PARA-INTERFACE:

A Novel Interface for Para-AEH Software Suite

by

Michael Clay Ginn

A thesis submitted to the faculty of The University of Mississippi in partial fulfillment of the requirements of the Sally McDonnell Barksdale Honors College.

Oxford

May 2015

Approved by

Advisor: Professor Byunghyun Jang

Reader: Professor Carl Jensen

Reader: Professor Philip Rhodes

ACKNOWLEDGEMENTS

Foremost I would like to acknowledgement my gratitude to my advisor Dr. Byunghyun Jang. Dr. Jang has supported through my tenure at the University of Mississippi from when I first took a class with him. From there he first introduced me to his research group the Heterogeneous Systems Research Lab. Dr. Jang and the research group have been very supportive of me and provided me an opportunity to participate in research even as an undergraduate student.

Besides Dr. Jang, the Computer Science Department as a whole has been very supportive, faculty and staff alike. My time with the department has been a great experience. The department has supported me through my academic experience and has been a great resource in assisting me throughout the completion of this thesis.

ABSTRACT

Para-AEH is a software suite designed to the Asymptotic Expansion Homogenization method. This method is commonly used to model structures by using differential equations to represent larger structure from a smaller structure input. In order to facilitate the University of Mississippi's Department of Mechanical Engineering's research purposes the software needed to be installed. A novel interface was designed to remove the overhead of interacting with the software on a command line based level. The software allows the management of users that can access the software and jobs that package several commands together for asynchronous execution without manual intervention.

Table of Contents

LIST OF TABLES

LIST OF SYMBOLS AND ABBREVIATIONS

AEH	Asymptotic Expansion Homogenization
API	Application Programming Interface
DOM	Document Object Model
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
MCSR	Mississippi Center for Supercomputing Research
MPI	Message Passing Interface
TLS	Transport Layer Security
URL	Uniform Resource Locator
VTK	Visualization Toolkit
XML	Extended Markup Language

LIST OF FIGURES

Introduction

Asymptotic Expansion Homogenization is a commonly used method to study material microstructures in the field of mechanical engineering. The method involves applying differential equations and modeling microstructures of a larger macrostructure to gain a better understanding of the material properties involved. AEH is a powerful in the field of materials research so researchers can discover the strain and stress properties of materials through a mathematical analysis rather than destructive forms of physical analysis. This can be quite useful in many different applications such as making sure a material can handle certain types of stresses or strains before the material reaches a point of failure. This type of analysis can be crucial for situations where knowing the strength of materials is important for construction, defense, aerospace, or even manufacturing.

The University of Mississippi's Department of Mechanical Engineering conducts research following this method and received a software suite called Para-AEH. This software suite implements tools necessary to perform academic research using AEH but leverages capabilities of MPI to conduct analysis and modeling in parallel. The software suite is distributed directly as source code meaning the suite must be compiled and linked with the proper dependencies.

The Department of Mechanical Engineering requested assistance in working with the software suite, primary concern being installing the software suite to be used for research immediately. Project specifications later were added in order to expand the usefulness and accessibility to the software backend to improve the ease and quality of research. Modifications and additions to the software suite will aid long-term research goals.

Para-AEH was designed to be run on the Linux platform by executing a series of commands on a shell. In order to facilitate research, it was decided to create a graphical based interface to assist researchers creating commands to be executed in logical unit of a job. Instead of having to write out a series of shell commands with several command line arguments each, a user can simply fill out a form and have a program dynamically create the commands and issue them to the backend shell. An interface such as this allows users to quickly execute jobs without wasting time properly formatting shell commands with the correct flags. Job submission also allows asynchronous execution so as soon as it has been submitted a user can begin developing the next job or work with previous results.

Theory

Asymptotic Expansion Homogenization is a technique in computational mechanics that allows a user to model a macroscale heterogeneous material by modeling its periodic microstructure. By modeling the spatial distribution of the particles or features within the microstructure and solving homogenization equations researchers can represent the materials properties of the macrostructure as a whole. Two scales are introduced in the beginning of analysis, a macro scale and a micro scale. Equation manipulations begin in the micro scale. Nodal displacements are expanded asymptotically and then substituted to create a microstructure equation. This equation can then be volume averaged to become an equation on the macro scale. This methodology, with more manipulations, creates a coupling between the micro and macro variables. By assuming that micro displacements have a linear relationship with the macro strain the variables can be decoupled. Eventually the manipulated equations can be transformed into a finite element form. Finite elements are the basis for solving for the localized strains and stresses [1]. AEH approaches significantly decreases the costs in solving equations to model composite material structures. The homogenization step decreases the computational size as compared to other methods [2].

Para-AEH

The Para-AEH software suite was developed to provide a framework for studying the materials properties of heterogeneous materials. The suite implements the AEH briefly touched upon before. This software was designed to leverage several popular mathematical based software frameworks to parallelize some of the computations to increase speed and efficiency. The software allows a researcher to efficiently discover properties such as local strains and stresses and potentially discover potential stresses that could weaken or break a material [1]. Para-AEH was developed out of a need for a modernized update to previous software applications. Previously similar AEH applications were designed in FORTRAN which needed an overhaul so it could be distributed to other researchers in the field. C++ was designated to be the language that it would be designed in because C++ allows the implementation of the object oriented programming model. There are also many popular software libraries that can be leveraged such as the Trilinos or Boost frameworks.

