Distributed Hybrid Earthquake Engineering Experiments: Experiences with a Ground-Shaking Grid Application

Laura Pearlman[1], Carl Kesselman[1], Sridhar Gullapalli[1], B.F. Spencer, Jr.[2], Joe Futrelle[3], Kathleen Ricker[3], Ian Foster[4,5], Paul Hubbard[5], Charles Severance[6]
[1]*USC Information Sciences Institute, University of Southern California, Los Angeles, CA*
[2]*Department of Civil and Environmental Engineering, UIUC, Champaign, IL*
[3]*National Center for Supercomputing Applications, UIUC, Champaign, IL*
[4]*Argonne National Laboratory, Argonne, IL* [5]*University of Chicago, Chicago, IL*
[6]*University of Michigan, Ann Arbor, MI*
laura@isi.edu

Abstract

Earthquake engineers have traditionally investigated the behavior of structures with either computational simulations or physical experiments. Recently, a new hybrid approach has been proposed that allows tests to be decomposed into independent substructures that can be located at different test facilities, tested separately, and integrated via a computational simulation. We describe a Grid-based architecture for performing such novel distributed hybrid computational/physical experiments. We discuss the requirements that underlie this extremely challenging application of Grid technologies, describe our architecture and implementation, and discuss our experiences with the application of this architecture within an unprecedented earthquake engineering test that coupled large-scale physical experiments in Illinois and Colorado with a computational simulation. Our results point to the remarkable impacts that Grid technologies can have on the practice of engineering, and also contribute to our understanding of how to build and deploy effective Grid applications.

1. Introduction

Until recently, earthquake engineers studied the effects of ground motion on structures in one of two ways: by running a computational simulation or performing a physical experiment. (In the latter case, a physical model is constructed and instrumented, forces are applied to it, and results are measured and logged.) In contrast, *tightly-coupled hybrid experiments* combine the two approaches [14, 19]: one part of a structure is modeled computationally and another part as a physical experiment, and the computation and the physical experiment's control system communicate and influence each other's behavior over the course of the experiment.

Hybrid experiments are relatively straightforward to perform when they involve a single physical experiment, as the computational simulation and physical apparatus can be co-located. However, for some earthquake engineering problems, it would be desirable to construct a hybrid experiment that involves more than one physical experiment—for example, an experiment involving a large geotechnical centrifuge to model soil motion and a large shake table to model the motion of a structure above ground. Physical experiments (and the physical components of hybrid experiments) are often performed at a large scale—specimens weighing 50 tons are not uncommon—and require specialized facilities. Thus, such multi-component hybrid experiments will typically require coupling over multiple geographically distributed sites.

A large-scale distributed hybrid experiment is fundamentally about *sharing heterogeneous resources* (simulation, experimental apparatus), each owned and controlled by a different institution, and integrating them so as to enable a collaborative experiment to take place. Thus, a distributed hybrid experiment maps well into the concept of a virtual organization [8] and would appear ideally suited to the application of Grid technology. Recognizing this, we have created a Grid-based framework for

Published in: *Proc. 13th IEEE Symposium on High Performance Distributed Computing (HPDC-13), 2004.*

conducting distributed hybrid experiments. Building on mechanisms provided by the Globus Toolkit's implementation of the Open Grid Services Infrastructure (OGSI) specification [6], we have created domain-specific Grid services, platform-specific interfaces, and user interface tools that make it possible to construct, perform and monitor distributed hybrid earthquake engineering experiments conducted across geographically and organizationally distributed sites with heterogeneous equipment and policy. This framework is called NEESgrid [11, 12, 18] and is being deployed and used as part of the NSF-funded Network for Earthquake Engineering Simulation (NEES).

In this paper, we describe this framework and explain how it both enables secure, reliable distributed hybrid experiments and provides a secure, consistent, uniform collaborative environment for remote participation in such experiments. We describe our integration of teleobservation and teleoperation capabilities, data and metadata services, and Grid protocols and technologies: specifically, the Globus Toolkit version 3 implementation of OGSI and associated security protocols [20]. We also describe the application of our framework within the recent Multi-site Online Simulation Test (MOST) [2], an unprecedented experiment that linked physical facilities at two sites (Colorado and Illinois) and a computational simulation.

Our focus in this paper is on the architecture and engineering of the NEESgrid software system and our experiences with its application to challenging earthquake engineering problems. We provide this report with the goal of communicating to the larger community the requirements that arise in this field, the techniques that we have found useful in addressing these requirements, and the lessons that we have learned in practical settings.

2. Experiment Architecture

The complete NEESgrid system comprises a variety of different resource types, including experiment facilities, data repositories, and computers used for simulations. In order to capture these different functions while also exploiting commonality when it is present, we have designed the NEESgrid software as a service-oriented architecture. The various functionalities required to implement a complete distributed experiment are expressed as service interfaces, and a particular resource is defined by the service interfaces that it supports. We describe in this section a subset of those interfaces that are particularly relevant to distributed hybrid experiments, namely those concerned with controlling remote experimental equipment, monitoring the progress of an experiment, acquiring experimental data from local site-specific data acquisition equipment, and making that data available to remote experimenters. Our implementations of the resulting services make good use of OGSI mechanisms, such as soft state management and service data elements. In addition, communications within the NEESgrid system are securely authenticated and authorized via the use of Grid Security Infrastructure (GSI) mechanisms [5, 7, 20]. NEESgrid also makes use of components from the NSF Middleware Initiative (NMI) software release.

