CONCURRENCY AND COMPUTATION: PRACTICE AND EXPERIENCE
Concurrency Computat.: Pract. Exper. (2014)
Published online in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/cpe.3234

SPECIAL ISSUE PAPER

Batchsubmit: a high-volume batch submission system for earthquake engineering simulation[‡]

Anup Mohan[1], Thomas Hacker[2,*,†], Gregory P. Rodgers[3] and Tanzima Islam[4]

[1]*Electrical & Computer Engineering, Purdue University, West Lafayette, IN, USA*
[2]*Computer & Information Technology, Purdue University, West Lafayette, IN, USA*
[3]*Advanced Micro Devices (AMD) Research, Austin, TX, USA*
[4]*Lawrence Livermore National Laboratory (LLNL), Livermore, CA, USA*

SUMMARY

Network for Earthquake Engineering Simulation (NEES) is a network of 14 earthquake engineering labs distributed across the USA. As a part of the NEES effort NEESComm operates a comprehensive cyberinfrastructure that consists of the *NEEShub* and the *NEES Project Warehouse*. NEESComm provides consistent access to several high performance computing (HPC) venues. These venues include Extreme Science and Engineering Discovery Environment, the Open Science Grid, Purdue Supercomputers, and NEEShub servers. In this paper, we describe the system we developed, *batchsubmit*, which allows NEES researchers to make use of all these venues through the NEEShub science gateway. Copyright © 2014 John Wiley & Sons, Ltd.

Received 7 November 2013; Revised 27 January 2014; Accepted 29 January 2014

KEY WORDS: high performance computing; simulation; earthquake engineering

1. INTRODUCTION

1.1. Network for Earthquake Engineering Simulation overview

The George E. Brown Network for Earthquake Engineering Simulation (NEES), funded by the National Science Foundation, is focused on accelerating research in the field of seismic infrastructure design to reduce the impact of earthquakes and tsunamis on the built environment. The NEES network of facilities is comprised of 14 laboratories across the USA focused on performing advanced experimental work.

Traditional experimentation for testing structural resilience has been primarily focused on the physical simulation of the effects of earthquakes and tsunamis on test structures using infrastructure such as geotechnical centrifuges and shake tables. To supplement this physical experimentation, the use of computational simulation of ground motion and structural effects using computational finite element methods has grown significantly in the last decade thanks to better simulation and modeling techniques.

The Network for Earthquake Engineering Simulation (NEES) provides access to a suite of analysis and simulation software through the *NEEShub* [1–3], a science gateway for the NEES research community [1–3]. In addition to these software tools, NEES also provides a data repository called the *Project Warehouse* that allows researchers to upload, archive, and disseminate data from their

*Correspondence to: Thomas Hacker, Computer & Information Technology, Purdue University, West Lafayette, IN, USA.
†E-mail: tjhacker@purdue.edu

‡Categories and Subject Descriptors
category D.2.6

physical, cyber, and hybrid experiments and simulations. The data in the Project Warehouse includes results obtained from experiments carried out in both NEES and non-NEES sites. The Project Warehouse provides users with tools to organize data into projects, experiments, trials, and results. The NEEShub also provides tools such as *inDEED* to visualize the data obtained from experiments. The data in the Project Warehouse can be cited through the use of a Digital Object Identifier and shared with researchers across the world or only within a research group. To ensure the long-term preservation of valuable NEES data, NEES also provides a facility and service to curate the data in the repository for long-term access and analysis.

The NEES cyberinfrastructure is working to disseminate knowledge and access to tools for civil engineering researchers and practitioners across the world. The NEEShub has been accessed by over 100,000 users over the past year [4], and since its inception in 2009, it has been used by users from 211 countries.

An example of our efforts is a joint project with researchers from Oxford University (England) and the Joint Research Centre in Ispra, Italy to develop a data system (named *Celestina Data*) to allow data sharing between the NEES community and the European earthquake engineering community [5].

In this paper, an extended revision of a publication presented at the Extreme Science and Engineering Discovery Environment (XSEDE) 13 conference [6], we describe the system we developed and some of our experiences in facilitating access to high performance computing resources for the NEES community.

1.2. The need for high performance computing simulation

The development and simulation of finite element analyses of the effects of earthquakes and tsunami on structures is inherently complex. On a personal computer, it can take days or weeks to complete a set of simulations. An efficient way to overcome this problem is to use high performance computing (HPC) resources to run a large number of simulations quickly (i.e. parameter sweep) or to simulate a large model that requires a large number of tightly coupled processors. To provide facilitated access to the research community, NEES utilizes HPC venues made available by the NSF XSEDE project and the Open Science Grid (OSG), as well as local Purdue supercomputers. For the past 4 years, NEES has maintained allocations on XSEDE resources Kraken, Stampede, and the Open Science Grid and participates in the Purdue Community Cluster program to access the Hansen and the Carter systems. Access to HPC resources can allow researchers to more thoroughly explore a problem in significantly less time by utilizing these resources to rapidly execute a large number of high volume simulations.

1.3. Developing a high-volume batch submission system

The design developed by the NEES IT team for the high-volume batch submission system posed several challenges that the team worked to solve. The basic requirements for the system involved several steps needed to submit a simulation job to a remote HPC venue.

