

Towards A Grid Enabled System for Multicomponent Materials Design *

Keita Teranishi

Padma Raghavan

Zi-Kui Liu

Abstract

We are developing a portal for multicomponent materials design using grid-enabled large-scale simulations. In this paper, we report on our services based application architecture which allows integration of simulation software with grid services to provide a web-based computational laboratory for the modeling of Al-Cu-Mg-Si alloys. We examine user requirements and describe the design of our framework. Our architecture is implemented using existing middleware such as the Globus and Java CoG toolkits. An interesting feature of our design is the separation of the high-level specification of the materials modeling system from the implementation through the use a markup language such as XML. We use markup languages with domain-specific extensions to specify (in architecture-independent form) rules and constraints that allow meaningful composition of simulation tasks and experimentally determined material properties. In addition, we use them to specify application code interfaces so that our simulation server can be dynamically reconfigured to include new software and constraints.

1 Introduction

In the last few decades there have been significant advances in computational materials science ranging from first-principles calculations of atomic structures to continuum modeling of complex microstructures. We seek to use these computational techniques with recent developments in grid technology to provide a portal for materials design automation. More specifically, our project concerns the development of a system to automate investigations that predict macroscopic properties (such as the mechanical response) of Al-Cu-Mg-Si alloys by combining a four stage multi-scale, multi-physics computational process with empirically

obtained material properties [10]. The four main steps include (i) ab-initio calculations at the atomistic level, (ii) data optimization to determine thermodynamic properties, (iii) generation of microstructures using phase-field simulations, and (iv) performing finite-element analysis on the simulated microstructures to determine properties such as the mechanical response. This report concerns the design of MATCASE, our multicomponent materials modeling portal using domain-specific simulation codes, existing computational grid infrastructure such as the Globus Toolkit, and tools for web-based grid access such as the Java Commodity Grid Toolkit.

The design of our system must necessarily be extensible to allow the incorporation of new developments in materials science. To a large extent, the design of our system is complicated by the fact that algorithmic aspects of four main stages are still under development. Additionally, there are many open questions, related to the composition of results from each of the four steps and the modeling capabilities of each individual step. Many of them center around the modeling and verification of properties of metastable phases which affect the microstructures and can greatly influence macrostructural properties under different heating and cooling regimes. To meet these requirements from the scientific community, the server needs a sophisticated control flow to adaptively reconfigure simulations based on rules and heuristics and interactions with the user.

The computational demands of multicomponent materials design are in large-part the motivation for using a grid-enabled portal. The computational part is not in the form of a monolithic code; instead, it is a collection of codes with varying computational costs and memory needs. For example, two of the steps need to utilize SPMD parallelism (using MPI) to be feasible. Even though the computation in these steps is substantial, the results generated are sufficiently compact. This makes the linkage between successive steps through intermediate database more natural. In this model, each step could execute on disjoint sets of processors of suitable size. The coupling between steps should satisfy precedence requirements but it need not be direct nor synchronous. Processors for a given step proceed with the computation after asynchronously accessing the database updated by an earlier step. In the long term,

*Departments of Computer Science and Engineering and Material Science and Engineering, The Pennsylvania State University, University Park, PA 16802. E-mail:{teranish,raghavan}@cse.psu.edu and liu@matse.psu.edu. This work was supported in part by the National Science Foundation through the Information Technology Research Grant DMR-0205232.

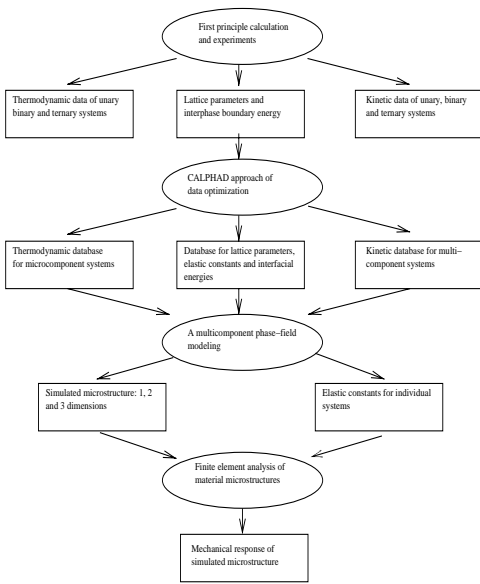


Figure 1. The main computational steps in multicomponent materials design.

the goal is to provide a portal which can cater to concurrent exploration of the materials design space by potentially dozens of clients. This would not be possible without the use of a grid-enabled simulation handler to effectively utilize resources over a wide-area network.