2.1. Control Components

A crucial observation underlying the NEESgrid software architecture is that, from the perspective of a hybrid experiment, a physical experiment and a computational simulation are indistinguishable. Thus, we define a single Grid service interface, the NEESgrid Teleoperations Control Protocol (NTCP) [15], that can be used to interact with either. NTCP provides for remote access to control systems (e.g., servo-hydraulic systems) and simulated control systems (e.g., computational simulations that model the actions of servo-hydraulic systems on experiment specimens). This strategy of making both numerical and physical simulations accessible through the same service interface has many advantages. For example, it allows us to first test hybrid experiments with purely simulation components and then seamlessly replace the simulation components with physical simulations. This flexibility has proven invaluable in MOST, described in Section 3.

The design goals for NTCP were driven by characteristics of physical earthquake engineering experiments. The simulation tools used to design an experiment and implement computational components will vary from experimenter to experimenter and may range from high-level workbenches such as MATLAB to simulations written in C or Fortran. Furthermore, the control systems used to drive the physical experiments (e.g., by positioning hydraulic actuators) vary from facility to facility. Facility managers want to retain some control over what commands are acceptable (e.g., to set limits on the amount of force that can be applied on the local specimen, and to be able to terminate the

local experiment at any time). Finally, it may be impossible to "undo" an action in a physical experiment without tearing down the specimen and rebuilding it, an expensive and time-consuming process. Thus, we concluded that NTCP must:

- provide a uniform control interface, separating the definition of these control interfaces from details of specific control systems or simulation,
- control both physical experiments and computational simulations, enabling experimenters to switch easily between the use of physical experiments and computational simulations;
- support fault-tolerance, so that transient problems (such as network interruptions) during a distributed hybrid experiment need not cause the experiment to terminate; and
- allow for the negotiation of request parameters prior to execution, so that a client may discover in advance whether a request would violate a site's local policy or cause physical damage.

These requirements led to the design of a transaction-based [9] protocol. NTCP is designed and implemented as an OGSI compliant Grid Service, and as such can leverage Grid security models, lifetime management, and state observation. As shown in Figure 1, an NTCP operation is initiated by a client sending a *proposal* (a set of requested actions) to an NTCP server. If a proposal is accepted, the client can complete the transaction by issuing an *execute* command to cause the proposed action to occur. This separation of proposal and execution enables a client to ensure that the actions for a testing step are acceptable at all experimental sites before causing any action to take place. If any of the requested proposals is rejected, the client may send a request to *cancel* the transaction. When the execution completes, the client receives the transaction results, which can be used to determine the desired set of actions for the next time-step. The NTCP protocol supports at-most-once semantics, so that if a client makes a request and does not receive a reply, the client can re-send the request without any danger of the same action being executed twice.

Each transaction in the NTCP server is represented by an OGSI service data element [6] that includes the transaction name and state, the requested actions and timeout values specified in the proposal that resulted in the creation of the transaction, the transaction results (if available), and timestamps representing each state change in the lifetime of the transaction. Thus, OGSI inspection mechanisms can be used to query the state of any transaction. In addition, a service data element representing the "most recently changed" transaction can be used to monitor the behavior of the server as a whole.

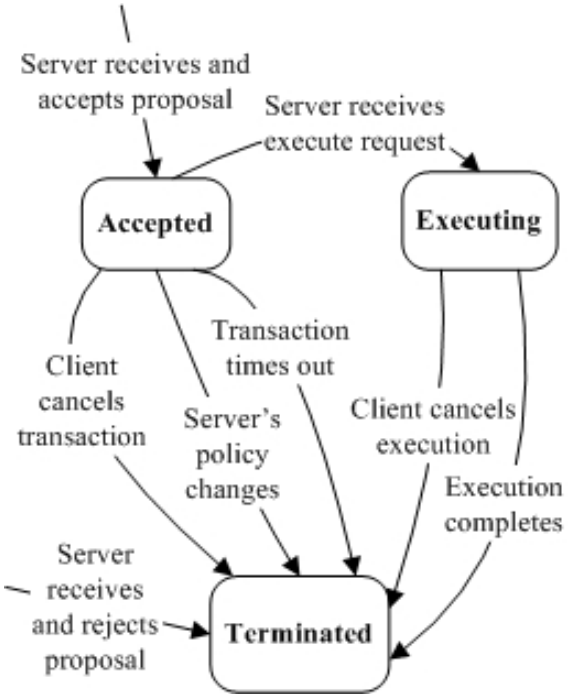

Figure 1: State transitions in NTCP

To facilitate the use of NTCP across different control and simulation environments, we structured the implementation as shown in Figure 2, with a core NTCP server implementing the generic parts of the NTCP service, such as managing the transaction state, and a control plugin interface [16]. The implementation of the plugin is responsible for mapping NTCP requested into appropriate actions in the local site's control system or simulation engine.

