December 2012

Volume 3 Issue 2

# JoCsE

## Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

**Volume 3, Issue 2** **December 2012**

# JOCSE

Journal Of Computational Science Education



The Journal Of Computational Science Education (JOCSE), ISSN 2153-4136, is published quarterly in online form, with more frequent releases if submission quantity warrants, and is a supported publication of the Shodor Education Foundation Incorporated. Materials accepted by JOCSE will be hosted on the JOCSE website, and will be catalogued by the Computational Science Education Reference Desk (CSERD) for inclusion in the National Science Digital Library (NSDL).

**Subscription:** JOCSE is a freely available online peer-reviewed publication which can be accessed at `http://jocse.org`.

# Contents

# Introduction to Volume 3 Issue 2

Steven I. Gordon
Editor
Ohio Supercomputer Center
Columbus, OH
sgordon@osc.edu

## Forward

This issue of JOCSE provides a diverse set of approaches to computational science education along with several very sophisticated student projects. Kim *et al* describe a comprehensive framework for learning about sustainable design through a series of steps to design a bicycle pedal and to evaluate the impacts of production on several sustainability measures. They have developed a Sustainable Product Development Collaboratory that allows students to make changes in the product design and to then see its impacts on energy use and the carbon footprint of the production process.

Gadepally et al use a combination of an analogous physical system and simulation environment to demonstrate the principles of building autonomous vehicles. Students in a summer workshop use a programmable Roomba and program it to avoid obstacles on a physical course setup to represent city streets. This is augmented by a related simulation environment where various commands can be tested. In that process, they learn the mathematical concepts, programming tools, and modeling processes that are used by engineers creating and testing such systems.

Toth and Franco present a virtual lab focused on the screening of drugs as part of a medicinal chemistry course. Students are introduced to the process of screening drugs using a supercomputer program to identify inhibitors for a number of diseases. Students used a typical workflow that included identifying a protein that has been found to be a good drug target, discovering whether its 3D structure has been solved, using a docking program to screen potential compounds, and generating a visualization of the final docking results.

The student papers in this issue demonstrate the diverse skills that students have acquired through internship experiences. Those include the generation of metadata that allows annotation of scientific datasets, the porting and testing of a parallel version of a computational chemistry code, and testing the speedup of codes using GPUs. Each of those projects has made a significant impact on the academic careers and future career goals of the participating students.

We hope the issue provides insights that you can use in your classrooms. Make sure you also encourage others to start reading and contributing to JOCSE so that we broadly share our experiences in computational science education.

# Cyber Collaboratory-based Sustainable Design Education: A Pedagogical Framework

Kyoung-Yun Kim
Department of Industrial and Systems Engineering
Wayne State University
Detroit, MI, USA
+1-313-577-4396
kykim@eng.wayne.edu

Karl R. Haapala
School of Mechanical, Industrial, and Manufacturing Engineering
Oregon State University
Corvallis, OR, USA
+1-541-737-3122
haapalak@engr.orst.edu

Gül E. Okudan Kremer
School of Engineering Design & Department of Industrial and Manufacturing Engineering
The Pennsylvania State University
University Park, PA, USA
+1-814-863-1530
gek3@engr.psu.edu

Michael K. Barbour
College of Education
Wayne State University
Detroit, MI, USA
+1-313-577-1693
mkbarbour@gmail.com

## ABSTRACT
Educators from across the educational spectrum are faced with challenges in delivering curricula that address sustainability issues. This article introduces a cyber-based interactive e-learning platform, entitled the *Sustainable Product Development Collaboratory*, which is focused on addressing this need. This collaboratory aims to educate a wide spectrum of learners in the concepts of sustainable design and manufacturing by demonstrating the effects of product design on supply chain costs and environmental impacts. In this paper, we discuss the overall conceptual framework of this collaboratory along with pedagogical and instructional methodologies related to collaboratory-based sustainable design education. Finally, a sample learning module is presented along with methods for assessment of student learning and experiences with the collaboratory.

## Keywords
Sustainable design education; sustainable product development collaboratory; constructivist learning theory; manufacturing analysis

## 1. INTRODUCTION
This paper introduces an NSF CI-TEAM Demonstration Project, entitled *A Sustainable Product Development Collaboratory*, which aims to develop and test a collaborative e-learning laboratory for sustainable design and manufacturing. This article discusses the collaboratory framework development and a sample learning module from the project.

Due to challenges of existing science and engineering curricula in addressing technical solutions from a holistic perspective that considers economic, environmental, and social aspects (e.g., availability of instructional materials with the requisite multidisciplinary focus), engineers within modern manufacturing companies often undertake *ad hoc* approaches to sustainable product and process development; often without proper tools or training to do so. One other contributing factor challenging the proliferation of sustainable science and engineering in industry is the focus on recruiting new graduates who demonstrate the potential to make an immediate contribution to technical corporate goals based on their experience [12, 24, 25]. Such practices do not necessarily promote a preference for individuals with a broader knowledge set blending two or more disciplines, a need for adequately addressing sustainability goals.

Researchers and practitioners alike recognize that a vast majority of product cost, quality, and overall sustainability is decided during early design. Despite this fact, sustainable design and manufacturing education remains in its infancy, although Allen *et al*. [1] described the significant, emerging levels of "grassroots" activities for sustainable design and manufacturing. At the same time, an NSF MT21 Study [19] highlighted the need to improve K-12 student interest in STEM (Science, Technology, Engineering, and Mathematics) disciplines, which is in a "State of Emergency." By coupling traditional engineering skills with a broader sustainability perspective, it is posited that the next generation will be more effectively attracted to careers in engineering.

The collaboratory developed as part of this project will provide a much needed cyber-based tool in support of K-12 online learning. In the United States, the first K-12 schools to begin using online learning included a private school and several public school districts in California, in the early 1990s [4]. This adoption was followed by the introduction of statewide and intra-state virtual schools in Utah, Florida, and New England in the middle of the 1990s [3, 11]. Watson *et al*. [28] reported that online learning activity is surging in all 50 states and the District of Columbia today. During the 2000-01 school year, Clark [10] estimated that there were between 40,000 and 50,000 K-12 students enrolled in one or more distance education courses. Estimates for the 2010-11 school year placed K-12 online learning enrollment at around

4,000,000 students [2]. In 2006, Michigan became the first state to require that all students complete some form of online learning in order to graduate from high school (other states, such as New Mexico, Alabama, Florida, and Idaho, as well as a number of individual school districts elsewhere, have followed Michigan's lead). Some experts have even predicted the majority of K-12 education will be delivered using some kind of online learning by the year 2020 [9].

Despite these recent advances however, Barbour and Reeves [5] wrote, "[T]here has been a deficit of rigorous reviews of the literature related to virtual schools" (p. 402). Similarly, Cavanaugh *et al*. [7] found only a small percentage of the open access literature was based upon systematic research, while most of the literature was based on the experiences or opinions of K-12 online learning practitioners. Further, Rice [23] indicated that "…a paucity of research exists when examining high school students enrolled in virtual schools, and the research base is smaller still when the population of students is further narrowed to the elementary grades" (p. 430). Simply put, while the practice of K-12 online learning is growing at an exponential rate, the availability of empirical research to guide that growth has been lacking. As a response to this need, the collaboratory described herein will also be used as a platform to collect data focusing on how it can enhance learning. The following sections describe the development of the Sustainable Product Development Collaboratory and its use as a pedagogical tool, including the description of a teaching module focused on product design and manufacturing and supply chain analysis, and methods for student assessment.

## 2. PROJECT OVERVIEW

The overarching objective of the CI-TEAM Demonstration Project discussed herein is to convey sustainability principles in the context of product architectural design, manufacturing, assembly, and supply chain decisions to a wide spectrum of active learners, ranging from K-12 students, to university students, and to practitioners. The project will actively engage learners in the development of, and research conducted within the collaboratory. The collaboratory is enabled by user-friendly, license-free web-based tools (e.g., Google SketchUp) to deliver a holistic and broadly usable cyber-platform. The specific goals of this CI-TEAM project include:

- Deploying a Sustainable Product Development Collaboratory that includes modules to support conceptual design variant generation, life cycle cost and environmental analysis, and supply chain optimization;
- Developing and disseminating educational materials that can provide project-based activities in support of interaction with the Sustainable Product Development Collaboratory;
- Assessing the educational effects, or more specifically, the cyberinfrastructure competency gained through interaction with the Sustainable Product Development Collaboratory, including assessment of activities at the participating universities and user adoption of the cyber-platform; and
- Engaging underrepresented groups and high-school students to promote a diverse workforce that is ready to exploit cyberinfrastructure tools.

Below, we first explain the underlying educational philosophy adopted during the development of the collaboratory and then we present a sample learning module and methods of assessment. Finally, we discuss conclusions and observations based on the collaboratory and learning module development efforts.

## 3. PEDAGOGICAL AND INSTRUCTIONAL METHODOLOGIES FOR COLLABORATORY-BASED SUSTAINABLE DESIGN EDUCATION

Although we had a clear vision that a cyber-based tool and interactive e-learning platform had to be built as introduced above, we opted to think critically and learn from prior literature about what pedagogical and instructional methodologies we should follow to make it more effective. Below, we provide a summary of our findings along with our philosophical direction.

Carew and Mitchell [6] studied engineering academics' conceptions of sustainability and stated that variation in conceptions of sustainability and explicit contestation of the variation in the engineering classroom offers opportunities to enrich undergraduate learning and teaching. In their study, Carew and Mitchell [6] concluded that sustainability education requires a diversity of teaching and learning methods that can consider the role of values and assumptions in sustainable decision-making. One of the ways in which instructional design can be varied is in the autonomy the learner may have in completing learning activities. Prior literature points to the potential positive effect of increasing autonomy as the learners develop intellectually.

Vygotsky [26] observed that learning for children and adolescents is a social process that focuses upon interaction within a zone of proximal development. The zone of proximal development "…is the distance between the actual developmental level as determined by independent problem solving and the level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers" (Vygotsky [27], p. 86). Cavanaugh *et al*. [8] suggested, "[S]ince adults have progressed through these stages of cognitive development, delivery of web based education at the adult level need not concentrate on methods that help the learner develop these cognitive skills" (p. 7). Methods designed to help younger learners develop cognitive skills are intended as guidance to ensure that these learners remain in the zone of proximal development. Further, Moore [18] noted that K-12 educators typically are expected to maintain control of the content and method of delivery within the classroom. In fact, Moore even posited that K-12 students "should not be compelled to assume a degree of autonomy they are not ready to handle, and so it is customary in child education for the preparatory and evaluation processes to rest entirely in the hands of the teacher" (p. 84). Simply put, children are not ready to assume high degrees of autonomy, and thus child and adolescent learners require more structure in their educational settings.

The approach employed for scaffolding of learning is an important concern when autonomy of learning is not left to the learner. One compelling approach for scaffolding is constructionism. As a form of constructivist learning theory, constructionism is essentially the process of learning through constructing, or designing or making a product. This learning theory is based on Papert's [20] work with students using the Logo programming language, where they programmed an electronic "turtle" to move about on the screen or a physical "turtle" to move about the floor and leave a marking of

where the object had traveled. Papert believed that through a process of trial and error, the students learned how to command and debug the "turtle" to create specific geometric shapes (and thus learned mathematical problem-solving and geometry). Papert illustrated how computer programming could be used to help teach these mathematical concepts to students who traditionally struggled with the subject. Recently, constructionism has been adopted by researchers who are interested in what students can learn through the process of designing games [14, 15, 17, 21, 22].

The constructionist line of inquiry has regularly been found to enable students to attain a deeper understanding of the concept being taught, have richer discussions about that content, and retain the knowledge longer than students taught in more traditional, instructor-centric environments. Given these findings, we have been developing the Sustainable Product Development Collaboratory to provide a medium for learning sustainability concepts relevant to product development, manufacturing and supply chain design through constructed knowledge across carefully crafted learning modules.

## 4. CONCEPTUAL LEARNING MODULES FOR THE COLLABORATORY

Learning modules have been developed to demonstrate the effects of different product designs on supply chain costs and environmental impacts by using the Sustainable Product Development Collaboratory, which is comprised of several web application technologies. The collaboratory framework consists of three main modules, i.e., design module, manufacturing analysis module, and supply chain analysis module, as shown in Figure 1. The design platform, which uses Google SketchUp, a freely available 3D modeling tool, communicates with a web-based design/analysis interface, called the "collaboratory portal."

Alternatively, learners can access previously modeled products in the Product Design Database (PDDB) for further cost and environmental analysis. In consideration of the educational context for learners, in particular for K-12 students, a simple and easily accessible design platform is needed, so that learners do not require additional training in model generation and design modification. Accordingly, Google SketchUp was selected as the design platform for the collaboratory.

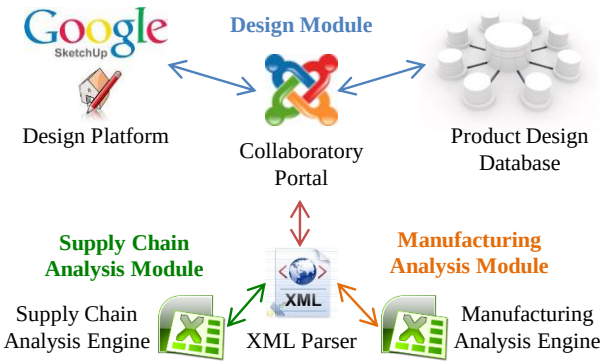


**Figure 1. Collaboratory framework showing the portal and design, manufacturing, and supply chain analysis modules.**

With limited geometric and engineering analysis functionality, SketchUp represents a 3D modeling tool for beginners. A plugin was developed for the collaboratory to provide basic functions to extract geometric and engineering information. Figure 2 displays the SketchUp plugin for volume calculation developed under this project. If several models or geometries are in the SketchUp platform, the volume calculator will process only the active model, i.e., the component or assembly in the bounding box.

A geometry slicing method is used to determine the solid volume within the bounding box. The selection of accuracy level depends on the complexity (irregularity) of the geometric shape. If the bounding box is assumed to be in stock material dimensions, for instance, subtracting the actual part volume from the bounding box volume determines how much material will be removed during manufacturing. Using basic functions in SketchUp, learners can modify an existing product model or generate a new product model according to their own desire. In addition, the collaboratory library supports the learners with preprocessed component and assembly models. Currently, the library contains the components and assembly of a bicycle pedal.

