Volume 3, Issue 2, February 2018 ISSN (online): 2456-0006

International Journal of Science Technology Management and Research

Available online at: www.ijstmr.com

High Performance Computing

Pratap Bangosavi
HOD, Dept. of Computer Engineering
Kala Vidya Mandir Institute of Technology Polytechnic
Mumbai, India

Abstract: ***From networks for workstations through to Internet, the high-performance computing community has long advocated composing individual computing resources in an attempt to provide higher quality of service in terms of processing time, size of data store, bandwidth/latency, remote instrument access, special algorithm integration. In recent years this progression has been driven by the vision of "Grid computing" where the computational power, storage power, and specialist functionality of arbitrary networked devices is to be made available on-demand to any other connected device that is allowed to access them. The focus of High-performance computing (HPC) has shifted towards enabling the transparent and most efficient utilization of a wide range of capabilities made available over networks; in as seamless a way as the electrical grid delivers electricity. Modern HPC is as much concerned with access to data and specialized devices in wide-area networks as it is with crunching numbers as quickly as possible.***

Keywords: High-performance computing; Grid computing; Electrical grid delivers electricity.

I. INTRODUCTION

Wide area distributed computing has been a strong focus of research in high performance computing. This research project investigated techniques to develop a High Performance Computing (HPC) grid infrastructure to operate as an interactive research and development tool. Current HPC grid architectures are designed for batch applications, where users submit their job requests, and then wait for notification of job completion. In the batch environment, the user lacks direct control over the execution of their application [7]. This has resulted in the development of software infrastructures like Message Passing Interface (MPI), and in the creation of the National Machine Room and the Grid by DOE. Growing interest in the use of networks of workstations to perform high performance computation has been spurred by the remarkable growth of their computational performance [3]. In bringing the benefits of component-based software development to the domain of high-performance computing, our work does not seek to create a complete component framework. Instead, we have concentrated on providing the integration mechanism that will allow the community to obtain the advantages of such architectures while maintaining high performance. The approach for developing such an HPC grid capability involved evaluating the various developing protocols for interactive grid computing, and then selecting the one with the most growth potential. The Globus Toolkit was selected for the grid development environment. This toolkit is rapidly becoming the de-facto standard for grid research. The grid architecture was evaluated by assembling and demonstrating an in-house 2 interactive demonstration grid using in-house cluster assets and existing code, to verify proper operation on a small scale.

To validate successful operation of the interactive grid implementation, the Hyper spectral Image Exploitation (HIE) framework software, was used [10]. Truth data already exists for this code, thus simplifying the final data analysis. This architecture also has the potential to be applied to streamline the management of Air Force logistics, which currently have about 20 thousand people at each of the three Air Logistics Centers, all of which use legacy software. This paper demonstrates Echo's high performance across heterogeneous hardware platforms, using networked machines resident at Georgia Tech.

A.High performance sharing of distributed data

Echo transports distributed data with performance similar to that achieved by systems like MPI. This level of performance is required if the integration mechanism is to support the normally large data flows that are part of high performance applications. For a distributed visualization, for example, this level of performance enables end-users to interact via meaningful data sets generated at runtime by the computational models being employed.

B.Dynamic data provision and consumption

Echo supports the publish/subscribe model of communication. Thus new components can be introduced into an Echo-based system simply by registering them to the right set of events in the system, without need for re-compilation or re-linking. In addition, components could be dynamically replaced without affecting other components in the system, facilitating system evolution. Event-based publish/subscribe models, like the one offered by Echo, have become increasingly popular and their utility within a variety of other Environments, including Internet- and E-commerce applications [4].

II. THE HIE FRAMEWORK GRID INTERFACE

The Hyper spectral Image Exploitation (HIE) Framework implements a general purpose Web interface for access to remote applications, usually running on HPCs. The Web interface is driven by objects that implement interfaces to specific codes, allowing the Web interface to collect application specific inputs and deliver them to remote applications appropriately. This section describes the steps that use grid services to implement Framework application interface objects. The application interface object is used by the Framework server to determine the variables that have to be supplied through the Web interface The original Framework implementation used a Corba interface to connect the Framework server with the code interface object. For this project, we added a Grid interface to allow

developers to implement code interface objects using the Globus Grid software or the Corba software. For both Globus Grid and Corba, the code to implement the interface and establish the connection between the Framework server and the code interface object can be generated automatically from an interface description that specifies remote functions (implemented in the code interface object) and parameters that are needed to call the functions.