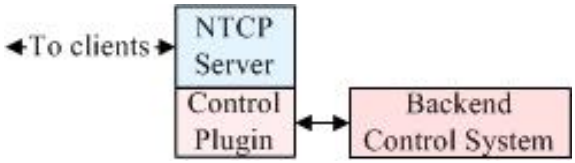

Figure 2: NTCP server and control plugin

2.2. Remote Monitoring Components

For a multi-site experiment, remote observers need the ability to see what is happening and observe data. NEESgrid includes a telepresence system [4],

which uses commodity hardware and software to provide a video feed and basic camera control (pan/tilt/zoom) to remote observers, providing a general sense of lab activity, and two mechanisms for accessing experimental data. The NEESGrid Streaming Data Service (NSDS) [13] provides a best-effort stream of real-time data from the data acquisition (DAQ) system. In addition, the complete data set can be accessed following completion of each time step via the NEESGrid data and metadata repository, as we describe in the next subsection.

2.3. NEESgrid Data and Metadata Repository

Earthquake engineering experiments often produce more data than can be streamed reliably in real-time. In addition, data may be of interest after an experiment has completed. Thus, NEESgrid includes a *data and metadata repository* for storing and providing access to experiment data. This repository and associated NEESgrid services allow data and metadata from an experiment to be archived incrementally by an *ingestion tool* as an experiment is run; researchers can later download this data for analysis or visualization.

The NEESgrid data and metadata repository uses GSI for authentication and GridFTP for file transport. Metadata objects are managed using the NEESgrid Metadata Service (NMDS), and files are managed using the NEESgrid File Management Service (NFMS). These components are coupled using the Façade pattern, but may be used independently.

NMDS is used to create, update, manage, and validate metadata and metadata schemas; it differs from most other metadata management systems in that metadata schemas are represented by first-class objects and can be managed just like any other object. In addition, it supports per-object version control and authorization. We plan to add support for the Community Authorization Service [17].

NFMS provides two main capabilities: logical file naming and transport neutrality. Applications negotiate file transfers with NFMS, which resolves a transfer request for a logical file to a protocol request for a physical resource. NFMS uses GridFTP to provide transport and has a plug-in API that allows other transport protocols to be used if desired.

We have also developed an ingestion tool to upload data and metadata to the repository as an experiment is run and a servlet that acts as a bridge between GridFTP and https.

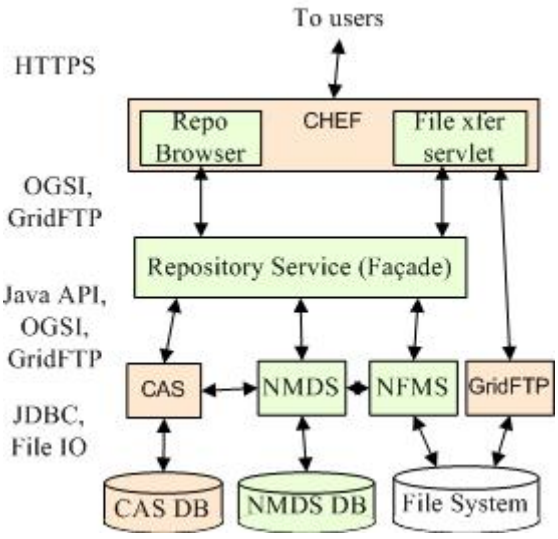

Figure 3: NEESgrid Repository Architecture

3. Case Study: The MOST Experiment

The Multi-Site Online Simulation Test (MOST) distributed hybrid experiment took place on July 30, 2003 [2]. This large-scale experiment linked physical experiments in the Newmark Civil Engineering Laboratory at the University of Illinois at Urbana-Champaign (UIUC) and at the Structures and Materials Testing Laboratory at the University of Colorado, Boulder (CU) with a numerical simulation at the National Center for Supercomputing Applications (NCSA), also in Urbana-Champaign.

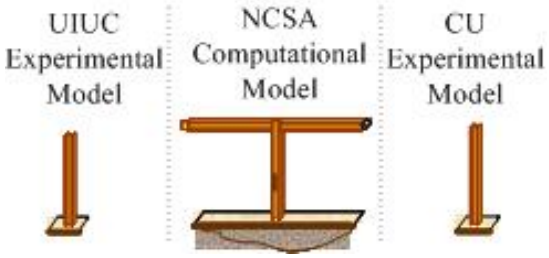

Figure 4: MOST Experiment Structure

The structure used in the experiment (Figure 4) represents a two-bay single-story steel frame, like that of the interior of a multistory building. To distribute the test structure we applied a method called Multi-Site Pseudo-Dynamic Substructure (MS-PSDS) testing [19] in which the structure to be tested is divided into various substructures, each of which is physically tested or numerically simulated at the

same time at a different location. A simulation coordinator controls the overall experiment and communicates with the test sites and simulation computers. This technique allows for testing a wide range of large structures that might otherwise be beyond the capabilities of many laboratories.

