JWORB - Java Web Object Request Broker for Commodity Software based Visual Dataflow Metacomputing Programming Environment

Geoffrey C. Fox, Wojtek Furmanski and Hasan T. Ozdemir
Northeast Parallel Architectures Center, Syracuse University, Syracuse NY
{gcf,furm,timucin}@npac.syr.edu

Submitted for HPDC7

1 Introduction

Programming environments and tools that are simultaneously sustainable, highly functional, robust and easy to use have been hard to come by in the HPDC area. This is partially due to the difficulty in developing sophisticated customized systems for what is a relatively small part of the worldwide computing enterprise. As the commodity software becomes naturally distributed with the onset of Web and Intranets, we observe now a new trend in the HPDC community [1,8,12] to base high performance computing on the modern enterprise computing technologies.

This approach was not possible still a few years ago when: a) the enterprise computing was still mainly custom TP Monitors based client-server (2-tier) only; and b) Web computational extensions such as CGI were too naive to grant high performance and/or quality of service. However, the ongoing covergence of Web and Enterprise Computing accelerates the development of scalable (multi-server)and open 3-tier standards such as CORBA, DCOM or Enterprise JavaBeans and our vision becomes quickly a reality these days. We can now start prototyping both user friendly and powerful HPDC systems by merging the commodity technologies in tier 1 (front-end) and tier 2 (application logic) with the high performance technologies in tier 3 (legacy software).

Initial Web/Commodity based HPDC prototypes appeared in the pure Java domain. We summarize here our experience with one such early system, WebFlow [2-5] for visual dataflow metacomputing developed at NPAC in '96/'97. Basically, we found Java to be a very useful framework for middleware development whereas both front-end and back-end development require both multi-platform and multi-language support - which led us in a natural way to the CORBA model. We are now building and reporting here on the current status of our new CORBA based WebFlow with the middleware/bus layer given by a mesh of Java Web Object Request Brokers (JWORB).

JWORB [9] is a multi-protocol Java server under development at NPAC, currently capable of handling HTTP and IIOP prototocols. Hence, JWORB can be viewed as a Java based Web Server which can also act as a CORBA broker. We present here JWORB rationale, architecture, implementation status, results of early performance measurements and we illustrate its role in the new WebFlow system under development.

2 WebFlow Current State

WebFlow is a 3-tier distributed visual dataflow model. Layers 1 and 2 are written in Java and provided as part of the release together with a set of trial/demo modules. Layer 3 is left open for further specifications and refinements which allows us now to wrap any backend code as webflow module and link it to its tier 2 Java proxy via a customized socket connection. WebFlow middleware is given by a mesh of Web servers written in Java and hence offering a natural computational extensibility model via the dynamic properties of the underlying JavaVM. In '96, we analyzed two natural standard candidates in this area: Jeeves server by JavaSoft (later renamed as Java Web Server (JWS)) and Jigsaw server by the World-Wide Web Consortium. We found light-weight Servlets in Jeeves to offer more attractive dynamic extensibility model than more heavy-weight Resources in Jigsaw and we selected Jeeves/JWS as a base of WebFlow middleware.

Each JWS node of WebFlow manages its portion of a distributed computation in terms of three management servlets: Session Manager, Module Manager and Connection Manager. Each Session Manager exports the externally visible URL as a WebFlow entry point and it controls the concurrent user sessions on this server. Connection Manager handles the module connection requests. Module Manager is responsible for loading the WebFlow modules to the Java VM and controling module life cycle (init, run, destroy).

WebFlow Module is a Java Object which implements webflow.backend.Module interface. This interface contains three methods: a) init() - called after the module is created by the Module Manager; b) run() - called by the Module Manager when the module is fully instantiated, connected and ready to start the regular operation as a distributed dataflow node; c) destroy() - called when the Module Manager needs to stop the execution and to release the module object.

WebFlow front-end is given by a Java applet, served by any of the JWS Session Managers and offering the visual interactive tool for dataflow authoring. In the current prototype, we based our front-end on the GEF (Graph Editing Framework) package from UCI [6] and we suitably extended it by building the URL and socket based communication between the applet and the JWS Session Manager servlet.

So far, we developed a set of simple proof-of-the-concept backend modules testing various aspects of the system including: a) selected HPCC/HPF simulations; b) AVS-style library of image processing filters; c) simple collaboratory sessions; d) interfaces to SciVis packages [21]; e) dynamic generation of breakpoint modules during a visual debugging session of an HPF application using the NPAC DARP (Data Analysis and Rapid Prototyping) [10] package support for the HPF environment and demonstrated at Supercomputing'97 [11].

