



# Java HotSpot™ 虚拟机中的垃圾收集 选项及权衡

**Peter Kessler**  
**Jon Masamitsu**  
**John Coomes**  
Sun Microsystems  
<http://www.sun.com>

[java.sun.com/javaone/sf](http://java.sun.com/javaone/sf)



# 改善 Java™ 平台的性能

## 选择垃圾收集器

搞清 Java HotSpot™ 虚拟机中的各种垃圾收集方式，以及选用不同收集方式将带来的影响

有趣的内容

# 内容安排

欢迎来到“今日 GC”

介绍

GC 的新发展

并行收集器

并发收集器

工效特性

收集器的选择

未来

来自“我们的听众”的问题

# GC 现状

## 收集器的选择

- 串行收集器
  - 复制年轻代
  - 标记 / 清扫 / 缩并年老代
- 增量 (**train** 算法) 收集器
  - 年老代的增量收集
- 并行收集器
  - 并行复制年轻代
- 主体并发收集器
  - 并行复制年轻代
  - 并发收集年老代

# 介绍

## 今天的嘉宾演讲人

- **John Coomes**
  - 并发垃圾收集
- **Jon Masamitsu**
  - 工效特性，并行垃圾收集

# 垃圾收集有哪些新进展？

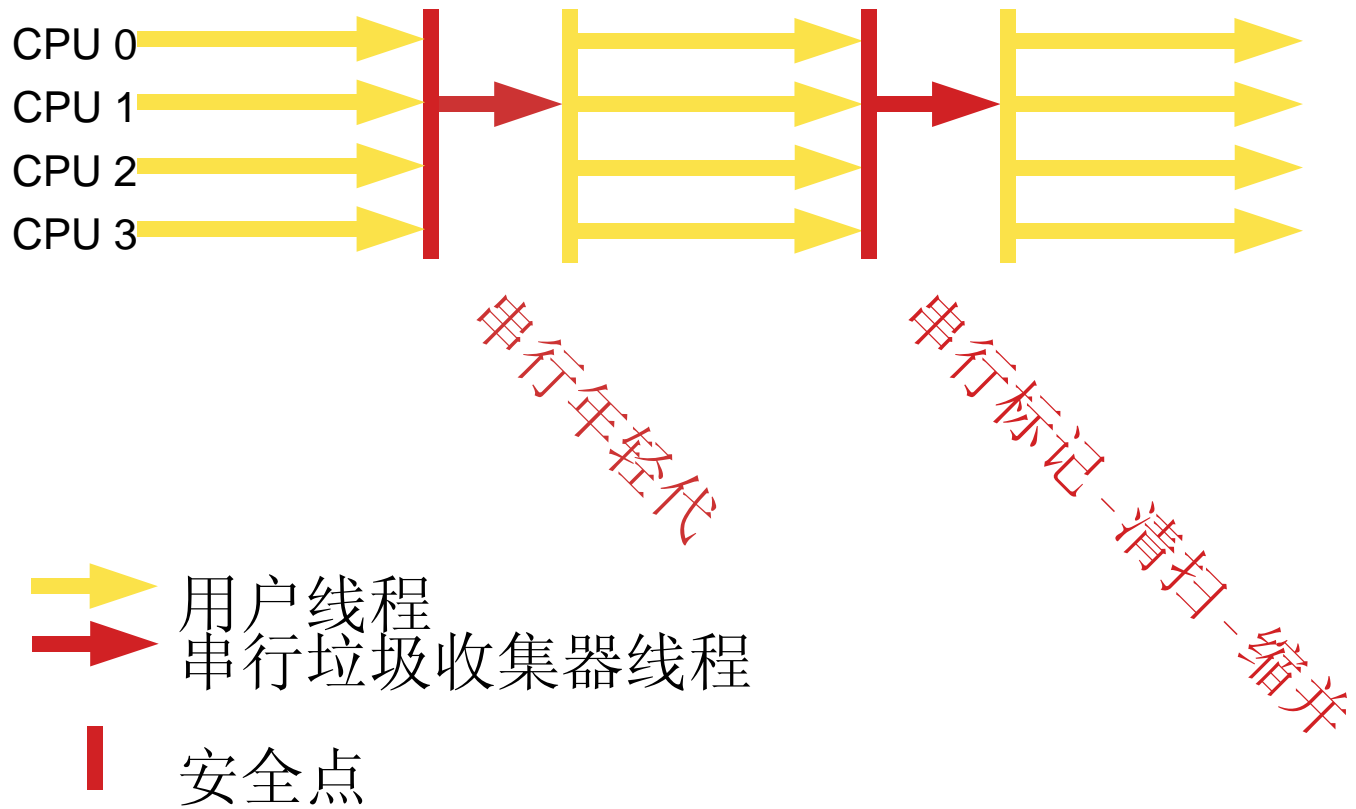
- 并行收集器
  - 基于行为的调整
  - 动态调整代的大小
- 主体并发收集器
  - 并行性更高
  - 动态调度
  - 提升失败处理
- 增量收集器不再受重视
  - `-Xincgc` 现在选择的是并发收集器
- 线程本地分配缓冲区
  - 根据使用情况调整大小