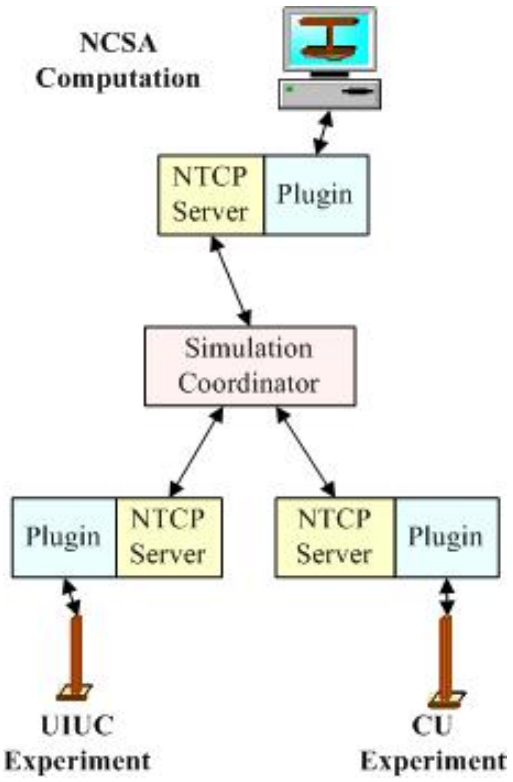

Figure 5: Modular framework of MOST

Figure 5 illustrates how NEESgrid services are used to implement the MS-PSDS methodology. Specifically, NTCP was used to integrate each test structure. A *Simulation Coordinator* provides overall management of the experiment. This component repeatedly issues a set of NTCP proposals based on current simulation state, collects information about the resulting state of all the substructures, and, based on that resulting state, computes the next set of NTCP commands to send. The coordinator also handles exceptions such as lost network connections or invalid responses. To help manage complexity, MOST was developed incrementally. First, we implemented and tested a distributed simulation-only experiment. Once the correctness of the distributed simulation was verified, two of the numerical simulations were replaced with physical substructures. The use of NTCP made this substitution transparent to the coordinator.

The physical experiments are shown in Figure 6 and Figure 7. The left column of the experimental frame was tested at UIUC and the right column at CU. The column is a cantilever column because of the beam-column pin connection that connected the right hand column to the frame to the simulated horizontal beam. Like the UIUC column, the CU column was tested in a horizontal position; however, it was rigidly connected to a vertical supporting steel structure suppressing all translational and rotational degrees of freedom. The central section of the frame was modeled by a simulation performed at NCSA on a Pentium 2.4 GHz Windows machine with 512 MB of memory. (This simulation was performed at NCSA primarily to exercise the deployment at NCSA and to further distribute the experiment.)

Figure 6: The physical substructure test at the University of Illinois

For each time step simulated in MOST, force data was fed to the computational model at NCSA; the correct displacements were calculated and sent to the Illinois and Colorado physical test sites; displacements were applied to the physical models;

and forces for the next iteration were measured and sent back to the computational model at NCSA. This cycle was repeated 1,500 times during the five hour experiment.

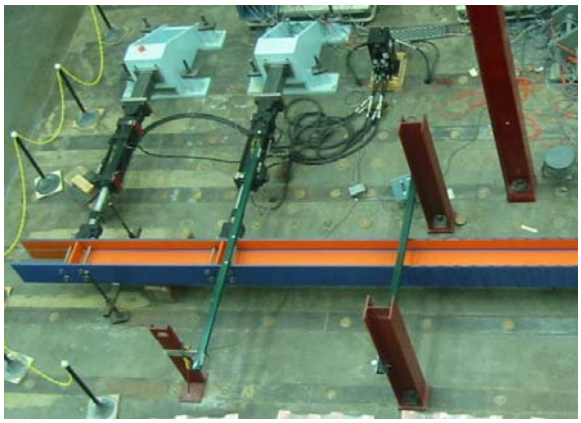

Figure 7: The physical substructure test at the University of Colorado

During the experiment, the structural response was streamed to remote users and simultaneously stored in the main data repository for archiving. To observe the test and collaborate with others, users remotely accessed tools via logging in to MOST via a NEESgrid specific collaboration interface built using the CHEF collaboration framework [1]. The CHEF interface used the various NEESgrid protocols to authenticate to NEESgrid resources, access the metadata catalog and download experimental data so that it could be viewed immediately by remote participants. CHEF also provided a range of useful collaboration tools such as a message board, access to an electronic notebook and an interactive chat.