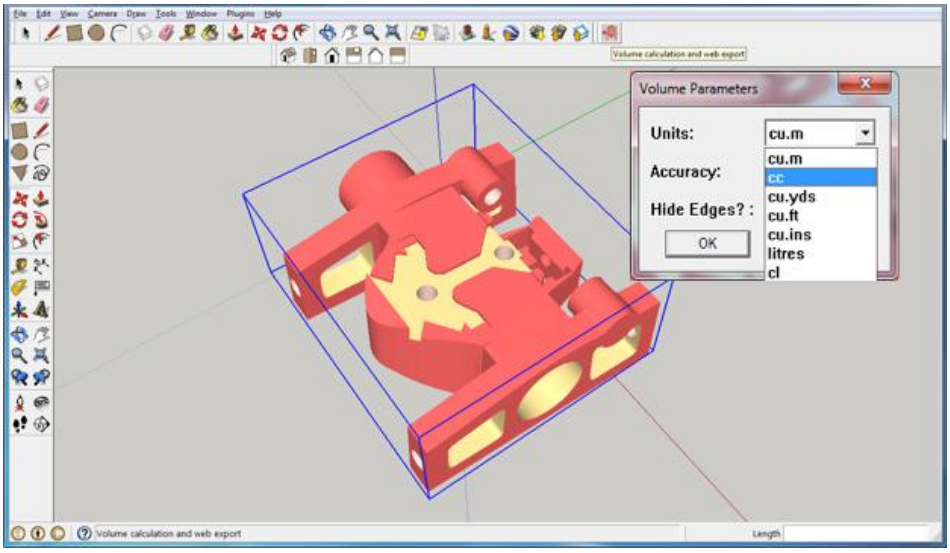


**Figure 2. Design platform plug-in for geometry and bounding box volume calculation.**


**Figure 3. Welcome page of the analysis interface.**

In addition to product design capabilities, the collaboratory portal provides an interface for the manufacturing analysis module and the supply chain analysis module. The prototype welcome page of the analysis interface is shown in Figure 3. The analysis interface includes the pre-processed model analysis interfaces, a PDDB communication interface, an XML parsing interface, a system-solver communication interface, and a post-processing interface. The pre-processed model analysis interface provides the user an opportunity to view and select the pre-processed models from the collaboratory library (PDDB). Both assembly level and component level models are available in the library. Learners can browse the assemblies and components, and the design-analysis system interface displays an image of the selected component (Figure 4). Learners can use this interface to download the SketchUp-compatible drawing file from the collaboratory library for further processing and design modification. The file can be modified using SketchUp and exported to the collaboratory library for manufacturing and/or supply chain analysis.


**Figure 4. Design-analysis interface showing the body plate component model.**

The design-analysis interface works for both the pre-processed models and newly designed models. For pre-processed models, design, manufacturing, and other analysis data are stored in the PDDB. The PDDB has been designed using MySQL, and the communication between the web portal and the MySQL has been developed using Java. In the case of pre-processed models, the Java code receives the properties from the PDDB corresponding to the selected pre-processed model ID. On the other hand, for a newly designed model, the design properties are stored in the PDDB as required for manufacturing process modeling. This interface has the intelligence to recognize whether the analysis command was initiated for a newly designed model or a pre-processed model. The interface exhibits the corresponding properties, collected from the PDDB, for the selected model and provides the user a place to define additional input parameters. The portal displays basic geometric information taken from the PDDB along with representative input fields (Figure 5).

Ongoing development is extending the PDDB and the input fields based upon the requirements of the manufacturing and supply chain analysis modules. The interface sends all the parameters displayed in the portal to the analysis engines through XML parsers. The manufacturing and supply chain analysis solvers are stored on a central server along with the collaboratory portal. The solver has separate worksheets for input parameters and output parameters. For performing analysis, the analysis interface reads the Excel workbook template stored in the PDDB and creates a copy of the workbook in the PDDB. The purpose of copying the workbook is to keep the workbook template protected from malicious activities.

After creating the new workbook, the interface reads all the input fields and adds the input parameters to the corresponding input fields. If an Excel worksheet contains any formulas, logic, and/or links; the updates made in the input fields are not executed automatically. Execution of the formulas and logic steps is forced by reading all the worksheets. The execution time varies depending on the size and the contents of the workbook. After completing analysis, the interface reads the output worksheet and the output fields. The output parameters are sent to the XML parsing interface for storage and transmission to the post-processing portal.

Figure 6 illustrates the flow of the manufacturing analysis solver for a set of processes that might be used to fabricate a bicycle pedal body (PB), i.e., casting, boring, and milling. From the input parameters, which describe the materials and stock and final part geometries, the manufacturing analysis solver calculates total process energy use and equivalent $CO_2$ emissions (kg $CO_2$ eq.). The process carbon footprint (kg $CO_2$ eq.) values for two variants are then displayed numerically and graphically for interpretation.

With the design and manufacturing/supply chain analysis functionalities thus available in the collaboratory, learning modules can be constructed for use in the classroom at multiple complexity and comprehensiveness levels to educate a wide spectrum of learners about the concepts and practice of sustainable product development. In the sample learning module presented herein, we use the design of a bicycle pedal as a sample project. The sample learning module includes four parts as shown in Table 1; these modules are discussed in greater detail below.

**Table 1. Key parts of the sample learning module**

| |
|---|
| Module Part I. Introduction to the Activity |
| Module Part II. Software Demonstration |
| Module Part III. Bicycle Pedal Analysis Project |
| Module Part IV. Discussion |

In Part I and Part II, the overall process, anticipated activity, and software (collaboratory) capabilities are explained to the participating students. Students at all levels are familiar with bicycles, but may not be aware of the variety of pedal types available. Thus, the module would start with an introduction and discussion of bicycle pedal types, which include platform, clipless, and pedals with toe clips. Images could be displayed using a projector, or actual pedals could be passed around the classroom to show the many types and styles.

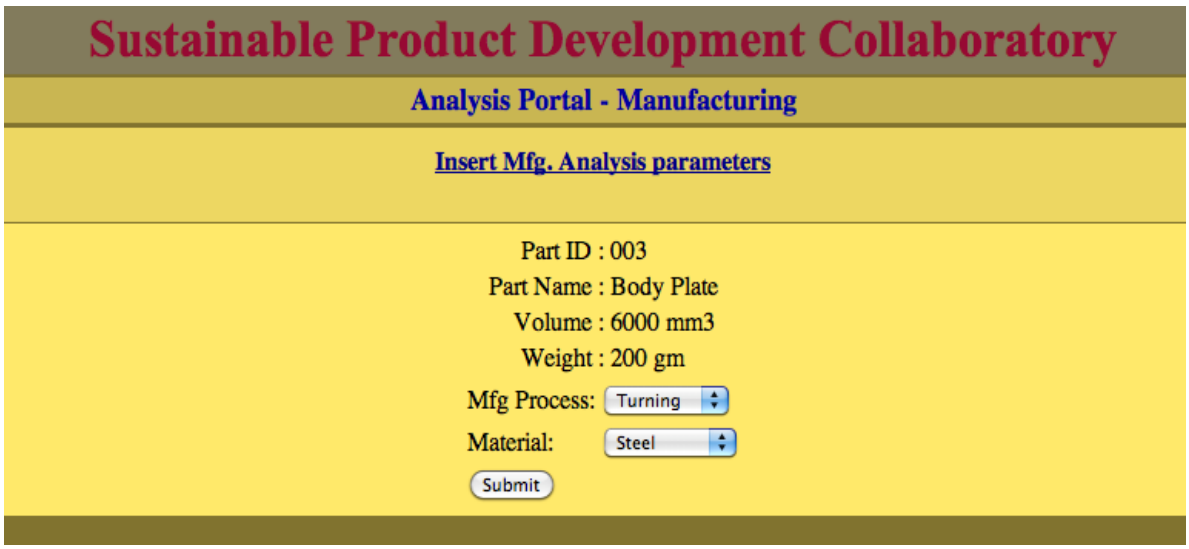


**Figure 5. Interface showing properties collected from the PDDB and user defined input fields.**

| PB 1 - Machining | | | | |
|---|---|---|---|---|
| **Operation** | **Process Parameter** | **Var.** | **Value** | **Units** |
| **Casting** | Length of Bounded Vol. | $l$ | 72.000 | mm |
| | Width of Bounded Vol. | $w$ | 61.000 | mm |
| | Height of Bounded Vol. | $h$ | 24.000 | mm |
| | Void Vol. w/in Bound. Vol. | $V_v$ | 27629 | $mm^3$ |
| | Material Density | $d$ | 0.000 | $kg/mm^3$ |
| | Mass of Steel Plate | $m$ | 0.210 | kg |
| | Specific Energy Required | $C_E$ | 10.990 | MJ/kg |
| | **Energy Consumption** | **$E_C$** | **2307.936** | **kJ** |
| **Boring** | Length of Cut | $L$ | 72.000 | mm |
| | Initial Diameter | $D_i$ | 10.160 | mm |
| | Final Diameter | $D_f$ | 14.097 | mm |
| | Unit Power | $U$ | 120000 | $in\text{-}lb/in^3$ |
| | Unit Power | $U$ | 0.001 | $kJ/mm^3$ |
| | **Energy Consumption** | **$E_C$** | **0.298** | **kJ** |
| **Milling** | Length of Cut | $L$ | 101.443 | mm |
| | Width of Desired Cut | $W$ | 0.508 | mm |
| | Depth of Cut | $d$ | 24.003 | mm |
| | Cutter Diameter | $D_c$ | 6.350 | mm |
| | Tool Passes | $P$ | 0.080 | |
| | Unit Power | $U$ | 120000 | $in\text{-}lb/in^3$ |
| | Unit Power | $U$ | 0.001 | $kJ/mm^3$ |
| | **Energy Consumption** | **$E_C$** | **2.274** | **kJ** |

Calculate totals from design and process information

| Process Type | Energy Use (kJ) | | Eq. $CO_2$ (g) | |
|---|---|---|---|---|
| | PB 1 - Machining | PB 2 - Net Casting | PB 1 - Machining | PB 2 - Net Casting |
| Casting | 3127.77 | 2307.95 | 420.00 | 310.00 |
| Machining | 5.98 | 2.57 | 0.80 | 0.35 |
| Totals | **3133.75** | **2310.52** | **420.80** | **310.35** |

Display results

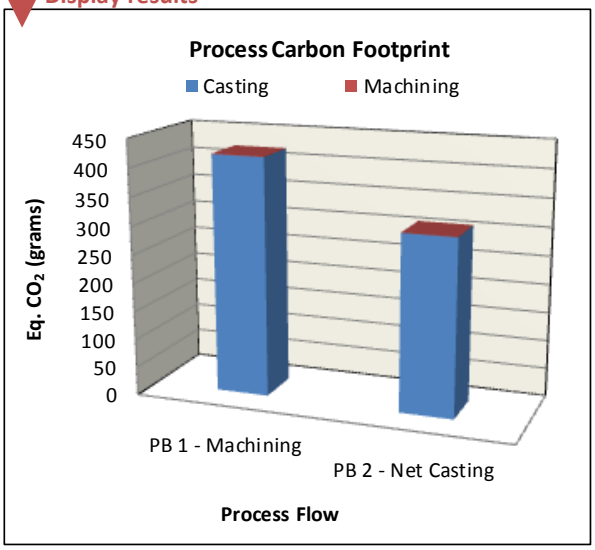


**Figure 6. Manufacturing analysis solver operation.**

In Part III, students would undertake a pedal design project using the collaboratory, working individually or in pairs, to evaluate the different pedal designs and/or approaches to produce and assemble the pedals. Based on what the students discover, the instructor can lead a discussion in Part IV of the module to further cement the concepts of cost and environmental impact, as well as how they can be influenced by product and process designs. The instructor may conclude the discussion with how this might relate to purchasing decisions students make in their own lives.

This module would be preceded by and concluded with subject matter pre- and post-tests to assess the knowledge gains in students. The tests are designed to assess multiple topics related to design activities completed with the collaboratory e-learning platform. Each pedal design requires different types and amounts of materials, different manufacturing processes to produce, and different supply chains to provide parts and materials for the pedal. By evaluating the set of pedal types within the collaboratory library, students at different levels of learning can thus explore different environmental effects (e.g., carbon footprint and energy consumption) of design changes. At higher levels of learning, students can be asked to change the design parameters (e.g., size) and engineering properties (e.g., material) using Google SketchUp along with the collaboratory.

## 5. FOCUS GROUP STUDY

To validate the concept of employing the collaboratory within a learning module, an interview was conducted with a focus group consisting of middle and high school teachers in Michigan. In the State of Michigan, Next Generation Science Standards (NGSS, http://www.nextgenscience.org/) are being implemented with strong emphasis in ecosystems, sustainability, and human impacts. During the focus group interviews, the teachers supported adoption of this collaboratory concept into the new curriculum. They opined that the subject of *human impact on the environment*, which is covered in eighth and ninth grades, is the topic where the sustainability design education fits well. In general, the teachers agreed that "understanding how an end product was realized and delivered to consumers" should be emphasized more, especially with respect to *human impact on the environment*. The scenario based sustainable design education activity aims to tackle these curricular needs.

In order to test the usability of the collaboratory in the classroom, another focus group study was conducted with a modified Task-Technology Fit questionnaire [13]. Ten graduate students responded to this survey, which consisted of 20 questions. The respondents indicated the ability of the system to conduct the assigned design task using a 7 point Likert scale (1: strongly agree – 7: strongly disagree). The assigned design task was to evaluate the pedal types and explore the effect on environmental performance (i.e., energy consumption) of design changes. Each pedal design requires different types and amounts of materials, different manufacturing processes to produce, and different supply chains to supply parts and materials.

Most questions received an average response of approximately 2 points (Figure 7), which indicates that respondents strongly agreed with the statements. In addition, the standard deviations for most of the responses are 1 to 1.5, pointing to the fact that most of the respondents evaluated the system with the positive portion of the scale (i.e., 1-4).

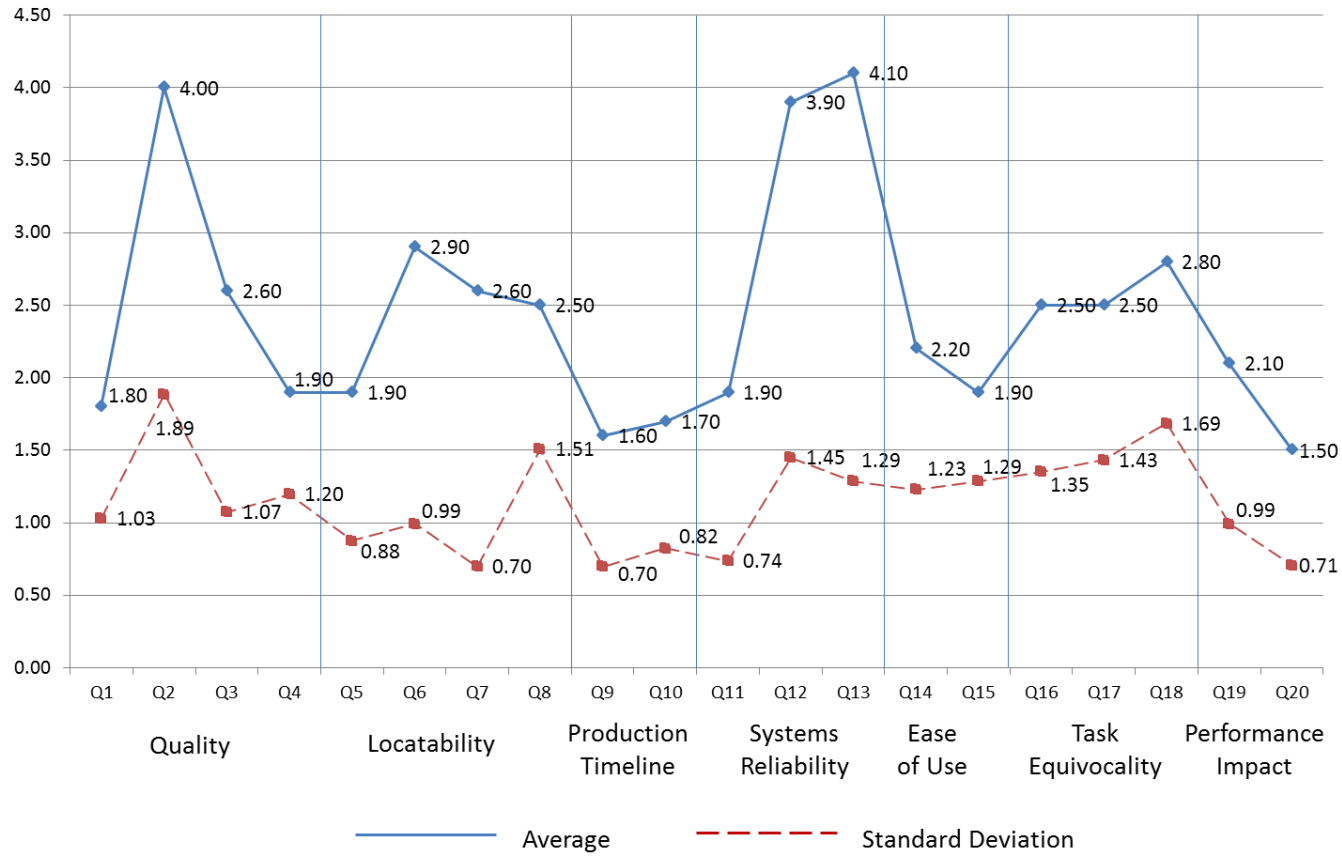


**Figure 7. Collaboratory usability test results.**

However, two questions about Systems Reliability, i.e., "The Collaboratory system is subject to unexpected or inconvenient down times, which makes it harder to do this work" (Q12) and "The Collaboratory system is subject to frequent problems and crashes" (Q13) had averages of 3.9 and 4.1, respectively. Thus, the system reliability must be improved to be more robust for a better user experience. In addition, the average of the question about Quality, "The Collaboratory system is missing critical data that would be very useful in this job" (Q2) was 4.0 (standard deviation of 1.8). Q2 relates to the ability of the system to maintain the data, which was needed by the users, thus improved ability of the system to maintain data is needed for users to identify changes in the data and to access the previous and current data easily.