3 Emerging Object Web Technologies

Since the fall '96 when the WebFlow prototype design was finalized, we were witnessing several new developments in the Web/Commodity software standards which offer now more promising and robust framework for the WebFlow production version. Most notably, the distributed object and/or component technologies are rapidly growing now and we decided to perform a major upgrade of the WebFlow design that converts it from the previous custom componentware model to the new emergent standards based model.

Selecting a specific direction in the exploding field of distributed object and component technolo-

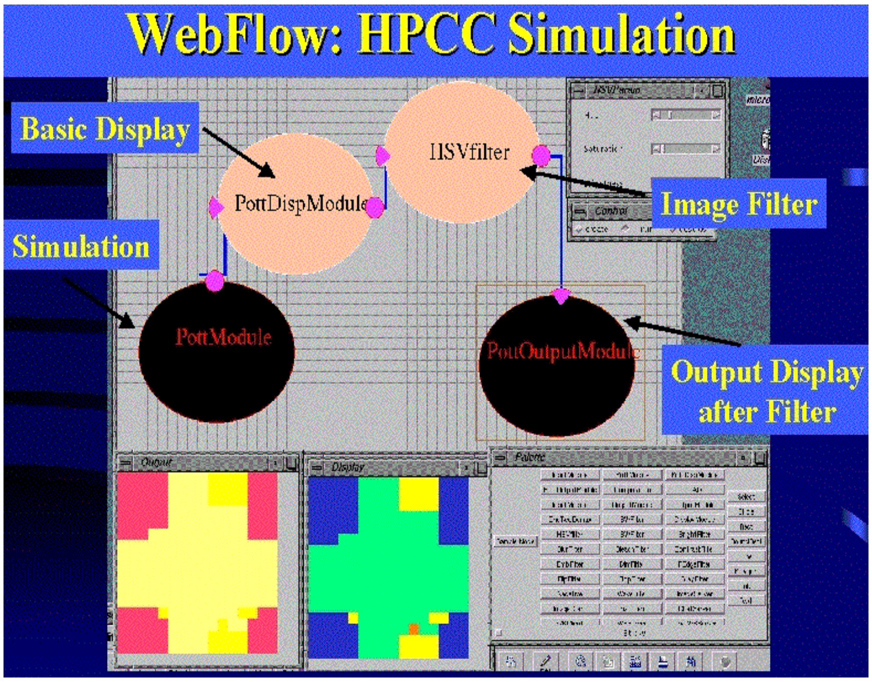

Figure 1: Sample WebFlow Application: HPCC simulation (Potts spin system) is wrapped as a WebFlow module and its real-time output stream is passed to the Display and Image Filter Modules which enable real time control and fine-tuning of the visualization display.

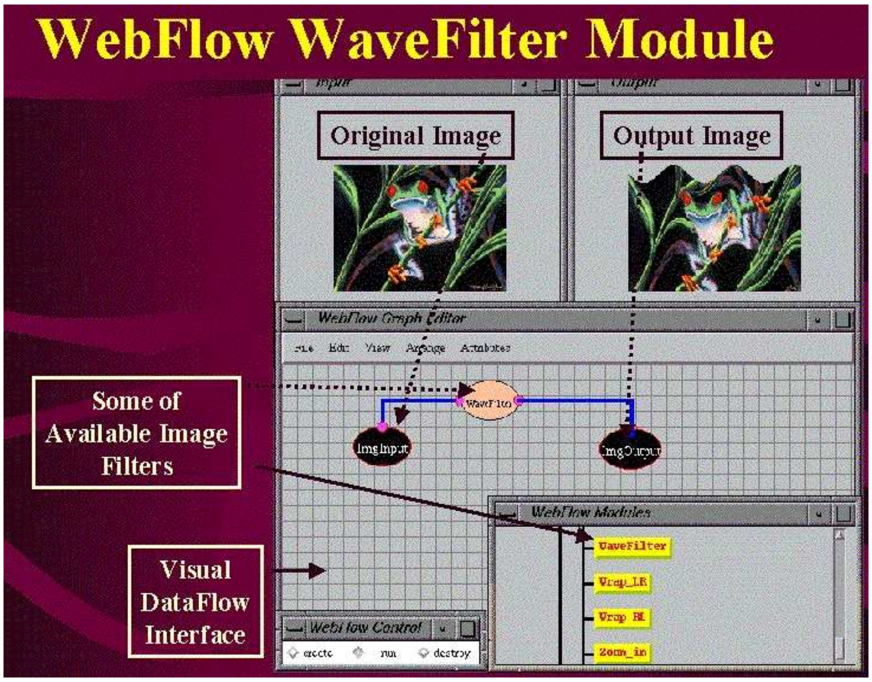

Figure 2: A simple (pure Java) Image Filter module (Wave Filter). Image input, output and filter are represented as visual icons in the WebFlow editor applet. Modules/icons can be selected from a tree-structured palette/navigator.