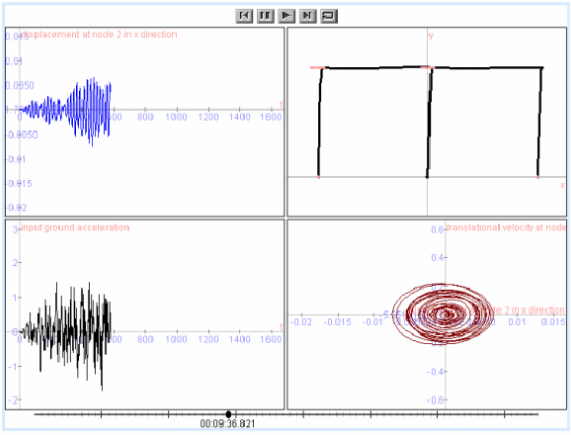

Figure 8: CHEF data viewers

Figure 8 shows some of the data viewers available via the CHEF interface These viewers provided near real-time visualization of Figure the structure response, time services data from a sensor, as well as hysteresis plots. Arrangements of one or more views can be saved or viewed, and the Data Viewer automatically organizes a given arrangement to allow users to see each of the views. At the top of the Data Viewer, a set of VCR buttons allows users to play, pause, rewind, and fast-forward the data viewer, while at the bottom a clickable timeline allows users to see the state of the Data Viewer at any given time point.

During MOST, real-time video from both of physical testing sites was also available, with at least one accessible camera at each site. To access the camera at either Colorado or UIUC, users could click on the appropriate Video button.

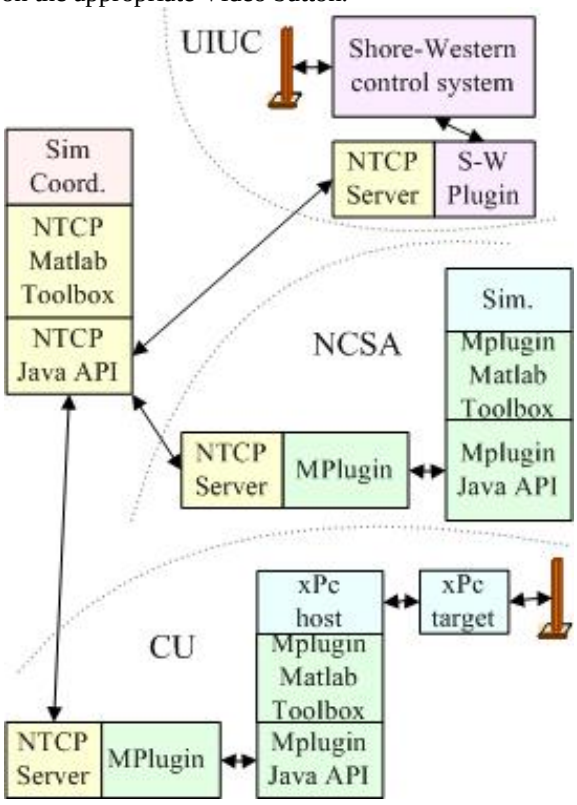

Figure 9: Control components used in MOST

3.1. MOST Software Configuration

Details of the configuration of the NTCP control structure used in MOST are shown in Figure 9. The simulation coordinator, on the left, was written by an earthquake engineer using a Matlab toolbox that we

developed to provide a convenient interface to NTCP; this toolbox in turn called the NTCP Java API to send requests to the remote NTCP servers.

At each time-step, the simulation coordinator sends requests to the NTCP servers at UIUC, NCSA, and CU. Each NTCP server does some generic processing (e.g., state management) and then calls a plugin to perform actions.

At UIUC, the NTCP server was configured to use a plugin that communicated, via a simple TCP/IP protocol, with a Shore-Western control system, which in turn controlled the UIUC servo-hydraulics.

At NCSA, the NTCP server was configured to use a plugin (called the "Mplugin") that communicated with the Matlab simulation. In this case, instead of pushing requests out to the back-end as they were received, the plugin buffered requests and implemented a separate service to provide information about them. The Matlab simulation running at NCSA would then poll that service for requests; when the simulation received a request, it would perform an appropriate computation then call the plugin-implemented service to notify the NTCP server of the results.

The CU NTCP server was configured to use the same plugin code used by NCSA; however, instead of processing requests by performing computations, the CU Matlab application used Matlab's xPC feature to communicate with a target machine running Matlab's real-time operating system, which would in turn control the servo-hydraulics at CU.

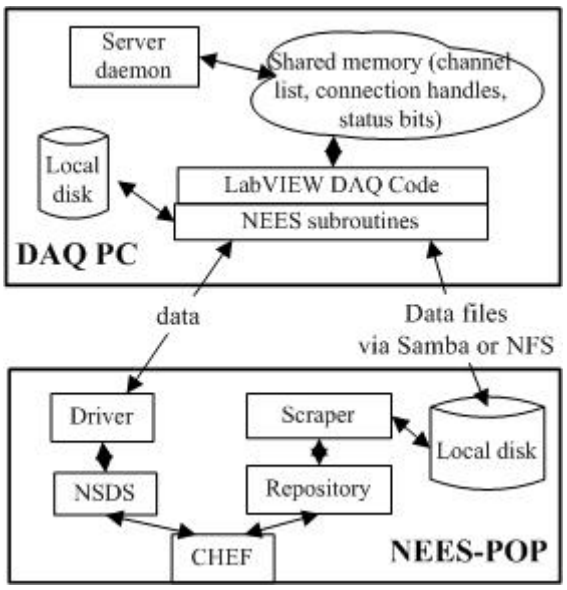

Figure 10: Major DAQ components

3.2. Remote Monitoring in MOST

Sensor data from the two physical experiments were collected by a local data acquisition (DAQ) system. Conveniently, both sites choose LabVIEW as the software for their data acquisition. Thus, to interface the DAQ to NEESgrid, a simple LabVIEW interface was built that ran at the UIUC and Colorado sites and periodically gathered data deposited by the DAQ in a network-mounted file system; NFMS and GridFTP were then used to upload it securely to a NEESgrid accessible data repository. Once there, the combined data could be visualized using the CHEF-based data viewer. The same strategy was used to capture data generated by the simulation at NCSA.