This section demonstrated the collaboratory usability assessment, which shows the effectiveness of the collaboratory for the given design task, i.e., evaluating the impact of different pedal designs on environmental performance. The following section describes a proposed method for knowledge assessment.

## 6. KNOWLEDGE ASSESSMENT

The knowledge assessment targets the cyberinfrastructure competency gained through interaction with the Sustainable Product Development Collaboratory as well as content knowledge gained through pre- and post-tests. Pre- and post-testing focuses on the following three learning objectives:

1) Developing an awareness and understanding about the impacts of product architecture, manufacturing process, and supply chain decisions on the economic and environmental sustainability of a product;
2) Articulating the impacts of product architecture, manufacturing process, and supply chain decisions on the economic and environmental sustainability of a product; and
3) Developing product design solutions that address technical requirements, in addition to economic and environmental sustainability goals.

These objectives cover students' knowledge gains through abstract means as well as a more applied project-based approach, and thus, we use Kolb's Learning model [16] as a basis in crafting our assessment questions. In this model, knowledge construction is assumed to progress in various stages, which are not necessarily experienced in order. These stages include

Stage 1: Observation of concrete situations from different perspectives (Concrete Experience – CE)

Stage 2: Observation and reflection of the experiences (Reflective Observation – RO)

Stage 3: Formation of abstract concepts and generalizations based on experiences and reflections (Abstract Conceptualization – AC)

Stage 4: Testing the implications of the concepts and generalizations (Active Experimentation – AE).

In its essence, the collaboratory is a medium for students to actively experiment with a concrete situation (product design) to test the learned concepts, in addition to providing guidance as critical domain knowledge. Active experimentation also fits well with the constructionist approach, which encourages learning through constructing, or designing or making a product [20].

The knowledge-gain assessment questions that we have developed are open-ended in nature, and tap into awareness of the concepts and the level of articulation. The questions also involve solving problems using the concepts learned; therefore, they cover all stages in Kolb's Learning model. Sample questions that can be used to assess knowledge gain include the following:

*In your own words, explain what you understand about the environmental impact of a product*.

*Explain the contribution of different life cycle stages on the environmental impact of a product*.

*Which of the following statements best describes your understanding of current product design practice*?

Student responses to knowledge assessment pre- and post-tests will be evaluated based on the pre-recorded correct answers to assess the level of knowing on this particular subject – sustainable design, manufacturing, and supply chain management.

By using this sample learning module and the design activities, it is anticipated that students will be able to analyze the relative impacts of components of a particular pedal, as well as the effects of changes to their related geometries, manufacturing processes, and supply chains. The actual implementation and assessment results will be reported in an upcoming article.

## 7. CONCLUDING REMARKS

This article presented a pedagogical framework and a sample learning module developed under an NSF CI-TEAM Demonstration project, entitled "A Sustainable Product Development Collaboratory." This project aims to educate a wide spectrum of learners (K-12, university, industry) in sustainable design and manufacturing by demonstrating the effects of different product designs on supply chain costs and environmental impacts. The presented collaboratory has the potential to create an evolving design repository, promote empirical/experimental investigation to model life cycle costs and environmental performance, and advance methods for joint optimization of design variants and supply chains, while being readily available and reusable by students and practitioners. In addition, the collaboratory stands to benefit educational research by providing a platform for experimental learning module development, implementation, and assessment in the classroom environment at multiple levels and in multiple regions.

A focus group study was conducted to understand middle school and high school teacher's perspectives. While they stated the importance of sustainability education and relevancy of the collaboratory concept to their curricula, they also emphasized that student constructivist learning behavior should be addressed. The teachers indicated that a game type or competition based learning environment is effective. The collaboratory will be further enhanced to support this constructivist pattern of learning.

Collaboratory development is focused on designing a bicycle pedal by considering sustainability principles in design, manufacturing, and supply chain activities. However, evaluating sustainability implications of a product design decision should include the impacts of the overall product life cycle. In other words, products that are superior when manufacturing performance metrics are taken into account may not be the ideal choice when considering other life cycle aspects (e.g., service or end of life). Thus, performance of other life cycle stages will be continuously included in this scalable collaboratory environment.

## 8. ACKNOWLEDGMENTS

This work is funded by the National Science Foundation under grant numbers OCI-1041423 (Oregon State), OCI-1041328 (Penn State), and OCI-1041380 (Wayne State).

## 9. REFERENCES

[1] Allen D, Allenby B, Bridges M, Crittenden J, Davidson C, Hendrickson C, Matthews S, Murphy C, Pijawka D (2008). *Benchmarking Sustainable Engineering Education: Final Report*. EPA Grant Number: X3-83235101-0.

[2] Ambient Insight. (2011). *2011 learning technology research taxonomy: Research methodology, buyer segmentation, product definitions, and licensing model*. Monroe, WA: Author. Retrieved from www.ambientinsight.com/Resources/Documents/AmbientInsight_Learning_Technology_Taxonomy.pdf

[3] Barbour, M. K. (2009). Today's student and virtual schooling: The reality, the challenges, the promise... *Journal of Distance Learning*, 13, 1, 5-25.

[4] Barbour, M. K. (2011). The promise and the reality: Exploring virtual schooling in rural jurisdictions. *Education in Rural Australia*, 21, 1, 1-20.

[5] Barbour, M. K., & Reeves, T. C. (2009). The reality of virtual schools: A review of the literature. *Computers and Education*, 52, 2, 402–416.

[6] Carew, A., Mitchell, C. (2008). Teaching sustainability as a contested concept: capitalizing on variation in engineering educators' conceptions of environmental, social and economic sustainability. *Journal of Cleaner Production*, 16, 1, 105-115

[7] Cavanaugh, C., Barbour, M. K., & Clark, T. (2009). Research and practice in K-12 online learning: A review of literature. *International Review of Research in Open and Distance Learning*, 10, 1. http://www.irrodl.org/index.php/irrodl/article/view/607

[8] Cavanaugh, C., Gillan, K. J., Kromrey, J., Hess, M., & Blomeyer, R. (2004). *The effects of distance education on K–12 student outcomes: A meta-analysis*. Naperville, IL: Learning Point Associates. Retrieved July 4, 2005, from http://www.ncrel.org/tech/distance/k12distance.pdf.

[9] Christensen, C. M., Horn, M. B., & Johnson, C. W. (2008). *Disrupting class: How disruptive innovation will change the way the world learns*. New York : McGraw-Hill.

[10] Clark, T. (2001). *Virtual schools: Trends and issues - A study of virtual schools in the United States*. San Francisco, CA: Western Regional Educational Laboratories. Retrieved July 4, 2005 from http://www.wested.org/online_pubs/virtualschools.pdf

[11] Clark, T. (2007). Virtual and distance education in North American schools. In M. G. Moore (Ed.), *Handbook of*

*Distance Education* (2nd ed., pp. 473–490). Mahwah, NJ: Lawrence Erlbaum Associates, Inc.

[12] Dankwort, C. W., Weidlich, R., Guenther, B., and Blaurock, J. E. (2004). Engineers' CAx education - it's not only CAD. *Computer-Aided Design*, 36, 1439-1450.

[13] Goodhue, D. L. and Thompson, R. L., (1995) Task-Technology Fit and Individual Performance. *MIS Quarterly*, 19, 2, 213 – 236

[14] Kafai, Y. (2001). The educational potential of electronic games: From games–to–teach to games–to–learn. Paper presented at the Playing by the Rules: The Cultural Policy Challenges of Video Games, Chicago, IL.

[15] Kafai, Y., Ching, C. C., & Marshall, S. (1997). Children as designers of educational multimedia software. *Computers in Education*, 29, 2, 117-126.

[16] Kolb, D. A. (1984). *Experiential Learning: experience as the source of learning and development*. New Jersey: Prentice-Hall (0 13 295261 0)

[17] Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: urban youth learning programming with scratch. *Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education*, Portland, OR, USA.

[18] Moore, M. G. (1973). Toward a theory of independent learning and teaching. *Journal of Higher Education*, 44, 12, 661-679.

[19] NSF 2007 MT21 Study, http://www.nsf.gov/news/news_summ.jsp?cntn_id=110845

[20] Papert, S. (1980). *Mindstorms: Children, Computers, and Powerful Ideas*. New York: Basic Books.

[21] Peppler, K. A., & Kafai, Y. (2007). From SuperGoo to Scratch: Exploring creative digital media production in informal learning. *Learning, Media and Technology*, 32, 2, 149-166.

[22] Resnick, M. (2009). Scratch programming for all. *Communications of the ACM*, 52, 11, 60.

[23] Rice, K. L. (2006). A comprehensive look at distance education in the K-12 context. *Journal of Research on Technology in Education*, 38, 4, 425-448.

[24] SME Competency Gap Research, http://www.sme.org/downloads/foundation/Competency_Gap.pdf, Accessed 12/28/06.

[25] Thota, R. and Dwivedi, S. (2006). Implementation of product realization concepts in design and manufacturing course. ASEE paper, 2006-2180.

[26] Vygotsky, L. S. (1962). *Thought and Language* (E. Hanfmann & G. Vakar, Trans.). Cambridge, MA: The M.I.T. Press.

[27] Vygotsky, L. S. (1978). *Mind in society: The development of higher psychologist processes*. Cambridge, MA: Harvard University Press.

[28] Watson, J., Murin, A., Vashaw, L., Gemin, B., & Rapp, C. (2011). *Keeping pace with K–12 online learning: An annual review of policy and practice*. Evergreen, CO: Evergreen Education Group.

# A Hands-on Education Program on Cyber Physical Systems for High School Students

Vijay Gadepally*
Ohio Supercomputer Center
1224 Kinnear Road
Columbus, Ohio
vijayg@osc.edu

Ashok Krishnamurthy
Ohio Supercomputer Center
1224 Kinnear Road
Columbus, Ohio
ashok@osc.edu

Umit Ozguner
The Ohio State University
Electrical and Computer Engineering
umit@ece.osu.edu

## ABSTRACT

Cyber Physical Systems (CPS) are the conjoining of an entities' physical and computational elements. The development of a typical CPS system follows a sequence from conceptual modeling, testing in simulated (virtual) worlds, testing in controlled (possibly laboratory) environments and finally deployment. Throughout each (repeatable) stage, the behavior of the physical entities, the sensing and situation assessment, and the computation and control options have to be understood and carefully represented through abstraction.

The CPS Group at the Ohio State University, as part of an NSF funded CPS project on "Autonomous Driving in Mixed Environments", has been developing CPS related educational activities at the K-12, undergraduate and graduate levels. The aim of these educational activities is to train students in the principles and design issues in CPS and to broaden the participation in science and engineering. The project team has a strong commitment to impact STEM education across the entire K-20 community.

In this paper, we focus on the K-12 community and present a two-week Summer Program for high school juniors and seniors that introduces them to the principles of CPS design and walks them through several of the design steps. We also provide an online repository that aids CPS researchers in providing a similar educational experience.

## Categories and Subject Descriptors

K.3.2 [**Computers and Education**]: Computer and Information Science Education – computer science education

## General Terms

Education, Design

*Vijay Gadepally is a PhD candidate at The Ohio State University and corresponding author



## Keywords

Education, Supercomputing, Human Factors, Cyber Physical Systems

## 1. INTRODUCTION

As the Cyber-Physical Systems Group (CPS) at The Ohio State University, under an NSF funded project entitled "Autonomous Driving in Mixed Environments", we have been planning educational activities to promote student interest in the Science, Technology and Mathematics (STEM) fields. The need for promoting STEM related education to middle-high school students in the STEM fields has been widely documented [7, 11, 5] and can be summarized as highlighting the need for the United States to prepare a sufficient number of STEM professionals capable of innovation. As described in [6], STEM education can be visualized as a pipeline that begins in early education, extends through college and ends with employment with critical transition points that includes the high school to college transition.

The CPS group at Ohio State has significant experience working designing K-12 education related activities (in addition to college undergraduate and graduate student courses). In [8], the authors discuss a project designed for middle school girls as a part of a week-long workshop, entitled "Future Engineers' Summer Camp" held at The Ohio State University. The authors describe their approach to introducing middle school girls to fault tolerant computing through a variety of kinesthetic learning activities. Kinesthetic learning activities [10], are proposed by the authors as a process by which students learn about theoretical concepts by carrying out physical activities, as opposed to passively listening to lectures. The authors describe their success using kinesthetic learning activities to explain complex algorithms, such as sorting, to younger audiences. In [9], the authors discuss coursework developed at the undergraduate level to teach students the important concept of abstraction.

In this paper, we present a two-week educational program for high school students that introduces them to Cyber Physical Systems, and the design and development of such systems through the modeling and simulation at differing levels of abstraction. Students are introduced to CPS through the engineering of *autonomous vehicles* or *driverless cars*. In particular, students are asked to develop algorithms for a vehicle (in this case, a Roomba) that can avoid obstacles. The aim of the program is to help students emulate the scientific process employed by CPS researchers while learning to use common techniques and tools. This paper aims

to describe the activities and provide sufficient resources to facilitate the emulation of such a program.

CPS design, due to the cost and difficulty of direct physical testing, typically goes through four phases in system design:

1. **Conceptual Modeling**: Understand the mathematics of the problem, and propose a theoretical solution (develop equations of motion, develop analytical solutions where possible.)

2. **Simulated Testing**: Use computer simulations to validate algorithms (use a software package with a simulated test bed to test obstacle avoidance algorithms.)

3. **Controlled Environment Testing**: Use a physical test-bed of a simulated environment to validate algorithms (use the developed algorithm on a physical Roomba within a simulated environment.)

4. **Real World Deployment**: Test the algorithm in the real world (use the developed algorithm on a physical vehicle on actual city streets.)

The activities described in this paper are conducted as a part of the Summer Institute (SI) held at the Ohio Supercomputer Center (OSC). SI is a two-week residential program for gifted highschool freshman, sophomores, and juniors designed to raise students' interest and awareness of the STEM fields. SI challenges students to cultivate their research ability through the use of cutting-edge tools, modeling and simulation, and interaction with active researchers. Students are also encouraged to develop interpersonal skills through presentations and participation in a variety of science related field trips, and teambuilding activities.