Para-AEH consists of two primary tools with 8 additional tools. The primary tools deal with the primary theory of AEH. The first tool **mkHomogPropsAndCorrGrad** outputs the homogenized properties and the value representing a corrector gradient that can be used to calculate localized strains and stresses in a structure. After creating these outputs a user can then actually calculate these localized strains and stresses by using the **getMicroStr** tool. The rest of the tools available in the software suite create the data to input into the primary tools. The software suite is designed to be leveraged in a very hand on approach to get to the end result of finding the homogenized properties to discover the localized stresses and strains of whatever structure being worked with. In regards to reaching the end goal, there is a logical sequence of commands that a researcher must follow. The order for a typical research job and descriptions of each stage can be found in in Table 1.

Table 1. Order of Para-AEH Commands

Command(s)	Description
1. **mkRegularBrickMesh**	Creates a regular mesh of finite element bricks in the first stage.
2. **mesh2scotchGraph** **dgpart** **scotchMap2Metis**	Partitions the newly created mesh in partitions to be distributed via a MPI job.
3. **findPerImgs3D**	Identifies which nodes are periodic images of each other.
4. **findElemCentroids**	Assigns material names to every element in a mesh.
5. **nativSimp2vtk.sh**	Optional stage that creates a VTK based file to allow visualization.
6. **mkHomogPropsAndCorrGrad**	Determines homogenized elastic properties of the mesh
7. **getMicroStr**	Finds the localized strains of the mesh once the homogenized properties have been discovered.

Development Platform

In order to facilitate rapid development a different platform was chosen to develop on rather than developing on the target machine. The target machine had limited access so development took place on another computer. As with the target machine a Linux platform was designated to be the operating system to be developed on. Ubuntu 14.04 was the version of Linux chosen to be the primary system. The specifications of the system can be found in Table 2. As noted this machine is fairly standard for a basic development environment for the Para-AEH software suite as well as the Para-Interface software developed to interact with Para-AEH. The development machine had root access and was a standard desktop environment as compared to the server environment of the target platform. This allowed the use of the popular IDE, Eclipse. Eclipse is a powerful tool for software development and has tools to debug, profile, and format code.

Despite being a different version of Linux than the target platforms the variations themselves were not enough to warrant concern. The only main difference was the package management systems provided by each platform. Ubuntu is a Debian based system and uses **apt-get** as its package manager whereas the target machine was a different version which uses **yum** as its package manager. The packages managed by both systems respectively have widely used libraries therefore they remain consistent among different Linux platforms.

Table 2. Development Platform Specifications

System Component	Component Specifications
Operating System	Ubuntu 14.04
Processor	Intel Core i7 2.0 Ghz
Cores	8
RAM	4.0 GB

Target Platform

In the beginning of the project there was originally the target platform was one of the supercomputers hosted by the Mississippi Center of Supercomputing Research. The motivation behind leveraging this platform was the high performance of the computing resources available. As Para-AEH is an MPI based program this would have suited perfectly for large modeling projects that the Department of Mechanical Engineering may have developed. MCSR is open for researchers to use freely with a system of queueing jobs to allow a system of sharing resources. As noted in Table 3, the number of CPUs potentially as well as RAM would have greatly assisted in computationally expensive research for advanced projects the department might participate in with Para-AEH. Problems however with the current status of the target platform prevented the installation of Para-AEH. Several libraries on the cluster are outdated and cannot be updated as is. As Para-AEH leverages newer libraries and frameworks it would be very difficult to bootstrap a new system within the current constraints. Potentially upgrading the system as a whole would require administrative access and the potential of interrupting and damaging other important research projects took the secondary option off of the list of potential fixes. Bootstrapping a system locally could be a possibility but proved to be difficult.

Table 3. Supercomputer Target Platform

System Component	Component Specifications
Operating System	SUSE Enterprise
CPUs	(variable specifications)
Cores	1304
RAM	3.44 TB

Another platform arose as a potential opportunity to host the software as well as the interface when the department offered another system owned by a researcher. This machine is significantly smaller in terms of resources as compared to the MCSR computers but for the intents and purposes of the project it was deemed sufficient. A plus of using this machine is the fact it is managed by Engineering’s IT Department greater access and control could be allowed in setting up the software and submitting research jobs. With the supercomputer one would have to use the queuing system provided to run jobs in order to promote fair usage of the resources. With greater control of the resources available it would be easier for the research staff to work with jobs and faster turnaround of job submission.

Table 4. Engineering Dept. Computer Specification

System Component	Component Specifications
Operating System	Red Hat Enterprise 6.6
CPU	Intel Xeon
Cores	24
RAM	128 GB

Project

This project was designated to consist of two primary stages with additional sub-stages within the primary ones. First and foremost it was important to compile and install the software. This was necessary to validate the provided code base was capable of being compiled and run. It was important to gain an understanding regarding the dependencies Para-AEH relied upon because without these compilation would be impossible. During the compilation and installation stage there were four sub-stages that each correlated with a specific dependency or the final software product. These included: Linear Algebra libraries, Boost Framework, Trilinos Framework, and Para-AEH.