3.3. Metadata in MOST

For MOST, metadata was mostly generated manually and data was generated automatically from sensors. Experimenters developed metadata that described each of the three components of the experiment in terms of the structural configuration, material properties, and instrumentation, and uploaded the metadata to the repository prior to the experiment. The metadata was designed so that non-participants viewing the stored data can understand the meaning of the sensor data in the context of the experiment..

An early version of the NEESgrid data and metadata repository was used for MOST. The experiment served to exercise the data functionality and helped identify areas to be more fully developed in later releases, such CAS-based access control.

3.4. MOST Results

The results from MOST can be divided into two categories: the hybrid simulation experiment, and user interaction and participation.

The full, 1500-timestep distributed experiment was actually run twice: once as a "dry run" of the components directly involved in the simulation (the NTCP servers, physical experiments, and simulations), and then as the full experiment, available for viewing by remote participants. The dry run took about 5.5 hours and ran successfully to completion. The public experiment ran for more than 5 hours but exited prematurely at step 1493 (out of 1500). The fault tolerance features of NTCP enabled the simulation to detect and recover from several transient network failures throughout the day;

however, the simulation coordinator had not been coded to take advantage of all the fault-tolerance features, and a final network error caused the simulation to terminate prematurely.

During the execution of the experiment, over 130 remote participants logged on to observe MOST. CHEF's chat feature was crucial to user interaction. It allowed developers to communicate with one another, while keeping other participants informed of status and progress. The sense of participation of the remote users was enhanced by the three telepresence cameras, which could be operated remotely.

3.5. Mini-MOST

Once MOST was complete, there was a desire for a less-expensive, self-contained version that could be installed into an average lab. Mini-MOST (Figure 11) is a tabletop-sized system, with a single (1m by 10cm) beam, using stepper motors. It is an emulation of the UIUC portion of MOST and provides an excellent platform for education, training, and outreach for NEESgrid.

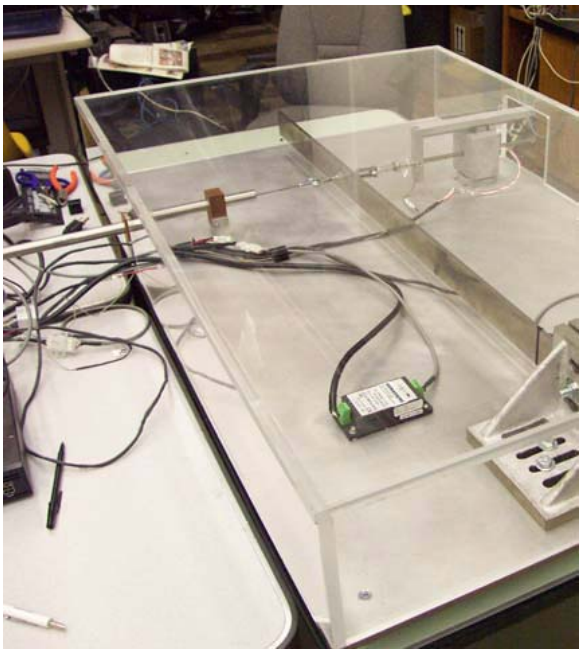

Figure 11: Mini-MOST

The control and DAQ are run from a single Windows-based PC, which can also host the MATLAB simulation coordinator if required. Sensors are also scaled back to a strain gauge, LVDT for position, and a load cell for force. In the first version, a single 24lb through-hole stepper motor was used. Work is currently underway to add the second stepper motor and a rotation sensor to more accurately model the full scale MOST experiment.

Other than scale differences, the main software change was a new NTCP plugin to communicate with LabVIEW. The second substantial change is in the simulation coordinator: the smaller beam has different mass, spring constant, inertia and so forth. We made small changes to the MATLAB code to accommodate these differences.

For simulation and debugging, we have test code corresponding to the MOST variations. We also have a program where the beam is replaced by a first-order kinetic simulator. It is also applicable for testing when the actual hardware is not available.

The control code is developed in LabVIEW, with a daemon program for NTCP communications.

4. Security Considerations

Telecontrol incurs serious health and safety risks, as well as the risk of damaging expensive equipment [10]. We provide several mechanisms to help alleviate these risks: the usual Grid-based authentication and access control [5, 7], and the ability in NTCP for sites, through the control plugin mechanism, to enforce limits on what actions are allowed. We have also designed our services in such a way that the actual control systems do not need direct access to the external Internet.

