(1048576K), 0.2911506 secs] 1992262K->1600132K(1992320K), [Metaspace: 66181K->66181K(111
0016K)], 0.2914061 secs] [Times: user=0.29 sys=0.00, real=0.29 secs]
```

CMS会退化为Serial Old

G1

内存结构:

G1堆由多个区（region）组成，每个区大小1M~32M，逻辑上区有3种类型，包括（Eden、Survivor、Old），按分代划分包括：年轻代（Young Generation）和老年代（Old Generation）



- 同时工作在年老代和年轻代
- 适用于多核处理器, 大内存容量(>6G)的系统
- 分代收集
- 空间整合
- 可预测的停顿
- GC模式: young gc, mixed gc, full gc

ZGC

1. 暂停时间 < 10ms
2. TB级别的堆内存管理能力
3. 兼顾吞吐率和响应时间
4. 暂停时间不会随堆或实时设置大小而增加
5. 随JDK 11 正式发布
6. 支持Linux X64
7. 详细信息：http://openjdk.java.net/jeps/333

常用工具

1. jvisualvm
2. jstat
   - gcutil
   - gccause
   - class
3. jmap
   - heap
   - histo
   - dump
4. jinfo
5. http://gceasy.io

如果你有任何问题，欢迎与我联系；

https://www.yzhu.name/