# JVM内存管理

# 第一章、jvm内存区域

## 1.1 JVM内存区域图

运行时数据区

方法区

Method Area

堆

Heap

虚拟机栈

VM Stack

本地方法栈

Native method Stack

程序计数器

Program Counter Register

执行引擎

本地库接口

本地方法库

由所有线程共享

线程隔离的数据区

## 1.2 程序计数器

### 1.2.1概念：

是一块很小的内存区域，是当前线程所执行的字节码的行号指示器

### 1.2.2作用：

为字节码解释器工作时提供下一条需要执行的字节码指令

## 1.3 虚拟机栈

### 1.3.1概念：

描述的是java方法直向那个的内存模型:每个方法在执行时都会创建一个栈帧(stack frame)

### 1.3.2栈帧(stack frame)

栈帧包含：局部变量表、操作数栈、动态链接、方法出口

* 局部变量表：存放了编译期可知的各种基本数据类型、对象应用类型
* 操作数栈：存放的是程序运行过程中需要参与运算的数据
* 动态链接：就是将这些符号引用转换为调用方法的直接引用
* 每个栈帧内部都包含了一个指向运行时常量池中该栈帧所属方法的引用
* 在.java文件被编译到.class文件中，所有的变量和方法都作为符号引用保存在class文件的常量池里，比如：描述一个方法调用了其他方法时，就是通过常量池中指向方法的符号引用来表示的
* 方法出口：保存的是调用该方法的指令的下一条指令的地址

### 1.3.2 生命周期

与线程的生命周期相同

## 1.4本地方法栈

本质于虚拟机栈相同，只是服务对象不同而已，虚拟机栈是为虚拟机执行java方法服务的，而本地方法栈则是为虚拟机使用到的Native方法服务的

## 1.5 java堆

是java虚拟机所管理的内存中最大的一块。被所有的线程共享

### 1.5.1分类

新生代和老年代，新生代中有分为：Eden空间、From Survivor空间、To Survivor空间

### 1.5.2线程私有的分配缓存区(TLAB/Thread Local Allocation Buffer )

## 1.6 方法区

与java堆一样，是各个线程共享的内存区域，用于存储虚拟机加载的类信息、常量、静态变量、即时编译器编译后的代码等数据，非堆（Non-Heap），jdk1.8以前放在PerNew中，jdk1.8之后放在了metaspace中，jdk1.8之后方法区会进行垃圾回收，1.8以前不会进行回收

### 1.6.1运行时常量池

存放的是：编译器生成的各种字面量和符号引用，字符串常量，1.8之前放在PerNew中，1.8之后放在堆中

## 1.7 堆中对象的分配方式

* 若java堆内存规划是完整的，即：已使用过的在一边，未使用过的再另一边，且在两块内存的分界线中设置一个指针。则针对这种情况分配对象内存，只需将该指针向空闲区域移动对象大小个单位即可，这种分配方式称为：指针碰撞(Bump the Pointer)
* 若java堆内存规划是不完整的，即：已使用过的和未使用过的混杂在一起，这时java虚拟机就需要维护一张表：记录那些内存区域使用过，那些未使用过。则针对这种情况分配对象内存，只需从未使用过的内存区域中划分出一个对象大小的内存区域，并更新这一区域的状态为已使用，这种分配方式称为：空闲列表(Free List)

## 1.8 堆内存分配同步机制

* 一种是采用cas的方式
* 一种是通过TLAB的方式来避免对象分配时需要进行同步

## 1.9 对象结构

对象在内存中的储存布局分为3块区域：对象头（Header）、实例数据（Instance Data）和对齐填充（Padding）、数组长度

### 1.9.1对象头：-XX:UseCompressedClassPointers

对象头数据有分为两部分：Mark Word和类型指针

#### 1.9.1.1 Mark Word：



|  |  |  |
| --- | --- | --- |
| 储存内容 | 标志位 | 状态 |
| 对象哈希码、对象分代年龄 | 01 | 未锁定 |
| 指向锁记录的指针 | 00 | 轻量级锁定 |
| 指向重量级锁指针 | 10 | 膨胀（重量级锁定） |
| 空，不需要记录信息 | 11 | GC标记 |
| 偏向线程ID、偏向时间戳、对象分代年龄 | 01 | 可偏向 |

#### 1.9.1.2 类型指针

对象指向他的类元数据的指针，虚拟机通过这个指针来确定这个对象是那个类的实例

### 1.9.2对象的访问方式

句柄和直接指针访问

* 若使用句柄来访问的话，那么java堆中将会划分出一款内存来作为句柄池，reference中存储的就是对象的句柄地址
* 若使用直接指针访问，那么java堆对象的布局就必须考虑如何放置访问类型数据的相关信息，而reference中储存的直接就是对象地址

# java内存模型

## 硬件内存一致性：

### MESI cache内存一致性协议：缓存锁

* modified（缓存修改）、Exclusive（缓存cpu独占）、Shared（缓存共享）、Invalid（缓存失效）
* 现代缓存数据一致性：缓存锁+总线锁
* 缓存行cache line：一次性加载多个缓存数据组成：cache line，比如64字节
* 伪缓存：位于同一个缓存行的两个不同数据，被两个不同CPU锁定，产生互相影响的伪共享问题

## JMM：用来屏蔽各种硬件和操作系统的内存访问差异，以实现Java程序在各种平台都能达到一致的内存访问效果

### 硬件方面的支持：

* Sfence: 在sfence指令钱的写操作当必须sfence指定后的写操作完成
* Lfence: 在lfence指令的读操作当必须在lfence指令后的读操作前完成
* Mfence: 在mfence指令钱的读写操作当必须在mfence指令后的读写操作前完成
* 原子指令：lock、总线锁

### JVM提供的：

* LoadLoad：对于Load1；LoadLoad；Load2在Load2及后的读取操作要读取的数据被访问前，保证Load1要读取的数据被读取完毕
* StoreStore：对于Store1；storeStore；Store2在store2及后续的读取执行前，保证store1的写入操作对其它处理可见
* LoadStore：对于Load1；LoadStore；Store2在store2及后续写入操作被刷出前，保证load1要读取的数据读取完毕
* StoreLoad：对于这样Store1；storeLoad；Load2在load2及后续所有读取操作前，保证store1的写入对所有处理可见

## Volatile：禁止指令重排序、内存一致性

* 字节码层面：添加了ACC\_VOLATILE
* JVM层面：在读写内存时都添加了读写屏障

## Happens-Before原则（先行发生规则）

* ****程序次序规则****：在一个线程内，书写在前面的操作先行发生于书写在后面的操作
* ****管程锁定规则****：一个unlock操作先行发生于后面对同一个锁的lock操作
* ****volatile变量规则****：对一个volatile变量的写操作先行发生于后面对这个变量的读操作
* ****传递性****：如果操作A先行发生于操作B，操作B先行发生于操作C，那就可以得出操作A先行发生于操作C的结论

# 第三章、垃圾收集器和内存分配策略

## 1.1判断对象是否存活的算法

### 1.1.1引用计数法：

给对象添加一个引用计数器，每当有一个地方引用它时，计数器值就+1；当引用失效时，计数器值就-1；任何时刻计数器值为0的对象就是不可在被使用的

### 1.1.2 可达性分析算法

就是通过一系列的称为：“GC Roots”的对性爱那个作为起始点，从这些节点开始向下搜索，搜索锁走过的路径称为引用链（Refrence Chain），当一个对象到GC Roots没有任何引用链相连（用图论的话来说，就是从GC Roots到这儿对象）时，则证明此对象是不可用的

#### 1.1.1.1 java语言中GC Roots对象

* 虚拟机栈（栈帧中的本地变量表）中应用的对象
* 方法区中类静态属性引用的对象
* 方法区中常量引用的对象
* 本地方法栈中JNI（一般说的Natvie）引用的对象

## 1.2 引用类型

* 强引用（Strong Reference）：类似：Object obj = new Object();其中obj引用就是强引用，只要Strong Refrence还存在，GC就不会回收
* 软引用（Soft Reference）：用来描述一些还有用但并非必需的对象，在系统将要发生内存溢出之前，将会把这些对象列进回收返回之中进行第二次回收
* 弱引用（Weak Reference）：同样是用来描述一些非必需对象的，被弱引用关联的对象只能生存到下一次垃圾收集器发生之前
* 虚引用（Phantom Reference），为一个对象设置虚引用关联的唯一目的就是能在这个对象被收集器回收时收到一个系统通知

## 1.3回收过程

通过可达性算法分析出该对象是不可达的，GC会将该对象进行一次标记和进行一次筛选，筛选的条件：此对象是否有必要执行finalize()方法（当对象没有覆盖finalize()方法，或finalize方法已经被虚拟机调用过，虚拟机将这两种情况都视为没有必要执行）。当该对象被判定为有必要执行finalize时，会将该对象放置到F-Queue的队列中，梢后有gc来触发该对象的finalize的执行，若该对象在执行一次finalize()时重新是可达的，则第二次标记是它将被溢出“即将回收”的集合；若该对象执行玩一次finalize()方法时仍不可达，则是gc将会将其回收

## 1.4 回收方法区

### 1.4.1 回收方法区的主要区域

废弃常量和无用的类，回收废弃的常量与回收java堆中的对象非常类似

#### 1.4.1.1 无用的类

* 该类所有的实例都已经被回收
* 加载该类的ClassLoader已经被回收
* 该类对应的java.lang.Class对象没有在任何地方被引用，无法在任何地方通过反射访问该类的方法

## 1.5 垃圾收集算法

### 1.5.1标记-清除算法

首先标记出所有需要回收的对象，在标记完成后统一回收所有被标记的对象

#### 1.5.1.1标记-清除算法的不足

* 效率不高
* 空间利用率不高，容易造成内存碎片

### 1.5.2 复制算法

为了解决标记-清除算法中的效率问题，另一种复制算法孕育而生，它将内存区域划分为容量大小相等的两块，每次只使用其中一块，当这一块用完后，就将还存活的对象复制到另一块上面，然后再把已使用过的内存空间一次清理掉。这种做法内从利用率太低，只有50%。

java研究人员发现java对象符合“朝生夕死”的特点，而将内存划分为三块局域：一块较大的Eden空间和两块相同大小的Survior空间，每次只使用Eden空间和一块Survior空间，当回收对象时，将Eden和Surivor空间中还存活的对象复制到另一块Surivor中，由于无法保证每次存活对象另一块Surivor都能存放完毕，故需要另一块备用内存老年代做担保，当无法存放完毕时放置到老年代内存中，这种做法内从利用率达到：90%

### 1.5.3 标记-整理算法

针对老年代内存局域而设计的一套内存回收算法。算法的基本过程：将老年代中对象进行标记，然后将所有已存活的对象全都向一端移动，然后直接清理掉端边界以外的内存。

### 1.5.4 分代算法

根据内存局域中年龄的不同，采用不同的回收算法

## 1.6 HotSpot的算法实现

### 1.6.1 GC停顿点

可达性算法必须在一个一致性（是指在整个分析期间整个执行系统看起来就像冻结在某个时间点上，不可以出现分析过程中对性爱那个引用关系还在不断变化的情况）的快照中进行

### 1.6.2 OopMap数据结构

jvm通过OopMap数据结构来存储对象的应用关系

### 1.6.3 GC安全点

jvm没有在每条指令执行时都生成OopMap结构，而是在一个被称为GC安全点（Safepoint）的地方进行生成

#### 1.6.3.1 抢先式中断

在GC发生时，首先把所有线程全部中断，若发现有线程中断的地方不是安全点，就恢复线程，让它运行到“安全点”

#### 1.6.3.2 主动式中断

当GC需要中断时，不直接对线程操作，仅仅设置一个标志，各个线程执行时主动去轮询这个标志，发现中断标志为真时就自己中断挂起。轮询标志地点就是安全点

### 1.6.4 GC安全区域

是指在一段代码片段中，引用关系不会发生变化。则在这个局域内的任意地方开始GC都是安全的，称该局域为：安全区域

## 1.7垃圾收集器

### 1.7.1垃圾收集器种类图

Serial

ParNew

Parallel

Scavenge

G1

Young generation

CMS

Serial Old

(MSC)

Parallel Old

Tenured generation

### 1.7.2 Serial收集器

#### 1.7.2.1工作示意图

CPU 0

CPU 1

CPU 2

CPU 3

GC线程

新生代采用复制算法暂停所有用户线程

SafePoint

GC线程

老年代采用标记-整理算法暂停所有用户线程

用户线程1

用户线程2

用户线程3

用户线程4

SafePoint

Serial收集器依然是虚拟机运行在Client模式下的默认新生代收集器，特点：简单而高效，对于限定单个CPU环境来说，Serial收集器由于没有线程交互的开销，专心做垃圾收集自然可由获得最高的单线程收集效率

### 1.7.3 ParNew收集器

#### 1.7.3.1 工作示意图

CPU 0

CPU 1

CPU 2

CPU 3

GC线程1

新生代采用复制算法暂停所有用户线程

SafePoint

GC线程

老年代采用标记-整理算法暂停所有用户线程

用户线程1

用户线程2

用户线程3

用户线程4

SafePoint

GC线程2

GC线程3

ParNew收集器是Serial的多线程版，可供设置的参数：-XX:SurvivorRatio、-XX:PretenureSizeThreShold、-XX:HandlePromotionFailure。仍是许多运行在Server模式下的虚拟机首选的新生代收集器，同样可以通过使用：-XX:+UseConcMarkSweepGC/-XX:+UseParNewGC来命令来指定收集器为ParNew收集器

### 1.7.4 Parallel Scavenge收集器

Parallel Scavenge同样也是一个新生代垃圾收集器，采用复制算法，且同样也是多线程收集。

#### 1.7.4.1 吞吐量

Parallel Scavenge收集器关注的是一个可控制的吞吐量（cpu用于运行用户代码的时间与cpu总消耗时间的比值），即：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）

#### 1.7.4.2 影响吞吐量的参数

* **-XX:MaxGCPauseMillis**：控制最大垃圾收集停顿时间,单位：ms，取值范围：> 0
* **-XX:GCTimeRatio**：直接设置吞吐量大小,取值范围：>0且<100的整数
* **-XX:UseAdaptiveSizePolicy**：开关参数设置以后，用户就无需手动设置新生代大小（-Xmn）、Eden与Survivor区的比例（-XX:SurvivorRatio）、晋升老年代对象的大小（-XX:pretenureSizeThreshold）,jvm会自动调整这些参数，这种调节方式称为：GC自适应的调节策略

### 1.7.5 Serial Old收集器

#### 1.7.5.1收集器器示意图

CPU 0

CPU 1

CPU 2

CPU 3

GC线程

新生代采用复制算法暂停所有用户线程

SafePoint

GC线程

老年代采用标记-整理算法暂停所有用户线程

用户线程1

用户线程2

用户线程3

用户线程4

SafePoint

采用标记-整理算法收集老年代对象内存。主要意义：在Client模式下虚机使用，用作CMS收集器后备预案，在并发收集发生Concurrent Mode Failure时使用

### 1.7.6 Parallel Old收集器

#### 1.7.6.1 Parallel Old收集器示意图

CPU 0

CPU 1

CPU 2

CPU 3

GC线程1

新生代采用复制算法暂停所有用户线程

SafePoint

GC线程1

老年代采用标记-整理算法暂停所有用户线程

用户线程1

用户线程2

用户线程3

用户线程4

SafePoint

GC线程2

GC线程3

GC线程2

GC线程3

### 1.7.7 CMS收集器

#### 1.7.7.1 CMS的简介

CMS的全称：Concurrent Mark Sweep，是一种以获取最短回收停顿时间为目标的收集器，采用的是“标记-清除”算法。

#### 1.7.7.2 CMS回收内存的步骤

* 初始标记（CMS initial mark），需要中断所有用户进程的执行，仅仅标记一下GC Roots能直接关联到的对象，时间很快
* 并发标记（CMS concurrent mark），进行GC Tracing的过程
* 重新标记（CMS remark），需要中断所有用户进程的执行，修正并发标记期间因用户程序继续运作而导致标记变动的那一部分对象的标记记录，时间比初始标记长，远小于并发标记时间
* 并发清除（CMS concurrent sweep），并发地清除那些不可达对象的内存