However, because our NTCP implementation was not designed as provably secure software, and because NTCP and related components run on commodity operating systems (Linux and Windows), it is the responsibility of the experiment sites to employ appropriate operational procedures. In the case of MOST, these procedures included powering up the servo-hydraulics only when no one is near the experiment specimen, running a plugin/backend system that required a human to approve each action (used only during initial testing at UIUC), and, whenever the servo-hydraulics are powered up, always having engineers nearby monitoring the experiment and prepared to turn it off if necessary. The small-scale nature of Mini-MOST makes the associated health and safety risks far smaller; the primary precaution taken was the creation of a plexi-glass cover for the Mini-MOST apparatus.

5. Ongoing Work

The software used in MOST was released in October 2003, and several new experiments using the NEESgrid framework are being planned.

A UCLA team of earthquake engineers plan to perform field testing of a four-story office building in Los Angeles. They intend to apply earthquake-type and harmonic force histories to the building, gathering acceleration, strain, and displacement data using wireless sensor arrays (802.11 wireless telemetry) to evaluate response and behavior. Data and video streams will be recorded and archived at a mobile command center before transmission to the laboratory using satellite telemetry.

Earthquake engineers at RPI, UIUC and Lehigh University plan to use the NEESgrid framework to study soil-structure interaction in an experiment involving two structural sites (UIUC and Lehigh), one geotechnical site (RPI), and a computational simulation node at NSCA. The experiment will focus on an idealized model of the Collector-Distributor 36 of the Santa Monica Freeway that was damaged in the 1994 Northridge earthquake of California.

Engineers at UC Davis are working on an experiment that uses the NEESgrid framework to characterize how the properties of soil change during shaking or ground improvement. This experiment includes remote operation of a robot arm that will be attached to their centrifuge and of piezo-electric bender element sources and receivers embedded within the centrifuge model. The robot arm has exchangeable tools: a stereo video camera tool for telepresence, an ultrasound tool for imaging, a cone penetrometer, a needle probe for high resolution imaging, and a gripper tool for installation of piles and manipulation/loading.

At the University of Minnesota, an experiment is planned that will use the NEESgrid framework to operate a six-degree-of-freedom controller, to apply realistic deformations and loading quasi-statically to large-scale structures. This experiment will also use video and still images as data, using the NEESgrid framework to trigger still image capture.

MOST and most follow-on experiments have lax performance requirements; even long delays can be tolerated without affecting results. We are working with engineers from UC Berkeley, the University of Colorado, SUNY-Buffalo, the University of Minnesota, and Lehigh University to support distributed experiments with near-real-time requirements. This work has two facets: we are working on improving NTCP performance, while the earthquake engineers are developing simulation and control software that can better tolerate delays.

6. Conclusion

Hybrid earthquake engineering tests produce more accurate models, for some complex structures, than physical or computational tests alone. Distributed hybrid earthquake engineering tests produce more accurate models for structures that are even more complex that those that can be evaluated by the more simple hybrid methods. As such, they are important tools for helping improve the quality of our physical infrastructure. The distribution of these experiments is not gratuitous, but is a direct consequence of the scale of the experiments and the different types of testing and simulation modalities required: e.g. coupling of soil response measured by a scale model on a centrifuge and beam response measured by large-scale structure stressed by hydraulic actuators requires fundamentally different test facilities. From this perspective, hybrid testing is a prototypical Grid application, as resource sharing is an essential aspect of the experiment.

The experience of working on MOST with earthquake engineers reinforced our views on the importance of fault-tolerance in a telecontrol service. It has also demonstrated that having support for fault tolerance in the service isn't enough; domain scientists will generally need some guidance in pushing these features to the outer edges of the system (that is, to the clients and to the back-end simulations and control systems).

NEESgrid is an important new test facility. A Grid-based framework for distributed hybrid tests makes these tests more practical to design and perform, providing tools to deal with heterogeneity and policy issues. In addition, as part of the exercise, we have demonstrated the effectiveness of the service oriented architecture and the stateful service model that is at the core of OGSI. We believe that many of the technologies that we have developed for NEESgrid will have application in domains outside of earthquake engineering. For example, NTCP and NSDS can be used to control and observe a wide range of devices, and we plan to investigate this in the setting of other remote sensing and control applications such as tele-microscopy. In summary, NEESgrid has the potential to be of great value to the earthquake engineering community as well as representing an important class of Grid application.

7. Acknowledgements

This work was supported by the George E. Brown, Jr. Network for Earthquake Engineering Simulation (NEES) Program of the National Science Foundation, Awards CMS-0117853 (NEESgrid), CMS-0217325 (NEES MUST-SIM, UIUC), and CMS-0086592 (NEES FHT, CU-BOULDER).

We gratefully acknowledge the contributions of Dan Abrams, Cristina Beldica, Ian Buckle, Ben Clifford, Mike D'Arcy, Amr Elnashai, Tom Finholt, David Gehrig, Dan Horn, Erik Johnson, Young Suk Kim, Dan Kuchma, Lee Liming, Ravi Madduri, Doru Marcusiu, Gilberto Mosqueda, Narutoshi Nakata, Gokhan Pekcan, Pawel Plaszczak, Tom Prudhomme, Chase Phillips, Andrei Reinhorn, Hatem A Seliem, Benson Shing, Eric Stauffer, Bozidar Stojadinovic, Guangqiang Yang, and Nestor Zaluzec.