# 并行收集器的改进

## 指定所期望的行为

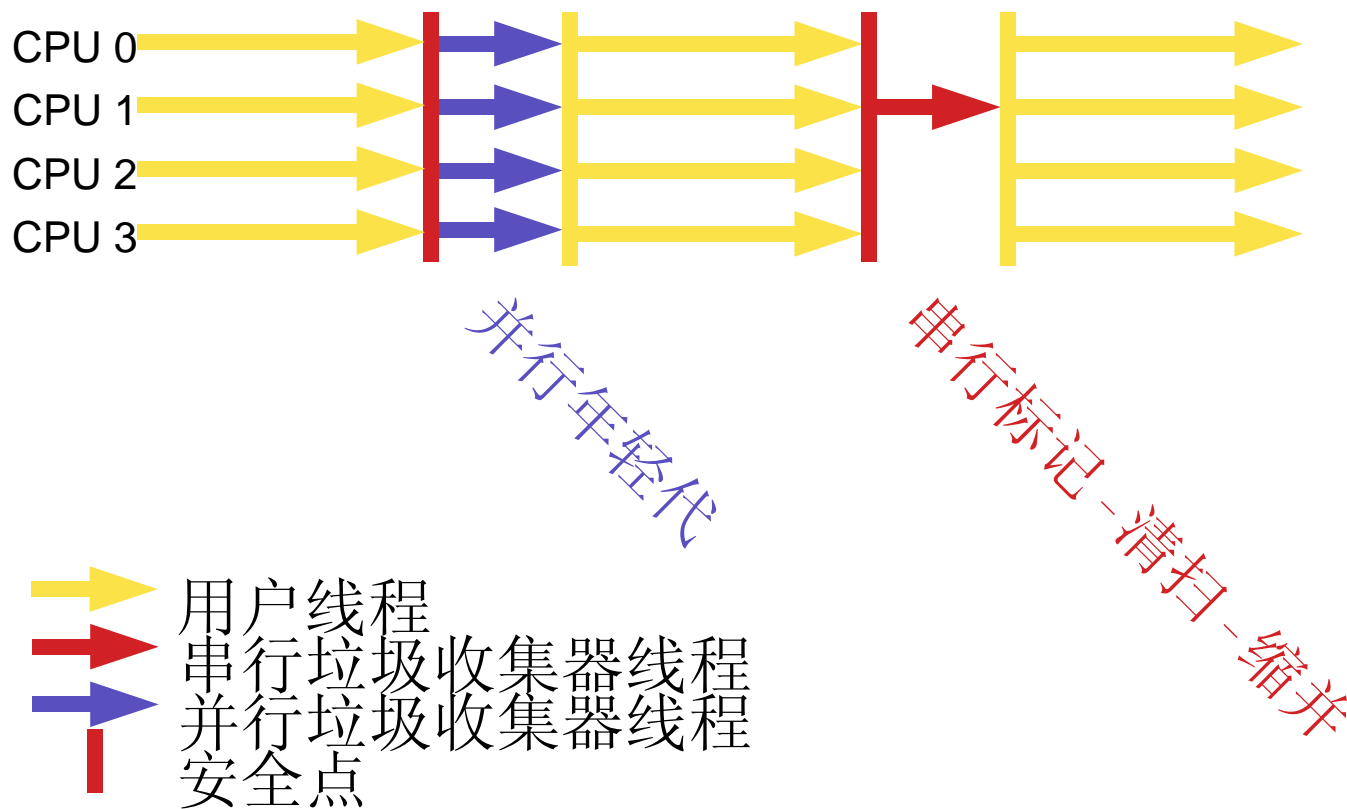
- 暂停时间目标
  - 所期望的最大暂停时间
- 吞吐能力目标
  - 用于垃圾收集的时间比例
- 使足迹最小化
  - 当其它目标都已实现时
- 垃圾收集的时间限制
  - 什么是“内存不足”？