III. GENERATING STUBS AND SKELETONS

For both Globus and Corba, a standard interface description is used to automatically generate the interface functions needed by the client (called a stub) and the server (called a skeleton). Generally, all of the service code needs to be built into the server, by implementing the code that is executed when the service interface (skeleton) is invoked. The term skeleton seems appropriate since the interface code is just a container in which the service must be implemented. The stub, by contrast is just the client's interface to make the call and pass the parameters to the service implementation. In Corba, the interface is specified using a standard called Interface Definition Language (IDL). The Grid uses an Ex (tensible) M (arkup) L (anguage) XML XSLT specification. Extensible style sheet language transformation (XSLT) is a language for transforming XML documents into other XML documents. XSLT is designed for use as part of XSL, which is a style sheet language for XML.

IV. THE COMMON COMPONENT ARCHITECTURE

Formally, the Common Component Architecture is a specification of an HPC-friendly component model. This specification provides a focus for an extensive research and development effort. The research effort emphasizes understanding how best to utilize and implement component based software engineering practices in the high-performance scientific computing arena, and feeding back that information into the broader component software field. In addition to defining the specification, the development effort creates practical reference implementations and helps scientific software developers use them to create CCA-compliant software. Ultimately, a rich marketplace of scientific components will allow new component based applications to be built from predominantly off-the shelf scientific components. The purpose of the CCA is to facilitate and promote the more productive development of high-performance, high-quality scientific software in a way that is simple and natural for scientific software developers. The CCA intentionally has much in common with commodity component models but does not hesitate to do things differently where the needs of HPC dictate. High performance and ease of use are more strongly emphasized in the CCA effort than in commodity component models.

A GLOBUS FUNCTION STUB

The stub is used by the client to call a function that is actually implemented as a service in the grid services container. The code shown in Figure 1 was generated by a Globus utility program for the fwGetFunc interface to the code interface object that is described above.

BREAKDOWN OF COSTS

We have excluded Java RMI from the breakdown because it performs its network ' send ()' operations incrementally during the marshaling process. This allows Java to pipeline the encode and network send operations making a simple cost breakdown impossible. However, as a result of this design decision Java RMI requires tens of thousands of kernel calls to send a 100Kb message, seriously impacting performance. Using XML as a wire format is obviously a decision which has a significant performance impact on an event system. Makes clear two of the most significant issues: the large encode/decode times, and the expanded network transmission times. The former is a result of the distance between the ascii representation used by XML and the native binary data representation. XML encoding costs represent the processing necessary to convert the data from binary to string form and to copy the element begin/end blocks into the output string. Just one end of the encoding time for XML is several times as expensive as the entire round-trip message exchange for the other infrastructures. Network transmission time is also significantly larger for XML because the ASCII-encoded data can be significantly larger than the equivalent binary representation. How much larger depends upon the data, the size of the field labels and other details in the encoding.

INTERGRID

C. Grid Technology Applications

This project has paved the way for future work on an intergrid or a grid of grids, allowing even greater expansion of the network over which jobs can be run and providing the basis for an experimental tested where new concepts and techniques in parallel and distributed computing can be explored.

D. Applications of Grid Protocols

Grid technology has matured to the point that there are now services that directly support access to HPCs. These services use the Global Resource Access Manager protocol, implemented in the Globus toolkit, to negotiate processing resources and to submit jobs through standard HPC batch access systems. Grid protocol implementations are widespread, making them a defacto standard for modern distributed computing.

E. New Computing Models

Grid protocols have a wide range of potential applications. Some of the most promising are for the Teragrid, a large grid-based effort, originally funded by the NSF to connect five large computing facilities and other sites [8]. Other applications include local grids, and multiple site grids, such as hosted by the DoD High Performance Computing Modernization Program (HPCMP) or the Condor Program.

F. Security Considerations

Other application communities include the HPCMP community and the Condor-G system. The Globus Grid security mechanisms can support Kerberos and are likely to be approved for use for HPCMP access. HPCMP centers are developing support for public key

infrastructure (PKI) and we expect that it will be approved eventually to open up implementing Grid services. Condor-G is another grid-based system that uses both Condor and Globus to enable users to access multiple systems as if they are just one [1].