We develop a server that can be reconfigured or extended by using a hierarchy of specifications in an extended markup-language. Our approach also decouples the software components for computation (on the grid) from those for presentation, i.e., interaction with the user. In Section 2, we discuss the four step computational modeling of multicomponent materials with a simple example of Al-Cu modeling. In Section 3, we provide an overview of the basic architecture of our system. In Section 4, we describe in greater detail the main functional units of the server and we provide concluding remarks in Section 5.

2 Multicomponent Materials Modeling

Our multicomponent materials design framework for Al-Cu-Mg-Si systems involves four major computational steps as shown in Figure 1. We now describe these in more detail using as an example, the methodology for predicting the mechanical response (stress-strain) curves for an Al-Cu binary system with face centered cubic (fcc) and Θ' (metastable) phases.

First-principles calculations. The Vienna ab-initio simulation package (VASP) [4] is used to compute various enthalpy values. Starting with initial atomic positions

of the Θ' phase (Al₂Cu) and using density function theory, the minimal energy configuration is obtained along with the value of the total energy of the Θ' phase at 0K. The energy of formation of the Θ' phase is calculated by the energy difference between the Θ' phase and of pure Al and Cu. It is assumed that this energy of formation is independent of temperature and equivalent to the enthalpy of formation. Additionally, cluster expansion methodology is used in combination with the Monte Carlo simulation to calculate the enthalpy of mixing in the fcc phase (ΔH_{mix}). These computations are typically expensive and can require several hours to days on modest clusters. Furthermore, a process may fail to converge and may produce degenerate solutions. Domain-specific rules, for example, comparisons with values from other simulations and/or heuristics based on the difference between initial and relaxed structures can be used to flag potential problems.

Thermodynamic calculations. This step uses CALPHAD [3] to determine the Gibbs energy functions of the phases (fcc and Θ'), phase equilibria and driving forces under various conditions. Input values include enthalpy values calculated in the first step and experimentally data such as the specific heat. CALPHAD methodology is also used to develop other databases such as atomic mobility and lattice parameters. This step is relatively inexpensive; however, it has a complex updating problem associated with it. For example, if unary system properties are changed, then binary, ternary, and quaternary systems containing these unary systems have to be re-computed. A serious issue is that of maintaining data integrity when such updates cascade from different simulations by different clients. We initially plan to manage this issue by allowing only a few periodic updates to the main database based on careful verification of computed results.

Phase-field simulations. This step involves the solution of two sets of dynamic equations: the Cahn-Hilliard equations for the composition fields, and the time-dependent Ginzburg-Landau equations for the phase-field parameters. Both equations are driven by the reduction of the system Gibbs energy. This step needs the Gibbs energy, atomic mobility, kinetic coefficients related to the interface mobility, interfacial energy, and gradient energy. The results from phase-field simulations are represented by the values of compositions, phase-field parameters, and other properties of interest on each grid point. These calculations are extremely compute-intensive especially for three dimensional models for large numbers of time-steps and grid points.

Finite element analysis of microstructures. The Object Oriented Finite-Element analysis package [9], is used to perform finite-element analysis on the simulated microstructures to determine material properties as a function of time. Although current versions of OOF execute sequentially, future versions will typically include a compute-

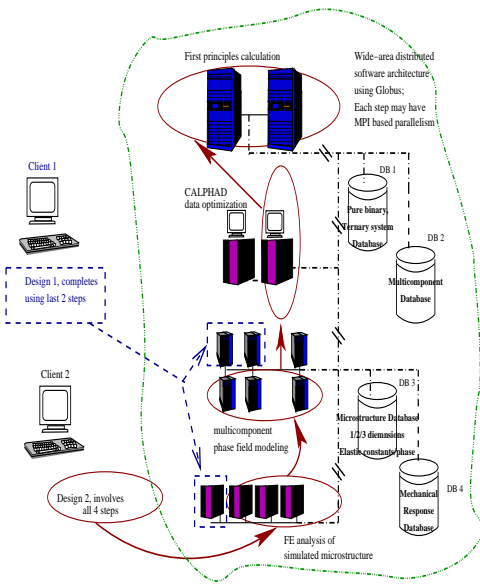


Figure 2. A simplified view of a grid-enabled system for materials design.

intensive back-end that must run on message passing multi-processors.

