WebFlow - High-Level Programming Environment and Visual Authoring Toolkit for High Performance Distributed Computing

Erol Akarsu
Northeast Parallel Architectures Center
at Syracuse University
Syracuse, NY

akarsu@npac.syr.edu

Geoffrey C. Fox
Northeast Parallel Architectures Center
at Syracuse University
Syracuse, NY

gcf@npac.syr.edu
http://www.npac.syr.edu/

Wojtek Furmanski
Northeast Parallel Architectures Center
at Syracuse University
Syracuse, NY

furm@npac.syr.edu

Tomasz Haupt
Northeast Parallel Architecture Center
at Syracuse University
Syracuse, NY

haupt@npac.syr.edu
http:www.npac.syr.edu/users/haupt/homepage/

Abstract:

We developed a platform independent, three-tier system, called WebFlow. The visual authoring tools implemented in the front end integrated with the middle tier network of servers based on the industry standards and following distributed object paradigm, facilitate seamless integration of commodity software components. We add high performance to commodity systems using GLOBUS metacomputing toolkit as the backend. We have explained these ideas in general before, and here for the first time we describe a fully operational example which is expected to be deployed in an NCSA Alliance Grand Challenge.

Keywords:

WebFlow, Metacomputing, Globus, Visual Authoring Tools, Distributed Objects, Java, Web Server, Three Tier Architecture, HPCC

Introduction

Programming tools that are simultaneously sustainable, highly functional, robust and easy to use have been hard to come by in the HPCC arena. This is partially due to the difficulty in developing sophisticated customized systems for what is a relatively small part of the worldwide computing enterprise. Thus we have developed a new strategy - termed HPcc: High Performance Commodity Computing[1] - which builds HPCC programming tools on top of the remarkable new software infrastructure being built for the commercial web and distributed object areas. This leverage of a huge industry investment naturally delivers tools with the desired properties with the one (albeit critical) exception that high performance is not guaranteed. Our approach automatically gives the user access to the full range of commercial capabilities (e.g. databases and compute servers), pervasive access from all platforms and natural incremental enhancement as the industry software juggernaut continues to deliver software systems of rapidly increasing power. We add high performance to commodity systems using a multi tier architecture with GLOBUS[2] metacomputing toolkit as the backend of a middle tier of commodity web and object servers. We have explained these ideas in general before[1],[3] and here for the first time we describe a fully operational example which is expected to be deployed in an NCSA Alliance Grand Challenge.

This approach was not possible a few years ago when the enterprise computing was still mainly client-server (2 tier) and based on custom expensive solutions such as proprietary TP Monitors. The onset of the Web and the associated Intranets accelerated the development of scalable and open 3-tier standards such as CORBA[4], DCOM[5] and Enterprise JavaBeans (EJB)[6]. We can therefore prototype now a dedicated and advanced and yet commercial quality HPDC system for HPCC applications by integrating open commodity standards for distributed enterprise

computing with traditional (such as MPI or HPF) and emerging (GLOBUS) HPCC infrastructure, which is optimized for performance.

Figure 1: Top level view of the WebFlow environment.

Our research addresses needs for high level programming environments and tools to support distance computing on heterogeneous distributed commodity platforms and high-speed networks, spanning across labs and facilities. More specifically, we are developing WebFlow - a scalable, high level, commodity standards based HPDC system that integrates (c.f. Fig. 1):

- High-level front-ends for visual programming, steering, run-time data analysis and visualization, and collaboration built on top of the Web and OO commodity standards (Tier 1).
- Distributed object-based, scalable, and reusable Web server and Object broker Middleware (Tier 2) High Performance Backend implemented using the metacomputing toolkit of GLOBUS (Tier 3)

Note this can be applied to either parallel or metacomputing applications, and provides a uniform cross platform high level computing enviroment.

WebFlow overview

The visual HPDC framework delivered by this project offers an intuitive Web browser based interface and a uniform point of interactive control for a variety of computational modules and applications, running at various places on different platforms and networks. New applications can be composed dynamically from reusable components just by clicking on visual module icons, dragging them into the active WebFlow editor area, and linking by drawing the required connection lines, as shown in Fig.2. The modules are executed using GLOBUS optimized components combined with the pervasive commodity services where native high performance versions are not available. For instance today one links GLOBUS controlled programs to WebFlow (Java connected) Windows NT and database executables. When GLOBUS gets extended to full PC support, the default WebFlow implementation will be replaced by the high performance code.

Figure2: WebFlow frontend: new applications can be composed dynamically from reusable

components just by clicking on visual module icons, dragging them into the active WebFlow editor area, and linking by drawing the required connection lines.

