

Bayanihan Computing .NET

A generic framework
for volunteer computing

17 October 2001
Microsoft Asia Student .NET Competition
Seoul, Korea
(This document updated on 20 October 2001.)

Ateneo de Manila University
Philippines
Chua, Sandra Jean
Echevarria, Paul
Mendoza, Joey
Santos, Russelle
Tan, Stanley

Faculty mentors:
Sarmenta, Luis, Ph.D.
Cañimo, Roland (alternate)

Abstract

BayanihanComputing.NET is a generic framework for volunteer computing, that allows you to quickly and easily tap the power of networked computers to perform complex calculations much faster than a single computer, or even a supercomputer, can. BayanihanComputing.NET is the first system in the world to allow programmers to write their own volunteer computing applications with the convenience, flexibility, and power of Microsoft's .NET technologies and tools. It is also the first system in the world to use XML web services to offer "computation web services" that allow programmers to easily tap the power of volunteer computing networks through simple method calls in their .NET programs. BayanihanComputing.NET brings something that no one has offered before: supercomputing power that you can access anytime, anywhere, and on any device.

What is Bayanihan?

In the Philippines, we have a tradition called "bayanihan." When somebody needs to move a house, the whole village gets together and carries the house. Thus, the word "bayanihan" means the spirit of working together to achieve something we can't do alone.

BayanihanComputing.NET makes it possible to achieve Bayanihan on the network. Through the ideas of volunteer computing and computation web services, BayanihanComputing.NET makes it easy for computers to work together and solve problems many times faster than a single computer or even a supercomputer - can.



painting by Joselito Barcelona (1993)

BayanihanComputing.NET

In school ...

Tap school laboratories and outside volunteers to perform large-scale research projects such as

- protein folding, biomedical research
- particle simulations, modelling
- statistical analysis and regression

... and many other projects that would have required a supercomputer. Best of all, you use existing resources more efficiently!

... at work ...

Maximize return on investment of office computers by using them to perform data analysis, trend prediction and other complex tasks much faster than any one computer can. Save money and time by achieving supercomputer power at personal computer price.

... and even at home!

Contribute your computing power to a large-scale project like searching for a cure for cancer. Volunteer your computer quickly and easily using the BayanihanComputing.NET volunteer client. You might even earn money!

Basic Concepts

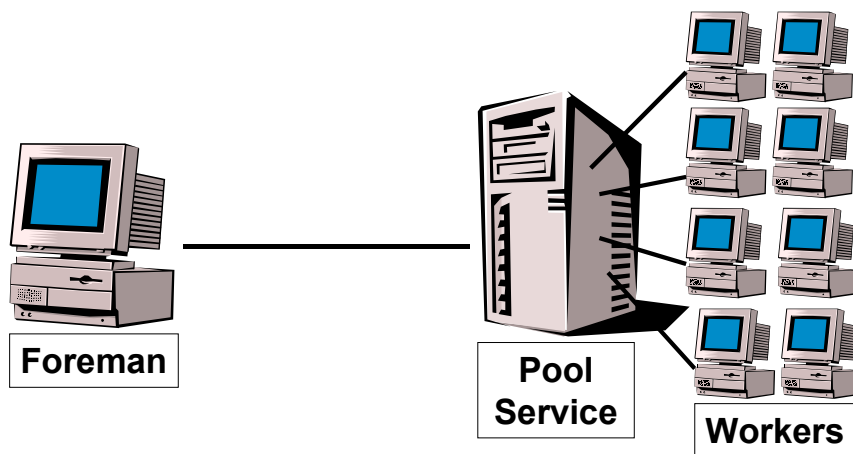
Idea #1: Volunteer Computing

The idea behind "volunteer computing" is to allow people to pool together the idle computing power of computers on the Internet. By letting computers work together on different parts of the same computationally intensive problem, we can solve the problem much faster than possible on a single computer.

Other volunteer computing projects like SETI@home have reached well over 12 trillion floating point operations per second - tens of thousands of times faster than a single desktop PC, and even faster than expensive supercomputers that cost hundreds of millions of dollars each.

