Building plasma simulation infrastructure with Microsoft .NET

Steven R. Lantz, *Cornell Theory Center, Cornell University*

One of the most onerous tasks associated with constructing large-scale plasma simulations has nothing to do with physical modeling, but rather with data handling. The well-known glut of numbers becomes particularly acute when the many gigabytes produced by a three-dimensional simulation are linked into a visualization system, even though this step is absolutely crucial if meaningful interpretations are to be made of the results. This talk describes an attempt to create the necessary scientific data management tools by leveraging commercial software originally targeted for e-business applications and databases.

Microsoft .NET is an emerging platform for client-server and server-to-server software that possesses several attractive features for assisting simulation science. First, the .NET Framework provides the low-level middleware needed for connecting servers (such as supercomputers) with databases and remote clients (on workstations). By design, the .NET "plumbing" is hidden from developers by encapsulating it in components that are accessible through object-oriented interfaces. Under these wrappers, the default middleware is built around Internet standards such as XML and IP, but it can also be customized as needed. Second, .NET offers some degree of programming language independence, since the namespaces defining its objects and interfaces are available in multiple languages. Different .NET components are able to talk to each other regardless of their origin because the code is first compiled into CIL, the common intermediate language that is just-in-time compiled down to machine language only at runtime. Third, in the Windows OS there exists an IDE or integrated development environment, Visual Studio .NET, that enables RAD or rapid application development of .NET code—flagging syntax errors on the fly, offering drop-down lists of valid method names, etc. Finally, .NET integrates particularly well with SQL Server on the one hand and ordinary Web browsers on the other, making it very well suited as an intermediary for desktop access to extremely large databases.

The talk will focus on two applications of .NET technology to the infrastructure surrounding a 3D MHD research code used for studying solar convection. The key concept, which has been realized in the Fusion Grid and elsewhere, is to store and retrieve simulation data through a database like SQL Server rather than a traditional file system. (Reportedly, in future versions of Microsoft Windows, the distinction between the two storage mechanisms will be blurred in any case.) This enables researchers to interact with their data in new and creative ways, especially if they can easily incorporate database queries into special-purpose applications. What .NET does is to facilitate the development of such tools: for example, one might custom-design a GUI for building valid SQL queries using Win Forms or Web Forms, then have this GUI pass its output request to SQL Server through ADO.NET. In our work in progress, "go-between" code of this sort is being implemented in two places. As the parallel simulation runs, it enlists SQL Server to act as a kind of parallel (or at least asynchronous) file system. Domain decomposition maps rather naturally into per-node transactions with the database involving subsets of the full dataset. Similarly, a general-purpose visualization tool called OpenDX is instrumented to obtain its input via SQL Server queries instead of files. Specific data of interest are then transferred in real time according to parameters (such as mouse position) received from the user via the OpenDX GUI. The translation from OpenDX user input to SQL is embedded in reasonably straightforward C# code that is invisible to the user. Other possibilities for this kind of technology will be discussed.