有一段代码用到了ArcGIS的组件（实例中用到的是Geometry命名空间下的投影转换函数），正常在主线程下运行时是正常的，但由于运行时间长，用户体验较差，就想把这部分代码放在BackGroundWorker下面运行，以提高用户体验，但迁移到BGW下时，发现无法正常运行了，

上网查了一下，似乎是ArcGIS的COM组件必须在STA模式下运行

（关于什么是STA  STA:single thread apartment    MTA:mutiple thread apartment

可以参考这个：<http://blog.sina.com.cn/s/blog_4e5ce4610100b645.html>）

但BGW只能是在MTA下运行，所以ArcGIS组件运行不正常

这个观点没有实际验证，有人针对这个给出了解决方案，大致思想是，不使用BackGroundWorker，而使用Thread实现（因为Thread可以设置在STA模式下运行，而BGW是包装过的，不可以）

实现方法在此：<http://stackoverflow.com/questions/658301/threading-and-arcgis>

还有一种观点：可以在BGW里通过Invoke调用，可以解决线程间的复杂问题，具体的方法为：

void bgw\_DoWork(object sender, DoWorkEventArgs e)

{

        this.Invoke((MethodInvoker)delegate

        {

                //ArcGIS相关操作

        });

}

经测试，成功，但也有人说这样同样会造成线程阻塞，用户体验下降，这个问题可以尝试这样：

如果BGW里的主要耗时部分是ArcGIS的代码，那么就只能尝试第一种解决方案

如果BGW里的主要耗时部分是非ArcGIS的代码，那么把ArcGIS相关代码使用Invoke包起来即可，其它的代码可以放在invoke外面

附，<http://stackoverflow.com/questions/658301/threading-and-arcgis>的原文

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 down vote[favorite](http://stackoverflow.com/questions/658301/threading-and-arcgis)  **1** | I have just stumbled across the Backgroundworker object, and it seems like the tool I'm looking for to make my GUI responding while performing calculations. I am writing IO plugins for ArcGIS.  I am doing some data processing outside ArcGIS, which works fine using the backgroundworker. But when I'm inserting the data into ArcGIS, the backgroundworker seems to increase the duration time by a factor 9 or so. Placing the processing code outside the DoWork method, increases the performance by a factor 9.  I have read about this several places on the net, but I have no experience in multithreaded programming and the terms like STA and MTA means nothing to me. [link text](http://edndoc.esri.com/arcobjects/9.2/NET/2c2d2655-a208-4902-bf4d-b37a1de120de.htm)I have also tried to use a simple implementation of threading, but with similar results.  Does anyone know what I can do to be able to use ArcGIS processing and maintaining a responsive GUI?  EDIT: I have included a sample of my interaction with the background worker. If I put the code located in the StartImporting method in the cmdStart\_Click method, it executes much faster.  private void StartImporting(object sender, DoWorkEventArgs e)  {      DateTime BeginTime = DateTime.Now;      // Create a new report object.      SKLoggingObject loggingObject = new SKLoggingObject("log.txt");      loggingObject.Start("Testing.");        SKImport skImporter = new SKImport(loggingObject);      try      {          // Read from a text box - no writing.      skImporter.Open(txtInputFile.Text);      }      catch      {      }      SKGeometryCollection convertedCollection = null;        // Create a converter object.      GEN\_SK2ArcGIS converter = new GEN\_SK2ArcGIS(loggingObject);        // Convert the data.      convertedCollection = converter.Convert(skImporter.GetGeometry());        // Create a new exporter.      ArcGISExport arcgisExporter = new ArcGISExport(loggingObject);        // Open the file.      // Read from a text box - no writing.      arcgisExporter.Open(txtOutputFile.Text);        // Insert the geometry collection.      try      {      arcgisExporter.Insert(convertedCollection);      }      catch      {      }      TimeSpan totalTime = DateTime.Now - BeginTime;      lblStatus.Text = "Done...";    }    private void ChangeProgress(object sender, ProgressChangedEventArgs e)  {      // If any message was passed, display it.      if (e.UserState != null && !((string)e.UserState).Equals(""))      {      lblStatus.Text = (string)e.UserState;      }      // Update the progress bar.      pgStatus.Value = e.ProgressPercentage;  }    private void ImportDone(object sender, RunWorkerCompletedEventArgs e)  {      // If the process was cancelled, note this.      if (e.Cancelled)      {      pgStatus.Value = 0;      lblStatus.Text = "Operation was aborted by user...";      }      else      {      }    }    private void cmdStart\_Click(object sender, EventArgs e)  {      // Begin importing the sk file to the geometry collection.        // Initialise worker.      bgWorker = new BackgroundWorker();      bgWorker.RunWorkerCompleted += new RunWorkerCompletedEventHandler(ImportDone);      bgWorker.ProgressChanged += new ProgressChangedEventHandler(ChangeProgress);      bgWorker.DoWork += new DoWorkEventHandler(StartImporting);      bgWorker.WorkerReportsProgress = true;      bgWorker.WorkerSupportsCancellation = true;        // Start worker.      bgWorker.RunWorkerAsync();    }    private void cmdCancel\_Click(object sender, EventArgs e)  {      bgWorker.CancelAsync();  }  Kind Regards, Casper  [c#](http://stackoverflow.com/questions/tagged/c%23) [multithreading](http://stackoverflow.com/questions/tagged/multithreading) [backgroundworker](http://stackoverflow.com/questions/tagged/backgroundworker) [arcgis](http://stackoverflow.com/questions/tagged/arcgis)   |  |  |  | | --- | --- | --- | | [share](http://stackoverflow.com/q/658301)|[improve this question](http://stackoverflow.com/posts/658301/edit) | [edited Mar 25 '09 at 9:48](http://stackoverflow.com/posts/658301/revisions) | asked Mar 18 '09 at 13:42  [[C:\Users\wyz\Documents\My Knowledge\temp\fbaa142b-5a33-40bf-a6dc-3a0e85a30bd8_4_files\cf3f5dda4a8d1034fdbee468a288640e[1].jpg](http://stackoverflow.com/users/62361/chau)](http://stackoverflow.com/users/62361/chau)  [Chau](http://stackoverflow.com/users/62361/chau) 2,77522853 | |
|  | |  |  | | --- | --- | |  | What kind of IO extension are you writing for ArcGIS? Are you using geoprocessing, catalog extension, arcmap extension,...? – [Jordan Parmer](http://stackoverflow.com/users/20133/jordan-parmer) Mar 18 '09 at 14:18 | |  | It is for use in ArcMap as a button, and it either creates a personal geodatabase or reads the features from a geodatabase. Does that answer your question? My venture into ArcGIS is a slow-steep-learning process :-) – [Chau](http://stackoverflow.com/users/62361/chau) Mar 18 '09 at 14:37 | |

