

**Run FINCAD Developer on a Digipede Grid**

Visit [www.fincad.com](http://www.fincad.com) for a free 7-day trial version of FINCAD® Developer, a financial analytics library containing over 1,400 financial functions used for valuing and measuring the risk of financial securities and derivatives.

Source files for this tutorial are available in the zip file at [www.fincad.com](http://www.fincad.com)

**Overview**

The previous newsletter article from June 2006 introduced the idea of grid computing and provided a high level overview of some of the basic concepts involved. It also suggested some reasons that you might choose to use grid computing as a means of improving the performance of web applications. The purpose of this article is to provide a practical example (tutorial) which demonstrates how to develop a simple financial application using FINCAD Developer and then use the Digipede grid solution to run that application on a grid.

Two of the most popular programming languages that are in use today for developing web applications are C# and Java. For this tutorial, we have chosen to use the C# language. This tutorial will describe how to use the trial versions of FINCAD Developer and Digipede together to demonstrate how to run a simple financial application on a grid. Digipede was chosen as the grid solution because it is compatible with the .NET framework and their grid solution is easier to setup than many of the other grid computing products available.

**Requirements**

In order to use the trial versions of Digipede and FINCAD Developer to run this sample program, you need to install separate FINCAD Developer trials on each of the two compute nodes (Agent machines) and you also need to run the Digipede Agent as the same user who installed the FINCAD Developer trials on the compute nodes. These measures are only necessary if you are using the trial versions of the products, as a means of meeting restrictions associated with the trial licensing. The remainder of this tutorial will describe how the sample application works, the steps necessary to install the Digipede server and the steps needed to setup two additional machines with a FINCAD Developer trial and get them to act as Digipede Agents. The Digipede server is made up of a management console application and a database. The Digipede server runs a web service which allows the compute nodes (Agent machines) to login and provide information which enables them to act as Agents. The Digipede server then controls the distribution of calculations to the compute nodes (Agent machines). There are various ways to submit jobs to the server including a Workbench application, a web application and a programming API. For simplicity this tutorial uses the programming API rather than documenting how to use the Workbench application. You can find out more about the Workbench application by going to the Digipede website.

**Source files for the tutorial** (see zip)

There are five files included in the sample application that we will run on the grid.

Simple.csproj	(project file)
AssemblyInfo.cs	(auto-generated file which can be customized to associate file property and version information with the application)
App.ico	(a generic icon that is displayed when the application is running)
ThreadWorker.cs	(calculation jobs which will be distributed to the grid)
Simple.cs	(main program file)

**Simple.csproj**

A project file (sometimes called a makefile) contains a list of files that need to be compiled when building an application. In C# applications, the project file also stores references to other components (DLLs) that the application is dependent on. In this case, the sample program refers to the fcmwl.dll (FINCAD wrapper for C#) and also to a Digipede.Framework library. The sample project file includes a list with the five files above and references to the two DLLs.

**AssemblyInfo.cs**

The AssemblyInfo.cs file contains symbolic information which describes who wrote the application, when it was last modified and various other properties. The content is similar to what is displayed in the File Properties dialog that you may have used in the Windows File Explorer application. C# also provides a mechanism that allows you to store a unique key in the file. This key can be used to help other people who use your application to be certain that the application was developed by you. The process of creating the unique identifier is referred to as signing. There is no need to edit or sign this file before running the sample application on the grid. More information about signing is available in the Visual Studio.NET documentation but it does not become important until you are at the point where you are ready to distribute an application to a large user base.

**App.ico**

The App.ico is a generic icon that is displayed when the sample application is running. You could choose to replace it with your own icon as a way of customizing the sample.

**ThreadWorker.cs and Simple.cs**

The two files from the list above that you really need to pay attention to are ThreadWorker.cs and Simple.cs. These files do the work of connecting to the Digipede server and submitting jobs to the compute nodes (Digipede Agents) using the programming API. You can choose to run the application on the grid or locally (off the grid) by providing the letter 'g' or 'l' as a command line argument (or after the program begins execution by typing the appropriate letter). This makes it easy to see the improved performance that is associated with running the application on the grid and compare it to the normal (local calculation) performance.