Once the primary stage of compilation and installation was completed the primary interface software could be developed. The idea behind the interface was to allow a graphical based approach to allow researchers to craft research jobs without a great understanding of the Linux platform. Developing the interface was divided into three primary stages. The stages were as follows: Java Backend, Java Servlet Middle End, and the Foundation Web Frontend. These correlate with specific aspects of the interface that when all in unison with each other effectively interact with the software suite to dynamically create research jobs.

Compilation

Para-AEH is a computational expensive software suite but powerful for AEH research. The software suite leverages several shared Linux libraries that are often leveraged for mathematically intensive applications. All of these dependencies must be resolved in a specific order to properly install on the system that a user wishes to install the suite on. The three main dependencies include BLAS & LAPACK, Boost Framework, and Trilinos Framework. These dependencies must be compiled and installed in the order described. The system itself must also be MPI capable before approaching these dependencies [1].

Basic Linear Algebra Subprograms or commonly known as BLAS is a package of 38 subprograms designed for linear algebra manipulations. This package dates back to 1979 and was developed in FORTRAN [3]. BLAS is still managed in FORTRAN to this day because it is considered efficient and portable. It is commonly used in development for higher level linear algebra software [4]. LAPACK is another linear algebra package built on top of BLAS. This package expands upon the capabilities of BLAS and targeted for solving systems of linear equations [5]. BLAS and LAPACK both should be installed first with the former being the first of the two. Since these libraries are widely used they are often found in most Linux package manager repositories so should be installed through this method if possible.

Boost is a collection of C++ libraries that serves as a framework that complements the standard C++ library. The goal of the Boost Project is to allow developers easily implement functionality that goes beyond the capabilities of the C++ standard library. The popularity of Boost had even influenced the latest iteration of the

C++ standard library released in 2011 [6]. Boost 1.52.0 or later should be installed to use with Para-AEH. Boost can be compiled from source or may be found in a package management repository. CentOS versions of Linux may have Boost available as a package but the version available is not sufficient. If compiled from source there are two scripts to assist in setting up the installation environment: *bootstrap.sh* and *b2*. *Bootstrap.sh* is the primary script to setup the environment. The modules that are necessary for installation are: **filesystem**, **mpi**, **program_options**, **regex**, **serialization**, and **system**. Care should be taken to verify in the output of the script to ensure that all modules are verified in the output of the first script. The command *./b2 install* should be then issued to install the boost framework to the system [1].

The Trilinos Framework is a collaborative project designed to supplement developers with robust algorithms and libraries designed for maximum flexibility. The Trilinos Project is the primary developer of the framework with efforts from Sandia. Trilinos leverages the Boost framework and comprises of various modules targeting large-scale scientific and engineering work. The project aims to decrease the work of implementing new algorithms and applications by standardizing the existing Trilinos APIs. Trilinos also allows the focus on the development of robust parallel implementations of scientific and engineering algorithms [7]. Trilinos requires the most care to compile and install on a system. The framework source leverages cmake to build and setup the environment. Para-AEH's manual gives a sample shell script to be modified and invoked to properly setup the build environment for Trilinos. Several variables must be modified and resolved. These variables refer to previous dependencies and the locations of their installation for reference. The build directory should be a directory different from the source directory to ensure cmake and the build process does not interfere with the source. If no errors occur during the issuing of *cmake*, *make*, and *make install,* then the framework should have been successfully installed [1].

Para-AEH can be successfully installed once the primary dependencies described earlier are resolved. The software suite also leverages the cmake system to create a build environment. Cmake should be given arguments to point to the locations of Boost and Trilinos's installation. A variable can also be set to force Para-AEH to leverage 64-bit indices if desired. The command should also be given in a separate directory because the software cannot be built in the same directory as the source. Once cmake has been invoked the commands *make* and *make install* can then be issued. Mpmetis and dgpart from Metis and PT-Scotch respectively should later be installed but are not necessary to compile the software. If compilation and installation is successful Para-AEH's tools are readily available through the shell [1].

Interface

The software suite exists as a collection of command line based tools. Knowledge of the Linux platform is necessary to run the software which many people that may use the software won’t necessarily have the experience. The goal for designing an interface is to allow researchers interact with the suite in an intuitive manner so as to remove the overhead of using the software via the command line. Several tools in the suite reuse arguments or require commands to be issued in a certain order. By creating a higher level interface to act as an intermediary can help overcome some of these issues and maximize research efforts.

The language that was chosen for this interface was Java. Java is a popular language due to its design. The design goals of Java fell in line with the requirements for the interface design so it was chosen. First and foremost the language is fairly simple and high level when compared to others. The interface software needed to be simple enough that a non-experienced programmer could understand and modify as the department’s research goals grow and change for Para-AEH. The object oriented design allowed the software to be designed in stages and modules with different capabilities for each object designed. Several different types of software development methodologies such as singletons or interfaces could easily be implemented due to the object-oriented nature. Java is also an architecture independent piece of software. Java code is compiled into bytecode which is run on the Java Virtual Machine. This allows the software to be easily ported from machine to machine without the necessity of recompiling. This prevents the hassle of worry about architecture dependence. Java performs quite well and provides high level sophisticated implementations of multithreading. Java threads will allow for the asynchronous execution of commands while a user may began to analyze previous data or create a new job to be executed [8].

