VLab: a Science Gateway for Distributed First Principles Calculations in Heterogeneous High Performance Computing Systems

A DISSERTATION
SUBMITTED TO THE FACULTY OF THE
UNIVERSITY OF MINNESOTA
BY

PEDRO RODRIGO CASTRO DA SILVEIRA

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS
FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY

RENATA WENTZCOVITCH

February 2014

Acknowledgments

I want to thank Professor Renata Wentzcovitch for her advice and support during this work, Professor David Yuen for his collaboration and spirit of research, and my former undergraduate Physics instructor, collaborator, and adviser, Cesar da Silva, for his contribution to and kindness during my work. I also want to express my sincere thanks to Minnesota Supercomputing Institute staff members for helping me materialize this work.

Abstract

This thesis describes the development and deployment of a cyberinfrastructure for distributed high-throughput computations of materials properties at high pressures and/or temperatures – the Virtual Laboratory for Earth and Planetary Materials – VLab. VLab was developed to leverage the aggregated computational power of grid systems to solve intrinsically high-throughput problems that appeared practically intractable in materials computations for planetary science applications. The VLab cyberinfrastructure was developed as a service oriented architecture (SOA) and consisted initially of a web portal, web services, and databases. These components are all integrated and used in the deployment of complex distributed workflows for calculations of equations of state, elastic constant tensor, and thermodynamics properties of materials. The deployment of workflows for high throughput calculations through the Internet transmuted VLab into a Science Gateway. Its development preceded the existence of software frameworks for workflow or gateway development. It is a unique Science Gateway deployed in the Extreme Science and Engineering, XSEDE.

Table of contents

List of Figures

List of Tables

List of Acronyms

VLab	Virtual Laboratory for Planetary Materials
SG	Science Gateway
VLab SG	VLab Science Gateway
VLabWs	VLab Web Site
SOA	Service Oriented Architecture
MSI	Minnesota Supercomputing Institute
VLabWp	VLab Web portal
PI	Principal Investigator
QE	Quantum Espresso
XSEDE	Extreme Science and Engineering Development Environment
SGI	Science Gateway Institute
WS	Web Service
HTC	High-throughput calculations
HPC	High Performance Computing
DFT	Density Functional theory
CPU	Central Processor Unit
MPI	Message Passing Interface
OpenMP	Open Multi-Processing
BoT	Bag of Tasks
Cij	Static Elastic constant
EOS	Equation of State
VDOS	Vibrational density of state

MPSC	MgO Pressure Scale Calculator
PPSC	Platinum Pressure Scale Calculator
LDA	Local density approximation
QHA	Quasi-harmonic
PBE	Perdew-Burke-Ernzerhof
ThoM	Thermodynamcs of Minerals
ThoM-UHP	Thermodynamics of Minerals at Ultra-High pressure
VLab CIFPC	VLab Cyberinfrastructure for First Principle Calculation
PBS	Portable Batch System
PE	Project Executor
TD	Task Dispatcher
TE	Task Executor
API	Application Programming Interface
JSR	Java Specification Request
JSP	Java Server Page
NF	Normal Form
WSDL	Web Service Description Language
SOAP	Simple Object Access Protocol
PWscf	Plane-Wave Self-Consistent Field
Ph	Phonon
HighPT EOS	High Pressure Temperature Equation of State
DHT	Distributed hash table

List of software names and their definitions

PostgreSQL	Open source database capable of store data in tables, search for data using complex structure query language SQL, and easy to integrate into Internet application such as web portal and web service.
Axis 2.0	Web Service framework
QE	Quantum Espresso package
Maven	It is a project management tool mostly used for compilation, organize code dependency, and easy to deploy.
Castor	It is a Java library used to convert JavaBean into XML and vice-versa.
SourceForge	It is a web based source code repository; it helps maintain the most recent code into one single location, track user changes, and easy access to deploy.
Apache Tomcat	It is an open source web server and servlet container used to integrate Web Portal and web services into the web.
GridSphere	It is a deprecated open source framework that integrates web pages into small portlet. It has the benefits of having many functional portlets in one single web page.
Python	It is a high level programming language used for scientific development
Matlab	Numerical computing environment with features capable of producing scientific applications.
PBS	Computer software platform that perform job scheduling in resource grid.
SLURM	An open source computer software platform that perform job scheduling in resource grid.
Load Leveller	Initially developed only for Sun OS, it is another computer software platform that performs job scheduling in resource grid.

The members of the committee

Professor Renata Wentzcovitch
Dissertation Adviser

Professor Dave Yuen
Committee member

Professor Paul Woodward
Committee member

Professor Abhishek Chandra
Committee member

Chapter 1

1. Introduction

This thesis describes the research involved in the development and deployment of a pioneering cyberinfrastructure for distributed high-throughput computations of materials properties at high pressures and/or temperatures – the Virtual Laboratory for Earth and Planetary Materials, ***VLab*** [1]. Today, ***VLab*** is a Science Gateway hosted by the Minnesota Supercomputing Institute (MSI) [2]. It has been developed under NSF/ITR and NSF/EAR support.

VLab was conceived as a means to leverage the aggregated computational power of grid systems to solve intrinsically high-throughput problems that appeared practically intractable in materials computations for planetary sciences applications. These are essentially problems related with phase space sampling, i.e., computations involving sampling of points in multi-parameter spaces of physical variables (volume, temperature, pressure, strain, atomic configuration, vibrational mode, crystal structure, etc..). In addition, ***VLab*** was envisioned as an Internet launch pad for these distributed calculations consisting of portal, web services, and databases integrated in multiple ways. The development of workflows for high throughput calculations and their deployment in the Internet transmuted ***VLab*** into a Science Gateway. Its development preceded the existence of software frameworks for workflow or gateway development. It is a unique service oriented architecture (SOA) deployed in the Extreme Science and Engineering Development Environment, XSEDE [3]. However, it has been fully functional in MSI systems since well before the creation of the XSEDE.

This document is organized as follows: section 2 gives a short explanation of Science Gateways (SG) and puts the ***VLab*** SG in context. It also describes its overall organization and the ***VLab*** website components. The ***VLab*** website (***VLabWs***) is the public access portion of the ***VLab*** SG. This section ends with a

summary of the usage and impact of the ***VLab*** SG. Section 3 describes the cyber-infrastructure for first principles calculations – the ***VLab*** web portal. Access to this part of the ***VLab*** cyber-infrastructure requires authentication, and its usage requires expertise in first principles calculations. This section describes the technology used in the development of the ***VLab*** web portal (***VLabWp***), the main types of web pages, the metadata schema, the main web services, and ends with a description of code organization. Section 4 describes the scientific workflows deployed in ***VLabWp*** and tools used for workflow management. Section 5 describes the development of an internal scheduler to coordinate execution of multiple high-throughput workflows by multiple users in distributed environments, currently MSI and XSEDE. Final remarks and an outlook of future work are presented in section 6.

Chapter 2

2. The Virtual Laboratory for Earth and Planetary Materials – *VLab*

2.1 What is a Science Gateway?

Online science communities emerged and grew considerably in the last decade along with the popularity of the Internet. The science gateway (SG) idea was not totally formalized until late in 2010 when many well-funded scientific projects, initially intended to leverage scientific communities, started to use software applications interacting over the Internet consistently and benefited from an improved development of middleware applications. It was noticed that entire communities could benefit from resources placed online for analysis, teaching, learning, sharing of ideas, etc.

In general, a Science Gateway is a community-developed set of tools and data collections integrated in a Web Portal by a suite of applications. Gateways provide access to a variety of capabilities including, workflows, visualization, resource discovery, and job execution service [4]. An SG can be a Web Portal or a mobile application providing, for example, access to digital services of research materials, execution of specific scientific applications, e.g., Matlab or Quantum ESPRESSO (QE) software [5], or providing community-specific engaged discussions, research, teaching, and learning. Popular SGs are GENIUS/Engin Frame [6], Computational Infrastructure for GeoDynamics [7], Network for Computational Nanotechnology and nanoHUB [8]. With the increased number of SGs and the immersion of SGs into social networks in the US, some challenges regarding sustainability of all the software developed became apparent. Thus, a group of SG developers and project investigators (PIs) decided to create a Science Gateway Institute (SGI) [4]. The SG Institute aims to incubate services, extend gateway development teams, gateway forum, and work force development. The perspective and acceptance of the SG Institute is positive

among researches benefitting from SGs. The SG Institute hopes to establish itself as a community to define patterns and tools used for development and guarantee sustainability for continuous development of SGs.

It is important to mention some roles in the development and utilization of an SG. Unlike traditional open source software, where the code is distributed and researchers change and maintain their own versions of the software based on their needs, an SG is maintained by a single or a team of developers with an undisclosed source code that is only modified based on a science community needs and requirements. Figure 1 provides a simple overview of each role in an SG environment. The responsibility of each member is not limited to what is described in Figure 1. However, the main duties in an SG development team are summarized to clarify the basic attribution of development team members in an SG.

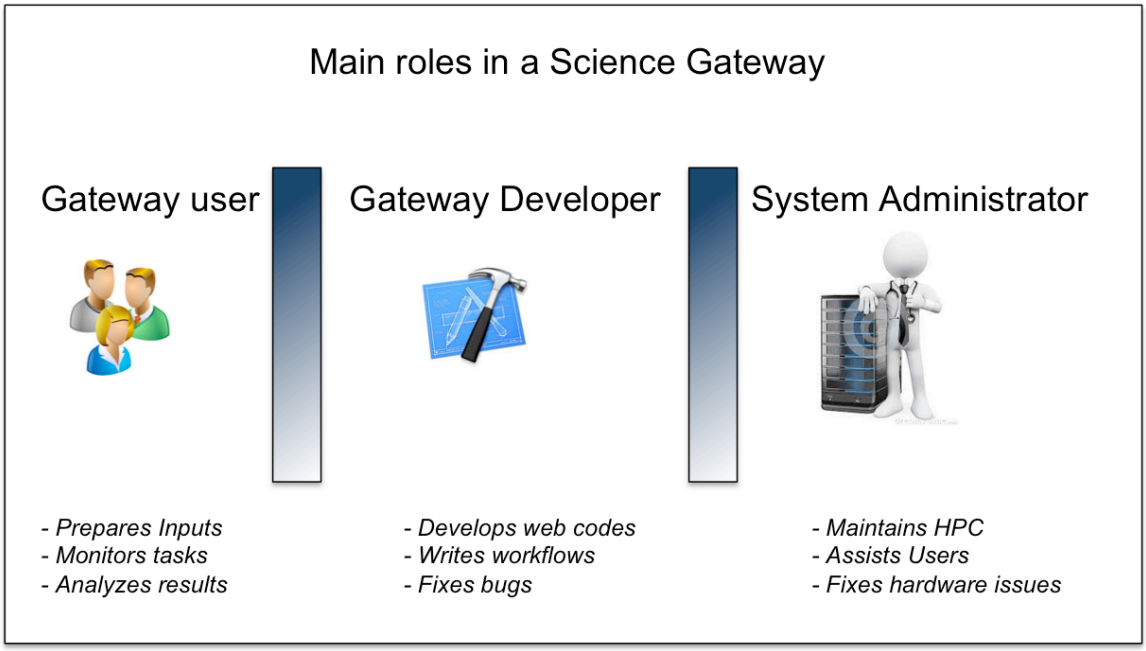


Figure 1. Main roles in a Science Gateway environment.

During the development of an SG, the software engineer or the gateway developer interacts directly with a knowledgeable user of the software supported

by the SG. The gateway developer implements workflows to facilitate tests by a gateway user in a collaborative environment. The system administrator role is to provide access to the main resource and assist with issues regarding software installation. In the particular case of the ***VLab*** SG, gateway users consisted of students and post-docs in the Wentzcovitch group. Myself, the author of this thesis, was the sole gateway developer and the system administrator of the ***VLab*** server where databases reside. The system administrator depicted in Figure 1 also represents the contact person of the computation center where high performance computing (HPC) resources are utilized and where web services resides.

An SG also fulfills important pedagogical roles. It can **i)** help students climb more easily steep learning curves by abstracting the complexity of calculations that happen behind a web portal, e.g., by eliminating the need of users to specify compilation parameters, to figure out the right optimization of a tool, to understand queue policies in batch systems, parallelization parameters, etc; **ii)** it can facilitate the execution of complex and large sequences of operations by providing predefined workflows allowing users to skip development of new scripts and other formalities involved in job submissions; **iii)** it can save organized sequences of inputs/outputs performed in a workflow inside of a database that can trace back the entire history of the calculation; **iv)** it can provide an interactive outlet for detailed reproduction online of important results for analysis before and after publication. In a nutshell, SGs allow researchers to focus on science and complicated problems instead of spending time on issues related with technologies behind complex computation.

2.2 The *VLab* Science Gateway

The Virtual Laboratory for Earth and Planetary Materials, ***VLab***, supported by the National Science Foundation and hosted by the Minnesota Supercomputing Institute at the University of Minnesota (MSI), is a Science Gateway dedicated to the development and promotion of the theory of planetary

materials [1]. Computational determination of materials properties at extreme conditions is fundamental to geophysics. It provides today and maybe for a long time to come, the most accurate information for interpretation of seismic data and for use as input in more sophisticated and reliable modeling of planets. The laboratory was established in 2004 with the aim of accelerating developments in this emergent area by addressing materials physics and physical chemistry issues of importance to planetary sciences by:

• Developing and improving first principles simulations methodologies, integrating highly tested first principles software with utility programs, and creating novel human/software interfaces to facilitate and automate time-consuming human tasks.

• Developing an educational program to provide training and bridge the gap between mineral physicists and materials theorists.

Since 2010, the ***VLab*** development has been funded primarily to implement workflows for sophisticated distributed high-throughput calculations (HTC), most recently in the XSEDE [9]. Numerous other SG projects have recently been developed and are listed in the XSEDE website [10]. Among them, the Computational Infrastructure for GeoDynamics [7] and the Network for Computational Nanotechnology and nanoHUB [8] are relevant to our efforts since they have some overlapping scientific interests. ***VLab*** was the first to enable distributed HTC-HPC calculations in the XSEDE environment – perhaps still is the only one.

Distributed computing emerged as a virtual platform to leverage computing power integration over a networked resource. It is most commonly used in Cloud Computing through the Internet and in HPC platforms in private networks. HPC is most often used to solve complex large-scale scientific problems in different science communities, e.g., by physicists, biologists, astronomers, etc. In this work, HPC is mainly used to facilitate and increase speed of calculations of material properties at high pressures and high

temperatures. In addition to the complexity HPC brings to distributed computing, numerous other issues emerge related with auxiliary software usability of HPC systems, e.g., software for queuing systems, compilers to interpret multiple processors, software to map multiple disks, and integration of complex network topologies. Therefore, nowadays hundreds of software applications for HPC are being developed mainly to help users interact easily with distributed computing, monitor compute nodes, submit tasks for large scale problems, manage transfer of large files, visualize results of simulations, and create scientific workflows. In this thesis most of these applications have been combined under the ***VLab*** SG.

The scientific goal of calculations performed in ***VLab*** is to obtain magnetic, thermodynamics, and thermal elastic properties of solids for geophysics or materials science applications using the QE software. QE [5] is an integrated suite of open source codes for electronic-structure calculations and materials modeling, implemented using density functional theory (DFT), in standard and extended forms, and the pseudopotential-plane-wave method. No input parameters other than fundamental constants of nature are required for these calculations and they are referred as “first principles”. Realistic, complex, and demanding computations of materials’ properties by first-principles need efficient and reliable executions on complex software-hardware infrastructures. On the hardware side, CPU and memory scaling for running optimally the QE vary by orders of magnitude depending on the type of material and/or calculated property. They increase with the number of atoms in the unit cell of a particular system. On the software side, running QE on supercomputers is an intricate task that requires Message Passing Interface (MPI) and/or OpenMP parallelization strategies and involves QE compilation issues, such as specific processor brand and Fortran compiler version. For each materials property, the number of tasks involved varies with the complexity of the calculation, the range of thermodynamic conditions, number of different QE codes required, etc. These tasks must be decoupled or, at least, loosely coupled in such a way that they can be executed also in distributed environments. High performance on massively

parallel architectures is achieved by distributing both data and computations in a hierarchical way across available processors, ending up with multiple parallelization levels that can be tuned to the specific application and to the specific architecture.

Complex simulations, such as those performed in ***VLab***, usually require execution of hundreds of tasks, preparation of inputs, post-processing of results, comparison of data, and sometimes repetition of the process with different parameters until all steps are successfully executed. It can be burdensome to perform these activities manually since the preparation and handling of large numbers of tasks are very error prone. Some researchers develop their own workflows using script languages, e.g., Unix/Linux shell script, Perl, Python, and Ruby to connect directly from a Terminal Window application to HPC compute nodes. Others, along with assistance from computational scientists, prefer to create their own workflows using workflow framework systems Pegasus [11], Kepler [12], Swift [13], and Apache Airavata [14]. These workflow framework systems usually have full compatibility to distribute tasks on HPC nodes, provide options to control data flow, design workflow using graphical user interface, define tasks dependencies, organize metadata into databases, and use SG as a front-end. Most workflow framework systems, as those mentioned above, do not have scheduling of tasks for scientific workflows as described in this thesis. Although, they do have resource allocation capability and options to manage scientific workflow tasks in an orderly manner, these frameworks expect the user to define which resource to use every time they are needed. They basically rely on the user to check the availability and compatibility of every resource before submitting tasks.

In this work, the distribution of tasks in heterogeneous HPC resources is made through the ***VLab*** SG. The ***VLab*** SG is a service oriented architecture (SOA) with a web portal as the main interface for handling scientific workflows for first principles calculations at high pressures and temperatures. It uses web services for metadata management, job submission, task monitoring, and task

interaction. ***VLab*** runs specific scientific workflows for calculations of thermodynamics properties, equation of state, and elasticity of materials [15]. The ***VLab*** SG has a set of just-in-time scheduling policies for each project submitted. Thus, the user can choose how tasks should be submitted to HPC systems, or allow the ***VLab*** SG decide which resources to allocate calling predefined algorithms using bag-of-tasks (BoT). For example, when a project phase consisting of N independent tasks is submitted, all N tasks might be placed into a single or multiple BoTs submitted to one or multiple HPC resources. This way, the required resource allocation can be easily calculated by multiplying the number of tasks, N, by the number of cores per task. Depending on availability of resources, BoT sizes and numbers can be defined using various criteria. One of the charges of the ***VLab*** task scheduler is to define the most efficient use of multiple BoTs (by multiple users) to leverage jobs distribution in heterogeneous HPC systems.

The ***VLab*** SG consists of two distinct parts: **i)** the ***VLab*** cyberinfrastructure for distributed first principles calculations accessible through the ***VLab*** web portal (***VLabWp***), where quantum mechanical calculations of material properties are performed by workflows using web services. These are usually long sequences of complex and large scale high throughput HPC calculations; **ii)** the ***VLab*** website (***VLabWs***) containing tutorial course contents, open source software available for download, interactive applications to reproduce and/or expand on fully or partially published first principles results, visualization tools for inspection of these results, online resource applications such as pressure scale calculators, a search engine to access raw data, i.e., organized sequences of input/output from large scale calculations (projects), etc.

The ***VLab*** CIDFPC, henceforth referred ***VLabWp***, was developed as a service oriented architecture cyberinfrastructure [16] to manage the execution of extensive workflows of calculations in distributed environments. The ***VLabWp*** enables users to perform first-principles calculations in an easy and transparent way when compared to manual execution of these workflows – commonly Unix

shell scripts executed in a single workstation. A complex workflow may use several large files, e.g., HPC codes, multiple inputs and outputs and a plethora of post-processing codes with auxiliary files to create new input files for subsequent phases of the workflow from outputs of previous phases, etc. In the ***VLabWp***, the workflow complexity is hidden from users, and all the controlling is done through a Web Portal sending requests to Web Services. Workflows implemented in the ***VLab*** SG have the following features implemented: they can execute bag-of-tasks in a pool of distributed servers, manage metadata to access previous results, do post-mortem statistical analyses of execution attributes, post-process intermediate data for analysis during workflow execution, visualize final and intermediate results in the Web Portal, and export results to other applications.

2.3 The *VLab* website components

The ***VLab*** website contains information and tools useful for high pressure and high temperature calculations in materials science and mineral physics. As with most SGs, it contains downloadable free software, tutorial lectures and notes, publications list, events details, etc. Specific to ***VLab***, there is a set of tools available online with easy interaction for users interested in research performed by those registered in the ***VLab*** portal. These tools allow users to interact in details with calculations performed in ***VLab***, but whose results could not fit in a finite number of pages of a journal publication. The idea is not unique, but it is common in geophysics. For example, tomography models report 3D fields that can only be visualized in publications. Numerical details of these models are made available in the form of downloadable files. A similar idea applies to thermodynamics and thermal elastic properties of materials presented in the ***VLab*** website, where not images but only linear plots of limited results can actually be published. Thermodynamics and thermal elastic properties of minerals are extremely useful as input for sophisticated geodynamics simulations and are absolutely necessary for interpretation of seismological models. These properties can also be visualized online. Other interactive resources include

applications concerning pressure scale calculations (MgO and platinum) and a database search engine. In the subsequent subsections, I describe these resources available in the ***VLab*** website and the analytical data showing how much these resources have been used.

2.3.1 Database Search Engine

Generally, after some long sequence of calculations organized as a "project", e.g., a calculation of thermodynamics or thermal elastic properties performed in a distribute HPC system, all files created and data entered in the simulation input page are usually understandable only to the author who created the project. Outputs of long HCP calculations should be preserved for easy access and to avoid future repetitive executions. Problems usually arise in identified a posteriori file names, calculation parameters and execution parameters, software version, etc. These issues can affect results, performance statistics, etc. Complete information about a project can be addressed by organizing data about the whole simulation – metadata – into a database. The metadata can point to all the data (input, output, pseudopotentials, executable version, post-processing results, etc…) by organizing the data into a single format and enabling a search engine capability to look up for information on simulations performed.

The *Database Search* is a web application designed to search for results of calculations performed in the ***VLab*** portal. A user can search for projects whose information has been stored in the database using criteria such as chemical formula, structure, exchange correlation functional, material name, author (first or last) name, project type, year created, etc. Once the *Search* button is hit, the Java code in the background of the web page creates dynamic queries to the PostgreSQL database, retrieves results, and makes it available to the user in a friendly interface developed in Java Server Pages. Within the result of a search, the user can see all steps executed in the project, can navigate onto the execution packages, can see detailed information about the project execution

and inputs and output, and can also check properties information about the project submitted (e.g. computational requirements, QE version, project classification properties, etc).

The search result shows a summary of stored data containing occurrences of “author first name”, “author last time”, “project title”, “project type” (Cij, Equation of State, Thermodynamics), “chemical formula”, “material name”, “structure name”, “exchange correlational functional”, etc. Figure 2 shows the database search interface. This resource is open and freely available for consultation without requiring authentication in the ***VLab*** portal. However, if a user wants to look up the details of the project, an authorized username and password must be provided.

2.3.2 Online applications with visualization

A set of web pages was designed to perform interactive calculations of thermodynamics, thermoelastic, and ultra-high-pressure properties of minerals. These web pages are very intuitive and have instructions on how to perform smoothly online calculations. Today, this is possible for a series of minerals for which first principles calculations of vibrational density of states (VDOSs) have been performed and published. These calculations are usually quite costly. Instructions and research articles related to these first principles calculations are also available. These interactive calculations permit published results to be reproduced and/or tested online for accuracy by choosing different equations of state, pressure and temperature ranges, methods used in numerical algorithms, etc. They also provide in customized form large amounts of results that could not be published in the limited space available in journals.

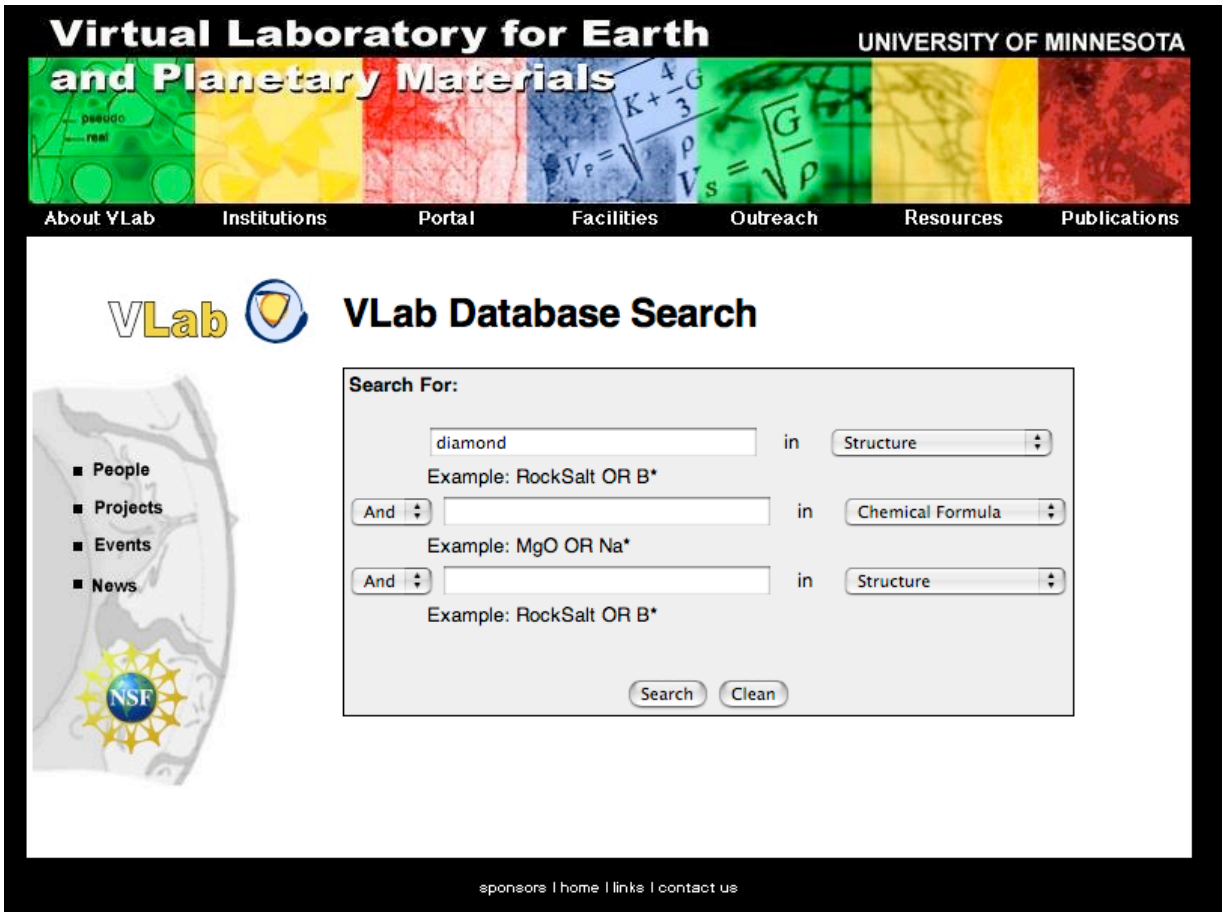


Figure 2. Database Search Interface.