**TUTORIAL****Part 1 - How the Sample Program Works****1) Setup a DigipedeClient object**

The first step is to setup a DigipedeClient object which will communicate with the Digipede Server. The command below will create an instance of this object.

```
DigipedeClient dc = new DigipedeClient();
```

The two member functions SetCredentials(userId, password) and SetUrlFromHost(hostname) are used to tell the DigipedeClient object what it needs to know to connect to the server. The values for userId, password and hostname will vary depending on the specifics of how you installed the Digipede server.

## 2) Create a JobTemplate object

The next step is to create a JobTemplate object. The JobTemplate contains information about how to run a job including the files necessary, which class is grid enabled, and how to invoke it. For many cases, the Digipede Framework can determine all of this information from the data type definition for the class that contains the DoWork method. Below is an example which creates a JobTemplate object using the ThreadWorker class.

```
JobTemplate jt =  
JobTemplate.NewWorkerJobTemplate( typeof( ThreadWorker ) );
```

Creating the JobTemplate is made easier if you add a DoWork() method to your worker class. This is what has been done in this tutorial. Otherwise you will need to provide more information about your worker class when setting up the JobTemplate object. Later in the code, we will create objects with the ThreadWorker class. Those objects will be serialized, distributed on the grid and the DoWork method will be invoked on them. This development pattern is called the Worker Model and is one of several that are documented in the Digipede Framework library.

## 3) Create a Job instance

Once you have created a JobTemplate you can now create a Job instance using the Job object by adding the code below.

```
Job j = new Job();
```

You need to add a member function to the instance of the Job object using an anonymous delegate or by creating an instance of the TaskStatusEventHandler class. This basically allows the Job object to respond to a particular event. In the sample application we are interested in when individual tasks that make up our job complete and also when the entire job completes. To create an event driven method for the completion of a task event you can use the code below.

```
        j.TaskFinished += delegate(object sender, TaskStatusEventArgs  
e) {  
            if (e.TaskStatus == TaskStatus.Completed) {  
                ThreadWorker tw = (ThreadWorker)e.Worker;  
                Console.WriteLine(tw.ResultString);  
            } else {  
                Console.WriteLine("Task {0} failed with  
error: {1}", e.TaskId, e.TaskStatusSummary.FailureMessage);  
            }  
        };
```

This event will be needed by the WaitForJob method later so that it can track the tasks as they complete. Similar code to that above is needed to create a method that will be executed when the job completes. You can refer to the sample program to see that statement being used.

Tasks can be associated with an instance of a job object. In the sample application we create a task for each calculation thread using the tasks.add member function of the job object and this function stores a reference for each task in the job object.

#### 4) Submit the job to the Digipede server

To submit the job to the Digipede server you need to use the SubmitJob method from the DigipedeClient class. It is important to put this method call into a try-catch block since it is possible for this method to throw an exception.

The WaitForJob method of the DigipedeClient class is then used to poll for completion of the job.

The ThreadWorker class consists mainly of the FINCAD Developer related code which repeatedly calculates two FINCAD functions. The function call to aaBIN2 takes an iterations parameter which controls the size of the binomial tree used inside the function for calculating the results. This value could be increased if you want to run a computationally intensive test that uses more than two compute nodes. You can also change the constant values for numIterations and numThreads in Simple.cs in order to increase the calculation effort required to complete the job.

When the job is submitted to the grid, the ThreadWorker objects are serialized and distributed to compute nodes on the grid. On those nodes, the DoWork method is invoked and then the objects are reserialized and returned to the application via the TaskFinished delegate. The DoWork method in this example causes two FINCAD function calls to be calculated repeatedly. Different computers on the grid can process the objects in parallel and can accomplish more work in a shorter amount of time than if the work was done on a single machine.

This completes the description of how the program runs FINCAD functions on the grid. The sample program has been kept very straight forward so that everyone can fully understand how the program works. The sample program was implemented using approximately 200 lines of code.

#### Part 2 - Setting up the Digipede Server and Installing FINCAD Developer on the Compute Nodes

All that is required now for you to be able to run this sample program on the grid is a description of the process of setting up the Digipede server and the compute nodes and installing FINCAD Developer on the compute nodes. The remainder of this tutorial will describe how this is done.