由于整个回收过程中并发标记和并发清除的时间远远大于初始话标记时间和重新标记时间，故可近视地认为：CMS收集器的内存回收过程是与用户线程一起并发执行的

#### 1.7.7.3 CMS收集器示意图

CPU 0

CPU 1

CPU 2

CPU 3

SafePoint

用户线程2

用户线程4

用户线程3

用户线程1

初始标记

用户线程1

用户线程2

并发标记

用户线程4

重新标记

重新标记

重新标记

重新标记

用户线程1

用户线程2

并发清理

用户线程4

用户线程1

用户线程2

重置线程

用户线程4

SafePoint

SafePoint

SafePoint

SafePoint

#### 1.7.7.4 CMS浮动垃圾（Floating Garbage）

* 由于CMS并发清理阶段用户线程还在运行，伴随程序运行自然就会有新的垃圾产生，这一部分垃圾的出现出现在标记之后，只能等待下一次GC的清理，这一部分的垃圾称为：浮动垃圾
* 通过设置：**-XX:CMSInitiatingOccupancyFraction**当老年代垃圾达到多少时，才进行一次GC，当这个参数设置过大时，容易导致大量的Full GC（使用预备Serial old来进行回收老年代内存）

#### 1.7.7.5 主要参数

* **-XX:UseCMSCompactAtFullCollection**，用于在CMS收集器即将要进行一次Full GC时开启内存碎片的合并整理过程，默认是开启的
* **-XX:CMSFullGCsBeforeCompaction**：用于设置多少次不压缩的Full GC后，来一次带压缩的，默认值为0，表示每次Full GC都进行压缩内存

### 1.7.8 G1收集器

#### 1.7.8.1 G1的简介

* G1收集器是将java堆内存划分为大小相同的若干个region区域，G1负责跟踪各个Region区域中的垃圾堆积的价值大小，在后台维护一个有限列表，每次根据允许的收集时间，优先回收价值最大的Region。
* G1采用Remembered Set：记录了其它Region中的对象到本Region的引用，来避免进行全堆扫描。G1中每个Region都有一个与之对应的Remembered Set，虚拟机发现程序在对Reference类型的数据进行写操作时，会产生一个Writer Barrier暂时中断写操作，检查Reference引用的对象是否处于不同的Region之间，若是，便通过CardTable把相关引用信息记录到被引用对象所属的Region的Remembered set之中
* CardTable：YGC时有可能老年代的对象需要扫描整个OLD区，效率非常低，所以JVM设计了CardTable，如果一个OLD区CardTable中有对象指向Y区，就将它设为Dirty区，下次扫描时，只需要扫描Dirty Card在结构上，CardTable用BitMap实现
* CSet=Collection Set：一组可被回收的分区的集合在CSet中存活的数据会在GC过程中被移动到另一个可用分区，CSet中的分区可以来自Eden、survivor空间、老年代。CSet会占用不到整个堆空间的1%大小

#### 1.7.8.2 G1收集器的特点

* 并行与并发
* 分代收集
* 空间整合
* 可预测的停顿

#### 1.7.8.3G1 GC种类

* YGC：采用复制算法，触发条件：Eden空间不足
* Mixed：-XX:InitiatingHeapOccupacyPercent=45,当o超过这个值时，启动MixedGC
* FGC：采用Serial收集器收集
* 如何避免FGC：提升cpu、内存性能，缩小InitiatingHeapOccupacyPercent参数值

#### 1.7.8.3 G1Mixed收集器示意图

CPU 0

CPU 1

CPU 2

CPU 3

SafePoint

用户线程2

用户线程4

用户线程3

用户线程1

初始标记

用户线程1

用户线程2

并发标记

用户线程4

最终标记

最终标记

最终标记

最终标记

筛选回收

筛选回收

筛选回收

筛选回收

用户线程1

用户线程2

用户线程3

用户线程4

SafePoint

SafePoint

SafePoint

SafePoint

## 1.8三色算法：

### 1.8.1核心概念：

* 黑色：自身、成员变量均已标记完成
* 灰色：自身被标记、成员标量未被标记
* 白色：未被标记的对象

### 1.8.2漏标：

在remark标记过程中，灰色对白色对象的应用被移除，但在并发标记过程中，黑色对象对白色又新增了引用，若此时没有对黑色对象重新进行标记，则该白色对象会被回收

### 1.8.3漏标解决方案：

* + incremental update：增量更新，关注引用的增加，把黑色重新标记为灰色，下次重新扫描属性(CMS使用到的算法)
  + SATB snapshot at the beginning：关注引用的删除，当灰色引用的白色消失时，要把这个引用推到GC的堆栈，保证白色还能被GC扫描到(G1使用)

# 第四章、Class文件

## 1.1 Class文件的格式

class文件采用的是一种类似c语言结构的伪结构来存储数据的，这个伪结构只有两种数据类型：表和无符号数

### 1.1.1 无符号数

无符号数属于基本的数据类型，以u1、u2、u4、u8分别表示1个字节、2个字节、4个字节、8个字节，无符号数：用于描述数字、引用、数量值、按照utf-8编码构成的字符串值

### 1.1.2 表

表是由多个无符号数或其他表作为数据项构成的符合数据类型，所有表都习惯性以“\_info”结尾

## 1.2 Class文件列表

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 数量 |
| u4 | magic | 1 |
| u2 | minor\_version | 1 |
| u2 | major\_version | 1 |
| u2 | constant\_pool\_count | 1 |
| cp\_info | constant\_pool | constant\_pool\_count-1 |
| u2 | access\_flags | 1 |
| u2 | this\_class | 1 |
| u2 | super\_class | 1 |
| u2 | interfaces\_count | 1 |
| u2 | interfaces | interfaces\_count-1 |
| u2 | fields\_count | 1 |
| field\_info | fields | fields\_count |
| u2 | methods\_count | 1 |
| method\_info | methods | methods\_count |
| U2 | attributes\_count | 1 |
| attribute\_info | attributes | attributes\_count |

无论是无符号数还是表，当需要描述同一类型但数量不定的多个数据时，经常会使用一个前置的容量计数器加若干个连续的数据项的形式，这时称这一系列连续的某一类型的数据为某一类型的集合

### 1.2.1 magic与minor\_version、major\_version

* **magic**: 确定这个文件是否为一个能被虚拟机接收的class文件，4个字节值：0XCAFEBABE
* **minor\_version**: class文件的次版本号，2个字节
* **major\_version**: class文件的主版本号，2个字节

### 1.2.2 Constant pool

* 由于常量池的数量不固定的，所以在常量池的入口需要放置一项u2类型的数据，代表常量容器计数值class文件结构中只有常量池的容量计数是从1开始，对于其他集合类型，包括接口索引集合、字段表集合、方法表集合等的容量计数都与一般习惯相同，从0开始
* 常量池中主要存放的两大类常量：字面量、符号引用
* 类和接口的全限定名
* 字段和名称和描述符
* 方法的名称和描述符
* 常量池中每一项常量都是一个表，共有14张表，每张表第一位都是一个u1类型的标志位

#### 1.2.2.1常量池项目类型

大小为u1，标识表的类型

|  |  |  |
| --- | --- | --- |
| 类型 | 标志 | 描述 |
| CONSTANT\_Utf8\_info | 1 | Utf-8编码的字符串 |
| CONSTANT\_Integer\_info | 3 | 整型字面量 |
| CONSTANT\_Float\_info | 4 | 浮点型字面量 |
| CONSTANT\_Long\_info | 5 | 长整型字面量 |
| CONSTANT\_Double\_info | 6 | 双精度浮点型字面量 |
| CONSTANT\_Class\_info | 7 | 类或接口的符号引用 |
| CONSTANT\_String\_info | 8 | 字符串类型字面量 |
| CONSTANT\_Fieldref\_info | 9 | 字段的符号引用 |
| CONSTANT\_Methodref\_info | 10 | 类中方法的符号引用 |
| CONSTANT\_InterfaceMethodref\_info | 11 | 接口中的方法的符号引用 |
| CONSTANT\_NameAndType\_info | 12 | 字段或方法的部分符号引用 |
| CONSTANT\_MethodHandle\_info | 15 | 表示方法的句柄 |
| CONSTANT\_MethodType\_info | 16 | 表示方法的类型 |
| CONSTANT\_InvokeDynamic\_info | 18 | 表示一个动态方法调用点 |

#### 1.2.2.2 常量池中14中常量项的结构总表

|  |  |  |  |
| --- | --- | --- | --- |
| 常量 | 项目 | 类型 | 描述 |
| CONSTANT\_Utf8\_info | tag | u1 | 值为1 |
| length | u2 | UTF-8编码的字符串占用的字节数 |
| bytes | u1 | 长度为length的utf-8编码的字符串 |
| CONSTANT\_ Integer \_info | tag | u1 | 值为3 |
| bytes | u4 | 按高位在前存储的int值 |
| CONSTANT\_Float\_info | tag | u1 | 值为4 |
| bytes | u4 | 按高位在前存储的float值 |
| CONSTANT\_ Long \_info | tag | u1 | 值为5 |
| bytes | u8 | 按高位在前存储的long值 |
| CONSTANT\_Double\_info | tag | u1 | 值为6 |
| bytes | u8 | 按高位在前存储的double值 |
| CONSTANT\_ Class\_info | tag | u1 | 值为7 |
| bytes | u2 | 指向全限定名常量项索引 |
| CONSTANT\_ String\_info | tag | u1 | 值为8 |
| bytes | u2 | 指向字符串字面量索引 |
| CONSTANT\_ Fieldref\_info | tag | u1 | 值为9 |
| index | u2 | 指向声明字段的类或接口描述符CONSTANT\_ Class\_info的索引项 |
| index | u2 | 指向名称以及类型描述符CONSTANT\_NameAndType\_info的索引项 |
| CONSTANT\_ Methodref\_info | tag | u1 | 值为10 |
| index | u2 | 指向声明方法的类或接口描述符CONSTANT\_ Class\_info的索引项 |
| index | u2 | 指向名称以及类型描述符CONSTANT\_NameAndType\_info的索引项 |
| CONSTANT\_ InterfaceMethodref\_info | tag | u1 | 值为11 |
| index | u2 | 指向声明方法的类或接口描述符CONSTANT\_ Class\_info的索引项 |
| index | u2 | 指向名称以及类型描述符CONSTANT\_NameAndType\_info的索引项 |
| CONSTANT\_NameAndType\_info | tag | u1 | 值为12 |
| index | u2 | 指向该字段或方法名称常量项的索引 |
| index | u2 | 指向该字段或方法名称常量项的索引 |
| CONSTANT\_MethodHandle\_info | tag | u1 | 值为15 |
| reference\_kind | u1 | 值必须[1-9]，决定了方法句柄的类型，方法句柄类型的值表示方法句柄的字节码行为 |
| reference\_index | u2 | 值必须是对常量的有效索引 |
| CONSTANT\_MethodType\_info | tag | u1 | 值为16 |
| descriptor\_index | u2 | 值必须是对常量的有效索引，常量池在该索引出的项必须是CONSTANT\_Utf8\_info结构，表示方法的描述符 |
| CONSTANT\_invokeDynamic\_info | tag | u1 | 值为15 |
| bootstrap\_method\_attr\_index | u2 | 值必须是对当前class文件中引导方法表的bootstrap\_method[]数组的有效索引 |
| Name\_and\_type\_index | u2 | 值必须是对常量的有效索引，常量池在该索引出的项必须是CONSTANT\_Utf8\_info结构，表示方法名和方法描述符 |

### 1.2.3访问标志

常量池结束后，紧接着的两个字节代表：访问标志，若访问标志位的值不在这些值之中，这是下列值一些值通过：“|”得到，比如：0x0021=0x0001|0x0020得到

|  |  |  |
| --- | --- | --- |
| 标志名称 | 标志值 | 含义 |
| ACC\_PUBLIC | 0x0001 | 是否为public类型 |
| ACC\_PRIVATE | 0x0002 | 是否为private类型 |
| ACC\_PROTECTED | 0x0004 | 是否为protected类型 |
| ACC\_ STATIC | 0x0008 | 是否被static关键字修饰 |
| ACC\_ FINAL | 0x0010 | 是否被声明为final |
| ACC\_SYNCHRONIZED | 0x0020 | 是否被synchronized关键字修饰 |
| ACC\_VOLATILE | 0x0040 | 是否被volatile关键字修饰 |
| ACC\_TRANSIENT | 0x0080 | 是否被transient关键字修饰 |
| ACC\_NATIVE | 0x0100 | 是否被native关键字修饰 |
| ACC\_ INTERFACE | 0x0200 | 是否为接口 |
| ACC\_ ABSTRACT | 0x0400 | 是否为abstract类型 |
| ACC\_STRICT | 0x0800 | Strict |
| ACC\_ BRIDGE | 0x0040 | Bridge |
| ACC\_ VARARGS | 0x0080 | Varargs |
| ACC\_ SYNTHETIC | 0x1000 | 标识这个类并非由用户代码产生的 |
| ACC\_ ANNOTATION | 0x2000 | 标识这是一个注解 |
| ACC\_ ENUM | 0x4000 | 标识这是一个枚举 |
| ACC\_ MANDATED | 0x8000 | 标识这是一个mandated |

### 1.2.4类索引和父类索引与接口索引集合

类索引(this class)和父类索引都为一个u2类型的数据，而接口索引节后是一组u2类型的数据的集合，class文件中有这三项来确定这个类的继承关系

### 1.2.5 字段表集合

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | access\_flags | 1 | 字段访问标识 |
| u2 | name\_index | 1 | 字段的简单名称 |
| u2 | descriptor\_index | 1 | 字段的描述符 |
| u2 | attributes\_count | 1 | 属性列表长度 |
| attribute\_info | attributes | attributes\_count | 属性 |

* access\_flags：字段访问标识，参照1.2.3表中列出的访问标识
* name\_index：代表字段的简单名称，即是指没有类型和参数修饰的字段的名称
* descriptor\_index：字段的描述符，描述字段的数据类型，具体参照如下1.2.6表列出的描述符标识字符
* attributes\_count：属性列表的长度
* attributes：属性列表
* 注意：字段表集合中不会列出从超类或父接口中继承而来的字段，但又可能会列出原本java代码之中不存在的字段，例如：在内部类中为了保持对外部类的访问性，会自动添加指向外部类实例的字段

### 1.2.6 描述符标识字符

描述符的作用：描述字段的数据类型；描述方法的参数列表（包括数量、类型以及顺序）和返回值，其中基本数据类型（byte、char、double、float、int、long、short、boolean）以及代表返回值的void类型都用一个大写字符表示，而对象类型则用字符L加对象的全限定名来表示

|  |  |
| --- | --- |
| 标识字符 | 含义 |
| B | 基本数据类型byte |
| C | 基本类型char |
| D | 基本类型double |
| F | 基本类型F |
| I | 基本类型int |
| J | 基本类型long |
| S | 基本类型short |
| Z | 基本类型boolean |
| V | 特殊类型void |
| L | 对象类型，例如：Ljava/lang/Object |

### 1.2.7方法表集合

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | access\_flags | 1 | 方法访问标识 |
| u2 | name\_index | 1 | 方法的简单名称 |
| u2 | descriptor\_index | 1 | 方法的描述符 |
| u2 | attributes\_count | 1 | 属性列表长度 |
| attribute\_info | attributes | attributes\_count | 属性 |

* access\_flags：方法访问标识，参照1.2.3表中列出的访问标识
* name\_index：代表方法的简单名称，即是指参数列表和返回值的方法名称
* descriptor\_index：方法的描述符，描述方法的参数列表和返回值，具体参照如下1.2.6表列出的描述符标识字符
* attributes\_count：属性列表的长度
* attributes：属性列表