The CPS project developed for this purpose is called "Obstacle Avoidance Roombas," and is a direct product of autonomous vehicle research carried out by CPS researchers at Ohio State. Over two years, 2010 and 2011, eight motivated students were chosen to participate in the project. These students were introduced to real-life aspects of CPS designs, trained in C/C++ programming, and taught relevant mathematics and physics concepts. Students were then asked to use a simulator, called Player/Stage, to program (in C/C++) Roombas (as shown in Figure 1) to complete the project. The project was divided into a sequence of four subproblems to help students understand the logical project progression.

1. Program a Roomba to follow a set of coordinates entered by a user

2. Program a Roomba to acquire a target, and plan the optimum path to reach the target

3. Program a Roomba to acquire a target, and avoid a single obstacle to reach the target

4. Program a Roomba to acquire a target, and avoid multiple obstacles to reach the target

Students are taught how simulations can provide a path to real world implementation, and use developed code on a set of robots and obstacles in a laboratory setting at Ohio State University.

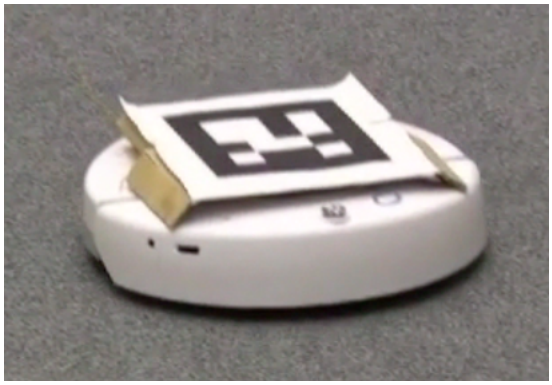

**Figure 1: Roomba fitted with GPS tag**

In this article, we present details of this project, the educational materials developed, and results obtained. We begin by giving an overview of the SI program, the CPS related project, and the intended competencies. We present the logical progression of the project through the four step process taken by CPS researchers. We conclude the project description with student feedback and lessons learned by the CPS staff. In order to facilitate similar projects, we end with a list of the resources that were used in developing this project.

## 2. ABOUT OSC'S SUMMER INSTITUTE

For over 20 years, the Ohio Supercomputer Center (located in Columbus,OH) has offered the Summer Institute (SI) to Ohio's gifted students entering their sophomore, junior or senior years of high school and their teachers. SI is a two-week residential program designed to raise students' interest in the Science, Technology, Engineering and Mathematics (STEM) fields through a collaborative and dynamic research environment and hands on experience with the latest in cutting edge technology. The program is held in Columbus, OH and students live in the dormitories of The Ohio State University for this two-week period. Students typically arrive at OSC by 9 AM every morning, and work on their chosen projects until 5 PM when they are taken back to their dorm rooms and take part in a variety of social activities. Each year, a number of projects are chosen by SI staff that appeal to students. Projects are decided taking into account previous student feedback, real-world applicability of project, staff expertise and funding.

The program begins by teaching students UNIX, the operating system of the computers they use. Next, students learn a programming language (C/C++/MATLAB) and any software required to complete their projects. Students are required to do their own work from code implementation to final presentations. The ability to develop algorithms and an understanding of the project's science/engineering basis are needed.

### 2.1 SI 2010 and 2011

Thirty two students, and four teachers participated in the SI's held in 2010 and 2011. In SI 2010 and 2011, there were four projects ranging from robotics to medical imaging. The four projects with descriptions (as given to students) are given in [1]. The project described in this paper, Obstacle Avoidance Roombas, was presented to SI participants as

follows:

**Obstacle Avoidance Roombas**:

*Many organizations, such as the US Army, require vehicles that are capable of avoiding dangerous objects. These objects may be explosive obstacles, booby trapped buildings, or armed personnel. This sponsored project involves students using robots and robotic simulators to design an "Obstacle Avoidance Roomba."*

*The project uses a Roomba, an autonomous robotic vacuum cleaner, fitted with LIDAR (a light based ranging device). The goal of the project is to program the Roombas to avoid randomly placed obstacles to reach a destination. Students work with a vehicle simulator, called Player/Stage, to design "the brains" of a Roomba . After successfully simulating the behavior, the research group will do "real-life" tests of the Roomba in a specially fitted laboratory.*

The SI program began with a presentation of four projects and students were allowed to choose one based on their interests.

## 2.2 Project: Obstacle Avoidance Roombas

The project involves the design of algorithms and code that directs a Roomba to avoid obstacles and reach a target or goal. Initial programming and testing of code is done using the Player/Stage [3] simulator. Player is a network server for robot control that supports a variety of robot hardware. Stage simulates (in 2.5D) a population of mobile robots, sensors and objects. The Player/Stage package allows quick prototyping of algorithms for implementing embedded computers. The Player/Stage environment is designed to simulate a set of roads and intersections set up in the Control and Transportation Laboratory (CTL) testbed, at The Ohio State University. Sensors used in the testbed are simulated in the Player/Stage environment. Thus, the students see how a "real-world" laboratory environment can be abstracted and modeled in a simulated environment. The necessity of a "cyber" environment is further demonstrated by the ease with which code prototyped in the simulator can be transferred to testbed equipment. Students are also introduced to limitations of the simulated world, and situations in which the simulator may allow a violation of physics or mathematical possibilities in the real world such as a simulator valid Roomba position that translates to a Roomba hovering above the ground that cannot be attained physically.

In general, the two weeks of SI are divided into training and project components. The first week consists primarily of providing students with tools and any mathematical or physical foundations required to complete their projects. Students also make multiple laboratory visits to gain an understanding of the environment that they are simulating.

## 2.3 Competencies

The CPS research team decided on various competencies that would need to be taught to students.

### 2.3.1 Mathematics

Students were given a two hour interactive lecture that introduced them to the mathematical concepts required to complete Obstacle Avoidance Roomba project. The lecture began with a refresher on coordinate geometry and covered


**Figure 2: OSU ACT Vehicle and Sensors**

concepts such as: frames of reference, coordinate and homogeneous transformations. Student competency was tested with simple mathematical problems such as: "The roomba is facing 45 ° in the Roomba frame, what would be the corresponding angle in the Earth frame?"

### 2.3.2 Physics

Students were given a guest lecture about the physics behind CPS fundamentals. Students were taught that vehicles are often modeled as a point-mass or a bicycle to simplify calculations. Students were then taught about the point-mass model and Bicycle model of vehicles. Students also reviewed Newton's laws, friction, and simple dynamics. Student competency was tested informally through questions and answers.

### 2.3.3 Tools

Students were introduced to various tools used by CPS researchers in the design and deployment of autonomous vehicles. Students were given a tour of Ohio State's Center for Automotive Research and one of the autonomous vehicles as shown in Figure 2. Instruction concentrated on the sensors used in the vehicle, namely the GPS systems, Laser Rangefinder and Radar systems. Student competency was tested informally through questions and answers.

### 2.3.4 Programming

Students were introduced to programming of different languages used in CPS design. Instruction concentrated on C/C++ (two popular programming languages) and MATLAB (a very high level programming language popular with engineers). Students were given a two hour lecture on programming basics, that concentrated on syntax, and commands. Training materials was taken largely from previously developed training material available at [2]. Student competency was tested throughout the project.

### 2.3.5 Scientific Process

The scientific process employed by CPS researchers, described in Section 1, was the central theme of the Obsta-

cle Avoidance Roomba project. Students were reminded throughout the project of the process and that the logical progression of the project followed this process.

## 3. PHASE 1: CONCEPTUAL MODELING

In the first phase, students are provided with details of the problem to be solved and asked to develop algorithms for each sub-part of the problem. The goal of this phase is to introduce students to the process of learning the mathematical and physical competencies described in section 2.3 and turning them into a proposed algorithm. Another goal is to introduce students to the concept of peer review.

Students are also given real life examples of autonomous vehicles performing obstacle avoidance. Student also learned about the sensors used in Roombas, giving them an understanding of the type of data that they can use for developing their algorithms. Student learning is tested continuously through simple exercises.

Once students are comfortable with the competencies required for project completion, they are split into two groups of two student each. Students then work in their sub-group collaboratively develop high level algorithms for the four parts of the problem. In order to simulate the peer-review process of scientific development, students are asked to present their algorithms for each of the problem parts to the other group and instructor for comments and feedback.

## 4. PHASE 2: SIMULATED TESTING

Once students have developed algorithms that pass the peer review process, they are asked to use computer simulations to validate their algorithms for each of the sub-problems. The goal of this phase is to provide students with hand-on experience with the simulation tools (one of the intended competencies) used by CPS researchers, in addition to introducing students to the process of simulated validation of algorithms.

### 4.1 Training and Tools

Students use the Player/Stage package, a commonly used robot simulation program. Player provides a network interface for a variety of robot hardware, such as the Roomba. Stage is a mobile robot simulator that provides handles to a variety of sensor models. The environment used within Player/Stage has been developed by the Center for Intelligent Transportation Research (CITR), and simulates the physical testbed, called SimVille, available for laboratory testing at CITR. SimVille is described later in this paper. The Player/Stage program uses the C/C++ programming language to control the simulated Roombas movements. Developed code can be transferred directly to the actual Roombas in the physical testbed.

To introduce students to the Player/Stage syntax, students are walked through the solution of the first problem. Students are then asked to convert the algorithms developed in Phase 1 into Player/Stage compatible C/C++ code.

### 4.2 Activities

Students spent approximately twenty hours programming in C/C++ with the Player/Stage environment. Their goal was to solve the remaining three parts of the problem as mentioned in section 2.2. At the end of each problem, students are asked to present their code (to emulate the code review process), and simulations to other students and the project lead (for peer review).

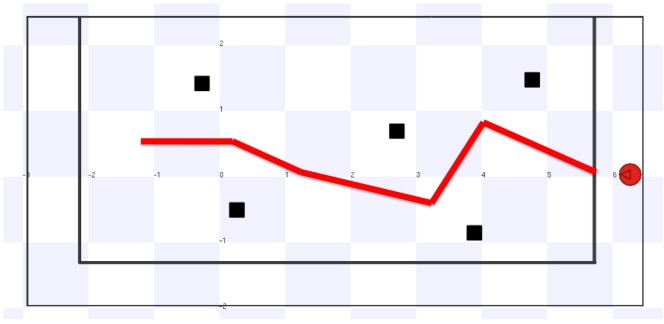

**Figure 3: Simulated Version of Third Part of Project (Red Circle represents the Roomba, Black Squares Represent Obstacles)**

```
if(x < 1.5 & y >= 1 & horizontal >= 0 & vertical
    >= 0) // Left X, Above Y, Left mote, Above
    mote
  {
  robot.Read();
  ControlPoint *mote = new ControlPoint(mote_x +
      .5, mote_y);
  mote->setXandY(mote_x + .5, mote_y);
  act(mote, x, y, yaw, pp, gps, setsteer, robot,
      target_x, target_y, mote_x, mote_y,
      target_gps, mote_gps,          matters,
      target_to_roomba, target_to_mote, distance)
      ;
```

**Figure 4: Code Sample of Group 1**

In order to promote the development of robust algorithms and code, students try to provide conditions that may "break" the code or algorithm. Further, the instructor may modify the environment, such as moving the obstacle. The aim of this task is to teach students the fundamentals of designing robust code, and the fact that research and development is an iterative process. Once students and instructor are satisfied with the robustness of code and algorithm, they proceed to the next part of the problem. An example of the simulated output for Problem 4 is given in Figure 3. For this part, students use Player/Stage to program the Roomba (red circle) to pass through all the Obstacles (black squares).

### 4.3 Difference in Student Approaches

Students were encouraged to use the the math and physics they were taught to develop their own algorithm for controlling the Roomba. For example, in the third part of the project, where students were asked to program a Roomba to reach a target by avoiding a single obstacle, one group approached the problem by coding a set of switch statements that would move the Roomba in a deterministic manner depending on which quadrant of the screen the obstacle is in.

Another group programmed the Roomba to travel to a fixed point away from the obstacle. The two code samples of Figures 4 and 5 show one such difference in approach.

## 5. PHASE 3: CONTROLLED ENVIRONMENT TESTING

Once students have sufficiently robust code, they are asked

```
if(y>=v1y && obsy == v1y){//looks at wheter the
    robot is above or below the target. Since the
    target is assumed to be within the walls, the
    robot knows which wall it is closer to. The
    check with the obsticle is to prevenmt the "
    zone of death", where the robot repeatedly
    turns around and around because it is right on
    te axis.
        yaw-=80*M_PI/180;//turns the robot to avoid
            the top wall.
}
else {
    yaw+=80*M_PI/180;//turns the robot to avoid the
        bottom wall, if it's closer to it than the
        top.
}
    robot.Read();
double dist = sqrt((xLoc[i-1]-xLoc[i-2])*(xLoc[i-1]
    -xLoc[i-2]) + (xLoc[i-1]-xLoc[i-2])*(xLoc[i-1]-
    xLoc[i-2]));
if(i>=2 && i%5 == 0 && dist<0.2 && dist>0.0){//
    check if the robot is stuck.

            if(y<v1y) yaw+=45*M_PI/180;//turn
                around the robot and back it out of
                a sticking point.

    else yaw-=45*M_PI/180;

}
  }
```

**Figure 5: Code Sample of Group 2**

to use a controlled representation (laboratory setting) of the real world - a testbed, called SimVille. SimVille was created in 2007 in order to expedite research efforts in urban environment scenarios. SimVille [12] is a 1/7 scale road network that is designed to provide easy access to a road network. Additionally ceiling mounted cameras provide a "Virtual GPS" system that robots (Roombas) using SimVille can get information about their location, location of obstacles, etc. The Player program works as the network interface to communicate between Roombas, GPS sensors, and control code. Control code is a slightly modified version of code written for the Stage simulator. A sample configuration of SimVille is given in Figure 6. The goal of this phase is to introduce students to the concept of a controlled (laboratory) environment, and illustrate the parallels between code developed for the simulated environment and code needed for the testbed environment. Students also learn the difference between these settings. For example, some of the Roomba speed values used in the simulated environment cannot be physically realized in the testbed environment because of physical limitations of the Roomba motors, and floor material which are approximated in the simulator.

## 5.1 Presentation and Tools

Students are given multiple tours of the testbed, over the course of the algorithm development and simulator proof-of-concept phases, to give them an understanding of the lab capabilities, and physical constraints. Additionally, students are allowed to *play* with the roombas to understand the dynamics of movement. Students are shown how to translate code written in Stage to the actual Roombas, through a varied compilation technique. Student are also given a detailed view of the sensors present in the testbed.

Once students have programmed their algorithms for the four problems, they are asked to implement their code on the testbed. Students are able to modify the code they have written for Stage and use their algorithms on the actual Roombas.

**Figure 6: Testbed**

## 5.2 Activities

While students are testing their code with the Roombas in SimVille, SI staff introduce perturbations to the environment by moving obstacles to give students an understanding of working with moving obstacles. Students are asked to modify their code, if necessary, to solve any issues that are brought about by these perturbations. Students complete their testing for all four parts of the problem by the end of the second week

# 6. PHASE 4: REAL WORLD DEPLOYMENT

While deploying the developed code on an actual vehicle is not feasible, given the short time span, students are introduced to this important phase through tours of several laboratories, including the Center for Automotive Research (CAR), which houses the OSU Autonomous Vehicle. Students also have an opportunity to visit other laboratories at The Ohio State University such as: a Bio-dynamics Laboratory, and a Virtual Reality Laboratory. The goals of these laboratory visits is to show students the practical aspects of their work. The goal of this phase is to show students the final phase of the CPS research process - real world deployment.