In the initial web page the user sees the list of minerals available and can click over the selected mineral name to launch calculations of that mineral's (e.g.) thermodynamics properties. This action opens a web page presenting a form for the user to define parameters such as temperature range and temperature grid for finite difference calculations of thermodynamics properties, and temperature and pressure grids for plotting desired properties.

Figure 3 illustrates the input form for calculations of thermodynamics properties of Mg_2SiO_4 forsterite (LDA), the magnesium end-member of olivine, the most important mineral phase of the Earth's upper mantle. This information is then captured in a text file by a Java software (a Java class) encapsulated in a Java Server Page. The calculation is performed by a code developed in C

language - *thermo.c* - that computes quasiharmonic free energy and all thermodynamics properties that can be derived from it.

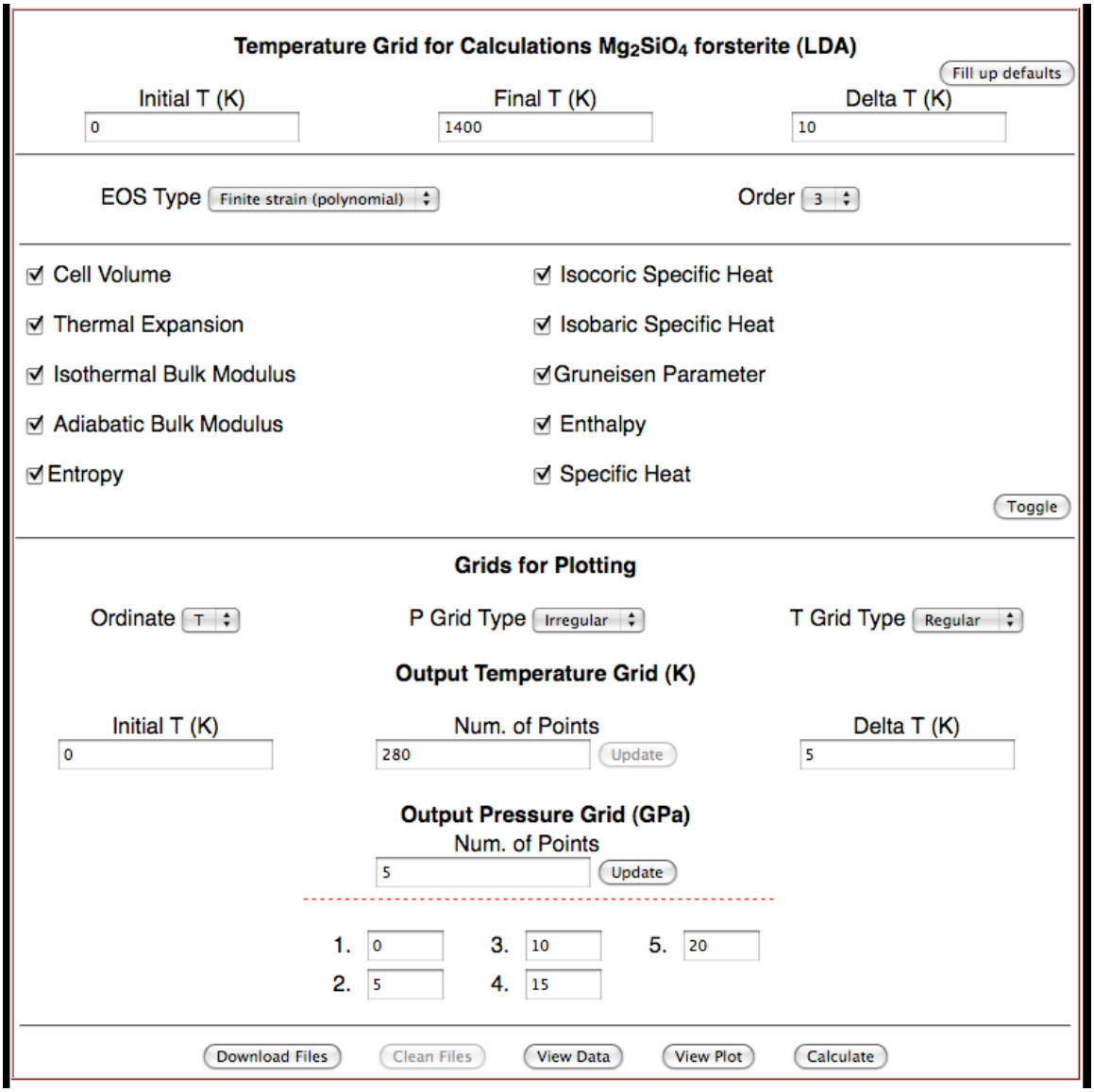


Figure 3. Input form page where the user enters information for calculations and information on how results are to be displayed.

The program captures in text format information entered in the webpage by the user and uses a previously generated file containing the VDoSs of the chosen material. A Java class then stores all output files paths in the Local Web Container. These output files are organized in a table format delimitated by

spaces. Results in these tables are visualized in the ***VLab*** SG using Gnuplot 4.2.2. The Gnuplot scripts containing input and VDoS paths are generate in the background of the web page using Java. The scripts are executed in temporary directories to distinguish multiple sessions accessing the interactive visualization page. Each user session is associated with a unique temporary directory to avoid data overwrite. Inside these temporary directories, Gnuplot creates graphic files in JPG format, which are then displayed in the visualization web page. A controlling mechanism to edit the graphics was created by allowing the Gnuplot to keep processing scripts continuously to update and modify the JPG files. Figure 4 depicts the procedure described above.

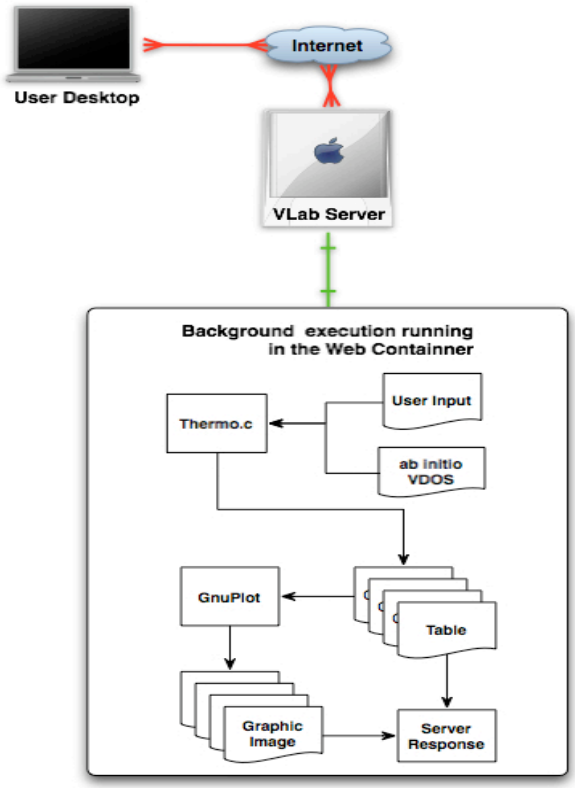


Figure 4. Information flow behind the execution triggered by the user request.

The JPG files created by Gnuplot are displayed in a separate display window along with the controlling mechanism input as shown in Figure 5. In this window the user can browse and edit the graphic to change several details such

as thickness of borders and lines, font type, font size, titles, etc. Outputs containing all thermodynamics properties of interest in the desired display format (plot versus P or T in course or sparse grids in T or P, etc), can also be retrieved in form of HTML tables displayed on the screen or to be downloaded.

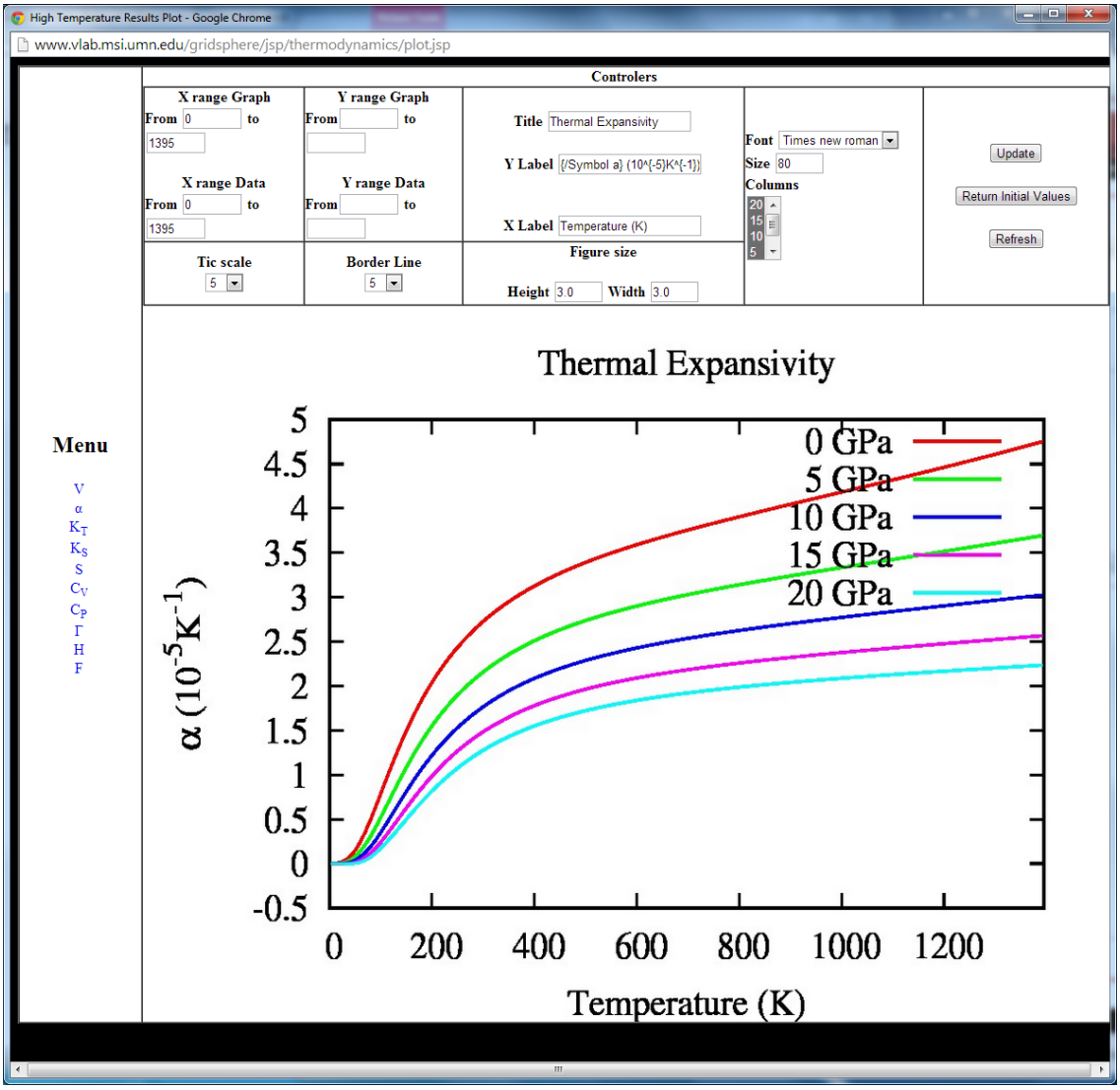
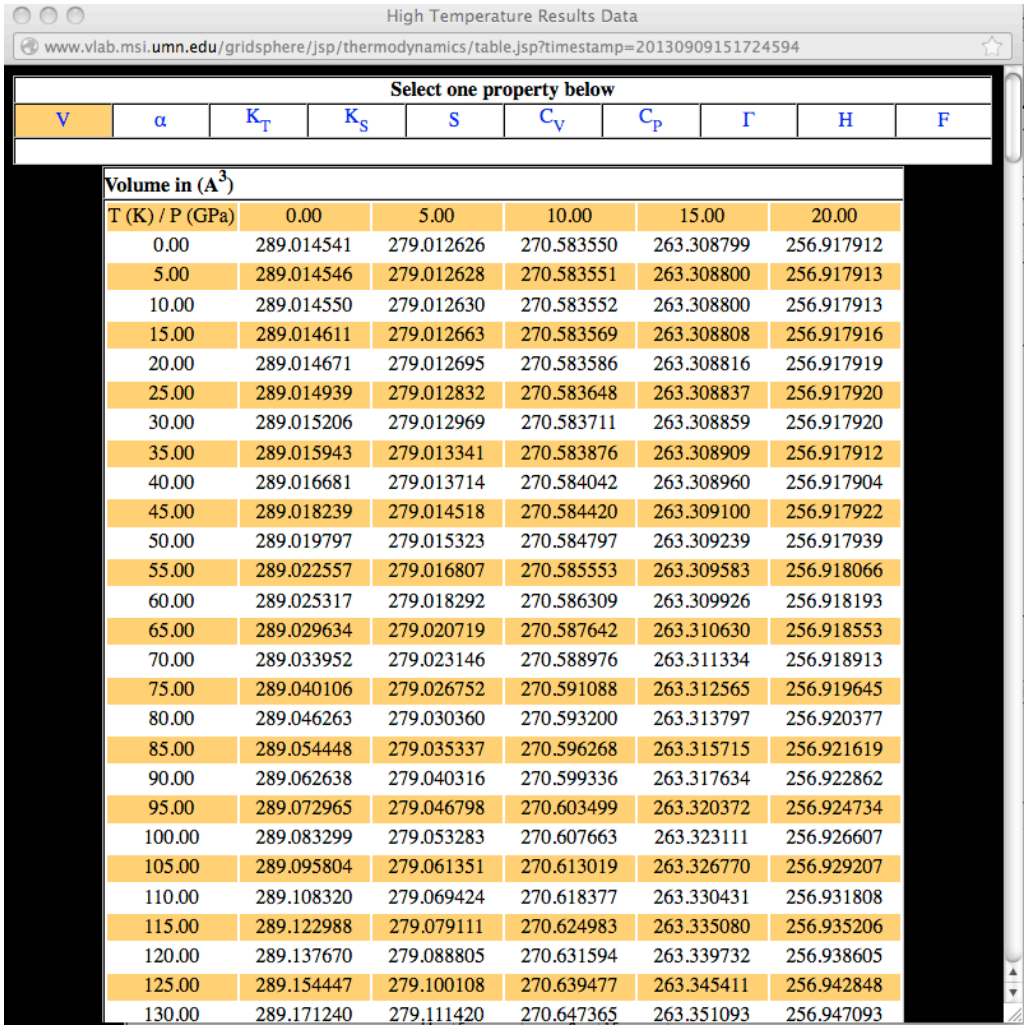


Figure 5. Typical output plot created in the Interactive Visualization resource displaying results of (LDA) calculations of thermodynamics properties for forsterite (or peridot).

Figure 6 shows a set of thermodynamics properties options for forsterite (LDA type calculation). The user can then analyze plots or data by selecting one of the thermodynamic property, e.g., Volume (V), Thermal Expansivity (a), Isothermal Bulk Modulus (KT), Adiabatic Bulk Modulus (KS), Entropy (S), Isocoric Specific Heat (CV), Thermal Grüneisen Parameter (g), Enthalpy (H), and Helmholtz Free Energy (F).



Select one property below

V | α | K_T | K_S | S | C_V | C_P | Γ | H | F

Volume in (A^3)

T (K) / P (GPa)	0.00	5.00	10.00	15.00	20.00
0.00	289.014541	279.012626	270.583550	263.308799	256.917912
5.00	289.014546	279.012628	270.583551	263.308800	256.917913
10.00	289.014550	279.012630	270.583552	263.308800	256.917913
15.00	289.014611	279.012663	270.583569	263.308808	256.917916
20.00	289.014671	279.012695	270.583586	263.308816	256.917919
25.00	289.014939	279.012832	270.583648	263.308837	256.917920
30.00	289.015206	279.012969	270.583711	263.308859	256.917920
35.00	289.015943	279.013341	270.583876	263.308909	256.917912
40.00	289.016681	279.013714	270.584042	263.308960	256.917904
45.00	289.018239	279.014518	270.584420	263.309100	256.917922
50.00	289.019797	279.015323	270.584797	263.309239	256.917939
55.00	289.022557	279.016807	270.585553	263.309583	256.918066
60.00	289.025317	279.018292	270.586309	263.309926	256.918193
65.00	289.029634	279.020719	270.587642	263.310630	256.918553
70.00	289.033952	279.023146	270.588976	263.311334	256.918913
75.00	289.040106	279.026752	270.591088	263.312565	256.919645
80.00	289.046263	279.030360	270.593200	263.313797	256.920377
85.00	289.054448	279.035337	270.596268	263.315715	256.921619
90.00	289.062638	279.040316	270.599336	263.317634	256.922862
95.00	289.072965	279.046798	270.603499	263.320372	256.924734
100.00	289.083299	279.053283	270.607663	263.323111	256.926607
105.00	289.095804	279.061351	270.613019	263.326770	256.929207
110.00	289.108320	279.069424	270.618377	263.330431	256.931808
115.00	289.122988	279.079111	270.624983	263.335080	256.935206
120.00	289.137670	279.088805	270.631594	263.339732	256.938605
125.00	289.154447	279.100108	270.639477	263.345411	256.942848
130.00	289.171240	279.111420	270.647365	263.351093	256.947093

Figure 6. Thermodynamics data for “V” (volume) displayed as an HTML table. Tables for other properties can be displayed by clicking on the links at the top (e.g., “S”, “F”, “H”, etc…).

2.3.3 Pressure Scale Calculators

Similar web pages were developed to enable public access to a pressure calibration according to our best equation of state of MgO – “MgO Pressure Scale Calculator (MPSC) (see Figure 7) - and of platinum, - “Platinum Pressure Scale Calculator” (PPSC) (see Figure 8).

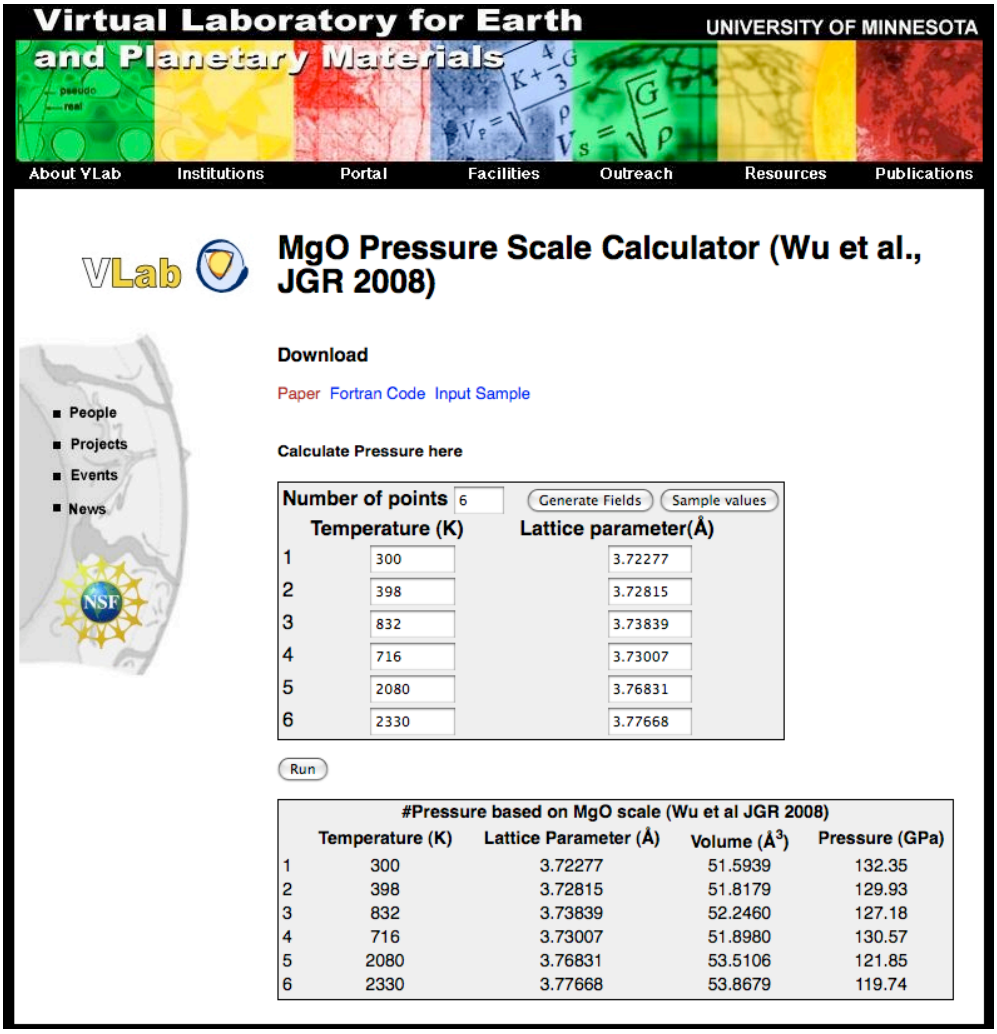


Figure 7. MgO Pressure Scale Calculator (MPSC) interactive user interface.

In general, users interested in these equations of state write their own programs based on less accurate data. In the ***VLab*** Pressure Scale Calculator website, the publication containing such information is available in the same page along with the source codes for these equations of state. Users can run these calculations online or download the source codes and input samples and run the program offline. The user input consists in defining the number volume/temperature (usually) experimental data points. The web page extracts the user input, prepares an input file for the pressure scale calculator software developed in the Fortran 95, runs the software in the background of the web container, and produces an output in HTML format. Figure 7 shows the MPSC web page containing the input list of Temperature (K) and Lattice parameter (Å) entered along with the output table containing two extra columns with Volume (A^3) and Pressure (GPa) results.

The MPSC application introduced a new thermal equation of state (EOS) of MgO, which appears to be predictive up to the multi megabar pressures and thousands of Kelvin range. It is obtained by combining first principles local density approximation (LDA) quasi-harmonic (QHA) calculations with experimental low-pressure data. This EOS agrees exceptionally well with shock compression data.

The PPSC application uses results of QHA calculations to obtain the thermal equation of state of platinum up to 550 GPa and 5000 K. The static lattice energy was computed by using the linearized augmented plane-wave method with local-density approximation LDA, Perdew-Burke- Ernzerhof (PBE), and the recently proposed Wu-Cohen (WC) functional.

