Visual Basic 6 中的性能陷井及预防方法

叶孝斌 （长沙 国防科大计算机学院 410073）

摘要：本文利用性能分析工具——TrueTime 分析了 Visual Basic 存在的性能问题，找出了 Visual Basic 语言的性能陷阱，最后给出了预防措施。

关键词：性能 p 代码 优化 函数调用

一、引言

由于 Visual Basic 存在的性能问题，Visual Basic 用户长期以来成为 C++ 编程者的笑柄。大多数的 Visual Basic 性能问题都来源于 p 代码。随着带有本地代码(native-code)编译器的 Visual Basic 5.0 的推出，终于为广大开发者带来了一些欣喜。但是使用一段时间后，人们发现本地代码并没有取得全面的性能提高。本地代码编译器使像循环、算术运算那样集中使用 CPU 的代码速度提高很多，但对用户接口代码却没有太多提高。这些调用 ActiveX 控件和 Windows API 的用户接口代码运行慢的原因是它们被本地代码而不是 p 代码调用。

除了替换执行慢的 ActiveX 控件外，优化 Visual Basic 程序最有效的方法是优化所使用的语句。本文将介绍一些经过试验并被证明是正确的、可使用于所有编程语言的、提高速度的技术和仅能使用于 Visual Basic 的技术。

二、分析工具—True Time

True Time 是 Compuware 公司的 Visual Basic 程序性能分析工具，它可以将原代码中的每一行、函数、模块和可执行模块的执行信息详细显示出来。

Count 是程序执行过程中该行代码的执行次数。% with Children 是该行代码的执行时间，以占整个程序执行时间的百分比表示。Functions 是执行该行代码所需要调用的函数个数。True Time 还可以对原代码进行其他统计，其中这三个统计数字最有用。Count 列使程序员很容易找出执行频繁的代码。因此，即使很小的性能提高都会使原程序的性能有实质性的提高。Functions 列使你能在一行原代码进行多次函数调用时重新考虑一种简单的方法。% with Children 显示的是一行相对于整个应用程序的性能。这使得比较不同方法的效果，从而找出较好的方法变得很容易。True Time 还可以显示每行代码的实际执行时间(以 CPU 周期、微秒、毫秒或秒为单位)。本文中使用一种技术比另一种技术快百分之 x 的方法进行比较。

Visual Basic 作了大量的令人疲劳的工作而使编程变得容易。不幸的是这些令人疲劳的工作还是得做。这意味，比如，表面上看起来无伤大雅的一行源代码可以引起一系列耗时的函数调用。

三、性能陷阱分析

本节中将介绍一系列在开发 Visual Basic 代码时所要面临的带有性能陷阱的使用语句的选择性问题。如应该使用数组还是集合？在临时变量中保存属性值是一个好方法吗？变体类型真的象人们说的那样不好吗等。下面一一解答这些问题。

1. 固定数组和动态数组

动态数组在运行时可以调整尺寸，这是 Visual Basic 语言中增加的一个强大的重要功能。然而随着功能的增强，开销也随着增加。动态数组是在运行时通过对 Visual Basic 运行库进行函数调用来创建和存储的。大小在编译时设置的固定数组速度快。

2. 集合和动态数组

Visual Basic 的集合允许将类型不同的多个相关数据存储在一起，并使用索引或者任意键值查找数据项。而数组的元数必须是相同类型的，且只能用索引存取。

上例给一个动态字符串数组增加 100 个元数，给一个集合加入 100 个字符串项。然后增加数组的尺寸并加入另外 100 个元数，而集合可以自动加入 100 个其他字符项。最后比较从数组和集合中分别取出一个字符串的读取速度。

3. 存储属性值

似乎把对象属性看作变量是件很诱人的事，然而其实是另外一回事。对象属性一点也不象变量。Visual Basic 以我们的利益出发使属性看起来象并表现得也象变量，然而这使我们付出了性能代价，尤其是在大的循环