# 7. CLOSING AND FEEDBACK

At the end of the two-week period, students are asked to present their research, and results to all of the SI students, Ohio State University faculty, SI Staff, and parents. Students are also asked to give their feedback, and any suggestions for future SI programs.

## 7.1 Participant Feedback

Upon completion of the Summer Institute program in 2010 and 2011, students were asked for their feedback of the project, and suggestions for how to improve the project in

subsequent years. Excerpts from their comments are given below:

#### 7.1.1 Student 1

"*For the past two weeks, I have been enrolled in the Obstacle Avoidance Roomba project at the OSC Summer Institute. I have found it both informative and entertaining, and strongly urge that it be offered again next year. Learning C and the rudiments of autonomous-vehicle programming is engaging, and I enjoy the relaxed and informal working environment. However, I felt that the competitive arrangement of two pairs of programmers against one another was somewhat counterproductive: we probably would have been able to accomplish considerably more if we had pooled all of our resources.*" Video feedback can be found at [4].

#### 7.1.2 Student 2

"*At first I was really just looking forward to an easy project with roombas since I've used them before. Not using any other sensors was a bummer, but I realize now that it'd be impossible to incorporate the sensors in our time frame. I liked working in a big group better than in our teams of 2, because it got really competitive at times. My favorite parts of the project were when the program actually worked when compiled and driving the roombas at Dreese labs. I've actually learned a lot about psuedocode, high level coding, and different approaches to obstacle avoidance and using the GPS sensors. Overall I really enjoyed working on the project, and it's definitely the highlight of my summer.*"

#### 7.1.3 Student 3

"*During the past two weeks I had lots of fun learning about Roombas, C Programming, and Quake 3. I thought that the pace of the project was slow enough that I did not feel rushed, yet fast enough to allow us to be productive. I think that it might have been interesting to replace the multi-obstacle lab with a lab having to do with sensors. We talked a lot about the importance of sensors, so I was a bit disappointed to discover that we would not be using them in our project. The camp was extremely fun because of all of the participating students and staff. This was probably one of the best two weeks I have had in a while.*"

#### 7.1.4 Student 4

"*When I was first assigned this project, my head was intrigued by the possibilities of what we could program the Roomba to do. Shortly after we started programming, however, we ran face to face to the difficulties of using C to tell the Roomba where and how to move. Project Obstacle Avoidance Roomba is an great assignment to enlighten those unfamiliar with Autonomous Vehicles. The Roomba Project illustrates the challenges of avoiding the walls, moving around the obstacles, even turning the Roomba. Although I had no part in writing the functions of controlling the Roomba, there was plenty of work of just coding in an algorithm the robot can follow to drive itself. Obstacle Avoidance Roomba Project would be a great stepping-stone to help interested newcomers step into the field of Autonomous Driving.*"

### 7.2 Lessons and Future Projects

The SI Staff learned many valuable lessons from students, and will use these to provide improved iterations of the project in subsequent years.

#### 7.2.1 Scope

Students particularly enjoy the coding components of the project. The scope of the project is sufficient for students to have an understanding of CPS fundamentals without excessive training. Coding training given in C/C++ and is intended to help students with the Player/Stage programming. Basic mathematical training in coordinate transformations, homogeneous transformations in addition to basic physics is also provided to give students further understanding of the Roombas. Students expressed interest in using a larger set of sensors (as opposed to purely GPS coordinates). Inclusion of further sensors may be difficult within the 2 week length of the program.

#### 7.2.2 Competition

Students at times felt that the competitive process was counterproductive and that they would rather that component not be in the next iteration of the project. Students indicated a preference for group based work - something they felt would be more productive. This modification was made in SI 2011 and groups worked in a collaborative manner while maintaining the peer review process. Students were also allowed to switch groups based on approaches or areas of interest. The collaborative group structure was more successful than the competing group structure.

#### 7.2.3 Working with varying student capabilities

One of the difficulties faced by the project teams was instructing students with differing technical capabilities, especially in knowledge of programming. In SI 2010 students were paired such that each group would have one student proficient in programming working alongside a student who was not as familiar with programming. With student feedback that they felt this process required them to "carry" another student, in SI 2011, an additional instructor (who was also a SI 2010 student) was brought in to give personalized attention to students who required technical help - so as to ensure proficient students were not slowed down.

#### 7.2.4 Metrics to Judge Project

One metric used to judge program success in SI 2010 and 2011 is student willingness to participate in future robotics related activities after the Summer Institute program. Another metric, student learning of competencies, was tested through informal systems. In future iterations, the authors wish to devise a formal evaluation to judge student competencies.

#### 7.2.5 Project Evaluation

In SI 2010 and 2011, evaluation was collected through a daily online journal that asked students the following questions:

1. What did you learn today?
2. Who did you help out today and how?
3. Who helped you out today and how?
4. What did you like best about today's activities?
5. What did you like least about today's activities?

Such information was collected over eight days culminating in a final comprehensive survey about the overall experience. Data was also collected about student-instructor interactions and improvements that could be made to the overall program and particular project. In the final survey, 100% of the students "Strongly Agree" that the instructors were helpful. Multiple students cited that the "High Point" of the experience was in the Roomba testbed. One student cited a "Low Point" when they had difficulty with the coding component of the project. Overall, 75% of the students felt that the programming portion of the project was enjoyable with 25% feeling that they needed greater prior programming experience. Students were also asked to comment about the experience such as lab tours, residence halls, etc. As a measure of being taught in a way that corresponds to learning style, 36% said "Strongly Agree", 43% said "Agree" and 21% said "Neutral." 100% of the students said that the project has deepend their desire to work in the field of robotics or engineering.

## 8. TOOLS AND RESOURCES

The aim of this paper is to present a CPS related study that can be recreated. This section outlines the tools/material used and basic instructions on creating a similar project.

- The first step is to download and install the Player/Stage project from `http://playerstage.sourceforge.net`. Download and installation instructions for a variety of hardware configurations is included in the instruction manual.
- The programmable roombas used in the testbed component of the project can be purchased from `http://store.irobot.com`. The Player/Stage simulator can be configured to work with this Roomba.
- To view and download the SI 2011 Source Code: `http://dl.dropbox.com/u/1268613/codesamples.zip`
- To view and/or download the SI 2010 Student Presentation and Videos: `http://dl.dropbox.com/u/1268613/RoombaProject.zip`

Other related links and resources:

1. Student Feedback Video: `http://www.youtube.com/watch?v=Ke8ONfF-Q64`
2. NSF CPS Program: `http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=503286`
3. C/C++ Training Material: `http://www.osc.edu/supercomputing/training/`

## 9. CONCLUSIONS

We present a two week educational program for High School students as a part of the Summer Institute program at the Ohio Supercomputer Center. Students are introduced to CPS related fundamentals, and develop the algorithm and code for an obstacle avoidance Roomba. Students are taught the scientific process of moving from simulated to real world testing, and are taught the CPS core competencies of mathematics, physics, programming languages, and other tools. Additionally, students are introduced to the concepts of peer review, and iterative development. Student feedback showed that students greatly enjoy the program, and students indicate interest in future participation in CPS related research activities. We also give the reader the tools and resources required to recreate the summer educational program.

## 10. ACKNOWLEDGMENTS

This work was supported by a National Science Foundation Grant. The authors would also like to thank Paul Sivilotti, Keith Redmill, Scott Biddlestone, Arda Kurt, and Micheal Vernier for their support in developing the education activities.

## 11. REFERENCES

[1] Ohio supercomputer center, si 2010 project listing: http://www.osc.edu/education/si/projects/index.shtml.

[2] The ohio supercomputer center, training material: http://www.osc.edu/supercomputing/training/.

[3] The player/stage project: http://playerstage.sourceforge.net/.

[4] Upper arlington student discusses osc's 2010 summer institute and computational science: http://www.youtube.com/watch?v=ke8onff-q64.

[5] J. Kuenzi. Science, technology, engineering, and mathematics (STEM) education issues and legislative options. In *Library of Congress, Washington DC Congressional Research Office*, 2006.

[6] B. Lowell, H. Salzman, H. Bernstein, and E. Henderson. Steady as she goes? Three generations of students through the science and engineering pipeline. In *Annual Meetings of the Association for Public Policy Analysis and Management Washington, DC on November*, volume 7, pages 9–10, 2009.

[7] L. Perez-Felkner, S. McDonald, and B. Schneider. What Happens to High-Achieving Females after High School? Gender and Persistence on the Postsecondary STEM Pipeline.

[8] P. Sivilotti and M. Demirbas. Introducing middle school girls to fault tolerant computing. In *Proceedings of the 34th SIGCSE technical symposium on Computer science education*, pages 327–331. ACM, 2003.

[9] P. Sivilotti and M. Lang. Interfaces first (and foremost) with Java. In *Proceedings of the 41st ACM technical symposium on Computer science education*, pages 515–519. ACM, 2010.

[10] P. Sivilotti and S. Pike. The suitability of kinesthetic learning activities for teaching distributed algorithms. *ACM SIGCSE Bulletin*, 39(1):362–366, 2007.

[11] W. Tyson, R. Lee, K. Borman, and M. Hanson. Science, Technology, Engineering, and Mathematics (STEM) Pathways: High School Science and Math Coursework and Postsecondary Degree Attainment. *Journal of Education for Students Placed at Risk (JESPAR)*, 12(3):28, 2007.

[12] M. Vernier. Virtual sensor system: Merging the real world with a simulation environment, 2010.

# Using Supercomputing to Conduct Virtual Screen as Part of the Drug Discovery Process in a Medicinal Chemistry Course

David Toth
University of Mary Washington
1301 College Avenue
Fredericksburg, VA 22401

dtoth@umw.edu

Jimmy Franco
Merrimack College
315 Turnpike Street
North Andover, MA 01845

jimmy.franco@merrimack.edu

## ABSTRACT
The ever-increasing amount of computational power available has made it possible to use docking programs to screen large numbers of compounds to search for molecules that inhibit proteins. This technique can be used not only by pharmaceutical companies with large research and development budgets and large research universities, but also at small liberal arts colleges with no special computing equipment beyond the desktop PCs in any campus' computer laboratory. However, despite the availability of significant quantities of compute time available to small colleges to conduct these virtual screens, such as supercomputing time available through grants, we are unaware of any small colleges that do this. We describe the experiences of an interdisciplinary research collaboration between faculty in the Chemistry and Computer Science Departments in a chemistry course where chemistry and biology students were shown how to conduct virtual screens. This project began when the authors, who had been collaborating on drug discovery research using virtual screening, decided that the virtual screening process they were using in their research could be adapted to fit in a couple of lab periods and would complement one of the instructors' courses on medicinal chemistry. The resulting labs would introduce students to the virtual screening portion of the drug discovery process.

## General Terms
Supercomputing, Computational Chemistry Education, Drug Discovery, Medicinal Chemistry

## Keywords
Docking, Virtual Screening, AutoDock Vina, PyMOL



## 1. INTRODUCTION
Identifying novel chemotherapeutics has become increasingly challenging and expensive. For every 10,000 compounds evaluated in animal trials, only 10 will make it to clinical trials. The average cost to bring a drug to market is estimated to be about 800 million dollars [1]. Thus the need for more efficient methods of identifying compounds has become increasingly important. One of these methods is virtual screening. The increasing amount of available computing power and the number of protein structures that have been solved have made this an increasingly attractive approach. As of April 10, 2012, there were 80,710 structures in the Protein Data Bank (PDB), which offer a plethora of possibilities for conducting virtual screens [2]. The number of solved structures will only increase as thousands of structures are deposited annually in the PDB.

Many of the chemistry and biology curriculums lack sufficient computational instruction to prepare the next generation of scientists. Proficiency in computational science has become increasingly important. Many industrial companies including big pharmaceutical companies such as Pfizer, Genentech, Eli Lilly & Co and Johnson & Johnson have begun using methods like virtual screening to improve their efficiency in the drug discovery process. Thus student graduating with experience using computational tools and methods will be much more employable.

In this paper we describe our experiences with students using a supercomputer to conduct a virtual screen using AutoDock Vina to identify inhibitors for a number of diseases [3]. The docking program calculates the binding affinity of each of the compounds in a library of compounds specified by the user. The compounds are sorted by binding affinity using Microsoft Excel and subsequently the top hits can be visualized in PyMOL [4]. Visualizing the compounds in PyMOL allows the student to confirm that the predicted binding conformation would induce the required inhibitory affect. The project described here can be incorporated to into a large drug discovery project. The compounds identified as hits from the docking

program could subsequently be screened in a wet laboratory.

Having undergraduate students work on drug discovery in an academic environment is now feasible with the minimal computational power available on any campus and the supercomputer time one can obtain with grants. This type of applied project stimulated interest amongst our students, as they were able to envision what the impact of the project would be if they found a good potential inhibitor. This project also allowed us to highlight the interdisciplinary nature of the modern drug discovery process, which relies on computer science, chemistry, and biology. The project outlined here creates a platform for a drug discovery research project.

## 2. RELATED WORK

While there have been several articles published about using virtual screens in a curriculum, none to our knowledge have used supercomputing [5]. An advantageous attribute of this project is the ability to adapt this project to any number of diseases or disorders. Along with a large variety of targets the concept of using super computing power can be adopted to a wide variety of simulations and modeling programs [6, 7].

A recent article by Sutch et al. described an activity focused on a structure based drug design [8]. One of the programs used in that activity to conduct the virtual screen was MEDock, which is a simple docking program. Unfortunately, this simplicity also imposes several imitations on MEDock's versatility. It only allows for areas of 300 atoms to be evaluated in a virtual screen. It also limits the number jobs that can be submitted. The largest problem with this program is that it perpetuates the black box thinking of virtual screening. Students need much less insight into the program to be able to successfully screen compounds, thus requiring less understanding of the science behind the project. Also in the project described herein students use PyMOL, which is a commonly used program for visualizing macromolecules in both academia and industry.

Other articles describing small molecule interactions with drug targets have focused on the specifics of the compounds' conformations and chemical properties in relation to protein, but do not address the greater issue of the drug discovery process or virtual screening [9, 10]. None of the previous lab activities we have found in the literature required the student to engage in the computer science aspect as much as this activity does. Most of the activities used programs that are less versatile but have a graphical user interface. While this can be very advantageous for large classes, it does allow the students to conduct the exercise with out much understanding of the docking program.

## 3. THE GOALS AND ACTIVITIES

There were a number of goals for the labs, including getting students to learn the important role that computers can play in the drug discovery process. Students were also supposed to learn how to use a docking program, gain experience using software other than the commercial off-the-shelf software they use on a daily basis, and get exposure to the Linux operating system and a command line interface. Other goals were to gain an appreciation for how much supercomputers can speed up the virtual screening process and understand that supercomputing time can be obtained at no cost even by small academic institutions that do not have the financial resources to buy a supercomputer or time on a supercomputer. Students also were shown how to use PyMOL, a protein visualization program. Finally, students also were shown some new data analysis skills with Excel.

Students learned a little about supercomputers as part of the lab. The most striking thing that students learned is that the virtual screening process is significantly faster using a supercomputer, because they can screen many molecules at once, rather than only a few at a time, when using a single CPU core per molecule. They also learned that as opposed to desktop computer or a server where you can just start tasks whenever you want, on a supercomputer, you must submit your task to the queuing system and the queuing system controls when your task is run. The students learned that the queuing system using a number of factors to determine when a task should be run, including the number of CPU cores needed and the amount of time requested for the task. Therefore, they understood that while using more CPU cores might finish the virtual screen faster once the task was started, requesting many more resources might delay when the task was started and could ultimately result in the virtual screen being completed later than if they requested fewer CPU cores. Students also learned how to check the queue on the supercomputer to see whether their task was waiting or being run. Figure 1 shows a screen capture from the lab manual where the students would check the queue.