# 串行收集器



# 并行收集器

## “吞吐能力问题”



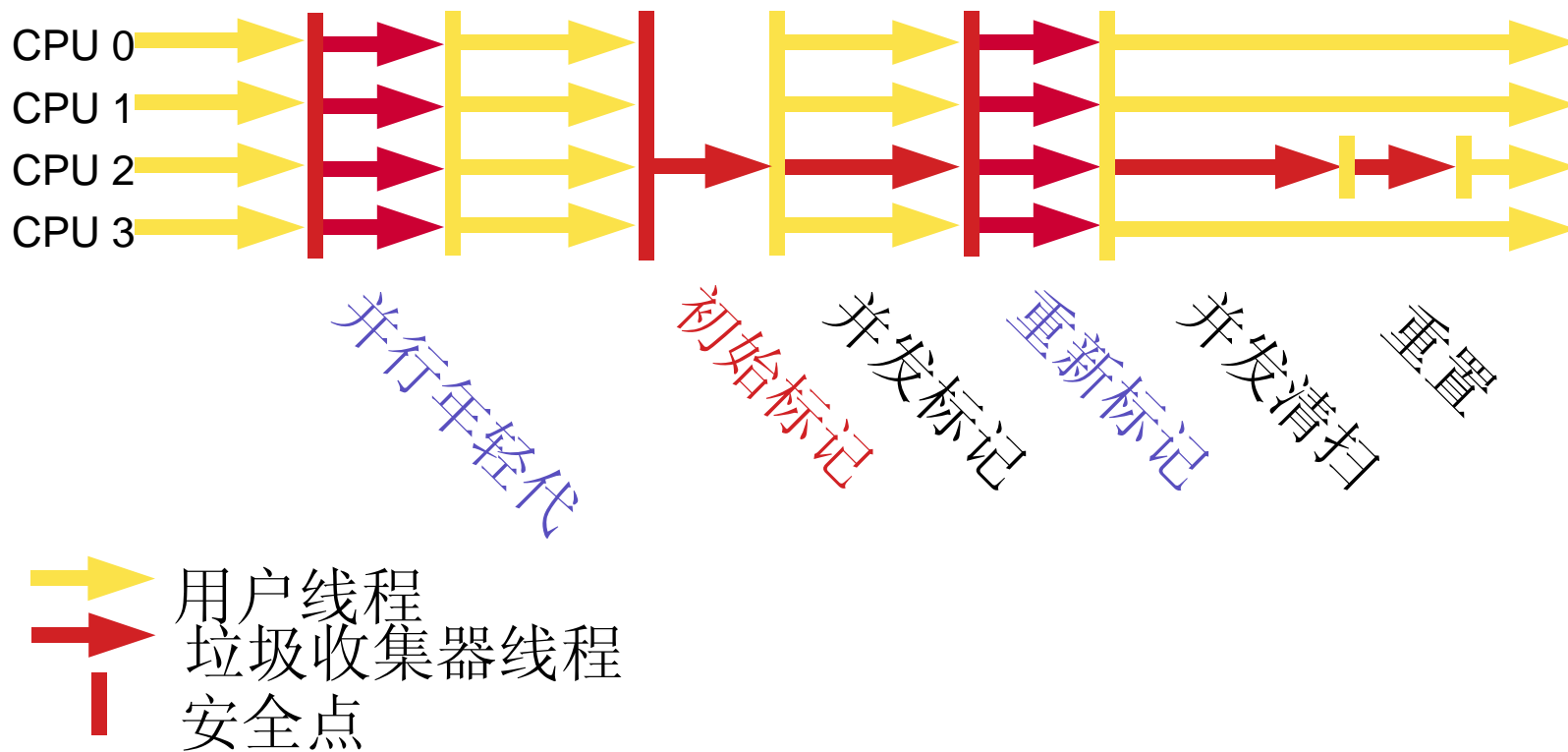
# 并行收集器的改进

## 指定所期望的行为

- 暂停时间目标
  - 所期望的最大暂停时间
- 吞吐能力目标
  - 用于垃圾收集的时间比例
- 使足迹最小
  - 当其它目标都已实现时
- 垃圾收集的时间限制
  - 什么是“内存不足”？

# 主体并发收集器

## “延迟问题”



# 主体并发收集器的改进

- 暂停期间并行性更高
  - 年轻代扫描
  - 引用对象处理（软引用，弱引用等）
- 动态调度
  - 年老代收集的开始
  - 年老代收集期间的暂停
- 提升失败处理
  - 更好地利用堆空间
  - 较少的全收集

# 内容安排

欢迎来到“今日 GC”

介绍

GC 的新特性

并行收集器

并发收集器

工效特性

选择收集器

未来

来自“我们的听众”的问题

# 并行收集器

## “吞吐量问题”

- 使应用程序有最长的可运行时间
  - 减少垃圾收集的时间
- 没有暂停时间要求
  - 或暂停时间要求较为宽松
- 举例：
  - 后台处理
  - 科学计算

# 要点： 强调吞吐量的收集器

## 并行收集器

- 并行年轻代收集
  - 高效
  - 可扩展性好