- Upload all input files to the remote HPC venue.
- Ensure that the simulation program is available on the remote venue and copy the software to the venue if necessary.
- Prepare a venue specific script file specifying parameters for job submission, for example, the PBS file.
- Submit the job to the correct job class and wait for the result.
- Download the resulting output files and job status to a local system for analysis and visualization.

Repeating this process manually for multiple simulations is tedious and time consuming. Hence, we needed an easier way to execute a large number of simulation jobs by executing all the above steps automatically and consistently for all available venues.

Each HPC resource requires a slightly different set of commands and parameters for executing simulation jobs. Keeping track of all these variations for different resources is a cumbersome task. The level of knowledge needed to submit jobs to the variety of HPC venues was significant enough to create a barrier that limited the adoption and use of these resources by the user community. Hence, there was a need to simplify the process of accessing and using HPC resources for the NEES

DOI: 10.1002/cpe

community. Based on the IT team's assessment of the needs of community, it was clear that the users needed to be able to simply select an HPC system for running the simulation, and the batch submission system would take care of all of the other details automatically as much as was possible.

Another problem we encountered was the need for a common community account for each HPC resource. Multiple HPC resources are available for running simulation jobs. However, to access resources, a user account is needed on each system. Managing accounts on different resources is difficult and costly. Instead of creating accounts for each user and managing them separately, a better solution was to create a single community account on each of the HPC resources. The users can be added to the community account of a resource if they want to execute their jobs on that resource. Thus, the user is relieved from the job of managing their account and allocations, for example, XSEDE allocations on different resources and can execute their jobs without any concerns.

The accessibility of the HPC resources can be increased if they could all be accessed from a common location. Hence, the NEEShub was decided to be used as a science gateway to mediate access to HPC resources. The HPC resources needed to be integrated to the NEEShub environment and the hub interface.

Another challenge was the need to increase the number of HPC resources available for the user to submit their jobs. Apart from XSEDE venues, other venues such as Open Science Grid, Purdue resources like Hansen and Carter were added. This increased the choice of venues for the users. In case some of the venues are down for maintenance, users can easily migrate to other venues.

Typically when a user submits a job, the session managing the job needs to be active until the job finishes execution. If the session is terminated, the job will be lost. This synchronous method of job submission limits the ability of users to submit and run a large number of jobs when synchronous jobs take a long time to finish. An asynchronous job submission approach grants more flexibility to the user. In asynchronous submission, once the job is submitted, the user can terminate the session or perform other tasks. When the job finishes execution, the result will be copied automatically, and the user can analyze the result at a later time.

Each user might have their own version of a software package (e.g.: OpenSees [7]). Users need access to a local copy of the software, and it is common for the users to modify the input files between submissions. Manually copying all the required files for each job submission is an elaborate task for most users. Hence, there was a need to automatically send the current version of the executable and all the input files to the remote HPC resource for execution.

Another common scenario is that users might want to submit multiple jobs by changing just the input parameters to implement something like a parameter sweep. Hence, a feature to submit multiple jobs with different input parameters is useful for researchers.

Many NEES researchers use OpenSees for simulation. However, there are quite a few cases when they use other software such as MATLAB [8] or OCTAVE [9] for simulation. It would be useful to execute these simulations on different platforms automatically using the same job submission system so that the user does not have to worry about platform specific configuration needed to execute jobs.

The research community extensively uses the *Project Warehouse*, which is the NEES data repository. To add simulation output to the Project Warehouse, a feature was needed to accelerate the simulation cycle that would automatically recover the output from simulation and analysis and upload it into the NEES Project Warehouse.

Based on the needs of the user community, and the challenges we identified, it was clear that an automatic and easily usable job submission system was needed by the NEES community to effectively utilize the HPC resources under XSEDE.

2. RELATED WORK

2.1. Related cyberinfrastructure efforts

In related work, the iPlant project [10] developed a cyberinfrastructure focused on meeting the needs of the biology community working to address grand challenges in plant sciences. The iPlant cyberinfrastructure is comprised of components that are tailored to meet the specific science needs

DOI: 10.1002/cpe

of the users. For example, *DNA Subway* provides a workflow-oriented system to help guide users through the use of the cyberinfrastructure to solve specific genomic problems, such as phylogenetic analysis. The NEES cyberinfrastructure is similar to iPlant in that it employs a science domain-oriented approach that provides a user interface tailored to the way in which earthquake engineering researchers prefer to organize their projects and conduct their work.

The Galaxy project [11] is a comprehensive web-based system that seeks to provide a cyberinfrastructure for the computational biology community. Galaxy provides a 'web-based platform for performing accessible, reproducible, and transparent genomic science' [11]. Similar to Galaxy, the NEES Project Warehouse and the NEEShub provide an integrated environment that simplifies the uploading, management, and reuse of earthquake engineering data, and the integration of these data with software tools within the NEEShub environment.

In comparing this related work with our work on the NEES cyberinfrastructure, there are several similarities. First, all of these projects provide facilitated access to high performance computing resources as a part of their cyberinfrastructure and use a web-based science gateway as the main portal to their cyberinfrastructure. Second, each of these projects focused on developing specific cyberinfrastructure solutions that are tailored to the science domain they seek to serve. Although all of these approaches use the same basic technologies (e.g. Linux and web servers), the specific solution is unique to the science domain.