BayanihanComputing.NET takes the idea of volunteer computing even further by providing a *generic* framework which is not limited to specific applications like SETI@home, but allows programmers to write their own applications.

As shown in the figure, we provide a PoolService web service that harnesses the power of the volunteer worker machines. All a programmer needs to do is to create a Foreman application that divides the computation into separate work packets that the PoolService can distribute to the workers.



Idea #2: Computation Web Services

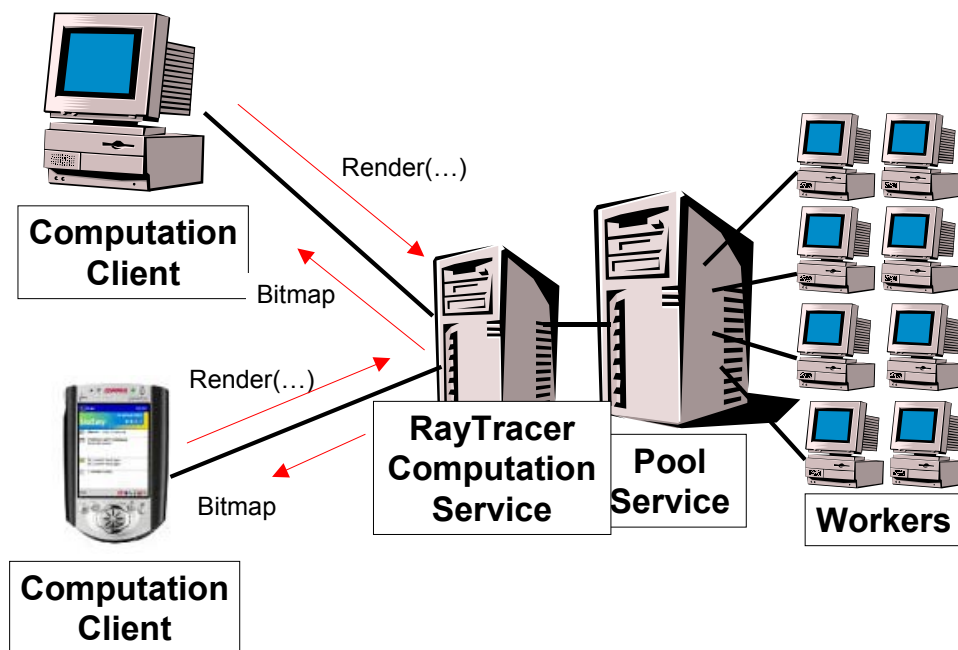
We make tapping into the power of volunteer computing even easier through the new idea of "computation web services".

A computation web service is a web service that allows programmers to access the supercomputing power of the volunteer network without even knowing that they are using a supercomputing resource.

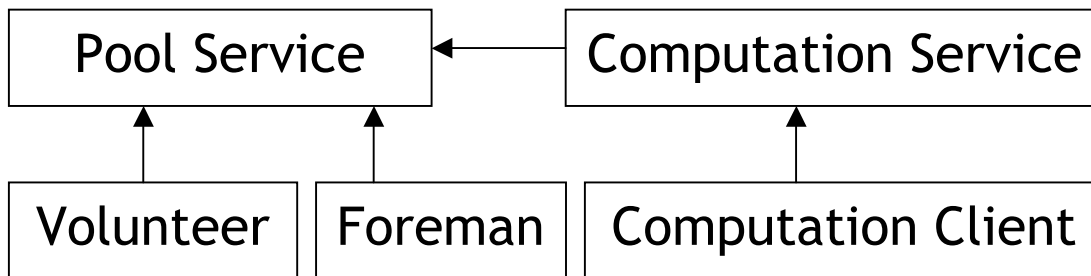
For example, if a film production company needs to have 3D scenes rendered for their movies, they only need to write a simple "computation client" program that calls the Render function on the RayTracer computation web service. The RayTracer computation web service takes care of splitting up the scene into different work objects and passing them to the PoolService, which distributes the work to the workers. When the work is all done, the RayTracer then returns the rendered scene to the computation client in the form of a bitmap.