- 针对大型服务器的调整
  - 2 个到数十个 **CPU**
- 在配有 2 个或多个 **CPU** 的机器上快于串行收集器
- 串行年老代收集

# 要点：并行收集器 吞吐量目标

## 自动调整大小

- 吞吐量目标
- 吞吐量的测度
  - 垃圾收集所用的时间
  - 除垃圾收集之外所用的时间
- 为实现目标而增大代的大小
  - 代较大，意味着两次垃圾收集之间的间隔较长
  - 年轻代和年老代都增大
- 与不断变化的应用程序行为相适应

# 质疑：并行收集器吞吐量目标

## 自动调整大小

- 为什么把堆设置为最大？  
把 `-Xmx` 和 `-Xms` 设为相同的较大值
- 应用程序行为有较快变化时怎么办？
- 编译时间会怎样？

# 要点：并行收集器暂停时间目标

## 自动调节大小

- 垃圾收集最大暂停时间目标
  - 最大努力
  - 均值 + 方差
- 年轻代和年老代暂停时间的测度
  - 分别测度
- 为实现目标而对代进行缩并
  - 通常，代较小，收集速度就更快些
  - 缩并代并非总是可能的

# 质疑：并行收集器暂停时间目标

## 自动调节大小

- 这一目标是否总能得以实现？
  - 可能导致性能降低
- 若只有一种代的暂停时间太长，情况会怎样？
  - 没有实现目标
- 若两种代的暂停时间都太长，情况会怎样？
  - 每次缩并一种代