The submit command available at *hubzero.org* [12] allows users to execute applications on remote resources. In comparison with batchsubmit, the submit command takes care of copying the data to the venue and retrieving the results back from the venue. The submit command also takes care of venue specific configurations automatically. The submit command is synchronous in nature compared to the asynchronous nature of batchsubmit. After submitting a job using submit command, the session should not be terminated or else the job would be lost. Currently, batchsubmit supports more HPC resources compared to the submit command.

2.2. Related work in civil engineering simulation software

Civil engineering research involves understanding the response of structures to forces over time. This inherent physical nature of the built environment has led to an emphasis on experiments conducted in structural laboratories based on building portions of structures, subjecting to them to artificial time-varying forces, and measuring structural response from data collected from sensors and photographs. These projects can generate thousands of data and analysis files for each experiment. This is demonstrated by the large number of projects (227), experiments (over 1500), and project files (1,711,424) in the NEES Project Warehouse, from which 4,777,252 files (as of October 2013) have been downloaded from the NEEShub over the past 12 months.

Building physical structures and testing them to failure is by nature time consuming and resource intensive. To supplement physical testing, computational simulation is another technique that allows civil engineering researchers to quickly explore a large space of options and conditions to help guide the design of physical simulations. In addition to pure computational simulation, another area of work—*hybrid simulation* and *real-time hybrid simulation*—involves the coupling of physical experiments with computational simulations through sensors and force actuators [13].

Computational simulation today in the civil engineering community exists in a continuum that ranges from desktop scale computing for simple simulations to complex simulations that require the use of high performance computing resources for large-scale structures and complex simulations that could not be completed in a reasonably short time on a single processor.

The nature of the physical and computational simulations developed by civil engineering researchers and practitioners varies. Researchers focus on the development and testing of new structural elements and novel building structures. Practitioners are interested in exploring the robustness and resilience of a building to be constructed for a range of potential natural hazards (i.e., earthquakes and hurricanes).

Much of the computational simulation is performed today on desktop systems using commercial (e.g., MATLAB and Abaqus) and open source (e.g., Octave and OpenSees) software. Other examples of commercial and open source simulation software such as Zeus-NL [14] and SAPWOOD [15], which

DOI: 10.1002/cpe

are available and can be run on the NEEShub (see http://nees.org/simulation for details). This approach can be used for many problems. However, for researchers seeking to scale up models in complexity of size, the closed source, expense, and license complications involved in using commercial simulation software on a high performance computing system has led to the popularity of open source simulation packages in the research community. The most popular open source simulation package (as measured by use on the NEEShub) is OpenSees [7,16, 17] developed by Frank McKenna at the University of California, Berkeley.

OpenSees is an open source software framework used by the earthquake engineering community to simulate the impact of earthquakes on structural systems. The OpenSees framework enables users to run their simulations in either serial or parallel modes, depending on the command and the solver in use. The serial operational mode is straightforward—the OpenSees interpreter reads input files and solves equations sequentially. This mode of operation is useful for application developers to test their scripts or run small test programs for verifying new concepts. However, applications run by the earthquake engineering community can be categorized into two major types of simulations—studying a small to medium model for different parametric values or studying a very large model. Depending on the type of problem in hand, a user can either use OpenSeesMP, which is tailored toward running parameter studies (for example, ground motion) or OpenSeesSP, which utilizes parallel processing to solve very large models (for example, soil models) [16].

OpenSees can be used to simulate the effects of earthquakes on structures modeled as *nodes* (representing connections between structural elements) and *elements* (representing columns and beams). Users can describe the structure and the masses of the elements in a *tcl* script as well as specify virtual sensors (*recorders*) on each node. The resulting *tcl* script is then input to OpenSees, which produces as output horizontal and vertical displacements collected from the recorders specified in the *tcl* input file.

Three versions of OpenSees are available: one serial version and two parallel versions: *OpenSeesMP* and *OpenSeesSP*. OpenSeesMP is designed to allow users to run parameter studies across a large number of processors with moderately sized models that can be analyzed in a limited time on one processor. OpenSeesSP, another parallel version, is specifically designed to support the analysis of very large models that cannot be tenably run on a single processor.

Capability runs can be defined as utilizing parallel processing to solve a large or complex model. In contrast to running smaller simulations with different parameter values, the success of a capability run heavily depends on the underlying solver used by the user. OpenSeesSP uses a single foreman node to process the *tcl* script provided by the user to decompose the structure into a system of equations and assign tasks to worker nodes. Users are given a choice of a parallel solver to be used: Mumps, Petsc, or SuperLU. OpenSeesSP is the easiest between the two versions of parallel OpenSees because it requires very little (if any) change in the input file.

In contrast to capability run, a *capacity run* can be defined as solving a large number of small to moderate sized problems (models) with a large number of parallel processes, where each process solves one or more such models in parallel. An example of this type of problem is a parameter study where each process may simulate the impact of earthquake on a 20-story building by varying the scale.

The benefit of the approach used by OpenSees is that a user need not become an expert in parallel computing to use the serial or parallel version—all versions can accept the same model described in the input *tcl* file. Details on the characteristics of the parallel versions of OpenSees are described by McKenna in [16]. In terms of programmability, OpenSeesMP requires a little more work than OpenSeesSP but provides much more flexibility in terms of the range of simulations and better scalability in general. While running with OpenSeesMP, each processor is running the same interpreter with the same main input script. The user has control at both the command line level and scripting level to specify the different work that each processor performs.