Figure 2 shows a conceptual view of our material modeling system with computational tasks that are scheduled on grid resources using grid services. With respect to our four step process, computations start and complete at step 4 if the input database for this step contains all necessary microstructure data. Otherwise, step 3 is initiated; potentially, step 3 could proceed if the required data is available in its input database, and provide its results in its output database. The latter can then be used to complete step 4 and hence this instance of the application. Alternately, a different instance may require tracing requirements all the way back to step 1, and proceeding through all steps (in turn) to completion.

3 A Framework with Three Sets of Defined Services

At a high level, our software architecture is a client-server system. The clients initiate particular designs and the server satisfies the design requirements by ultimately scheduling appropriate tasks on grid resources. Each client instance is associated with a request handler on the server side. The design of the request handler is quite complicated because there are several sets of requirements that must be satisfied to allow effective design space exploration.

The request handler should meet the requirements of the materials science community while allowing a portable and extensible system that can utilize resources on a computa-

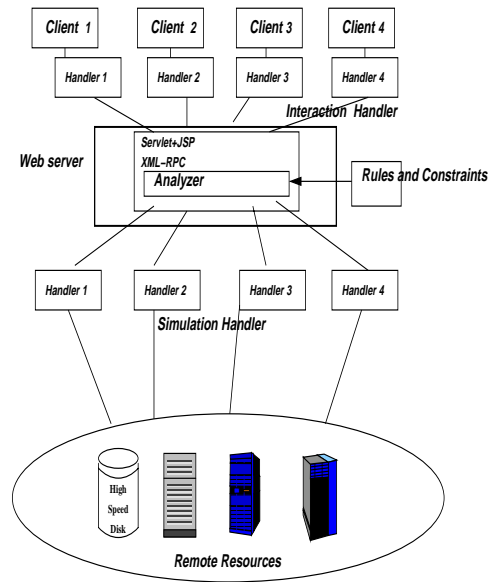


Figure 3. A high-level view of the MATCASE web portal.

tional grid. The design requirements come from three different sources. First, there is a clear need to interact with the user to set up simulation parameters and modify them as the computations proceed. Second, a design investigation will need coupling of tasks corresponding to one or more of the four main steps. The computational demands imposed by application codes for each of the four stages are varied. Furthermore, there can be multiple implementations of the same step with different costs and limitations and interface descriptions. Third, the materials science community provides only a partial set of rules and heuristics for determining meaningful explorations of the multicomponent materials design space. These rules will clearly influence both the interaction with the user and the specifics of the tasks to be used for a given design instance.

We define our overall request handler in functional terms with a simple pattern of interaction and control flow between its main components. We view the system as providing three sets of services.

- The interaction service which allows an initial problem specification and further interaction to define constraints, refine model, present and evaluate results.
- The simulation service which is responsible for remote execution of tasks, their interaction and data management.
- The analysis service which elaborates the initial problem specification using rules and heuristics from the materials science community into an instance of the

four-stage simulation process. It continues to control the design status and specifies the actions of the simulation and interaction handlers.

Each of these sets of services is executed by a corresponding component; we henceforth refer to these as interaction and simulation handlers, and the analyzer. The analyzer will interface with both the interaction and simulation handlers as shown in Figure 3.

The interaction handler is first used to generate an initial screen for obtaining the initial version of the design problem from the client. This information is passed on the analyzer which will try to refine the problem to create a design instance within the capabilities of the system; these specifications will then be sent to the interaction handler which will manage the user view. A typical usage pattern consists of translating functional requirements into design parameters, and then into variables in the process domain. This process will depend to a large extent on rules and constraints defining meaningful design investigations. The need for extensibility cannot be overemphasized because the domain-specific knowledge and rules are incomplete at best and are expected to change rapidly based on the experience of users with the system. As a first step towards providing this extensibility, we plan to model all the rules and heuristics as an XML database with keywords and associated semantics (extensions) suitable for our application. The analyzer will use these specifications to refine the design details; these refinements will be transmitted to and from the user through the interaction handler. At the same time, the analyzer will provide a specification of tasks to the simulation handler. The latter will deploy and manage these tasks on the computational grid. The analyzer maintains a record of the status of each clients design. It attempts to ensure a correct design using rule-based analysis to control and adapt the user interaction data and the intermediate steps tasked by the simulation handler. Figure 4 shows our grid-enabled portal in terms of its three main service components.