Java Backend

The interface has a basic structure as noted by Figure 1 below.

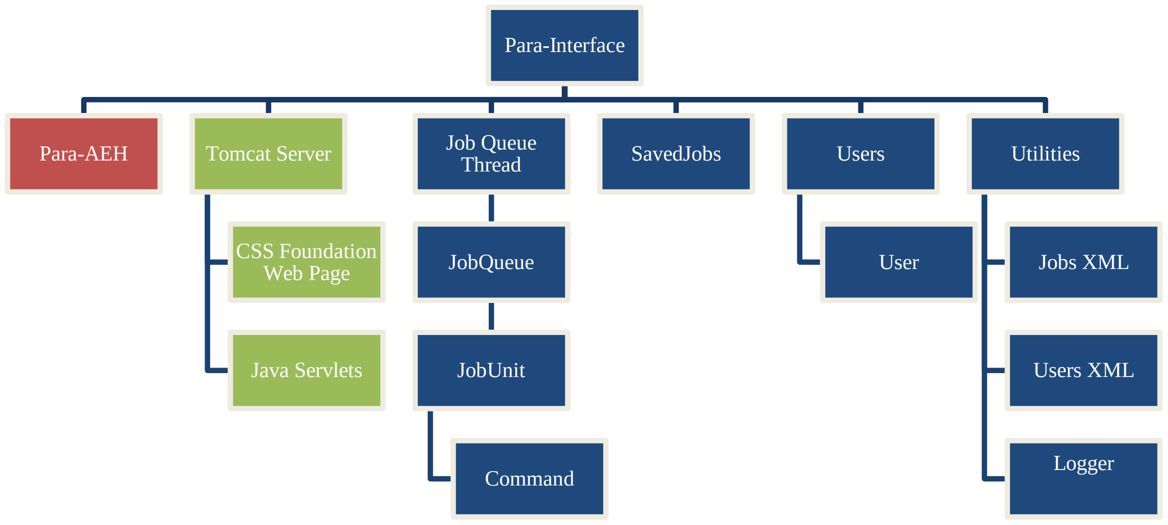
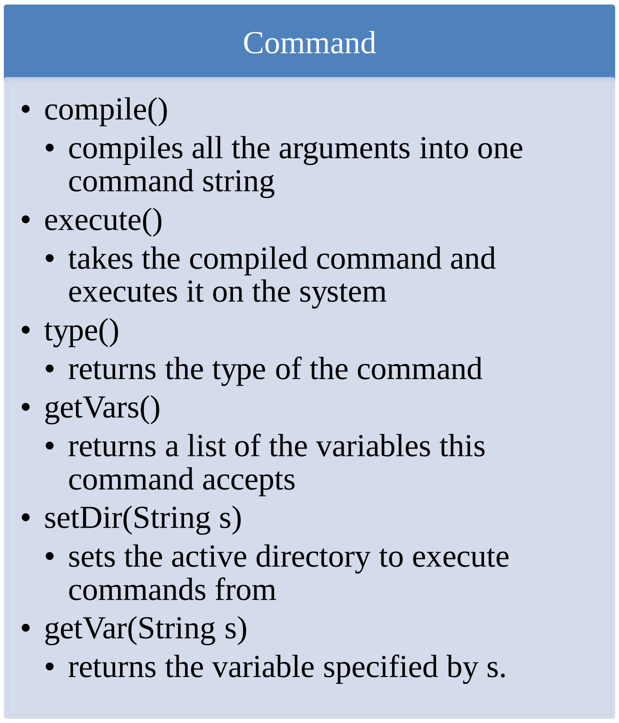


Figure 1. Para-Interface Basic Structure

Para-AEH is the very backend of the interface. This must be installed on the Linux environment before the interface can be used. Java provides the capability of gaining access to the runtime and issuing commands to the software.

The basic low-level component of the software is the Command interface. The software suite consists of several commands that can be executed in succession. The goal of designing an interface for commands allows a standardized approach to managing commands in the software. Every command has a variable amount of arguments and



Command

- compile()
  - compiles all the arguments into one command string
- execute()
  - takes the compiled command and executes it on the system
- type()
  - returns the type of the command
- getVars()
  - returns a list of the variables this command accepts
- setDir(String s)
  - sets the active directory to execute commands from
- getVar(String s)
  - returns the variable specified by s.

Figure 2. Command Interface Structure

methods needed to get and set the arguments so a discrete object would not suffice. Seven different commands exist in the program. The command interface consists of six methods as noted in Figure 2. Every command needs to be compiled and executed. The compilation of commands creates the formatted command that can then be executed by the program. The benefit of this allows a user to dynamically create commands and execute them without having to know the order of arguments or the specific format.

The next level up is the JobUnit object noted in Figure 1. A JobUnit consists of a list of commands. When using Para-AEH several commands must be executed in succession. By creating a JobUnit a user could set up the different commands that would be executed in the order defined. In this fashion a user would define all the arguments and commands to be used beforehand in one packaged unit that could either be executed later or immediately via the JobQueue. The object also has the attributes of identifying the owner of a job and giving a descriptive name.