OpenSees is available on the NEEShub for download or can be run as a tool directly on the NEEShub without the need to install the software on a user's computer.

OpenSees Laboratory, shown in Figure 1, provides a convenient interface in the NEEShub to allow users to easily initiate OpenSees simulations [18]. One of the simulations immediately available to users is *SDOF Earthquake Response*. Another option allows users to submit to high performance computing platforms. Figure 1 shows an example of a job submission to the Carter supercomputer at Purdue.

DOI: 10.1002/cpe

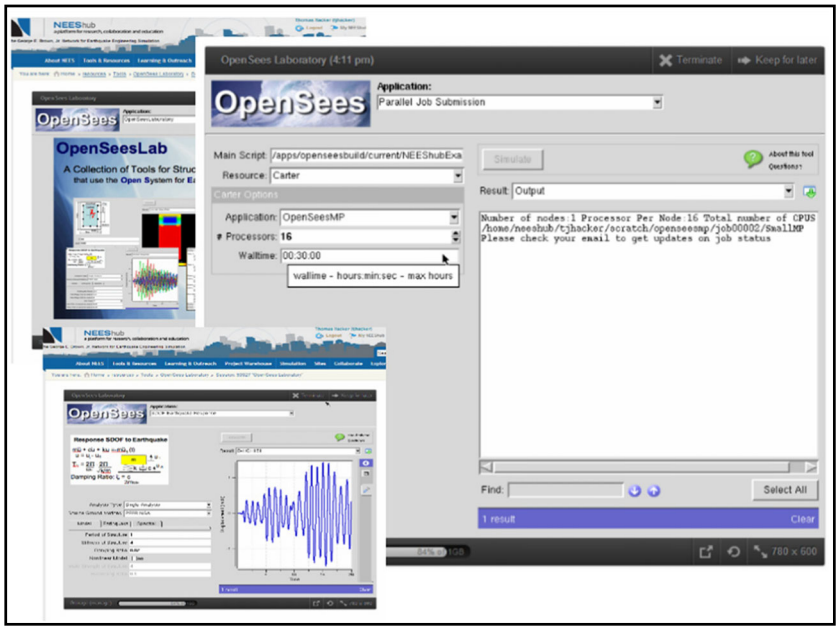

Figure 1. OpenSees laboratory running in the NEEShub.

The need for HPC resources in order to run OpenSees Lab as a tool in the NEEShub is one of the motivating factors for our work.

3. OUR APPROACH

3.1. Salient features

To simplify the process of job submission to HPC resources, we developed a command named *batchsubmit*, which serves as the client interface to the NEES high-volume batchsubmit system. The batchsubmit command takes different parameters for the batchsubmit system followed by the actual application command and application parameters if required. Parameters for the batchsubmit system all begin with two dashes. For example, the venue where job needs to run is--*venue*, the number of cores needed for parallel processing is--*ncpus*, and nonstandard specification of input files and directories. It processes the parameters, creates a scratch directory, copies and builds necessary files and then automatically submits the simulation job to the HPC venue. All parameters have a default value, which makes it easy for first time users to be comfortable with the system. For example, the command

% batchsubmit hostname

creates a scratch directory and job to run on the local venue to execute the simple command 'hostname', which is a Linux command to print the name of the host. A first time user will see the results of this job in the directory $HOME/scratch/hostname/job00001. All attributes of this naming convention are customizable. Furthermore, this does not use quota from the user's home directory because it is actually a link to a directory on a large storage server.

As a second example, one can see how easy it would be to do bulk synchronous parallel processing with the command

% batchsubmit –ncpus 4 env

This will run four copies of the *env* command in parallel. By looking at the standard output file found in the run directory, you could see different values for certain environment variables, which could be useful in scripting.

DOI: 10.1002/cpe

Batchsubmit has the ability to send input files, executables, and runtime libraries automatically to the HPC venue. A user can specify the path to the executable and path to the input file as parameters and command will automatically do the job of transferring the files to the remote venue. The results of the simulation are copied automatically from the HPC resources back to the user directory. An email will be sent to the user when the jobs start execution at the resource and when the job finishes execution.

A user can easily chose different venues for submitting their jobs by specifying the venue name like Kraken, Stampede, etc. as an input parameter. The user can also set the wall clock limit for a job. An advanced user can also select the specific queue to be used on the target resource.

To facilitate parallel processing, the batchsubmit command accepts the number of cores, number of processors per node, and number of nodes required as inputs. The user can specify all three of them or if not sure can specify one and others will be automatically calculated based on the default values corresponding to each venue.

Batchsubmit offers a feature to monitor job status at different venues. The command also offers the ability to cancel a job.

Users can access the batchsubmit command to submit and manage jobs through NEEShub web-based science gateway. Batchsubmit supports use through either a command line interface or a tool with a graphical user interface, such as OpenSees Laboratory [18].