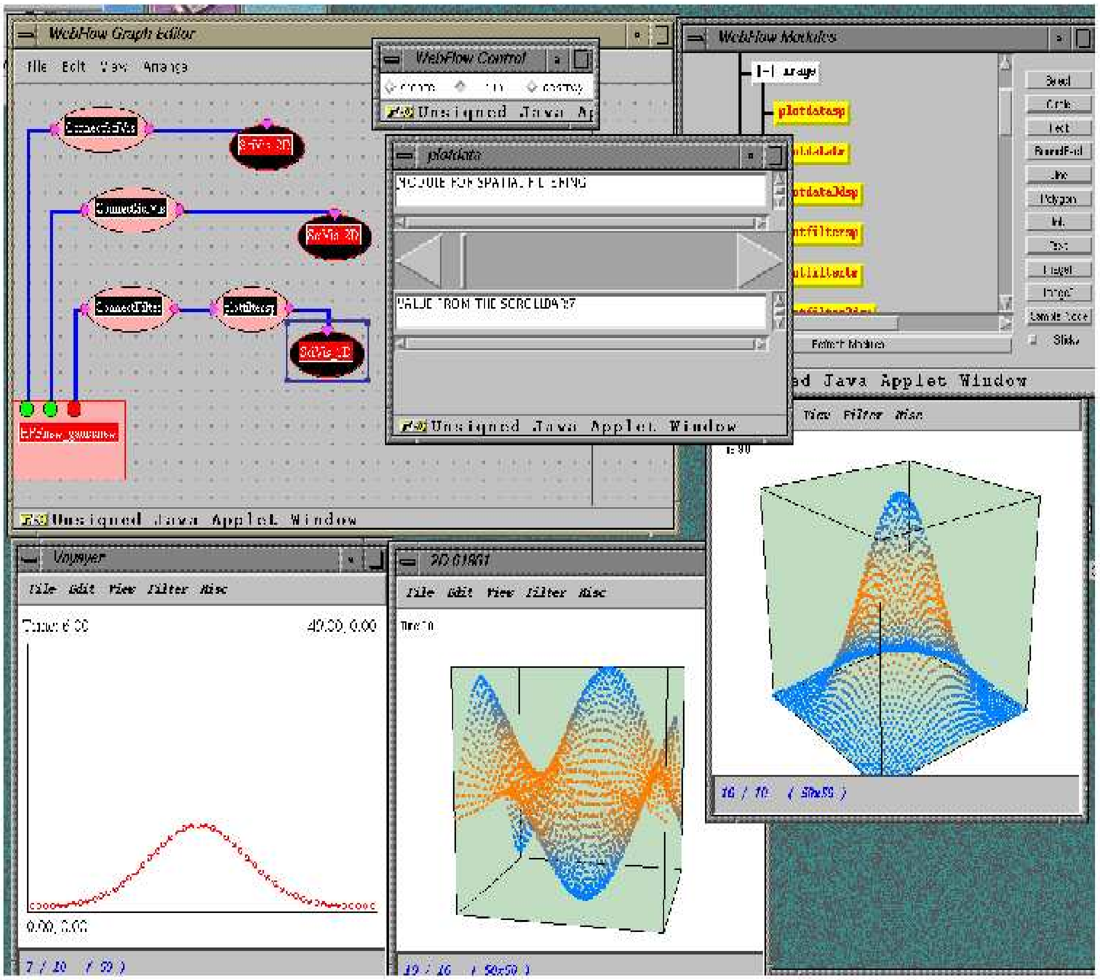

Figure 3: WebFlow demonstration at Supercomputing '97. HPCC simulation (Binary Black Holes) with the DARP based scripting/debugging support is wrapped as WebFlow module with variable number outputs, specified interactively and represeting visual debugging probes. Real-time data streams extracted this way from the simulation as passed (via optional filters) to suitable visualization modules that wrap and control NPAC SciVis display windows.