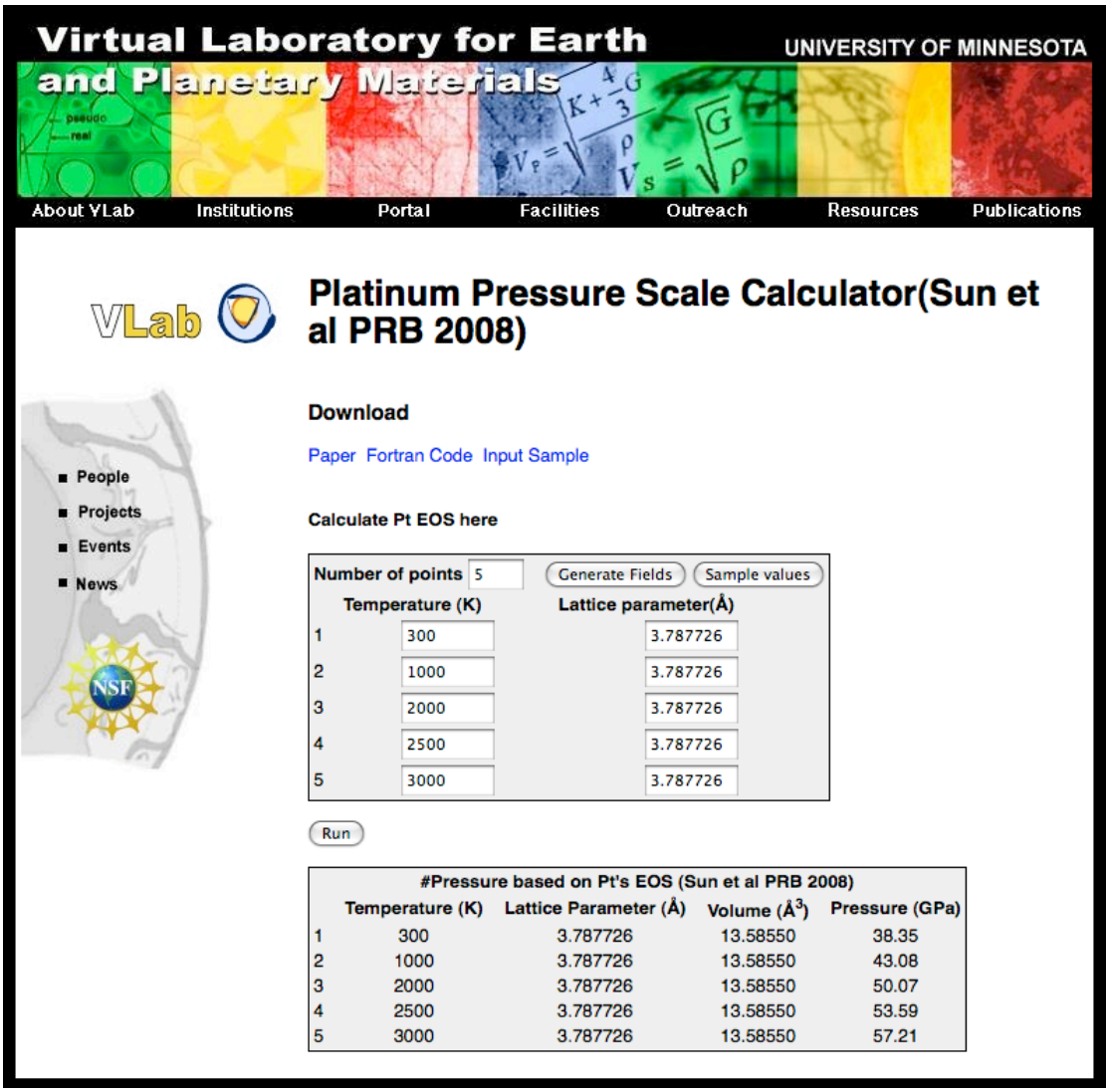


Figure 8. Platinum Pressure Scale Calculator (PPSC) interactive user interface.

2.4 Impact

VLab resources were developed simultaneously with all the other material discussed in this dissertation. Access to these resources was tracked using the Google Analytics Tool [17] from the moment the implementation was made available online. Several kinds of statistics, e.g. demographics, technology used, user behavior, traffic source, etc, can be obtained with this tool. Table 1 provides a summary of usage data collected for these ***VLab*** applications during the September 20th of 2009 to February 15, 2014. One key benefit of this tool is the

capability to perceive a resource utilization peak after a conference presentation or after a paper is published pointing to the ***VLab*** resources. The comparatively high average session time and page view of “Thermodynamics of mineral (ThoM)”, is most likely caused by the robust nature of the application and C code used to generate these properties and by the flexibility the user has to play with the data and re-plot in customized form. The high usage of this resource is also related with the number of materials for which thermodynamics properties can be calculated. In addition, the “***VLab*** website” item in Table 1 shows the highest number of visits because it is the main entrance to the ***VLab*** research and most of the access comes from internet search engines.

Chapter 3

3. *VLab* cyberinfrastructure for first principles calculations (*VLabWp*)

In this section we focus on the ***VLabWp*** architecture and software. The ***VLabWp*** consists of three main components: **i)** the Web Portal described in section 3.1, **ii)** the database schema and metadata explained in section 3.2, and **iii)** the Web Services described in section 3.3. A brief description of the overall code organization is presented in section 3.4. Figure 9 shows the connection between users, these three components, and the computing grids currently used by the ***VLabWp***.

Resource	Unique Visitors	Page view	Average session time
Database search	241	559	00:01:17
Thermodynamics of minerals (ToM)	712	8,333	00:09:31
Thermoelasticity of minerals	123	625	00:02:59
Thermodynamics of minerals at ultra-high pressure (ToM-UHP)	189	963	00:04:30
MgO pressure scale calculator	387	1,045	00:03:19
Platinum pressure scale calculator	207	359	00:01:25
VLab website	43,114	84,862	00:01:10

Table 1. Summary of access to ***VLab*** resources.

3.1 The Web Portal

The system architecture implemented in ***VLabWp*** uses a web portal as the primary connection to the cyberinfrastructure for first principles calculations. In

the portal, the user is able to prepare inputs on web pages with pre-defined fields, enter custom inputs, define properties and upload additional files for the calculation, e.g., pseudopotentials stored in a repository. The web portal is directly connected with the metadata database and has access to the web services. ***VLab*** web services are stateless services that carry out pre-defined tasks such as running heterogeneous software, e.g., shell scripts and programs developed in C language to create inputs, select HPC nodes to execute heavy processes, or just to submit tasks to a Portable Batch System - PBS [18].

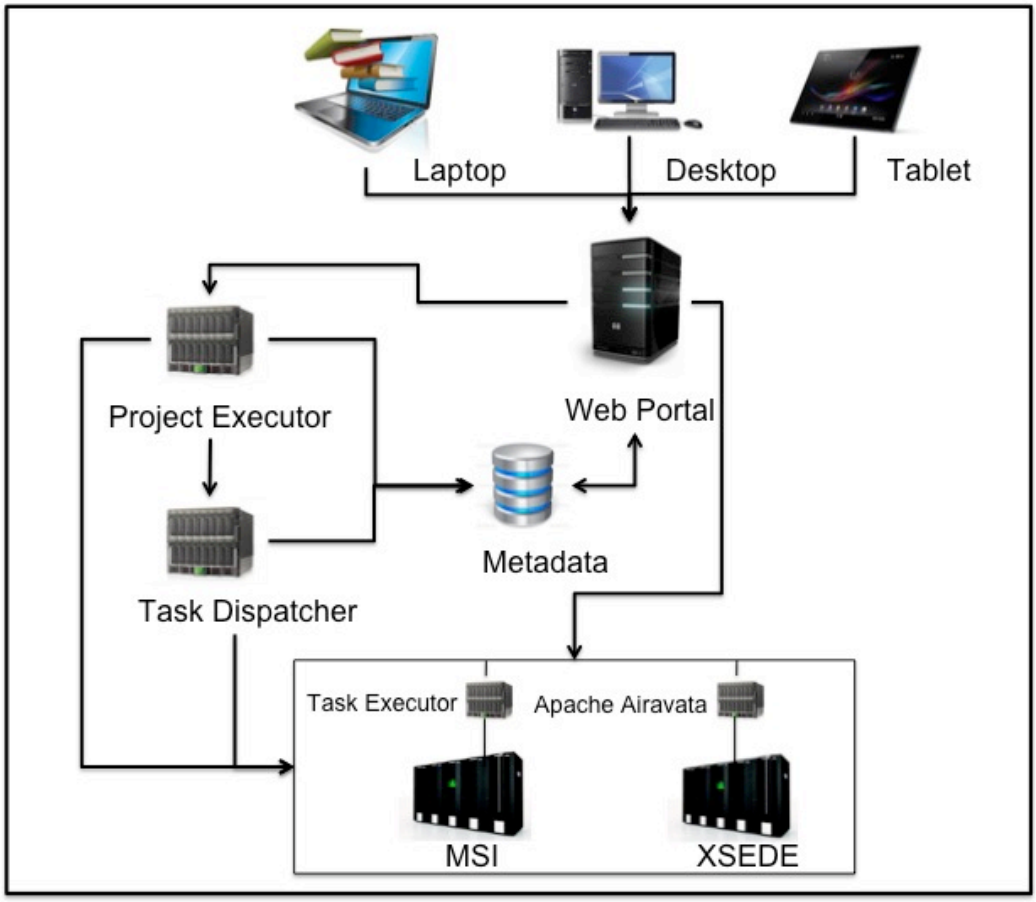


Figure 9. VLab architecture overview: Web Portal, Web Services, Database, and HPC systems.

After an initial input is finally prepared, the web portal receives a request from the user to submit a prepared project for execution. The web portal connects to the web service *Project Executor* to create a workflow and

associated tasks (see section 4 for more details on types of tasks that can be created). The *Project Executor* communicates with the web service *Task Dispatcher* to schedule workflow tasks based on HPC nodes available. The "*Dispatcher*" contains a list of servers pre-defined by the ***VLab*** administrator and interacts with HPC resources using two different approaches, depending on where tasks are executed. First at MSI [2], our local HPC resources, the *Dispatcher* interacts with the web service *Task Executor* installed directly in one HPC node. The *Task Executor* is responsible for a) submitting tasks created by the *Project Executor* to the PBS and b) for running these tasks using HPC parameters, e.g., MPI directives, number of cores per task, etc predefined by the user in the web portal. Second, in the XSEDE, the Extreme Science and Engineering Development Environment, the *Dispatcher* interacts with HPC resources using the Apache Airavata Application Programming Interface (API) [14]. More details about how this approach is used will be provided in section 5. This communication process between web portal, database, web services, and HPC systems is the base of the ***VLab*** system architecture as depicted in Figure 9.

There are several definitions of Web Portal in the Internet. According to one of the most popular portal framework makers, IBM, "a portal is a Web Site that provides users with a single point of access to Web-based resources by aggregating those resources in one place and by requiring that users log in only to the portal itself, and not to each portlet they use." [19]. The ***VLab*** Web Portal was developed using the Gridsphere portlet container [20] in the Apache Tomcat [21] web server. The Gridsphere Portal Framework [20] is an open source web-based framework installed in the web container Apache Tomcat in the ***VLab*** CIDFPC. The Gridsphere permits the creation of an interface for the user to log in with user name and password and access various portlets with different functionalities in one Window. The code behind the Gridphere portlet is compliant with JSR-168 specifications, making it a flexible framework to create code based on Java and non-based-portlet.

Main ***VLabWp*** user interfaces are designed to enable users to submit QE simulations to supercomputers. These job submissions are organized as user abstractions called *projects*. Users can create/delete projects, define input-specific parameters, control execution settings such as CPU, memory and wall times, search for similar existing projects stored in the database, attach auxiliary files such as pseudo-potentials [22] retrieved from a database or uploaded by the user, monitor state of runs, store relevant outputs on database, etc. Stored summaries of *project* outputs can be viewed through the ***VLab*** *Resources* web page on database of results as described in section 2.3.1, but only authenticated ***VLab*** users can manipulate them. All user interfaces were developed using Java Server Pages and are deployed as portlets.

The web portal is characterized by an interface combining the function to interact with first principles calculation numerical parameters and to access the main services. It allows access to metadata, visualization services, analysis tools, and interaction with web services *Project Executor* and *Task Dispatcher*. Figure 10 shows the Web Portal monitor page where users can see details of a project's phases and attributes such as start time, end time, servers where tasks are/were executed, tasks' and phases' current status, inspect outputs as they are produced, and stop or restart single tasks or phases of a project in execution.

3.1.1 Input Preparation

In ***VLabWp***, the user has the ability to create the input in custom forms for four different project types. They are described in section 4 of this dissertation. The input page prompts the user to enter data that are subsequently saved in to the metadata and used to create the workflow. First, a user must enter the project *details*. Project *details* are properties entered at the portal and later on inserted in the metadata for classification purpose. Figure 11 shows a portlet window with a set of parameters filled for a project type "Static Elastic Constant (Cij)" for the mineral wadsleyite. Those properties are transmitted to the web services when a user submits a project for execution. These properties are used

to define the initial project input type, HPC nodes, parameters for QE codes, and to be used on project searches in the database. A description of project details is given below:

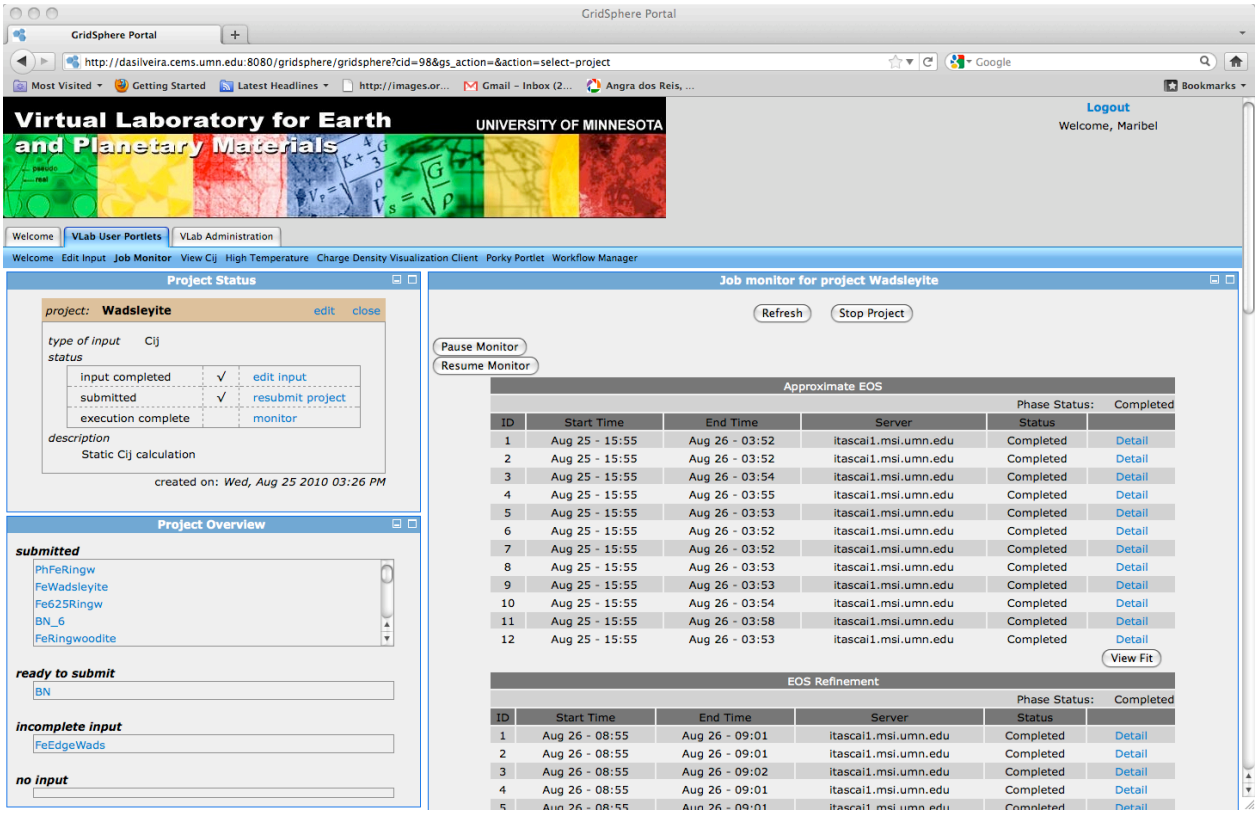


Figure 10. Web portlet accessing the database

Project Name: A user-level abstraction associated with an instance of a workflow. It is used to identify a particular project.

Project Type: It is a predefined type of workflow described in section 4. They are Single Calculation, Equation of State, Static Elastic Constant, Thermo EOS.

Chemical Formula: It is the chemical formula of the material to be simulated.

Structure: Crystal structure name of the material to be simulated (e.g., zincblende, rocksalt, etc). The user can select from a pre-existing list or create structure names if it is not available in the list.

Material: It is the name of the material, when it exists (e.g., stishovite). The user can select from a pre-existing list or create material names if it is not available in the list.

Exchange Correlation Functional: Specifies Exchange Correlation Functionals that can be used in the project (e.g., LDA). The user can select from a pre-existing list or create Exchange Correlation Functional names if it is not available in the list.

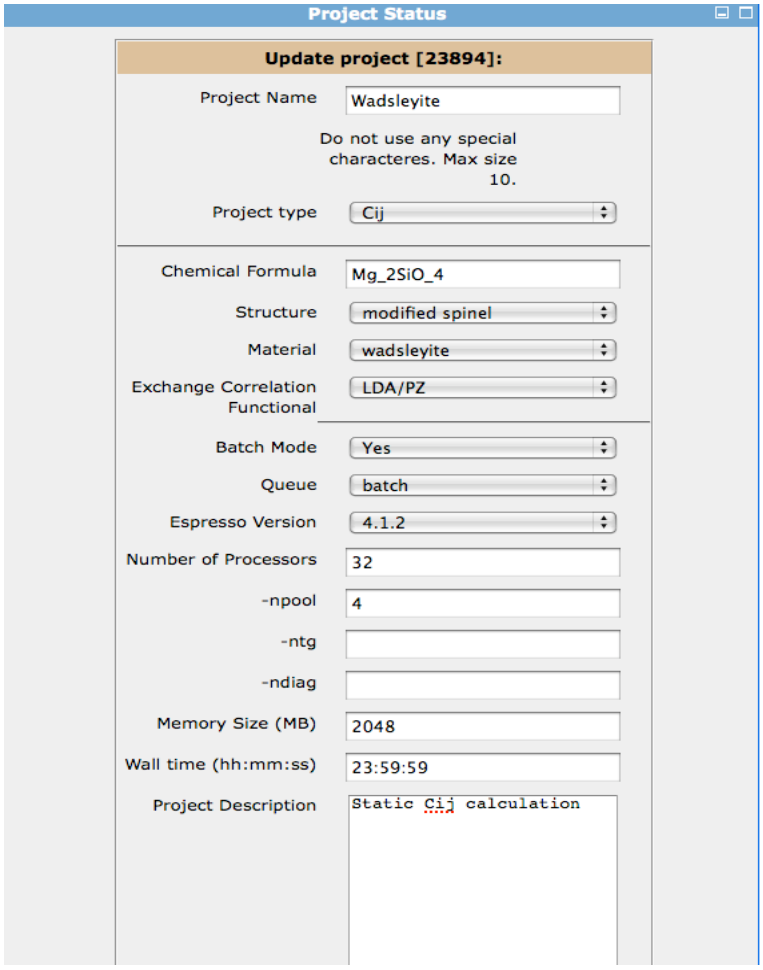


Figure 11. Project detail portlet defined by the user.

Batch Mode: It is a Boolean field specifying whether the project will be submitted to a queue system such as PBS or run interactively.

Queue: This field is used only in case the Batch Mode is set to true. A list of queues is based on the defined HPC node queues. The user has the option to created more queues if it is not available in the initial list.

Quantum ESPRESSO version: Specifies the QE version used in the project.

Number of Processors: It is a very special field because it defines the number of cores to be request in the HPC node. This number defines the amount of CPU requested by each QE task and is related with CPU time and Wall Time for each PBS request.

Npool, Ntg, and Ndiag: These are mpirun parameters used in the parallel execution of the QE code PWscf.

Memory Size: The amount of memory needed to run one QE task.

Wall Time: The total time for completing one phase of the workflow in the PBS.

Description: It is an open space for a project description text.

For a Static Elastic Constant project in the ***VLabWp***, the user must define the number of pressures and the pressures for which calculations will be performed, the number of atoms, number of atom types and other parameters specific to the crystal structure. Each User Input is saved on the database associate with each user account in the ***VLabWp***, and subsequently available for review and analysis. The user input is passed to the web services and, using the set of information entered, the web service *Project Executor* creates project specific workflow tasks. Figure 12 shows an example of a user input page. In this example, those parameters are associated with QE parameters for a “Static (Cij)” workflow.

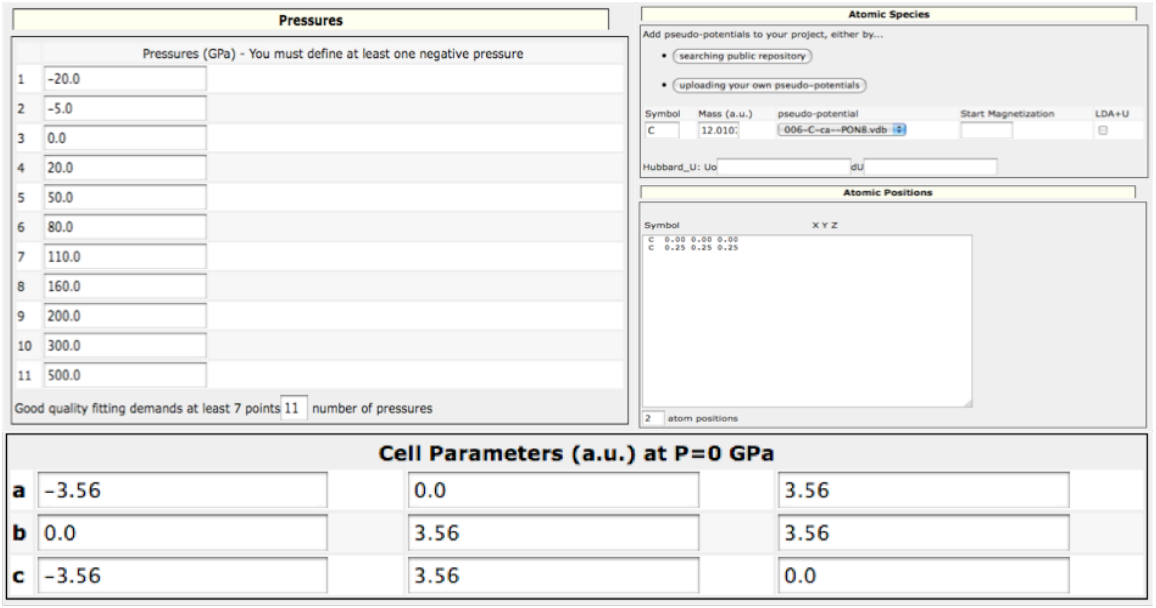


Figure 12. Input preparation for a Static Elastic Constant simulation.

3.1.2 Monitor Page/Task Detail

For each simulation performed in the ***VLabWp***, a monitoring process session starts in the web service *Project Executor* (see section 3.3.1 for more details). These sessions are started by the user after input preparation and project submission for execution. They are stateless, i.e., the portal does not need to wait for a response. The monitor interface portlet allows the user to check the progress of every phase and individual tasks submitted to the HPC. Figure 10 shows a monitor interface for a “Cij” project for the wadsleyite mineral. The monitor interface has web links to check details of every task and check the post-processing step called “View Fit”, which fits an equation of state to the data produced after the completion of each phase (see section 4. for workflow detail). The task detail interface presents data such as the task ID, phase name, server host name where the task is running, full path, process ID, QE Input and Output. In addition, the user has the ability to inspect inputs and outputs, interrupt and/or resubmit a single task, phase, or the whole project. Using the web portal the user can also plot data collected from outputs. Figure 13 shows a sample of a task

detail on the monitor interface where a task can be edited and resubmitted in case of failure.

Figure 13. Task details as viewed/entered in the ***VLab*** Portal.

3.1.3 Data Plotting

Some workflows in the ***VLabWp*** are composed by phases. After the completion of a workflow phase or whole workflow, certain parameters are immediately available for review and analysis in the Web Portal. In these pages, results are presented in 3 different formats: as an HTML table with all values printed on screen, as a JPEG file in which the data has been dynamically plotted using GNU Plot [23], with options to changing labels, define abscissa and ordinate boundaries, font size and format, and lines thickness, or as a download, from which the data can be plotted with any program of user's choice. For instance, the elastic type calculations, which are post-processing step of a static

elastic constant workflow, can be displayed in table format, as shown in Figure 14.

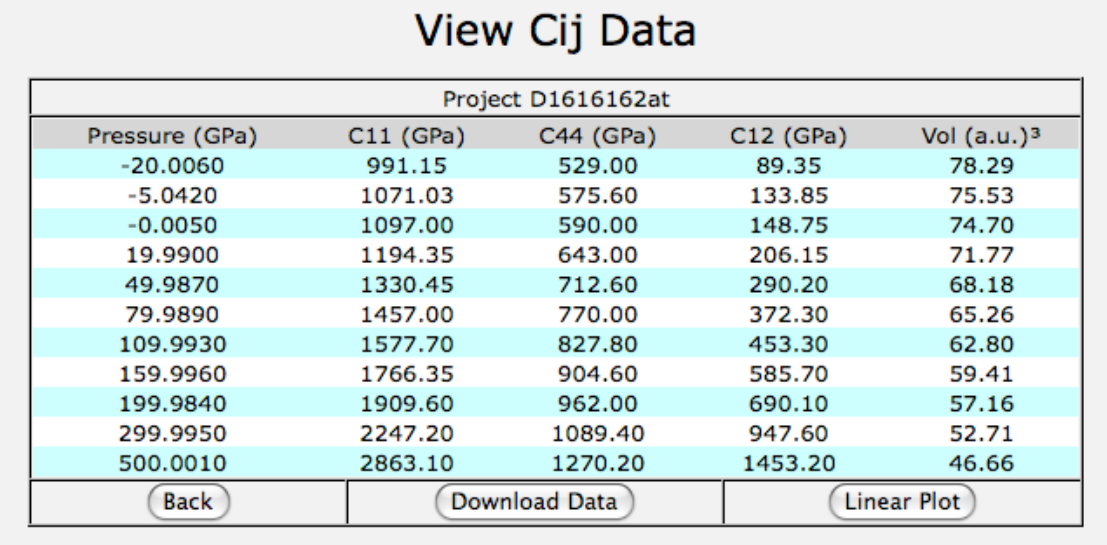


Figure 14. Final results for calculated diamond's elastic coefficients, C_{ij}.

There are also options to download the data and to display it as a linear plot on the web portal shown on Figure 15. The plot available in the ***VLab*** SG uses the same pattern and technique described in section 2.2.2 for the interactive visualization.

3.2 Metadata

Definition of metadata is crucial in designing a system required to support concurrent execution in a heterogeneous distributed environment. ***VLabWp*** must keep metadata to track the execution of tasks and the locations of a plethora of input and output data files that lie scattered throughout HPC nodes. These tracks are stored in a database structure in way to allow the user to retrieve status of each task through the portal, save properties for each task, and allow collaborative work among users avoiding race conditions.