3.2. *Batchsubmit and OpenSees*

Figure 2 explains how the initialization and the analysis phases occur in an OpenSeesSP simulation. The batchsubmit command to run an OpenSeesSP simulation on a parallel machine with Message Passing Interface library is as follows:

% mpirun –np N OpenSeesSP main.tcl

where *N* is the number of cores the user wants to use for parallel processing. However, this command assumes that the user has access to a parallel machine with at least *N* cores available with the Message Passing Interface library installed on it, which may not necessarily be true for all users. In this regard, NEEShub provides a great interface through the *batchsubmit* command for running these parallel simulations. The *batchsubmit* command for running an OpenSeesSP simulation is

'batchsubmit –venue available_venue –appdir dir_with_OpenSeesSP_executable –ncpus N OpenSeesSP full_path_to_main_input_script'

It can be observed from the figure that OpenSeesSP will not scale well because creating all these elements and nodes in the address space of a single process will limit the size of the problem that can be analyzed using this interpreter.

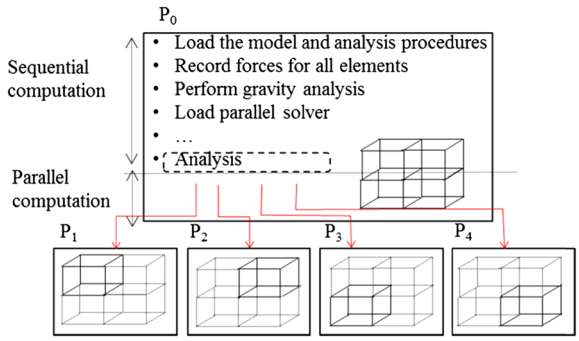

Figure 2. This figure illustrates how workload in an OpenSeesSP simulation is distributed. Process P_0 performs the sequential task of model formation and task distribution, whereas the rest of the processes perform parallel processing of chunks of the input model. The scalability of this method suffers as problem size is limited by the amount of memory available on P_0 [13].

DOI: 10.1002/cpe

4. ARCHITECTURE

The batchsubmit command creates a venue specific submission script file incorporating the specified input parameters. For PBS venues like the Hansen and Carter systems at Purdue, batchsubmit creates a submission script file using PBS syntax. For the Open Science Grid, batchsubmit creates a submission script file using Condor syntax. The job submission script file contains information about the input files, executable version and its dependencies, number of cores, execution time limit (walltime), and other miscellaneous configuration information.

The job submission script is submitted as a batch job at the venue using the venue specific submission command (eg: *qsub, condor_submit*). The job execution system at the venue reads the job submission script file and allocates the number of cores as specified in the file. The input files are copied to each core and executed using the specified version of executable. After executing the input file, the job submission script will invoke scripts to copy results back to the batchsubmit server.

The architecture of the batchsubmit command is as shown in Figure 3. The batchsubmit architecture has four main parts: (1) the workspace; (2) the batchsubmit client; (3) the batchsubmit server; and (4) the remote HPC resource.

The batchsubmit command can be accessed by the user through the High Volume Job submission Tool available in the NEEShub. The tool provides users with a *workspace*, which is a Linux based command line interface. The batchsubmit command is already installed in the workspace. Users can execute jobs by typing the batchsubmit command and specifying the required input parameters.

There are three companion commands to batchsubmit. They are *batchstatus*, *batchcancel*, and *batchsave*. The batchstatus command shows the status of all jobs belonging to the user or all jobs specified by the arguments to the batchstatus command. The batchcancel command will send a request to the batch server to cancel a job. The batchsave command will move the job directory from the scratch data server to the local directory and remove the symbolic link. This causes the data to be backed up not only by NEEShub servers but also requires sufficient quota.

The result of the submitted jobs can be found in the $HOME/scratch directory in the users workspace. The results will be organized according to the type of the job (e.g., OpenSees and OpenSeesMP [7]) and the job number.

Once the user executes *batchsubmit*, the batchsubmit client verifies the job. The batchsubmit client verifies that the venue is available, the executable path provided is valid, the parameters specified are valid, and that the path to input files are correct. If there is an invalid case, the job will terminate displaying the proper error message. If the job is a valid one, then the client uses the specified input parameters along with default parameters and generates a job submission script file for the specified HPC venue. A new job directory is created on a scratch data server for the job. This is called the *scratch directory*. A symbolic link to this directory is made under the users $HOME/scratch directory. All the input files and executables are copied to the scratch directory. The venue specific wrapper script is generated in the scratch directory. This directory contains all the components necessary to encapsulate the execution of the job for remote execution. This requires that the application or application wrapper uses the 'current directory' paradigm of referencing input and output files. That is, all input, output, and control files are referenced relative to the current directory

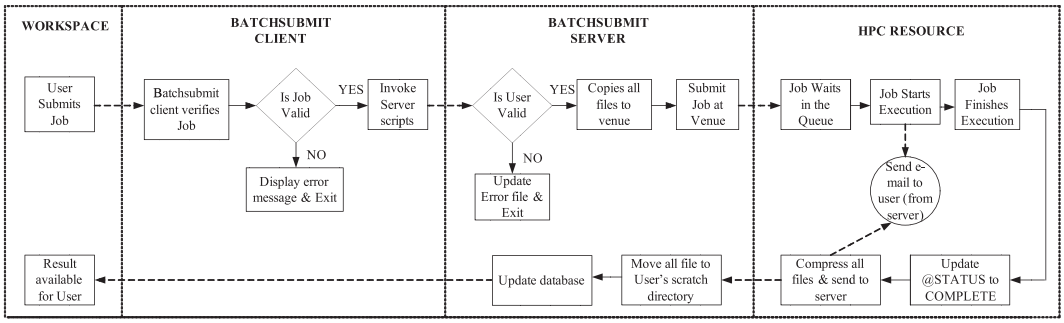

Figure 3. Architecture of the NEES batchsubmit system.