G. Recommendations

Further work can be performed by experimenting with a heterogeneous model. Future funding may be sought to experiment with HPCMP centers. AFRL is in a unique position to leverage distributed and parallel computing expertise, including the DIHT and the HIE to evolve an integrated parallel and distributed computing system where we envision that all future systems will play a role. Figure 2 shows an InterGrid Environment. HPCMP centers are developing support for public key infrastructure (PKI) and we expect that it will be approved eventually to open up implementing Grid services. Condor-G is another grid-based system that uses both Condor and Globus to enable users to access multiple systems as if they are just one. In terms of the flexibility offered to applications, ECho's features most closely resemble the features of systems that support the marshalling of objects as messages. In these systems, sub classing and type extension provide support for robust system evolution that is substantively similar to that provided by Echo's type variation. However, object based marshalling often suffers from prohibitively poor performance. Echo's strength is that it maintains the application integration advantages of object-based systems while significantly outperforming them. As the measurements in the next section will show, Echo also outperforms more traditional message-passing systems in many circumstances.

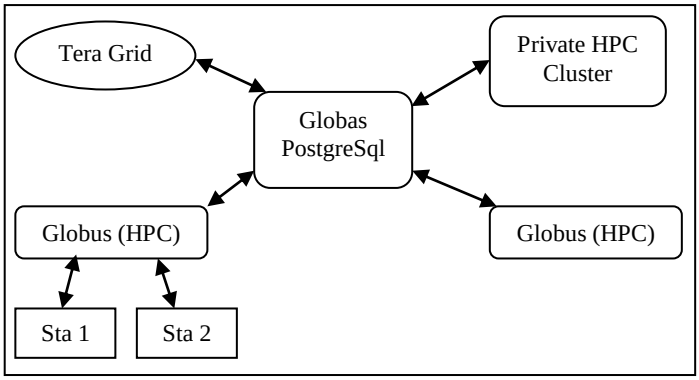

Figure 2: InterGrid Environment

CONCLUSION

This is a significant step up from traditional HPC grid architectures, which are designed for batch applications, where the user lacks direct control over the execution of their application. Applications, such as the HIE framework require a near real-time response and an interactive environment. Lessons learned were how to successfully use the Globus toolkit to run these services, including initializing variables, creating and running the counter service, and using the Globus browser. Future work in the area is to use this capability to run interactive grid based work, supporting the war fighter in areas such as the Joint Battles pace Info sphere.

REFERENCES

1. Frey, J., Tannenbaum, M.L., Livny, M., Foster, I., Tuecke, S., "Condor-G: A Computation Management Agent for Multi-Institutional Grids", Cluster Computing, Volume 5, Number 3, July 2002.
2. C. Chambers, S. J. Eggers, J. Auslander, M. Philipose, M. Mock, and P. Pardyak. Automatic dynamic compilation. Support for event dispatching in extensible systems. In *Proceedings of the Workshop on Compiler Support for Systems Software (WCSSS'96)*. ACM, February 1996.
3. MARIO LAURIA, ANDREW CHIEN, "MPI-FM: High Performance MPI on Workstation Clusters", JOURNAL OF PARALLEL AND DISTRIBUTED COMPUTING **40,** 4–18 (1997) ARTICLE NO. PC961264.
4. R. Strom, G. Banavar, T. Chandra, M. Kaplan, K. Miller, B. Mukherjee, D. Sturman, and M. Ward. Gryphon: An information flow based approach to message brokering. In *International Symposium on Software Reliability Engineering '98 Fast Abstract*, 1998.
5. Ross, V.W. and Spetka, S., "Grid Computing for HPC Data Centers", AFRL Technical Report TR-2006-298, October 2006.
6. On Web site http://www.dtic.mil/cgi-bin/GetTRDoc?AD=ADA466685.
7. "GRID COMPUTING FOR HIGH PERFORMANCE COMPUTING (HPC) DATA CENTERS "AFRL-IF-RS-TR-2007-91 In-House Final Technical Report March 2007.
8. Spetka, S.E., Ramseyer, G.O., Fitzgerald, D.J., Linderman, R.W., "A Distributed Parallel Processing System for Command and Control Imagery", 7th International Command and Control Research and Technology Symposium, Quebec City, QC, Canada, September 16-20, 2002.
9. Ross, V.W. and Spetka, S., "Grid Computing for HPC Data Centers", AFRL Technical Report TR-2006-298, October 2006.
10. Habanero. NCSA and University of Illinois at Urbana. http://notme.ncsa.uiuc.edu/SDG/Software/Habanero.
11. B. MacIntyre and S. Feiner. Language-level support for exploratory programming of distributed virtual environments, In *Proceedings of Symposium on User Interface Software and Technology (UIST'96)*, pages 83–95, November 1996.
12. Y. Ashlad, B. E. Martin, M. Marathe, and C. Le. Asynchronous notifications among distributed objects. In *Proceedings of the Conference on Object-Oriented Technologies and Systems*. Usenix Association, June 1996.