High-Speed Network and Grid Computing for High-End Computation: Application in Geodynamics Ensemble Simulations

S. Zhou[1], W. Kuang[2], W. Jiang[3], P. Gary[2], J. Palencia[4], G. Gardner[5]
[1]Northrop Grumman/TASC, [2]NASA Goddard Space Flight Center, [3]JCET, UMBC, [4]Raytheon ITSS, [5]INDUSCORP

Abstract- **High-speed network and grid computing have been actively investigated, and their capabilities are being demonstrated. However, their application to high-end scientific computing and modeling is still to be explored. In this paper we discuss the related issues and present our prototype work on applying XCAT3 framework technology to geomagnetic data assimilation development with distributed computers, connected through an up to 10 Gigabit Ethernet network.**

I. INTRODUCTION

The method of ensemble simulation has been widely used to analyze observations and make predictions with numerical models in which known physics and observations are integrated to forecast changes in the future. A well-known example is weather forecasting [1]. In this approach, a suite (ensemble) of numerical tests is used to obtain the best estimation used for optimal forecasting. The ensemble size often varies for different geophysical problems. A typical ensemble size is approximately 30 independent numerical tests. It is not unusual to have a much larger ensemble size.

More recently, data assimilation has been used in solid Earth science, where even more complicated global models are used to predict geophysical environment changes over much longer geological periods [2]. Another on-going area of research is geomagnetic data assimilation [3], in which surface geomagnetic data over the past several thousand years will be assimilated into geodynamo models to predict geomagnetic secular variation on time scales of several decades and longer.

One challenge these solid Earth science research activities share is the unprecedented demand on computing resources. Consider one geodynamo model, the MoSST (Modular, Scalable, Self-consistent, Three-dimensional) core dynamics model [4,5], as an example for illustration. With a modest truncation level of (40, 40, 40), the numerical model requires 500 MB RAM and 10^{12} flops for a single numerical test. Considering that for geomagnetic data assimilation, the truncation level may increase to (200, 200, 200), then one numerical test shall require 700 GB RAM and 10^{16} flops. For a successful assimilation, we expect a minimum of 30 such tests. With current available computing capabilities, it is neither practical nor obvious that such a grand-scale computation can be carried out in a single supercomputing facility.

However, such ensemble tests are nearly independent: individual initial states are generated for each test. The final assimilated results will then be collected for analysis. And new initial states will consequently be generated for further simulation. Therefore, there is very limited communication among the individual ensemble tests. Such a modeling nature can be ideal for grid-computation application.

Here we will focus on establishing a system to predict geomagnetic secular variation on decadal and longer time scales, utilizing surface geomagnetic/paleomagnetic records and the MoSST core dynamics model. In this approach, model forecast results and observations are weighted to provide the initial state for assimilation. Typically 30 independent numerical tests are necessary for a reasonable ensemble size. This could easily require a computing cycle on the order of petaflops and larger.

A single supercomputing facility for such studies is an optimal choice, but is not practical given that requirements for computational time and memory exceed its capabilities. However, it is relatively easy for users (researchers) to manage because of a unified system environment.

Grid computing can be a much better choice so that independent numerical tests can be carried out on different systems. However, researchers (users) have to deal with heterogeneous system environments and other problems, such as those in network communication.

In this paper, we will discuss the issues of effectively applying grid computing to high-end computation and present a practical case: managing a distributed ensemble simulation. We will describe a prototype based on the XCAT3 [6] framework and illustrate it through running an ensemble of geodynamic applications on distributed computers, which are connected through an Ethernet network with speeds up to 10 gigabits per second. The relevant activities of building a high-speed network at NASA Goddard Space Flight Center (GSFC) will be reported.

II. GRID COMPUTING IN HIGH-END COMPUTATION

Grid computing integrates networking, communication, computation, and information to provide a computation and data management capability that forms a virtual platform for information [7]. Recently the Open Grid Services Architecture (OGSA) has been proposed to standardize grid computing. Grid-enabled applications are being demonstrated. A recent development in high-speed networks, the National LambdaRail, enables transportation of huge amounts of data across the U.S. [8, 9]. At GSFC, a project is being carried out to connect GSFC to the National LambdaRail [10].

Grid computing, armed with LambdaRail technology, provides a good opportunity for distributed high-end computing applications. There are two kinds of such applications: tightly as well as loosely coupled. One tightly coupled application is climate modeling. A typical Earth system model consists of several complex model components, such as atmosphere, ocean, land, and sea-ice, coupled together through frequent data exchanges. In a decadal climate simulation, an ocean model has to exchange information with an atmosphere model at least once a calendar day. But ensemble forecasting is a loosely coupled application. In that case, individual simulations are independent of each other. Only the ensemble driver needs to take the feedback of the ongoing individual simulations and consequently constructs the running conditions, such as initial conditions, for the individual simulation to be dispatched.