The laboratory was conducted in three phases over the course of two days due to the laboratory time being 1 hour and 15 minutes. However, we note that the activity would fit very well in a traditional 3 hour laboratory. Prior to the laboratory students had to choose a protein that was known to be a good drug target from the PDB. To identify a protein student conducted a literature search using SciFinder, PubMed or Google Scholar. Students were instructed to identify a protein that had been previously shown to be a good drug target, either through chemical inhibition, knockout study, or methods that demonstrated the proteins potential as a drug target. Secondly, students had to very verify that 3D structure had been solved. This was easily done by searching the PDB site for the structure. With only these two constraints, students have a large number of targets to choice from. The variability of the drug target selection was purposely done was to allow the students to take ownership over the project as well to force the student to think critically about the target. The

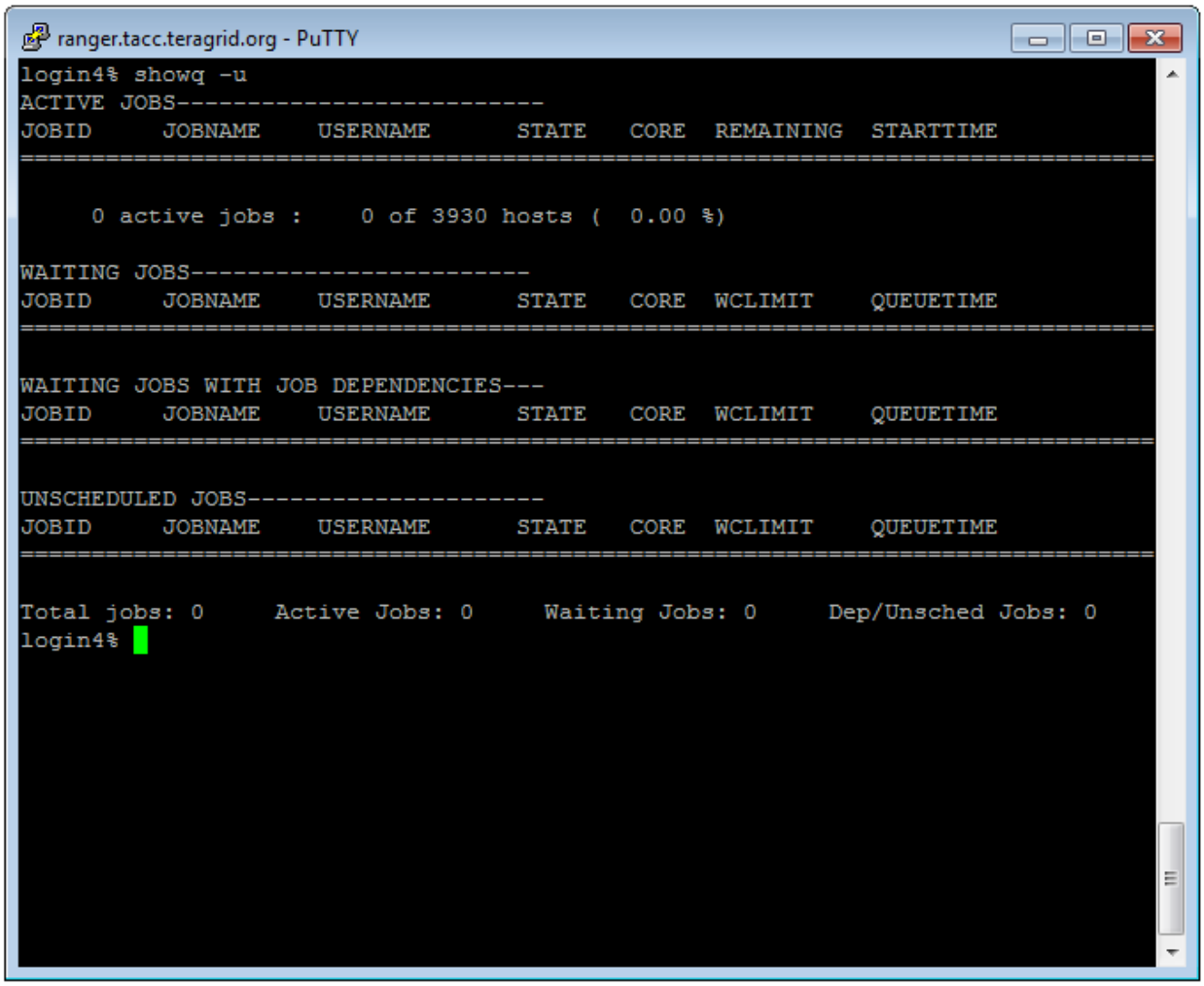


Figure 1: A screen capture from the laboratory manual showing the output from querying the supercomputer's queue.

structures were then converted to pdbqt files (the file format that AutoDock Vina uses) and the search grid was set using AutoDock Tools [11]. Although students were required to choose a protein, convert it, and find the search grid, three pre-converted proteins structures and search grids have been included in the supplemental materials with this paper (Sample_Targets.doc) for readers wanting to test the lab without having to first find a protein, convert it to a pdbqt file, and find the search grid. Those three proteins are targets for Alzheimer's disease, Cancer, and HIV.

On the first day, students carried out Phase one of the laboratory. Before beginning the laboratory, students were given a 26-page full-color laboratory manual that included numerous screen captures to help them with the laboratory. Figures 2 and 3 show portions of the lab manual showing the students how to lo on to a remote computer with ssh and scp. The lab manual has been included in the supplemental materials with this paper (Laboratory_Manual.docx). Students began the laboratory by downloading a protein pdbqt file from the course Blackboard site and then a secure shell (ssh) program and a secure copy (scp) program from the Internet. Next, each student was given a distinct username and password to log onto a server on campus. Students logged on to the server, transferred the protein file to the server, and using the docking program, tested how well the protein bound to a potential drug molecule. We showed the students how to use AutoDock Vina, an open source software package from the Scripps Institute [3].

Students continued working on the laboratory on the second day by starting with Phase two. In Phase two, students created a shell script to automate the screening of multiple compounds. While Phase two of the laboratory manual included instructions to perform the analysis of the data from the virtual screening, to save time, students were assigned to do that portion of the laboratory at home. Students then did Phase three of the laboratory. In Phase three, students logged on to a supercomputer located across

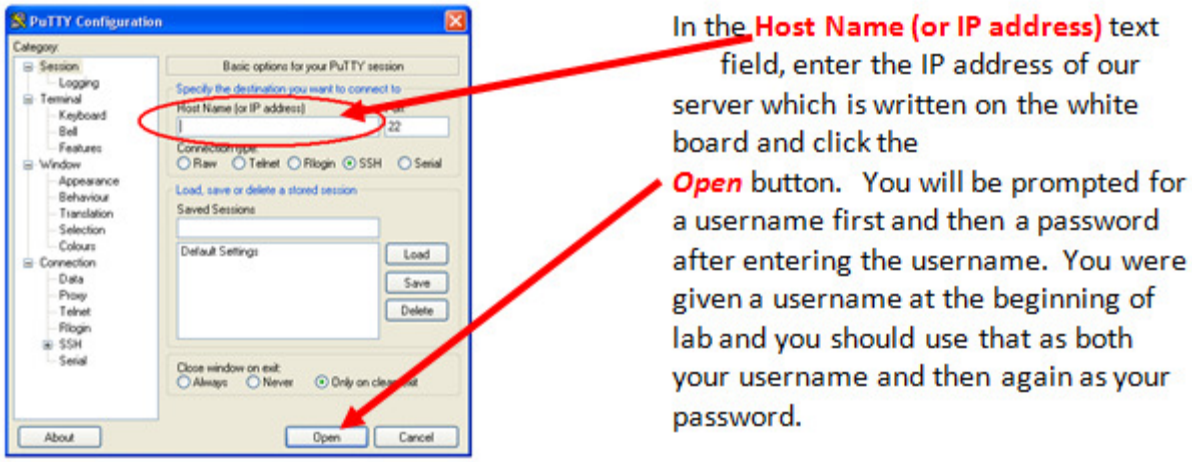


Figure 2: Portion of the laboratory manual showing how to log on to a supercomputer or a remote server using ssh.

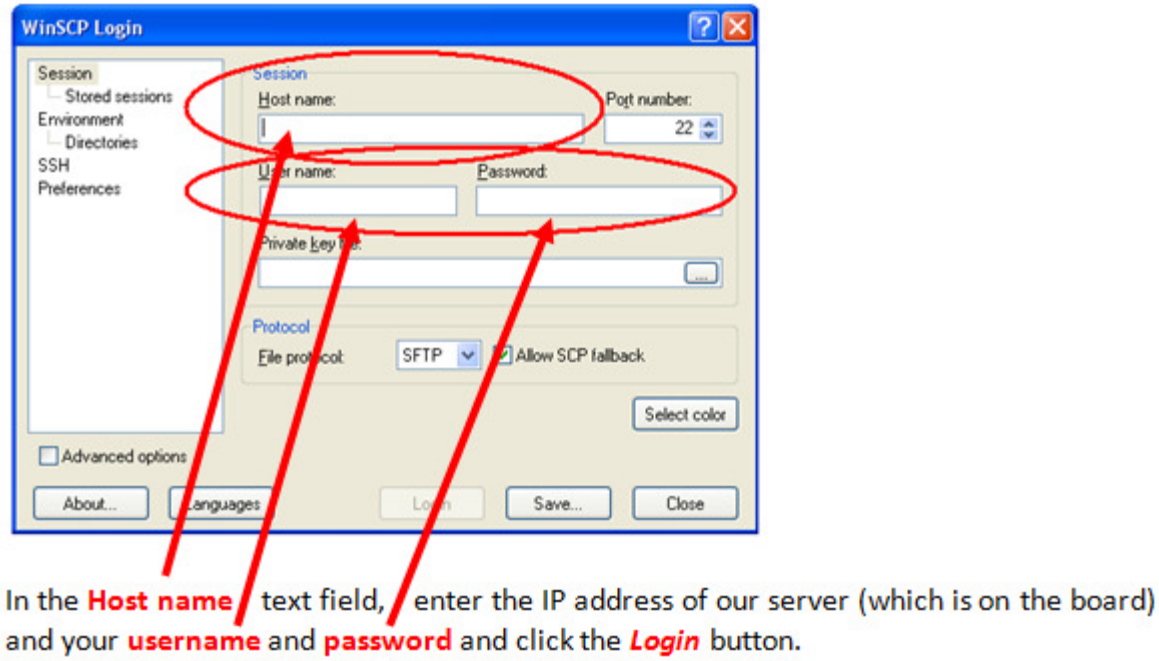


Figure 3: Portion of the laboratory manual showing how to log on to a supercomputer or a remote server using scp.

the country and uploaded a protein file. Finally, they edited a shell script that they could use to automate the virtual screening, and submitted the job to the batch scheduling system on the supercomputer.

Finally, the students were asked to complete their projects in their assigned groups. The output files from the virtual screens were posted on blackboard for the convenience of the students. The results of each group's screen were posted as text files. Each group converted their text file to an Excel file so they could quickly identify the top binding affinity compounds; those compounds are termed hits.

AutoDock Vina identifies several binding conformations for each of the compounds screened and outputs those values, as shown in Figure 4. The compounds the students screened came from the ZINC database (http://zinc.docking.org/pdbqt/) [12]. However, the conformation with the best binding affinity is the one most likely to occur, so the students were supposed to remove the data for the other conformations of the same compound from the data. Using Excel, the students were able to remove the extra conformations for each compound and sort the remaining data to quickly identify the compounds with the best binding affinity. The hits are subsequently visualized in PyMOL. An example is shown in Figure 5.

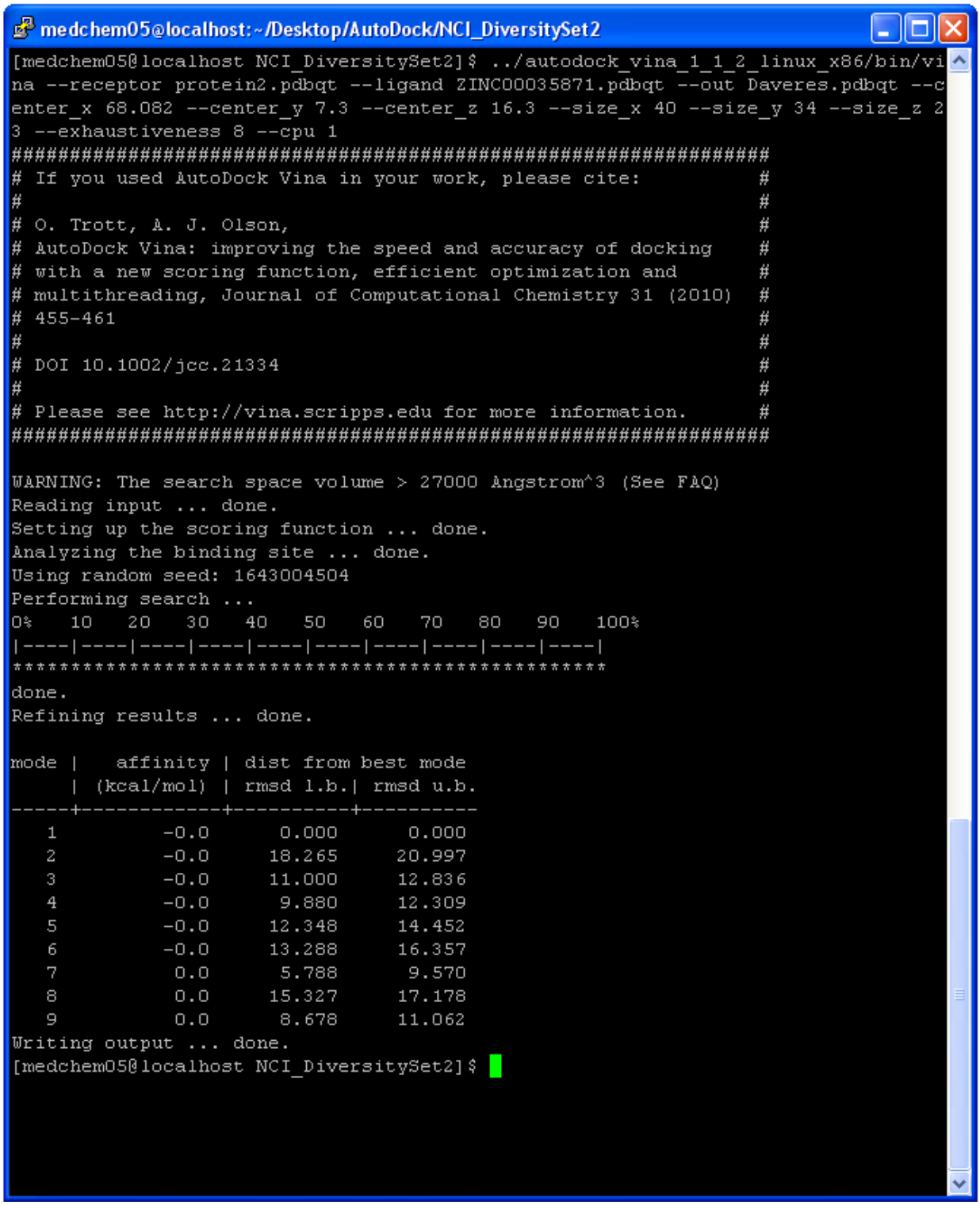


Figure 4: Portion of the laboratory manual showing the output of AutoDock Vina after screening a molecule.