gies is not an easy task. CORBA offers one promising avenue, especially for the large scale 'extreme' metacomputing but there are several alternatives. For example, Java is continuously promoting the 100% pure Java metodhology - this can hardly be viewed as realistic by today's HPDC community but the same Java is now offering some CORBA support via the JavaIDL framework, to become soon the mainstream as part of JDK1.2. Microsoft claims they already solved all HPDC problems within their DCOM framework. Some aspects of DCOM are indeed interesting from the high performance computing perspective but it remains to be seen if and how fast will DCOM penetrate the UNIX domain. Finally, the World-Wide Web Consortium is developing a set of new standards such as XML, DOM, RDF and HTTP-NG which, when combined, can be viewed as yet another, new emergent distributed object model (sometimes referred as WOM [7]) which is likely easier to be adopted by simple to medium complex distributed object/component applications.

Recent OMG/DARPA workshop on compositional software architectures [7] illustrated very well both the growing momentum and the multitude of options and the uncertainty of the overal direction in the field. 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 excells 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 complete solution.

4 Towards Object Web Based WebFlow

In the new WebFlow design, we adopt the integrative methodology i.e. we setup a multiple-standards based framework in which the best assets of various approaches cummulate and cooparate rather than competing. We start the design from the middleware which offers a core or a 'bus' of modern 3-tier systems and we adopt Java as the most efficient implementation language for the complex control required by the metacomputing middleware. We adopt CORBA as the base distributed object model at the Intranet level, and the (evolving) Web as the world-wide distributed (object) model. System scalability requires fuzzy, transparent boundaries between Intranet and Internet domains which therefore translates into the request of integrating the CORBA and Web technologies. We implement it by building a Java server which handles multiple network prototocols and includes support both for HTTP and IIOP. This can be easily done as IIOP requests are distingished by the 'GIOP' magic word whereas HTTP requests start from the corresponding 'GET', 'POST' etc. string indentifiers.

We called such server JWORB (Java Web Object Request Broker) since it can act both as Java Web Server and as ORB for the Java objects. It can also act as a CORBA client or server for Java objects. Unlike in the WebFlow prototype where we used the JWS from JavaSoft, we decided to implement both HTTP and IIOP support of WORB from scratch. The reason is that neither Jigsaw (which is huge and still evolving) nor Jeeves/JWS (which evolved from core Java / JDK candidate to a commercial product) turned out to be useful for our purposes. Further, we believe that Web servers nowadays should be implemented using solid CORBA services. In the CORBA sector, we have some free Java ORB support such as OrbixWeb, VisiBroker for Java or JavaIDL but none comes with the source release. There is also an evolving GNU Java ORB called JacORB which however started before JavaIDL mapping was standardized, it already grew to a large volume and is now being slowly converted to the current OMG standards. Therefore, we decided to develop our own Java ORB sector as well and to make it fully compliant with the source level components of the coming JavaIDL (such as the org.omg.CORBA package).

With the JWORB based middleware, we can now address both the back-end and front-end layers of WebFlow in a uniform and elegant way using the IIOP protocol. Back-end components will be typically packaged as C/C++ CORBA servers (possibly wrapping some Fortran codes) and represented by the corresponding Java proxies managed by JWORB. The emergent CORBA components standard under development by OMG offers a natural model for the WebFlow middleware encapsulation of such proxies, mapped directly into JavaBeans at the JWORB level. Front-end remains fully factorized as in the current WebFlow prototype and can be given by any visual authoring package with a suitable 'bean box' functionality.

5 Front End: Commodity Tools for Visual Authoring

As part of a study of new visual programming standards, we have analyzed recently a suite of new tools appearing on the rapidly growing visual authoring market, including VisualStudio from JavaSoft, VisualAge from IBM, VisualCafe from Symantec, Rational Rose from Rational and VisualModeler from Microsoft. It appears that the visual componentware authoring products are still based on early custom models but there is already a promising stable standard initiative in the area of visual object-oriented design, represented by Uniform Modeling Language (UML).

UML is developed by an industry consortium led by Rational and supported by Microsoft, and was recently adopted as CORBA standard by OMG (together with the associated Meta Object Facility). UML offers a broad spectrum of visual authoring graph topologies, some initial constructs for parallel processing and an extensibility model. We are currently analyzing this new standard from the perspective of adopting it as a visual model for new WebFlow front-end.