注意：若父类的方法没有在子类中被重写，方法表集合中就不会出现来自父类的方法新，但又可能会列出有编译启自动添加的方法，典型：便是

### 1.2.8 属性列表

|  |  |  |
| --- | --- | --- |
| 属性名称 | 使用位置 | 含义 |
| Code | 方法表 | Java代码编译成的字节码指令 |
| ConstantValue | 字段表 | Final关键字定义的常量值 |
| Deprecated | 类、方法表、字段表 | 被声明为deprecated的方法和字段 |
| Exceptions | 方法表 | 方法抛出的异常 |
| EnclosingMethod | 类文件 | 仅当一个类为局部类或匿名类时才能拥有这个属性，这个属性用于标识这个类所在的外围方法 |
| InnerClasses | 类文件 | 内部类列表 |
| LineNumberTable | Code属性 | Java源码的行号与字节码指令的对应关系 |
| LocalVariableTable | Code属性 | 方法的局部变量描述 |
| StackMapTable | Code属性 | JDK1.6中新增的属性，供新的类型检查验证器（Type Checker）检查和处理目标方法的局部变量和操作数栈所需要的类型是否匹配 |
| Signature | 类、方法、字段表 | JDK1.5中新增的属性，这个属性用于支持泛型情况下的方法签名，在java语言中，任何类、接口、初始化方法或成员的泛型签名如果包含了类型变量（Type Variables）或参数化类型（Parameterized Type），则Signature属性会为他记录泛型签名信息。由于java的泛型采用擦除法实现，在为了避免类型信息被擦除后导致签名混乱，需要这个属性记录泛型中的相关信息 |
| SourceFile | 类文件 | 记录源文件名称 |
| SourceDebugExtenSion | 类文件 | JDK1.6中新增的属性，SourceDebugExtension属性用于存储额外的调试信息。例如在进行JSP文件调试时，无法通过java堆栈来定位到jsp文件的行号，JSR-45规范为这些非java语言编写，却需要编译成字节码并运行在java虚拟机中的程序提供一个进行调试的标准机制，使用SourceDebugExtension属性就可以用于存储这个标准所新加入的调试信息 |
| Synthetic | 类、方法表、字段表 | 标识方法或字段为编译器自动生成 |
| LocalVariableTypeTable | 类 | JDK1.5新增的属性，它使用特征签名（signature）代替描述符，是为了引入泛型语法之后能描述泛型参数化类型而添加 |
| RuntimeVisibleAnnotations | 类、方法表、字段表 | JDK1.5中新增的属性，为动态注解提供支持。RuntimeVisiableAnnotations属性用于指明那些注解是运行时（实际上运行时就是进行反射调用）可见 |
| RuntimeInvisibleAnnotations | 类、方法表、字段表 | JDK1.5中新增的属性，为动态注解提供支持。RuntimeVisiableAnnotations属性用于指明那些注解是运行时不可见 |
| RuntimeVisible-  ParameterAnnotations | 方法表 | JDK1.5中新增的属性，为动态注解提供支持。作用与RuntimeVisibleAnnotations类似，只是作用对象为方法参数 |
| RuntimeInVisible-  ParameterAnnotations | 方法表 | JDK1.5中新增的属性，为动态注解提供支持。作用与RuntimeInVisibleAnnotations类似，只是作用对象为方法参数 |
| AnnotationDefault | 方法表 | JDK1.5中新增的属性，用于记录注解类元素的默认值 |
| BootstrapMethods | 类文件 | JDK1.7中新增的属性，用于保存invokedynamic指令引用的引导方法限定符 |

#### 1.2.8.1 Code属性

java程序方法体中的代码经过javac编译器处理后，最终变为字节码指令存储在Code属性内。Code属性出现在方法表的属性集合之中，但并非所有的方法表都必须存在这个属性，例如：接口或抽象类中的方法就不存在Code属性。

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attribute\_name\_index | 1 | 指向常量池中CONSTANT\_Utf8\_info项的索引，值为：Code |
| u4 | attribute\_length | 1 | 指示了属性值的长度 |
| u2 | max\_stack | 1 | 操作数栈（Operand Stacks）深度的最大值，jvm运行时根据该值来分配栈帧（Stack Frame）中的操作栈深度 |
| u2 | max\_locals | 1 | 局部变量表所需存储空间，单位为：Slot |
| u4 | code\_length | 1 | 代码字节长度 |
| u1 | code | Code\_length | 用于存储字节码的一系列字节流 |
| u2 | exception\_table\_length | 1 | 异常表的长度 |
| exception\_info | excetion\_table | Exception\_table\_length | 异常表，具体参照：1.2.8.1.1表所示 |
| u2 | attributes\_count | 1 | 属性表的长度 |
| attribute\_info | attributes | Attributes\_count | 属性，Code中可拥有的属性：Signature（LocalVariableTypeTable）、LineNumberTable、StackMapTable、LineNumberTable、 |

特别说明：

* Slot是虚拟机为局部变量分配内存所使用的最小单位，对于基本数据类型（byte、short、char、int、float、boolean、returnAddress）等长度不超过32位的数据类型，每个局部变量占用1slot
* 局部变量表中的slot可共用，当代码执行超出一个局部变量的作用域时，这个局部变量所占用的slot可以被其他局部变量所使用
* code\_length和code用来存储java源程序编译后生成的字节码指令

##### 1.2.8.1.1 Code属性中exception\_info类型

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 数量 |
| u2 | start\_pc | 1 |
| u2 | end\_pc | 1 |
| u2 | handler\_pc | 1 |
| u2 | catch\_type | 1 |

* 当代码在start\_pc～end\_pc(不含end\_pc)行之间出现了类型为catch\_type或其子类的异常（catch\_type为指向一个CONSTANT\_Class\_info型常量的索引），则转到handler\_pc行继续处理，当catch\_type=0，代表任意异常情况都需要转向handler\_pc处进行处理
* 异常表实际上就是java代码的一部分，编译器使用异常表而不是简单的跳转命令来实现java异常及finally处理机制

#### 1.2.8.2 Exceptions属性

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attribute\_name\_index | 1 | 异常表属性名称，指向常量池中CONSTANT\_Utf8\_info的索引，值为：Exceptions |
| u4 | attribute\_length | 1 | 异常表属性的长度 |
| u2 | number\_of\_exceptions | 1 | 异常表的数目 |
| u2 | exception\_index\_table | number\_of\_exceptions | 一个指向常量池中CONSTANT\_Class\_info型常量的索引 |

#### 1.2.8.3 LineNumberTable属性

用于描述java源码行号与字节码行号（字节码偏移量）之间的对应关系

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attribute\_name\_index | 1 | lineNumberTables属性名称，指向常量池中CONSTANT\_Utf8\_info的索引，值为：LineNumberTable |
| u4 | attribute\_length | 1 | lineNumberTables属性的长度 |
| u2 | line\_number\_table\_length | 1 | lineNumberTables表的数目 |
| line\_number\_info | line\_number\_table | line\_number\_table\_length | 描述java源码行号与字节码行号（字节码偏移量）之间的对应关系 |

##### 1.2.8.3.1 line\_number\_ info属性

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | start\_pc | 1 | 字节码行号 |
| u2 | line\_number | 1 | Java源码行号 |

#### 1.2.8.4 LocalVariableTable属性

用于描述栈帧中局部变量表中的变量与java源码中定义的变量之间的关系，它不是运行时必需的属性，但默认会生成到Class文件中，可以在javac中分别使用：-g:none或-g:vars选项来取消或要求生成这项信息

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attribute\_name\_index | 1 | LocalVariableTable属性名称，指向常量池中CONSTANT\_Utf8\_info的索引，值为：LocalVariableTable |
| u4 | attribute\_length | 1 | LocalVariableTable属性的长度 |
| u2 | local\_variable\_table\_length | 1 | LocalVariableTable表的数目 |
| local\_variable\_info | local\_variable\_table | local\_variable\_table\_length | 描述栈帧中局部变量表中的变量与java源码中定义的变量之间的关系 |

##### 1.2.8.4.1 local\_variable\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | start\_pc | 1 | 局部变量的生命周期开始的字节码偏移量 |
| u2 | length | 1 | 局部变量作用返回覆盖的长度 |
| u2 | name\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引 |
| u2 | descriptor\_index/signature | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引 |
| u2 | index | 1 | 该局部变量在栈帧局部变量表中Slot的位置 |

#### 1.2.8.5 SourceFile属性

用于记录生成这个Class文件的源码文件名称。是可选的，可以分别使用javac的-g:none或-g:source来关闭或要求生成这项信息

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attribute\_name\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引，常量值为SourceFile |
| u4 | attribute\_length | 1 | SourceFile属性的长度 |
| u2 | source\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引，常量值为源码文件的文件名 |

#### 1.2.8.6 ConstantValue属性

通知虚拟机自动为静态（Static）变量赋值，若一个变量同时被static和final修饰，且数据类型为基本数据类型或java.lang.String，就生成ConstantValue属性来进行初始化。否则，该静态变量的初始化操作在<clinit>方法中进行初始化

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attribute\_name\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引，常量值为ConstantValue |
| u4 | attribute\_length | 1 | ConstantValue属性的长度，值必须为2 |
| u2 | constantvalue\_index | 1 | 指向常量池中基本数据类型常量的索引，索引值为：CONSTANT\_Long\_info、CONSTANT\_Float\_info、CONSTANT\_Integer\_info、CONSTANT\_Double\_info、CONSTANT\_String\_info |

#### 1.2.8.7 InnerClasses属性

用于记录内部类与宿主之间的关联

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attribute\_name\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引，常量值为InnerClasses |
| u4 | attribute\_length | 1 | InnerClasses属性的长度 |
| u2 | number\_of\_classes | 1 | 代表有多少给内部类信息 |
| inner\_classes\_info | inner\_classes | number\_of\_classes |  |

##### 1.2.8.7.1 inner\_classes\_info

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | inner\_class\_info\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引，内部类的符号引用 |
| u2 | outer\_class\_info\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引，外部类的符号引用 |
| u2 | inner\_name\_index | 1 | 指向常量池CONSTANT\_Utf8\_info型常量的索引，若为匿名内部类，该值为0 |
| u2 | inner\_class\_access\_flags | 1 | 内部类的访问标志 |

#### 1.2.8.8 Deprecated

* Deprecated属性属于标志类型的布尔属性，只存在有和没有的区别，没有属性值的概念
* Deprecated属性用于表示某个类、字段或方法，已经被程序作者定为不在推荐使用，通过在代码中使用@deprecated注解进行设置

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attrubute\_name\_index | 1 | 指向常量池中CONSTANT\_Utf8\_info中的索引值，常量值为: Deprecated |
| u4 | attribute\_length | 1 | 值必须为0X00000000，没有任何属性值需要设置 |

#### 1.2.8.9 Synthetic

* Synthetic属性属于标志类型的布尔属性，只存在有和没有的区别，没有属性值的概念
* Synthetic属性代表该字段或方法并不是由java源码直接产生的，而是由编译器自行添加的，所有由并非用户代码产生的类、方法都应当至少设置Synthetic属性和ACC\_SYNTHETIC标志位中的一项

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attrubute\_name\_index | 1 | 指向常量池中CONSTANT\_Utf8\_info中的索引值，常量值为: Deprecated |
| u4 | attribute\_length | 1 | 值必须为0X00000000，没有任何属性值需要设置 |

#### 1.2.8.10 StackMapTable属性

* StackMapTable属性JDK1.6新增的，作用于Code属性中的属性域中
* StackMapTable属性会在虚拟机类加载字节码验证阶段被新类型检查验证器使用
* 1.7后明确表示，当版本号大于50.0的class文件中如果方法的Code属性中没有附带StackMapTable属性，那就意味着它带有一个隐式的StackMap属性，这个属性作用等同于number\_of\_entries值为0的StackMapTable属性

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attrubute\_name\_index | 1 | 指向常量池中CONSTANT\_Utf8\_info中的索引值，常量值为: StackMapTable |
| u4 | attribute\_length | 1 | 属性的长度 |
| u2 | number\_of\_entries | 1 | stack\_map\_frame集合的数目 |
| stack\_map\_frame | stack\_map\_frame entries | number\_of\_entries | stack\_map\_frame |

#### 1.2.8.11 Signature属性

* Signature属性JDK1.5新增的，作用于类、字段表、方法表中
* Signature属性会在虚拟机类加载字节码验证阶段被新类型检查验证器使用
* 1.7后明确表示，当版本号大于50.0的class文件中如果方法的Code属性中没有附带StackMapTable属性，那就意味着它带有一个隐式的StackMap属性，这个属性作用等同于number\_of\_entries值为0的StackMapTable属性

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attrubute\_name\_index | 1 | 指向常量池中CONSTANT\_Utf8\_info中的索引值，常量值为: Signature |
| u4 | attribute\_length | 1 | 属性的长度 |
| u2 | signature\_index | 1 | 指向常量池中CONSTANT\_Utf8\_info中的索引值，表示类签名、方法类型签名或字段类型签名 |

#### 1.2.8.12 BootstrapMathods属性

* BootstrapMathods属性JDK1.7新增的，作用于类文件的属性表中

|  |  |  |  |
| --- | --- | --- | --- |
| 类型 | 名称 | 数量 | 含义 |
| u2 | attrubute\_name\_index | 1 | 指向常量池中CONSTANT\_Utf8\_info中的索引值，常量值为: BootstrapMathods |
| u4 | attribute\_length | 1 | 属性的长度 |
| u2 | number\_bootstrap\_methods | 1 | bootstrap\_method的数目 |
| bootstrap\_method | bootstrap\_methods | number\_bootstrap\_methods |  |

##### 1.2.8.21.1 bootstrap\_method

|  |  |  |
| --- | --- | --- |
| 类型 | 名称 | 数量 |
| u2 | bootstrap\_method\_ref | 1 |
| u2 | num\_bootstrap\_arguments | 1 |
| u2 | bootstrap\_arguments | 1 |

* bootstrap\_method\_ref：值必须是一个对常量池的有效索引，常量池在该索引出的值必须是一个CONSTANT\_MethodHandle\_info结构
* num\_bootstrap\_arguments：值给出了bootstrap\_arguments[]数组成员的数量
* bootstrap\_arguments[]：每个成员必须是一个对常量池的有效索引，常量池在该索引出必须是如下结构之一：CONSTANT\_String\_info、CONSTANT\_Class\_info、CONSTANT\_Integer\_info、CONSTANT\_Long\_info、CONSTANT\_Float\_info、CONSTANT\_Double\_info、CONSTANT\_MethodHandle\_info、CONSTANT\_MethoodType\_info

## 1.3.0综合案例

  

### 1.3.1 magic与minor\_version、major\_version

|  |
| --- |
| CA FE BA BE 00 00 00 34 |

### 1.3.2 Constant pool与Constant pool count

|  |
| --- |
| 003D 0A000B0028 ......... 070033 ........... 0100063C696E69743E 010003282956 .......... ....... 0C00160017 ......... ......... ......... ......... ......... ......... ......... ......... 010015284C6A6176612F6C616E672F4F626A6563743B295A |

解释：

* **003D**，表示的含义：Constant poo count = 3d=61,即：常量池中常量表的个数为60，由于常量池中的常量表个数是从1开始计数的
* **0A000B0028**，表示的含义：常量池第1个常量表，其中0A=10(1个字节)表示常量表的项目类型，即：CONSTANT\_Methodref\_info；000B=11(2个字节)，表示指向声明方法的类或接口描述符CONSTANT\_ Class\_info的索引项，值为指向常量池中第11张表；0028=40(2个字节)，表示指向名称以及类型描述符CONSTANT\_NameAndType\_info的索引项，值为指向常量池中第**40**张表；
* **070033**，表示的含义：常量池第**10**个常量表，其中07=7(1个字节)表示常量表的项目类型，即：CONSTANT\_Class\_info；0033=51(2个字节)，表示指向声明方法的类或接口描述符CONSTANT\_ Utf8\_info的索引项，值为指向常量池中第51张表；
* **0100063C696E69743E**，表示的含义：常量池中第**22**项常量表，其中01=1(1个字节)表示常量表的项目类型，即：CONSTANT\_Utf8\_info；0006=6(2个字节)，表示UTF-8编码的字符串占用的字节数；3C696E69743E（16进制数）对应的ASCII的值为：<init>；
* **010003282956**，表示的含义：常量池中第**23**项常量表，其中01=1(1个字节)表示常量表的项目类型，即：CONSTANT\_Utf8\_info；0003=3(2个字节)，表示UTF-8编码的字符串占用的字节数，即该utf8编码的字节数为3个字节；282956（16进制数）对应的ASCII的值为：()V
* **0C00160017**，表示的含义：常量池中第**40**项常量表，其中0C=12(1个字节)表示常量表的项目类型，即：CONSTANT\_ NameAndType \_info；0016=22(2个字节)，表示指向字段或方法常量项的索引，即指向常量池第**22**项表；0017=23（16进制数），表示指向字段或方法描述符常量项的索引，即指向常量池第**23**项表
* **0100106A6176612F6C616E672F4F626A656374**，表示的含义：常量池中第**50**项常量表，其中01=1(1个字节)表示常量表的项目类型，即：CONSTANT\_Utf8\_info；0010=16(2个字节)，表示UTF-8编码的字符串占用的字节数，即该utf8编码的字节数为3个字节；**6A6176612F6C616E672F4F626A656374**（16进制数）对应的ASCII的值为：java/lang/Object
* **010015284C6A6176612F6C616E672F4F626A6563743B295A**，表示的含义：常量池中第**60**项常量表，其中01=1(1个字节)表示常量表的项目类型，即：CONSTANT\_Utf8\_info；0015=21(2个字节)，表示UTF-8编码的字符串占用的字节数，即该utf8编码的字节数为3个字节；**284C6A6176612F6C616E672F4F626A6563743B295A**（16进制数）对应的ASCII的值为：(Ljava/lang/Object;)Z