中存取属性时。

缓存属性值,把它们保存在变量中并在以后的代码中通过变量引用它们,将减少 Visual Basic 访问属性时必须调用函数的次数。

4. 移动控件

由于访问属性值较慢,如果你有其他方法可以获得同样的效果就使用它。如果想移动控件,使用控件的 Move 方法,而不要设置它的 Left 和 Top 属性。这样作速度快的原因是因为设置属性将导致控件或表单移动两次,每一个属性移动一次,而调用 Move 方法则只要移动一次。Move 方法的 Width 和 Height 参数是可选的,因此在调用 Move 方法时没有必要取得高和宽的值。另外 Move 方法一步就完成移动过程,而如果分别改变 Left 和 Top 属性值则要一步一步地移动控件或表单。

5. 变体类型

变体类型十分有用而且易于使用,因为不用担心为变体变量声明类型,也不用处理复杂的类型检查。但是当要创建应用程序时,肯定不想受变体的内在低性能的限制。虽然可以认为变体是无类型的,但实际上它只是一个可以存储任意类型数据的变量,变体存储空间的一部分存放的是时间提供数据的类型标识符。Visual Basic 必须自动在数据的实际类型和执行某一功能所需要的数据类型之间进行转换,这种转换需要花费时间。

大多数其他的数据类型都可以在计算机的处理器中直接表示和操作,如 bytes, integers, longs, singles, 和 doubles,它们都可直接由硬件单元处理。但变体则不能,Visual Basic 必须调用其运行库中的函数来操作变体,因而减慢了速度。另外,每一变体数据要占用 16 字节的内存,相当于 Double 类型变量的两倍,会造成内存的浪费。

以下是两种避免使用变体的方法:

·用 Option Explicit 对变量进行显式地声明,不然 Visual Basic 将忘记声明的变量看作一个变体。

·给变量一个明确的类型

Dim a, b, c, d As Integer 会将 a,b,c 创建成变体;

使用下面的说明语句替代:

Dim a As Integer, b As Integer, c As Integer, d As Integer

6. 后期绑定和前期绑定

后期绑定一定比前期绑定慢,这一点大家都不会惊奇。但究竟慢多少也许会使人感到惊讶。

7. With 语句

也许你会认为,象保存属性值可以提高性能一样,使用带属性值的 With 语句同样可以提高性能。其实不然,读取一个属性值的性能开销来自于获得该属性值时的隐含方法调用,而不是对象本身。因此,对一对象使用 With 语句并不能带来明显的性能改善。

8. 整数和长整数

对于一个使用 C++ 编译器很长时间的人来说,在 32 位 CPU 和 32 位操作系统中,32 位的整数比 16 位的整数快是肯定的。实际上,C++ 编译器根据 CPU 和操作系统改变普通整数数据类型的大小。这样做的原因是为了提高性能,32 位的 CPU 可以自然地处理 32 为整数,而 16 位的 CPU 则需要作额外的工作。但是 Visual Basic 不是这样,即使经过 162,000 多个循环之后,使用 16 位整数和 32 位整数的差别只有 0.39 微秒。因此不要在这上面浪费时间。

9. 符号:"+" 和 "&"

在 Visual Basic 的过去几个版本中,字符串连接操作符"+"和"&"的性能已经改进了很多。Visual Basic6.0 中是这样说明的:当连接字符串时,使用"&"操作符,因为它更快,并且它还使用更少的强制类型转换和变体。Visual Basic 中关于"+"操作符的说明如下:用于求两数之和。紧跟着的是一系列规则和当一个操作数是字符串或字符串变量时"+"操作符工作的例外处理。因此,当需要求数字之和时,使用"+"操作符;当要连接字符串时,使用"&"操作符,这样会提高速度。

四、结束语

Visual Basic 已经有一个速度慢的坏名声。这个坏名声大部分是由 Windows 3.x 上 Visual Basic 的 16 位版本得到的。在 Windows 95, Windows 98 和 Windows NT 上, Visual Basic 6.0 可以象 Visual C++ 和 Delphi 那样通过公认的速度监控程序保持较高的性能。

然而象其他计算机工具一样,清除的垃圾=带进的垃圾。在赞叹变体类型和后期绑定给我们带来的好处时,也容易创建速度慢的代码。因此,使用本文的技巧可以写出速度更快、性能更好的代码。

参考文献

[1] Steve Brown 著,余青霓、周钢等译,Visual Basic 5.0 教程,电子工业出版社,1998.2

[2] Microsoft Corporation 著,微软(中国)有限公司 译,Visual Basic 6.0 Programmer' Guide

(来稿时间:1999 年 9 月)