In principle, grid computing over a Lambda network should support loosely as well as tightly coupled applications. However, the political issues among different supercomputing centers, such as coordinated job schedules and immature network connectivity, make the tightly coupled applications less favorable, although a few special simulations can still be demonstrated. We believe that loosely coupled applications, such as ensemble simulations, will prevail in the current environment.

In the next section, we will present our prototype of managing distributed ensemble simulations based on the XCAT3 framework.

III. DESCRIPTION OF THE PROTOTYPE

Our goal is to use grid computing technology as middleware, dispatching a set of ensemble jobs to different computers across networks, collecting feedback from dispatched ensemble jobs, constructing new running conditions (such as initial conditions), and dispatching another set of ensemble jobs. This middleware should be portable across different computer platforms and user-friendly.

We chose XCAT3, a Common Component Architecture (CCA)-compliant framework, since it is implemented in Java, which ensures portability. In addition, its component employs CCA's Use-Provide System Pattern [11] and can be assembled and executed easily with a Jython [12] script as well as a Web-based graphical user interface (GUI) tool [6].

XCAT was also developed to satisfy the Open Grid Standard Interface (OGSI) specification. In this way, its components are accessible via standard grid clients, too. XCAT employs the Remote Method Invocation (RMI) mechanism implemented by XSOAP [13] to allow communication and control among local and remote components. Merging the two standards (CCA and OGSI), XCAT uses an approach where a component is modeled as a set of grid services.

In our prototype, the XCAT3 framework is used to handle network communication, but we develop XCAT3 components to send and receive messages as well as execute the commands on those ensemble members. In each component, there are five standard interfaces [initialize(), run(), finalize(), provideCMD(), and useCMD()] to be implemented. In addition, a standard data type, geoCMD, is used to exchange data among components. This definition of a component is similar to the ESMF-CCA Prototype [14, 15].

There are two kinds of implementation for this type of components: One is to dispatch the messages to distributed ensemble members or collect the messages from ensemble members. Another is to receive the message from the driver and execute the message on ensemble members, such as invoking the command, mpirun, and sending the feedback. The function, initalize(), sets up initial messages; provideCMD() offers its message for other component to use; useCMD() takes the message from other component; run() acts on the taken message and gives the feedback; finalize() cleans up the system, such as file closure and port disconnection.

CCA components interact by adhering to the Uses-Provides design pattern. This means that each component publishes the functionality that it allows other components to access. These published methods are known as *Provides Ports*. Each component also publishes the functionality that it needs to have other components perform for it. These published methods are known as *Uses Ports*. Conceptually a 'port' can be thought of as a contract between components of a system. It is the equivalent of Java interfaces and pure abstract class definitions in C++. The CCA framework includes one additional type of port. The '*go*' port is the starting point for executing systems of components. Driver components implement the '*go'* port. They schedule and control the running sequence of components. A CCA-compliant framework, like Ccaffeine [11] or XCAT, is responsible for connecting and managing ports. For example, a component, driver, with a *Uses Port* of name geo1Use and type CMD, can be connected to another component, geo1, with a *Provides Port* of name geo1Provide and the same type, CMD (see Figure 1). The connection is carried out in the run time.

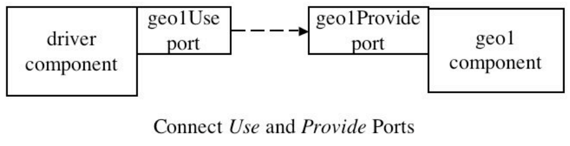

Figure 1 Illustration of coupling two components using CCA.

In XCAT3, each component is developed in Java, compiled as a Java class (.class), and stored in an individual directory. With a Jython script, a user lists *Provides* components and *Uses* components, specifies their locations (e.g., file directory and computer machine), and then chooses one of three mechanisms to handle creation of the components: local, SSH, or gram [which uses the Globus' Grid Resource Allocation and Management (GRAM)]. After that, live instances of the components are created on the target computer machine and the *Provides Port* and *Uses Port* of the components are connected, and the execution of the program starts by invoking the go method of the driver component.

IV. CONFIGURATION TO TEST THE PROTOTYPE

We dedicated five computer nodes in a Linux cluster to mimic a distributed computing environment. There are two processors in each node. The five nodes are divided into three groups: one, two, and two nodes. Each group has its own local disk space. Each two-node group acts as a small cluster. We put a driver component and a dispatcher component, "dispatch," on the host computer with one node, and two receiver components, "geo1" and "geo2", on other two small clusters, respectively. For simplicity, we chose SSH to connect the two-node small clusters and the host computer.