DOI: 10.1002/cpe

without absolute path names. This does not preclude the use of subdirectories within the current directory; it just requires that these subdirectories be referenced relative to the current directory.

After the creation and preparation of the scratch directory, the batchsubmit client would then signal server scripts to execute the actual submission process if the execution venue was not local.

The batchsubmit server, on receiving the request from the client, validates the user credentials. To submit jobs using batchsubmit the user has to be a part of the HPC access group in the NEEShub. The server ensures that the submitting user has access to the group prior to submitting the job to the HPC venue. If the user is not authorized, the job is not submitted and an error file is created in the job directory ($HOME/scratch) of the user. If the user is valid then the server copies the contents of the job directory to the remote HPC resource. Once the server successfully copies the file it would then go ahead and submit the job at the HPC resource. All activity on the batchsubmit server is asynchronous to the execution of the batchsubmit command (the client). So upon termination of the batchsubmit command the status of the job is 'PRESUBMIT'. When the server finishes submission to the HPC resource, it changes the job status to 'SUBMITTED'.

Once the job is submitted, the job will be placed in the queue waiting for its turn to execute on the HPC resource. When the turn arrives the job will start executing the venue specific script file. Before and after executing the user job, the venue specific script file communicates with the batchsubmit server and the server sends email to the user indicating the start and completion of the job. Once the execution is complete, the job epilog of the venue specific script file compresses and copies all the results back to the server.

When the batchsubmit server receives the results back from the HPC resource, the server uncompresses the results and places them in scratch directory for the user. The server also adds the job log information to the database to maintain information about batchsubmit job statistics.

At the completion of this process, the user has all of the results available in their job directory. Currently, if the user wants to save the simulation results in NEES Project Warehouse, they need to copy the data using the Project Explorer for NEES (PEN) tool running locally on the NEEShub that can directly access the scratch space and the Project Warehouse. The work to move the results to Project Warehouse automatically is still under progress.

The batchsubmit command has a help file where user can find all necessary information about the command syntax and input parameters. The help file also contains some examples on submitting different kinds of jobs. More information about batchsubmit can be found at the batchsubmit 'about page' as well as batchsubmit wiki page in the NEEShub. There is also a detailed document available at these locations about how to choose a suitable venue for a job.

5. OPENSEES WORKFLOWS THROUGH PEGASUS

Scientific workflows are becoming the new way of designing complex and multistage scientific computations. Workflows allow researchers to execute a series of computational simulations or tasks that perform data management in scientific applications. In a scientific workflow, the researchers would define a directed acyclic graphs of simulation jobs, where each vertex corresponds to the simulation jobs and edges corresponds to the dependencies between the jobs. The workflows also let researchers run pre-processing and post-processing scripts before or after the simulation jobs, mainly to collect required data, format the output data, configure the input parameters for jobs, etc. Hence, scientific workflows allow researchers to execute a large sequence of complex or simple jobs in a specified manner at one shot, which results in huge time savings and high research productivity.

Pegasus [19, 20] is a configurable system, which allows researchers to transform their complex computational tasks into workflows. Researchers can create workflows using Pegasus without bothering about the specific requirements of the execution hardware or the particular specifications of the underlying software interface. Pegasus supports the execution of one or a collection of simulation jobs over a wide variety of heterogeneous resources. Pegasus chains together dependent tasks and executes them in sequential manner and executes the independent tasks in parallel. Pegasus automatically collects the required input data and allocates the computational resources sufficient to execute the workflow.

DOI: 10.1002/cpe

Pegasus can be used on the NEEShub to manage the execution of a workflow of jobs consisting of OpenSees simulations [21]. One of the main applications of using Pegasus with OpenSees is to run simulations focused on parameter sweeps. In parameter sweep jobs, a series of independent simulations are executed in parallel in which the input parameters for each simulation vary over a range of values. The process to submit an OpenSees Pegasus Workflow on NEEShub is as follows.

- Develop and gather all the input simulation files required for the workflow (eg. tcl files).
- Plan the workflow that needs to be implemented.
- Generate the DAX (directed acyclic graphs xml) file that describes the workflow. It is recommended to use a script to generate the DAX file automatically. In the tutorial, a sample python code is provided for which a user can modify to construct the required workflows.
- Submit the DAX file.

Once the DAX file is submitted, Pegasus will automatically submit the individual jobs according to their dependencies. The result of the jobs is made available to the user automatically. A tutorial on running OpenSees through Pegasus is on the NEEShub [22, 23].

6. EXPERIENCES

NEEShub has enabled a number of very important civil engineering studies over the years. One example is the work of Francisco Flores Solano, a graduate student at Virginia Technological University [24]. Solano's research is focused on understanding the influence of gravity on the collapse of *Special Moment Steel Frames* (*SMF*) and floor accelerations in SMFs. He is analyzing nonlinear models subjected to *Incremental Dynamic Analysis* using OpenSees driven by a set of 44 different ground motions using nonlinear dynamic time history analysis. Solano can run 44 simulations in parallel using XSEDE systems and the Carter system at Purdue. Each individual simulation requires from 5 to 15 hours. Consequently, running 44 simulations sequentially would take up to 27.5 days. The use of OpenSees through the NEEShub is helping Solano perform more simulations in less time, which helps make a positive impact on his research.

