基于 VB.NET 的多线程技术应用①

Implementation of Multithread Technique Based on VB.NET

张焰林　(温州职业技术学院 计算机系 浙江 温州 325035)

摘　要： 采用多线程技术可充分提高应用程序运行效率，微软的.NET 框架提供了新的线程类库，从而可以方便地创建多线程应用程序。本文首先简述了多线程应用的意义，介绍了进程、线程以及应用程序域的相互关系，随后举例介绍了使用 Visual Basic.NET 进行线程的创建与管理的过程，通过对 Thread 基类的引用可创建一个线程，然后可利用线程的 start 等方法属性进行线程的管理，并强调了线程的同步技术的应用，通过多线程编程技术的应用来开发效率更高、响应速度更快的应用程序。

关键词： 多线程 同步 应用程序 .NET 框架 VB.NET

1　引言

1.1 多线程的意义

采用多线程技术的应用程序可以较好地利用系统资源。多线程技术的主要优势在于充分利用了 CPU 的空闲时间片，用尽可能少的时间来对用户的要求做出响应，使得进程的整体运行效率得到较大提高，同时增强了应用程序的灵活性。更重要的是，由于同一进程的所有线程是共享同一内存，所以不需要特殊的数据传送机制，不需要建立共享存储区或共享文件，从而使得不同任务之间的协调操作与运行、数据的交互、资源的分配等问题更加易于解决。

在一台拥有多个处理器的机器上，来自一个应用程序域 AppDomain 的线程可分配在所有的处理器上运行，从而允许同时地运作。在分布式应用时，可提升扩展性，因为更多的客户可以分享一个服务器上的 CPU 资源，而对于桌面的应用程序，例如 Excel 和 Word 等，也能够从线程中得到好处，比如可以执行后台的操作例如重新计算和打印等。

1.2 进程与线程和应用程序域

在操作系统中，进程是应用程序的运行实例，是应用程序的一次动态执行，其所有线程共享虚拟地址空间、全局变量和该进程的操作资源。而线程是进程内部程序执行的路径，是进程的一个执行单元。线程是操作系统分配 CPU 时间的基本单位，一个线程可以执行应用程序的任何部分，包括当前被其它线程执行的部分。一个应用程序至少包括一个主线程， 还可拥有其它辅线程，当一个应用程序中的线程多于一个时，就称该程序是多线程的。

为了运行所有这些线程，操作系统为每个独立线程安排一些 CPU，并通过其本身的调度机制来评价各个活动线程的优先级，优先级别高的活动优先执行，优先级别低的活动线程挂起。创建一个进程时，它的第一个线程称为主线程， 它由系统自动生成， 然后可由这个主线程生成辅助线程，这些辅助线程又可生成更多的线程。

在.NET 框架中，所有程序编译后生成的都是中间代码，而这些中间代码的隔离、加载和卸载以及安全边界的提供都是通过应用程序域来实现的。此时，一个进程可以包含一个或多个应用程序域，而一个应用程序域又可以包含一个或多个线程。这样实际上就相当于在进程和线程之间增加了一个新的安全边界。无论在同一个进程之内还是在不同的进程之间，每个应用程序域之间都是相互无关的，这些不同的应用程序域之间只能通过远程通讯来实现消息和对象的传递。

2　Visual basic.net对多线程的支持

2.1 VB 对多线程的支持

VB6.0 及其以前的版本较少涉及到多线程的问

① 收稿时间:2008-08-18

题，因为 VB 并不是用来处理多线程应用的，多线程模式的工作原理和编程机制对于 VB 并不完全适合，会导致访问违例和内存错误。在 VB6.0 的应用中，我们可以通过 Win32 CreateThread API 来创建一个多线程的应用，或者通过 COM 库而在一个独立的线程中创建一个组件，但这些技术设计调试起来都是有不小难度的，故应用面非常窄。

