链滴

对高性能 JAVA 代码之内存管理

作者：beejson

原文链接：https://ld246.com/article/1512108559729

来源网站：链滴

很多人在你写的代码，GC根本就回收不了，直接系统挂掉。GC是一段程序，不是智能，他只回他认为的垃圾，而不是回收你认为的垃圾。

GC垃圾回收：
Grabage Collection相信学过JAVA的人都知道这个是什么意思。但是他是如何工作的呢？
首先，JVM在管理内存的时候对于变量的管理总是分新对象和老对象。新对象也就是开发者new出来对象，但是由于生命周期短，那么他占用的内存并不是马上释放，而是被标记为老对象，这个时候该象还是要存在一段时间。然后由JVM决定他是否是垃圾对象，并进行回收。
所以我们可以知道，垃圾内存并不是用完了马上就被释放，所以就会产生内存释放不及时的现象，从降低了内存的使用。而当程序浩大的时候。这种现象更为明显，并且GC的工作也是需要消耗资源的所以，也就会产生内存浪费。

JVM中的对象生命周期里谈内存回收：
对象的生命周期一般分为7个阶段：创建阶段，应用阶段，不可视阶段，不可到达阶段，可收集阶段终结阶段，释放阶段。
创建阶段：首先大家看一下，如下两段代码：
test1：
    for（ int i=0; i《10000; i++）
    Object obj=new Object（）;
    test2：
    Object obj=null;
    for（ int i=0; i《10000; i++）
    obj=new Object（）;
这两段代码都是相同的功能，但是显然test2的性能要比test1性能要好，内存使用率要高，这是为什呢？原因很简单，test1每次执行for循环都要创建个Object的临时对象，但是这些临时对象由于JVM的GC不能马上销毁，所以他们还要存在很长时间而test2则只是在内存中保存一份对象的引用，而不必创建大量新临时变量，从而降低了内存的使用。
另外不要对同一个对象初始化多次。例如：
    public class A{
        private Hashtable table = new Hashable（）;
        public A（）{ table = new Hashtabl（）;
        // 这里应该去掉，因为table已经被初化。
    }
这样就new了两个Hashtable，但是却只使用了一个。另外一个则没有被引用。而被忽略掉。浪费了存。并且由于进行了两次new操作。也影响了代码的执行速度。
应用阶段：即该对象至少有一个引用在维护他。
不可视阶段：即超出该变量的作用域。这里有一个很好的做法，因为JVM在GC的时候并不是马上进回收，而是要判断对象是否被其他引用在维护。所以，这个时候如果我们在使用完一个对象以后对其obj=null或者obj.doSomething（）操作，将其标记为空，可以帮助JVM及时发现这个垃圾对象。

不可到达阶段：就是在JVM中找不到对该对象的直接或者间接的引用。
可收集阶段，终结阶段，释放阶段：此为回收器发现该对象不可到达，finalize方法已经被执行，或对象空间已被重用的时候。

JAVA的析构方法：
可能不会有人相信，JAVA有<a href="http://zhidao.baidu.com/search?word=%E6%9E%90%E6

9E%84%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" target="_blank">析函数？ 是的，有。因为JAVA所有类都继承至Object类，而finalize就是Object类的一个方法，个方法在JAVA中就是类似于C++<a href="http://zhidao.baidu.com/search?word=%E6%9E%90E6%9E%84%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" target="_blank"析构函数。一般来说可以通过重载finalize方法的形式才释放类中对象。如：
    public class A{
        public Object a;
        public A（）{ a = new Object ;}
        protected void finalize（） throws /span>java.lang<span style "color:#FF0000">.Throwable{
        a = null; // 标记为空，释放对象<br /

        super.finalize（）; // 递归用超类中的finalize方法。
        }
    }
当然，什么时候该方法被调用是由JVM来决定的。..。..。..。..。..。..。..。.
一般来说，我们需要创建一个destory的方法来显式的调用该方法。然后在finalize也对该方法进行调，实现双保险的做法。
由于对象的创建是递归式的，也就是先调用超级类的造，然后依次向下递归调用</a构造函数，所以应该避免类的构造函数中初始化变，这样可以避免不必要的创建对象造成不必要的内存消耗。当然这里也就看出来接口的优势。