8. References

1. CompreHensive collaborativE Framework (CHEF) Project Web Site, 2004. http://chefproject.org.
2. The MOST Experiment, July 30, 2003. NEESgrid, Technical Report, 2003. http://www.neesgrid.org/documents/MOST_document_v1.0.pdf.
3. Allcock, W., Bester, J., Bresnahan, J., Chervenak, A.L., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S., Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing. *Mass Storage Conference*, 2001.
4. Bonkalski, J., Anderson, R., Jones, S. and Zaluzec, N. Bringing TelePresence Microscopy and Science Collaboratories into the Class Room. *TeleConference Magazine*, *17* (9). 1998.
5. Butler, R., Engert, D., Foster, I., Kesselman, C., Tuecke, S., Volmer, J. and Welch, V. A National-Scale Authentication Infrastructure. *IEEE Computer*, *33* (12). 60-66. 2000.
6. Foster, I., Kesselman, C., Nick, J.M. and Tuecke, S. Grid Services for Distributed Systems Integration. *IEEE Computer*, *35* (6). 37-46. 2002.
7. Foster, I., Kesselman, C., Tsudik, G. and Tuecke, S., A Security Architecture for Computational Grids. *5th ACM Conference on Computer and Communications Security*, 1998, 83-91.
8. Foster, I., Kesselman, C. and Tuecke, S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of Supercomputer Applications*, *15* (3). 200-222. 2001.
9. Gray, J., The Transaction Concept: Virtues and Limitations. Proceedings of the Fifth Symposium on Reliability in Distributed Software and Database Systems, 1981, 144-154.
10. Johnston, W. Realtime Widely Distributed Instrumentation Systems. Foster, I. and Kesselman, C. eds. *The Grid: Blueprint for a New Computing Infrastructure*, Morgan Kaufmann, 1999, 75-103.
11. Kesselman, C., Butler, R., Foster, I., Futrelle, J., Marcusiu, D., Gulipalli, S., Pearlman, L. and Severance, C. NEESgrid System Architecture. NEESgrid, Technical Report, 2003. http://www.neesgrid.org/documents/NEESgrid_SystemArch_v1.1.pdf.
12. Kesselman, C., Foster, I. and Prudhomme, T. Distributed Telepresence: The NEESgrid Earthquake Engineering Collaboratory. *The Grid: Blueprint for a New Computing Infrastructure (2nd Edition)*, Morgan Kaufmann, 2004.
13. Kesselman, C., Pearlman, L. and Mehta, G. Design for NEESgrid Telepresence Referral and Streaming Data Services. NEESgrid, Technical Report NEESgrid-2003-09, 2003. http://www.neesgrid.org/documents/TR_2003_09.pdf.
14. Nakashima, M., Kato, H. and Takaoka, E. Development of real-time pseudo dynamic testing. *Earthquake Engineering and Structural Dynamics, 21*. 79-92. 1999.
15. Pearlman, L., D'Arcy, M., Johnson, E., Kesselman, C. and Plaszczak, P. NEESgrid Teleoperation Control Protocol (NTCP). NEESgrid, Technical Report NEESgrid-2003-07, 2003. http://www.neesgrid.org/documents/TR_2003_07_v0.4.pdf
16. Pearlman, L., D'Arcy, M., Plaszczak, P. and Kesselman, C. NTCP Control Plugin. NEESgrid, Technical Report NEESgrid-2003-16, 2003. http://www.neesgrid.org/documents/TR_2003_162.pdf.
17. Pearlman, L., Welch, V., Foster, I., Kesselman, C. and Tuecke, S., A Community Authorization Service for Group Collaboration. *IEEE 3rd International Workshop on Policies for Distributed Systems and Networks*, 2002.
18. Prudhomme, T., Kesselman, C., Finholt, T., Foster, I., Parsons, D., Abrams, D., Bardet, J.-P., Pennington, R., Towns, J., Butler, R., Futrelle, J., Zaluzec, N. and Hardin, J. NEESgrid: A Distributed Virtual Laboratory for Advanced Earthquake Experimentation and Simulation: Scoping Study. NEESgrid, Technical Report NEESgrid-2001-01, 2001. www.neesgrid.org.
19. Watanabe, E., Sugiura, K., Nagata, K., Yamaguchi, T. and Niwa, K., Multi-phase Interaction Testing System by means of the Internet. *1st International Conference on Advances in Structural Engineering and Mechanics*, Seoul, Korea, 1999, 43-54.
20. Welch, V., Siebenlist, F., Foster, I., Bresnahan, J., Czajkowski, K., Gawor, J., Kesselman, C., Meder, S., Pearlman, L. and Tuecke, S., Security for Grid Services. *12th IEEE International Symposium on High Performance Distributed Computing*, 2003.