Once a JobUnit object is created it is then submitted to the JobQueue. This is a singleton object to make sure consistent access is enforced the JobQueue is thread safe to prevent race conditions in modifying and accessing the queue. The singleton nature allows only one queue for all users to submit to. This prevents the corruption or loss of the queue. A thread is assigned to manage the queue. The thread allows for the asynchronous nature of execution for commands. The user can work on a separate thread while the execution takes place allowing increased productivity without several command line instances or waiting till the end of execution before the next command. Figure 3 represents the control flow of this thread. The base structure is a loop that checks to see if there is a JobUnit available to be executed. The code execution will branch depending upon if the queue is empty or not. If it is empty the thread will wait until it receives a notification by the primary thread that a new JobUnit is available to be processed. This will prevent a waste of resources of continuously looping.

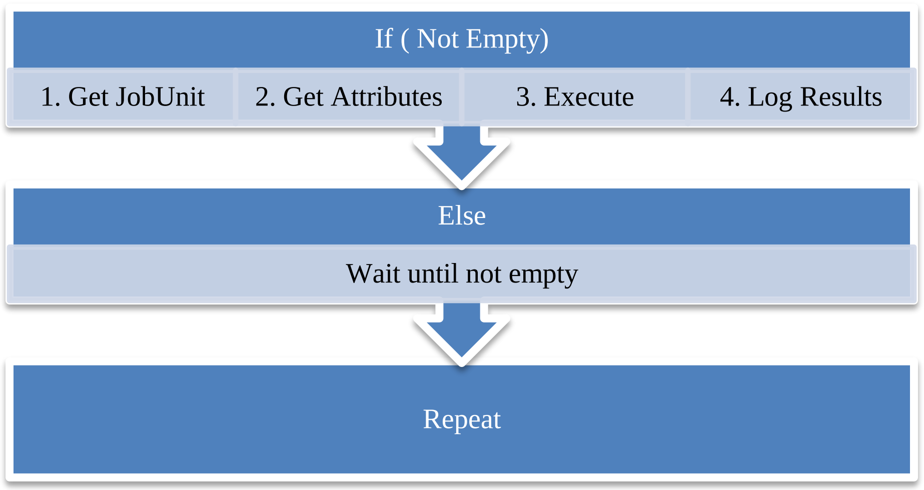


Figure 3. Basic Execution Structure of Job Queue Thread

In order to save the states of jobs or available users we must have some concept of serialization. Java Object Serialization is not an efficient method of storing and loading data into a program so an XML based approach was preferred. XML is one of many standards designed to represent data it is independent of platform and software. This standard allows easy parsing of data and representation of a tree based structure [9]. Two classes were created to handle the saving and opening of the XML files necessary for operation: JobXML and UserXML.

In order for users to interact with the software we must have the ability to store and load a list of authorized users of the software along with their attributes. We can leverage the tree style of structure of XML to store this data efficiently by following the XML Document Object Model (DOM). The DOM treats an XML file as a tree data structure a programming language can implement this model to allow the dynamic creation and modification of XML files [10]. The tree structure of the XML file that stores the users is presented in Figure 4. The root of the tree represents the list as a whole by the node UserList. Below that we can have multiple User children nodes dependent upon how many users we have on the system. Each User has three key nodes. Name represents the username of a User object. Password represents their account password for Para-Interface. The final node represents the directory which a user's data will be stored.

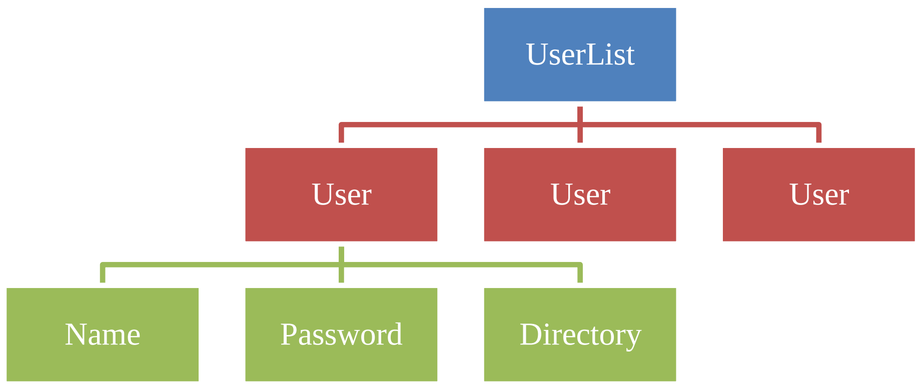


Figure 4. Hiearchary of User XML file

The JobXML class is a bit more complex due to the variable nature of commands. The XML file maintained by this program represents a list of jobs that a user may wish to save for future execution or modification. The key structure of this tree is the base node can have any number of children Job nodes with attributes representing the owner and name of the JobUnit object. Each Job can have multiple Commands. Each Command node has the attribute that defines the type. The type attribute will allow the software to determine which set of rules to parse or save the data with due to the variable nature of arguments.

In order to make an interface that is easy accessible a web based interface was decided to allow several researchers to access it at the same time. Web interfaces allow asynchronous access to an application without direct access to the machine hosting the application. Zurb’s Foundation framework was leveraged in designing an adaptive page that is accessible to a multitude of platforms. The core idea behind Foundation is a grid based structure that allows a dynamic organization of elements. The elements provided by Foundation allow a graphical experience for web elements that expand upon the base html code structure. Foundation also heavily relies upon JavaScript capabilities to provide a user friendly experience with advanced features [11].