**3 Answers**

[active](http://stackoverflow.com/questions/658301/threading-and-arcgis?answertab=active#tab-top) [oldest](http://stackoverflow.com/questions/658301/threading-and-arcgis?answertab=oldest#tab-top) [votes](http://stackoverflow.com/questions/658301/threading-and-arcgis?answertab=votes#tab-top)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| up vote2down vote | It is correct that you should use STA threads when working with the COM objects in ArcGIS. Still, you can get the convenience of the BackgroundWorker, which always is an MTA thread from the system's thread pool.  private static void OnBackgroundWorkerDoWork(object sender, DoWorkEventArgs e)  {      BackgroundWorker worker = (BackgroundWorker)sender;      ToolToStart tool = e.Argument as ToolToStart;        if (tool != null)      {          tool.BackgroundWorker = worker;            // The background worker thread is an MTA thread,          // and should not operate on ArcObjects/COM types.          // Instead we create an STA thread to run the tool in.          // When the the tool finishes the infomation from the STA thread          // is transferred to the background worker's event arguments.          Thread toolThread = new Thread(STAThreadStart);          toolThread.SetApartmentState(ApartmentState.STA);          toolThread.Start(tool);            toolThread.Join();          e.Cancel = m\_ToolCanceled;          e.Result = m\_ToolResult;      }  }  The STA thread can now use the BackgroundWorker's methods, such as reporting progress, checking for cancellation and reporting results.  protected virtual void StatusUpdateNotify(ProgressState progressState)  {      if (BackgroundWorker.CancellationPending)      {          throw new OperationCanceledException();      }        BackgroundWorker.ReportProgress(progressState.Progress, progressState);  }  In addition to only using STA threads when operating on ArcGIS objects, you should not share objects between two threds. From your code it seems like you access the GUI from the background worker:lblStatus.Text = "Done...";, which could be done in e.g. the delegate for RunWorkerComplete.   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/825335)|[improve this answer](http://stackoverflow.com/posts/825335/edit) | answered May 5 '09 at 15:10  [[C:\Users\wyz\Documents\My Knowledge\temp\fbaa142b-5a33-40bf-a6dc-3a0e85a30bd8_4_files\87bc17306d1373de6091cb97ed1bbf81[1].png](http://stackoverflow.com/users/98350/zoidberg)](http://stackoverflow.com/users/98350/zoidberg)  [Zoidberg](http://stackoverflow.com/users/98350/zoidberg) 1,171269 | | |
|  |  | |
| up vote1down vote | | Typically in order to maintain a responsive GUI you will want to execute your code that does the work in a different thread. This is made very easy with .net using the BeginInvoke method:[http://msdn.microsoft.com/en-us/library/aa334867(VS.71).aspx](http://msdn.microsoft.com/en-us/library/aa334867%28VS.71%29.aspx)  In a nutshell, include all the non GUI code in a separate class (or classes) and rather than calling each metho directly you create a delegate and call the BeginInvoke method on that. The method will then go off and do it's thing with no further interaction with the GUI. If you wish to have it update the GUI (e.g. a progress bar) then you can raise events from the class and catch them from the GUI however you will need to ensure that controls are updated in a thread safe fashion. If you want the GUI to update when the method completes then you can use the EndInvoke method to handle that   |  |  | | --- | --- | | [share](http://stackoverflow.com/a/658540)|[improve this answer](http://stackoverflow.com/posts/658540/edit) | answered Mar 18 '09 at 14:37  [[C:\Users\wyz\Documents\My Knowledge\temp\fbaa142b-5a33-40bf-a6dc-3a0e85a30bd8_4_files\481ab95f1ad91345a6faf2c93cdde4da[1].png](http://stackoverflow.com/users/79129/macros)](http://stackoverflow.com/users/79129/macros)  [Macros](http://stackoverflow.com/users/79129/macros) 4,2271236 | |
|  | | |  |  | | --- | --- | |  | Except for using the BeginInvoke end EndInvoke, I understand the "Nutshell" and I'm using events to update the GUI. But it is the ArcGIS vs. threading performance decrease I don't get. Probably, because I don't understand threading well enough. – [Chau](http://stackoverflow.com/users/62361/chau) Mar 18 '09 at 15:39 | |  | I'm not sure why it should decrease performance so much - do you have the code examples (using backgroundworker and not using it)? – [Macros](http://stackoverflow.com/users/79129/macros) Mar 18 '09 at 17:30 | |