##### Digipede server requirements and installation

The server machine that you install Digipede on should be running Windows 2000 Professional (service pack 4), Windows 2000 Server (service pack 4), Windows XP (service pack 2) or Windows Server 2003. You also need to have IIS (Internet Information Server) and Visual Studio.NET 2005 installed. The Server versions of Windows will have IIS installed by default. The install process will deploy SQL Server 2000 Desktop Edition if you don't already have SQL Server installed on your machine. The Developer Edition setup will request for the location of a trial license file. You should make arrangements with Digipede to get a trial license before installing the Developer Edition. You also want to record the server name, user id and password that you enter during the server install. You will need this information later when logging into the DigipedeControl web service to add Digipede Agents to the grid. The default options used in the Digipede install will work for most users or you can talk with your system administrator to ensure that you are using the recommended database security protocols. The setup will ask for the authentication method and as mentioned previously it will ask you to pick a server name, user id and password.

After installing the Digipede server, you need to install the Digipede Agents in order to distribute your application to the grid. You can choose to run one of the Digipede Agents on the same machine as the Digipede server. This allows you to run the sample program on the grid using only two machines and still see a significant performance benefit. The agent machines also need to be running Windows 2000 (sp4), Server 2000 (sp4), XP (sp2) or Server 2003 and have the Microsoft .NET Framework 1.1 SP1 installed. You could choose to install Visual Studio.NET 2005 on both machines as a way to ensure that the Microsoft .NET Framework 1.1 SP1 is available or you can choose to simply install the .NET Framework on its own

(whichever is easier). You should ensure that each Agent machine has at least 100MB of free disk space to ensure that the machine has the capacity to act as an Agent.

### The Digipede Agent

The Digipede Agent consists of three components

1. nisvc.exe – The Digipede Agent Service (which Windows can start automatically)
2. Nicore.exe – starts and monitors processes and communicates with the server.
3. NIUser.exe – provides a user interface for the Agent Service.

The Digipede Agent stores all data files in c:\Documents and Settings\All Users\Application Data\Digipede\Agent. By default, the Agent service starts automatically and logs on to the local system as the Local System account. This account will typically have limited privileges. In order to use the trial versions of Digipede and FINCAD Developer, you need to run the Digipede Agent as the same user who installed the FINCAD Developer trials on the compute nodes. Follow the steps in the installation wizard, and make sure you select "Specific User" for the Agent Service. Enter the Domain, Windows Network user name and password of the correct user. On a development machine where you have installed the FINCAD Developer trial yourself, you can enter your own network domain name, user name and password.

The Digipede Agent is installed using the web browser on your machine through the Digipede Control web page at <http://YourServerName/DigipedeControl>

It is useful to record the server name when running the Digipede server install. If your system administrator did that server installation for you then you may need to check with him to get the correct server name. You also need the User ID and password that was entered during the server installation in order to login to the DigipedeControl web service on the server and when you install the Digipede Agent on a compute node using the DigipedeControl service.

Use the "Click to download the Digipede Agent" link on the home page to download and install the Digipede Agent on each machine that you want to act as a compute node. You also want to install a trial version of FINCAD Developer on each of these machines.

As a first test you may chose to install the Digipede Server and one Digipede Agent on the same machine. Since the grid at this point has only one compute node, the performance should be relatively similar between running the application locally and running it on the grid. It can be advantageous to test the application with one node so that you can ensure that any security or permissions related issues are resolved before distributing the application to a larger set of nodes.

Once you have a machine setup which has the Digipede Server, Digipede Agent and a FINCAD Developer trial installed, you are now almost ready to try running a first test of the grid application. You will need to install the Digipede Framework SDK before you will be able to compile the sample program. The Digipede.Framework class used in the sample program is part of the Framework SDK. Once the Digipede Framework SDK is installed you should be able to compile the sample program. If the program fails to compile then you will want to check that the references to fcmwl.dll and the Digipede.Framework library are both being found. This is done through the Solution Explorer dialog in Visual Studio.NET 2005. Sometimes a problem with the relative path to the library will require that the reference be removed and added again with the correct path. The Add Reference dialog has a browse tab which makes this very easy.

