Using Vsync from C++ or C or Java

Ken Birman

Notes for C++ users

The Vsync system can be used from C++ with the same ease as from C#, and in fact in nearly the same style of coding. There really isn't any special preparation needed, and the full API is supported. By loading the ".chm" help file from the documentation page and then opening it with your browser (try Internet Explorer if you have a problem with other browsers; not all support "compiled HTML"), you'll see examples of precisely what notation to use. At that point every example in the Vsync manual can trivially be coded in C++. There shouldn't be any performance change relative to C#.

On Linux, C++ users would need to compile their C++ programs using the Mono C++ compiler. This is because the Vsync library is a .NET dll, and can only be directly accessed by other .NET programs. When you compile C++ with compilers other than the Mono C++ compiler, you end up with an executable, but the code isn't in the .NET common language runtime's byte code representation, and direct use of .NET libraries is extremely difficult. In contrast, when you compile with the Mono C++ compiler, which is a free and very robust download from Microsoft, you end up with a CLR version of your program that can access any .NET feature or dll, including Vsync, simply by adding the library as a "Reference" to your project and then accessing it at runtime just as you would access any other library.

To access Vsync from a C++ program (or one in C, Java, etc) that isn't compiled using Mono and hence isn't running as .NET managed code, we actually recommend a two step procedure:

1. First, you should build a small stand-alone service that was built with C#, .NET C++, VB or IronPython. For example, perhaps your service will implement the Memcached API (get and put for (key,value) pairs).
2. Designate the API for your service as one that can be accessed "remotely". In .NET this is a simple thing to do and involves pointing to the class and filling out a small properties sheet to tell the system who can access the service and what its name should be.
3. Install and launch the service.
4. Now you can access it from the same or different machines using the .NET remote invocation features. If you are on the same machine, performance will be nearly as good as for a direct library call to a method in your own address space. When calling from a different machine, some overheads apply, but they will be low if the programs are near one-another.

Notes for C and Java Programmers

At the present time, Vsync can be used directly from J# on a Windows .NET platform in exactly the same way that the system is used from C#. Everything is fully supported. But J# is not a standard Java and in particular, you will not be using the standard JDK (the runtime methods are different).

For a C program, or a Java program running on Linux or on Windows but outside of the .NET framework, we currently do not have any direct options that would work safely. Instead, we recommend that you read the suggestions about creating a small stand-alone "service" and then employing remote method invocation (even from the same machine) to access the methods in that service.

We have considered building a small service that would just expose all of the Vsync functionality in this same remote access manner, but doing so is tricky because remote method invocations are difficult to employ in settings that involve threads, objects and high speed delivery of events. For these reasons, for now, we have no plans to actually build a service of this kind. But we think that a remote service that offers an API focused on some application-oriented functionality, of the kinds mentioned above, shouldn't be hard for most people to build and at that point, your C or Java (or whatever) application can access Vsync "mechanisms" with relative ease.

Contact us if you have suggestions

Ken Birman, ken@cs.cornell.edu

http://www.cs.cornell.edu/ken

http://Vsync.codeplex.com