4 Implementation Details

We now discuss some of the implementation details of the three main components with particular emphasis on the grid-enabled simulation handler component. To implement our server, we use Java Server Pages (JSP) [14], Java servlet [15], DOM [20] enabled Java-XML parsing, Java commodity Grid (CoG) kit [17], and XML-RPC [2]. We currently use Globus-2.4 for controlling access to remote resources and GRAM to execute each of the four steps of the application. We also use GridFTP for accessing simulation data using GridFTP.

Our implementation of the user interface is somewhat similar to the Grid Portal Development Kit [11] devel-

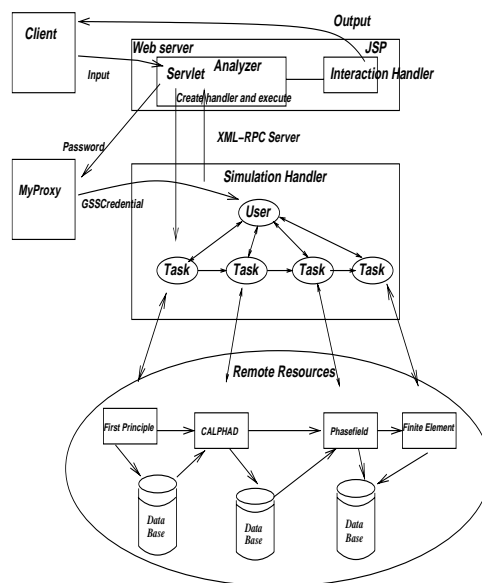


Figure 4. Our framework with three main system components.

oped for the previous version of Globus. Our implementation also use the idea of Model-View-Controller (MVC) [6] where the software components for the presentation and underlying computation are separated. Observe that in our system, the simulation handler manages access to the grid and the results are directed through the analyzer to the interaction handler which then controls the user view through Java Server Pages. The intermediate analyzer is needed for managing the complexity of the materials design process while separating computing from the presentation of results. The intermediate analyzer is also essential for extensibility with respect to the rules governing the multicomponent materials design process.

In developing the interaction handler, we have to take into account the fact that our application will require processing a large number of parameters, types of data, and types of design options. Many of the choices of such values are interdependent and meaningful only within certain subspaces of the design space. Hence, a simple static interface will not be adequate. Furthermore, the interface content can depend on the nature of the intermediate results of any of the four simulation steps. We therefore start with a simple static page and then generate interface pages automatically using state descriptions of the design stored in an XML data base. The description can be modified as needed by the analyzer to incorporate information from the computational steps.

Our scheme starts with the user input entered into a comprehensive form in a static web-page. Information on this form can be mapped to an XML description and this map-

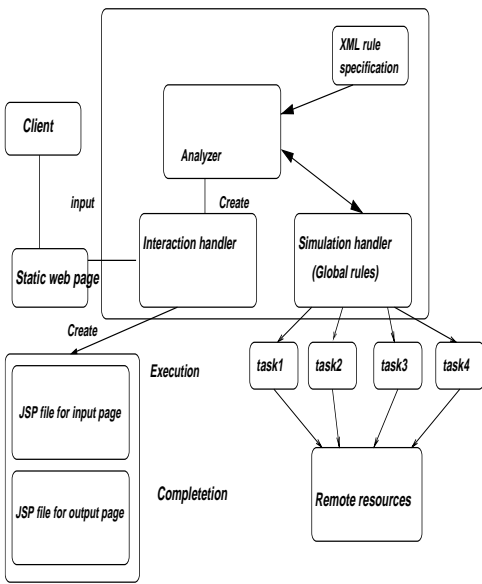


Figure 5. Execution sequence to set up simulations and user interactions.

ping is performed by the interaction handler. This XML specification is then used by the analyzer to specify modifications and refinements in XML form. The latter can then be used to generate interface pages dynamically to interact with the user. At the same time, the analyzer can use the XML form of the specification to generate an instance of the simulation handler with suitable task specifications. Currently, we use a DOM [20] enabled Java-XML parser for converting XML script into simulation handler code and JSP scripts in the interaction handler. The simulation handler code is compiled and registered to the server. We attach XML-RPC [2] to the servlet in order to handle the registry and execution of new simulation handlers without the restart needed to add a new class. This process of automatically generating the content of the user interface pages and the task specifications for the simulation server is shown in Figure 5.