In the small cluster where the receiver component, "geo", is installed, an ensemble member of the application is also installed. The code of an ensemble member can be serial as well as parallel (see Fig. 2).

A parallel geodynamics code, MoSST, is chosen to represent an application. This code uses the master-slave parallel paradigm. All four processors in the small cluster were used for running MoSST. The receiver component, such as "geo1", is located at the master processor. The number of processors for slave is determined by the application code rather than the grid computing code.

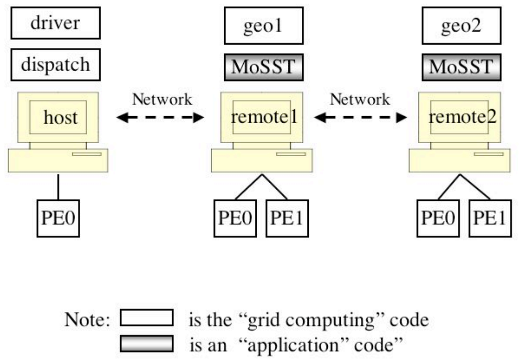

Figure 2 System architecture.

V. CURRENT OPERATING ENVIRONMENTS

Our prototype is designed to support various running scenarios by varying the exchanged messages among components as well as the running sequence in the driver. In addition, our component interface ensures that adding a new receiver component such as "geo3" can be very easy. It is more or less a duplication of "geo". In the following section, we demonstrate a typical scenario: invoking a parallel job in MPI on distributed ensemble members.

At the beginning of a simulation, the XCAT3 framework creates each component. Its setService utility registers the components of driver, dispatch, geo1, and geo2. The *Uses Ports* of the driver component connects to the *Provides Ports* of dispatch, geo1, and geo2, respectively (See Fig. 3). The components interact through the string message of type geoCMD (see Fig. 4). The dispatch component first *provides* the message to the driver, and then the driver passes the message to the geo1 and geo2 components, respectively. The geo1 and geo2 components take the message and invoke the mpirun command through a Java "exec" system call. A feedback message on whether the invocation is successful or not is sent back to the driver.

Using the same grid computing code, we also successfully ran a similar scenario for the configure where two computer nodes in another Linux cluster are connected through a 10 Gigabit Ethernet (GE) network. This network represented a local prototype for the National LambdaRail. It consists of two Force 10 E300 10-GE switches connected to two computer nodes, respectively. In the near future one Force 10 E300 switch will be replaced with an Extreme Network Summit 400-48t 1-GE switch, which has two 10-GE uplinks. The replaced Force 10 E300 will be used to expand the local prototype to other computer resources at GSFC.

Since the messages used in communicating local and remote components are not long, the performance degradation in communication is not noticeable, as expected. However, we did observe a delay occurring in the process of establishing the connection between local and remote components through SSH at the beginning of an ensemble simulation.

The design philosophy of this prototype is to have minimum intrusiveness to the supported applications in the coding as well as in computational efficiency. The programmer only needs to modify part of the supported application to interact with the grid computing "geo" component. For example, a simulation status file with a specific name needs to be created so that "geo" can detect it and report back to the driver component. However, such actions take tiny time and are not expected to have any impacts on the computational efficiency of a computation-intensive application. That has been verified in our tests. As a matter of fact, a production-quality MoSST ensemble simulation with a modest truncation level of (40,40,40) takes three days to complete in a cluster with two dual-processor nodes without the XCAT framework.

VI. DISCUSSION

By testing our ensemble-dispatching prototype based on XCAT3 with a production-quality parallel geodynamics code, we believe that appropriately applying grid computing technology to high-end computation applications can be very appealing to scientists and engineers.

Existing approaches used in running a production-quality ensemble simulation are more or less done by hand on a single supercomputer. For example, the NOAA/National Centers for Environmental Prediction (NCEP) use shell scripts and computer platform-specific libraries to submit one or more ensemble weather forecasting jobs into their IBM supercomputer, collect simulation results, and resubmit ensemble jobs manually. One single supercomputer is simply not capable of supporting an ensemble simulation with the necessary resolution. That leads the forecaster to reduce resolution as well as the number of ensemble members. A similar practice is also used at the Center for Ocean-Land-Atmosphere Studies (COLA) to predict El Niño-Southern Oscillation (ENSO) events with its coupled atmosphere-ocean model. In short, a tool that is user-friendly and capable of using multiple-supercomputers is needed for the application of computation-intensive ensemble simulations.