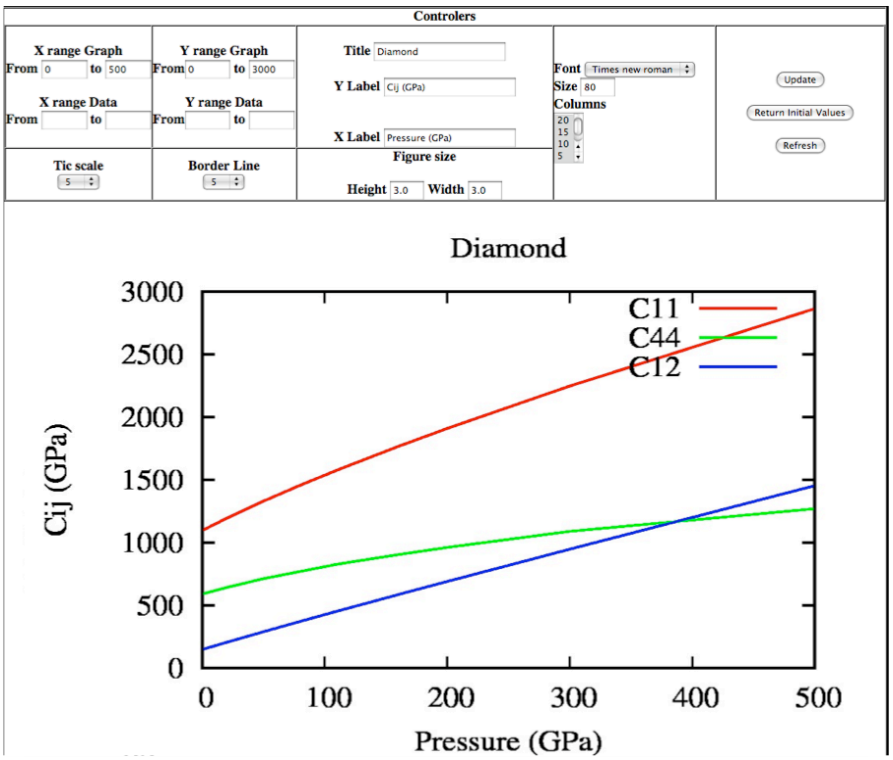


Figure 15. Line Plot for diamond's calculated C_{ij}.

3.2.1 Database Schema

Based on a requirement analysis for ***VLabWp***, three objectives for the database structure can be established: **1)** it must have a user friendly interface where all project information is entered (see Figure 11); **2)** it must be designed to be capable of storing in an organized manner all results from successful projects; **3)** database searches must be user-friendly (see Figure 2). A project's "data", i.e., results plus metadata, are stored in 8 tables and identified by a project "id". The main table is called Project Descriptor. The attributes stored in this table include: chemical formula, title, crystal structure, pressure, and pseudopotentials. Other tables store data with author information, material information, project type, project phase, exchange-correlation functional, and structure. The last table called "execution", stores execution packages with all inputs and outputs plus project execution dates. This table has pointers to directories where inputs and outputs are located.

With the exception of Project Descriptor table, all the other tables use the properties of decomposition 2 Normal Form (NF) to save information related to author names, structure, material, exchange correlation functional, and project phases. On the decomposition tables there are the primary key that identifies many items like material and structure. But as criteria to search for a calculation, some hash indexes were created over the author's name (first or last), material name, structure name, and exchange correlation functional name. Therefore, using this index the search returns faster results. On the *Project Descriptor* table it receives only the foreign keys from the decomposition tables. Figure 16 illustrates how these tables can be associated. When the user defines a project through the Web Portal s/he specifies the attributes to be used in the user-friendly search interface, e.g., project title, author name, material, structure, exchange correlation functional, and year of project execution (see Figure 11).

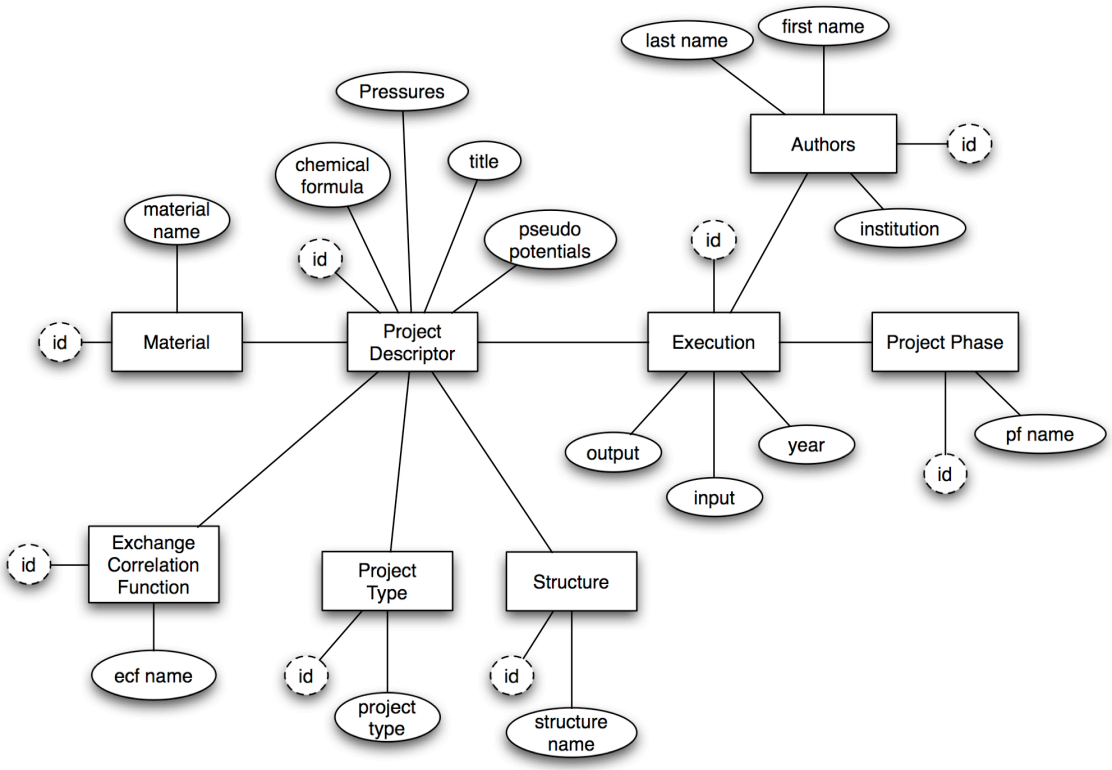


Figure 16. Relationship between database tables.

3.3 Web Services

The utilization of web services is made via web portal by invocation of a class stub on the client side of the web service. Web services submit projects for execution, get input and output data to read and/or write, change compute node parameters, use analyzing tools, etc. When a web service is called for, a set of user information is passed to the Service Oriented Architecture (SOA) protocol to establish the connection and make the appropriate request using the receipt metaphor [24]. The receipt metaphor is a set of Java classes that contain a data structure to make requests between web services. Those receipt structures contain necessary information for the request, such as task id, file path, user name, server IP address, etc (See section 4.2.2).

The web service utilized in the ***VLab*** CIDFPC are built over the framework Apache Axis 2.0 [25]. The Apache Axis 2.0 is a web service core engine that is incorporated into the web container Apache Tomcat that handles multiple remote requests from users using the Simple Object Access protocol (SOAP) to perform predefined services. The Axis 2.0 permits connection from web services clients using the Web Service Description Language (WSDL). WSDL is equivalent to a framework to establish connection to web services from a web portal or from another web service. ***VLab*** web services using Axis 2.0 are written in Java language. There are three main web services: *Project Executor*, *Task Dispatcher*, and *Task Executor*. They are described next.

3.3.1 Project executor

Project Executor is the main web service with the function of creating workflows based on parameters entered in the web portal by the user. The *Project Executor* handles multiple requests from the User in the ***VLab*** portal such as splitting the calculation into tasks to be distributed across a set of previously defined and user dependent HPC nodes, and saving the project execution status of all projects on a database using the receipt metaphor better described in

section 4.2.2. A detailed overview of workflows supported by ***VLab*** was given by da Silva et al. [26]. The *Project Executor* is a stateless service that monitors the progress of a workflow in one single session inside the Apache Tomcat. The Project Executor also makes calls to shell scripts to parse input and output files, to create new input files, to run C programs to calculate predefined tasks, and Fortran codes to perform predefined tasks on outputs generated in HPC nodes. The *Project Executor* is installed in a server that stores considerable large amounts of data, simply because all the raw information generated during the workflow execution is stored in a disk and is accessible through the Web Portal by the user. Thus the user can also access input and output files created by the workflow using the ***VLab*** web portal.

3.3.2 Task Dispatcher

The main function of the *Task Dispatcher* is to inform the *Project Executor*, which compute node to use when a user wants to have a project executed on a HPC resources available. The web service *Task Dispatcher* receives information about tasks to be launched for execution across compute nodes. The *Dispatcher* has an internal scheduler with basic information about queues on various compute nodes and uses this data to selectively launch different numbers of tasks in different compute nodes. Section 5 Is dedicated to more details about the scheduling process. Figure 9 shows the connection between the web portal to the web service *Project Executor* and from the *Project Executor* to the *Task Dispatcher*. The web service communication between the *Project Executor* and the *Task Dispatcher* is made through the Axis 2 framework.

3.3.3 Task Executor

The web service *Task Executor* is proposed to provide the large-scale computation necessary to study materials properties at high pressures and temperatures. In the ***VLabWp***, this kind of computation is achieved by using the web service Task Executor at MSI, and Apache Airavata API at XSEDE

resources. The web service *Task Executor* is installed only on compute nodes at MSI. It is the software that receives “execution packages” and submits tasks to the local HPC queue. Execution package in this case stands for one input and additional files required to execute one of the QE codes. The *Executor* also sends run-time information and outputs to other web services, interrupts task executions, receives requests from a ***VLab*** user through the web portal to modify a task input, and resubmits tasks. ***VLab***’s web service infrastructure is currently used to manage jobs running in batch queues at MSI and XSEDE resources. The description of how the ***VLabWp*** interacts with XSEDE is described un detail in section 5.

To distribute tasks across one or more HPC nodes at MSI, a simple job scheduler was developed to leverage the maximum usage of resources among ***VLab*** users. The scheduler lines up all ***VLab*** tasks submitted by all users logged in the portal into an internal queue. Currently, ***VLab*** tasks submission to supercomputers make use of an extra level of parallelization developed to leverage the use of computer resources among ***VLab*** users. It makes use of bag-of-tasks (BoT) to allocate simultaneously computer resources with maximum flexibility for as many ***VLab*** projects as possible. Tasks are submitted to MSI queues as a single job in a BoT container, therefore the number of cores required to run BoT applications scales linearly with the number of tasks in the BoT.

The computational time required running an entire phase in a BoT scales inversely with the number of cores allocated. An entire phase running in a BoT is an embarrassingly parallel calculation. On MSI servers, primarily Calhoun and Itasca, all processors required to execute M tasks using K cores each, i.e., M*K= N cores, must be allocated before a BoT container job starts running, with no more than one task running in a single core, as illustrated in Figure 17. The submission of one BoT application is more convenient than submission of 10^2 to 10^3 mid-size jobs to queues in tera- and peta-scale systems. Besides, the size and number of BoTs can be tuned to the specific application (via mpirun

parameters) and to the size and number of compute nodes available. The BoT approach described above was initially designed to simply fit an entire workflow phase, which usually includes 10-100 decoupled tasks. More advanced scheduling strategies were developed after a beta version of the scheduler was tested.

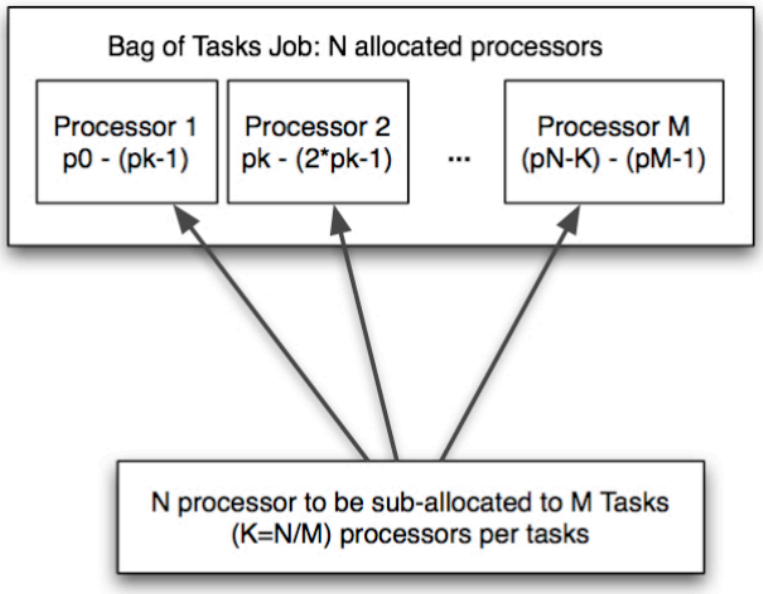


Figure 17. Representation of a BoT container job. It is composed of M concurrent tasks, each one allocating K processors. The container must allocate N=M*K processors.

3.3.4 Code Organization

It is important to describe how the ***VLab*** SG code took shape during development. The development process was user driven, but it combined the waterfall and agile approaches.

For the waterfall method, the requirements were established and implemented by interacting with and with help of scientist used to performing thermodynamics and thermoelasticity calculations manually – students and post-docs in the Wentzcovtich group. A highlight feature well understood at this point

was the nature of the files that required user intervention before being sent for execution in an HPC resource. Modification and creation of numerous files was demanding and error prone manual work. From the start, Unix scripts were created to handle files preparation automatically as part of the calculation procedure in progress. A set of Java Bean classes was created to wrap Unix scripts and deploy them as Web Services.

The agile development approach was employed next, after more extensive understanding of the entire sequence of operations was clarified, and separate scripts were in place and fully functional to execute different parts of a workflow. It was then possible to build a common collection of libraries used throughout the entire ***VLab*** SG, e.g., ***VLab*** DB, ***VLab*** base, Service Stubs, and Input Bean. A schematic overview of the ***VLab*** code organization is shown in Figure 18.

The pseudopotential repository: this is a Java code defined as a library and is mainly used by the Pseudopotential web pages in the portal to search for pseudopotentials and insert them in execution packages along with other calculation inputs. This repository is integrated into common libraries because when a pseudopotential is inserted into a project, it must remain associated with that specific project and saved into a unique session of the ***VLab*** Portal.

*The **VLab** Common*: this is also a Java code defined as a library of tasks that manipulate data structures associated with projects. Project data structures consist of metadata plus data files of simulations parameters entered by the user in the project input page. Tasks in the ***Vlab*** Common library are invoked for example by the project “monitoring” web page, the “results analysis” web page, etc. It is the most commonly assessed library because it is literally used by all the other modules of the ***VLab*** CIDFPC. It contains the receipt metaphor and its components.

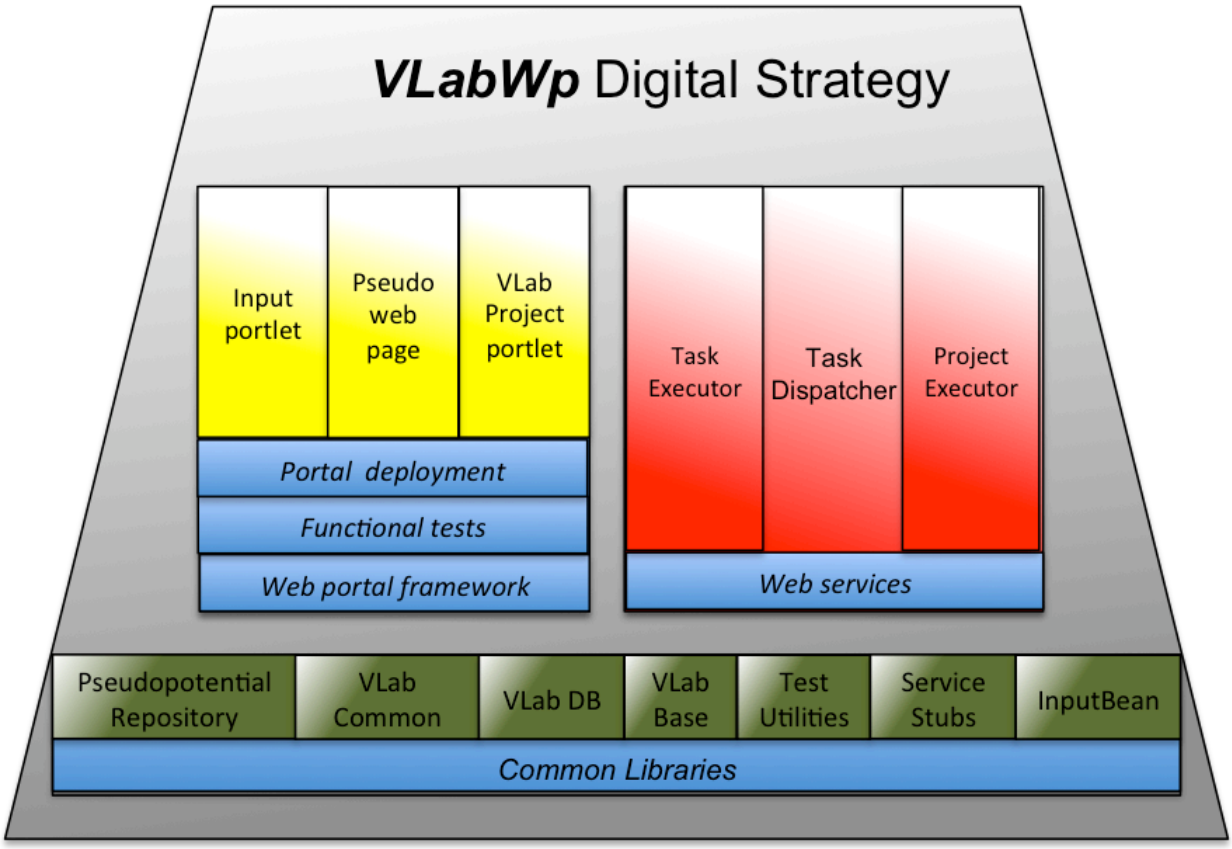


Figure 18. VLabWp code hierarchy

*The **VLab** DB*: is a Java library of communication methods with the database. ***VLab*** DB uses the Hibernate technology to handle multiple queries to the database by multiple users using one single authentication connection. This library is also used by the web services.

*The **VLab** Base*: is a library of functions invoked through the web portal user interface. Such functions are used in various pages of the portal, e.g., project management page, project monitoring page, data post-processing page, etc. This library contains functions used by the web portal framework (Gridsphere) to create portlets with different functionalities.

InputBean: is a Java library of functions that manipulate project input data entered in the web portal. Input data include project parameters, e.g., strains, temperatures, etc., and QE parameters used throughout the simulation, e.g.,

pressures, number of atoms, atomic positions, execution parameters, etc. This library has one Java class corresponding to each project type supported in the CIDFPC. This library is also used by the web services Project Executor, Task Dispatcher, and Task Executor to create tasks, create BoTs, run different versions of the QE, etc.

Service stubs: is a java library of communication protocols between callers and web services. Each web service has a set of protocols of its own. They are used for communication between web services and between web portal and web services.

Test Utilities: is a set of Java unit test classes created to tryout functionalities of various parts of the ***VLabWp***. These utilities are essentially mock, e.g., mock project creation, mock input preparation, mock QE call, mock web service calls, etc.

Web portal framework, functional tests, portal deployment and web services shown in Figure 18 are entities that encapsulate software frameworks used in different part. The Web service framework is used for the three main web services: *Task Executor*, *Task dispatcher*, and *Project Executor*. The web portal framework, functional tests, and portal deployment are used in the input portlet, pseudopotential web page, and in the ***VLab*** project portlet.

This digital strategy was defined to have web Services and web portal using one single communication protocol for use in workflow development. Thus, if one task running on an HPC resource has the status updated by one web service, this data is automatically saved on the metadata using the receipt metaphor. This same metaphor must be understood by the web portal interface and by the web service that modifies the receipt. Both web portal and web services use the ***VLab*** common libraries. Another example is the Input Portlet code, set in yellow on Figure 18. It uses the same Service Stubs for communication with the Web Services. For this particular case, a user logged on the web portal is capable to check the post-processing of one simulation, which

is located on the web service *Project Executor*. Web services and Input Portlet use the Service Stubs.

The ***VLabWp*** software architecture uses three main entities to compile, store, and manipulate data. They are the SourceForge repository [27], Maven building manager [28], and Castor Java library [29]. The SourceForge repository was created to incubate the whole ***VLabWp*** code, even though the code is not public. This repository allows maintenance a tracking of code modification indicating progress achieved. The software utilized to compile, update class dependencies, perform Unit testing, and create web archive files for the web services and web portal is the Maven building manager. The Castor java library is used to convert a Java Class to XML format. It permits reading an XML file and conversion to a Class and vice-versa. By using this feature ***VLabWp*** is able to save the progress of a workflow assisted by a Java Class.

Chapter 4

4. Scientific Workflows

The definition of a scientific workflow is: a sequence of automated computational action steps made by scripts that call input data and programs and produce outputs, including analysis of results and data visualization [30]. Workflows can encapsulate large sequences of procedures that can be executed repetitively and easily. Scientific workflows can also be implemented using numerical computing software environment such as Python or Matlab, or script languages such as Python, Perl, and Unix shell scripting. Today, community-specific or generic e-infrastructure such as the European Grid Infrastructure (EGI) [31] or XSEDE encourage researches to utilize scientific workflow framework systems such as Kepler [12], Pegasus [11], Apache Airavata [14], Galaxy [32], and ASKALON [32]. The goal in using scientific workflow systems is to have a common set of easy to use computational tools among researchers interested in specific complex procedures expressing a well-defined sequence of calculations and data manipulations.

4.1 Scientific Workflows in *VLab*

Although the development of scientific workflows using workflow software frameworks is becoming popular, ***VLab*** workflows were developed using its own proprietary ideas and algorithms. These algorithms were created before workflow framework systems were released. Algorithms used in ***VLab*** Workflows have similarities with those of other framework, systems. However, they are more specific.

Workflow executions performed in ***VLab***, such as elasticity or thermodynamics properties, use modules of the Quantum ESPRESSO (QE) software package for first principles calculations, e.g., the Plane-Wave Self-Consistent Field (PWscf) or Phonon (Ph) modules. Only one workflow, *Single*

Calculation project type (e.g., for structure optimization), is performed with one single input for PWscf. All the remaining projects might require hundreds to thousands of inputs addressing, in addition to structure optimization, other types of calculations using other modules of the QE. ***VLab***'s currently implemented workflows can do:

1. Single calculation with the QE.
2. Static (athermal) equation of state (EOS) in two phases (*Approximate EOS* and *EOS Refinement*).
3. Static elasticity (C_{ij}).
4. Thermal (high temperature) equation of state EOS (*HighPT EOS*). This workflow can also computer thermodynamics properties.
5. High temperature elasticity (*HighPT* C_{ij}).

When the user defines a project on the Web Portal, s/he must choose from one of the above five options. These project types are better described in subsequent sections 4.1.1, 4.1.2, 4.1.3, 4.1.4, and 4.1.5. Projects are defined, submitted, and monitored from the ***VLab*** Portal. Within each project type there are predefined complex workflows, which read/manipulate data, create tasks from defined parameters entered by the user in the ***VLab*** Portal, execute tasks on remote computing nodes, parse outputs from previous calculations for post-processing or to create new inputs for new sets of tasks to be executed in subsequent steps. These project workflows are executed by scripts using the bag of tasks (BoT) approach on distributed servers. Section 5 gives a more detailed description of how workflows are processed on the server side by the task scheduler. Each workflow preparation requires a user to connect to the ***VLab*** Portal edit the input parameters for the first-principles calculation and submit the project for execution. A predefine workflow creates tasks, sends one BoT for each phase to avoid waiting on the queue, monitors the status of each task and captures results.

Figure 19 indicates the basic workflow structure implemented. It indicates the relationship between different kinds of projects and their final results until the analysis of data. The user has the ability to see intermediate inputs and outputs, interrupt a single task under execution, interrupt the whole project, edit one input and resubmit a single task, or resubmit the whole project. Figure 13 shows the option to resubmit, and edit an input task. The user has also the ability to generate plots of results through the Web Portal.