Individual modules are typically represented by visualization, control, steering or collaboration applets, and the system also offers visual monitoring, debugging and administration of the whole distributed applications and the underlying metacomputing resources. New applications, created within the WebFlow framework will likely follow a natural modular design in which one accumulates a comprehensive problem domain specific module library in the first phase of a project. Then one would explore the computational challenges of the project in a visual interactive mode, trying to compose the optimal solution of a problem in a sequence of on-the-fly trial applications. The scripting capabilities of WebFlow coupled with database support for session journaling will facilitate playback and reconstructing optimal designs discovered during such rapid prototyping sessions.

Our prototype WebFlow middle-tier is given by a mesh of Java enhanced Web Servers, running servlets that manage and coordinate distributed computation. This management is currently implemented in terms of the three servlets: Session Manager, Module Manager, and Connection Manager. These servlets are URL addressable and can offer dynamic information about their services and current state. Each of them can also communicate with each other through sockets. Servlets are persistent and application independent.

Although our protoype implementation of the WebFlow proved to be very successful, we are not satisfied with this to large extend custom solution. Pursuing HPcc goals, we would prefer to base our implementation on the emerging standards for distributed objects, and take the full advantage of the possible leveages realised by employing commercial technologies. Our research led us to the following observations.

While the "Java Platform" or "100% Pure Java" philosophy is being advocated by Sun Microsystems, industry consortium led by the OMG pursues a multi-language approach built around the CORBA model. It has been recently observed that Java and CORBA technologies form a perfect match as two complementary enabling technologies for distributed system engineering. In such a hybrid approach, referred to as Object Web [7], CORBA is offering the base language-independent model for distributed objects and Java offers a language-specific implementation engine for the CORBA brokers and servers.

Meanwhile, other total solution candidates for distributed objects/components are emerging such as DCOM by Microsoft or WOM (Web Object Model) by the World-Wide Web Consortium. However, standards in this area and interoperability patterns between various approaches are still in the early formation stage. A closer inspection of the distributed object/component standard candidates indicates that, while each of the approaches claims to offer the complete solution, each of them in fact excels only in specific selected aspects of the required master framework. Indeed, it seems that WOM is the easiest, DCOM the fastest, pure Java the most elegant and CORBA the most realistic complete solution. Consequently, we plan to adopt CORBA as the base distributed object model to implement next generation of the WebFlow middle-tier.

The module API is very simple. The module implements a specific WebFlow Java interface, metamodule. In practice, the module developer has to implement three methods: initialize, run, and destroy. The initialize method registers the module itself and its ports to the Session

Manager, and establish the communication between itself and its frontent applet - the module controls. The run method implemets the desired functionality of the module, while the destroy method performs clean-up after the processing is completed. In particular, the destroy methods closes all socket connections which are never destroyed by the Java garbage collector.

For a high performance metaapplications more adequate backend solution is needed. As usual we go for a commodity solution. Since commercial solutions are practically nonexistent, in this case we use technology that comes from the academic environment: the metacomputing toolkit of Globus. The Globus toolkit provides all functionality we need. The underlying technology is a high performance communication library: Nexus. MDS (Metacomputing Directory Services) allows for the resource identification, while GRAM (Globus Resource Allocation Manager) provides secure mechanisms to allocate and scheduling of the resources. GASS package (Global Access to the Secondrary Storage) implements high performance, secure data transfer which is augmented with RIO (Remote Input/Output) library that provides access to parallel data file systems.

WebFlow interacts with the Globus via GRAM gatekeeper. A dedicated WebFlow module serves as a proxy of the gatekeeper client, which in turn sends requests to GRAM. Currently, the proxy is implemented using the Java native interface. However, in collaboration with the Globus development team, we are working on a pure Java implementation of the gatekeeper client.

WebFlow test application

As a test application for the WebFlow we selected Quantum Simulations[8]. This application can be characterized as follows. A chain of high performance applications (both commercial packages such as GAUSSIAN or GAMESS or custom developed) is run repeatedly for different data sets. Each application can be run on several different (multiprocessor) platforms, and consequently, input and output files must be moved between machines. Output files are visually inspected by the researcher; if necessary applications are rerun with modified input parameters. The output file of one application in the chain is the input of the next one, after a suitable format conversion. The logical structure of the application is shown in Fig. 3

Figure 3. Logical structure of the Quantum Simulation application

This example mataapplication demonstrates strength of our WebFlow approach. The WebFlow editor provides an intuitive environment to visually compose the chain of data-flow computations from preexisting modules. The modules encapsulate many different classes of applications: from massively parallel to custom developed auxiliary programs to commodity commercial ones (such as DBMS or visualization packages). The seamless integration of such heterogeneous software components is achieved by employing distributed objects technologies in the middle tier. The high performance part of the backend tier in implemented using the GLOBUS toolkit. In particular, we use MDS (metatacomputing directory services) to identify resources, GRAM (globus resource allocation manager) to allocate resources including mutual, SSL based authentication, and GASS (global access to secondary storage) for a high performance data transfer. The high performance part of the backend is augmented with a commodity DBMS (servicing Permanent Object Manager) and LDAP-based custom directory service to maintain geographically distributed data files generated by the Quantum Simulation project. The diagram illustrating the WebFlow implementation of the Quantum Simulation is shown in Fig. 4.