In this way, computation web services make accessing supercomputing power as easy as a function call. In fact, it is so easy that you can do it on a Pocket PC.

The end result: supercomputing power that you can access anytime, anywhere, and on any device.



System Diagram



Components	
Pool Service	Web service that manages tasks and results from different volunteers
Volunteer	Contributes processing power
Foreman	Creates tasks and asynchronously displays results
Computation Service	Web service that provides a simple interface to a complex, Bayanihan-backed computation
Computation Client	Synchronously invokes computation service

Comparisons

Other projects	BayanihanComputing.NET
Limited to a few platforms	Runs on any SOAP or .NET machine
Limited to a particular language	Write in any combination of .NET languages you want: C++, C#, VB and more, all in the same project!
Specialized libraries/applications	Generic framework, good for any application
Most allow you only to volunteer	Volunteer for it, program for it, use it!
Complex programmer interface	Simple computation service

.NET Technology Highlights

Using the Best of .NET

BayanihanComputing.NET is not just a web application. We don't just do web pages that you can do with existing technologies like HTML and ASP. We make use of the newest technologies only available in .NET to do new things that you could not do before.

We've made web services not just a way for people to share information, but also a way for people to share their computers' power with each other and the world.

Here is a list of just some of the .NET technologies that we used:

Web services

Empower your applications with a few web method calls. Provide a simple interface to complex calculations. Consume web services not just for data access, but also for compute power. Coordinate between multiple volunteers with a single pool service.

Web applications

Take advantage of BayanihanComputing.NET's power through any web browser, thanks to ASP.NET.

Simple object access protocol

Use BayanihanComputing.NET even behind firewalls. Call web methods from any SOAP-aware program.

Serialization

Base64 encoding and byte serialization transmits data through the wire, allowing data to be deserialized into objects in any language.

Common language runtime

Develop Bayanihan solutions in any combination of .NET languages. Deploy anywhere with a .NET runtime. Use existing skills - no need to learn new languages or complicated libraries

Assemblies

Package project-specific code into assemblies. Run many different projects at the same time.

Automatic versioning

Volunteer clients automatically download latest code if necessary, thanks to DLL versioning.

Delegates

Display results as they arrive or be notified when all results have been received through the use of delegates. Enjoy the flexibility of synchronous/asynchronous invocation, depending on the needs of your application.

XML documentation and VS.NET

Develop Bayanihan applications quickly using the Visual Studio .NET IDE. Select the service you want to use, and let XML documentation and IntelliSense™ guide you through the process. Take advantage of documentation at your fingertips!

ADO.NET and SQL Server 2000

Store usage statistics and other information in a database for easy search and retrieval.

How to use BayanihanComputing.NET

To run a Volunteer:

1. Download the generic client application from the BayanihanComputing.NET site.
2. Run the client.

Clients automatically download necessary components as tasks are received, so it can run quietly in the background.

To use a foreman:

1. Run the foreman.
2. Specify whatever parameters are necessary.

To use a computation web application:

1. View the web application.
 2. Supply all the necessary parameters and submit the form.
- Results will be returned to you after a short delay.

To create a program that uses a computation web service:

1. Add a web reference to the computation web service.
2. Invoke the desired web method, which will return the complete result set.
3. Catch and deal with timeouts.

To create a foreman:

1. Read the technical documentation for the Bayanihan.NET framework.
2. Write the necessary data structures and application-specific code.
3. Package all application-specific code needed by volunteers in an assembly.
4. Upload the assembly to the machine with the Pool Service.
5. Write a foreman program that communicates with the Pool Service and does the following:
 - a. Create tasks and add them to the pool.
 - b. Provide two delegates: a ResultListener for handling results as they arrive, and a DoneListener for performing final computations after all the results have arrived.
6. Run the program.