Henry Sweat at the University of California, Berkeley is working to understand the response of reinforced concrete buildings to ground motions when brittle unreinforced masonry walls are added to the structure [25]. His model studies the addition of unreinforced masonry walls to a reinforced concrete building. His work is based on a nonlinear model running in OpenSees that is used to explore the coupled in-plane and out-of-plane motion of the structure leading to collapse. In the past, because of the computationally intensive nature of the models, this area was not thoroughly explored. Sweat estimates that a single processor running OpenSees for his simulation can take up to 500 h of computation for the large number of ground motions being analyzed. Using XSEDE resources through the NEEShub, Sweat estimates that his simulation can be completed in approximately one day—a speedup of approximately 20×. The use of XSEDE through the NEEShub allows him to run more ground motion simulations with more realistic models in much less time.

6.1. Statistics

The usage statistics for batchsubmit is shown in Table I. The table shows the number of users using batchsubmit and the number of jobs being submitted using batchsubmit per month for the last year. In terms of the performance gain experienced by users, the normal process users would employ for submitting jobs is to *ssh* to a head node or job submission host, create a job submission script for the scheduling system, and manually submit a job request. In contrast to this cumbersome manual process, batchsubmit provides easy access to multiple HPC venues through a simplified Web application that can create and submit job submission scripts on behalf of the users. This allows researchers to reap the full benefits of these systems without going through the time-consuming protracted manual process. Researchers can now run their simulations much faster and easier compared to running simulations manually at any of the HPC venues or running simulations sequentially. Hence, batchsubmit results in huge time savings for the researchers which can be

DOI: 10.1002/cpe

Table I. Batchsubmit usage statistics.

Year	Month	Submitted number of jobs
2012	September	3341
	October	6360
	November	2753
	December	3485
2013	January	5301
	February	3002
	March	48,163
	April	4017
	May	2125
	June	1661
	July	2893
	August	3121
	September	3507
	October	3705

utilized to run more simulations as well as in studying and understanding the given research problem in great detail.

6.2. *Lessons learned*

One important lesson learned during the development of batchsubmit was the need to use a community gateway account for each venue instead of individual accounts for each user on multiple venues. This greatly reduces the complexity of managing accounts and keeping track of allocations on multiple venues.

The selection of the correct venue is critical to ensure efficient use of our XSEDE allocation. Large simulations should be run at venues like Kraken and Stampede, the medium level jobs can be run at other venues like Hansen, Carter, and OSG. The NEES machines can be used for smaller jobs. We have added documentation on the batchsubmit wiki in the NEEShub on how to select a venue.

Simulation environments can generate massive amounts of data that are difficult for IT organizations to backup. It is certainly less critical to backup simulation data than manually created data such as model specifications. By using a scratch data server for simulation input and output, we not only modularize the job for remote execution, we also avoid the use of local user home directories or other critically managed data space thus avoiding a big problem for backup procedures.

The most common mistake that users make is specifying an insufficient *walltime limit*. Walltime is the time required to complete a computational task as measured by the 'clock on the wall'—hence, the term *walltime*. Users tend to specify low walltime limits that can prematurely preempt the job prior to completion. It is important that the users calculate the expected walltime before specifying it or else leave it to the default value.

One other lesson we learned was when using HPC resources such as XSEDE or the OSG, a comprehensive approach in needed to transfer the files to the HPC resource and to collect the output files and error files from the computation. Along these lines, we are working with Andre Barbosa, a researcher working at Oregon State University, to create a comprehensive facility to allow researchers running OpenSees [7] jobs on OSG (and soon XSEDE), to collect simulation output, and to transfer the most meaningful simulation results to the NEES Project Warehouse as a new type of NEES project.

Some of the HPC resources have long queue wait times, which means the job has to wait for its turn to execute. In that case, a user would like to know the status of their job. Hence, the addition of the feature to check the status of the job really helped users to monitor jobs status. Another useful feature added based on a user request was to the ability to cancel a job. This allowed users to terminate a job, which is either running or waiting at the remote venue.

The HPC resources undergo maintenance at times during which they are inaccessible. Hence, the users should be made aware of this fact or else they will end up submitting jobs that will never run.

DOI: 10.1002/cpe

Hence, a new option was added to the batchsubmit command through which the users can view which all venues are available for job submission. The batchsubmit command was also modified to automatically prevent the submission of a job to an unavailable venue.

When using batchsubmit, the user will obtain an error message if there are any errors in the input file or if the job is lost. To increase the reliability of the batchsubmit system, we added *verification points* at each stage of the batchsubmit system. Verification points are check points to ensure that each stage of batchsubmit executed correctly. At each verification point, a check is made to ensure that the previous operation executed successfully by reading the status using the *$?* variable in Linux. If an error is present at any stage of batchsubmit, an email will be sent to the NEEScomm IT staff, and the issue is fixed, and job is resubmitted without any involvement from the user. The verification points ensure that the job successfully passed each stage, and if errors occur, the user will be notified of the job submission failure and the need to resubmit the job. This prevents a scenario where the user never obtains an update on a submitted job when it was not submitted successfully. Because batchsubmit follows an asynchronous job submission method, the user does not know the time at which the job completes execution. Hence, the need for email notifications of job status for users was unavoidable.