### Run your test on a one node grid

Now that you have the sample application compiling, you should be able to run your first test with a one node grid. Hit F5 to run the application. The application will display a message indicating that you can type 'L' to run the application locally, 'G' to run the application on the grid or 'Q' to quit. If the application displays a "Calculation failed" error message when you run the application on the grid then this is an indication that the security permissions are not setup correctly. The trial version of FINCAD Developer needs to have access to the computers registry to verify the trial license. The permissions associated with the default user (local system account) in the DigipedeControl application may not allow for registry access and this will lead to the function returning an error. This problem only comes up with the trial version of FINCAD Developer since the redistributable DLLs in the purchased version do not have this registry check.

If you run into a permissions issue when running the sample program then you need to follow the steps below to change the user permissions for the user who is running the application on the agent machine. If you did not run into a problem then you can skip the eight steps below.

- 1) Click the start button and open the "Control Panel" application
- 2) Open the "Administrative Tools" program from the Control Panel.
- 3) Open the "Services" program from the Administrative Tools
- 4) Select the "Digipede Agent Service" from the list and then right-click and choose properties
- 5) Select the "Log On" tab
- 6) Change it from "Local System Account" to "This Account"
- 7) Provide a log on name and password. If you are working on your own machine at this point then you can use your own Windows username and password. (Note: The login name may need to contain a domain) i.e. \
- 8) You need to stop and restart the "Digipede Agent Service" using the "Services" program.

When doing performance comparisons it is best to run the application from the command-line rather than running it from the debugger. In my testing the local test case ran much faster from the command line and the grid test ran slightly faster. It is also useful to check the value of the Check-in frequency using the DigipedeControl web application. Login to Digipede control and select the Administration tab. Click on Agent Installation/Administration. Changing the Check-in frequency to 2 seconds (if the value is higher) will improve performance on the grid for smaller test applications. In my tests this reduced the execution time of the grid test by about 10 seconds. The local test took 25 seconds and the grid test took 30 seconds (1 node grid).

### Add additional compute nodes to the grid

Once you have the application running on a one node grid, it is quite easy to add an additional compute node to the grid. Simply follow the steps below

1. Install the Microsoft.NET Framework 1.1 SP1 on the machine if it is not already present.
2. Install a FINCAD Developer trial
3. Use your browser to go to the DigipedeControl web service as before and download and run the Digipede Agent software providing the necessary information about the Digipede server.

You may need to refresh the browser page to see that the new Compute Resource (node) is available.

You are now able to run your application on a two node grid and can compare the performance to the one node grid case. You should see that the calculations now run in approximately half the time. Congratulations!!

My tests showed a calculation time of 18 seconds with a two node grid compared to a local calculation time of 42 seconds. So the application ran in less than half the time on the two node grid. When the second compute node was shutdown (returning to a one node grid), the calculation time increased to 32 seconds which was still faster than the local calculation time in this case. There may be a performance advantage associated with the way that Digipede loads the application in memory or it may be that a service running on my machine during the second test had a negative impact on the performance of the local calculation test. More testing would be required to determine which statement is true. The results show that the two compute node grid was 1.4 times as fast as the best execution time for a local test and 2.3 times as fast as the worst case. This is a significant performance increase given that it is a grid with only two compute nodes.

A white paper, *Running Microsoft Excel on the Digipede Network*, is also available and may be of particular interest to FINCAD XL users who have very large spreadsheets and are looking for creative ways to improve their performance.

#### **Special Thanks**

Thank you to Dan Ciruli (Director of Products) at Digipede who provided invaluable assistance with the development of the sample code in this article and Nathan Trueblood (Vice President of Client Services) who provided an overview of the Digipede grid solution.

#### **Disclaimer**

Your use of the information in this article is at your own risk. The information in this article is provided on an "as is" basis and without any representation, obligation, or warranty from FINCAD of any kind, whether express or implied. We hope that such information will assist you, but it should not be used or relied upon as a substitute for your own independent research.

Copyright © 2006 FinancialCAD Corporation. All rights reserved. FinancialCAD® and FINCAD® are registered trademarks of FinancialCAD Corporation. Other trademarks are the property of their respective holders. This email is for informational purposes only. FinancialCAD Corporation MAKES NO WARRANTIES, EXPRESSED OR IMPLIED, IN THIS SUMMARY.