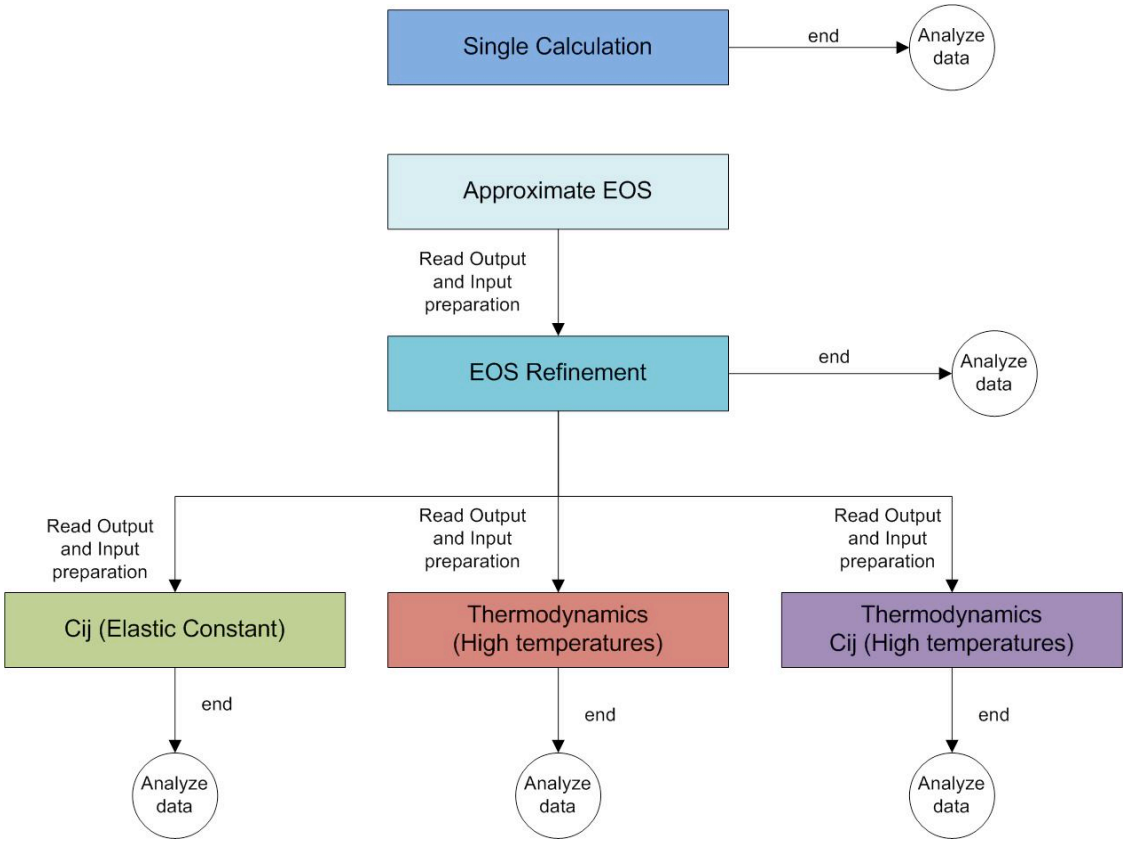


Figure 19. Single calculations are used to examine a single point in the phase diagraph of the modeled material. EOS calculations are collections of single calculations at many different points in the pressure-temperature phase diagram.

Besides the use of PWscf, the only QE code used to obtain static properties, the PHonon code is also required to calculate phonon frequencies for

subsequent calculation of high temperature properties. For example, a calculation of thermodynamics properties – *High PT EOS* type project - involves the following sequence:

1. *Approximated EOS*: atomic degrees of freedom are approximately estimated at each pressure P_i (PWscf of QE is executed). This step optimally decouples structural refinements at different pressures.

2. *EOS refinement*: cell and atomic degrees of freedom are fully optimized at desired P_i (PWscf of QE is executed).

3. Vibrational density of states calculation: dynamical matrices and phonon frequencies are calculated for several **q** vectors of the Brillouin Zone using results previously obtained at each P_i by the PHonon code of the QE.

4. Post-processing of phonon results to calculate interatomic force constants and phonon frequencies in a much sparser grid of **q** grid.

5. Computation of thermodynamics properties in a pre-defined range of pressures and temperatures: post-processing using a thermodynamics code, thermo.f or thermo.c.

The typical number of pressures and phonons is approximately ten each. Computations in unit cells with approximately twenty to hundred atoms requires up to few CPU hours to execute steps 1) and 2) per pressure Pi. The phonon calculation in step 3) is highly demanding and the CPU time can increase by one or two orders of magnitude. It can take up to 48 hours to calculate six phonons simultaneously in one BoT. For example (see Figure 20), for the mineral olivine with a unit cell containing 28 atoms, sampling 2x2x2 **k**-point mesh (needed for electronic bands) and 2x2x2 **q**-points mesh (needed for vibrational frequencies) in the irreducible Brillouin zone, the number of pressures sampled can be approximately eight. Without ***VLab,*** this project requires someone to prepare, submit, and monitor 84 tasks. In general, each task should run on 32 or on 32*N cores, where N = 2, 4, or 8 (number of **k**-points), i.e., from 32 to 256 cores,

depending on system size and core availability.

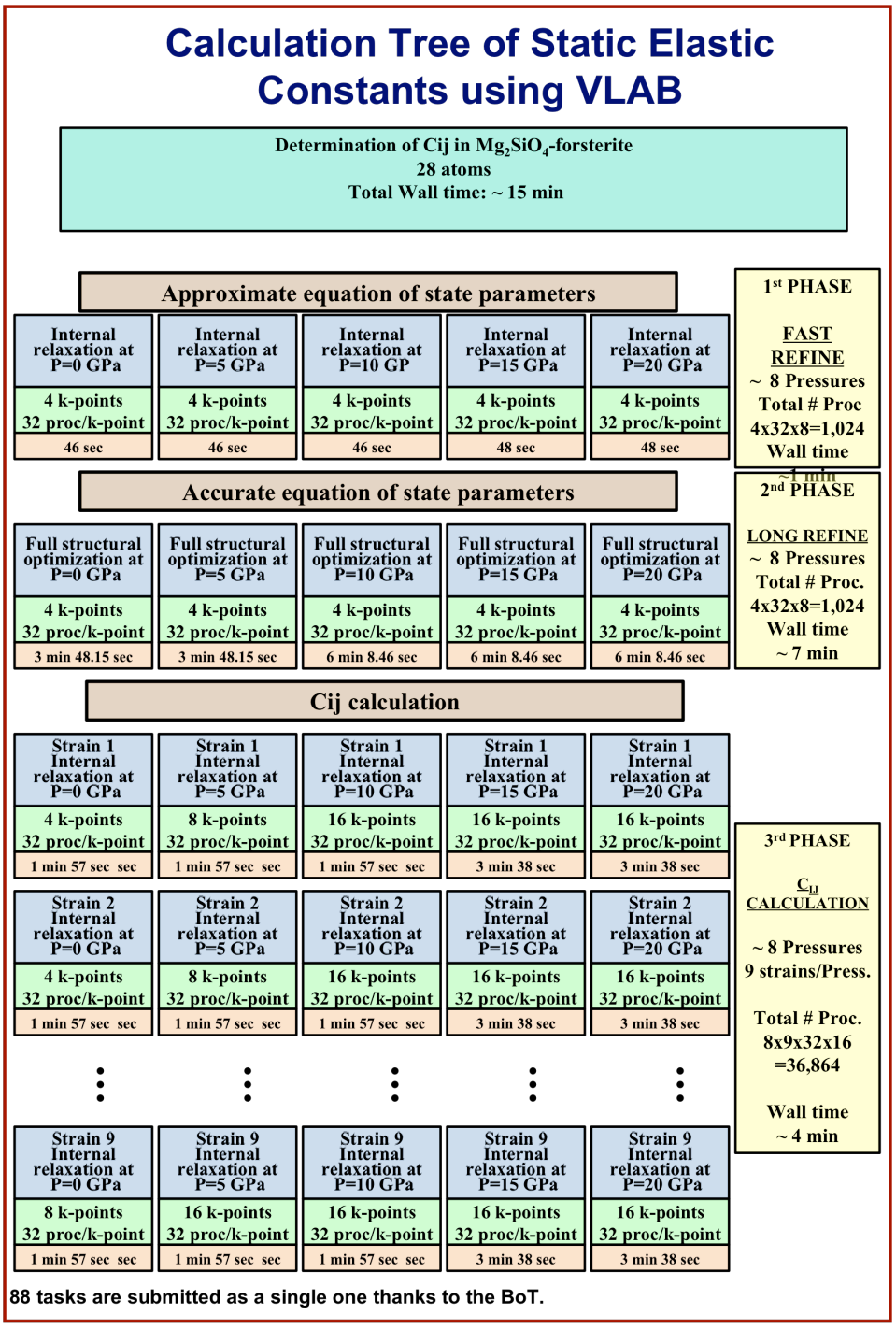


Figure 20. Detailed overview of a static elastic constant calculation along with tasks properties in ***VLabWp***.

For management and load balancing sake, these jobs would run (usually) in one MPI system, whose queuing policies restrict the number of tasks running or waiting in queues simultaneously. In ***VLab***, users only need to provide some initial key information of the target system, i.e., crystal structure and electronic structure information, **q**-points mesh and parallelization parameters (mpirun parameters).

Preparation of runs for phase 1) with 8+4 tasks (+4 to better constrain the EOS in the extremities of the pressure interval) involves the use of pre-processing Java and C or Fortran codes in a local server. After completing step 1), 8 input files are generated for step 2) in the local server using information from previous step's output files. As soon as the EOS Refinement is finished, 64 new input files are prepared for phonon calculations used in step 3) and 4). Each phase in this workflow is executed on multiple nodes requesting 32, or N*32 CPU/task (N defined above) by submission of bag of tasks (BOT) applications using PBS scripts into high performance computing nodes. The advantage of ***VLabWp*** is that each phase, with all its tasks, can be submitted and queued as a single job to an HPC node, or let the scheduler decide and monitor all by Web Services with a visual interface to display input and output created without having to connect to the HPC nodes in use. Therefore the time to complete the whole phase can be approximately the time to complete one individual task in that phase. Currently implemented workflows in ***VLab*** can perform 4 different types of computations as explained in the following sections. Figure 20 describes a Static Elastic Constant simulation performed in ***VLabWp*** for a material called forsterite with 28 atoms per unit cell. The three phases are clearly specified: Approximate EOS, Accurate EOS, and C_{ij}. In each phase, each task has its own pressure in GPa and the wall time to calculate it. The C_{ij} phase additionally provides a strain for each pressure.

Results produced by execution of the C_{ij} workflow in ***VLabWp*** agree completely with results produced manually [34]. The advantage of executing the "project" in ***VLabWp*** is enormous: input preparation is very easy, it gives better

execution response time, all strains required to compute C_{ij} for all crystal symmetries are internally defined, all workflow steps and intermediate results are saved in databases, and results readily displayed in numerical and graphic form in the web Portal. Figure 21 shows a typical final result.

C_{ij} Results in Mg_2SiO_4-Forsterite

P	0	5	10	15	20
C11	332.47	365.66	397.14	427.4	457.08
C22	207.63	232.69	255.76	277.1	297.13
C33	243.03	268.53	291.59	313.2	334.17
C44	70.88	78.36	84.32	89.25	93.63
C55	83.26	89.21	93.76	97.28	100.14
C66	85.69	94.03	100.65	106	110.65
C12	78.76	97.9	116.17	133.9	151.31
C13	79.55	98.35	116.91	135.2	153.37
C23	76.46	94.63	113.91	132.7	149.51

P and Cij are in GPa.

Figure 21. Final result of a C_{ij} calculation performed in ***VLab Wp***.

4.1.1 Single Calculation

In a Single Calculation, the user defines an initial input through the ***VLab*** Web Portal as described in section 3.3.1. This workflow uses PWscf (pw.x) for first principle electronic structure calculations for a single atomic configuration. As in any other service, the user submits the task to one HPC node using the distributed web services, and monitors its execution via Web Portal. However, this kind project type also requires the user to set memory size, number of cores, QE version, queue name, and the project properties e.g. material, chemical

formula, and etc. This calculation has only one phase, which indicates as the task complete the whole simulation is done.

4.1.2 Equation of State

Figure 22 represents an input web page, where the user enters 8 pressure points, atomic species, and atomic positions. The interface on the Web Portal is very intuitive to the user.

The block diagram of the *Static Equation of State* (*Static EOS*) type project is shown in Figure 23. This type of project is prepared in two phases: “Approximate EOS”, where a crude guess for the Vinet equation of state for a rough estimation of the cell volume at each pressure is performed, and “EOS Refinement”, where an accurate determination of equilibrium volumes at various pressures is performed to allow for fast convergence in a given uniform or non-uniform pressure grid, P_i, for $i=1,n$.

For this project type, the user is required to specify a minimum set of eight pressures to meet the minimum requirement for a good EOS fitting. The phase “Approximate EOS” is only executed as part of requirement to run “EOS Refinement”. In this sense, the output created by the phase “Approximate EOS” is used to create the input for the subsequence phase “EOS Refinement”. On this project phase it is possible to view a fitting of the Vinet EOS for “Approximate EOS” and “EOS Refinement”.

The EOS workflow performs structural optimization for a series of representative pressures and fits a finite strain expansion to the total energy versus volume relation. The EOS workflow itself involves small steps, and data manipulation. Figure 23 illustrates the series of steps to complete an EOS workflow, which can be used essentially for calculations on any material possible with the QE.

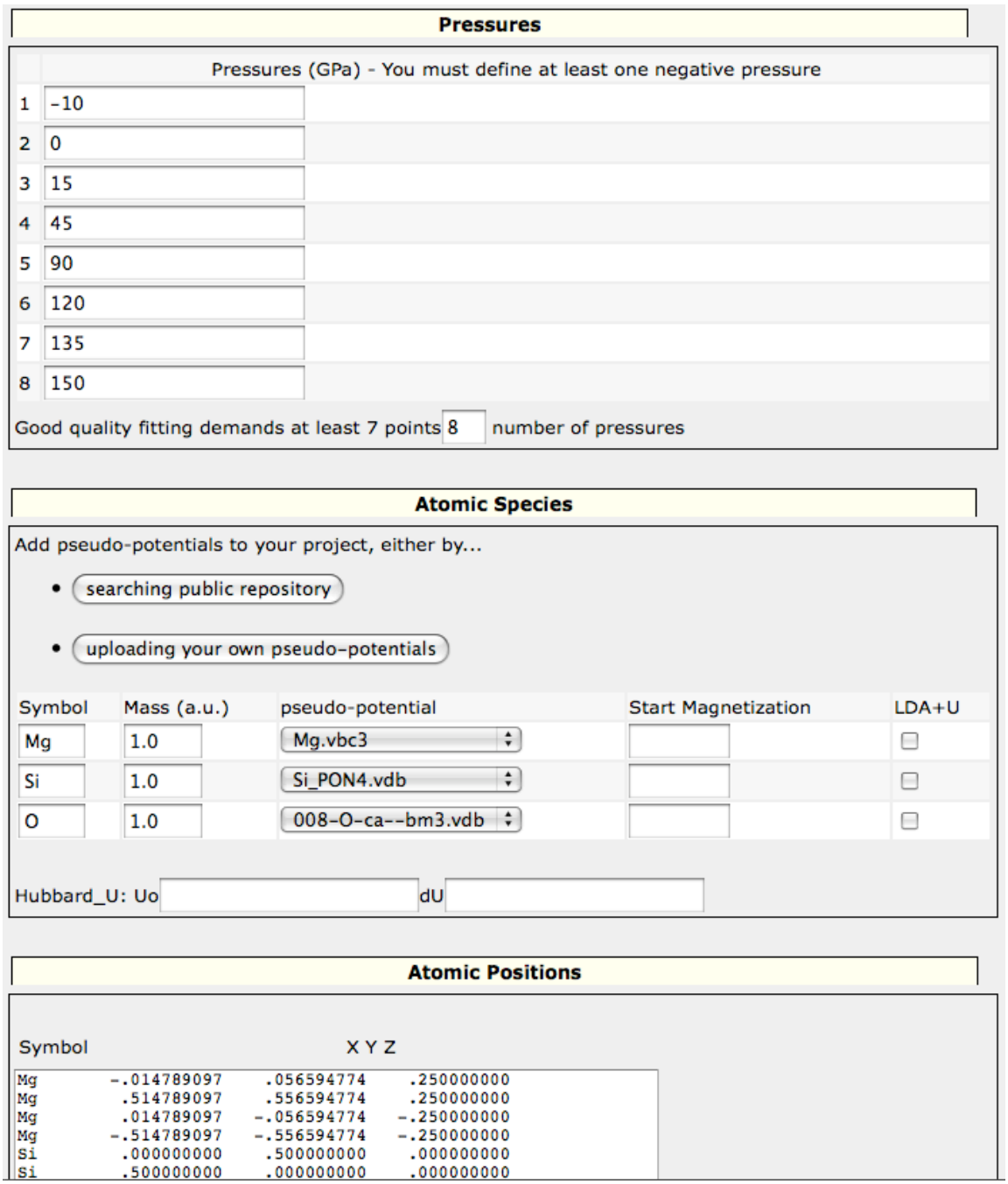


Figure 22. Equation of state input with eight pressures.

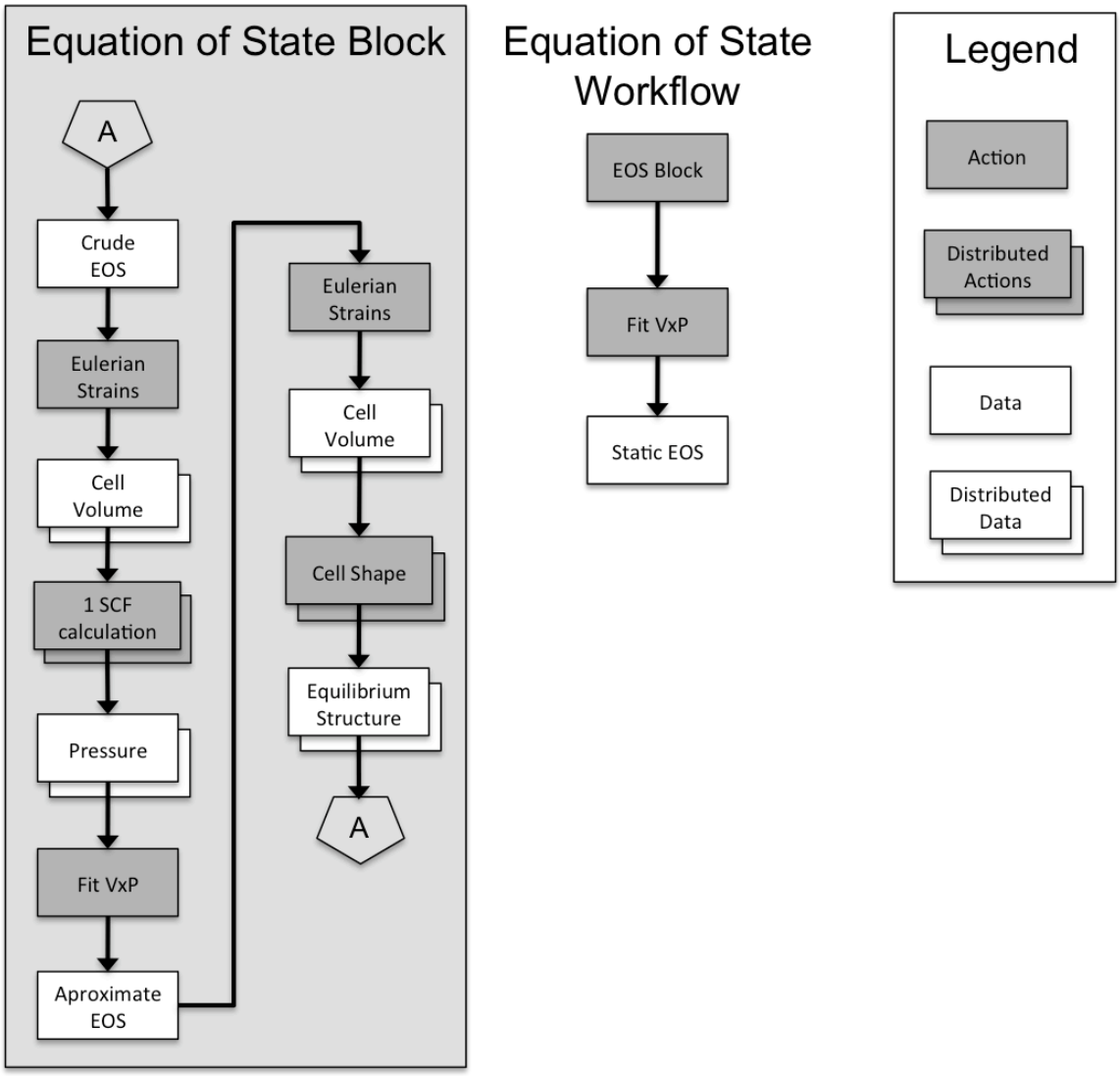


Figure 23. Equation of State Block.

4.1.3 Static Elastic Constant Tensor

The Static Elastic Constant Tensor (*Static* C_{ij}) project type in the ***VLabWp*** requires the user to define a minimum set of eight pressures, similar to the Equation of State (*Static EOS*) project. In addition, it involves definition of strains, i.e., a set of lattice deformations to be applied to equilibrium configurations at each pressure P_i. (See Figure 24)

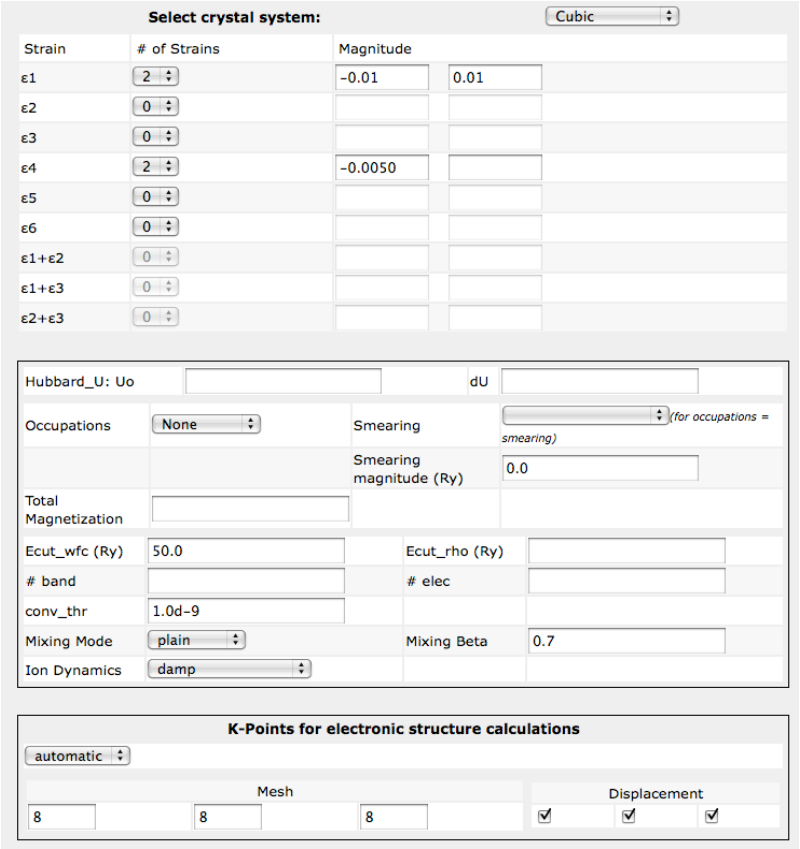


Figure 24. Static strains being defined by the user on a C_{ij} project.

In order to run the Static Elastic Constant project, it is necessary to run the EOS Refinement phase first. Static elasticity (*Static C_{ij}*) obtained from a set of specified strains, e_I, and pressures, P_i, for *J=1 to N, N ≤ 6* and *i=1 to n*, with *n* ~ 10. The number of calculations is *≤ s * 6 * 10, s* ~ 2 to 5. The factor *s* originates in the need to apply strains *e*, of various magnitudes simultaneously.

The selection of the strains for a C_{ij} project can be defined by chosen the proper crystal system. As depicted in Figure 24 crystal system cubic and 3 strains fill up, the decision of the crystal system enables the appropriate fields of strains to be entered. Table 2 enumerates seven systems, together with the minimum number of strains necessary to obtain them. Figure 25 shows the sequence of steps executed to acquire the elastic coefficients of a material from the initial configuration preparation to the computation of C_{ij}'s.

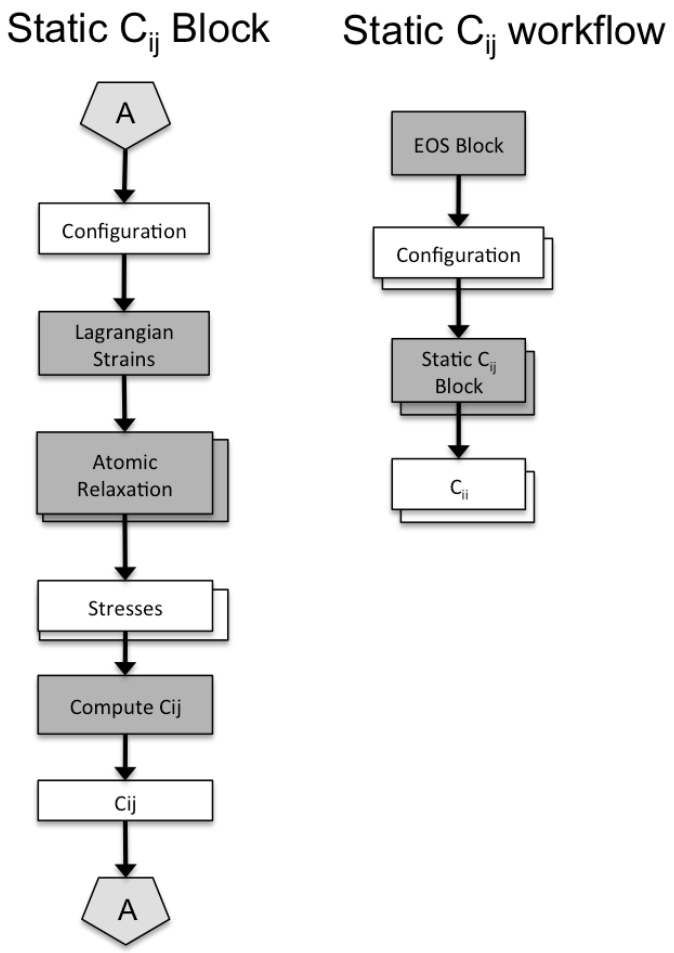


Figure 25. Scientific workflow for static C_{ij} calculation including the EOS workflow.