The simulation handler is a class object created for every client and serves as the back-end of the server to marshal and execute tasks on remote resources. This component contains several task modules to execute each computational step. A task module manages submitting jobs, accessing files and setting up the RSL script to run a batch job. Task modules encapsulate some basic Globus function calls such as GRAM, GridFTP and MDS in order to shield users from manipulating the CoG kit directly. Each task module is activated by a user module that stores the profile of a user and the associated job history. An important entity of the user module concerns storing credentials obtained by the

MyProxy [12] server. In addition to job control, the simulation handler also manages the sending and receiving of input and output. The embedded XML-RPC server [2] in the webserver is used as a registry of each simulation handler and it also allows its automatic generation using a scripting language.

We use an XML script that describes the simulation process (by the analyzer) to create and configure the simulation handler. However, it is often the case that the result of a computational step is needed to configure the application codes for the subsequent step. Therefore, providing only a static interface to connect successive computational steps will not meet the application needs. For example, the result from CALPHAD can lead to a new definition of the Gibbs energy function. This function will be needed for the phase-field simulation and must be dynamically linked in. In general, intermediate results will often result in a change of parameters and software functions for subsequent steps. We therefore allow the dynamic reconfiguration of the simulation sequence. This cannot be implemented by modifying the simulation handler because it has been already created with the initial input from user. Instead, the modification is made on the remote resources. We provide an intermediate task between two computational steps to generate new input files and function definitions such as those for the Gibbs energy function for the phase field simulation program. The interesting point is that the intermediate task works on the remote computers invoked by the main task on the web server. This execution detail is shown in Figure 6. The Open Grid Services Architecture (OGSA) implemented with Globus-3.0 [13, 5] advocates a similar approach to encapsulate remote resources and services to create a distributed object.

5 Conclusions

Our project concerns generating a domain-specific grid-enabled portal. We will test and report on preliminary results on the performance of our grid-enabled simulation server. We also plan to report on our experience in migrating the implementation to OGSA. We anticipate that our rule-based extensible design will allow broader application within the materials science community. We also conjecture that our system has the potential to be easily adapted for use in another application domain through the high-level specification of a different set of rules and constraints.

Our project is related to several projects that attempt to construct grid-enabled portal for computational science applications. Examples include the Astrophysics Simulation Collaboratory (ASC) [19] and the Active Thermochemical Table Framework [18]. The ASC implements reconfigurable remote service in conjunction with the Cactus [1] framework. The system compiles the simulation program

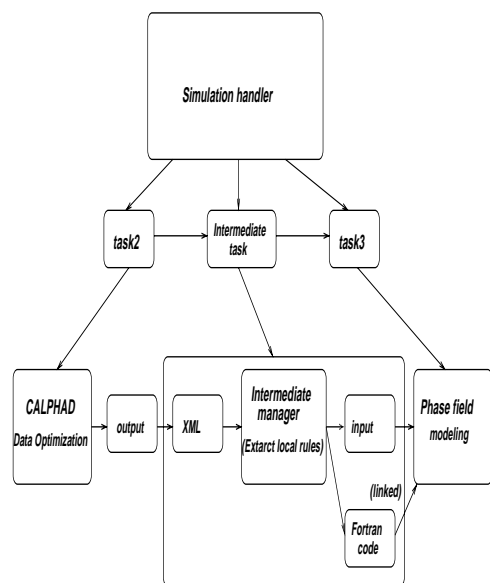


Figure 6. Execution sequence in the simulation handler to process inter-task local requirements.

remotely to create a new simulation application within Cactus; it differs from our application in that it concerns the execution and interface of a monolithic software system. The Active Thermochemical Table Framework describes the use of various web services that foresee the emergence of OGSA. It is related to our design of the interaction handler. Our work is also influenced by the grid portal development kit [11] and the Ninf-Portal [16] for the automatic generation of the user interface. There is also some relation to the XCAT project [7, 8] which concerns the construction of a general purpose grid application.