A handout with commands for PyMOL is included in the supplementary materials with this paper (PyMOL_Commands.doc). Visualization of the binding conformations with the protein is important since a compound may have a high binding affinity for a protein, but may not inhibit its activity. When visually inspecting the hits with PyMOL, students were instructed to verify that the compounds bind to the active site or to a known

allosteric site of the protein. When ranking the hits and trying to identify a few lead compounds to pursue, if specific details about the protein are known, such as, if a residue is essential to the protein's function, then any compounds interacting with these residues should be given extra consideration.

During the evaluation of the hits in PyMOL, students had to critically evaluate how the potential inhibitor was predicted to bind to the targeted protein. Two main aspects were focused on during the evaluations of binding: orientation and the interactions between the compounds and the protein. First, did it bind in a manner that would inhibit enzymatic activity? Typically this could be determined by it binding to either the active site or a known allosteric site. Also compounds predicted to interact with residues previously shown to be important were especially noteworthy. Secondly, students examined what interactions the compounds have with the targeted protein, such as hydrogen bonding, ionic interactions, and hydrophobic interactions, as these interactions determine its binding affinity. The students learned the relationship between how the thermodynamics of the calculated values relate to how the compounds interact with the protein. By being able to visualize the predicted binding conformations in PyMOL, students were able to see the interactions that lead to the predicted binding affinity. Compounds displaying a large number of favorable interactions displayed the greatest binding affinity. Lastly, many students fail to see the importance of understanding thermodynamics and this project allows the students to see a real world application of thermodynamics. During the lab section, students were given a brief explanation on how the docking program measures the energy between the compounds and the protein.

For this course, Medicinal Chemistry, thirteen students consisting of biology, chemistry, and biochemistry majors participated in the activity. Each group of students had to create a written report of their finding where they had to give some background about the disease, explain the function and importance of the selected target, and demonstrate that they had identified a potential inhibitor for the targeted protein. The groups were also required to present their work to the rest of the class with a PowerPoint presentation.

## 4. STUDENT REACTIONS

The students in the course were given an anonymous survey at the end of the semester. The survey is included in the supplementary materials (survey.pdf). The survey responses indicated a significantly increased awareness of the availability of supercomputing resources. The surveys also showed that the students learned how to use the software for the project, including AutoDock Vina and PyMOL, and that the students learned new techniques in Excel. The surveys demonstrated that the students became more comfortable using the command prompt and they also learned some simple UNIX commands. Because there is a significant amount of scientific software that must be run from the command prompt, increasing the students' comfort with the command prompt is very important as we try to prepare them for their future careers. The students

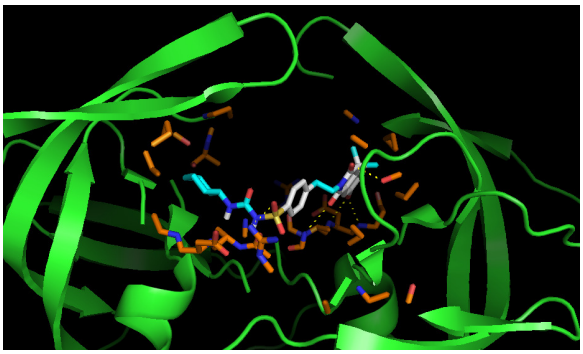

Figure 5: An image of one of the top binding inhibitors as calculated by AutoDock Vina. This is an example of an image the students will generate during the project using PyMOL.

also learned the importance of computation in science, as an alternative method of solving problems, so they understand that science can be done outside of a wet laboratory. They also understood that supercomputing could be applied to problems in other domains and would recommend its use for other projects. In addition to learning computational science techniques, students also demonstrated an increased understanding of fundamental chemistry concepts.

The students in the class reacted very favorably to the laboratory. All of the students felt that the laboratory manual was easy to follow. At the end of the course, over one third of the students expressed a desire to continue working on the projects and in particular, work more on the computational aspect of the project and conduct virtual screens of more compounds to try to find more potent inhibitors. These students, who were completing their Junior year, will be working on directed research projects related to the course projects in the upcoming year. A large percentage of the graduating Seniors also expressed that they would have continued working on the projects if they were not graduating, and several said they might be willing to come back over the summer to continue until they had found jobs. One student commented that she thought the computing aspect of the project was "extremely interesting and educational." Because this student was not very comfortable with using computers for science to begin with, we found that this was very encouraging.

## 5. INSTRUCTOR REACTIONS AND LESSONS LEARNED

It took 2.5 hours spread over two days for students to complete the laboratory activities other than the data analysis portion. The students were able to complete the

Phase one activities in 1 hour, with each student working on their own computer. During the first day, although the students were given the laboratory manuals, the instructor showed the students how to do the tasks on a computer where the screen was projected at the front of the classroom. This was done to try to get the students more comfortable with some of the tasks that they might be less familiar with. On the second day, students worked in their project groups, with one group per computer and the students were told to follow the laboratory manual's instructions but to feel free to ask questions whenever they had any trouble. Because of the detail of the laboratory manual, which included numerous screen captures, this worked well. The instructors felt that the second day went smoother than the first day and believed that forcing the students to follow the laboratory manual rather than having one instructor demonstrating the tasks at the front of the room worked very well. However, it may have been important to help the students get comfortable with the tasks during the first day of the laboratory by having them watch the instructor rather than having them simply follow the instructions on the laboratory manual.

During the process of the students downloading the ssh client and the scp client, we learned that although the laboratory manual was very detailed, the students tended to have difficulty entering URLs correctly. As we progressed through the laboratory, we discovered that the same idea held for places where the students needed to type commands. Instructors using this lab should be aware of the difficulties that the students had with entering URLs and commands so they are prepared for the inevitable questions about why something does not work.

One issue that we did not anticipate was the amount of time that it took the server the students used in Phase one and Phase two of the laboratory to run the virtual screens. The server used was a dual-core desktop computer that was 4-5 years old and while it did the processing quickly enough during our testing of the laboratory materials before giving the laboratory to the students, multiplying the tasks the computer needed to do by 13 proved to be too much for the computer to handle gracefully. While it completed all the tasks, it was slow enough in Phase one of the laboratory that we put the students into their project groups for Phase two and Phase three. We recommend that others use a computer better equipped to handle the computational demands of the number of students.

There are a few suggestions that we have for other instructors who will use this module when teaching their courses. The number of students in each group in our class ranged from 2 to 5 students. The groups were allowed to divide the work up as they wanted. In the smaller groups, it appeared that all the student were extremely active. In the larger groups, the amount of work varied vastly between students. Thus, one suggestion we have is limiting the size of the student groups to 2-3 students. The instructor may also want to load the protein files that students will use and the shell script onto the server before the laboratory, if they want to shift the focus more towards the chemistry aspect and minimize the computer science portion of the project.

The instructor may want to mention before the laboratory that not every command that the students enter will result in significant visual output in the command window, as this confused several of our students.

## 6. CONCLUSIONS

We were able to develop a hands-on laboratory project that allows students to gain valuable experience in computational science with real-world applications. We have been able to present the material in a manner that engaged the students and stimulated interest in computational science research. Because all of the software is open source and all the required resources beyond what exist in any college computer lab are freely available through grants, this project can be done at schools of any size at no cost. The project can be run as a pre-packaged standalone laboratory assignment just to introduce students to virtual screening and computational chemistry or as a large semester-long project. If the project is run as a full course project, it could be used to prepare students for directed research projects in drug discovery and senior thesis work. Since computational science has become a more substantial part of a number of scientific disciplines, this project could be used as a model to develop other computational science lab projects.

## 7. FUTURE WORK

In continuing the development of this project, we will extend it to a full semester project. In the full semester project, students would be required to use the information gathered from evaluating the hits in virtual screen to generate a second generation of inhibitors. Thus, when students are evaluating the hits in PyMOL they will have to identify any additional interaction that can be utilized by an inhibitor. This could be done by adding and additional hydrogen bonding acceptor or donor, creating a hydrophobic group to utilize a hydrophobic pocket, or removing a group that is creating an unfavorable steric effect. This deals with the properites a compounds should posses to be a more likely drug candidate. Once the students have designed a set of compounds they will virtually construct them using Jmol (http://jmol.sourceforge.net/) and Open Babel (http://openbabel.sourceforge.net/), which are both open source software. The compounds will then be re-screened to indentify which chemical modications had the largest effect on the binding affinity. Lastly students would be asked to propose a synthesis for their top three hits.

In future work, we will create an electronic lab kit, containing a step-by-step laboratory manual and either all the files or links to all the files, depending on licensing restrictions, that instructors would need to recreate the laboratory at their own institutions.

## 8. ACKNOWLEDGEMENTS

This work used the Extreme Science and Engineering Discovery Environment (XSEDE), which is supported by National Science Foundation grant number OCI-1053575.

We wish to thank the Extreme Science and Engineering Discovery Environment (XSEDE) program, which supported this work by providing the supercomputer time through grant TG-MCB120071. We also wish to thank the Texas Advanced Computing Center (TACC), which provided the supercomputer we used for this work. We would also like to acknowledge the students in the course for their participation in this project.

## 9. REFERENCES

[1] Silverman, Richard B. The Organic Chemistry of Drug Design and Drug Action. 2nd ed. Amsterdam ; Boston: Elsevier Academic Press, 2004.

[2] RCSB Protein Data Bank http://www.rcsb.org/pdb/home/home.do.

[3] Trott, O.; Olson, A. J. *Journal of Computational Chemistry* **2010,** *31*, 455-461.

[4] The PyMOL Molecular Graphics System, Version 1.2r3pre, Schrödinger, LLC. http://www.pymol.org/

[5] Baudry, J.; Hergenrother, P. J. *Journal of Chemical Education* **2005,** *82*, 890.

[6] Artavanis-Tsakonas, K.; Weihofen, W. A.; Antos, J. M.; Coleman, B. I.; Comeaux, C. A.; Duraisingh, M. T.; Gaudet, R.; Ploegh, H. L. *The Journal of biological chemistry* **2010,** *285*, 6857-66.

[7] Sotomayor, M.; Weihofen, W. A.; Gaudet, R.; Corey, D. P. *Neuron* **2010,** *66*, 85-100.

[8] Sutch, B. T.; Romero, R. M.; Neamati, N.; Haworth, I. S. *Journal of Chemical Education* **2012**, 89 (1), pp 45–51.

[9] Yuriev, E.; Chalmers, D.; Capuano, B. *Journal of Chemical Education* **2009**, 86 (4), p 477.

[10] Manallack, D. T.; Chalmers, D. K.; Yuriev, E. *Journal of Chemical Education* **2010**, 87 (6), pp 625–627.

[11] AutoDock Vina – molecular docking and virtual screening program http://vina.scripps.edu/tutorial.html.

[12] Irwin, J. J.; Shoichet, B. K. *Journal of Chemical Information and Modeling* **2004,** *45*, 177-182.

# Metadata Management in Scientific Computing

Eric L. Seidel
The City College of New York
eseidel01@ccny.cuny.edu

## ABSTRACT

Complex scientific codes and the datasets they generate are in need of a sophisticated categorization environment that allows the community to store, search, and enhance metadata in an open, dynamic system. Currently, data is often presented in a *read-only* format, distilled and curated by a select group of researchers. We envision a more open and dynamic system, where authors can publish their data in a *writeable* format, allowing users to annotate the datasets with their own comments and data. This would enable the scientific community to collaborate on a higher level than before, where researchers could for example annotate a published dataset with their citations.

Such a system would require a complete set of *permissions* to ensure that any individual's data cannot be altered by others unless they specifically allow it. For this reason datasets and codes are generally presented read-only, to protect the author's data; however, this also prevents the type of social revolutions that the private sector has seen with Facebook and Twitter.

In this paper, we present an alternative method of publishing codes and datasets, based on Fluidinfo[1], which is an openly writeable and social metadata engine. We will use the specific example of the Einstein Toolkit, a shared scientific code built using the Cactus Framework, to illustrate how the code's metadata may be published in writeable form via Fluidinfo.

## 1. INTRODUCTION

Data management is quickly becoming a challenge in large scale simulations and modeling as compute resources increase in size, and simulations integrate with observational and experimental data. Not only do these simulations produce increasingly large datasets, which must then be analyzed and categorized, but the codes themselves become more and more complex, often being developed by distributed teams. The Cactus Computational Toolkit[2] is one such software framework, comprising over 500 software modules (known as *Thorns*), of which a subset must be compiled to produce a full simulation stack.

The Cactus Thorns specify their public interface using the Cactus Configuration Language (CCL), which describes the mechanics of the thorn, but provides little semantic data. This makes it difficult to determine which of the hundreds of thorns may be needed for a particular simulation. There are two standard methods for dealing with these ambiguities:

1. Detail the semantics of every thorn in documentation within the source tree. This is somewhat helpful when a user has already downloaded the thorn in question, but it does not help a new user discover useful thorns.

2. Collect documentation and use-cases for each thorn on the main webpage for the framework. This is much more helpful to new users in search of thorns, but it raises new issues. Who maintains the website and keeps the web-based documentation synchronized with the source code? Thorns are generally maintained by individual authors, not the community, so should all authors have write access to the web server? If so, how does one prevent authors from misrepresenting each other's codes? The end user is still presented a *read-only* interface, meaning a user cannot easily annotate and recommend useful thorns to others.

In the following sections, we will describe how Fluidinfo may be used to annotate these datasets in a writeable manner, while preserving the safety and integrity of the author's original data. We aim to show that the concept of "tagging," as introduced by social networking services, is well suited to building and maintaining distributed scientific collaborations in the computational sciences. Our approach is based on loosely structured data, in contrast to other data formats used in metadata and semantic web research. Section 2 examines other approaches to similar problems. Section 3 describes the *Cactus Configuration Language*, which contains a substantial amount of Thorn metadata. Section 4 introduces Fluidinfo, the writeable metadata engine, and its core concepts. Section 5 describes specifically our strategy for

[1] http://www.fluidinfo.com

[2] http://www.cactuscode.org

publishing the Einstein Toolkit metadata to Fluidinfo. Section 6 investigates how the strategy presented in Section 5 may be adapted for publishing datasets as opposed to codes. Section 8 reflects on the educational value of this project, and the Blue Waters Undergraduate Petascale Education Program that supported it.

## 2. RELATED WORK

Before discussing our approach to solving this problem, let us examine other systems that could be used to support distributed collaboration. *RDFPeers* [3] is a distributed RDF repository designed to solve scalability issues faced by many centralized metadata stores. It uses a peer-to-peer architecture to spread metadata across many machines, and efficiently route queries to the appropriate machine. A distributed system like RDFPeers would be a natural fit for our problem, as it could encourage authors to maintain the metadata pertaining to their codes and datasets alongside the actual data; however, we feel that RDF as a data format may be excessively complex for our purposes. We believe that a simpler format based on social tagging, like that used by the Delicious bookmarking service[3], would be sufficient for our needs. In particular, RDF is based on triples of *subjects*, *predicates*, and *objects*, whereas the tagging method we describe only needs *objects* and *attributes*. Clearly we could use RDF triples with a constant predicate `hasAttribute`, but we gain little by doing so and incur additional complexity.

The *Social Accessibility* [13] project attempts to help site-owners keep up with accessibility standards by crowd-sourcing some of the work. It is comprised of three pieces: (1) a browser script with which end-users may register complaints about websites and receive patches, (2) a browser plugin to allow volunteers to investigate accessibility issues and submit patches, and (3) a server that stores the complaints and patches. When an end-user visits a website, the browser script searches the server for any applicable patches, retrieves them, and applies them to the page. The user's browsing experience is immediately enriched by the knowledge of the community with little effort on the user's part. This project appears to have a similar goal to our own, enriching content via collaborative editing, albeit applied to a different problem domain.