Website Front End

In order to merge the Java backend program with the frontend Foundation web interface a technology called Java Servlets was leveraged. Java Servlets allow the mapping of classes to specific URL. Servlets act as a middle layer between a browser and a backend program. Using servlets allow access to Java programs and functionality. Servlets often implement methods that interact directly with HTTP requests that follow a doMethod() format where Method represents the HTTP request methods. Servlets provide several useful features such as form processing, dynamic page building, session tracking, page redirecting, and modular based design. These features were all necessary in order dynamically manage the interface. Servlets are commonly deployed on Apache Tomcat servers which provide a portable web server. [12].

When a browser is directed to any page of the web application a user will automatically be redirected to the Login page as noted in Figure 5. At this time a user will automatically be redirected to another port to initiate TLS. This prevents traffic to and from the site from being monitored. This allows the secure transfer of data related to the research being done. we see a basic form to get credentials from a user. The form when submitted creates a POST request to post the credentials for an authentication challenge. If the challenge is passed we open a new session so a user can maintain access throughout several pages. Before any page, besides the login page, can be accessed, the session is checked to verify if the user is logged in.

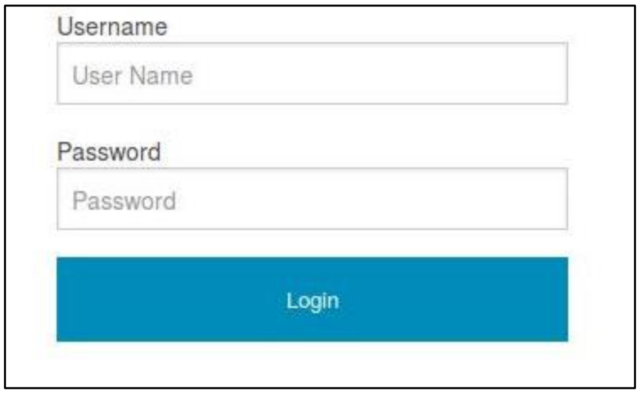


Figure 5. Login Screen

Upon logging in a user is redirected to the main menu screen. Here a user is offered with 4 primary modules to interact with. The design can be seen in Figure 6. Each of these modules allows a researcher to interact with the backend software. In the menu we have the Foundation grid system managing the content. Each module is hosted in its own servlet page to keep the source modular. Foundation interchange is leveraged to allow pages to dynamically be hosted inside of other pages with greater ease than using HTML iframes. The primary frames include: Job Logs, Saved Job Editing, Job Data Download, and the Para-View Web Interface button.

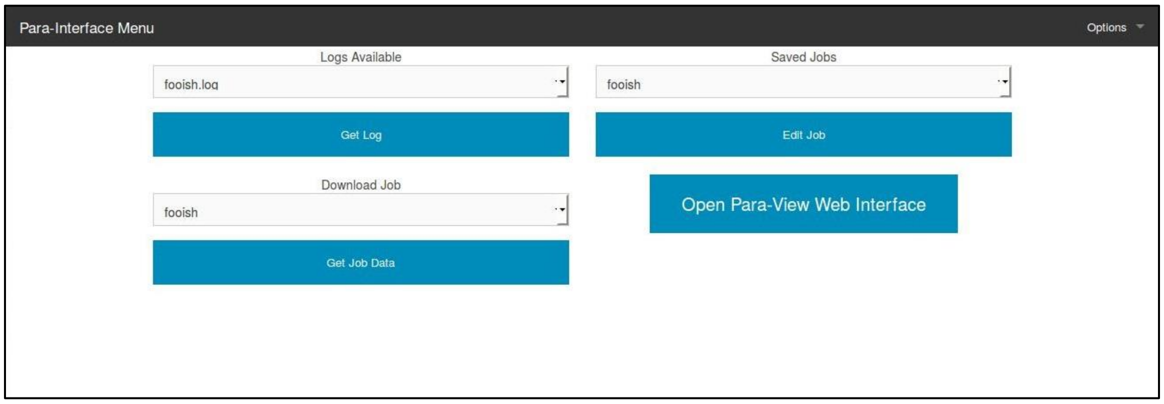


Figure 6. Main Menu Page of Web Interface

Figure 7 details the options drop-down box on the main menu screen. These options deal with user management and job creation. These interactions were best suited for the options drop-down box. A user can add another user via the options menu provided or change their credentials. There is also an option to logout of the application which invalidates the session and prevents unauthorized access. The job creation option is a bit more complex than the other features. Upon interacting with the job creation feature a user is presented with a foundation reveal modal. This screen asks a user to name the job they wish to create as well as what commands a user wishes to add to the job. There is also an option for a user to save the job after creation for future manipulation. Upon submitting the form a user is redirected to a new page which is dynamically populated with fields to fill out for the research job. The servlet that manages page uses form data from the previous page to determine which variables and options will be necessary for the job to be run. All available variables to the commands selected are presented. Set functionality was implemented in the backend servlet to prevent variable duplication for options that are used in more than one command.