|  |  |  |
| --- | --- | --- |
| up vote1 down voteaccepted | I have continued trying to find a solution, and the following is what I ended up doing. The code is cut and paste from various files and presented to give an idea of what I did. It demonstrates how I can call methods which are communicating with ArcGIS using a thread. The code allows me to update the GUI in the main thread, abort the operation, and do post-operation stuff. I ended up using the first threading part from the link I posted initially.  The reason for the initial loss of performance is probably due to the single-threaded apartment (STA) which is required by ArcGIS. The Backgroundworker seems to be MTA, thus not appropriate for working with ArcGIS  Well here it goes, I hope I haven't forgotten anything, and feel very free to edit my solution. It will both help me and probably also other people developing stuff for ArcGIS.  public class Program  {      private volatile bool AbortOperation;      Func<bool> AbortOperationDelegate;      FinishProcessDelegate finishDelegate;      UpdateGUIDelegate updateGUIDelegate;        private delegate void UpdateGUIDelegate(int progress, object message);      private delegate void FinishProcessDelegate();        private void cmdBegin\_Click(...)      {          // Create finish delegate, for determining when the thread is done.          finishDelegate = new FinishProcessDelegate(ProcessFinished);          // A delegate for updating the GUI.          updateGUIDelegate = new UpdateGUIDelegate(UpdateGUI);          // Create a delegate function for abortion.          AbortOperationDelegate = () => AbortOperation;            Thread BackgroundThread = new Thread(new ThreadStart(StartProcess));          // Force single apartment state. Required by ArcGIS.          BackgroundThread.SetApartmentState(ApartmentState.STA);          BackgroundThread.Start();      }        private void StartProcess()      {          // Update GUI.          updateGUIDelegate(0, "Beginning process...");            // Create object.          Converter converter = new Converter(AbortOperationDelegate);          // Parse the GUI update method to the converter, so it can update the GUI from within the converter.          converter.Progress += new ProcessEventHandler(UpdateGUI);          // Begin converting.          converter.Execute();            // Tell the main thread, that the process has finished.          FinishProcessDelegate finishDelegate = new FinishProcessDelegate(ProcessFinished);          Invoke(finishDelegate);            // Update GUI.          updateGUIDelegate(100, "Process has finished.");      }        private void cmdAbort\_Click(...)      {          AbortOperation = true;      }        private void ProcessFinished()      {          // Post processing.      }        private void UpdateGUI(int progress, object message)      {          // If the call has been placed at the local thread, call it on the main thread.          if (this.pgStatus.InvokeRequired)          {              UpdateGUIDelegate guidelegate = new UpdateGUIDelegate(UpdateGUI);              this.Invoke(guidelegate, new object[] { progress, message });          }          else          {              // The call was made on the main thread, update the GUI.              pgStatus.Value = progress;              lblStatus.Text = (string)message;          }      }  }    public class Converter  {      private Func<bool> AbortOperation { get; set;}        public Converter(Func<bool> abortOperation)      {          AbortOperation = abortOperation;      }        public void Execute()      {          // Calculations using ArcGIS are done here.          while(...) // Insert your own criteria here.          {              // Update GUI, and replace the '...' with the progress.              OnProgressChange(new ProgressEventArgs(..., "Still working..."));                // Check for abortion at anytime here...              if(AbortOperation)              {                  return;              }          }      }        public event ProgressEventHandler Progress;      private virtual void OnProgressChange(ProgressEventArgs e)      {          var p = Progress;          if (p != null)          {              // Invoke the delegate.          p(e.Progress, e.Message);          }      }  }    public class ProgressEventArgs : EventArgs  {      public int Progress { get; set; }      public string Message { get; set; }      public ProgressEventArgs(int \_progress, string \_message)      {          Progress = \_progress;          Message = \_message;      }  }    public delegate void ProgressEventHandler(int percentProgress, object userState);   |  | | --- | | [share](http://stackoverflow.com/a/680599)|[improve this answer](http://stackoverflow.com/posts/680599/edit) | |