**JVM性能调优探索**

## 首先，让我们了解一下jvm的基本属性。

### 栈

栈中存放的数据对象在程序执行完毕后会立即释放。线程私有。

栈分为两种：

本地虚拟机栈，为执行自定义java代码的方法而服务的栈；

本地方法栈，为原系统的运行而提供的栈。

栈中存放的数据是一个完整的对象。基本数据类型都存放在栈中，如：int i=6；而需要实例化出来的数据类型是以（对象-堆中地址）形式存放在栈中。例如：

一般通过实例化出来的对象:Object obj1=new String("str");Object obj2=new String("str");

这时obj的值是一个堆中的地址，使用obj1.equals(obj2)进行字符串比较.

而类似于基本数据类型直接赋予其值的也是直接以完整属性存放在栈中：

Object str1="str";Object str2="str";这时的字符串比较直接通过str1==str2完成.

特殊的：StringBuffer和StringBuilder在被实例化后是直接把值赋予对象，字符串比较也是用“==”进行。

### 堆

堆中只存放对象，每个对象都有各自对应的class、地址、类型、值等信息。堆中的对象被所有线程共享。对象一旦没有被引用，则会被及时清除。

### 方法区

方法区内存放的是整个程序中永远唯一的元素。如：class、static变量、final常量等。区中的对象被所有线程共享。

### 程序计数器

每个线程对应一个计数器。例如：行号指示器，程序执行所处代码行号的标志。

## 然后了解调优背景

jvm性能优化的关键在于找到当前服务器系统的性能瓶颈。性能优化分为系统层次、代码层次、算法层次... ...

jvm调优涉及诸多方面，作为初学者首先从1.内存分配；2.类加载机制；3.垃圾回收GC；入手。

### 类加载

JVM的类加载过程粗略的可以分成3步

    1、加载：把class文件加载到方法区，转成JVM可用的二进制数据,在堆中产生一个反射对象Class

    2、链接：这一阶段是把字符连接，转换成真正的地址指向

    3、初始化：加载的时候都是默认值，就算是类中写的是“honeycat"，分配内存的时候也是null。

初始化才会改成name="honeycat"。当然这里指向堆中的地址。

这时类加载才算完成，即不会抛出ClassNotFoundException。

类加载器分为启动加载器（C++写的）、扩展加载器、应用程序加载器、自定义加载器，四种.

加载器的编写是用了委托机制.低层次的加载器接到一个类后不会立刻加载，而是上传到上层加载，如果没有上层或启动加载器的话再传回来，这种安全机制起到防止自定义java文件覆盖原系统类的作用。

### 垃圾回收

垃圾回收涉及“垃圾的定义”和“回收的方法”

垃圾的定义

对于对象是否是垃圾的判定一般有两种方法：

1.对象引用计数法：给每一个对象添加一个引用计数器。如果被引用则++，如果断开则--。为0表示没有被引用，可以回收。

2.可达性算法：从根节点GC\_root开始，向下搜索，如果有被引用则添加到该树上，在该树上的对象表示正在被引用。（根节点可能是：1.类对象main开始；2.所有程序入口；）

两个方法的区别：

前者简单高效；但是没被引用的对象不能立刻被察觉，时效性不好；无法处理循环引用情况。

后者复杂，但是回收的更彻底。

回收的方法

标记-清除：在进行垃圾判定时，堆中的对象只要是null的就清除。容易产生锁片，一般不使用

    复制：分成两部分，把不为空或null的对象复制到另一部分中，前一部分的对象全部清除。占用两倍内存；使用率下降，来回复制很麻烦。

    标记-整理：把不是null的集中到一起顺序排放，null的清空

按分区对待分

增量收集：实时垃圾回收算法

    分代收集：分成持久代、老年代、新生代，对不同生命周期的对象使用不同的回收算法，根据对象的活跃性进行分配、清空。

按系统线程分

串行收集：使用单线程处理所有垃圾回收，实现容易效率高；局限：适合单处理器的机器。

并行收集：多线程处理垃圾回收，速度快，效率高，适合多处理器机器。

并发收集：串行收集和并行收集在进行垃圾回收时需要暂停运行环境，暂停时间会因为对越大而越长.

## 最后准备调优实践

监控和测试工具：LoadRunner压测;JConsole、JMap、JStack、JStat等进程监控，定位分析;JStack、JStat、GC日志报告分析;TProfiler性能代码精准定位。

参考：

jvm性能调优之五[http://www.importnew.com/19264.html](http://www.importnew.com/22434.html)

jvm性能调优实战之系统瓶颈寻找过程<http://www.importnew.com/22434.html>

TProfiler性能代码精准定位[http://www.importnew.com/22438.html](http://www.importnew.com/22434.html)