6 Middleware: Java Web Object Request Broker (JWORB)

JWORB is a multi-protocol extensible server written in Java. The base server has HTTP and IIOP protocol support. It can serve documents as an HTTP Web Server and it handles the IIOP connections as an Object Request Broker. As an HTTP server, JWORB supports Servlet and CGI mechanism. Any servlet developed with Java Servlet API can run with JWORB.

Since JWORB design is Object Oriented, it is very easy to add other protocols. As JWORB starts up, it looks at configuration files to figure out which protocols it is capable of handling and it loads the necessary protocol classes for each protocol. If we want to add a new protocol, we need to implement a few abstract classes defined for the protocol object and to register this protocol implementation in the configuration file.

After JWORB accepts a connection, it asks each protocol handler object whether it can recognize this protocol or not. If JWORB finds a handler which claims that it can serve this connection, then this protocol handler deals with this connection.

After the core JWORB server accepts a connection, it asks for a worker thread from the worker pool, it gives this connection to the worker thread and it returns its accepting state if there are available workers in the thread pool. The thread pool manager prevents JWORB from consuming all resources on the machine and creating a new thread object on each client request.

In the current design of core JWORB, the server process returns to accepting state if there is an available worker thread(s) for next request. Otherwise, it waits for a notification from the thread pool manager. In the next releases, we are planning to define an Event Queue and use CORBA Event Service to keep the request events in the queue for subsequent processing. This approach is very much like handling events in the distributed simulations and in fact our design here is driven by our DoD work on JWORB based HLA/RTI support for DoD Modeling and Simulation.

JWORB provides IIOP support which is fully compliant with Java/IDL mapping by the OMG. We implemented the server side support and we are currently using idltojava from JavaSoft's JavaIDL as our IDL compiler. idltojava will be the standard utility of the next JDK releases.

We tested the performance of server objects by echoing an array of integers and structures that contains only one integer value. We performed 100 trials for each array size and we got an average of these measurements. In these tests, client and server objects were running on two different machines. Since we only finished the server side support, we used JacORB on the client side to conduct the necessary tests for the current JWORB.

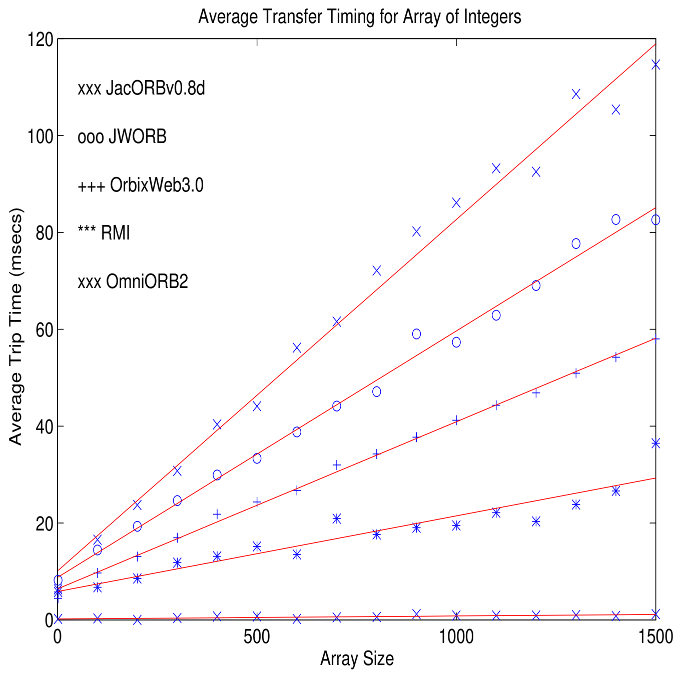

Figure 4: IIOP communication performance for variable size integer array transfer by four Java ORBs: JavaSoft RMI, JWORB, OrbixWeb and RMI. As seen, initial JWORB performance is reasonable and further optimizations are under way. RMI appears to be faster here than all IIOP based models.

The timing results presented above indicate that that JWORB performance is reasonable when compared with other ORBs even if we didn't invest yet much time into optimizing the IIOP communication channel. The ping value for various ORBs is the range of 3-5 msecs which is consistent with the timing values reported in the Orfali and Harkey book [13]. However, more study is needed to understand detailed differences between the slopes for various ORBs. One reason for the differences is related to the use of Java object serialization by RMI. In consequence, each structure transfer is associated with creating a separate object and RMI performs poorly for arrays of structure. JacORB uses object serialization also for arrays of primitive types and hence its performance is poor on both figures.