# 并行收集器小结

## “吞吐量问题”

- 可扩展到配备大量 CPU 的情况
- 自动调节大小
- 偶尔会出现长时间的暂停
  - 不是针对延迟限制而设计的

# 内容安排

欢迎来到“今日 GC”

介绍

GC 的新特性

并行收集器

并发收集器

工效特性

选择收集器

未来

来自“我们的听众”的问题

# 并发收集器

## “延迟问题”

- 延迟时间受暂停时间的影响
  - “典型”垃圾收集器在中止所有用户线程的情况下进行垃圾收集
  - 会给响应时间带来不利影响
- 暂停时间所影响的方面
  - 用户交互
  - 采用队列的系统
    - 超时导致额外处理或重试
- 举例：
  - Web 服务器 / 应用服务器
  - 电信交换
  - 集成开发环境

要点：缩短暂停时间

## 并发收集器

- 年轻代的收集
  - 通过并行而缩短暂停时间
- 年老代的收集
  - 大多数操作是并发完成的
    - 应用程序在继续运行
    - 垃圾收集占用 1 个 CPU
  - 2 次短暂的暂停
    - 通过并行而缩短暂停时间
- 暂停更为均衡

# 质疑：暂停时间短

## 并发收集器

- 是否需要专用占用某些 CPU?
  - 垃圾收集会占用某些处理器资源
  - 在单 CPU 平台上的性能
- 是否需要较大的堆？
  - 碎片
    - 不对年老代进行缩并
  - “浮动垃圾”

# 要点：暂停的调度

## 并发收集器

- 年轻代收集
  - 进行全收集时应用程序要停顿
- 年老代收集
  - 应用程序短暂停顿
    - “初始标记”和“重新标记”
- 年轻代和年老代收集的暂停相互独立
  - 不会同时出现
  - 否则就没有限制

# 要点：暂停的调度

## 并发收集器

- 问题：
  - 出现年轻代暂停
  - 之后马上出现年老代暂停
  - 应用程序则看到一个较长时间的暂停
- 解决办法：调度年老代暂停
  - 在年轻代收集之间

# 质疑：暂停的调度

## 并发收集器

- 是否能够缩短总的暂停时间？
- 分配速度有较快变化时会怎样？
- 对暂停进行调度的代价如何？

要点：并发收集的开始

## 并发收集器

- 动态启动并发收集
  - 获取统计数据
    - 收集的持续时间
    - 年老代被填满的速度
  - “及时”开始年老代的收集
- 根据应用程序的变化而进行调整
- 更充分地利用空间
  - 保留空间的需求降低

# 质疑：并发收集器

## 并发收集器

- 如果开始得太晚，会怎样呢？
  - 必须停止所有应用程序，进行垃圾收集
  - 更长的暂停时间
- 收集操作是否会开始得太早？
  - 会，但很少发生
  - 下一次收集时将会有更好的统计数据

# 并发收集器小结

## “延迟问题”

- 可以实现较短的延迟：
  - 500、200 甚至 100 毫秒的最大暂停时间
  - 不能保证
  - 依赖于应用程序、堆的大小、硬件等
- 某些自动调整
  - 年轻代的默认大小
  - 何时开始年老代收集
  - 年老代收集暂停的调度
- 尚不具备“工效特性”
  - 必须指定堆的大小而不是指定暂停时间

# 内容安排

欢迎来到“今日 GC”

介绍

GC 的新特性

并行收集器

并发收集器

工效特性

选择收集器

未来

来自“我们的听众”的问题

# 工效特性

## “让虚拟机选择”

- 选择垃圾收集器
  - 根据需要进行选择收集器
- 选择堆的大小
  - 一种大小不会适用于各种情况
- 选择运行时间编译器
  - 服务器端与客户端

# Java 2 平台标准版 (J2SE™) 1.4.x 及更早版本中的选择

各种应用程序都采用相同的选择

- 串行收集器
- 较小的堆
  - 初始堆 : ~4MB
  - 最大堆 : 64MB
- 客户端运行时间编译器

# J2SE 1.5.0 平台中的工效特性选择

## 更适用于某些应用程序

- 在服务器级的机器上
  - 并行收集器
  - 较大的堆
    - 初始堆：物理内存的 1/64，最大 1GB
    - 最大堆：物理内存的 1/4，最大 1GB
  - 服务器运行时间编译器
- 在客户端机器上
  - 同 J2SE 1.4.x 平台