数组的创建：
由于数组需要给定一个长度，所以在不确定数据数量的时候经常会创建过大，或过小的数组的现象。成不必要的内存浪费，所以可以通过软引用的方式来告诉JVM及时回收该内存。（软引用，具体查资）。
例如：
    Object obj = new char［10000000 00000000］;
    SoftReference ref = new SoftReference（obj）;
共享静态存储空间：
我们都知道静态变量在程运行期间其内存是共享的，因此有时候为了节约内存工件，将一些变量声明为静态变量确实可以起到节约内存空间的作用。但是于静态变量生命周期很长不易被系统回收，所以使用静态变量要合理，不能盲目的使用。以免适得其反。
因此建议在下面情况下使用：

变量所包含的对象体积较大，占用内存过多。
变量所包含对象生命周期较长。
变量所包含数据稳定。
该类的对象实例有对该变量所包含的对象的共享需求。（也就是说是否需要作为<a href="http
//zhidao.baidu.com/search?word=%E5%85%A8%E5%B1%80%E5%8F%98%E9%87%8F&f
=qb_search_exp&ie=utf8" target="_blank">全局变量）。

对象重用与GC：
有的时候，如数据库操作对象，一般情况下我们都需要在各个不同模块间使用，所以这样的对象需要
行重用以提高性能。也有效的避免了反复创建对象引起的性能下降。
一般来说对象池是一个不错的注意。如下：
    public abstarct class ObjectPool{<
r />
        private Hashtable locked，unlocked

        private long expirationTime;
        abstract Object create（）;
        abstract void expire（ Object o）;<
r />
        abstract void validate（ Object o）；
br />
        synchronized Object getObject（）
。..};
        synchronized void freeObject（Obj
ct o）{。..};
这样我们就完成了一个对象池，我们可以将通过对应的方法来存取删除所需对象。来维护这快内存提
内存重用。
当然也可以通过调用System.gc（）强制系统进行垃圾回收操作。当然这样的代价是需要消耗一些cp
资源。
不要提前创建对象：
尽量在需要的时候创建对象，重复的分配，构造对象可能会因为垃圾回收做额外的工作降低性能。
JVM内存参数调优：
强制内存回收对于系统自动的内存回收机制会产生负面影响，会加大系统自动回收的处理时间，所以
该尽量避免显式使用System.gc（），
JVM的设置可以提高系统的性能。

例如：
java -XX:NewSize=128m -XX:MaxNewSize=128m -XX:Survivor
atio=8 -Xms512m -Xmx512m

具体可以查看java帮助文档。我们主要介绍程序设计方面的性能提高。<b
/>
JAVA程序设计中有关<a href="http://zhidao.baidu.com/search?word=%E5%86%85%E5%AD%
8%E7%AE%A1%E7%90%86&fr=qb_search_exp&ie=utf8" target="_blank">内存管理
/a>的其他经验：
根据JVM<a href="http://zhidao.baidu.com/search?word=%E5%86%85%E5%AD%98%E7%AE
A1%E7%90%86&fr=qb_search_exp&ie=utf8" target="_blank">内存管理的工作
理，可以通过一些技巧和方式让JVM做GC处理时更加有效。，从而提高内存使用和缩短GC的执行时
。

尽早释放无用对象的引用。即在不使用对象的引用后设置为空，可以加速GC的工作。（当然果是返回值。..。.）
尽量少用finalize函数，此函数是JAVA给程序员提供的一个释放对象或资源的机会，但是却会大GC工作量。
如果需要使用到图片，可以使用soft应用类型，它可以尽可能将图片读入内存而不引起OutOfemory.
注意集合数据类型数据结构，往往数据结构越复杂，GC工作量更大，处理更复杂。
尽量避免在默认构造器（<a href="http://zhidao.baidu.com/search?word=%E6%9E%849%80%A0%E5%87%BD%E6%95%B0&fr=qb_search_exp&ie=utf8" target="_blank"构造函数）中创建，初始化大量的对象。
尽量避免强制系统做垃圾回收。会增加系统做垃圾回收的最终时间降低性能。
尽量避免显式申请数组，如果不得不申请数组的话，要尽量准确估算数组大小。
如果在做远程方法调用。要尽量减少传递的对象大小。或者使用瞬间值避免不必要数据的传递
尽量在合适的情况下使用对象池来提高系统性能减少内存开销，当然，对象池不能过于庞大，适得其反.