In a C_{ij} project example, the simulation preparation starts with a user login to access the ***VLabWp***. In the Portal, the user creates a project (simulation), and project properties such as project name, project type (currently supporting EOS/C*ij*/Thermodynamics), chemical formula, structure, material, exchange correlation functional, QE version, amount of CPU and memory used for each QE task, and wall time must be filled. These parameters are associated with the simulation throughout the execution life cycle. A C_{ij} project for diamond (2 carbon atoms per cell) is illustrated below. Figure 26 shows the first input page. The user enters a set of pressures (here ranging from -20 to 500 GPa, upper left side of Figure 26), atomic species with masses (upper right corner of Figure 26), and

atomic positions (middle right of Figure 26) in the unit cell with specific lattice parameters (bottom of Fig. 26) of the material.

System	Strain Components
Triclinic	$\pm\epsilon_1$ $\pm\epsilon_2$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_5$ $\pm\epsilon_6$
Monoclinic	$\pm\epsilon_1$ $\pm\epsilon_2$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_5$ $\pm\epsilon_6$
Orthorhombic	$\pm\epsilon_1$ $\pm\epsilon_2$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_5$ $\pm\epsilon_6$
Trigonal (8)	$\pm\epsilon_1$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_6$
Trigonal (7)	$\pm\epsilon_1$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_6$
Hexagonal	$\pm\epsilon_1$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_6$
Tetragonal (7)	$\pm\epsilon_1$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_6$
Tetragonal (6)	$\pm\epsilon_1$ $\pm\epsilon_3$ $\pm\epsilon_4$ $\pm\epsilon_6$
Cubic	$\pm\epsilon_1$ $\pm\epsilon_4$

Table 2. Elastic coefficients and necessary strains for nine unique systems.

These will be optimized for each pressure. Pseudopotential files are uploaded by the user or chosen from Quantum ESPRESSO pseudopotential database available in the ***VLab*** portal [15]. The second input page shown in Figure 24 contains strains used in the calculation of the three elastic coefficients of diamond, a cubic system (see Table 2 for strain specifications). Finally, the user specifies other parameters for the DFT calculation. These include kinetic energy cutoff (Ecut_wfc), **k**-point mesh for Brillouin zone sampling of electronic states, convergence criteria, etc. Default values are used for parameters unspecified in this input page. The simulation is then submitted for execution at which point all information entered in the web portal is saved into the metadata

and passed to web service PE. The PE splits tasks and groups them in “bags of tasks”, if necessary, for submission in distributed environments based upon the project type, number of tasks, and available resources.

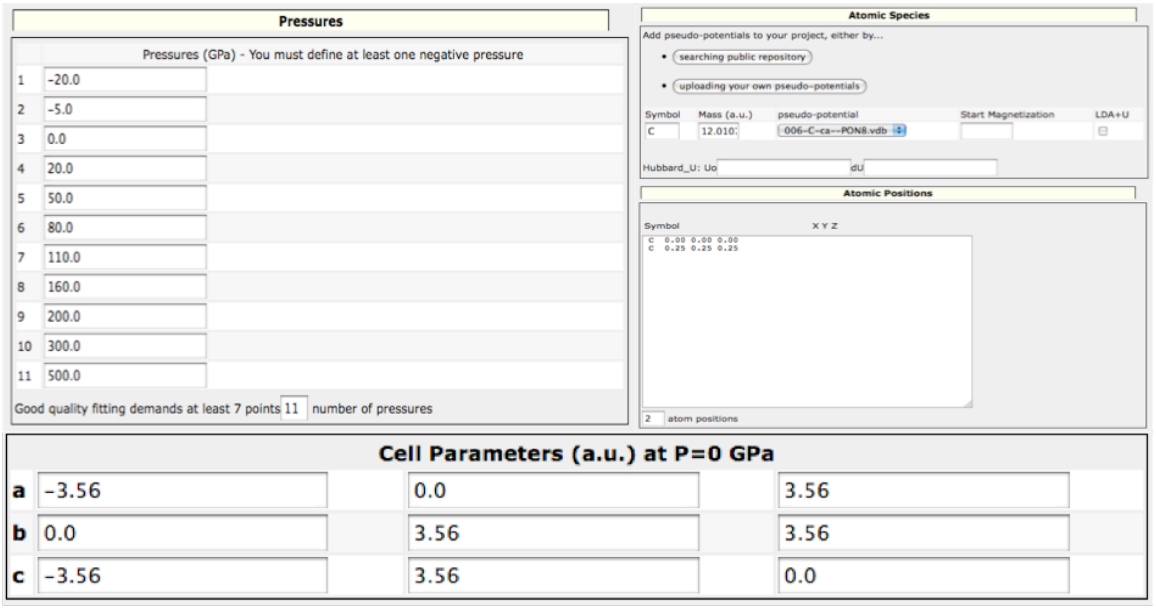
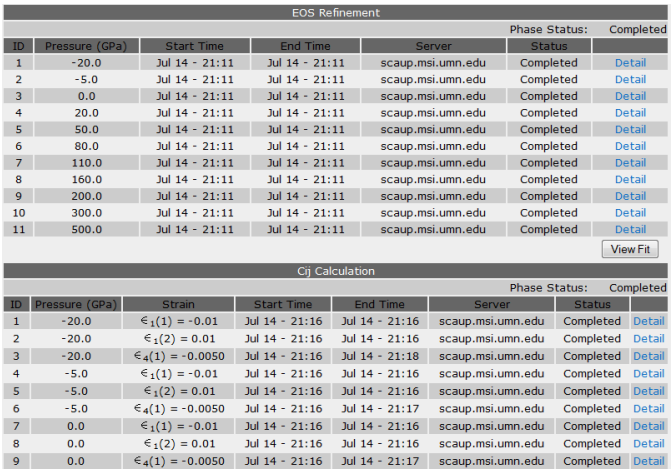


Figure 26. First input page specifying pressures and crystal structure for the calculation of diamond’s elastic coefficients.

Figure 27 shows a monitoring page containing two phases created for this calculation. The first phase, “EOS Refinement”, corresponds to structural optimizations of diamond at each pressure previously specified. This phase can also be executed as the standalone EOS workflow. Each line corresponds to the execution of a structural optimization task at a specific pressure. The fitting of the energy versus volume relation by a finite strain expansion can be viewed by clicking “View Fit”. The second phase, “C_{ij} Calculation”, obtains the stresses necessary for the calculation of C_{ij}. Each line is an independent task execution and corresponds to the application of a specific strain at a specific pressure. The time to complete these two phases varies with the availability of HPC resources, the material, and QE specifications (i.e. number of atoms, numerical accuracy,

number of pressures, etc.). Starting and finishing times are also shown in this page. At any time, the user can click on the “Detail” link to view input and output, edit and resubmit a task in case of failure, or check execution path as depicted in Figure 13.



EOS Refinement

Phase Status: Completed

ID	Pressure (GPa)	Start Time	End Time	Server	Status	
1	-20.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
2	-5.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
3	0.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
4	20.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
5	50.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
6	80.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
7	110.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
8	160.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
9	200.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
10	300.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail
11	500.0	Jul 14 - 21:11	Jul 14 - 21:11	scaup.msi.umn.edu	Completed	Detail

View Fit

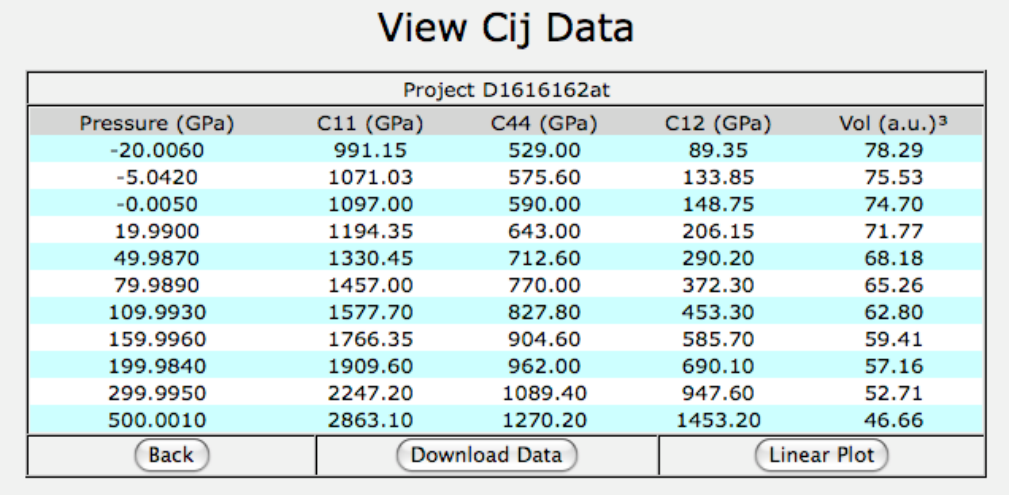
Cij Calculation

Phase Status: Completed

ID	Pressure (GPa)	Strain	Start Time	End Time	Server	Status	
1	-20.0	$\epsilon_1(1) = -0.01$	Jul 14 - 21:16	Jul 14 - 21:16	scaup.msi.umn.edu	Completed	Detail
2	-20.0	$\epsilon_1(2) = 0.01$	Jul 14 - 21:16	Jul 14 - 21:16	scaup.msi.umn.edu	Completed	Detail
3	-20.0	$\epsilon_4(1) = -0.0050$	Jul 14 - 21:16	Jul 14 - 21:18	scaup.msi.umn.edu	Completed	Detail
4	-5.0	$\epsilon_1(1) = -0.01$	Jul 14 - 21:16	Jul 14 - 21:16	scaup.msi.umn.edu	Completed	Detail
5	-5.0	$\epsilon_1(2) = 0.01$	Jul 14 - 21:16	Jul 14 - 21:16	scaup.msi.umn.edu	Completed	Detail
6	-5.0	$\epsilon_4(1) = -0.0050$	Jul 14 - 21:16	Jul 14 - 21:17	scaup.msi.umn.edu	Completed	Detail
7	0.0	$\epsilon_1(1) = -0.01$	Jul 14 - 21:16	Jul 14 - 21:16	scaup.msi.umn.edu	Completed	Detail
8	0.0	$\epsilon_1(2) = 0.01$	Jul 14 - 21:16	Jul 14 - 21:16	scaup.msi.umn.edu	Completed	Detail
9	0.0	$\epsilon_4(1) = -0.0050$	Jul 14 - 21:16	Jul 14 - 21:17	scaup.msi.umn.edu	Completed	Detail

Figure 27. Phases of C_{ij} calculation for diamond.

After completion of both phases, the elastic coefficients can be displayed in table format, as shown in Figure 28. There is also an option to download the data created and display it as a linear plot on the web portal shown on Figure 29.



View Cij Data

Project D1616162at

Pressure (GPa)	C11 (GPa)	C44 (GPa)	C12 (GPa)	Vol (a.u.)³
-20.0060	991.15	529.00	89.35	78.29
-5.0420	1071.03	575.60	133.85	75.53
-0.0050	1097.00	590.00	148.75	74.70
19.9900	1194.35	643.00	206.15	71.77
49.9870	1330.45	712.60	290.20	68.18
79.9890	1457.00	770.00	372.30	65.26
109.9930	1577.70	827.80	453.30	62.80
159.9960	1766.35	904.60	585.70	59.41
199.9840	1909.60	962.00	690.10	57.16
299.9950	2247.20	1089.40	947.60	52.71
500.0010	2863.10	1270.20	1453.20	46.66

Back | Download Data | Linear Plot

Figure 28. Final results showing diamond’s calculated C_{ij}.

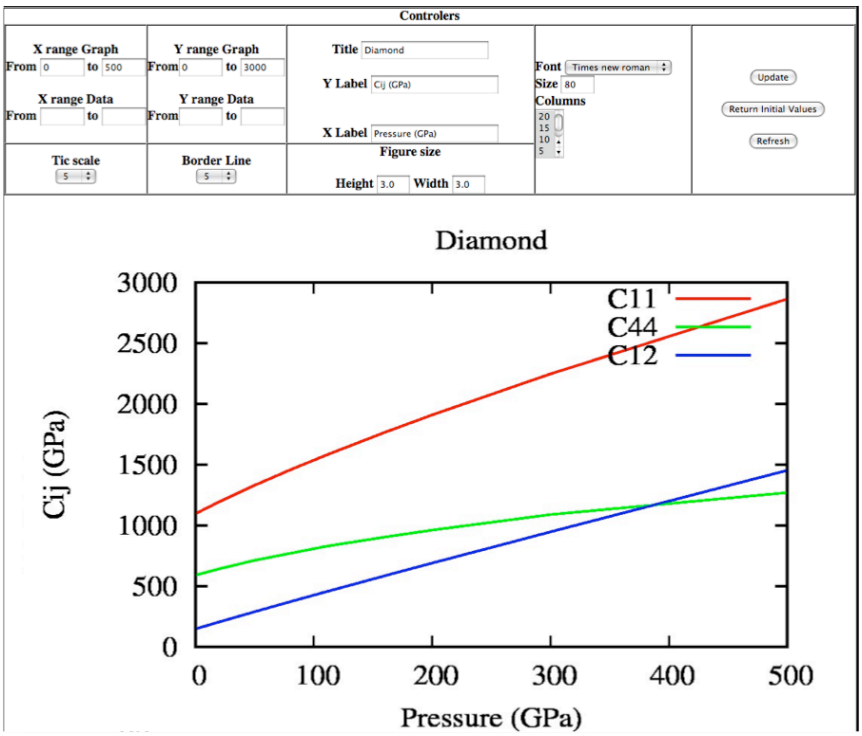


Figure 29. Line Plot for diamond's calculated C_{ij}.

As previously stated, results produced by execution of the Cij workflow in ***VLab Wp*** agree completely with results produced manually [34].

4.1.4 Thermodynamics

Thermodynamics properties workflow, a.k.a. *Thermal EOS* workflow, also requires the user to define a minimum set of eight pressures, P_i (i=8). In addition the user must define a **q**-point grid for lattice dynamics calculations, a temperature range and several pressures, P_i.

The number of phonon ***q***-vectors, *n,* is approximately ten. The number of calculations is *n*8.* In order to run the Thermo EOS, the ***VLabWp*** needs to execute first Approximate EOS and EOS Refinement phases to start generating the tasks for Thermo EOS. The block diagram of the Thermal EOS workflow is presented in Figure 30.

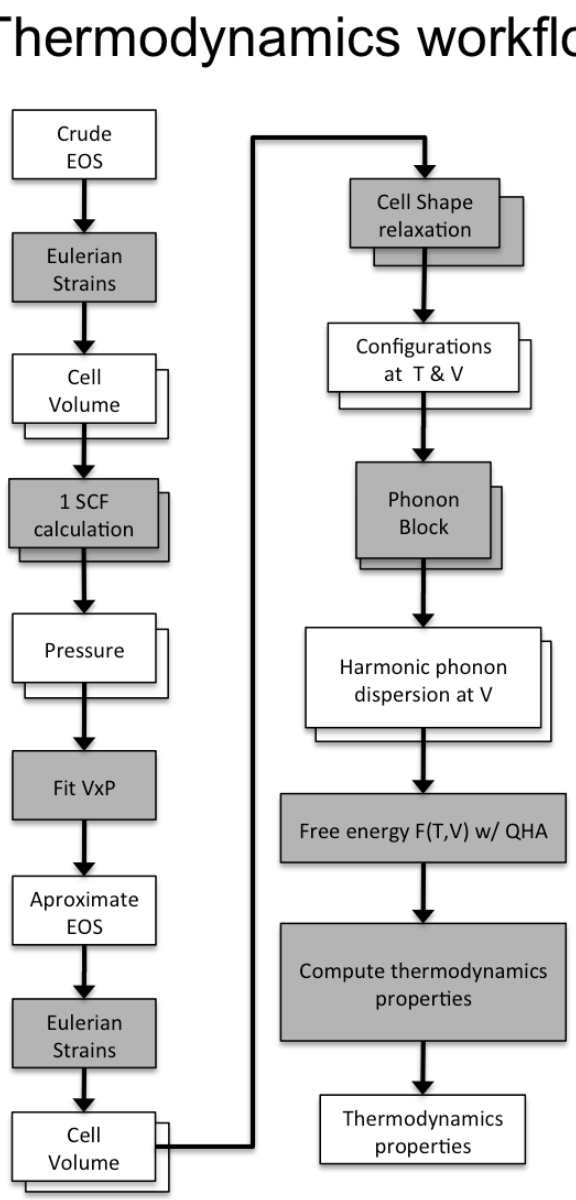


Figure 30. Scientific workflow for thermodynamics calculation including the EOS workflow.

The method to compute free energy from volume dependent vibrational density of states is based on the quasi-harmonic approximation. From free energies all thermodynamics properties can be computed. It is a simple "post-processing" step. Figure 31 shows a portlet where the user has the option to specify the desired thermodynamics properties and format to be plotted i.e., the

output grid, and pressure range. Figure 5 shows a plot of results created by *Thermal EOS*, the Helmholtz Free Energy (F).

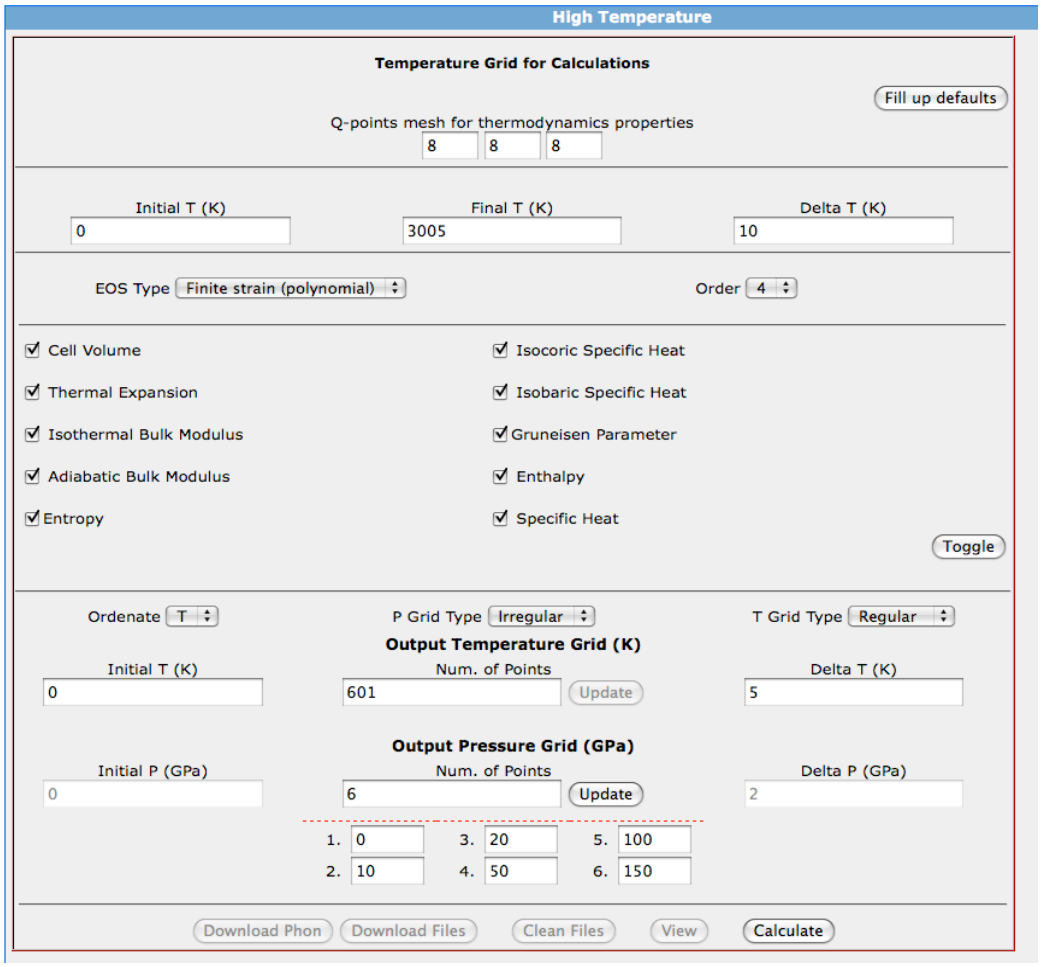


Figure 31. High Temperature menu to plot results.

Recently, Prof. Wentzcovitch and her post-docs Dr. Tao Sun and Dr. Dong-Bo Zhang have introduced a method that combines lattice dynamics and molecular dynamics and produces accurate free energies from zero Kelvin (quantum vibrations) to very high temperatures near melting (classical motion). A new workflow has been designed for this method. It extends the Thermodynamics workflow to include molecular dynamics simulations simultaneously with phonon calculations (see Figure 32).

The High PT-MD block, Quasi-particle block, and PGM block will enhance the workflow options in the ***VLabWp***. At first, these three news blocks will be incorporate into the current metadata and new input pages with new options will be developed to capture require information.

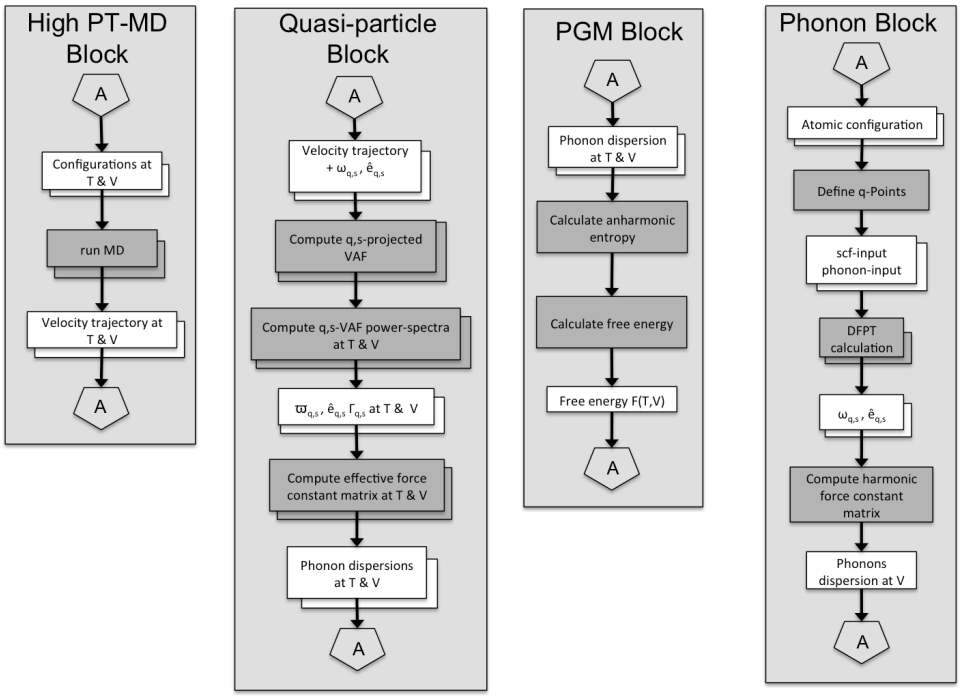


Figure 32. Scientific workflow for thermodynamics calculation. Application blocks for the thermodynamic calculation performed in ***VLabWp***.

4.2 Workflow Management

Definition of metadata is crucial in designing a system required to support concurrent execution in a heterogeneous distributed environment. ***VLabWp*** must keep metadata to track the execution of tasks and the locations of a plethora of input and output data files that lie scattered throughout several HPC nodes. These tracks are stored in a database structure in way to allow the user through

the Web Portal to retrieve status of each task, save properties for each task, and allow collaborative work among users avoiding race conditions.

4.2.1 Tools used

In ***VLabWp***, workflows are sets of operations executed by heterogeneous applications in a pre-defined sequence. It uses Unix shell scripts to parse text files for input preparation and to parse large output files. More specifically “wak” and “grep” software are employed due to much faster performance when compared to Java Application to parse large output files.

It also uses software developed in C to perform post-processing calculations, e.g., equation of state fits such as Vinet equation of state [35] or several orders of Finite Strain [36]. It also uses post-processing Fortran programs, such as “matdyn.x”, to compute and diagonalize Dynamical matrices to get phonon frequencies for calculations of Thermodynamic Properties. Java application was developed to calculate the elasticity coefficient for static elastic constant. The call for heterogeneous applications in ***VLab*** is made through web services, which are developed in Java encapsulated in Web Service and triggered by the user in the Web Portal.

4.2.2 Receipt Metaphor

Concerning metadata and the ***VLabWp*** database, a particular object of consideration is the protocol of communication understood as “receipt”. The receipt is a metaphor created in the ***VLab WP*** for requested tasks that require long execution times and are returned by a ticket as a receipt. The receipt ID is used to identify a task when requested to check for status and/or insert/get any other property inside of receipt task. One project in ***VLab Wp*** is stored as a receipt, one phase is considered as one receipt and it is stored as attachment to a project.

A task is also a receipt, but associated to phase receipt. For this reason, the receipt was designed as a recursive data structure as depicted in Figure 33. The receipt is a java class containing a few parameters as explained in da Silveira et el [24] and it is converted into an XML string to be saved in the database. The receipt class has a method called “attachment” that returns a collection data type, i.e., a list of receipts.