# 服务器级的机器

## 如何决策

- 服务器级的机器配有：
  - 2 个或多个 CPU
  - 2GB 或更大内存
- 为什么根据机器类型做选择？
  - 简单，易解释
  - 经常表明应用类型
  - 目前必须在最开始时做出选择
- 理想情况下，根据所期望的行为进行选择
  - 例如：最大暂停时间，所期望的吞吐能力
  - 在运行时间进行调整
  - 未来版本

# 收集器的选择

- 是否是垃圾收集方面的问题？
  - 如果不是，就采用虚拟机的选择
- 堆太小？
  - 20MB – 30MB 或更小，采用虚拟机的选择
- 有暂停时间或响应时间的限制？
  - 3 秒或更短：开始时用并发收集器
  - 10 秒或更短：采用并行收集器

# 关于选用并发收集器的更多思考

- 分配速度与收集速度
  - 大多数收集操作是由一个线程完成的
  - 用户的活动线程超过 8 – 12 个
    - 并发收集器可能速度跟不上
- 只有 1 个或 2 个 CPU
  - 并发收集占用 1 个 CPU
- 举例：某应用程序：
  - 4 个 CPU，1GB 的堆，20MB 的年轻代
  - 100–200 毫秒的暂停时间
  - 并发执行
    - 每次年老代收集大约用 10–15 秒
    - 每隔几分钟做 1 次年老代收集

# 内容安排

欢迎来到“今日 GC”

介绍

**GC** 的新特性

并行收集器

并发收集器

工效特性

选择收集器

未来

来自“我们的听众”的问题

# 未来

- 增强工效特性
  - 在各代之间利用内存空间
  - 扩展到并发收集器
  - 根据所期望的行为选择收集器
- 并发年老代收集器
  - 提高可扩展性，改善性能
- 较老的垃圾收集器
  - **Train** 收集器（改用并发收集器）
  - 串行收集器（由虚拟机选择）

# 命令行选项

详细内容请参阅 J2SE 平台 1.5.0 版发布说明

- 选择年老代收集器
  - XX:+UseParallelGC
  - XX:+UseConcMarkSweepGC
  - XX:+UseSerialGC
- 请求获取性能数据
  - XX:GCTimeRatio=
  - XX:MaxGCPauseMillis=
- 特性的调整
  - Xms , -Xmx
  - XX:+UseAdaptiveSizePolicy

# 更多信息

- 与 Java HotSpot™ 虚拟机开发团队交流 (BOF-2520)
- Java™ 平台性能 (TS-1218)
- Java™ 2 平台标准版 (J2SE™)、Java™ 2 平台企业版 (J2EE™) Web 服务器、Web 服务和门户的性能 (BOF-1217)
- J2SE 1.4.2 平台垃圾收集调整指南
  - <http://java.sun.com/docs/hotspot/gc1.4.2/>
  - 正准备针对 J2SE 1.5.0 进行更新
- J2SE 1.5.0 平台发布说明

# 内容安排

欢迎来到“今日 GC”

介绍

**GC** 的新特性

并行收集器

并发收集器

工效特性

选择收集器

未来

来自“我们的听众”的问题

# 问与答

John Coomes  
Jon Masamitsu  
Peter Kessler



## Java HotSpot™ 虚拟机中的垃圾收集 选项及权衡

**Peter Kessler**  
**Jon Masamitsu**  
**John Coomes**  
Sun Microsystems  
<http://www.sun.com>

[java.sun.com/javaone/sf](http://java.sun.com/javaone/sf)