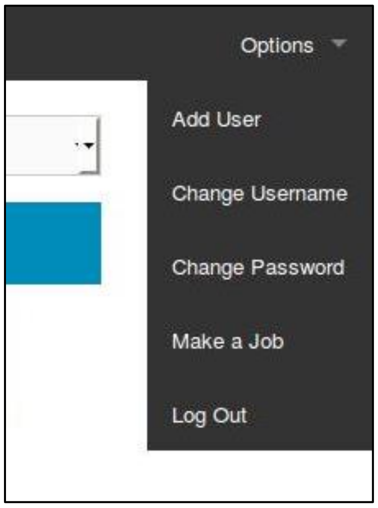


Figure 7. Main Menu Options Toolbar

Once a user is finished providing the information needed for their job results are POSTed to a page for processing. Java methods are called to parse the data to dynamically create the JobUnit object from a list of Commands. Once the object has been created it is submitted to the JobQueue for execution asynchronously. The user is then redirected to the main menu for further interaction with the application. A user can then not have to worry about waiting for the commands to be run before leaving or moving onto creating a new job.

Referring back to Figure 6, the first module to be discussed is the log module. As a job is being executed asynchronously by the backend threads, a log is created for reference. This log follows the format of *Job.log* where Job represents the JobUnit's name. In the top left corner a dynamic drop-down box is created that represents all logs that a User owns. A user's directory is dynamically searched to discover what all logs are available to be examined. Once a user selects a log to examine, the page creates an AJAX POST request. This allows a log to be loaded asynchronously loaded into a Foundation reveal modal. The reveal modal is then revealed so the user can examine the results of previous jobs.

The next module consists of downloading a job's data. As with the log module a user's directory is searched to see all jobs available. The difference in this module is the fact it allows a user to download the job's associated data as well as the log file itself. On the backend there is a servlet that manages requests. The program searches for the associated job and its log. A command is sent to the associated computer's shell to create a zip archive for ease of portability and ease of download. Zip files are highly portable and easy to access. The reason for adding this capability to download data is to allow a user to access the data associated with a research job while protecting the data on the backend server itself.

The third module provided allows a user to interact with saved jobs. This allows a user to create a job and save it for future modification. As noted in Figure 8 the menu provided by this provides four main pieces of functionality: Renaming, Editing, Deleting, and Running. Renaming the job will rename the job as it is stored within the system. Editing a job is a bit more comprehensive because it will allow a user to alter two main things. First a user can change the commands in a saved job. Commands can either be added or deleted from a job. Once the selection has be made a user will be redirected to a page that will dynamically populate itself with all possible variables to be assigned as well any previously stored data from the job. The motivation behind this capability is that users may wish to further fine tune parameters for a project or reuse old parameters. Saving a job and allowing the editing of the variables will decrease time spent recreating variables. If a user decides they no longer want to save an old job they can always decide to delete the job. Finally the next thing a user can do is after renaming the job or editing the job they may elect to rerun the job.

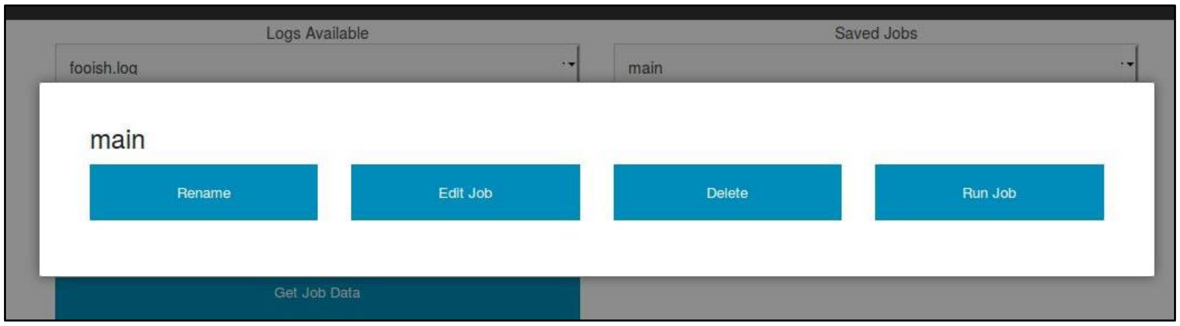


Figure 8. Saved Jobs Menu

The final module which exists within the bottom left area of the main menu in Figure 6 represents a tie in with a popular visualization application, Para-View. Para-View is a tool utilized researchers to quickly analyze and visualize data. Para-View is built on top of the Visualization Toolkit (VTK). One of the aspects of Para-AEH is to allow the creation of a VTK based file to allow researchers the option of visualizing the materials they are modeling in a research job. By integrating Para-View's web based framework a user can quickly interact with the results they received from a research job.

Without the integration a user would have to download the job data and open up Para-View locally. This allows users to access the visualization software regardless of the computer they are currently working from. This aspect of leveraging Para-View assumes that Para-View’s web interface is up and running on the proper port but once basic configuration is complete using Para-View via the web is trivial [13].