In the final version of the paper we will include more detailed performance analysis of various ORBs, including C/C++ ORBs such as omniORB2 or TAO [19] which is performance optimized for real time applications, as well as the comparison of the communication channels of various ORBs with the true high performance channels of PVM, MPI and Nexus. It should be noted that the WebFlow

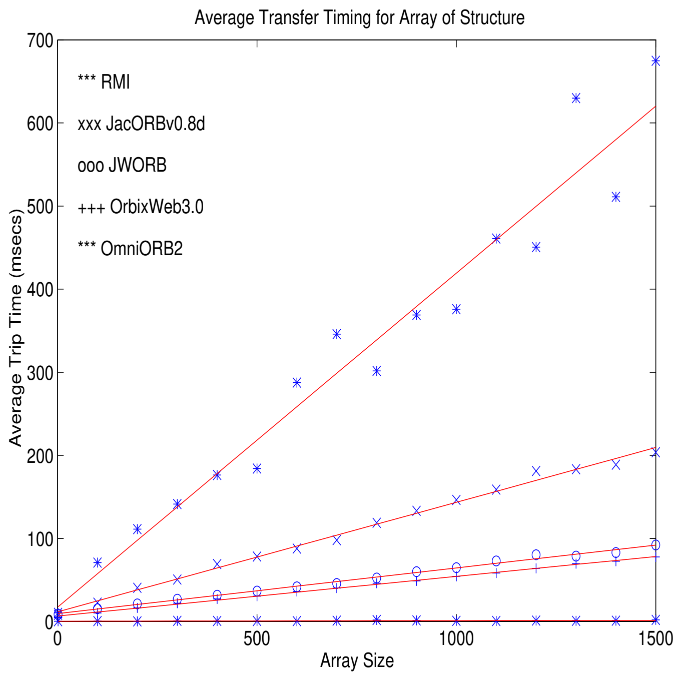

Figure 5: IIOP communication performance for transferring a variable size array of structures by four Java ORBs: JavaSoft RMI, JWORB, OrbixWeb and RMI. Poor RMI performance is due to the object serialization overhead, absent in the IIOP/CDR protocol.

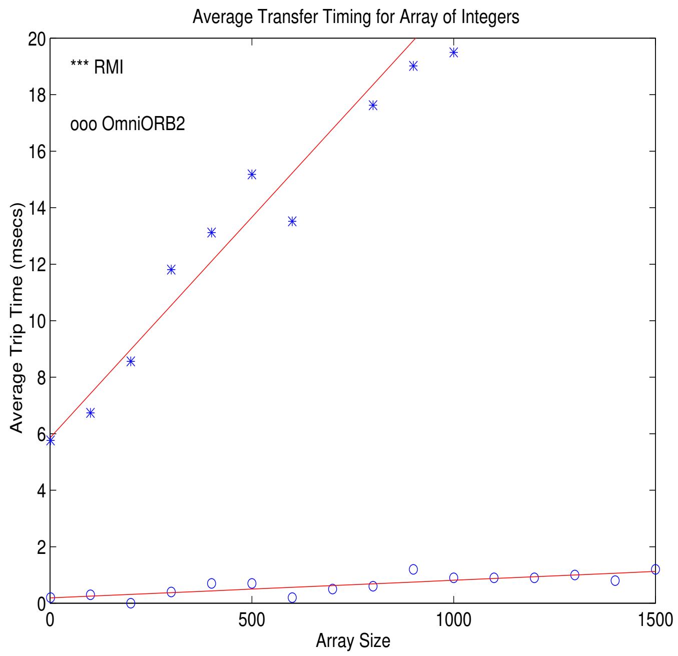

Figure 6: Initial performance comparison of a C++ ORB (omniORB) with the fastest (for integer arrays) Java ORB (RMI). As seen, C++ outperforms Java when passing data between distributed objects by a factor of 20.

based metacomputing will be based on Globus/Nexus [14] backend (see next Section) and the associated high performance RIO communication channels wrapped in terms of C/C++ ORBs (such as omniORB2) whereas the middleware Java based ORB channels will be used mainly for control, steering, coordination, synchronization, load balancing and other distributed system services. This control layer does not require high bandwidth and it will benefit from the high functionality and quality of service offered by the CORBA model.