2.2 Visual Basic.NET 对多线程的支持

Visual Basic.NET 是基于.NET 框架的，而.NET 框架的重要组成部分 CLR(Common Language Runtime，通用语言运行时)内置支持多线程应用，可以通过系统的 Threading 类直接建立多线程应用程序，并且支持线程池等高级功能。任何.NET 框架结构下的语言，包括 VB、C#等在编写多线程应用程序的时候，都可以利用系统类所提供的对象和方法，而不再需要使用 Win32 API，因此可以大大降低开发的难度，减少错误所发生的几率。

3 Visual Basic.NET中多线程编程的实现

3.1 线程的创建与管理

创建和维护线程的基类是 System. Threading. Thread 类，它能够创建并控制线程，设置其优先级并获取其状态。

在创建线程前必须引用 threading 基类：Imports System.Threading。然后可创建一个新的线程类实例，并使用 AddressOf 语句为要运行的线程指定任务。接着可以利用 Start、Resume、Suspend、Sleep、Stop、Abort 和 Join 等方法操纵线程，还可以通过如 IsAlive、IsBackground、Priority、ApartmentState 和 ThreadState 等属性查询和设置线程状态。如使用 Thread 类的 Sleep 方法可以阻滞当前线程，使用 Suspend 方法可以挂起线程，使用 Resume 可以重新启动挂起的线程，使用 Abort 方法可以停止一个线程，使用 Join 方法可以使当前线程等待其它线程运行结束。线程创建的实例如下：

首先定义了一个名为 compare 的类，并在其中创建了一个名为 max 的过程用于求两个数的最大值。

然后引用线程基类，定义一个名为 T1 的线程，并将线程 T1 指向 compare 类中的 max 过程，为 compare 类中的两个参数 x 和 y 赋值，最后启动线程 T1 以获取 x 和 y 中的最大值。

```
Public Class compare
    Public x As Double, y As Double, z As Double
    Public Sub max()
        If x > y Then
            z = x
        Else
            z = y
        End If
    End Sub
End Class
```

图 1　类的创建

```
Imports System.Threading '引用线程基类
Public Class Form1
    Dim t1 As Thread '定义线程
    Dim cmp As New compare()
    Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        t1 = New Thread(AddressOf cmp.max) '线程实例化
        cmp.x = 90
        cmp.y = 10
        t1.Start() '启动线程
    End Sub
End Class
```

图 2　线程的创建与启动

3.2 线程的优先级

不同的线程具有不同的优先级 Priority，而优先级决定了线程能够得到多少 CPU 时间。高优先级的线程通常会比一般优先级的线程得到更多的 CPU 时间，如果程序中存在不止一个高优先级的线程，操作系统将在这些线程之间循环分配 CPU 时间。一旦低优先级的线程在执行时遇到了高优先级的线程，它将让出 CPU 给高优先级的线程。在 VB.NET 中，Threading.Thread.Priority 枚举了线程的优先级别，这些级别包括 Highest、AboveNormal、Normal、BelowNormal、Lowest。新创建的线程初始优先级为 Normal。

3.3 线程的状态

线程从创建到终止，它一定处于某一个状态，而这个状态是由 ThreadState 属性定义的。当一个线程刚被创建时，它处在 Unstarted 状态，然后 Thread 类的 Start() 方法将使线程状态变为 Running 状态，如果不调用相应的方法使线程挂起、阻塞或者终止，则线程将一直保持这样的状态。挂起的线程处于 Suspended 状态，直到我们调用 resume()方法使其重新执行，这时候线程将重新变为 Running 状态。一旦线程被销毁或者终止，则线程处于 Stopped 状态，处于这个状态的线程将不复存在。线程还有一个 Background 状态，它表明线程运行在前台还是后台。而在一个确定的时间，线程可能处于多个状态。

3.4 线程池

使用多线程操作能优化应用程序性能，但是多线程往往需要花费更多的代码和精力去控制线程以实现线程之间的轮询和状态转换。使用线程池则可以自动完成这些工作，同时还可以优化计算机的访问性能，从而更加有效地利用多线程的优势。使用线程池，可以使用要运行的过程的委托来调用 ThreadPool 类的 QueueUserWorkItem 方法。以下代码说明了如何使用线程池启动多个任务。

```
Sub DoMyWork()
  Dim MPool As System.Threading.ThreadPool
  '将一个任务排队
  MPool.QueueUserWorkItem(New
  System.Threading.WaitCallback(AddressOf
Task1))
  MPool.QueueUserWorkItem(New
System.Threading.WaitCallback(AddressOf
Task2))
End Sub
```