Even though batchsubmit is an easy to use command line interface, we learned from the users that they prefer a GUI. Hence, we are working toward creating an exclusive GUI for batchsubmit incorporating all its features. The new GUI will have additional features like automatic selection of venue, predicting the queue wait time for a job, extended support for parameter sweep jobs and workflows and more.

7. CONCLUSIONS

We have successfully developed a high-volume batch submission system *batchsubmit* that provides the researchers of the NEES community with a simplified and consistent command interface to submit simulations jobs at various HPC resources at XSEDE, OSG and Purdue University. We believe that the batchsubmit system described in this paper could be easily extended for use by other science and engineering domains.

ACKNOWLEDGMENTS

The work described in the paper is supported by NSF grant CMMI-0927178 (NEES) and XSEDE allocation BCS110002. We would also like to thank Francisco Solano and Henry Sweat who are active users of high performance computing resources through the NEEShub. This work was performed under the auspices of the US Department of Energy by Lawrence Livermore National Laboratory under Contract DE-AC52-07NA27344 IM #LLNL-JRNL-645386.

REFERENCES

1. Eigenmann R, Hacker T, Rathje E. Nees cyberinfrastructure: a foundation for innovative research and education. *Proceedings of the 9th US/10th Canadian Conference on Earthquake Engineering*. July 25-29, 2010. Toronto, Ontario, Canada.
2. Hacker TJ, Eigenmann R, Bagchi S, Irfanoglu A, Pujol S, Catlin A, Rathje E. The neeshub cyberinfrastructure for earthquake engineering. *Computing in Science & Engineering* 2011; **13.4**:67–78 July-Aug. 2011 doi: 10.1109/MCSE.2011.70.
3. Hacker T, Eigenmann R, and Rathje E. (2013). Advancing Earthquake Engineering Research through Cyber-infrastructure. *J. Struct. Eng.* 139, SPECIAL ISSUE: NEES 1: Advances in Earthquake Engineering, 1099–1111.
4. NEEShub usage statistics. http://nees.org/usage.
5. Fiorini P. NEES-led data integration project aimed at exposing U.S. quake data to researchers in Europe, elsewhere globally. January 15, 2014. Purdue News Service. Available from http://www.purdue.edu/newsroom/releases/2014/Q1/nees-led-data-integration-project-aimed-at-exposing-u.s.-quake-data-to-researchers-in-europe,-elsewhere-globally.html
6. Mohan A, Hacker T, Rodgers GP. Developing a high-volume batch submission system for earthquake engineering. In *Proceedings of the Conference on Extreme Science and Engineering Discovery Environment: Gateway to Discovery* (p. 51). ACM. 2013.
7. OpenSees, OpenSeesMP http://opensees.berkeley.edu/
8. MATLAB http://www.mathworks.com/products/matlab/
9. OCTAVE http://www.gnu.org/software/octave/

DOI: 10.1002/cpe

10. Goff SA, Vaughn M, McKay S, Lyons E, Stapleton AE, Gessler D, Matasci N *et al*. The iPlant collaborative: cyberinfrastructure for plant biology, Frontiers in plant science **2**, 2011.
11. Goecks J *et al*. Galaxy: a comprehensive approach for supporting accessible, reproducible, and transparent computational research in the life sciences, Genome Biology **11.8**, 2010; R86.
12. McLennan M, Kennell R. HUBzero: a platform for dissemination and collaboration in computational science and engineering. *Computing in Science & Engineering* 2010; **12**(2):48–53.
13. Castaneda N, Gao X, Dyke S. A computational tool for real-time hybrid simulation of seismically-excited steel frame structures. *Journal of Computing in Civil Engineering* 2013; 10.1061/(ASCE)CP.1943-5487.0000341
14. Elnashai AS, Papanikolaou V, Lee D-H. *"ZEUS-NL A System for Inelastic Analysis of Structures", MAE CD Release 04-01, Mid-America Earthquake Center*. University of Illinois at Urbana-Champaign, 2004.
15. Gavin HP. 2010, "Frame3DD," http://nees.org/resources/frame3dd.
16. McKenna F, Fenves GL. Using the OpenSees interpreter on parallel computers. NEESit report no. TN-2007-16. 2008.
17. Mazzoni S, McKenna F, Scott MH, Fenves GL. OpenSees command language manual. *Pacific Earthquake Engineering Research (PEER) Center* 2005.
18. McKenna F, Christopher M, Pedro A, Joseph AH. OpenSees Laboratory. 2013. Available from https://nees.org/resources/openseeslab
19. http://nees.org/groups/pegasus
20. http://pegasus.isi.edu/
21. http://nees.org/groups/pegasus/wiki/GETTINGSTARTEDWITHPEGASUS
22. http://nees.org/groups/pegasus/wiki/MainPage
23. http://pegasus.isi.edu/wms/docs/latest/tutorial.php
24. Solano F. Personal communication, October 22, 2013.
25. Sweat H. Personal communication, October 22, 2013.

DOI: 10.1002/cpe