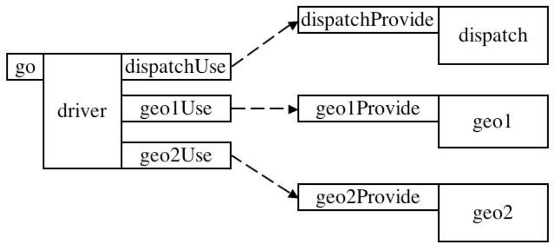

Figure 3 Relationship among driver, dispatcher, and receiver components.

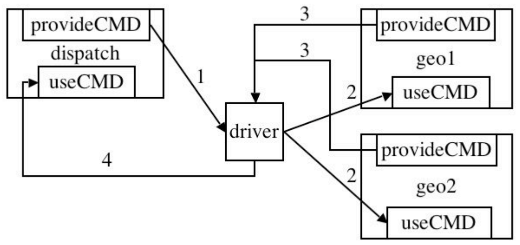

Figure 4 Flow diagram of invoking a distributed ensemble simulation. The data of type "geoCMD" is exchanged among components.

One major goal of our prototype design is to encapsulate the complexity of network programming and to provide a user-friendly environment, which is also one of the key factors for a middleware to succeed. Our prototype achieves that goal to a great extent. A user only needs to customize the content of messages and specify the computer names where an ensemble member is to be invoked. However, there are several areas worth improving:

- Add protocols for efficiently transferring a large amount of data among components since initial data or simulation outputs may be moved between local and remote computers. XCAT3 developers are replacing the default protocol, XSOAP, with GridFTP.
- Deal with feedback of a running ensemble simulation. An ensemble member may fail prematurely, output some simulation results, or finish. The dispatch component needs to treat the feedback following the flow diagram shown in Fig. 5. Basically, the grid computing code, "geo", will monitor the performance of the application code, MoSST. Once an event is detected, "geo" will compose a message and send it back to the "dispatcher" component via the "driver" component. The dispatcher will evaluate the message and send its response back to "geo". Recently, we have developed a system to detect the output of a simulation and report the finding back to the driver.
- Intelligently dispatch the next ensemble members. Currently we treat individual ensemble members independently. That is, the next dispatched ensemble member does not depend on the result of the current one. In some applications, there is a need to construct a new running environment for the next ensemble simulation. As shown in Fig. 6, the "dispatcher" component sends the message, via the "driver" component, to the receiver component, "geo1", at the first remote computer. After its corresponding application completes, "geo1" reports the simulation result back to the driver. The dispatcher evaluates the reported simulation result and assembles a new running environment and sends it to "geo2" at the second remote computer.

Besides providing a standardized way of accessing distributed computing resources, grid computing also allows an application to access distributed data resources, which is especially important for those applications with data assimilation components. For example, the accuracy of weather forecasting strongly depends on assimilated observation data. A variety of observation data are stored in a few geographically distributed centers. Currently the data have to be fetched together without selection, categorized, and then assimilated. This "centralization" approach is not optimal and complicates the data assimilation code. With grid computing technology, providing data can be implemented as a grid service. One NOAA laboratory is proposing to develop such a system. Since the components, "driver", "dispatcher", "geo", in our prototype are grid services through XCAT3, our prototype also can support those applications that need to access distributed data provided as a grid service.

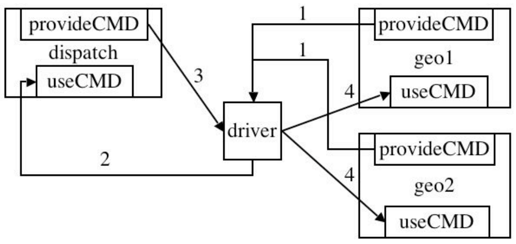

Figure 5 Flow diagram of dealing with feedback of a running distributed ensemble simulation.

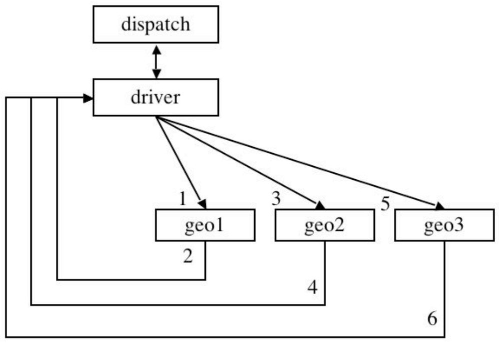

Figure 6 Flow diagram of intelligently dispatching next ensemble members.