6 Acknowledgments

The authors would like to thank I. Foster and G. von Laszewski at the Argonne National Laboratory for many useful comments.

References

[1] G. Allen, W. Benger, T. Goodale, H. Hege, G. Lanfermann, A. Merzky, T. Radke, and E. Seidel. The Cactus Code: A problem solving environment for the grid. In *Proc. High Performance Distributed Computing (HPDC-2000)*, pages 253–260, 2000.

[2] Apache <Web Service> Project. XML-RPC, November 2003.

[3] CALPHAD. Computer coupling of phase diagrams and thermochemistry. <http://www.calphad.org/>, 2003.

[4] T. Demuth and F. Mittendorfer. Vienna ab-initio simulation package, 2003.

[5] I. Foster, C. Kesselman, J. Nick, S. Tuecke, and O. G. S. I. W. Group. The physiology of the grid: An Open Grid Services Architecture for distributed systems integration. In *Global Grid Forum*, June 22 2002.

[6] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Patterns: Elements of Reusable Object Oriented Software*. Addison-Wesley, 1997.

[7] D. Ganon, R. Bramley, G. Foxa, S. Smallen, A. Rossi, R. Ananthakrisnan, Y. Bertrand, A. Slominski, Y. Ma, C. Olariu, and N. Rey-Cenvaz. Programming the grid: Distributed software components, p2p, and grid web services for scientific applications. In *Grid 2001*, 2001.

[8] S. Krisman, R. Bramley, D. Gannon, M. Govindaraju, R. Indurkar, A. Slominski, and B. Temko. The XCAT science portal. In *Supercomputing 2001*, 2001.

[9] S. Langer, C. Carter, E. Fuller, and A. Roosen. Oof: Object-oriented finite-element analysis of real material microstructures. <http://www.ctcms.nist.gov/oof/>, 2003.

[10] Z. K. Liu, L. Q. Chen, Q. Du, S. Langer, P. Raghavan, J. Sofo, and C. Wolverton. Computational tools for multicomponent materials design. <http://www.matcase.psu.edu>, 2002-2007.

[11] J. Novotny. The grid portal development kit. *Concurrency and Computation: Practice and Experience*, 14(13–15):1129–1144, 2002.

[12] J. Novotny, S. Tuecke, and V. Welch. An online credential repository for the grid: MyProxy, 2001.

[13] T. Sandholm and J. Gawor. Globus toolkit 3 core - A grid service container framework. <http://www-unix.globus.org/core/>, 2003.

[14] Sun Microsystems Inc. Java server pages, November 2003.

[15] Sun Microsystems Inc. Java servlet technology, November 2003.

[16] T. Suzumura, H. Nakada, M. Sato, S. Matsuoka, Y. Tanaka, and S. Sekiguchi. The ninf portal An automatic generation tool for grid portals. In *Joint ACM Java Grande - ISCOPE 2002 Conference*, Seattle, WA, 3–5, November 2002.

[17] G. von Laszewski, I. Foster, J. Gawor, and P. Lane. CoG kits: A bridge between commodity distributed computing and high-performance grids. *Concurrency: Practice and Experience*, 13:643–662, 2001.

[18] G. von Laszewski, B. Ruscic, P. Wagstrom, S. Krishnan, K. Amin, S. Nijssure, R. Pinzon, M. L. Morton, S. Bittner, M. Minkoff, A. Wagner, and J. C. Hewson. A grid service based active thermochemical table framework. In *Third International Workshop on Grid Computing*, Lecture Notes in Computer Science, Baltimore, MD, 18 November 2002.

[19] G. von Laszewski, M. Russell, I. Foster, J. Shalf, G. Allen, G. Daues, J. Novotny, and E. Seidel. Community software development with the astrophysics simulation collaboratory. *Concurrency and Computation: Practice and Experience*, to appear.

[20] World Wide Web Consortium. Document object model. <http://www.w3.org/DOM/>, November 2003.