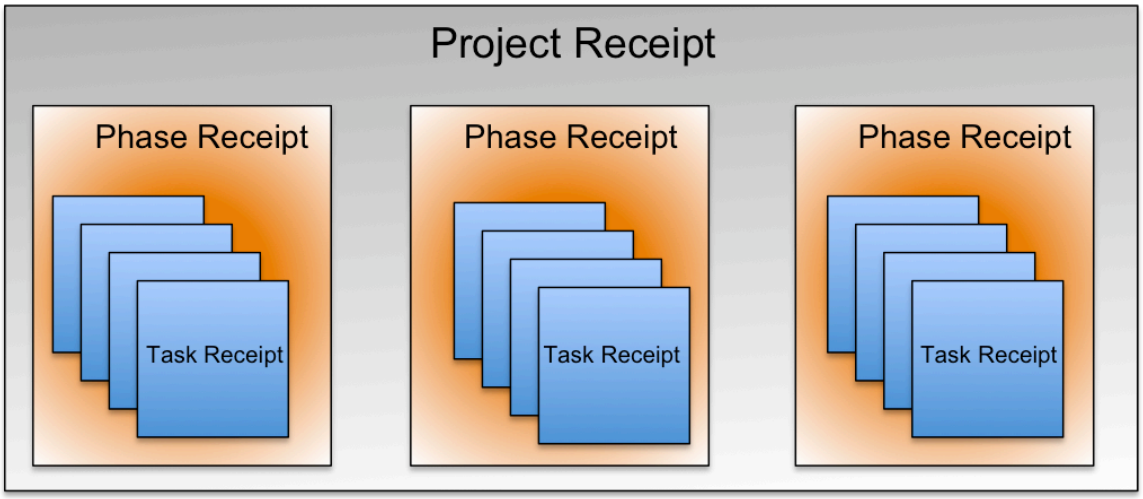


Figure 33. Receipt metaphor in ***VLab Wp***.

For *Project Executor* tasks corresponding to a given project phase, task receipts are attached to the phase receipt, and the latter is, in turn, attached to the project receipt. In this manner, the project receipt contains receipts of every task in the project. In order to achieve persistence, the project receipt is attached to a table called “project descriptor” and stored in a central database. The *Project Executor* also keeps a cached copy of the receipt.

The receipt metaphor identifies every action by the workflow and records the action properties inside a java class, which is then used to keep track and communicate the action’s status until it finishes. Every action is identified by its ID, execution path, host name, and by the port number where it is performed. Since a set of receipts forms one phase in the scientific workflow, a linked list of receipts is organized for their management. Therefore, one initial receipt is

created and identified as the project receipt, which subsequently receives a list of phase receipts. Similarly, each phase creates a list of task receipts [24]. Once a task is submitted for execution to the web service *Task Executor*, these receipts define the protocol for communication between web services to guarantee correct exchange of properties until the execution cycle is completed. A receipt-cached copy is used when a project needs to be resubmitted because of a failure, or in case of a task is changed by a user on web portal. Thus to keep consistency of receipt on Web Service cached copy is made. A full database diagram describing how the receipt is incorporated into the metadata is described in Ref. [24].

Chapter 5

5. Task Scheduling in Heterogeneous HPC systems

In distributed computing systems, algorithms for task scheduling, service oriented architecture, and fault tolerance are still topics of intense discussion especially because of the large-scale use of Cloud Computing, Hypermedia, and P2P algorithms optimization. One of the good reasons to develop a scheduler, such as the one designed for ***VLab*** is to provide multiple SG users with a customized option to place workflows for execution and provide HPC resources in one single environment. Moreover, the SGs listed on the XSEDE website do not have HPC resources manager in a browser interface. Therefore, the planned scheduler development can be useful to other XSEDE communities as well. Another good reason is to provide an efficient way to manage the use of a large diversity of HPC systems available today.

In general, researchers have a plethora of optional HPC resources with large numbers of processors, memory, and archive disks available for submitting scientific workflow tasks. Usually researchers are able achieve the desired high throughput by deciding which resource to use with best processors, most convenient memory distribution, or compiler maker. However, before such decision can be made some factors must be considered and understood: it takes long waiting hours on queues to make tests, tests must be made on several different computational systems, one must consider operating system errors, and understand resource management formalities, e.g., PBS, and maintenance on HPC. In addition, other issues may appear when dealing with different software versions, or with compilation steps depending on processor brands, system architecture, and operating system flavors. These inferences influence task preparation and submission to different distributed computing systems. All these concerns are still present when an SG manages a workflow distribution. However, an SG can collect tests results, analyze performance and do statistics and decide more easily, though not perfectly, on appropriate resource allocations

for particular workflow executions. Therefore, there is great need for a job scheduling software or framework to interface different HPC systems with the task submission service of an SG.

The main function of a scheduler is to plan tasks according to defined agenda and queue according to job types, resources requirements, resources availability, and scheduling policies [37]. HPC resources are typically shared with an unpredictable research workload. The HPC system's ability to schedule and complete high priority tasks in a timely manner is therefore essential in the daily execution of scientific workflows from SG models. The proposed job scheduler in this research has a distributed resource manager with all available HPC systems listed for job distribution. The main goal of the resource manager is to achieve the best utilization of resources and to maximize system throughput by orchestrating the process of assigning resources to users' workflow jobs. According to Yonghong [38], until now development of distributed resource managers were mostly from industry vendors with a focus on feature additions based on current systems that were designed to manage moderate-sized grid systems, such as MOAB [39], PBS Pro [40], SGE [41], Torque [42], SLURM [43] and Load Leveller [44]. Also, for the next-level of scales in high-performance systems, either petascale or extreme-scale in some contexts, there are a few research projects addressing resource management issues for scale system software and HPC Colony [45].

Management of a scheduler using a fault-tolerance algorithm approach in a distributed environment aims to reduce the risk of scheduling failure, guarantee that jobs are coordinated according to user policies, and keep the SOA intensively used without a system administration intervention.

The ***VLab*** scheduler has some similarities to meta-schedulers designed for grid computing. However, a meta-scheduler is commonly used in distributed management systems, e.g., PBS, SLURM interconnected with each other, etc. It is too complicated to have the real benefit of a meta-sheduler from user of HPC

point of view. The meta-sheduler is rarely used because there is no hardware or software interconnecting distributed management systems from two different computation centers for task distribution. In the VLab Scheduler on the other hand, it is easier to accomplish because all tasks come from one single location, execute the same piece of software, and user behavior of the software (the QE) is well understood.

There are also similarities with the dynamic resource provisioning used at the Large Hadron Collider (LHC) Computing Grid at CERN [46] for grid computation and workflow distribution. LHC distributes tasks in different grids in Europe and in the US by using tiers of services to solve big problems that require massive amounts of data. In a recent collaboration between XSEDE, Open Science Grid (OSG) [47] and CERN, the LHC was capable of distributing workload by using in-house schedule resources, establishing runtime environment, executing massive workload, handling results, and cleaning up. The LHC workload distribution uses BOSCO [48] for authenticate and job submission in Gordon SC from San Diego Supercomputing Center and OSG. It also uses pilot job framework [49] to create hundreds of tasks in one resource, and Globus Online to exchange files for analysis between server and user desktop.

The main goals of a task scheduler for SGs are: **1)** to accelerate results delivery and maximize utilization while simplifying workload management across complex, heterogeneous cluster environments, or even HPC cloud environments, and achieve high job throughput in SGs, **2)** reduce the makespan to the lowest time possible, and **3)** get an optimum utilization of resources available.

5.1 Overview

Intense research in distributed computing has been conducted especially on integration of large-scale computing power into scientific computation through the Internet. A hot topic discussed today in the HPC world is Cloud Computing and

its integration into scientific research projects. Companies such as Microsoft and Amazon are producing a lot of incentives for research projects to emerge in the cloud. There are several applications and algorithms implemented and tested for task distribution in cloud computing. Since large companies started selling services and software for Cloud Computing, academic research started exploring heterogeneous systems and dynamic distribution, such as in distribution of scientific workflow applications on Amazon EC2 [50]. As an example, Amazon already provides services for the execution of scientific workflows using strategic algorithms for dynamic scheduling, e.g., backfilling for workflows in distributed systems [51].

The problem of distributing tasks in heterogeneous HPC systems is similar to some problems that arise in graph theory and in the design of operating systems. For example, the Ant Colony Optimization algorithm [52] is a technique for solving computational problems that can be reduced to finding optimum paths through graphs. The hierarchical taxonomy of scheduling algorithms for general purpose parallel and distributed computing systems described in Casavant et al. [53] is suitable for grid computing as grid is also a form of parallel/distributed environment.

The ***VLab*** Scheduler is designed according to the taxonomy of scheduling algorithms as depicted in Figure 34. It is a decentralized fault tolerant task scheduler with scheduling algorithms for global resources. Currently, the computational resources in the ***VLab*** SG are geographically distributed and have static scheduling with sub-optimal features using heuristic properties. By definition, static task scheduling requires the user to define wall time, location, and task properties before submission. Unlike static task scheduling, the dynamic scheduling option is unaware of the location of available resources and of how the distribution will be made. The dynamic scheduling option assumes all tasks will run successfully independent of location, makespan, load balancing, and system architecture. A common example is the peer-to-peer (P2P) network. A

dynamic scheduling algorithm for grids has been developed in Ref. [54] by incorporating a predictive strategy into a static scheduling algorithm.

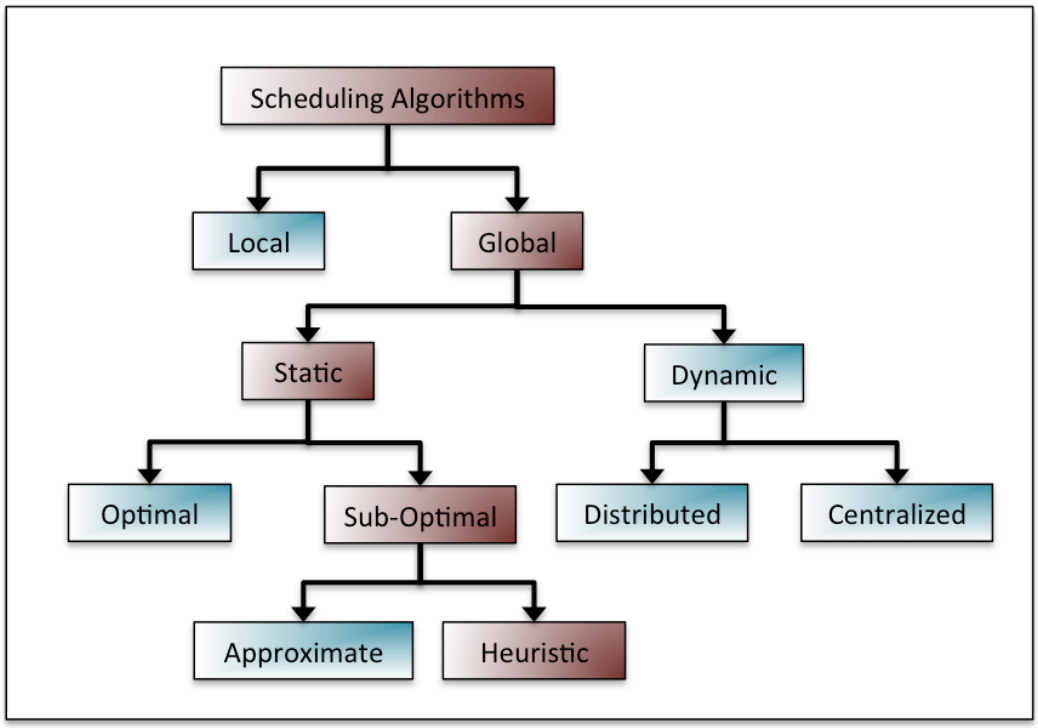


Figure 34. Taxonomy of scheduling algorithms (adapted from Ref. [53]).

By assuming the sub-optimal feature option, the task scheduler in ***VLabWp*** generally selects resources based on heuristic properties since resource information data are well understood and clear. Tasks from ***VLabWp*** workflows not always perform an optimal solution when placed for execution. Therefore, there is no real assumption for optimal solutions and sub-optimal scheduling is appropriate for periodic batch scheduling and for decentralized approaches.

5.2 Implementation

The work started by testing and implementing a list of scheduling algorithms and using a resource manager to handle all HPC's used in ***VLab***. The ***VLab*** Scheduler is integrated into the current ***VLabWp*** as depicted in Figure 35.

The web service Project Executor described in section 3.3.1, that is responsible for orchestrating scientific workflows of projects created by VLab users through the web portal, transfers tasks directly to the ***VLab*** Scheduler. The VLab Scheduler uses the same semantics of the Web Service *Task Dispatcher* [15]. For that reason, the Scheduler has client stubs libraries to communicate with computation nodes. As depicted in Figure 35, the Apache Airavata stubs are used on XSEDE HPC Stampede and Blacklight, and the TE Stubs to MSI HPC Itasca. Assuming the scheduler receives tasks from all users working on ***VLabWp***, it manages to distribute accordingly to the appropriate HPC. The communication process is the same already implemented in ***VLabWp*** using the web service framework Axis 2.0 [25] and the receipt metaphor [26] for metadata.

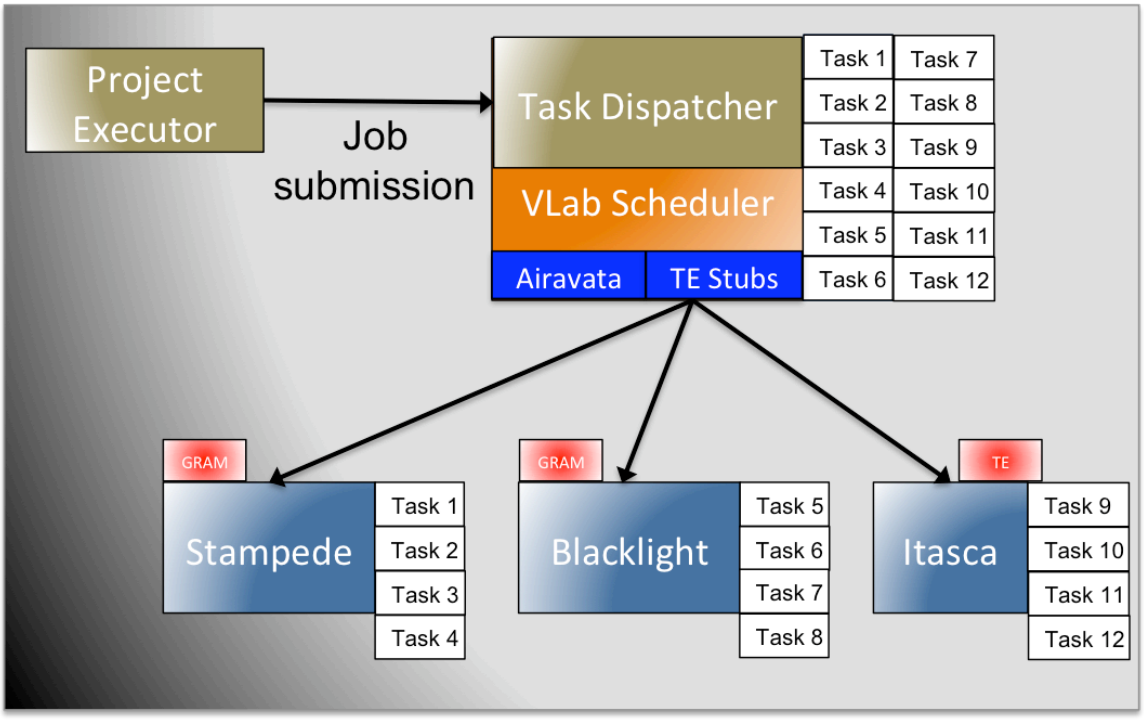


Figure 35. Overview of ***VLab*** Scheduler in the ***VLab*** SG.

The general goal of this task scheduling implementation is to provide users with a customized option to change/choose how tasks generated by a scientific workflow are distributed based on number of tasks, number of resources available, number of cores per task, wall time, and software version.

Essentially, the user, who must be knowledgeable about the systems available, can delegate how all tasks will be distributed or leave the obstacle of choosing the right resource for a task to the ***VLab*** Scheduler. The interface provides job-scheduling options for users to submit their "project" (scientific workflow) tasks in ***VLabWp***. These options are defined based on empirical tests of certain algorithms with best performance results.

Combination of several factors, e.g., number of cores per task, software version, memory requirement, wall time, and number of tasks in a BoT are normally considered when scheduling tasks from an entire scientific workflow in a distributed HPC environment. Providing customization options to allow the users to determine how tasks should be distributed is a good idea because the workload of HPC systems can vary considerably in time. Users can easily check the status of systems before submitting their tasks while a frequent inquiry by the scheduler would be an intrusive action on the HPC systems. As a result, users are able to change those parameters according to HPC status at the moment of task submission or allow the scheduler to determine the best approach for task distribution according to available information.

5.3 Resource Manager

According to A. Abraham [55], in general a grid resource manager estimates the resource requirements and provides functionalities for discovery of and publishing resources as well as scheduling, submission, and monitoring of jobs. HPC resource management systems should ensure that the necessary resources are available when needed and are guaranteed for the duration of the operational job. The resource manager in the ***VLab*** Scheduler keeps metadata on the current PostgreSQL 8.2 database already implemented in the ***VLabWp***. It has information associated with each resource, e.g., maximum number of tasks allowed, maximum wall time allowed, maximum number of requests allowed, maximum memory request allowed, file system path, and a list of Quantum Espresso (QE) versions supported. Those features are distinctive for each HPC

resource and the policies are defined differently in each computation center. Each resource also has its own specific way to establish connection, accept tasks, and monitor capability. This information is also available in the resource manager interface.

At MSI, for example, the web service Task Executor [15] is used for submitting and monitoring jobs on Calhoun and on Itasca HPC. On XSEDE resources, e.g., Stampede, LongHorn, Backlight and Kraken, use the Apache Airavata framework API [14]. The Airavata Application Programming Interface (API) is developed to integrate Science Gateways or external application into the XSEDE. In view of the fact that the XSEDE has heterogeneous HPC systems, each HPC has its own workload grid manager e.g. PBS, SLUR, and Load Leveller. The attraction of the Airavata API is that it uses the Grid Resource Acquisition Manager (GRAM) [56], which is a protocol capable of handling communication with workload grid managers. Therefore, the GRAM is capable of creating scripts for each one of the workload grid managers supported in the XSEDE HPC resources. Since the ***VLab*** Scheduler is part of the ***VLabWp***, it is developed as an extended version of the web service Task Dispatcher using AXIS 2 web service framework.

The ***VLab*** Scheduler receives all task execution requests from the web service Project Executor and saves into a local metadata. It will then place all tasks in order, as in a queuing system, and apply a scheduling algorithm based on properties the user requests before the scientific workflow is set to run. When the scheduler finds the appropriate resource, it will transfer a task and its files - aka, an execution package - to the suitable resource selected. As shown in Figure 35, each resource has its own queues of received tasks. It is common knowledge among HPC users that each resource only receives a certain number of tasks per user. Therefore, the ***VLab*** Scheduler allows users to access only his/her allocation and own resource limits. Table 3 provides the list of heterogeneous HPC resource options used for distribution of scientific workflow

tasks in ***VLabWp***. The first two resources are located at MSI Itasca and Calhoun, while the remaining are XSEDE HPC systems.

Resource	Processor	# cores	Memory per node	Grid resource manager
Itasca	Two quad-creo 2.8GHz Intel Xeon X5560 Nehalem EP	8,728	24G	TE/PBS
Calhoun	Two quad-core 2.66GHz Intel Xeon "Clovertwon"	1,440	16G	TE/PBS
Blacklight	Intel Xeon X7560	4,096	8G	Airavata/PBS
Ranger	AMD Opteron, four-socket quad-core	62,976	24G	Airavata/PBS
Kraken	Cray XT5 AMD Opteron SeaStart	112,896	16G	Airavata/PBS
Lonestar	2 Hex-code Xeon 5680	22,656	16G	Airavata/SGE
Stampede	2 8-core Xeon E5	102,400	32G	Airavata/SGE
Steele	Two 2.33 GHz Quad-Core Intel E5410	7,216	16G	Airavata/SGE
Gordon	Intel EM64T Xeon E5	16,384	64G	Airavata/PBS

Table 3. Resources available for task distribution for ***VLab*** SG.

As can be seen in Table 3, each resource has a different number of different processor, which means the QE must be configured and optimized to work properly on it. Also each resource has a certain sum of cores and memory per node. Those aspects are important to be understood ahead because it impacts on overhead added to run QE due to network communication. Cluster nodes with shared memory among nodes, such as Blacklight, tend to perform better when running QE than non-shared memory clusters.

5.4 Schedule Core

Scheduling algorithms implemented in the ***VLab*** Scheduler are integrated with metadata in the PostgreSQL database. Figure 36 gives an overview of the ***VLab*** Scheduler code diagram. The code diagram starts in the system admin block, where there is a set of methods and functions to add/delete/update

information about each accessible HPC resource in the resource repository. The SG system administrator must update the HPC resource information in the repository. The scheduler block will access the most recent resource information on the repository every time resource data is updated. The scheduler block diagram in Figure 36 contains all algorithms for task distribution. It also receives user settings and instructions for tasks submission.

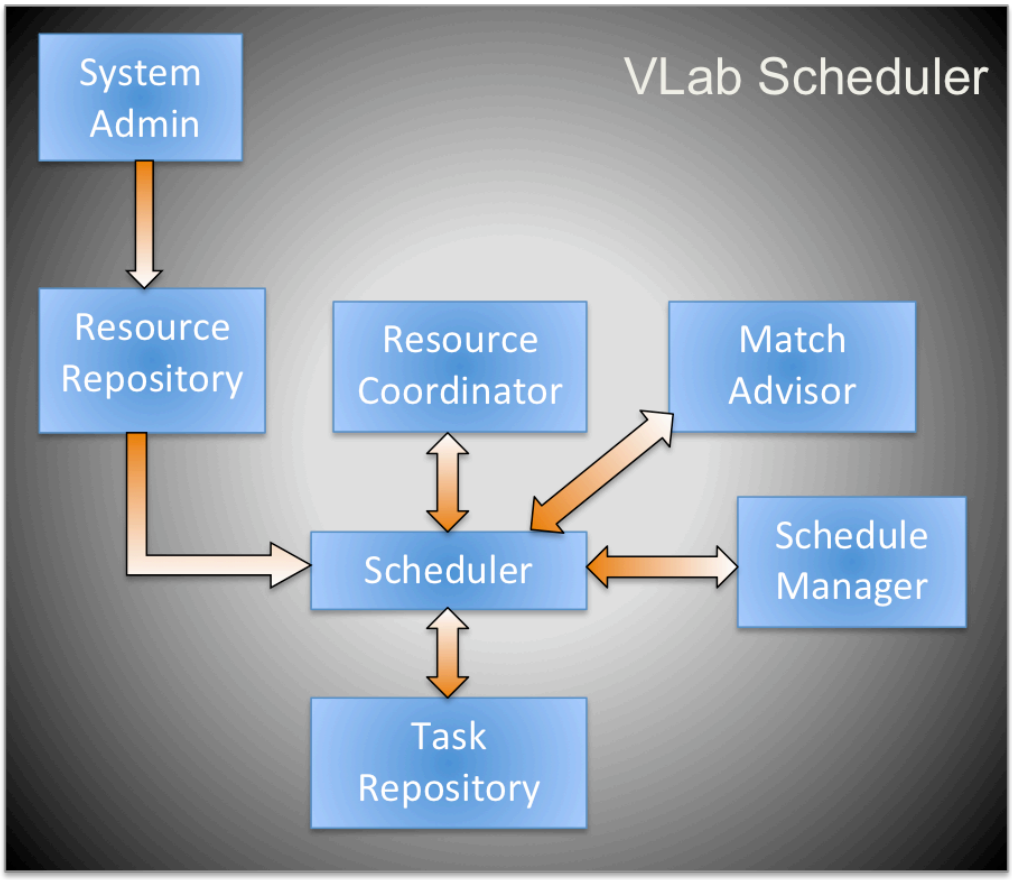


Figure 36. VLab scheduler code diagram.

The task repository block is a small fragment of metadata containing information about all tasks received. It contains Task ID, project name, phase name, number of cores for the task, memory required by the task, wall time, SG user name, and HPC user name. The difference between the SG user name and HPC user name emerges from the fact that SG users can submit tasks with their own HPC user name and credentials, or, alternatively, using community

accounts. Hence, for every task recorded inside the task repository one SG user name can have more than one option of HPC user name, but only one HPC user must be set. Tasks are stored until they finish their execution cycle. At first, all tasks are placed on a queue inside the task repository. After a task's execution cycle is completed a flag option is set to mark the task for removal from the queue. Therefore, every task inserted into the task repository pass by the following status on ***VLab*** scheduler, on Grid Queue, Running, Cycle completed (not considered on the schedule).

The resource coordinator block works directly with the scheduler to control every connection method to each HPC available. It contains API libraries for connections to Globus Tool Kit, Apache Airavata, and Web Service TE. This block is responsible for checking if a resource is available to be considered by the scheduling algorithm. This feature is necessary because HPC resources constantly set off for maintenance, and sometime, for a period of hours or days, is out of reach. The resources manager uses the following attributes from each HPC resource: IP address, host name, main scratch directory, maximum number of tasks allowed, maximum wall time, QE executables path location, accounts credentials and etc.

Match advisor is the block responsible for determining the right resource for the chunk of tasks found by the scheduler. It follows restrictions of every resource requirements normally imposed by the system administrator to avoid exceeding the maximum number of tasks, maximum memory request, and maximum wall time for an effective matchmaking. Thus, match advisor is capable of placing a collection of tasks from one particular user into one HPC resource that does have the required processing power to perform the operation.

The schedule manager is a block of code responsible for creating scripts according for selected resources. This block knows how many tasks can be included in individual submissions (BoT) for each resource. It follows all restrictions established by selected algorithms and by the match advisor. In

consequence, this block uses the appropriate API to communicate with the right HPC resource selected.