### 1.3.2访问标志

|  |
| --- |
| 0421 |

### 1.3.3类索引和父类索引

|  |
| --- |
| 000A000B0002000C000D |

* **000A**=10，指向类描述CONSTANT\_ Class\_info的索引项，即指向常量池中第10项表
* **000B**，指向声明父类描述描述CONSTANT\_ Class\_info的索引项，即指向常量池中第11项表
* **0002**，指定该类实现的接口数，为2
* **000C**与**000D**，表示实现的接口的描述CONSTANT\_ Class\_info的索引项，即分别指向常量池中第12项和13项表

### 1.3.4 字段表

|  |
| --- |
| 00030002000E000F00010010000000020011 |

* **0003**，表示filed\_count即字段表的长度
* **0002000E000F00010010000000020011**，为字段表中第一个字段
* **0002**：access\_flag，即：private
* **000E**，指向常量池中第14项CONSTANT\_Uft8\_info值，值为：filters
* **000F**，指向常量池中第15项ONSTANT\_Uft8\_info值，值为：Ljava/util/List;
* **0001**，表示attribute\_length，即只有一个属性
* **0010**，指向常量池中第16项CNSTANT\_Uft8\_info值，值为：Signature
* **0000000020011**，**000000002**表示属性Signature的长度值为2；**0011**，指向常量池中第17项CONSTANT\_Uft8\_info的索引值，值为：Ljava/util/List<Lcom/spdb/ocr/factory/netty/filters/NettyHttpFilter;>;

### 1.3.5方法表

|  |
| --- |
| 0005000100160017000100180000005A00030001000000202AB700012A01B500022A2AB60003B80004B500052ABB000659B70007B50002B1000000020019000000160005000000160004000F0009001300140017001F0018001A0000000C000100000020001B001C0000 |

* **0005**，表示methods\_count即方法表的个数
* 000100160017000100180000005A00030001000000202AB700012A01B500022A2AB60003B80004B500052ABB000659B70007B50002B1000000020019000000160005000000160004000F0009001300140017001F0018001A0000000C000100000020001B001C0000，是类文件AbstractNettyHttpRequest第一个方法<init>
* 0001，表示第一个方法<init>的访问标识，即是public
* 0016，表示name\_index，指向常量池：CONSTANT\_utf8\_info项的索引值,即常量池第22项表，该索引值所存储的值为：<init>
* 0017，表示descriptor\_index,指向常量池：CONSTANT\_utf8\_info项的索引值,即常量池第23项表，该索引值所存储的值为：()V
* 0001，表示attributes\_count属性的数量，为1
* 00180000005A00030001000000202AB700012A01B500022A2AB60003B80004B500052ABB000659B70007B50002B1000000020019000000160005000000160004000F0009001300140017001F0018001A0000000C000100000020001B001C0000为方法<init>的第一个属性Code
* 0018，attribute\_name\_index，指向常量池：CONSTANT\_utf8\_info项的索引值,即常量池第24项表，该索引值所存储的值为：Code，表明该属性项为Code项
* 0000005A，attribute\_length，该Code的属性的长度为：0x5A=90个字节
* 0003，max\_stack，方法中操作数栈的深度：3
* 0001，max\_locals，代表方法中局部变量表所需存储空间为1
* 00000020，表示Code\_length的长度，即经过编译期编译后字节码的长度为32
* 02AB700012A01B500022A2AB60003B80004B500052ABB000659B70007B50002B1为<init>方法的方法体翻译过来的指令就是：

|  |
| --- |
| 0: aload\_0  1: invokespecial #1 // Method java/lang/Object."<init>":()V  4: aload\_0  5: aconst\_null  6: putfield #2 // Field filters:Ljava/util/List;  9: aload\_0  10: aload\_0  11: invokevirtual #3 // Method java/lang/Object.getClass:()Ljava/lang/Class;  14: invokestatic #4 // Method org/slf4j/LoggerFactory.getLogger:(Ljava/lang/Class;)Lorg/slf4j/Logger;  17: putfield #5 // Field LOGGER:Lorg/slf4j/Logger;  20: aload\_0  21: new #6 // class java/util/ArrayList  24: dup  25: invokespecial #7 // Method java/util/ArrayList."<init>":()V  28: putfield #2 // Field filters:Ljava/util/List;  31: return |

* 0000，表示exception\_table\_length，即表示该方法异常表个数为0
* 0002，表示attributes\_count，即表示该方法的属性个数为2
* 0019000000160005000000160004000F0009001300140017001F0018001A0000，表示Code属性的第一个attribute，0019表示该属性为：LineNumberTable，00000016表示attribute\_length的长度为：25，000000160004000F0009001300140017001F0018分别表示：

|  |
| --- |
| line 22: 0  line 15: 4  line 19: 9  line 23: 20  line 24: 31 |

* 001A0000000C000100000020001B001C0000表示Code属性的第二个attribute，001A表示该属性为：LocalVariableTable，0000000C表示attribute\_length的长度为：12，000100000020001B001C0000，0001表示local\_variable\_table\_length的长度为1，00000020001B001C0000分别表示如下
* Start ：0--start\_pc，0000
* length：32-length，0020
* name：this-name\_index指向常量池CONSTANT\_Utf8\_info，索引值为：001B
* Signature: this-descriptor/Signature，指向指向常量池CONSTANT\_Utf8\_info，索引值为：001C，值为Signature
* Slot: 0-index，该局部变量在栈帧局部变量中Slot的位置

### 1.3.6 类文件中的attribute\_count和attribute

|  |
| --- |
| 00010026000000020027 |

## 1.4.字节码生成技术

参看文档：[CGLIB(Code Generation Library) 介绍与原理 | 菜鸟教程 (runoob.com)](https://www.runoob.com/w3cnote/cglibcode-generation-library-intro.html)

[字节码增强技术探索 - 美团技术团队 (meituan.com)](https://tech.meituan.com/2019/09/05/java-bytecode-enhancement.html)

* 采用：CallBack、CallbackFilter设计，来动态地过滤
* CallBack种类及CallBackType类型：
  + NoOp：直接调用目标方法
  + MethodInterceptor：
  + InvocationHandler：对jdk1.2的增强
  + LazyLoader：只在调用代理对象的目标第一个方法时执行loadObject(),且只执行一次
  + Dispatcher：每次调用代理对象的方法时都执行一次
  + FixedValue：调用代理方法时返回固定的值
* 置入方式：通过实现：CallbackGenerator接口来进行置入
  + MethodInterceptorGenerator
  + FixedValueGenerator
  + LazyLoaderGenerator
* 置入的过程：
  + create()
  + New ClassLoaderData()
  + AbstractClassGenerator.generate()
  + DefaultGeneratorStrategy.generate()
  + Enhancer.generateClass()
  + Enhancer.emitMethods()
  + CallbackGenerator.generate()

# 第五章、类加载机制

## 1.1 概念

虚拟机在把类的数据从class文件加载到内存时，并对数据进行校验、转换解析、初始化，并形成能被虚拟机使用的类数据，这就是java的类加载机制

## 1.2类加载生命周期

加载

Loading

验证

Verfication

准备

Preparation

解析

Resolution

卸载

Unloading

使用

Use

初始化

Initialization

连接（Linking）

* 触发类初始化的五个基本条件：（又称之为：主动引用）
* 遇到new、getstatic、putstatic、invokestatic指令时，若该类没有初始化就初始化该类
* 使用java.lang.reflect包的方法对类进行反射调用时，若该类没有初始化，就先初始化该类
* 当初始化一个类时，若其父类还未初始化，则先初始化其父类，再初始化其自己
* 当虚拟机启动时，虚拟机需要指定一个要执行的主类（包含main方法的那个类），虚拟机会先初始化这个主类
* 当使用jdk1.7的动态语言支持时，如果一个java.lang.MethodHandle实例最后的解析结果REF\_getStatic、REF\_putStatic、REF\_invokeStatic的方法句柄，并且这个方法句柄所对应的类没有进行过初始化，则需要先触发其初始化
* 类与接口有所区别的地方就在与触发类初始化的第3个条件上，即：接口在进行初始化是，并不要求其父接口全部都完成了初始化 ，只有真正使用到父类接口时才会初始化

## 1.3 类加载过程

类加载过程：加载、验证、解析、初始化

### 1.3.1加载

虚拟机在加载阶段，需要完成以下三个动作

* 通过一个类的全限定名来获取此类的二进制字节流
* 将这个字节流所代表的静态存储接口转化为方法区的运行时数据结构
* 在内存中生成一个代表这个类的java.lang.Class对象，作为方法区这个类的各种数据的访问入口

### 1.3.2 数据类的加载

* 如果数据组的组件类型（Component Type，指的是数组去掉一个维度的类型）是引用类型，那就采用递归的方式去加载这个组件类型，数组C将在加载该组件类型的类加载器的类名称空间上被标识
* 若数组的组件类型不是引用类型（例如int[]数组），java虚拟机将会把数组C标记为与引导类加载器关联
* 数组类的可见性与它的组件类型的可见性一致，若组件类型不是引用类型，那数组类的可见性将默认为public

### 1.3.3 加载后

加载阶段完成后，虚拟机外部的二进制字节流就按照虚拟机所需的格式存储在方法区之中，然后在实例化一个java.lang.Class类对象作为访问方法区中这些类型数据的外部接口

## 1.4验证

字节码的验证阶段和字节码的加载阶段没有明显的先后顺序，一般体现为交替地进行，验证阶段大致上会完成如下4个阶段的校验动作：文件格式检验、元数据验证、字节码验证、符号引用验证

### 1.4.1 文件格式校验

该验证阶段的主要目的：保证输入的字节流能正确地解析并存储于方法区之内，格式上符合描述一个java类型信息的要求

### 1.4.2 元数据验证

对加载到方法区内的字节码描述信息进行语义分析，以保证其描述的信息符合java语言规范的要求，例如：这个类是否有父类且这个父类是否继承了不允许被继承的类

### 1.4.3 字节码验证

是这个校验过程中最复杂的，通过数据流和控制流分析确定程序语义是合法的、符合逻辑的

#### 1.4.3.1 数据流验证

在jdk1.7之后改为通过：对StackMapTable属性表中的类型检查。

#### 1.4.3.2 符号引用验证

符号验证的目的是确保解析动作能正常执行，如果无法通过符号引用验证，那么将抛出一个java.lang.IncompatibleClassChangeError异常的子类，例如：java.lang.IllegalAccessError

## 1.5 准备阶段

是正式为类变量分配内存并设置类变量初始值的阶段，这些变量所使用的内存都将在方法区中进行分配。例如：public static int value = 123那变量在准备阶段过后的初始值为0，而不是123，因为这时候尚未执行任何java方法，而把value赋值为123的putstatic指令是程序被编译后，存放与类构造器<client>方法之中，故value赋值为123的动作是在初始化阶段才会被执行的。例如public static final int value = 124，那么value的赋值是在准备阶段产生的

## 1.6 解析

解析阶段是虚拟机将常量池内的符号引用替换为直接引用的过程。

* **符号引用**：符号引用一组符号来描述所引用的目标，符号可以是任何形式的字面量，只要使用时能无歧义地定位到目标即可
* **直接引用**：直接引用可以是直接指向目标的指针、相对偏移量、是一个能间接定位到目标的句柄
* 解析动作主要是针对7类符合引用进行的：CONSTANT\_Class\_info、CONSTANT\_Fieldref\_info、CONSTANT\_Methodref\_info、CONSTANT\_InterfaceMethodref\_info、CONSTANT\_MethodType\_info、CONSTANT\_MethodHandle\_info、CONSTANT\_InvokeDynamic\_info

### 1.6.1 类解析

* 将符号引用所对应的全限定名称对应的class文件记载到内存中
* 若符号引用所对应的类型为数组对象，这先加载类型元素对象然后再表明数组维度和元素的数组对象
* 在解析完成后，在通过对符号引用的所属数据类型进行访问权限控制，若不具备访问权限，则抛出java.lang.IllegalAccessError异常

### 1.6.2 字段解析

首先对字段表内class\_index项中索引CONSTANT\_Class\_info符号引用进行解析（即：进行类解析），也就是字段所属的类或接口的符号引用。若解析完成，且解析后的结果为C类型

* 若C本身就包含了简单名称和字段描述都与目标相匹配的字段，这返回这个字段的直接引用，即：属于本类
* 否则，若C中实现了接口或继承了类，在按照继承关系从下往上递归搜索各个接口和它的父接口，若接口中包含了简单名称和字段描述父都与目标相匹配的字段，则返回这个字段的直接引用，查找结束，即：该字段属于父类或接口
* 若查找过程成功完成，将会对这个字段进行权限校验，如果发现不具备对字段的权限校验，抛出java.lang.IllegalAccessError异常

### 1.6.3 类方法解析

类方法解析的第一个步骤与字段解析一样，也需要解析出类方法表的从class\_index项中索引的方法所属的类或接口的符号引用，若解析成功，我们用C来表示这个类

* 若C为一个接口，直接抛出java.lang.IncompatibleClassChangeError异常
* 在类C中递归查找是否有简单名称和描述符都与目标相匹配的方法，若有则返回这个方法的直接引用，查找结束
* 否则在类C实现的接口列表及它们的父接口之中递归查找是否有简单名称和描述符都与目标相匹配的方法，若有，则说明类C是一个抽象类，这时查找结束，抛出java.lang.AbstractMentodError异常
* 否则，宣告方法查找失败，抛出java.lang.NoSuchMethodError

### 1.6.4 接口方法解析

类方法解析的第一个步骤与字段解析一样，也需要解析出类方法表的从class\_index项中索引的方法所属的类或接口的符号引用，若解析成功，我们用C来表示这个类

* 与类方法解析不同，若在接口方法表中发现class\_index中的索引C是个类，而不是接口，则直接抛出java.lang.IncompatiClassChangeError异常
* 否则，在接口C中查找是否有简单名称和描述符都与目标相匹配的方法，如果有则返回这个方法的直接引用，查找结束
* 否则，在接口C的父接口中递归查找，知道java.lang.Object类为止，看是否有简单名称和描述符都与目标相匹配的方法，如果有则返回这个方法的直接引用，查找结束

## 1.7 初始化

初始化过程就是执行类构造器<client>方法的过程

* <clinet>方法是由编译器自动收集类中的所有类变量的赋值动作和静态代码块（static｛｝）中语句合并产生的，编译器收集的顺序是由语句在源文件中出现的顺序所决定的，静态语句块只能访问到定义在静态语句块之前的变量，定义在它之后的变量，在前面的静态代码块可以赋值，但不能访问
* <client>方法与类的构造方法（实例构造器<init>方法）不同，它不需要显式地调用父类构造器，虚拟机会保证在子类的<client>()方法执行之前，父类的<client>()方法执行完毕
* <client>()方法对于类或接口而言并不是必需的，若一个类中没有静态代码块，也没有对静态变量的赋值操作，那么编译器不为这个类生成<client>()方法
* 接口中不能使用静态代码块，但仍然有变量初始化的赋值操作，执行接口的<client>()方法不需要先执行父接口的<client>方法，只有当父接口中定义的变量使用时，父接口才会初始化
* 虚拟机会保证一个类的<client>()方法在多线程环境下被正确地加锁、同步。

