jvm编译优化技术

<https://blog.csdn.net/qq_42998380/article/details/81907164>

我们将继续查看如下的几项最有代表性的优化技术是如何运作的，它们分别是：

* 语言无关的经典优化技术之一：公共子表达式消除。
* 语言相关的经典优化技术之一：数组范围检查消除。
* 最重要的优化技术之一：方法内联。
* 最前沿的优化技术之一：逃逸分析。

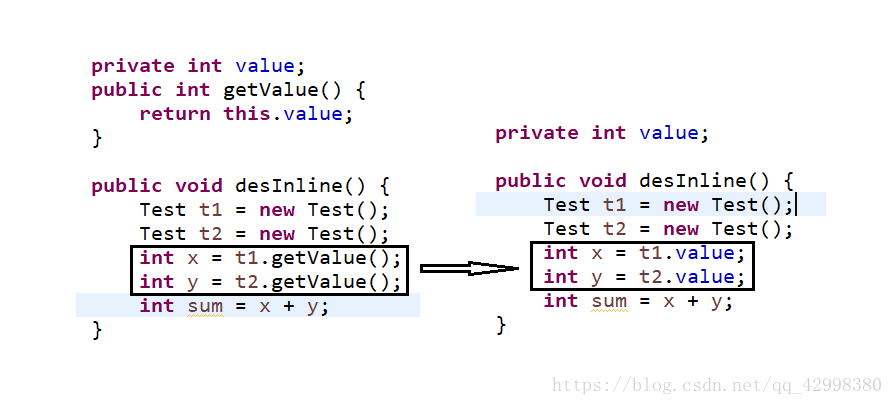
## 公共子表达式消除

公共子表达式消除是一个普遍应用于各种编译器的经典优化技术，它的含义是：  
如果一个表达式E已经计算过了，并且从先前的计算到现在E中所有变量的值都没有发生变化，那么E的这次出现就成为了公共子表达式。对于这种表达式，没有必要花时间再对它进行计算，只需要直接用前面计算过的表达式结果代替E就可以了。如果这种优化仅限于程序的基本块内，便称为局部公共子表达式消除（Local Common Subexpression Elimination），如果这种优化的范围涵盖了多个基本块，那就称为全局公共子表达式消除（Global CommonSubexpression Elimination）。

## 数组边界检查消除

数组边界检查消除（Array Bounds Checking Elimination）是即时编译器中的一项语言相关的经典优化技术。

## 方法内联

它是编译器最重要的优化手段之一，除了消除方法调用的成本之外，它更重要的意义是为其他优化手段建立良好的基础。  
编译器在进行内联时，如果是非虚方法，那么直接进行内联就可以了，这时候的内联是有稳定前提保障的。如果遇到虚方法，则会向CHA查询此方法在当前程序下是否有多个目标版本可供选择，如果查询结果只有一个版本，那也可以进行内联，不过这种内联就属于激进优化，需要预留一个“逃生门”（Guard条件不成立时的Slow Path），称为守护内联（GuardedInlining）。如果程序的后续执行过程中，虚拟机一直没有加载到会令这个方法的接收者的继承关系发生变化的类，那这个内联优化的代码就可以一直使用下去。但如果加载了导致继承关系发生变化的新类，那就需要抛弃已经编译的代码，退回到解释状态执行，或者重新进行编译。  
举个栗子：  
  
方法内联的重要性要高于其他优化措施，它的主要目的有两个：  
一是去除方法调用的成本（如建立栈帧等）。  
二是为其他优化建立良好的基础，方法内联膨胀之后可以便于在更大范围上采取后续的优化手段，从而获取更好的优化效果。

## 逃逸分析

逃逸分析（Escape Analysis）是目前Java虚拟机中比较前沿的优化技术，它与类型继承关系分析一样，并不是直接优化代码的手段，而是为其他优化手段提供依据的分析技术。

逃逸分析的基本行为就是分析对象动态作用域：当一个对象在方法中被定义后，它可能被外部方法所引用，例如作为调用参数传递到其他方法中，称为方法逃逸。甚至还有可能被外部线程访问到，譬如赋值给类变量或可以在其他线程中访问的实例变量，称为线程逃逸。

如果能证明一个对象不会逃逸到方法或线程之外，也就是别的方法或线程无法通过任何途径访问到这个对象，则可能为变量进行一些高效的优化：

1. 栈上分配（Stack Allocation）：Java虚拟机中，在Java堆上分配创建对象的内存空间几乎是Java程序员都清楚的常识了，Java堆中的对象对于各个线程都是共享和可见的，只要持有这个对象的引用，就可以访问堆中存储的对象数据。虚拟机的垃圾收集系统可以回收堆中不再使用的对象，但回收动作无论是筛选可回收对象，还是回收和整理内存都需要耗费时间。如果确定一个对象不会逃逸出方法之外，那让这个对象在栈上分配内存将会是一个很不错的主意，对象所占用的内存空间就可以随栈帧出栈而销毁。在一般应用中，不会逃逸的局部对象所占的比例很大，如果能使用栈上分配，那大量的对象就会随着方法的结束而自动销毁了，垃圾收集系统的压力将会小很多。
2. 同步消除（Synchronization Elimination）：线程同步本身是一个相对耗时的过程，如果逃逸分析能够确定一个变量不会逃逸出线程，无法被其他线程访问，那这个变量的读写肯定就不会有竞争，对这个变量实施的同步措施也就可以消除掉。
3. 标量替换（Scalar Replacement）：标量（Scalar）是指一个数据已经无法再分解成更小的数据来表示了，Java虚拟机中的原始数据类型（int、long等数值类型以及reference类型等）都不能再进一步分解，它们就可以称为标量。相对的，如果一个数据可以继续分解，那它就称作聚合量（Aggregate），Java中的对象就是最典型的聚合量。如果把一个Java对象拆散，根据程序访问的情况，将其使用到的成员变量恢复原始类型来访问就叫做标量替换。如果逃逸分析证明一个对象不会被外部访问，并且这个对象可以被拆散的话，那程序真正执行的时候将可能不创建这个对象，而改为直接创建它的若干个被这个方法使用到的成员变量来代替。将对象拆分后，除了可以让对象的成员变量在栈上（栈上存储的数据，有很大的概率会被虚拟机分配至物理机器的高速寄存器中存储）分配和读写之外，还可以为后续进一步的优化手段创建条件。

有时间再看

<https://www.jianshu.com/p/d09adaf6308a>