5.5 Tasks Preparation and Execution

Task preparation and submission to an HPC resource is trivial, but, considering that the same process must be repeated hundreds of times and in each occasion input and output must be slightly changed, it can be cumbersome and error prone process. In order to accomplish this process smoothly and seamlessly, each HPC resource has its own way of doing it. Therefore, here I present the way it is done in the XSEDE. At MSI resources, the sequence of steps is described in Ref. [24].

In the XSEDE, the Apache Airavata is the intermediate software (middleware software) used. In a nutshell, The Airavata is a tool to design scientific workflows with capability to submit task to XSEDE resources using GRAM [14]. The Airavata tool starts by using the client application on a desktop computer, the Visual Interface called XBaya. Once on Xbaya, an application service must be set for each resource intended to be used. This application service needs the basic properties for a communication with the XSEDE resource, e.g., user name as in the XSEDE Web Portal (the same as the XSEDE Web Portal for credentials propose at the HPC side), GRAM URL of the resource, standard Input and Output names with an addition of an index to identify every task, memory and cores per node (each HPC has its own specific architecture), etc. The GRAM URL identifies the host name of an HPC resource. In the Application Interface the location of the QE and a working directory where the user intends to run each instance of QE must be set. This working directory in an HPC resource is usually understood as a temporary scratch file system with very large amount of disk space.

After the application server is completed in the Xbaya workbench interface, a workflow engine is created and a set of inputs and outputs are

connected to it. This part means that a workflow engine using the application service previously defined will receive a set of inputs and will generate a set of outputs. The Web Service Project Executor sets the real name of the inputs and output files when running the workflow engine. This workflow engine is then saved in the Airavata Server, which is an instance application in the Web Container Apache Tomcat. In this scenario, the Apache Server is just another application in the same Apache Tomcat of the Task Dispatcher as depicted in the Figure 35. By having the workflow engine deployed on the Airavata Server and waiting for connections request, the Airavata API is capable of invoke the workflow engine and set all the required parameters for a workflow execution in a XSEDE resource.

Since the WS Project Executor and WS Task Dispatcher work in the same server node and file system, they are capable of having the same file path to call a PWscf input and output created by the WS Project Executor. When a workflow is set to execute, e.g. static elastic constant, all tasks associated with that workflow are created by the WS Project Executor in the share file system. The WS Project Executor invokes the WS Task Dispatcher to request to transfer each task file to XSEDE resources using the API Client, e.g., the Pwscf Input, Phonon Input, Pseudo-potential file. Those files are stored in the scratch file system of the HPC resource using the semantic $TIME_STAMP/InputData/ and $TIME_STAMP/OutputData/. The variable $TIME_STAMP means the date and absolute time (including milliseconds) of the moment the file was created. On the next step, the Airavata Server is called by the Airavata API to create the resource manager script for the task over the directory $TIME_STAMP/InputData and submit to the workflow grid manager, e.g., PBS, SGE, SLURM using the correct settings of the HPC.

The Airavata Server receives from the API a notification of tasks submitted with a task ID and the absolute time the task submission has completed. The API Client reports the receipt with the task ID to Project Executor and monitors the simulation cycle of the whole scientific workflow. After the task completed the

execution cycle, it returns to the WS Project Executor the path location of the output created. The API with the host name and path location establishes a new connection to XSEDE resource to transfer the output to Project Executor file System. This sequence of processes is repeated for many tasks in a whole project in ***VLabWp***.

In practice every user in the ***VLabWp*** can execute more than one project. All tasks of a project will be distributed in more than one server and each server supports a maximum number of cores. The scheduler distributes as many tasks as possible in all available servers and holds in the local queue tasks when the resources have been exausted. Figure 37 shows an example of a user case scenario of the ***VLab*** scheduler.

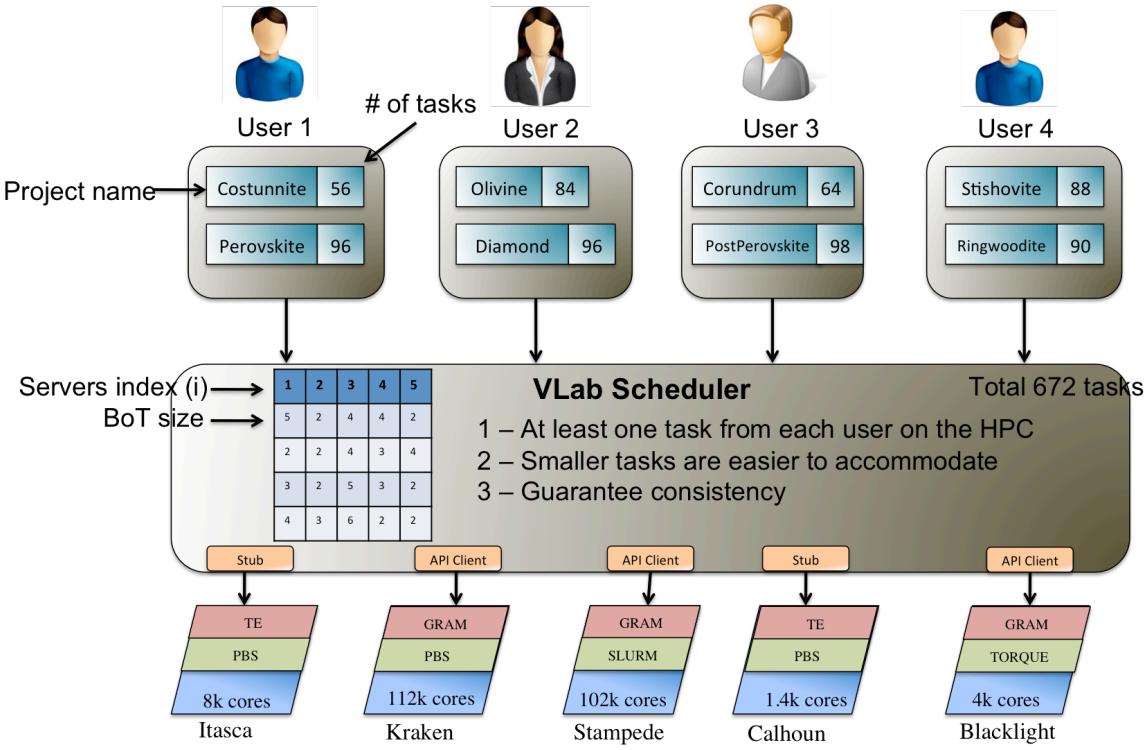


Figure 37. User case scenario of the ***VLab*** Scheduler.

5.6 Algorithms

Conventional scheduling strategies envisioned in workload cluster managers inspect job priorities at single points in time. The system generally has no advance knowledge of a up coming arrival of a high priority job and allocates the necessary resources through the suspension, checkpoint, swap or killing of lower priority jobs. This is in typically an invasive approach and can result in system thrashing and the appearance of artificially high system utilization aka overhead. In addition, today with main memory on petascale systems exceeding hundred of terabytes, the time to checkpoint or swap large portions of memory can be significant and will reduce the expectedness of the scheduling model. When an application is started on a set of nodes, all the resources of these nodes are fully dedicated to this application. In the optimal case, the resources are kept allocated until the application completes its execution cycle and there is no need for process level management. Benefits of this model include avoidance of system thrashing, increased predictability in the scheduling model, and reliable and repeatable runtimes benefiting both operational and research users.

In a scheduling strategy for tasks prepared inside an SG, suspension, checkpoint, swap or killing lower priority jobs are possible to achieve, however not desirable. The user activities in a workload grid manager, e.g., PBS, are taken in consideration for prioritize a task. As a rule of thumb in HPC resources, first only a certain number of tasks are acknowledged by the workload manager system when a user submits multiple tasks, second after many tasks are completed the priority level decreased for the next tasks, and third the system may increase the wait time on the queue based on the amount of cores requested. Some of the scheduling algorithms implemented and tested on the ***VLab*** Scheduler are: (i) first-come-first-serve with fair share over user/group and fair share by past consumption, (ii) shortest-job-first with support for large CPU/Memory request, and (iii) longest-job-first with support large CPU/Memory request.

The first-come-first-serve scheduling algorithm with fair share over user/group was developed with the aim of allowing at least one job distributed and running for every user/group in the queue. The main goal expected for this algorithm is avoid a single user to take over all resources. The ***VLab*** Scheduler makes every new user in the queue gets at least one task submitted to an HPC resource. The Scheduler monitors past consumption to guarantee fair share. Therefore, properties of tasks passed through the ***VLab*** Scheduler, e.g., CPU time, wall time, and BoT makespan are recorded.

In the shortest-job-first algorithm, with support for large CPU/Memory request, a single job with a relatively small wall time request is favored. That way, in a workflow simulation involving a large number of tasks the entire set can be placed in one single BoT. When an arbitrary large number of tasks with large amount of cores per tasks are required by a scientific workflow, this algorithm divides tasks into smaller groups and fills in BoTs half the size the resource is capable to run. That way, a resource will not receive requests exceeding its capacity and simulations are orderly distributed.

The longest-job-first algorithm with support for large CPU/Memory requests favors tasks requesting longer wall times over those requesting smaller wall times. Generally, the longest-job-first algorithm requests a significant amount of CPU and memory. That way, BoTs are filled with as many tasks as possible with the constraint of not exceeding half of each resource capability. For task requests with short wall time and small number of cores this algorithm groups tasks into BoTs using as few HPC resources as possible.

5.7 Fault Tolerance in the *VLab* Scheduler

In order to guarantee consistency and coordination of tasks distribution without wasting allocation resources, a fault tolerant service for ***VLab*** Scheduler is implemented. On ***VLab*** Scheduler, when all tasks are received from WS Project Executor, they are automatically included in a queue in the ***VLab***

Scheduler service. If the service breaks down or shuts down by mistake, all tasks in the queue can be lost and consequently tasks already submitted to HPC will be considered zombie (without a manager) consequently wasting resource. To avoid this situation, a decentralized and cooperative fault tolerance is implemented to a collection of ***VLab*** Scheduler. This fault tolerance algorithm works based on P2P coordination. According to A. Rowstron [21] the P2P coordination space provides a global virtual shared space that can be concurrently and associatively accessed by all participants. This access is independent of the actual physical or topological proximity of the objects or hosts. Figure 38 depicts the architecture difference between server-based network and peer-to-peer network.

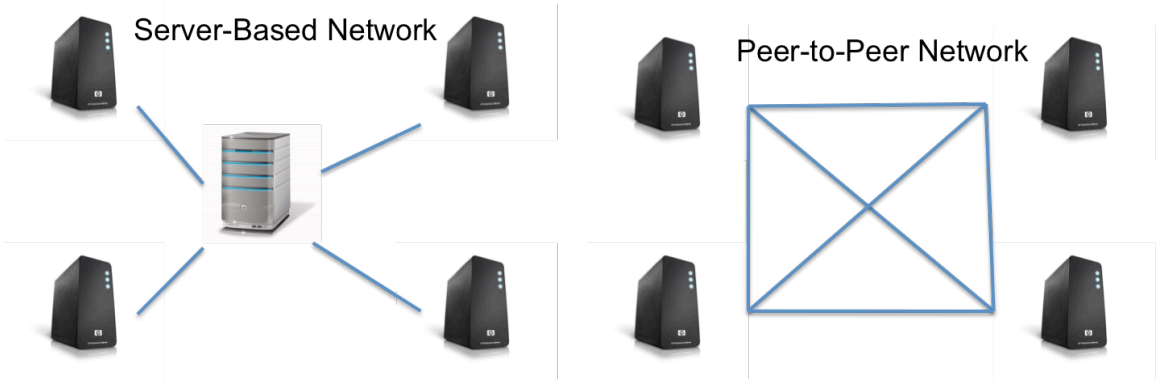


Figure 38. Server connection differences between server-based network and peer-to-peer-network.

New generation routing algorithms, commonly known as the Distributed Hash Tables (DHTs), form the basis for the organization of the P2P coordination space. A set of interconnected replicas of the WS Task Dispatcher [13], with the ***VLab*** Scheduler already included, communicates by sending notification messages to each other as in a ring network philosophy. This peer-to-peer algorithm guarantees that all tasks are managed by the system consistently. It automatically allows the ***VLab*** scheduler to undertake load balancing of all tasks submitted and avoid interruption of working service.

Chapter 6

6. Conclusion and future work

This thesis explained the design, implementation, and some features of the ***VLab*** SG. It described the web site with public resource, web portal features such as technology used (e.g. Gridsphere) and project property description entered by the user. It described workflows supported in ***VLab*** and some aspects of their implementation, the web service architecture and communication features, the database for data, receipt and metadata management, results search interface, and a task scheduler. The current system is capable of submitting workflows with large numbers of long HPC tasks to the XSEDE. A workflow for numerous and long (months of wall time) calculations of anharmonic thermodynamics properties is currently being implemented for use by the Wentzcovitch group. In this respect ***VLab*** is unique, but such sophisticated tasks are not necessarily popular, a necessary condition for a gateway to thrive.

In the near future, we envision implementation of small online applications such as those available under ***VLab*** resources. Examples are: construction of thermodynamics equilibrium phase diagrams, elasticity of aggregates, geotherm calculations, planet builder, and other applications of interest to the mineral physics community. This will bring the mineral physics community to use this infrastructure. As far as workflows are concerned, there is tremendous opportunity for development of materials science oriented workflows, such as near-edge absorption fine structure calculations (NEXAFS), core level shifts, optical spectra, infra-red spectra calculator, etc. This could bring the local experimental community to use it.

The ***VLab*** SG pioneered a new mode of performing large scale HPC/HTC calculations online. We see an ever expanding frontier for calculations of this type, with increased easiness, aggregated power, and flexibility.

Bibliography

[1] See http://www.vlab.msi.umn.edu/

[2] See http://www.msi.umn.edu

[3] See http://www.xsede.org

[4] See http://sciencegateways.org/

[5] P. Giannozzi, S. Baroni, N. Bonini, M. Calandra, et al., "Quantum ESPRESSO: a modular and open-source software project for quantum simulations of materials" J. Phys.: Condens. Matter, 21, 395502 (2009).

[6] R. Barbera, G. La Rocca, R. Rotondo, A. Falzone, P. Maggi, N. Venuti: Conjugating Science Gateways and Grid Portals into e-Collaboration environments: the Liferay and GENIUS/EnginFrame use case. http://doi.acm.org/10.1145/1838574.1838575

[7] Tan, E., Choi, E., Thoutireddy, P., Gurnis, M., Aivazis, M., "GeoFramework: Coupling multiple models of mantle convection within a computational framework", Geochem. Geophys. Geosyst., 1525-2027, http://dx.doi.org/10.1029/2005GC001155 , 10.1029/2005GC001155

[8] G. Klimeck *et al.,* "nanoHUB.org – Online Simulation and More Materials for Semiconductors and Nanoelectronics in Education and Research," *Proc. of IEEE Conf. on Nanotechnology,* pp. 17–23, 2008.

[9] See https://www.xsede.org/gateways-overview

[10] See https://www.xsede.org/web/guest/gateways-listing

[11] E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M. H. Su, K. Vahi, M. Livny, "Pegasus: Mapping Scientific Workflow onto the Grid", Across Grids Conference, Cyprus, 2004

[12] Altintas, I. and Berkley, C. and Jaeger, E. and Jones, M. and Ludascher, B. and Mock, S., “Kepler: an extensible system for design and execution of scientific workflows.”, (2004) Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on. Pages: 423--424.

[13] Michael Wilde, Mihael Hategan, Justin M. Wozniak, Ben Clifford, Daniel S. Katz, Ian FosterSwift: A language for distributed parallel scripting Parallel Computing 2011

[14] S. Marru, L. Gunathilake, R. Singh, M. Pierce, “Apache Airavata: A framework for Distributed Applications and Computational Workflows”, GCE '11 Proceedings of the 2011 ACM workshop on Gateway computing environments, ISBN: 978-1-4503-1123-6

[15] P. R. C. da Silveira, M. N. Valdéz, R. M. Wenzcovitch, M. Pierce, C. R. S. da Silva, D. A. Yuen, “Virtual Laboratory for Planetary Materials (***VLab***): An Updated Overview of System Service Architecture”, TG '11 Proceedings of the 2011 TeraGrid Conference: Extreme Digital Discovery, Article No. 33.

[16] D. Gannon, B. Plale, M. Christie, L. Fang, Y. Huang, S. Jensen, G. Kandaswamy, S. Marru, S. F. L. Pallickara, S. Shirasuna, Y. Simmhan, A. Slominski, andY. M. Sun, ”Service oriented architectures for science gateways on grid systems”, Lecture Notes in Computer Science 3826, 21 (2005).

[17] Ledford, J. L. and M. E. Tyler (2007), Google Analytics 2.0, Wiley Publishing, Inc.

[18] See http://www.pbsworks.com/

[19] See IBM WebSphere Portal, http://publib.boulder.ibm.com/infocenter/wpdoc/v510/index.jsp

[20] Novotny, Jason, Michael Russell, and Oliver Wehrens. "GridSphere: an advanced portal framework." Euromicro Conference, 2004. Proceedings. 30th. IEEE, 2004.

[21] Wiggers, Chanoch, et al. Professional Apache Tomcat. Wiley. com, 2004.

[22] Volker Heine, The Pseudopotential Concept, In: Henry Ehrenreich, Frederick Seitz and David Turnbull, Editor(s), Solid State Physics, Academic Press, 1970, Volume 24, Pages 1-36, ISSN 0081-1947, ISBN 9780126077247, 10.1016/S0081-1947(08)60069-7.

[23] Williams, Thomas, and Lars Hecking. "Gnuplot." (2003).

[24] da Silveira, P.R.C., da Silva, C.R.S., Wentzcovitch, R.M. Metadata management for distributed first principles calculations in ***VLab*** — a collaborative grid/portal system for materials computation, 2008, Computer Physics Communications, Volume 178, Issue 3.

[25] Graham, Steve, et al. Building Web services with Java: making sense of XML, SOAP, WSDL, and UDDI. SAMS publishing, 2004.

[26] da Silva, C.R.S., da Silveira, P.R.C, Karki, B.B., Wentzcovitch, R. M., Jensen, P. A., Bollig, E. F., Pierce, M., Erlebacher, G., Yuen, D. A., Virtual laboratory for planetary materials: System service architecture overview, Physics of The Earth and Planetary Interiors, 2007, Volume 163, Issues 1-4, Computational Challenges in the Earth Sciences.

[27] Van Antwerp, Matthew, and Greg Madey. "Advances in the sourceforge research data archive." Workshop on Public Data about Software Development (WoPDaSD) at The 4th International Conference on Open Source Systems, Milan, Italy. 2008.

[28] Massol, Vincent, and Timothy M. O'Brien. Maven: a developer's notebook. O'Reilly, 2009.

[29] Jasnowski, Mike. Java, XML, and Web services bible. John Wiley & Sons, Inc., 2002.

[30] Automatic capture and efficient storage of e-Science experiment provenance. Concurrency Computat.: Pract. Exper. 2008; 20:419–429

[31] See http://www.egi.eu/

[32] Altintas, I. and Berkley, C. and Jaeger, E. and Jones, M. and Ludascher, B. and Mock, S., “Kepler: an extensible system for design and execution of scientific workflows.”, (2004) Scientific and Statistical Database Management, 2004. Proceedings. 16th International Conference on. Pages: 423--424.

[33] Wieczorek, Marek, Radu Prodan, and Thomas Fahringer. "Scheduling of scientific workflows in the ASKALON grid environment." ACM SIGMOD Record 34.3 (2005): 56-62.

[34] Núñez Valdez M., K. Umemoto, and R. M. Wentzcovitch," Elasticity of Diamond at High Pressures and Temperatures". Submitted to Applied Physics Letters (2012) Preprint: http://arxiv.org/abs/1205.4227.

[35] Vinet, P., Ferrante, J., Smith, J.R., Ross, J.H., 1986. J. Phys. C 19, 467

[36] G.F. Davies, Quasi-harmonic finite strain equations of state of solids, Journal of Physics and Chemistry of Solids, Volume 34, Issue 8, August 1973, Pages 1417-1429, ISSN 0022-3697, 10.1016/S0022-3697 (73) 80042-3.

[37] L. Meye, D. Schefner, J. Vockler, M. Mattosos, M. Wilde, and I. Foster, “An Opportunistic Algorithm for Scheduling Workflows on Grids”, VECPAR'06 Proceedings of the 7th international conference on High performance computing for computational science.

[38] Y. Yan, B. Chapman, “Comparative Study of Distributed Resource Management Systems – SGE, LSF, PBS Pro, and LoadLeveler”.

[39] T. Mukherjee, Q. Tang, C.Ziesman, S.K. Gupta, and P. Cayton. Software architecture for dynamic thermal management in datacenters. In *Proc. of Intl. Conf. on Communications Systems Software and Middleware (COMSWARE)*, January 2007.

[40] Jang-uk In, Paul Avery, "Policy Based Scheduling for Simple Quality of Service in Grid Computing", 18"' International Parallel and Distributed Processing Symposium (IPDPS 2004), Santa Fe, New Mexico, USA. IEEE Computer Society, 2004.

[41] Sun Microsystems, Inc. Sun Grid Engine 5.3 Administration and User's Guide [M].http://gridengine.sunsource.net/project/gridengine-download/SGE53AdminUserDoc.pdf, April, 2022.

[42] Chlumský, V., Klusáček, D., Ruda, M.: The extension of TORQUE scheduler allowing the use of planning and optimization algorithms in Grids. Computer Science 13(2), 5–19 (2012).

[43] Yoo, Andy B., Morris A. Jette, and Mark Grondona. "SLURM: Simple linux utility for resource management." Job Scheduling Strategies for Parallel Processing. Springer Berlin Heidelberg, 2003.

[44] Brelsford, David Paul, and Joseph Francis Skovira. "External job scheduling within a distributed processing system having a local job control system." U.S. Patent No. 6,694,345. 17 Feb. 2004.

[45] Chakravorty, Sayantan, et al. "HPC-Colony: services and interfaces for very large systems." ACM SIGOPS Operating Systems Review 40.2 (2006): 43-49.

[46] Bird, Ian, Kors Bos, N. Brook, D. Duellmann, C. Eck, I. Fisk, D. Foster et al. "LHC computing Grid." *EGEE, Mar* 18 (2008).

[47] Pordes, Ruth, Don Petravick, Bill Kramer, Doug Olson, Miron Livny, Alain Roy, Paul Avery et al. "The open science grid." In *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012057. IOP Publishing, 2007.

[48] Weitzel, Derek, Dan Fraser, Brian Bockelman, and David Swanson. "Campus Grids: Bringing Additional Computational Resources to HEP Researchers." In *Journal of Physics: Conference Series*, vol. 396, no. 3, p. 032116. IOP Publishing, 2012.

[49] Luckow, André, Lukasz Lacinski, and Shantenu Jha. "SAGA BigJob: An extensible and interoperable pilot-job abstraction for distributed applications and systems." In *Cluster, Cloud and Grid Computing (CCGrid), 2010 10th IEEE/ACM International Conference on*, pp. 135-144. IEEE, 2010.

[50] H. Jiang, K. Huang, H. Chang2, D. Gu, P. Shih, “Scheduling Concurrent Workflows in HPC Cloud through Exploiting Schedule Gaps”, ICA3PP'11 Proceedings of the 11th international conference on Algorithms and architectures for parallel processing - Volume Part I, Pages 282-293.

[51] Juve, G. Deelman, E.; Vahi, K.; Mehta, G.; Berriman, B.; Berman, B.P. ; Maechling, P. , “Scientific Workflow Applications on Amazon EC2”, E-Science Workshops, 2009 5th IEEE International Conference.

[52] Dorigo, Marco, ed. Ant Colony Optimization and Swarm Intelligence: 5th International Workshop, ANTS 2006, Brussels, Belgium, September 4-7, 2006, Proceedings. Vol. 4150. Springer-Verlag New York Incorporated, 2006.

[53] T.Casavant, and Kuhl, “A Taxonomy of Scheduling in General – purpose Distributed Computing Systems,” IEEE Trans. on Software Engineering, vol. 14, no. 2, pp. 141 – 154, Feb. 1988.

[54] Nguyen The Loc, Said Elnaffar, “A Dynamic Scheduling Algorithm for Divisible Loads in Grid Environments,” Journal of Communications, vol. 2, no. 4, June 2007.

[55] Ajith Abraham, Hongbo Liu,Weishi Zhang , and Tae-Gyu Chang, “Scheduling Jobs on Computational Grids Using Particle Swarm Algorithm,” 10th International Conference on Knowledge-Based Intelligent Information and Engineering Systems, UK,2006, pp.500-507.

[56] Keahey, Kate, and Welch Von. "Fine-grain authorization for resource management in the grid environment." Grid Computing—GRID 2002. Springer Berlin Heidelberg, 2002. 199-206.

Appendix A

Vita

Pedro R. C. da Silveira is a native of Brazil. He completed his Bachelor degree in Computer Science from Universidade de Santo Amaro, São Paulo, Brazil. He is a Research Assistant in the Department of Chemical Engineering and Material Science at University of Minnesota working under Professor Renata Wentzcovitch's supervision. He is the developer of the ***VLab*** cyberinfrastructure for distributed first principles calculations and consults in high performance computing with the Wentzcovitch group. He is a Campus Champion for XSEDE helping researches at the University of Minnesota integrate computer XSEDE resources in their work. He will continue developing ***VLab*** as a consultant after completion of his PhD degree. He currently works as software engineer for Digital River, a Minnetonka based Internet Company specialized in e-business, e-commerce, and credit card services operating globally.