To create a computation web service:

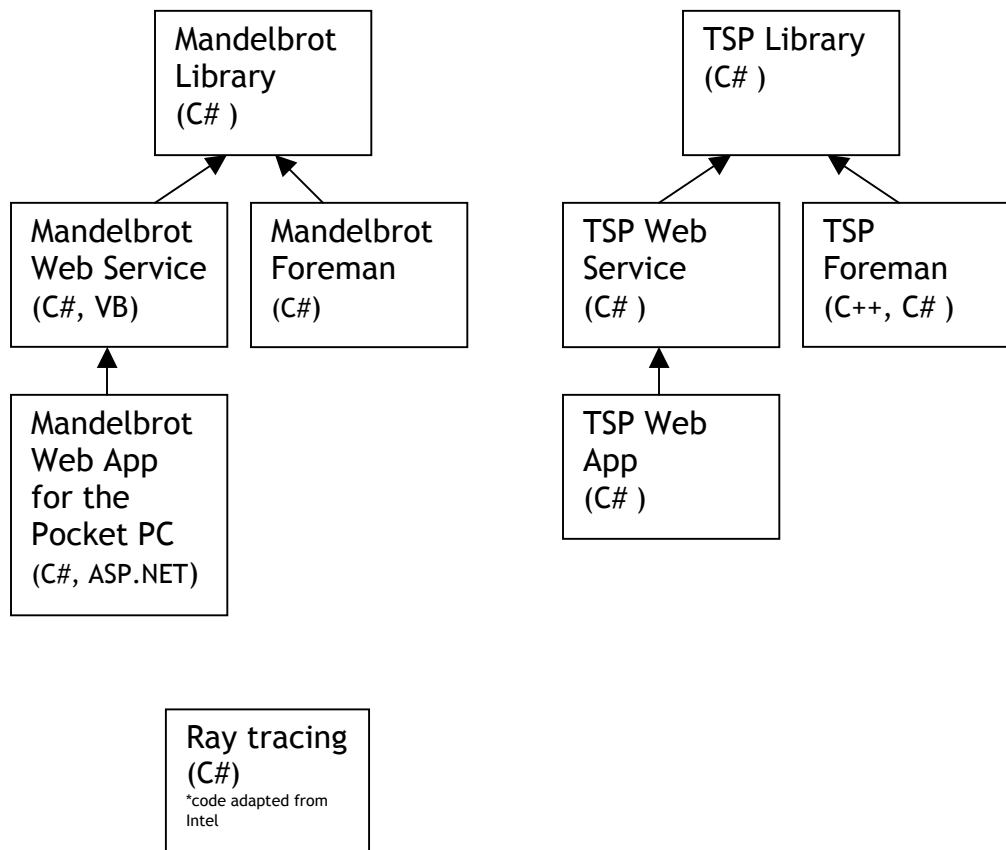
1. Create a foreman that collects all the results (in the ResultListener), and packages them into a single final result (in the DoneListener).
2. Define a web method that:
 - a. takes parameters from the client and passes these to the foreman
 - b. waits for the foreman to return the packaged result
 - c. returns the result to the client

Demo Applications

To demonstrate the flexibility of BayanihanComputing.NET, we have implemented several demonstrations, as shown below. BayanihanComputing.NET is not limited to these applications, however, and can be used for many different applications.

We show how easy it is for programmers to write their own applications, and even port existing ones, by porting Intel's RayTracing demo into BayanihanComputing.NET.