Initial performance comparison of a C++ ORB (omniORB2) and a Java ORB (RMI) indicates that C++ outperforms Java by a factor of 20 in the IIOP protocol handling software. The important point here is that both high functionality Java ORB such as JWORB and high performance C++ ORB such as omniORB2 conform to the common IIOP standard and they can naturally cooperate when building large scale 3-tier metacomputing applications.

So far, we got the base IIOP engine of the JWORB server operational and we are now working on implementing the client side support, Interface Repository, Naming Service, Event Service and Portable Object Adapter.

7 Back End: HPCC/Globus, HLA/RTI, Legacy Systems

In parallel with designing and prototyping new CORBA based WebFlow, we are also training Syracuse University students on building simple 3-tier metacomputing applications [20] and we start exploring the interfaces of the current WebFlow prorotype to a suite of external backend components. As part of the NCSA Alliance, we are working with the NCSA scientists on adapting WebFlow for their HPDC applications in the Quantum Monte Carlo/Nanotechnology domain. We are also working with the Argonne team to explore the WebFlow based visual authoring tools as a possible front-end for a broader family of emergent metacomputing applications based on the Metacomputing Toolkit Globus. As part of the DoD HPC Modernization Program, we are building JWORB based support for RTI (Run-Time Infrastructure) which acts as a distributed object bus for the HLA (High Level Architecture) - a new DoD-wide standard for Modeling and Simulation. We are also adapting WebFlow as a visual simulation tool for HLA federations and we are planning WebFlow integration with POOMA [17] as part of the ASCI Level 2 project [18]. Finally, we are also exploring WebFlow model a possible intergration framework for legacy systems such as heterogenous distributed RDMS systems of relevance for telemedicine, distance training, virtual prototyping and other Web/commodity based community networks.

8 Related Work

At the tier-3 level, we already mentioned Globus [14] and we also intend to inspect WebFlow interfaces to the Legion [15] system. Some visual metacomputing issues were also addressed in the related VDCE [16] project at Syracuse. It is plausible that the CORBA based middleware offered by JWORB will allow us to formulate jointly as the HPDC community a CORBA Facility for Metacomputing that would provide users with binding to Globus, Legion, VDCE or other favored metacomputing backends.

At the tier-2 level, we are working with several Java ORBs used for testing purposes and comparative analysis such as OrbixWeb, VisiBroker, and JacORB. In the C++ sector, we view omniORB2 as a useful model and we intend to use it initially to wrap C, C++ and Fortran codes as CORBA components. We are learning from JacORB source when implementing IIOP sector of JWORB and from the Jigsaw and HP-Nexus sources when implementing the HTTP part of JWORB.