Figure 4 : WebFlow implementation of the Quantum Simulations problem

The WebFlow can be applied to many different applications. At Supercomputing '97[9] we demonstrated two other applications. One is an AVS-like[10] image processing. Here we took advantage of our platform independent design (implemented in Java) to integrate computations running on both UNIX and Windows-NT system. The other employed HPF-based backend. More specifically, we demonstrated the DARP system[11], an integrated environment for compiled and interpreted HPF, wrapped as a WebFlow module.

Future Work

Not all applications follow the data flow paradigm as closely as those described here. Therefore it is necessary to define an interface so that different frontend package can be "plugged in" into the middle-tier, giving the user a chance to use the FrontEnd that best fits the application at hand. The user can use the FrontEnd that best fits the application at hand. Currently we offer visual editors based on GEF[12] and VGJ[13]. In the future, we will add an editor based on the UML[14] standard, and we will provide an API for creating custom ones. The middle-tier is implemented as a network of servlet enabled web servers. Currently we support Java Web Server from SunSoft and public domain Apache. We are evolving towards CORBA (see multiprotocol JWORB[15] server under development at NPAC/Syracuse University) distributed objects as well as the current java wrapped entities which will support the linkage of DCOM and DOM.

Conclusions

To summarize, we developed a platform independent, three-tier system: the visual authoring tools implemented in the front end integrated with the middle tier network of servers based on the industry standards and following distributed object paradigm, facilitate seamless integration of commodity software components. In particular, we use the WebFlow as a high level, visual user interface for GLOBUS. This not only makes construction of a meta-application much easier task for an end user, but also allows to combine this state of art HPCC environment with commercial software, including packages available only on Intel-based personal computers.

References

1. G. Fox and W. Furmanski, "HPcc as High Performance Comodity Computing", chapter for the "Building National Grid" book by I. Foster and C. Kesselman, http://www.npac.syr.edu/uses/gcf/HPcc/HPcc.html

2. I. Foster and C. Kesselman, "Globus: A Metacomputing Infrastructure Toolkit", International Journal of Supercomputing Applications, 1997.
See also Globus Home Page: http://www.globus.org
3. D. Bhatia, V. Burzevski, M. Camuseva, G. C. Fox, W. Furmanski and G. Premchandran, "WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing", Concurrency: Practice and Experience, Vol. 9 (6), pp. 555-577, June 1997. See also WebFlow Home Page at NPAC: http://osprey7.npac.syr.edu:1998/iwt98/products/webflow.
4. CORBA - OMG Home Page http://www.omg.org
5. COM Home Page http://www.microsoft.com/com
6. Enterprise JavaBeans http://java.sun.com/products/ejb/
7. Client/Server Programming with Java and Corba 2nd. Ed. by Robert Orfali and Dan Harkey, February'98, ISBN:047124578X

8. Quantum Simulations, http://www.ncsa.uiuc.edu/Apps/CMP/cmp-homepage.html
9. G. Fox, W. Furmanski and T. Haupt, SC97 handout: High Performance Commodity Computing (HPcc), http://www.npac.syr.edu/users/haupt/SC97/HPccdemos.html

10. Advanced Visualization System, http://www.avs.com/

11. E. Akarsu, G. Fox, T. Haupt, DARP: Data Analysis and Rapid Prototyping Environment for Distribute High Performance Computations, Home Page http://www.npac.syr.edu/projects/hpfi/

12. GEF: The Graph Editing Framework home page: http://www.ics.uci.edu/pub/arch/gef/
13. VGJ, Visualizing Graphs with Java home page: http://www.eng.auburn.edu/department/cse/research/graph_drawing/graph_drawing.html

14. UML Home Page http://www.rational.com/uml
15. JWORB Project Home Page http://osprey7.npac.syr.edu:1998/iwt98/projects/worb, see also G. C. Fox, W. Furmanski and H. T. Ozdemir, "JWORB (Java Web Object Request Broker) based Scalable Middleware for Commodity Based HPDC", submitted to HPDC98.