如果要启动很多单独的任务，但并不需要单独设置每个线程的属性，则线程池将非常有用。每个线程都以默认的堆栈大小和优先级启动。默认情况下，每个系统处理器上最多可以运行 25 个线程池线程。超过该限制的其它线程会被排队，直至另外的线程运行结束后它们才能开始运行。

线程池并不是在所有的情况下都适用，当需要特定优先级的线程时就没法通过线程池来实现。

3.5 线程的同步

在多线程应用中，我们需要考虑不同线程之间的数据同步和防止死锁。当两个或多个线程之间同时等待对方释放资源的时候就会形成线程之间的死锁。为了防止死锁的发生，我们需要通过同步来实现线程安全。

VB.NET 提供了一些语句用于操纵线程的同步。在图 1 和图 2 的代码中，若想同步执行比较的线程以便等到线程完成便可以获得结果，可利用 VB.NET 提供的 SyncLock 语句和 Thread.Join 方法。

SyncLock 语句可获取对传递给它的对象引用的独占性锁。通过取得这种独占锁，可以确保多个线程不会访问共享的数据或是在多个线程上执行代码。用于获取锁的对象是关联于每个类的 System.Type 对象。通过 GetType 方法可以获得 System.Type 对象，此时图 1 中的代码可修改为图 3 所示。

Thread.Join 方法允许等待一段特定的时间直到一个线程结束，如果线程在所确定的时间之前完成，则 Thread.Join 返回 True，否则的话返回 False。在图 2 的例子中，可以调用 Thread.Join 方法来确定比较是否已经结束。代码如图 4 所示。

```
Public Class compare
    Public x As Double, y As Double, z As Double
    Public Sub max()
        SyncLock GetType(compare)
            If x > y Then
                z = x
            Else
                z = y
            End If
        End SyncLock
    End Sub
End Class
```

图 3　获取 type 对象

```
Imports System.Threading
Public Class Form1
    Dim t1 As Thread
    Dim cmp As New compare()
    Private Sub Button1_Click(ByVal sender As System.Object, _
    ByVal e As System.EventArgs) Handles Button1.Click
        t1 = New Thread(AddressOf cmp.max)
        cmp.x = 90
        cmp.y = 10
        t1.Start()
        If t1.Join(500) Then
            MsgBox("最大值为：" & cmp.z, MsgBoxStyle.OkOnly, _
                "线程管理")
        End If
    End Sub
End Class
```

图 4　线程同步执行

4　结束语

本文阐述了 Visual Basic.NET 中多线程开发技术如何实现。多线程技术是实现需要并发执行的应用程序的较好选择，尤其对于大部分时间被阻塞的程序段，例如在开发访问网络资源，系统开销比较大的操作或实现快速的用户界面响应时，具有不可替代的作用。但由于每个线程都需要额外的内存来创建，在执行时操作系统要跟踪和确定线程的进度，还必须为每个线程分配内存，同时还需要处理器时间片来运行和管理线程，因此如果创建的线程过多，应用程序所需要的系统开销会比较大反而会降低应用程序的性能。所以在设计多线程应用程序时，应慎重对待，必须经过仔细考虑才能加入多线程的支持，恰当地创建和使用多线程，以建立合理的系统模型，以便能够改善应用程序的结构，才能提高应用程序的运行效率，使应用程序获得最佳的性能。

参考文献

1 李太君,张树亮.利用 Visual Basic.Net 开发多线程应用程序.现代计算机, 2002,11:35－37.

2 冯美霞.多线程应用程序的同步技术.计算机应用,2001,6:24－26.

3 辛璐.LX_PVM 分布式多线程通讯库的实现技术.长沙大学学报,1998,4:33－34.

4 杨海娟.windows 编程中的线程同步.兰州教育学院学报, 2003,3:56.

5 王晨,张毅谨,宋俊德.Solaris 多线程体系结构研究及多线程应用.计算机系统应用,2000,9(2):42－45.