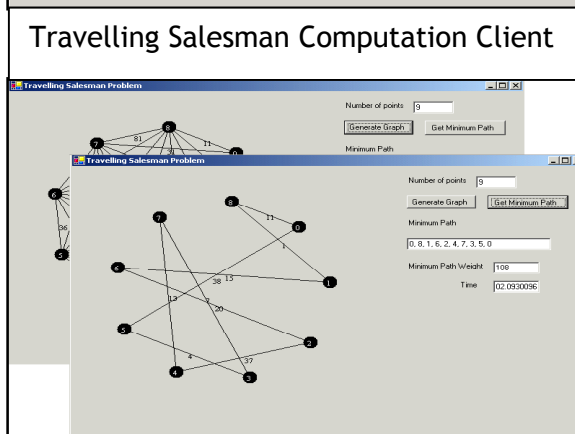
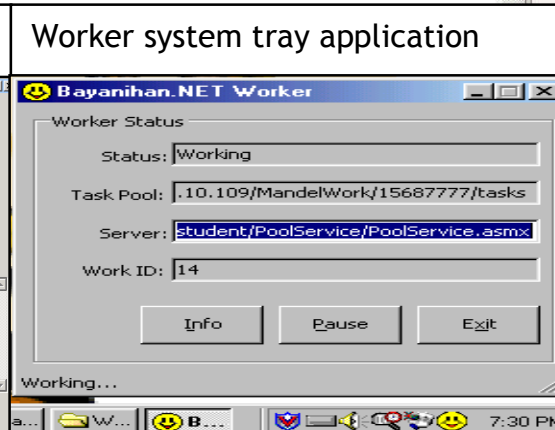
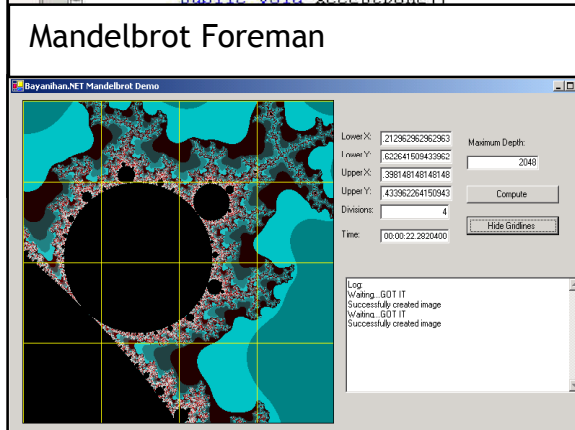
There is no limit to the applications that programmers can writing using BayanihanComputing.NET.



Screenshots

Mandelbrot Foreman source code

```
Console.WriteLine("Each block will be {0} x {1}", div_x, div_y);
int z = -1;
for(int x = 0; x < width; x += div_width)
{
    for(int y = 0; y < height; y += div_height)
    {
        work[++z] = new MandelLib.MandelWork(
            xCoor(x), yCoor(y),
            xCoor(x + div_width), yCoor(y + div_height),
            x, y, div_width, div_height,
            depth, colors.Length);
    }
}
base.AddWork(work);
GetResults(new ResultListener(AcceptResult),
    new DoneListener(AcceptDone), false);
public void AcceptDone()
```



XML Documentation

Code Comment Web Report

WorkerLib

- BlankWriter
- Worker
- WorkerInfo

WorkerLib.PoolService

WorkerLib.Worker Class

The Worker class is a generic worker that accepts "tasks" from a Pool Web Service. The generic Worker is able to process concrete work objects. Each work object implements the DoWork() method on the work object. Each work object implements the method returns the result as an object. Hence, the Worker is able to should have a DoWork() method.

Access: Public

Base Classes: Object

Members	Description
poolService	The proxy object to the Pool Web Service.
ServerUri	Gets or sets the URL of the Pool Web Service.
taskPool	The name of the task pool currently not assigned to any task pool at the time.

BayanihanComputing.NET
Generic framework for volunteer computing

Further References

BayanihanComputing.NET
<http://bayanihancomputing.net>, bayanihan@ateneo.edu
Project website

Project Bayanihan at MIT
<http://www.cag.lcs.mit.edu/bayanihan/>
The original Bayanihan Java-based framework for volunteer computing. We drew many high-level concepts from this project, but wrote our own system using .NET from scratch.

Intel Peer2Peer SharedCycles demo
<http://www.gotdotnet.com/p2p/>
Demonstrates how to do peer-to-peer cycle sharing on .NET. This example is application-specific (limited to ray-tracing), and requires workers to run IIS. We used it as reference for learning about .NET technologies, but wrote our own original system that is generic, doesn't require volunteers to have IIS, and is fault-tolerant. We then ported the ray tracing demo to our system.

Internet-based distributed computing projects
<http://www.aspenleaf.com/distributed/>
Survey of current distributed computing projects

SETI@home
<http://setiathome.ssl.berkeley.edu>
Popular volunteer computing project

distributed.net
<http://www.distributed.net>
Popular volunteer computing project