NASA GSFC is actively updating its network to connect to the National LambdaRail (NLR) (see [10]). This project involves three parts: the local network, the regional network, and the transcontinental network. The initial GSFC L-Net will have an inter-building 10-GE backbone implemented with 10-GE inter-connected switches. Then the NSF-funded Dynamic Resource Allocation via GMPLS Optical Networks (DRAGON) ring is used to connect GSFC to McLean, Virginia, where the NLR can be reached (see Fig. 7). We will extend this prototype and run ensemble simulations first among computers at GSFC campus via the L-Net and then across the U.S. via the NLR and report the ongoing progress in the workshop.

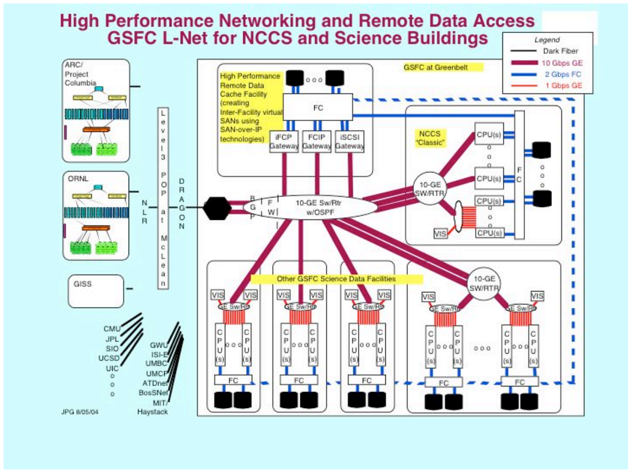

Figure 7 NASA GSFC IRAD work on regional fast network.

VII. CONCLUSION

We have developed a prototype for managing ensemble simulations with the features of grid computing. Preliminary testing on the prototype shows that parallel geodynamics ensemble simulations can be performed with grid technology in a user-friendly way.

ACKNOWLEDGEMENT

This is project is supported by the NASA Earth-Sun System Technology Office Computational Technologies Project. We would like to thank Sriram Krishnan, Madhusudhan Govindaraju, Dennis Gannon, and Randall Bramley for technical support in using the XCAT framework.

REFERENCES

[1] E. Kalnay, *Atmospheric Modeling, Data Assimilation and Predictability,* Cambridge, 2003.

[2] H.-P. Bunge, C. R. Hagelberg, and B. J. Travis, "Mantle Circulation Models with Variational Data Assimilation: Inferring Past Mantle Flow and Structure from Plate Motion Histories and Seismic Tomography," Geophy. J. Int., 152, 280-301, 2003.

[3] W. Kuang, A. Tangborn, and T. Sabaka, "A Atable Model Mapping Geomagnetic Data to Geodynamo Solution: Towards Geomagnetic Data Assimilation," submitted to Earth. Planet. Sci. Lett.

[4] W. Kuang, and J. Bloxham, "Numerical Modeling of Magnetohydrodyanmic Convection in a Rapidly Rotating Spherical Shell: Weak and Strong Field Dynamo Actions," J. Comp. Phys., 153, 51-81, 1999.

[5] W. Kuang, and B. F. Chao, "Geodynamo Modeling and Core-Mantle Interaction", in *The Core-Mantle Boundary Region* (eds. Dehandt, Creager, Karato, Zatman, AGU), Geodynamics Series, 31, 193-212, 2003.

[6] S. Krishnan and D. Gannon, "XCAT3: A Framework for CCA Components as OGSA Services," Proceedings of HIPS 2004, 9th International Workshop on High-Level Parallel Programming Models and Supportive Environments, April 2004. More information on XCAT is available at http://www.extreme.indiana.edu/xcat/

[7] F. Berman, G. Fox, and A.J.G. Hey, "Grid Computing," Wiley, 2002

[8] Prototyping Tomorrow's Optical Cyberinfrastructure, http://www.calit2.net/briefingPapers/optiputer.html

[9] GTP: Group Transport Protocol for Lambda-Grids, http://www.optiputer.net/publications/articles/CHIEN-GTP_CCGrid2004.pdf

[10] NASA Goddard Space Flight Center Lambda Network, http://cisto.gsfc.nasa.gov/IRAD_Lambda.html.

[11] Common Component Architecture, http://www.cca-forum.org/

[12] Jython, http://www.jython.org/

[13] XSOAP, http://www.extreme.indiana.edu/xgws/xsoap/

[14] S. Zhou et al., "Prototyping of the ESMF Using DOE's CCA," NASA Earth Science Technology Conference 2003

[15] S. Zhou, "Coupling Climate Models with Earth System Modeling Framework and Common Component Architecture," Concurrency Computation: Practice and Experience, in press.