Research Interest

The Department of Mechanical Engineering's research primarily focuses on leveraging Para-AEH to model composite structures to learn their effective engineering scale properties. These fundamental properties allow researchers to develop and test new materials to see if these materials can withstand high stress or strain rate applications. Examples of this would include situations such as blast or impact scenarios. Advances in technologies such as additive manufacturing are allowing researchers greater microscale control of both material and geometry. These advances require the development of advanced computational material modeling and mechanics. These computational advances hope to improve accuracy while concurrently reducing time and money spent on the development process of potential armor materials. Oftentimes this type of computational modeling can lead to testing of new type materials for construction and manufacturing. Researchers can also test for substitution of materials in construction to help reduce R&D costs, improve production time, and enhance the fundamental knowledge required for advanced material design through designed properties [14].

Military Applications

Modeling materials has great military application that would be of interest for defensive and offensive capabilities. AEH methodology allows an in depth analysis heterogeneous material systems to understand how the inherent microscale geometry and property variations determine effective responses to external stimuli. This is important to know for both capabilities to appropriately plan. On the defensive side it is important for soldier and infrastructure protection to effectively model structures before they are deployed. Researchers can learn how effectively materials such as light-armor, heavy-armor, buildings, and vehicles can handle stresses and strains during extreme loading conditions. On the offensive side structures can be analyzed and modeled to see at what levels of strains and stresses causes a structure to fail. This can be used to determine the most efficient way to offensively target a structural weak point.

Composite structural analysis is one of many examples where the AEH method has proved useful in analyzing and testing materials for military application. Composite structures have in past proven to have high strength to weight ratios which are important in the aerospace and land vehicle industry. Compared to other methods it has been proven to give reasonable and currently less computationally intensive than classical finite element approaches to analyze composite laminates and can even provide additional information that is useful for a mathematical analysis of various microstructures. [15]

Conclusion

Developing an interface for an important research tool like Para-AEH will help increase research turnaround for those in the field of materials research. As noted many times there is a multitude of applications for Para-AEH from biological, mechanical, and defense industries. An intuitive interface reduces the basic structure of Para-AEH to a logical grouping of research jobs that contain commands that normally would have to be manually typed and inputted into a command line based approach. Not every researcher may be familiar with the Linux shell and to craft these commands may take several flags. Adding a graphical interface allows a simple action of filling out a form to dynamically craft the commands and asynchronously execute them while a researcher can begin work on another job, view results, or many other commands via the graphical interface. The Java Servlet technology leveraged in this project will hopefully be simplistic enough for researchers to understand the code base and modify if needed if the requirements for the interface changes in the future.

Bibliography

[1] J. J. Ramsey and P. W. Chung, "Para-AEH User Manual," U.S Army Research Lab.

[2] P. W. Chung, K. K. Tamma and R. R. Namburu, "Asymptotic expansion homogenization for heterogeneous media: computational issues and applications," *Composites: Part A, vol. 32,* pp. 1291 - 1301, 2001.

[3] C. L. Lawson, R. J. Hanson, D. R. Kincaid and F. T. Krogh, "Basic Linear Algebra Subprograms for Fortran Usage," *ACM Transactions on Mathematical Software,* vol. 5, no. 3, pp. 308-323, September 1979.

[4] "BLAS (Basic Linear Algebra Subprograms)," Netlib Repository at UTK and ORNL, 08 September 2014. [Online]. Available: http://www.netlib.org/blas/. [Accessed 10 March 2015].

[5] "LAPACK - Linear Algebra PACKage," Netlib Repository at UTK and ORNL, 16 November 2013. [Online]. Available: http://www.netlib.org/lapack/. [Accessed 10 March 2015].

[6] B. Schäling, "The Boost C++ Libraries," 2008. [Online]. Available: http://theboostcpplibraries.com/. [Accessed 11 March 2015].

[7] M. Heroux, "The Trilinos Project," 2015. [Online]. Available: http://trilinos.org/. [Accessed 11 March 2015].

[8] J. Gosling and H. McGilton, "The Java Language Enviroment," Oracle, May 1996. [Online]. Available: http://www.oracle.com/technetwork/java/index-136113.html. [Accessed 19 March 2015].

[9] "XML Tutorial," W3Schools.com, [Online]. Available: http://www.w3schools.com/xml/. [Accessed 19 March 2015].

[10] "XML DOM Tutorial," W3Schools.com, [Online]. Available: http://www.w3schools.com/dom/. [Accessed 19 March 2015].

[11] "Zurb Foundation," Zurb, [Online]. Available: http://foundation.zurb.com/docs/. [Accessed 11 March 2015].

[12] "Servlets Tutorial," Tutorialspoint.com, 2014. [Online]. Available: http://www.tutorialspoint.com/servlets/. [Accessed 16 March 2015].

[13] "ParaView," KitWare, Inc., [Online]. Available: http://www.paraview.org/overview/. [Accessed 28 March 2015].

[14] M. Nelms, *private communication,* April 2015.

[15] F. Rostam-Abadi, C.-M. Chen and N. Kikuchi, "Design analysis of composite laminate structures for light-weight armored vehicle by homogenization method," *Computers & Structures,* vol. 76, no. 1-3, pp. 319-335, 2000.

[16] J. R. Ramsey and P. W. Chung, "Massively Parallel Asymptotic Expansion Homogenization for Complex Microstructures," 2013.

[17] J. J. Ramsey, *private communication*, March 2015.