## 1.8 类加载器

通过一个类的全限定名来获取描述此类的二进制字节流这个过程被称之为：类加载器

### 1.8.1 类与类加载器

对于任意一个类，都需要由加载它的类加载器和这个类本身一同确定其在java虚拟机中的唯一性。

### 1.8.2 双亲委派模型

启动类加载器

Bootstrap ClassLoader

扩展类加载器

Extension ClassLoader

应用程序加载器

Bootstrap ClassLoader

自定义加载器

User ClassLoader

* **启动类加载器（Bootstrap ClassLoader）**，有虚拟机自动实现，负责加载存放在<JAVA\_HOME>/lib目录中的，或者-Xbootclasspath参数所指定的路径中的并且是虚拟机识别的类库加载到虚拟机内存中
* **扩展类加载器（Extension ClassLoader）**，这个类加载器由sun.misc.Launcher$ExtClassLoader实现，它负责加载<JAVA\_HOME>/lib/ext目录中的或者被java.ext.dirs系统变量所指定的路径中的所有类库，开发者可以直接使用扩展类加载器
* **应用程序类加载器（Application ClassLoader）**，这个类加载器由sun.misc.Launcher$AppClassLoader实现，可以通过ClassLoader.getSystemClassLoader()方获得。它负责加载用户类路径（Classpath）上所指定的类库。若用户没有自定义过自己的类加载器，一般情况下就是程序中默认的类加载器
* 参考Launcher.class中的内容，里面有AppClassLoader、ExtClassLoader的实现
* 双亲委派模型的工作过程：如果一个类加载器收到了类加载请求，它首先不会自己去尝试加载这个类，而是把这个请求委派给父类加载器去完成，每一个层次的类加载器都是如此，因此所有的加载请求最终都应该传送到顶层的启动类加载器中，只有当父加载器反馈自己无法完成这个加载请求（它的搜索范围中没有找到所需的类）时，子加载器才会尝试自己去加载

### 1.8.3 线程上下文类加载器（Thread Context ClassLoader）

这个类加载器可以通过java.lang.Thread类的setContextClassLoaser()方法进行设置，如果创建线程时还未设置，它将从父线程中继承一个，如果在应用程序的全局范围内都没有设置过的话，那这个类加载器默认就是应用程序类加载器（Application Loader）

### 1.8.4 自定义类加载器

#### 采用双亲委派机制

|  |
| --- |
| static class CustomClassLoader2 extends ClassLoader {  @Override  protected Class<?> findClass(String name) throws ClassNotFoundException {  String classPath = name.replace(".", "/").concat(".class");  InputStream inputStream = Demo1.class.getClassLoader().getResourceAsStream(classPath);  try {  int length = inputStream.available();  byte[] bytes = IOUtils.readFully(inputStream, length);  return defineClass(name,bytes,0,bytes.length);  } catch (IOException e) {  e.printStackTrace();  }  return null;  }  } |

#### 不采用双亲委派机制

|  |
| --- |
| static class CustomClassLoader extends ClassLoader {  @Override  public Class<?> loadClass(String name) throws ClassNotFoundException {  if (name.startsWith("com.github.liuxg.jvm.classload")) {  String classPath = name.replace(".", "/").concat(".class");  InputStream inputStream = Demo1.class.getClassLoader().getResourceAsStream(classPath);  try {  int length = inputStream.available();  byte[] bytes = IOUtils.readFully(inputStream, length);  return defineClass(name,bytes,0,bytes.length);  } catch (IOException e) {  e.printStackTrace();  }  return null;  }  return super.loadClass(name);  }  } |

#### 使用Thread.currentThread().getContextClassLoader()

典型应用：ServiceLoad.load、DriverManager

参看：https://zhuanlan.zhihu.com/p/67665359

|  |
| --- |
| package java.awt;  /\*\*  \* An interface for events that know how to dispatch themselves.  \* By implementing this interface an event can be placed upon the event  \* queue and its <code>dispatch()</code> method will be called when the event  \* is dispatched, using the <code>EventDispatchThread</code>.  \* <p>  \* This is a very useful mechanism for avoiding deadlocks. If  \* a thread is executing in a critical section (i.e., it has entered  \* one or more monitors), calling other synchronized code may  \* cause deadlocks. To avoid the potential deadlocks, an  \* <code>ActiveEvent</code> can be created to run the second section of  \* code at later time. If there is contention on the monitor,  \* the second thread will simply block until the first thread  \* has finished its work and exited its monitors.  \* <p>  \* For security reasons, it is often desirable to use an <code>ActiveEvent</code>  \* to avoid calling untrusted code from a critical thread. For  \* instance, peer implementations can use this facility to avoid  \* making calls into user code from a system thread. Doing so avoids  \* potential deadlocks and denial-of-service attacks.  \*  \* @author Timothy Prinzing  \* @since 1.2  \*/  public interface ActiveEvent {  /\*\*  \* Dispatch the event to its target, listeners of the events source,  \* or do whatever it is this event is supposed to do.  \*/  public void dispatch();  }  public class CustomActiveEvent implements ActiveEvent {  @Override  public void dispatch() {  System.out.println("customer");  }  }  public static void main(String[] args) {  ServiceLoader<ActiveEvent> events = ServiceLoader.load(ActiveEvent.class);  events.forEach(activeEvent -> activeEvent.dispatch());  } |

# 第六章、字节码执行引擎

## 1.1 概述

在Java虚拟机规范中指定了虚拟机字节码执行引擎的概念模型，这个概念模型成为各种虚拟机执行引擎的统一外观

## 1.2栈帧结构

栈帧是用于支持虚拟机进行方法调用和方法执行的数据结构，他是虚拟机运行时数据区中虚拟机栈的栈元素

### 1.2.1 栈帧的基本结构

栈帧中存储了方法的局部变量表、操作数栈、动态连接、方法返回地址。每一个方法从调用开始至执行完成的过程，都对应着一个栈帧在虚拟机栈里面从入栈到出栈的过程

#### 1.2.1.1 局部变量表

是一组变量存储空间，用于存放方法参数和方法内部定义的局部变量，在java源码编译成class文件时，就在方法的Code属性的max\_locals数据项中确定了其大小。单位为：slot，但java虚拟机并没有明确指定一个sloat应占用的内存空间大小。

* 引用（reference）的作用：1、从此引用中直接或间接地查找到对象在java堆中的数据存放的起始地址索引（访问java堆中的对象）；2、此引用中直接或间接地查找到对象所属数据类型在方法区中的存储的类型信息（class）
* 对于执行实例方法时，局部变量表中第0位索引的Slot默认是用于传递方法所属对象实例的引用，即：this
* 局部变量表中定义的slot存储空间是可以共用的

#### 1.2.1.2 操作数栈

操作数栈（Operand Stack）也常成为操作栈，它是一个先进后出（Last In First Out，LIFO）栈。同局部变量表一样，操作数栈的最大深度也在编译时写入到Code属性的max\_stacks数据项中。32为数据类型所占的栈容量为1，64位数据类型所占的栈容量为2。

#### 1.2.1.3 动态连接

每个栈帧都包含一个指向运行时常量池中该栈帧所属方法的引用，持有这个引用视为了支持方法调用过程中的动态连接（Dynamic Linking）。class文件的常量池中存有大量的符号引用，字节码的方法调用指令就以常量池中指向方法的符号引用作为参数。

* 静态连接：这些符号引用一部分在类加载或者第一次使用的时候就转化为直接引用，这种转化被称为静态引用。
* 动态连接：这些符号引用在每一次运行期间转化为直接引用，这种转化方式称为动态连接

#### 1.2.1.4 方法返回地址

方法退出的过程实际上等同于：把当前栈帧出栈，故退出时可能执行的操作有：恢复上层方法的局部变量表和操作数栈，把返回值（如果有的话）压入调用者栈帧的操作数栈中，调整PC计数器的值以指向方法调用指令后面的一条指令

* 正常退出：执行引擎遇到任意一个方法的返回指令，这种退出方式称为正常退出
* 异常退出：在方法执行过程中遇到了异常，并且这个异常没有在方法体得到处理，这种退出方法方式称之为：异常退出

### 1.2.2 当前栈帧与当前方法

在活动线程中，只有位于栈顶的栈帧才是有效的，成为当前栈帧（Current Stack Frame），与当前栈帧相关联的方法称之为当前方法（Current Method）

## 1.3 方法调用

方法调用不等同于方法执行，方法调用阶段唯一的任务就是确定被调用方法的版本（即调用哪一个方法），暂时还不涉及方法内部的具体运行过程

## 1.4 方法解析

在类加载的解析阶段，就能将class文件中目标方法的符号引用转化为直接引用。调用目标的程序代码写好，编译器进行编译时就必需确定下来，这类方法的调用称之为：解析。可在编译器就被确定的方法引用有：静态方法、私有方法、实例构造器、父类方法、final修饰的方法对应的：方法调用指令：invokestatic、invokespecial。

## 1.5 方法调用指令

* invokestatic：调用静态方法
* invokespecial：调用实例构造器<init>方法、私有方法和父类方法
* invokevirtual：调用所有的虚方法
* invokeinterface：调用接口方法，会在运行时在确定一个实现此接口的对象
* invokedynamic：先在运行时动态解析出调用点限定符所引用的方法，然后再执行该方法，invokestatic、invokespecial、invokevirtual、invokeinterface4条调用指令，分派逻辑是固化在虚拟机内部的，而invokedynamic指令的分派逻辑是由用户所设定的引导方法决定的

## 1.6分派

### 1.6.1 静态分派

|  |
| --- |
| **public class** StaticDispatch {  **static abstract class** Human {}   **static class** Man **extends** Human {}   **static class** WoMen **extends** Human {}   **public void** sayHello(Human human) {  System.***out***.println(**"hello,guy!"**);  }   **public void** sayHello(Man man) {  System.***out***.println(**"hello,gentleman!"**);  }   **public void** sayHello(WoMen woMen) {  System.***out***.println(**"hello,lady!"**);  }   **public static void** main(String[] args) {  Human man = **new** Man();  Human women = **new** WoMen();  StaticDispatch staticDispatch = **new** StaticDispatch();  staticDispatch.sayHello(man);  staticDispatch.sayHello(women);  }  } |
| **public static void main(java.lang.String[]);**  **descriptor: ([Ljava/lang/String;)V**  **flags: ACC\_PUBLIC, ACC\_STATIC**  **Code:**  **stack=2, locals=4, args\_size=1**  **0: new #7 // class com/spdb/ocr/jvm/StaticDispatch$Man**  **3: dup**  **4: invokespecial #8 // Method com/spdb/ocr/jvm/StaticDispatch$Man."<init>":()V**  **7: astore\_1**  **8: new #9 // class com/spdb/ocr/jvm/StaticDispatch$Woman**  **11: dup**  **12: invokespecial #10 // Method com/spdb/ocr/jvm/StaticDispatch$Woman."<init>":()V**  **15: astore\_2**  **16: new #11 // class com/spdb/ocr/jvm/StaticDispatch**  **19: dup**  **20: invokespecial #12 // Method "<init>":()V**  **23: astore\_3**  **24: aload\_3**  **25: aload\_1**  **26: invokevirtual #13 // Method sayHello:(Lcom/spdb/ocr/jvm/StaticDispatch$Human;)V**  **29: aload\_3**  **30: aload\_2**  **31: invokevirtual #13 // Method sayHello:(Lcom/spdb/ocr/jvm/StaticDispatch$Human;)V**  **34: return**  **LineNumberTable:**  **line 29: 0**  **line 30: 8**  **line 31: 16**  **line 32: 24**  **line 33: 29**  **line 34: 34**  **LocalVariableTable:**  **Start Length Slot Name Signature**  **0 35 0 args [Ljava/lang/String;**  **8 27 1 man Lcom/spdb/ocr/jvm/StaticDispatch$Human;**  **16 19 2 women Lcom/spdb/ocr/jvm/StaticDispatch$Human;**  **24 11 3 staticDispatch Lcom/spdb/ocr/jvm/StaticDispatch;**  **}** |

虚拟机在重载时（准确地说是编译期）是通过参数的静态类型而不是实际类型作为判断依据的。所有依赖静态类型来定位方法执行版本的分配动作称之为：静态分派。静态分派的典型应用：方法重载

### 1.6.2 动态分派

invokevirtual指令执行的第一步就是在运行期确定接收者的实际类型，所以两次调用中的invokevirtual指令把常量池中的类方法符号引用解析到了不同的直接引用上，这个过程就是java语言中重写的本质。我们把这种在运行期根据实际类型确定方法执行版本的过程称为动态分派。

invokevirtual解析过程：

* 找到操作数栈顶的第一个元素所指向的对象的实际类型，记作C
* 如果在类型C中找到与常量中描述符和简单名称都相符的方法。则进行访问权限校验，如果通过则返回这个方法的直接引用，查找过程结束；如果不通过，则返回java.lang.IllegalAccessError异常
* 否则，按照继承关系从下往上依次对C的各个父类进行第2步的搜索和验证过程
* 如果始终没有找到合适的方法，则抛出java.lang.AbstractmethodError异常。

|  |
| --- |
| **public class** DynamicDispatch {  **static abstract class** Human {  **protected abstract void** sayHello();  }   **static class** Man **extends** Human {   @Override  **protected void** sayHello() {  System.***out***.println(**"hello,gentleman!"**);  }  }   **static class** Woman **extends** Human {  @Override  **protected void** sayHello() {  System.***out***.println(**"hello,lady!"**);  }  }   **public static void** main(String[] args) {  Human woman = **new** Woman();  Human man = **new** Man();  woman.sayHello();  man.sayHello();  man = **new** Woman();  man.sayHello();  } } |
| **public static void main(java.lang.String[]);**  **descriptor: ([Ljava/lang/String;)V**  **flags: ACC\_PUBLIC, ACC\_STATIC**  **Code:**  **stack=2, locals=3, args\_size=1**  **0: new #2 // class com/spdb/ocr/jvm/DynamicDispatch$Woman**  **3: dup**  **4: invokespecial #3 // Method com/spdb/ocr/jvm/DynamicDispatch$Woman."<init>":()V**  **7: astore\_1**  **8: new #4 // class com/spdb/ocr/jvm/DynamicDispatch$Man**  **11: dup**  **12: invokespecial #5 // Method com/spdb/ocr/jvm/DynamicDispatch$Man."<init>":()V**  **15: astore\_2**  **16: aload\_1**  **17: invokevirtual #6 // Method com/spdb/ocr/jvm/DynamicDispatch$Human.sayHello:()V**  **20: aload\_2**  **21: invokevirtual #6 // Method com/spdb/ocr/jvm/DynamicDispatch$Human.sayHello:()V**  **24: new #2 // class com/spdb/ocr/jvm/DynamicDispatch$Woman**  **27: dup**  **28: invokespecial #3 // Method com/spdb/ocr/jvm/DynamicDispatch$Woman."<init>":()V**  **31: astore\_2**  **32: aload\_2**  **33: invokevirtual #6 // Method com/spdb/ocr/jvm/DynamicDispatch$Human.sayHello:()V**  **36: return**  **LineNumberTable:**  **line 29: 0**  **line 30: 8**  **line 31: 16**  **line 32: 20**  **line 33: 24**  **line 34: 32**  **line 35: 36**  **LocalVariableTable:**  **Start Length Slot Name Signature**  **0 37 0 args [Ljava/lang/String;**  **8 29 1 woman Lcom/spdb/ocr/jvm/DynamicDispatch$Human;**  **16 21 2 man Lcom/spdb/ocr/jvm/DynamicDispatch$Human;** |

### 1.6.3单分派和多分派

* 宗量：方法的接收者与方法的参数统称为方法的宗量
* 根据分派基于多少种宗量，可以将分派划分为单分派和多分派两种
* 单分派是基于一个宗量对目标方法进行选择
* 多分派是基于一个以上的宗量对目标方法进行选择
* java语言中静态分派属于多分派（编译期获知）
* java语言中多态分派是单分派

