Alchemi: A .NET Grid Application Framework

Michael R. Head

Grid Computing Research Laboratory
Binghamton University
head@acm.org

April 20, 2007
Outline

1. GRIDBUS
2. Alchemi
3. Sample Application
GRID computing and Distributed Systems (GRIDS) Laboratory
The University of Melbourne, Australia
“The project name GRIDBUS is derived from its research theme: to create next-generation GRID computing and BUSiness technologies that power the emerging eScience and eBusiness applications.”

– http://www.gridbus.org/intro.html
GRIDS Projects

- Grid Economy and Scheduling
- Data Grid Brokering and Scheduling
- Cooperative Coupling of Clusters
- Grid Simulation (GridSim)
- Gridscape II: A Customisable and Pluggable Grid Monitoring Portal and its Integration with Google Maps
Alchemi - Plug & Play Desktop Grid Computing

- .NET-based
- Open Source framework
- for Building Grid Applications
User uses API's and tools to create and run grid applications.

Central controller allocates units of computation to workers and stores results.

Workers execute units of computation and return results to central controller.
Block Diagram

- e-Science Application
- e-Business Application
- e-Engineering Application
- e-Commerce Application

Precompiled executables

Alchemi .NET API (Object-Oriented Grid Programming)

Alchemi Jobs (XML representation)

Alchemi Console Interface

Alchemi Cross-Platform Manager

Grid Threads (.NET objects)

Alchemi Manager

Alchemi Executor

Windows-based machines with .NET Framework

Michael R. Head
Alchemi: A .NET Grid Application Framework
Interface with "Legacy" Grids

Legend
- U: User Node
- M: Manager Node
- E: Executor Node
- X: Cross Platform Manager Node

Custom Grid Middleware

Michael R. Head
Alchemi: A .NET Grid Application Framework
Intra-Application Communication

- Grid Brokering Node
  - ICrossPlatformManager (web service)
  - «SOAP»
- Manager
  - Cross Platform Manager
  - GManager
  - SQL Server / MSDE Database
  - «.NET Remoting»
- Executor
  - Screen Saver
  - «.NET Remoting»
- Owner
  - Grid API
  - «.NET Remoting»
- IManager
  - GManager
  - «.NET Remoting»
- IOwner
  - Grid API
  - «.NET Remoting»

Michael R. Head
Alchemi: A .NET Grid Application Framework
Performance vs. Number of Nodes with a Simple Applications

![Graph showing performance vs. number of nodes with a simple application. The x-axis represents the thread size (no. of digits of Pi) ranging from 1000 to 2200, and the y-axis represents execution time (seconds) ranging from 0 to 450. The graph includes data for 1 to 6 executors, with execution time decreasing as the number of executors increases.](image-url)
Limitations

- **Security**
  - Authorization framework in place
  - Authentication is weak (password only)
  - Passwords stored plaintext in a database
  - Application left to handle malicious Executors

- **Windows-only**
  - CrossPlatformManager is there, but may not be useful
  - Might run with Mono on Unix and OS/X

- **Releases aren’t dependable**
  - 1.0.6 didn’t work for me,
  - 1.0.5 Executor doesn’t shut down
CG Animation Rendering “for SOA”

- DeveloperWorks tutorial
- Using POV-Ray ray tracer
- Render frames in an animation scene into multiple PNG files
  - Uses one executor per frame
- SOAP interface
#declare Font = "C:\\WINDOWS\\Fonts\\arial.ttf"
camera {location<0,0,-10> look_at<0,0,0> }
text {ttf Font "Welcome", 0.25,<0,0,0>  
    pigment {rgbf<0,1,0,0.5>}
    translate <0.66,0,0>
    rotate <clock*720,clock*360,0>}</
text {ttf Font "To POV-Ray", 0.25,0  
    translate <0,-1,0>  
    pigment {rgbf<1,0,0,0.5>}
    rotate <clock*-720,clock*360>}</
sphere {<2.5-clock*5,0.5,-5>, 0.25  
    pigment{ rgbf<0,0,1,0.25> } }
light_source { <0,2,10>,  rgb<0,0,1> }  
light_source { <0,-2,-10>,  rgb<1,1,0> }
Frame One of “Hello” Scene

Welcome To POV-Ray
Frame One of “Hello” Scene
Application setup code

```csharp
GApplication GridApp = new GApplication();
GridApp.ApplicationName = "POVRay_Render_Grid_Application";
GConnection gc = new GConnection(hostname, port, username, password);
GridApp.Connection = gc;
GridApp.Manifest.Add(new ModuleDependency(
    typeof(SceneRenderer).Module));
GridApp.ThreadFinish +=
    new GThreadFinish(GridApp_ThreadFinish);
GridApp.ApplicationFinish +=
    new GApplicationFinish(GridApp_ApplicationFinish);
```
GridThread

[Serializable]
public class RenderThread : GThread
{
    // Serializable properties
    // ...

    public override void Start()
    {
        string povFileName = Path.GetTempFileName();
        File.WriteAllBytes(povFileName, sceneBytes);
        string pngFileName = Path.ChangeExtension(povFileName, ".png");
        // Render input file into output file
        // ...
        outputImage = File.ReadAllBytes(pngFileName);
        File.Delete(povFileName);
        File.Delete(pngFileName);
    }
}
byte[] sceneBytes = File.ReadAllBytes(sceneFileName);

// Create a RenderThread for each frame and dispatch them

for (int i = 0; i <= nFrames; i++)
{
    GridApp.Threads.Add(new RenderThread(sceneBytes, i, nFrames, xResolution, yResolution, megapovLocation));
}

GridApp.Start();

// Will wait here until application is complete

while (GridApp.Running)
{
    System.Threading.Thread.Sleep(2000);
}
Delegates

```csharp
void GridApp_ApplicationFinish() { }

void GridApp_ThreadFinish(GThread thread)
{
    RenderThread rt = (RenderThread)thread;
    string filename = String.Format("output{0:D4}.png", rt.FrameNumber);
    DirectoryInfo info = new DirectoryInfo(OutputDirectory);
    if (!info.Exists)
    {
        info.Create();
    }
    File.WriteAllBytes(Path.Combine(OutputDirectory, filename), rt.OutputImageBytes);
}
```
```csharp
[WebMethod]
public string RenderSceneToFolder(
    string sceneFilenameOnServer,
    int numberOfFrames,
    int xResolution, int yResolution)
{
    SceneRenderer renderer = new SceneRenderer();
    // Process parameters
    // ...
    renderer.RenderScene(/* .. Parameters .. */);
    string baseUrl = Context.Request.Url.GetLeftPart(UriPartial.Authority);
    Uri result = new Uri(new Uri(baseUrl),
    return result.ToString();
}
```
Gridbus

Alchemi

Sample Application