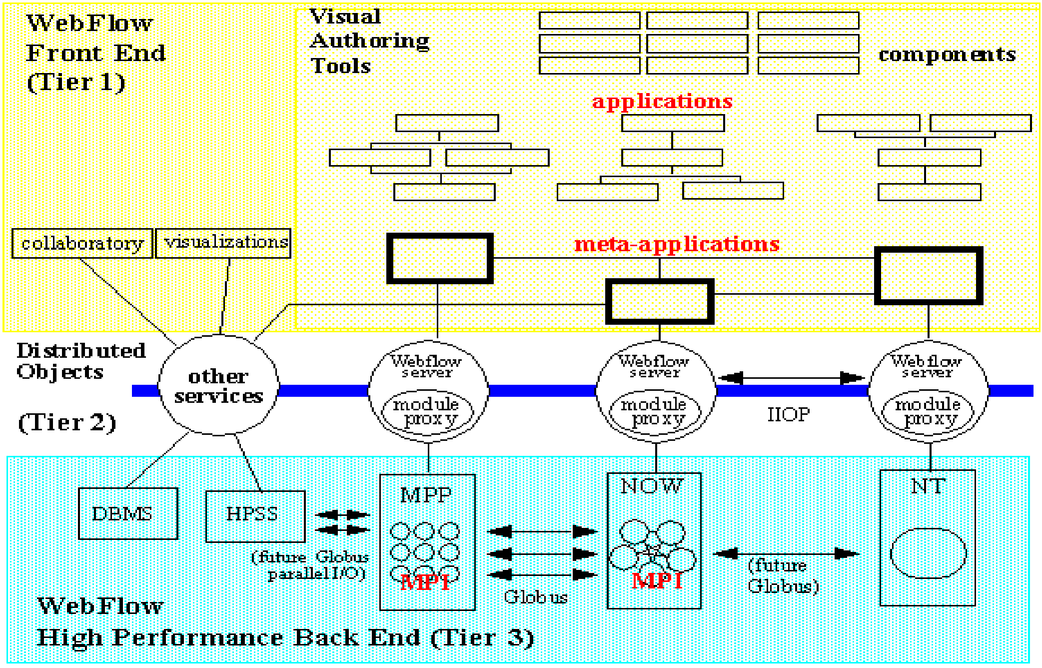

Figure 7: Overall view of the new WebFlow model. UML based front-end in tier 1 is linked to the JWORB based middleware which is linked via suitable module proxies to the metacomputing backend. WebFlow servers in tier 2 are given by JWORB servers dressed by a suitable collection of CORBA services, customized for metacomputing application purposes and supporting security, resource management, load balancing, fault tolerance, session control/journaling etc.

Several tier-1 packages were discussed in the text. We currently view UML [22] as the most promising model and we are at the planning stage of the prototype implementation.

We don't know of any other public domain effort similar to our new WebFlow model. One related commercial system is IBM Component Broker which involves IBM Mainframes in the backend, CORBA based middleware (based on IBM SOM model) and variety of commodity front-end options.

References

[1] Geoffrey Fox, Wojtek Furmanski, Marina Chen, Claudio Rebbi and Jim Cowie, *WebWork: Integrated Programming Environment Tools for National and Grand Challenges*, March 1995.

[2] *WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing* - Supercomputing'96 Handout Material
Also available at http://www.npac.syr.edu/projects/webspace/doc/sc96/handout/handout.ps

[3] *Web based Computing - WebFlow* at CEWES talk, March '97
Also available at http://www.npac.syr.edu/projects/webspace/doc/cewes-mar97/talk

[4] *WebFlow - a visual programming paradigm for Web/Java based coarse grain distributed computing*, February '97, special issue of Concurrency: Practice and Experience on Java for Scientific Computing. Also available at http://www.npac.syr.edu/projects/webspace/doc/cpande/feb97/feb97.ps

[5] WebFlow Project Home Page at http://osprey7.npac.syr.edu:1998/iwt98/projects/webflow

[6] Jason Robbins, GEF: Graph Editing Framework
Available at http://www.ics.uci.edu/pub/arch/gef/

[7] Craig Thompson, *OMG/DARPA Workshop on Compositional Software Architectures*, Monterey, CA January 6-8 1998. Available at http://www.objs.com/workshops/ws9801/

[8] Dennis Gannon, *Component Architectures for High Performance Distributed Meta-Computing*, OMG/DARPA Workshop, January 1998.

[9] JWORB Project Home Page at http://osprey7.npac.syr.edu:1998/iwt98/projects/worb

[10] E. Akarsu, G. Fox, T. Haupt, *DARP: Data Analysis and Rapid Prototyping Environment for Distributed High Performance Computations*, Home Page at http://www.npac.syr.edu/projects/hpfi/

[11] G. Fox, W. Furmanski and T. Haupt, *SC97 handout: High Performance Commodity Computing(HPcc)*, Also available at http://www.npac.syr.edu/users/haupt/SC97/HPccdemos.html

[12] G. Fox and W. Furmanski, *HPcc as High Performance Commodity Computing* Also available at http://www.npac.syr.edu/users/gcf/HPcc/HPcc.html

[13] Robert Orfali and Dan Harkey, *Client/Server Programming with Java and CORBA*, Wiley 1997.

[14] Globus Project Home Page at http://www.globus.org/

[15] Legion Project Home Page at http://www.cs.virginia.edu/l̃egion/

[16] VDCE Project Home Page at http://koshka.cat.syr.edu/projects/vm/

[17] POOMA Project Home Page at http://www.acl.lanl.gov/PoomaFramework/

[18] G. Fox, W. Furmanski and T. Haupt, *ASCI WebFlow: High-Level Programming Environment and Visual Authoring Toolkit for High Performance Distributed Computing.* Available at http://www.npac.syr.edu/projects/asci-webflow/

[19] TAO Project Home Page at http://www.cs.wustl.edu/~schmidt/TAO.html

[20] Tom Studer's CPS714 Final Project at http://osprey7.npac.syr.edu:3768/cps616spring97-docs/cb97tst/CPS714/

[21] NPAC Scivis Project Home Page at http://kopernik.npac.syr.edu:8888/scivis/index.html

[22] UML Home Page at http://www.rational.com/uml/index.shtml