## 1.7 分派实现

### 1.7.1 方法表

* 动态分派的不足: 由于动态分派在java中是非常频繁的动作，而且动态分派的方法版本选择过程需要运行时在类的方法元数据中搜索合适的目标方法，故在正在实现上不会真的进行如此频繁的搜索。
* 面对动态分派的不足，最常用的“稳定优化”手段就是为类在方法区（内存局域）中建立一个虚方法表（virtual method table，简称：vtable，与此对应地，在invokeinterface执行时也会用到的接口方法表：interface method Table，简称itable），使用虚方法表索引来代替元数据查找以提高性能
* 若某个方法在子类中没有被重写，那子类的虚方法表里面的地址入口和父类相同方法的地址入口是一致的，都指向父类的实现入口
* 比如：子类Son方法hardChoice(QQ)和hardChoice(\_360)均指向自己

|  |
| --- |
| **public class** Dispatch {  **static class** QQ {}  **static class** \_360 {}  **public static class** Father {  **public void** hardChoice(QQ qq){ System.***out***.println(**"father choose qq"**);}  **public void** hardChoice(\_360 arg) {System.***out***.println(**"father choose 360"**);}  }  **public static class** Son **extends** Father {  @Override  **public void** hardChoice(QQ qq) {System.***out***.println(**"son choose qq"**);}  @Override  **public void** hardChoice(\_360 arg) {System.***out***.println(**"son choose 360"**);}  }  **public static void** main(String[] args) {  Father father = **new** Father();  father.hardChoice(**new** QQ());  father = **new** Son();  father.hardChoice(**new** \_360());  } } |

## 1.8 动态类型语言与静态类型语言

* 动态类型语言的关键特征：是它的类型检查的主体过程是在运行期而不是编译期
* 静态类型语言的特征：在编译期就进行类型检查过程的语言称之为：静态类型语言
* 例如：invokevirual #4 //Method java/io/PrintStream.println:(Ljava/lang/String;)V这个符号引用包含了此方法定义在安哥具体类型之中、方法的名称以及参数顺序、参数类型和方法返回值等信息，通过这个符号引用，虚拟机可以翻译出这个方法直接引用。而在ECMAScript等动态类型语言中，变量obj本身是没有类型的，变量obj的值才具有类型，编译时最多只能确定方法名称、参数、fangfahi值这些信息，而不会去确定方法所在的具体类型（即方法接收者不固定），“变量无类型而变量值才有类型”这个特点也是动态类型语言的一个重要特征

### 1.8.1 MethodHandle

* 从本质上讲，Reflection和MethodHandle机制都是在模拟方法的调用，但Reflection是在模拟Java代码层次的方法调用，而MethodHandle是在模拟字节码层次的方法调用。在MethodHanles.lookup中三个方法findSpecial()、findStatic()、findVirtual()正是对应于invokespecial、invokestatic、invokevirtual&invokeinterface这几条字节码指令的执行权限校验行为，而这些底层细节在使用Reflection API时是不需要关心的
* Reflection中的java.lang.reflect.Method对象远比MethodHandle机制中的java.lang.invoke.MethHandle对象所包含的信息多。前者是方法在java一端的全面映像，包含了方法的签名、描述符以及方法的属性表中各种属性的java端表示方式，还包含执行权限等的运行期信息，Reflection是重量级的，而MethodHandle是轻量级
* 由于MethodHandle是对字节码的方法指令调用的模拟，所以理论上虚拟机在这方面做的各种优化（如方法的内联），在MethodHandle上也应当可以采用类似的思路去支持，而通过反射去调用方法则不行

|  |
| --- |
| **public class** MethodHandleTest {   **static class** Class1 {  **public** String name() {  **return** Class1.**class**.getName();  }  **public void** println(String name) {  System.***out***.println(name);  }  }   **static class** Class2 {  **public** String name() {  **return** Class2.**class**.getName();  }  }   **public static void** main(String[] args) **throws** Throwable {  Object clazz = **new** Class1();  System.***out***.println(*getNameHandle*(clazz).invoke());  *getPrintlnHandle*(clazz).invokeExact(**"clazz"**);  clazz = **new** Class2();  System.***out***.println(*getNameHandle*(clazz).invoke());  }   **public static** MethodHandle getNameHandle(Object obj) **throws** NoSuchMethodException, IllegalAccessException { *// MethodType methodType = MethodType.fromMethodDescriptorString("()Ljava/lang/String;",null);* MethodType methodType = MethodType.*methodType*(String.**class**);  **return** MethodHandles.*lookup*().findVirtual(obj.getClass(),**"name"**,methodType).bindTo(obj);  }   **public static** MethodHandle getPrintlnHandle(Object obj) **throws** NoSuchMethodException, IllegalAccessException, ClassNotFoundException {  MethodType methodType = MethodType.*methodType*(Void.***TYPE***,String.**class**);  System.***out***.println(methodType.toString());  **return** MethodHandles.*lookup*().findVirtual(obj.getClass(),**"println"**,methodType).bindTo(obj);  }  } |

### 1.8.2 invokedynamic

|  |
| --- |
| **public class** DynamicDemo {   **class** GrandFather {  **void** thinking() {  System.***out***.println(**"i am grandfather"**);  }  }   **class** Father **extends** GrandFather {  @Override  **void** thinking() {  System.***out***.println(**"i am father"**);  }  }   **class** Son **extends** Father {  @Override  **void** thinking() {  **try** {  MethodHandle handle = MethodHandles.*lookup*()  .findSpecial(GrandFather.**class**, **"thinking"**, MethodType.*methodType*(**void**.**class**),getClass());  handle.invoke(**this**);  } **catch** (NoSuchMethodException e) {  e.printStackTrace();  } **catch** (IllegalAccessException e) {  e.printStackTrace();  } **catch** (Throwable throwable) {  throwable.printStackTrace();  }  }  }   **public static void** main(String[] args) {  DynamicDemo.Son son = (**new** DynamicDemo()).**new** Son();  son.thinking();  }  } |

# JVM调优

## 查看所有命令

java -XX:+PrintFlagsFinal -version

## 打印GC：

-XX:+PrintGCDetails -XX:+PrintGCTimeStamps -Xloggc:d:/gc.log

## oom导出内存快照

-XX:+HeapDumpOnOutOfMemoryError -XX:HeapDumpPath=d:/gclogs/dump

## 常用命令

* jstat 实时查看jvm内存状况
* jps 查看运行的所有java进程
* jhat 分析dump文件
* jmap : jmap -dump:format=b,file=filepath pid：导出内存
* 可视化界面：jvisualvm.exe,jconsole.exe(远程连接)
* Jinfo 输出运行jvm进程相关信息
* Arthas:阿里监控工具

## 参看资料

[JVM性能调优监控工具jps、jstack、jmap、jhat、jstat、hprof使用详解 - 知乎 (zhihu.com)](https://zhuanlan.zhihu.com/p/294950372)

[retransform — Arthas 3.5.5 文档 (aliyun.com)](https://arthas.aliyun.com/doc/retransform.html)

## ParNews+CMS GC日志

-XX:+UseParNewGC -XX:+UseConcMarkSweepGC -XX:+PrintGCDetails

|  |
| --- |
| [GC (Allocation Failure) [ParNew: 5504K->640K(6144K), 0.0062942 secs] 5504K->1698K(19840K), 0.0063712 secs] [Times: user=0.00 sys=0.00, real=0.01 secs]  [GC (Allocation Failure) [ParNew: 6144K->475K(6144K), 0.0008627 secs] 7202K->2104K(19840K), 0.0008897 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (Allocation Failure) [ParNew: 5979K->640K(6144K), 0.0086585 secs] 7608K->4920K(19840K), 0.0087142 secs] [Times: user=0.13 sys=0.00, real=0.01 secs]  [GC (Allocation Failure) [ParNew: 6144K->640K(6144K), 0.0038154 secs] 10424K->8784K(19840K), 0.0038521 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (CMS Initial Mark) [1 CMS-initial-mark: 8144K(13696K)] 8850K(19840K), 0.0005628 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-mark-start]  [CMS-concurrent-mark: 0.004/0.004 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-preclean-start]  [CMS-concurrent-preclean: 0.000/0.000 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (CMS Final Remark) [YG occupancy: 705 K (6144 K)][Rescan (parallel) , 0.0006505 secs][weak refs processing, 0.0000184 secs][class unloading, 0.0004279 secs][scrub symbol table, 0.0003047 secs][scrub string table, 0.0000903 secs][1 CMS-remark: 8144K(13696K)] 8850K(19840K), 0.0015457 secs] [Times: user=0.13 sys=0.00, real=0.00 secs]  [CMS-concurrent-sweep-start]  [CMS-concurrent-sweep: 0.002/0.002 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-reset-start]  [CMS-concurrent-reset: 0.000/0.000 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (CMS Initial Mark) [1 CMS-initial-mark: 8042K(13696K)] 10237K(19840K), 0.0016240 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-mark-start]  [CMS-concurrent-mark: 0.010/0.010 secs] [Times: user=0.03 sys=0.00, real=0.01 secs]  [CMS-concurrent-preclean-start]  [CMS-concurrent-preclean: 0.001/0.001 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (CMS Final Remark) [YG occupancy: 2195 K (6144 K)][Rescan (parallel) , 0.0007633 secs][weak refs processing, 0.0000074 secs][class unloading, 0.0004250 secs][scrub symbol table, 0.0003664 secs][scrub string table, 0.0000986 secs][1 CMS-remark: 8042K(13696K)] 10237K(19840K), 0.0016993 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-sweep-start]  [CMS-concurrent-sweep: 0.002/0.002 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-reset-start]  [CMS-concurrent-reset: 0.000/0.000 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (CMS Initial Mark) [1 CMS-initial-mark: 8026K(13696K)] 10931K(19840K), 0.0015364 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-mark-start]  [CMS-concurrent-mark: 0.006/0.006 secs] [Times: user=0.03 sys=0.00, real=0.01 secs]  [CMS-concurrent-preclean-start]  [CMS-concurrent-preclean: 0.001/0.001 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-abortable-preclean-start]  CMS: abort preclean due to time [CMS-concurrent-abortable-preclean: 0.341/5.026 secs] [Times: user=0.27 sys=0.11, real=5.02 secs]  [GC (CMS Final Remark) [YG occupancy: 3906 K (6144 K)][Rescan (parallel) , 0.0006976 secs][weak refs processing, 0.0000073 secs][class unloading, 0.0003715 secs][scrub symbol table, 0.0003180 secs][scrub string table, 0.0000922 secs][1 CMS-remark: 8026K(13696K)] 11932K(19840K), 0.0015280 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-sweep-start]  [CMS-concurrent-sweep: 0.002/0.002 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-reset-start]  [CMS-concurrent-reset: 0.000/0.000 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (CMS Initial Mark) [1 CMS-initial-mark: 8026K(13696K)] 12326K(19840K), 0.0009344 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-mark-start]  [CMS-concurrent-mark: 0.007/0.007 secs] [Times: user=0.03 sys=0.00, real=0.01 secs]  [CMS-concurrent-preclean-start]  [CMS-concurrent-preclean: 0.001/0.001 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [CMS-concurrent-abortable-preclean-start]  Heap  par new generation total 6144K, used 4416K [0x00000000fec00000, 0x00000000ff2a0000, 0x00000000ff2a0000)  eden space 5504K, 68% used [0x00000000fec00000, 0x00000000fefb0270, 0x00000000ff160000)  from space 640K, 100% used [0x00000000ff160000, 0x00000000ff200000, 0x00000000ff200000)  to space 640K, 0% used [0x00000000ff200000, 0x00000000ff200000, 0x00000000ff2a0000)  concurrent mark-sweep generation total 13696K, used 8026K [0x00000000ff2a0000, 0x0000000100000000, 0x0000000100000000)  Metaspace used 5129K, capacity 5354K, committed 5504K, reserved 1056768K  class space used 562K, capacity 625K, committed 640K, reserved 1048576K |

## PS(ParallelGC+ParallelOldGC)

-XX:+ParallelGC -XX:+ParallelOldGC

|  |
| --- |
| [GC (Allocation Failure) [PSYoungGen: 5632K->504K(6144K)] 5632K->1616K(19968K), 0.0015200 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (Allocation Failure) [PSYoungGen: 6136K->504K(6144K)] 7248K->1864K(19968K), 0.0010576 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (Allocation Failure) [PSYoungGen: 6136K->508K(6144K)] 7496K->4557K(19968K), 0.0035257 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC (Allocation Failure) [PSYoungGen: 6140K->488K(6144K)] 10189K->8141K(19968K), 0.0051990 secs] [Times: user=0.00 sys=0.00, real=0.01 secs]  [GC (Allocation Failure) [PSYoungGen: 6120K->493K(6144K)] 13773K->11242K(19968K), 0.0042988 secs] [Times: user=0.00 sys=0.00, real=0.00 secs]  [Full GC (Ergonomics) [PSYoungGen: 493K->0K(6144K)] [ParOldGen: 10748K->10906K(13824K)] 11242K->10906K(19968K), [Metaspace: 5129K->5123K(1056768K)], 0.0591524 secs] [Times: user=0.41 sys=0.00, real=0.06 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->368K(6144K)] [ParOldGen: 10906K->13624K(13824K)] 16538K->13993K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0361865 secs] [Times: user=0.14 sys=0.00, real=0.04 secs]  [Full GC (Ergonomics) [PSYoungGen: 5631K->5631K(6144K)] [ParOldGen: 13477K->13475K(13824K)] 19109K->19107K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0214674 secs] [Times: user=0.13 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5631K->5631K(6144K)] [ParOldGen: 13477K->13475K(13824K)] 19109K->19107K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0208761 secs] [Times: user=0.25 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5631K->5631K(6144K)] [ParOldGen: 13477K->13475K(13824K)] 19109K->19107K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0211750 secs] [Times: user=0.13 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5631K->5631K(6144K)] [ParOldGen: 13477K->13475K(13824K)] 19109K->19107K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0213232 secs] [Times: user=0.20 sys=0.02, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13477K->13475K(13824K)] 19109K->19107K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0203762 secs] [Times: user=0.13 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13477K->13477K(13824K)] 19109K->19109K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0208900 secs] [Times: user=0.13 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13479K->13479K(13824K)] 19111K->19111K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0211988 secs] [Times: user=0.25 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13481K->13481K(13824K)] 19113K->19113K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0215757 secs] [Times: user=0.13 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13483K->13483K(13824K)] 19115K->19115K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0221002 secs] [Times: user=0.13 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13485K->13485K(13824K)] 19117K->19117K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0241532 secs] [Times: user=0.09 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13487K->13486K(13824K)] 19119K->19118K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0205796 secs] [Times: user=0.11 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13488K->13486K(13824K)] 19120K->19118K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0204373 secs] [Times: user=0.01 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13488K->13487K(13824K)] 19120K->19119K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0207041 secs] [Times: user=0.11 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13489K->13487K(13824K)] 19121K->19119K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0208447 secs] [Times: user=0.22 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13489K->13488K(13824K)] 19121K->19120K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0208835 secs] [Times: user=0.11 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13490K->13488K(13824K)] 19122K->19120K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0218812 secs] [Times: user=0.13 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13490K->13489K(13824K)] 19122K->19121K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0225827 secs] [Times: user=0.11 sys=0.00, real=0.02 secs]  [Full GC (Ergonomics) [PSYoungGen: 5632K->5632K(6144K)] [ParOldGen: 13491K->13489K(13824K)] 19123K->19121K(19968K), [Metaspace: 5123K->5123K(1056768K)], 0.0234526 secs] [Times: user=0.20 sys=0.00, real=0.02 secs]  Heap  PSYoungGen total 6144K, used 5632K [0x00000000ff980000, 0x0000000100000000, 0x0000000100000000)  eden space 5632K, 100% used [0x00000000ff980000,0x00000000fff00000,0x00000000fff00000)  from space 512K, 0% used [0x00000000fff00000,0x00000000fff00000,0x00000000fff80000)  to space 512K, 0% used [0x00000000fff80000,0x00000000fff80000,0x0000000100000000)  ParOldGen total 13824K, used 13613K [0x00000000fec00000, 0x00000000ff980000, 0x00000000ff980000)  object space 13824K, 98% used [0x00000000fec00000,0x00000000ff94b4f8,0x00000000ff980000)  Metaspace used 5131K, capacity 5354K, committed 5504K, reserved 1056768K  class space used 562K, capacity 625K, committed 640K, reserved 1048576K |

## G1

-XX:+UseG1GC

|  |
| --- |
| D:\dev-tools\Java\jdk1.8.0\_301\bin\java.exe -Xms20M -Xmx20M -XX:+UseG1GC -XX:+PrintGCDetails "-javaagent:D:\dev-tools\JetBrains\IntelliJ IDEA 2021.2\lib\idea\_rt.jar=64285:D:\dev-tools\JetBrains\IntelliJ IDEA 2021.2\bin" -Dfile.encoding=UTF-8 -classpath D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\charsets.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\deploy.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\access-bridge-64.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\cldrdata.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\dnsns.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\jaccess.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\jfxrt.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\localedata.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\nashorn.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\sunec.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\sunjce\_provider.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\sunmscapi.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\sunpkcs11.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\ext\zipfs.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\javaws.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\jce.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\jfr.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\jfxswt.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\jsse.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\management-agent.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\plugin.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\resources.jar;D:\dev-tools\Java\jdk1.8.0\_301\jre\lib\rt.jar;D:\work-spaces\JVM\target\classes;D:\mvn-repository\org\apache\commons\commons-lang3\3.12.0\commons-lang3-3.12.0.jar;D:\mvn-repository\commons-io\commons-io\2.11.0\commons-io-2.11.0.jar;D:\mvn-repository\cglib\cglib\3.3.0\cglib-3.3.0.jar;D:\mvn-repository\org\ow2\asm\asm\7.1\asm-7.1.jar;D:\mvn-repository\org\javassist\javassist\3.26.0-GA\javassist-3.26.0-GA.jar;D:\mvn-repository\org\projectlombok\lombok\1.18.22\lombok-1.18.22.jar com.github.liuxg.jvm.oom.T15\_FullGC\_Demo1  [GC pause (G1 Evacuation Pause) (young), 0.0013581 secs]  [Parallel Time: 1.0 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 127.5, Avg: 127.5, Max: 127.6, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.3, Avg: 0.4, Max: 0.4, Diff: 0.1, Sum: 2.8]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.0, Max: 0, Diff: 0, Sum: 0]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Object Copy (ms): Min: 0.5, Avg: 0.5, Max: 0.6, Diff: 0.1, Sum: 4.2]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.3]  [Termination Attempts: Min: 1, Avg: 6.0, Max: 11, Diff: 10, Sum: 48]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [GC Worker Total (ms): Min: 0.9, Avg: 1.0, Max: 1.0, Diff: 0.0, Sum: 7.7]  [GC Worker End (ms): Min: 128.5, Avg: 128.5, Max: 128.5, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 12.0M(12.0M)->0.0B(10.0M) Survivors: 0.0B->2048.0K Heap: 12.0M(20.0M)->1088.0K(20.0M)]  [Times: user=0.11 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young), 0.0052004 secs]  [Parallel Time: 4.1 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 19185.9, Avg: 19186.0, Max: 19186.1, Diff: 0.2]  [Ext Root Scanning (ms): Min: 0.5, Avg: 0.6, Max: 0.7, Diff: 0.2, Sum: 4.5]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.0, Max: 0, Diff: 0, Sum: 0]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.3, Diff: 0.3, Sum: 0.4]  [Object Copy (ms): Min: 2.9, Avg: 3.2, Max: 3.2, Diff: 0.4, Sum: 25.6]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination Attempts: Min: 1, Avg: 1.9, Max: 2, Diff: 1, Sum: 15]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 3.7, Avg: 3.8, Max: 3.9, Diff: 0.2, Sum: 30.6]  [GC Worker End (ms): Min: 19189.8, Avg: 19189.8, Max: 19189.8, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 1.0 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.8 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 10.0M(10.0M)->0.0B(10.0M) Survivors: 2048.0K->2048.0K Heap: 11.1M(20.0M)->5110.0K(20.0M)]  [Times: user=0.00 sys=0.02, real=0.01 secs]  [GC pause (G1 Evacuation Pause) (young) (to-space exhausted), 0.0325071 secs]  [Parallel Time: 28.8 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 51742.8, Avg: 51742.8, Max: 51742.9, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.2, Avg: 0.3, Max: 0.3, Diff: 0.1, Sum: 2.1]  [Update RS (ms): Min: 0.0, Avg: 0.5, Max: 0.7, Diff: 0.7, Sum: 4.4]  [Processed Buffers: Min: 0, Avg: 8.6, Max: 12, Diff: 12, Sum: 69]  [Scan RS (ms): Min: 0.0, Avg: 0.3, Max: 0.3, Diff: 0.3, Sum: 2.1]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.2, Diff: 0.2, Sum: 0.2]  [Object Copy (ms): Min: 27.3, Avg: 27.6, Max: 28.5, Diff: 1.1, Sum: 221.0]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 28.7, Avg: 28.7, Max: 28.8, Diff: 0.1, Sum: 230.0]  [GC Worker End (ms): Min: 51771.6, Avg: 51771.6, Max: 51771.6, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.0 ms]  [Other: 3.6 ms]  [Evacuation Failure: 3.3 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 10.0M(10.0M)->0.0B(1024.0K) Survivors: 2048.0K->2048.0K Heap: 15.0M(20.0M)->17.0M(20.0M)]  [Times: user=0.03 sys=0.03, real=0.03 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark) (to-space exhausted), 0.0169325 secs]  [Parallel Time: 14.0 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 54818.5, Avg: 54818.6, Max: 54818.7, Diff: 0.2]  [Ext Root Scanning (ms): Min: 0.4, Avg: 0.6, Max: 0.9, Diff: 0.4, Sum: 4.5]  [Update RS (ms): Min: 1.4, Avg: 1.8, Max: 1.9, Diff: 0.5, Sum: 14.4]  [Processed Buffers: Min: 6, Avg: 7.6, Max: 10, Diff: 4, Sum: 61]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 11.2, Avg: 11.3, Max: 11.6, Diff: 0.4, Sum: 90.6]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.2, Diff: 0.2, Sum: 1.0]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 13.7, Avg: 13.8, Max: 14.0, Diff: 0.2, Sum: 110.6]  [GC Worker End (ms): Min: 54832.4, Avg: 54832.5, Max: 54832.5, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 2.8 ms]  [Evacuation Failure: 2.4 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.2 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 2048.0K->1024.0K Heap: 18.0M(20.0M)->20.0M(20.0M)]  [Times: user=0.00 sys=0.02, real=0.02 secs]  [GC concurrent-root-region-scan-start]  [GC pause (G1 Evacuation Pause) (young)[GC concurrent-root-region-scan-end, 0.0002625 secs]  [GC concurrent-mark-start]  (to-space exhausted), 0.0041472 secs]  [Root Region Scan Waiting: 0.1 ms]  [Parallel Time: 3.5 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 54835.6, Avg: 54835.7, Max: 54835.7, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.3]  [Update RS (ms): Min: 0.5, Avg: 0.6, Max: 0.6, Diff: 0.2, Sum: 4.6]  [Processed Buffers: Min: 2, Avg: 4.4, Max: 7, Diff: 5, Sum: 35]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 2.6, Avg: 2.7, Max: 2.8, Diff: 0.2, Sum: 21.4]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.3]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 3.4, Avg: 3.5, Max: 3.5, Diff: 0.0, Sum: 27.6]  [GC Worker End (ms): Min: 54839.1, Avg: 54839.1, Max: 54839.1, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.5 ms]  [Evacuation Failure: 0.3 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->0.0B Heap: 20.0M(20.0M)->20.0M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [Full GC (Allocation Failure) 20M->11M(20M), 0.0211155 secs]  [Eden: 0.0B(1024.0K)->0.0B(4096.0K) Survivors: 0.0B->0.0B Heap: 20.0M(20.0M)->11.3M(20.0M)], [Metaspace: 5146K->5139K(1056768K)]  [Times: user=0.01 sys=0.00, real=0.02 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young), 0.0060233 secs]  [Parallel Time: 4.2 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 67782.1, Avg: 67782.1, Max: 67782.2, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.2, Avg: 0.2, Max: 0.3, Diff: 0.1, Sum: 1.6]  [Update RS (ms): Min: 0.0, Avg: 0.9, Max: 1.1, Diff: 1.1, Sum: 7.3]  [Processed Buffers: Min: 0, Avg: 8.3, Max: 13, Diff: 13, Sum: 66]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 2.1, Avg: 2.6, Max: 2.7, Diff: 0.6, Sum: 20.8]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [Termination Attempts: Min: 28, Avg: 34.4, Max: 39, Diff: 11, Sum: 275]  [GC Worker Other (ms): Min: 0.0, Avg: 0.2, Max: 1.7, Diff: 1.6, Sum: 1.8]  [GC Worker Total (ms): Min: 3.9, Avg: 4.0, Max: 4.0, Diff: 0.1, Sum: 31.8]  [GC Worker End (ms): Min: 67786.1, Avg: 67786.1, Max: 67786.2, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.6 ms]  [Other: 1.3 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.4 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.7 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 4096.0K(4096.0K)->0.0B(1024.0K) Survivors: 0.0B->1024.0K Heap: 15.3M(20.0M)->14.2M(20.0M)]  [Times: user=0.11 sys=0.00, real=0.01 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark), 0.0056157 secs]  [Parallel Time: 4.5 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 71099.9, Avg: 71100.1, Max: 71100.4, Diff: 0.5]  [Ext Root Scanning (ms): Min: 0.0, Avg: 0.4, Max: 0.7, Diff: 0.7, Sum: 3.1]  [Update RS (ms): Min: 2.0, Avg: 2.3, Max: 2.5, Diff: 0.5, Sum: 18.8]  [Processed Buffers: Min: 5, Avg: 8.5, Max: 11, Diff: 6, Sum: 68]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.3]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 1.3, Avg: 1.4, Max: 1.5, Diff: 0.2, Sum: 11.2]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination Attempts: Min: 23, Avg: 26.4, Max: 31, Diff: 8, Sum: 211]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 3.9, Avg: 4.2, Max: 4.4, Diff: 0.5, Sum: 33.6]  [GC Worker End (ms): Min: 71104.3, Avg: 71104.3, Max: 71104.3, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 1.0 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.7 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->1024.0K Heap: 15.2M(20.0M)->15.9M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.01 secs]  [GC concurrent-root-region-scan-start]  [GC concurrent-root-region-scan-end, 0.0006203 secs]  [GC concurrent-mark-start]  [GC concurrent-mark-end, 0.0137747 secs]  [GC remark [Finalize Marking, 0.0003019 secs] [GC ref-proc, 0.0000741 secs] [Unloading, 0.0009176 secs], 0.0020229 secs]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC cleanup 16M->16M(20M), 0.0005579 secs]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young), 0.0063372 secs]  [Parallel Time: 4.9 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 74338.1, Avg: 74338.1, Max: 74338.2, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.4, Avg: 0.5, Max: 0.5, Diff: 0.1, Sum: 3.8]  [Update RS (ms): Min: 1.4, Avg: 2.7, Max: 3.1, Diff: 1.7, Sum: 22.0]  [Processed Buffers: Min: 7, Avg: 8.4, Max: 11, Diff: 4, Sum: 67]  [Scan RS (ms): Min: 0.0, Avg: 0.1, Max: 0.2, Diff: 0.2, Sum: 1.1]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 1.1, Avg: 1.3, Max: 2.8, Diff: 1.7, Sum: 10.6]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 4.6, Avg: 4.7, Max: 4.8, Diff: 0.1, Sum: 37.6]  [GC Worker End (ms): Min: 74342.8, Avg: 74342.9, Max: 74342.9, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.5 ms]  [Other: 0.9 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.4 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.3 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->1024.0K Heap: 16.9M(20.0M)->16.9M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.01 secs]  [GC pause (G1 Evacuation Pause) (mixed) (to-space exhausted), 0.0098426 secs]  [Parallel Time: 8.8 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 77551.3, Avg: 77551.3, Max: 77551.4, Diff: 0.2]  [Ext Root Scanning (ms): Min: 0.4, Avg: 0.5, Max: 0.6, Diff: 0.1, Sum: 3.8]  [Update RS (ms): Min: 2.9, Avg: 3.1, Max: 3.2, Diff: 0.3, Sum: 25.0]  [Processed Buffers: Min: 8, Avg: 8.9, Max: 10, Diff: 2, Sum: 71]  [Scan RS (ms): Min: 0.1, Avg: 0.2, Max: 0.3, Diff: 0.2, Sum: 1.4]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 4.7, Avg: 4.8, Max: 4.9, Diff: 0.2, Sum: 38.5]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.1, Diff: 0.1, Sum: 0.5]  [Termination Attempts: Min: 1, Avg: 1.3, Max: 2, Diff: 1, Sum: 10]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 8.6, Avg: 8.7, Max: 8.8, Diff: 0.2, Sum: 69.4]  [GC Worker End (ms): Min: 77560.0, Avg: 77560.0, Max: 77560.0, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.0 ms]  [Other: 1.0 ms]  [Evacuation Failure: 0.7 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->1024.0K Heap: 17.9M(20.0M)->18.2M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.01 secs]  [GC pause (G1 Evacuation Pause) (young) (to-space exhausted), 0.0122412 secs]  [Parallel Time: 10.9 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 80710.5, Avg: 80710.6, Max: 80710.7, Diff: 0.2]  [Ext Root Scanning (ms): Min: 0.3, Avg: 0.4, Max: 0.5, Diff: 0.2, Sum: 3.4]  [Update RS (ms): Min: 0.9, Avg: 1.2, Max: 1.5, Diff: 0.6, Sum: 9.4]  [Processed Buffers: Min: 5, Avg: 8.8, Max: 12, Diff: 7, Sum: 70]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.3]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 8.9, Avg: 9.0, Max: 9.2, Diff: 0.3, Sum: 72.2]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.1, Diff: 0.1, Sum: 0.7]  [Termination Attempts: Min: 1, Avg: 1.3, Max: 2, Diff: 1, Sum: 10]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 10.7, Avg: 10.8, Max: 10.9, Diff: 0.2, Sum: 86.1]  [GC Worker End (ms): Min: 80721.3, Avg: 80721.3, Max: 80721.3, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.0 ms]  [Other: 1.3 ms]  [Evacuation Failure: 1.0 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->0.0B Heap: 19.2M(20.0M)->19.2M(20.0M)]  [Times: user=0.03 sys=0.00, real=0.01 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark), 0.0008846 secs]  [Parallel Time: 0.7 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 80722.8, Avg: 80722.9, Max: 80723.0, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.3]  [Update RS (ms): Min: 0.3, Avg: 0.3, Max: 0.4, Diff: 0.1, Sum: 2.7]  [Processed Buffers: Min: 2, Avg: 3.1, Max: 6, Diff: 4, Sum: 25]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.1, Diff: 0.1, Sum: 0.6]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.5, Avg: 0.6, Max: 0.6, Diff: 0.1, Sum: 4.6]  [GC Worker End (ms): Min: 80723.4, Avg: 80723.4, Max: 80723.5, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.0 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.2M(20.0M)->19.2M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC concurrent-root-region-scan-start]  [GC concurrent-root-region-scan-end, 0.0000056 secs]  [GC concurrent-mark-start]  [Full GC (Allocation Failure) 19M->16M(20M), 0.0243685 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.2M(20.0M)->16.1M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.02 sys=0.00, real=0.03 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young), 0.0049587 secs]  [Parallel Time: 3.5 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 84111.2, Avg: 84111.3, Max: 84111.3, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.5]  [Update RS (ms): Min: 1.9, Avg: 2.1, Max: 2.2, Diff: 0.3, Sum: 16.6]  [Processed Buffers: Min: 7, Avg: 9.1, Max: 11, Diff: 4, Sum: 73]  [Scan RS (ms): Min: 0.0, Avg: 0.1, Max: 0.2, Diff: 0.2, Sum: 0.5]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.8, Avg: 0.9, Max: 0.9, Diff: 0.2, Sum: 6.8]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination Attempts: Min: 1, Avg: 5.6, Max: 9, Diff: 8, Sum: 45]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [GC Worker Total (ms): Min: 3.2, Avg: 3.2, Max: 3.3, Diff: 0.1, Sum: 25.7]  [GC Worker End (ms): Min: 84114.5, Avg: 84114.5, Max: 84114.5, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.3 ms]  [Other: 1.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.2 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.8 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->1024.0K Heap: 17.1M(20.0M)->16.8M(20.0M)]  [Times: user=0.13 sys=0.00, real=0.01 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark) (to-space exhausted), 0.0079323 secs]  [Parallel Time: 6.6 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 85959.0, Avg: 85959.1, Max: 85959.3, Diff: 0.3]  [Ext Root Scanning (ms): Min: 0.0, Avg: 0.2, Max: 0.5, Diff: 0.5, Sum: 1.7]  [Update RS (ms): Min: 0.8, Avg: 1.1, Max: 1.1, Diff: 0.3, Sum: 8.7]  [Processed Buffers: Min: 8, Avg: 9.8, Max: 13, Diff: 5, Sum: 78]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.3]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 5.0, Avg: 5.1, Max: 5.1, Diff: 0.1, Sum: 40.5]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination Attempts: Min: 69, Avg: 79.9, Max: 92, Diff: 23, Sum: 639]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 6.3, Avg: 6.4, Max: 6.6, Diff: 0.3, Sum: 51.4]  [GC Worker End (ms): Min: 85965.6, Avg: 85965.6, Max: 85965.6, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 1.3 ms]  [Evacuation Failure: 0.9 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->1024.0K Heap: 17.8M(20.0M)->18.8M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.01 secs]  [GC concurrent-root-region-scan-start]  [GC pause (G1 Evacuation Pause) (young)[GC concurrent-root-region-scan-end, 0.0043294 secs]  [GC concurrent-mark-start]  (to-space exhausted), 0.0078051 secs]  [Root Region Scan Waiting: 4.1 ms]  [Parallel Time: 2.7 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 85971.4, Avg: 85971.5, Max: 85971.6, Diff: 0.2]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.1, Max: 0.3, Diff: 0.2, Sum: 1.1]  [Update RS (ms): Min: 0.2, Avg: 0.3, Max: 0.3, Diff: 0.2, Sum: 2.0]  [Processed Buffers: Min: 2, Avg: 3.3, Max: 4, Diff: 2, Sum: 26]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 2.0, Avg: 2.1, Max: 2.3, Diff: 0.3, Sum: 17.0]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.2, Diff: 0.2, Sum: 0.7]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 2.5, Avg: 2.6, Max: 2.7, Diff: 0.2, Sum: 20.8]  [GC Worker End (ms): Min: 85974.1, Avg: 85974.1, Max: 85974.1, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.0 ms]  [Other: 0.9 ms]  [Evacuation Failure: 0.7 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->0.0B Heap: 18.8M(20.0M)->18.8M(20.0M)]  [Times: user=0.02 sys=0.00, real=0.01 secs]  [Full GC (Allocation Failure) 18M->17M(20M), 0.0230589 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 18.8M(20.0M)->17.2M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.01 sys=0.00, real=0.02 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young), 0.0025391 secs]  [Parallel Time: 1.9 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 89360.5, Avg: 89360.6, Max: 89360.6, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.4]  [Update RS (ms): Min: 0.9, Avg: 1.1, Max: 1.2, Diff: 0.4, Sum: 9.0]  [Processed Buffers: Min: 8, Avg: 9.5, Max: 12, Diff: 4, Sum: 76]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.2]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.3, Avg: 0.4, Max: 0.6, Diff: 0.3, Sum: 2.9]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination Attempts: Min: 1, Avg: 5.0, Max: 9, Diff: 8, Sum: 40]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 1.7, Avg: 1.7, Max: 1.8, Diff: 0.1, Sum: 13.7]  [GC Worker End (ms): Min: 89362.3, Avg: 89362.3, Max: 89362.3, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.5 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.3 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->1024.0K Heap: 18.2M(20.0M)->17.9M(20.0M)]  [Times: user=0.13 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark) (to-space exhausted), 0.0122488 secs]  [Parallel Time: 10.7 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 92851.0, Avg: 92851.0, Max: 92851.0, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.8]  [Update RS (ms): Min: 1.1, Avg: 1.2, Max: 1.3, Diff: 0.2, Sum: 9.7]  [Processed Buffers: Min: 4, Avg: 9.3, Max: 13, Diff: 9, Sum: 74]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.4]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 8.9, Avg: 9.0, Max: 9.2, Diff: 0.4, Sum: 71.8]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.2, Diff: 0.2, Sum: 0.9]  [Termination Attempts: Min: 1, Avg: 1.1, Max: 2, Diff: 1, Sum: 9]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 10.6, Avg: 10.6, Max: 10.6, Diff: 0.0, Sum: 84.7]  [GC Worker End (ms): Min: 92861.6, Avg: 92861.6, Max: 92861.6, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 1.5 ms]  [Evacuation Failure: 1.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 1024.0K->0.0B Heap: 18.9M(20.0M)->18.9M(20.0M)]  [Times: user=0.03 sys=0.01, real=0.01 secs]  [GC concurrent-root-region-scan-start]  [GC concurrent-root-region-scan-end, 0.0000085 secs]  [GC concurrent-mark-start]  [GC pause (G1 Evacuation Pause) (young), 0.0009876 secs]  [Parallel Time: 0.7 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 92863.4, Avg: 92863.5, Max: 92863.5, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.1, Max: 0.2, Diff: 0.1, Sum: 1.2]  [Update RS (ms): Min: 0.3, Avg: 0.4, Max: 0.5, Diff: 0.1, Sum: 3.1]  [Processed Buffers: Min: 2, Avg: 3.1, Max: 4, Diff: 2, Sum: 25]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.2, Diff: 0.2, Sum: 0.7]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.6, Avg: 0.6, Max: 0.7, Diff: 0.1, Sum: 5.1]  [GC Worker End (ms): Min: 92864.1, Avg: 92864.1, Max: 92864.1, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 18.9M(20.0M)->18.9M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [Full GC (Allocation Failure) 18M->18M(20M), 0.0263279 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 18.9M(20.0M)->18.5M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.02 sys=0.00, real=0.03 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young) (to-space exhausted), 0.0086998 secs]  [Parallel Time: 7.4 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96124.5, Avg: 96124.6, Max: 96124.7, Diff: 0.2]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.3, Diff: 0.2, Sum: 1.5]  [Update RS (ms): Min: 1.3, Avg: 1.6, Max: 1.7, Diff: 0.4, Sum: 12.6]  [Processed Buffers: Min: 1, Avg: 9.9, Max: 14, Diff: 13, Sum: 79]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.2]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 5.2, Avg: 5.3, Max: 5.6, Diff: 0.4, Sum: 42.6]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.2, Diff: 0.2, Sum: 0.8]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 7.2, Avg: 7.2, Max: 7.3, Diff: 0.2, Sum: 57.8]  [GC Worker End (ms): Min: 96131.9, Avg: 96131.9, Max: 96131.9, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 1.3 ms]  [Evacuation Failure: 1.0 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 1024.0K(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.5M(20.0M)->19.5M(20.0M)]  [Times: user=0.01 sys=0.00, real=0.01 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark), 0.0007446 secs]  [Parallel Time: 0.5 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96133.3, Avg: 96133.3, Max: 96133.4, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.3]  [Update RS (ms): Min: 0.2, Avg: 0.2, Max: 0.3, Diff: 0.1, Sum: 1.8]  [Processed Buffers: Min: 1, Avg: 2.9, Max: 4, Diff: 3, Sum: 23]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.3]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.4, Avg: 0.4, Max: 0.5, Diff: 0.1, Sum: 3.6]  [GC Worker End (ms): Min: 96133.8, Avg: 96133.8, Max: 96133.8, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.0 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.5M(20.0M)->19.5M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC concurrent-root-region-scan-start]  [GC concurrent-root-region-scan-end, 0.0000057 secs]  [GC concurrent-mark-start]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0351299 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.5M(20.0M)->19.1M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0320397 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young), 0.0006424 secs]  [Parallel Time: 0.4 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96201.6, Avg: 96201.7, Max: 96201.7, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.7]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.3, Max: 1, Diff: 1, Sum: 2]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.3, Max: 0.3, Diff: 0.0, Sum: 2.0]  [GC Worker End (ms): Min: 96201.9, Avg: 96201.9, Max: 96201.9, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young), 0.0004241 secs]  [Parallel Time: 0.2 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96202.4, Avg: 96202.5, Max: 96202.5, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.1, Max: 0.2, Diff: 0.1, Sum: 1.2]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.1, Max: 1, Diff: 1, Sum: 1]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.4]  [GC Worker End (ms): Min: 96202.6, Avg: 96202.6, Max: 96202.6, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.1 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0322635 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0291211 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [GC pause (G1 Evacuation Pause) (young), 0.0013100 secs]  [Parallel Time: 0.5 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96265.0, Avg: 96265.0, Max: 96265.0, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.5]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Processed Buffers: Min: 0, Avg: 0.3, Max: 1, Diff: 1, Sum: 2]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.3, Diff: 0.1, Sum: 1.9]  [GC Worker End (ms): Min: 96265.2, Avg: 96265.2, Max: 96265.2, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.2 ms]  [Other: 0.6 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.3 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.3 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark), 0.0004501 secs]  [Parallel Time: 0.3 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96266.4, Avg: 96266.4, Max: 96266.4, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.4]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.4, Max: 2, Diff: 2, Sum: 3]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.7]  [GC Worker End (ms): Min: 96266.6, Avg: 96266.6, Max: 96266.6, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.1 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC concurrent-root-region-scan-start]  [GC concurrent-root-region-scan-end, 0.0000044 secs]  [GC concurrent-mark-start]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0297979 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0277417 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5139K->5139K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young), 0.0008069 secs]  [Parallel Time: 0.4 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96324.9, Avg: 96324.9, Max: 96324.9, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.3]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Processed Buffers: Min: 0, Avg: 0.4, Max: 1, Diff: 1, Sum: 3]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.6]  [GC Worker End (ms): Min: 96325.1, Avg: 96325.1, Max: 96325.1, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.2 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young), 0.0005650 secs]  [Parallel Time: 0.3 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96325.9, Avg: 96326.0, Max: 96326.0, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.3]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.1, Max: 1, Diff: 1, Sum: 1]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.3]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.6]  [GC Worker End (ms): Min: 96326.2, Avg: 96326.2, Max: 96326.2, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.2 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark), 0.0004533 secs]  [Parallel Time: 0.3 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96326.7, Avg: 96326.7, Max: 96326.7, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.3]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.3, Max: 1, Diff: 1, Sum: 2]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.2]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.6]  [GC Worker End (ms): Min: 96326.9, Avg: 96326.9, Max: 96326.9, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.1 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC concurrent-root-region-scan-start]  [GC concurrent-root-region-scan-end, 0.0000041 secs]  [GC concurrent-mark-start]  [GC pause (G1 Evacuation Pause) (young), 0.0004942 secs]  [Parallel Time: 0.3 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96327.4, Avg: 96327.4, Max: 96327.4, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.1, Max: 0.1, Diff: 0.0, Sum: 1.0]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.2]  [Processed Buffers: Min: 0, Avg: 0.6, Max: 1, Diff: 1, Sum: 5]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.1, Max: 0.1, Diff: 0.1, Sum: 0.8]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.3, Max: 0.3, Diff: 0.0, Sum: 2.1]  [GC Worker End (ms): Min: 96327.6, Avg: 96327.6, Max: 96327.6, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.1 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.0 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [Full GC (Allocation Failure)  Exception: java.lang.OutOfMemoryError thrown from the UncaughtExceptionHandler in thread "main"  19M->19M(20M), 0.0311597 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5140K->5140K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0262184 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5140K->5140K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young), 0.0010388 secs]  [Parallel Time: 0.7 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96386.0, Avg: 96386.1, Max: 96386.2, Diff: 0.2]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.1, Max: 0.3, Diff: 0.2, Sum: 1.2]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Processed Buffers: Min: 0, Avg: 0.1, Max: 1, Diff: 1, Sum: 1]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.3]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.1, Avg: 0.2, Max: 0.3, Diff: 0.2, Sum: 1.6]  [GC Worker End (ms): Min: 96386.3, Avg: 96386.3, Max: 96386.3, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.3 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark), 0.0005272 secs]  [Parallel Time: 0.3 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96387.2, Avg: 96387.2, Max: 96387.2, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.4]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.1]  [Processed Buffers: Min: 0, Avg: 0.3, Max: 1, Diff: 1, Sum: 2]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.1]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.1, Diff: 0.1, Sum: 0.4]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.9]  [GC Worker End (ms): Min: 96387.5, Avg: 96387.5, Max: 96387.5, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.1 ms]  [Other: 0.2 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.1 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.1 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC concurrent-root-region-scan-start]  [GC concurrent-root-region-scan-end, 0.0000049 secs]  [GC concurrent-mark-start]  [GC pause (G1 Evacuation Pause) (young), 0.0010777 secs]  [Parallel Time: 0.2 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96388.0, Avg: 96388.0, Max: 96388.0, Diff: 0.0]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.1, Max: 0.1, Diff: 0.0, Sum: 1.1]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.1]  [Processed Buffers: Min: 0, Avg: 0.4, Max: 1, Diff: 1, Sum: 3]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.3]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.2, Avg: 0.2, Max: 0.2, Diff: 0.0, Sum: 1.5]  [GC Worker End (ms): Min: 96388.2, Avg: 96388.2, Max: 96388.2, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.3 ms]  [Other: 0.6 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.3 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.3 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0266579 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5140K->5140K(1056768K)]  [Times: user=0.03 sys=0.00, real=0.03 secs]  [Full GC (Allocation Failure) 19M->19M(20M), 0.0273705 secs]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)], [Metaspace: 5140K->5140K(1056768K)]  [Times: user=0.14 sys=0.00, real=0.03 secs]  [GC concurrent-mark-abort]  [GC pause (G1 Evacuation Pause) (young), 0.0016480 secs]  [Parallel Time: 0.7 ms, GC Workers: 8]  [GC Worker Start (ms): Min: 96443.8, Avg: 96443.8, Max: 96443.8, Diff: 0.1]  [Ext Root Scanning (ms): Min: 0.1, Avg: 0.2, Max: 0.2, Diff: 0.1, Sum: 1.3]  [Update RS (ms): Min: 0.0, Avg: 0.0, Max: 0.3, Diff: 0.3, Sum: 0.3]  [Processed Buffers: Min: 0, Avg: 0.3, Max: 1, Diff: 1, Sum: 2]  [Scan RS (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Code Root Scanning (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Object Copy (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [Termination (ms): Min: 0.0, Avg: 0.2, Max: 0.3, Diff: 0.3, Sum: 1.9]  [Termination Attempts: Min: 1, Avg: 1.0, Max: 1, Diff: 0, Sum: 8]  [GC Worker Other (ms): Min: 0.0, Avg: 0.0, Max: 0.0, Diff: 0.0, Sum: 0.0]  [GC Worker Total (ms): Min: 0.4, Avg: 0.5, Max: 0.5, Diff: 0.1, Sum: 3.6]  [GC Worker End (ms): Min: 96444.3, Avg: 96444.3, Max: 96444.3, Diff: 0.0]  [Code Root Fixup: 0.0 ms]  [Code Root Purge: 0.0 ms]  [Clear CT: 0.4 ms]  [Other: 0.6 ms]  [Choose CSet: 0.0 ms]  [Ref Proc: 0.3 ms]  [Ref Enq: 0.0 ms]  [Redirty Cards: 0.3 ms]  [Humongous Register: 0.0 ms]  [Humongous Reclaim: 0.0 ms]  [Free CSet: 0.0 ms]  [Eden: 0.0B(1024.0K)->0.0B(1024.0K) Survivors: 0.0B->0.0B Heap: 19.1M(20.0M)->19.1M(20.0M)]  [Times: user=0.00 sys=0.00, real=0.00 secs]  [GC pause (G1 Evacuation Pause) (young) (initial-mark)  Process finished with exit code 1 |