## 3. CACTUS CONFIGURATION LANGUAGE

The Cactus Framework [15, 7] is an open source, modular, portable programming environment for HPC computing[4]. It was designed and written specifically to enable scientists and engineers to collaboratively develop and perform the large–scale simulations needed for modern scientific discoveries across a broad range of disciplines. Cactus is well suited for use in large, international research collaborations. For example, the Einstein Toolkit Consortium [16] is a collaboration of over 60 researchers who use Cactus for research into relativistic astrophysics, and who maintain a core set of some 175 modules.

[3]http://www.delicious.com

[4]This section was adapted from a previous paper on the Cactus Configuration Language [1].

**Figure 1: Cactus components are called *thorns* and the integrating framework is called the *flesh*. The interface between thorns and the flesh is provided by a set of configuration files writing in the Cactus Configuration Language (CCL).**

### 3.1 Architecture

Cactus is a component framework. Its components are called *thorns* whereas the framework itself is called the *flesh* (Figure 1). The flesh is the core of Cactus, it provides the APIs for thorns to communicate with each other, and performs a number of administrative tasks at build–time and run–time. Cactus depends on three configuration files and two optional files provided by each thorn to direct these tasks and provide inter–thorn APIs. These files are:

- `interface.ccl` Defines the thorn *interface* and *inheritance* along with variables and aliased functions.
- `param.ccl` Defines parameters which can be specified in a Cactus parameter file and are set at the start of a Cactus run.
- `schedule.ccl` Defines when and how scheduled functions provided by thorns should be invoked by the Cactus scheduler.
- `configuration.ccl` (optional) Defines build–time dependencies in terms of provided and required capabilities, e.g. interfaces to Cactus–external libraries.
- `test.ccl` (optional) Defines how to test a thorn's correctness via regression tests.

The flesh is responsible for parsing the configuration files at build-time, generating source code to instantiate the different required thorn variables, parameters and functions, as well as checking required thorn dependencies.

At run-time the flesh parses a user provided parameter file that defines which thorns are required and provides key-value pairs of parameter assignments.[5] The flesh then ac-

[5]Note that this parameter file is different from the file `param.ccl` which is used to define which parameters exist,

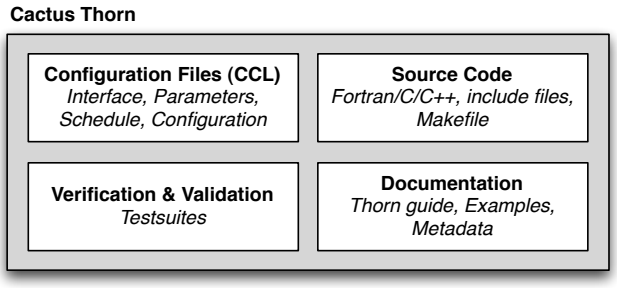


**Figure 2: Cactus thorns are comprised of source code, documentation, test–suites for regression testing, along with a set of configuration files written in the Cactus Configuration Language (CCL) which define the interface with other thorns and the Cactus flesh.**

tivates only the required thorns, sets the given parameters, using default values for parameters which are not specified in the parameter file, and creates the schedule of which functions provided by the activated thorns to run at which time.

The Cactus flesh provides the main iteration loop for simulations (although this can be overloaded by any thorn) but does not handle memory allocation for variables or parallelization; this is performed by a *driver* thorn. The flesh performs no computation of its own — this is all done by thorns. It simply orchestrates the computations defined by the thorns.

The thorns are the basic modules of Cactus. They are largely independent of each other and communicate via calls to the Flesh API. Thorns are collected into logical groupings called *arrangements*. This is not strictly required, but strongly recommended to aid with their organization. An important concept is that of an *interface*. Thorns do not define relationships with other specific thorns, nor do they communicate directly with other thorns. Instead they define relationships with an interface, which may be provided by multiple thorns. This distinction exists so that thorns providing the same interface may be interchanged without affecting any other thorns. Interfaces in Cactus are fairly similar to abstract classes in Java or virtual base classes in C++, with the important distinction that in Cactus the interface is not explicitly defined anywhere outside of the thorn.

This ability to choose among multiple thorns providing the same interface is important for introducing new capabilities in Cactus with minimal changes to other thorns, so that different research groups can implement their own particular solver for some problem, yet still take advantage of the large amount of community thorns. For example, the original driver thorn for Cactus which handles domain decomposition and message passing is a unigrid driver called `PUGH`. More recently, a driver thorn which implements adaptive mesh refinement (AMR) was developed called `Carpet` [10, 9, 4]. Carpet makes it possible for simulations to run with multiple levels of mesh refinement, which can be used to achieve great accuracy compared to unigrid simulations. Both `PUGH` and `Carpet` provide the interface `driver` and application thorns can relatively straightforwardly migrate from unigrid to using the advanced AMR thorn.

Thorns providing the same interface may also be compiled together in the same executable, with the user choosing in the parameter file, at run-time, which implementation to use. This allows users to switch among various thorns without having to recompile Cactus.

Thorns include a `doc` directory which provides the documentation for the thorn in LaTeX format. This allows users to build one single reference guide to all thorns via a simple command.

## 3.2 Tools

As a distributed software framework, Cactus can make use of some additional tools to assemble the code and manage the simulations. Oftentimes each arrangement of thorns resides in its own source control repository, as they are mostly independent of each other. This leads to a retrieval process that would quickly become unmanageable for end-users (for example the Einstein Toolkit is comprised of 135 thorns). To facilitate this process we use a *thornlist* written using the Component Retrieval Language [11], which allows the maintainers of a distributed framework to distribute a single file containing the URLs of the components and the desired directory structure. This file can then be processed by a program such as our own `GetComponents` script, and the entire retrieval process becomes automated.

In addition to the complex retrieval process, compiling Cactus and managing simulations can be a difficult task, especially for new users. There are a large number of options that may be required for a successful compilation, and these will vary across architectures. To assist with this process a tool called the *Simulation Factory* [12, 14] was developed. Simulation Factory provides a central means of control for managing access to different resources, configuring and building the Cactus codebase, and also managing the simulations created using Cactus. Simulation Factory uses a database known as the *Machine Database*, which allows Simulation Factory to be resource agnostic, allowing it to run consistently across any pre-configured HPC resource.

# 4. FLUIDINFO

Fluidinfo is an openly writeable datastore, whose goal is to extend collaborative tagging to all forms of data. Designed around the metaphor of post-it notes, it is a collection of *objects* and *tags* at its core, with a complete set of *permissions* to give users full control over their data. Fluidinfo is developed and hosted by Fluidinfo Inc., a start-up company. This section will give a brief overview of the basic concepts of Fluidinfo; a more detailed discussion may be found in the official documentation [6].

## 4.1 Objects

One of the core concepts of Fluidinfo is that objects are completely anonymous, having no owner and no inherent

while the former is used to assign values to those parameters at run-time.

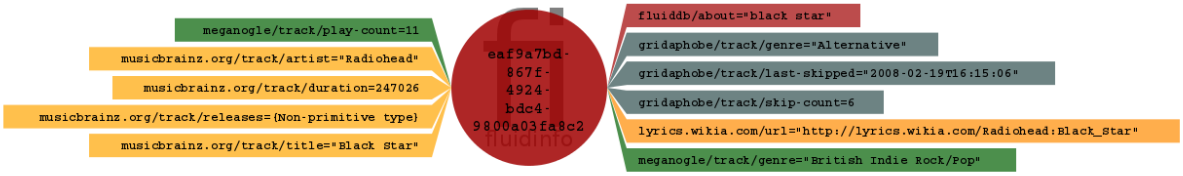


**Figure 3: Visual representation of the Fluidinfo object for the song "Black Star" by Radiohead. Note the combination of tags from a variety of users, with primitive and opaque values.**

meaning. Objects exist solely as a container for tags, which define their semantics.

## 4.2 Tags

Tags have owners and permissions, so while anyone can tag an object, tags may be read-only, read-write, or completely invisible to the outside world. When a tag is placed on an object, it may contain any value, and the type of value need not be consistent between tag-instances (although in practice this would be a good idea). Fluidinfo does, however distinguish between so-called *primitive* and *opaque* tag-values.

**Primitive tag-values** are a subset of the standard types found in many programming languages: integers, floating-point numbers, booleans, strings, the *null* value, and *sets of strings*. Note that arrays, or sets of anything other than strings are considered opaque values. Primitive values are useful because Fluidinfo allows indexing of these values, permitting more complex and specific querying of tags with primitive values.

**Opaque tag-values** include any type of value that is not considered primitive. This includes JSON arrays or objects, binary data, anything that can be assigned a MIME-type. Opaque values are not indexed, and therefore users cannot search based on the contents of opaque tags, merely their presence.

### 4.2.1 About Tag

If objects are anonymous and an instance of a tag may contain any value independent of the other instances, one may wonder how to identify a specific object. Fluidinfo allows objects to be uniquely identified by a *UUID* (Universally Unique Identifier[6]) and the so-called *about-tag*. The about-tag, `fluiddb/about`, is a unique, immutable tag that may optionally be provided when creating an object. This allows for an object to be given some basic semantic value without adding any user tags to it, which can be useful in establishing tagging conventions.

## 4.3 Namespaces

Tags can be grouped together in *Namespaces*. All of a user's tags will live inside the user's top-level namespace to avoid conflicts with other users' tags, but sub-namespaces can be used to logically group tags. As an example, suppose the Fluidinfo user `eric` created a `rating` tag in his top-level namespace, the qualified name of that tag would be `eric/rating`. If we look back at Section 4.2.1, we can surmise that there is actually nothing special about the about-tag, it is simply a tag belonging to the `fluiddb` user, who is guaranteed to never change the value.

[6]http://en.wikipedia.org/wiki/UUID

## 4.4 Permissions

The core mechanic that allows Fluidinfo to be flexible is its *permissions* system. Each namespace and tag has an explicit set of permissions, describing exactly how users may interact with the item in question. This affords users fine-grained control over their data. They can publish it in read-only, read-write, or write-only form, or even transfer entire control of a namespace/tag to another user[7]. As an example of how these permissions can be used, let us examine how Fluidinfo creates new users. There is a tag, `fluiddb/users/username`, placed on the object representing a user, that tells Fluidinfo that such a user exists. The `fluidinfo.com` user has *create* permissions for this tag, so when a new user signs up on http://fluidinfo.com, the `fluidinfo.com` user creates a new object and adds the `fluiddb/users/username` tag to it, signifying that a new user has been created.

## 4.5 Fluidinfo Query Language

Fluidinfo includes a simple query language to allow users to search the datastore for specific tags and tag-values. There are five basic types of queries in Fluidinfo's query language.

**Presence** queries are the simplest type. They check only for the presence of a tag on an object, and are written as `has <tag>`.

**Numeric** queries search for tags that have a specific value using the standard mathematical equality operators, and are written as `<tag> (=,<,>,etc.) <value>`.

**Textual** queries attempt to match the query text against the text contents of a tag, and are written as `<tag> matches <text>`.

**Set contents** queries check for the tags that contain the given string. Note the difference between set contents and textual queries: set contents apply to tags containing a *set of strings* while textual queries apply to tags containing a single string. Set contents queries are written as `<tag> contains <string>`.

**Logical** queries combine the above types using the `(`, `)`, `and`, `or`, and `except` operators. This allows arbitrarily

[7]For a more detailed and complete list of the allowed permissions, visit http://doc.fluidinfo.com/fluidDB/permissions.html

complex queries, such as
`(has eric/seen and (eric/rating > 4 or john/rating > 8)) except imdb.com/rating < 5`.

# 5. WRITEABLE METADATA ENGINE FOR CACTUS COMPONENTS

In this section we will describe the desired capabilities for handling metadata for simulation codes, such as the ability to support open data objects and metadata which can then be provided by any user, promoting community driven standards and enabling innovation. Such a system would allow researchers to annotate the codes with their opinions, experiences, or tips while preserving the integrity of the original data. Social networks have already solved a subset of this problem, but there is no equivalent system in use by the scientific community.

Foursquare is a location-aware social networking site. Users publish their presence at a physical location, e.g. a metro station, a restaurant, a school, and can add photos or tips for others. If the physical location does not exist, users can add their own selecting basic metadata to describe the site. Social features include the ability to see where your friends are and have been, and to read the tips left by others.

Learning from this flexible model we envisage similar tools for data that will encourage academic data to break free from the current constraints of rigid schema, proprietary and controlled databases and lack of social networking tools. The general scenario we envisage is described below, here for software components, although a similar methodology will work for general data sets.

1. Software components (e.g. Cactus Thorns) are added to Fluidinfo in the same manner as foursquare locations. Basic tags could for example be based on the Dublin Core [5], with fields for authors, software location, etc. These tags can only be edited by the original user unless specified otherwise. An object would be created in Fluidinfo for each software component, we suggest an about-tag convention of `CCTK:<arrangement>/<thorn>`; however, this is strictly optional as the thorns would also be identified by their tags. There could also be multiple objects for each thorn since they could be added by people other than the original authors.
2. Trusted experts or consortia can then tag thorns to provide a quality ranking, associate datasets generated by the thorn, or warn new users of an existing bug. For example, a maintainer for the Einstein Toolkit would tag Cactus thorns with the release for which they have been tested and verified. Users can then search for software which has been ratified by the Einstein Toolkit Consortium, or they could search for software that has been recommended by a trusted colleague.
3. A graduate student is working on a research project to develop a new ontology for scientific computing. She can easily add tags representing this ontology to the Cactus thorns, where the user community can test out her work without necessitating new servers, or without her having write access to the basic thorn tags.

We implemented a prototype of such a system for the Cactus Thorns, with a web front-end written in Python [8]. The initial set of metadata we extracted from each thorn came from the configuration files and the Readme, representing a subset of the functional and bibliographical metadata contained in each thorn, as seen in Table 1. These tags are added automatically by a Python script that parses the configuration and Readme files of a thorn. The intent is for thorn authors to run this script on their thorns, immediately populating Fluidinfo with a set of Cactus metadata. Once the basic set of metadata has been imported, we can begin to enhance the existing data by adding other relevant tags to the objects representing thorns.

The Einstein Toolkit is a small subset of all Cactus thorns, and thorns may be imcompatible with each other, e.g. if they implement the same interface. Therefore it would be useful for users to know if any given thorn is part of the Einstein Toolkit; we can implement this quite naturally by creating an `einsteintoolkit.org` user[8], which will tag all thorns in the toolkit with an `einsteintoolkit.org/includes` tag with the value set to `True`. Figure 4 illustrates what the resulting Fluidinfo object might look like.

Using this tag structure we created a simple web application, running on Google's AppEngine platform, to dynamically retrieve the objects representing the Einstein Toolkit, and insert the values into an HTML template for easy viewing of the thorn metadata. Figure 5 shows a sample page from this web application.

With these two sources of data we can already perform useful queries on the Cactus metadata. Cactus uses a tool called *GetComponents* [11] to automate the process of retrieving many thorns from different locations. To accomplish this, GetComponents essentially needs three pieces of information:

1. Where the thorn is located (URL).
2. How to retrieve the thorn (version control system).
3. Where to place the thorn on the local filesystem.

All of this data is contained in the Fluidinfo tags posted by the Python script[9]! So if we wanted to retrieve the Einstein Toolkit, we could dynamically generate a file in the CRL format GetComponents uses by querying Fluidinfo for all objects that have `einsteintoolkit.org/includes = True`, retrieving the tags

- `gridaphobe/CCTK/arrangement`
- `gridaphobe/CCTK/name`
- `gridaphobe/CCTK/url`

[8]Fluidinfo only allows the owner of a domain to create the user for that domain, so domain users can be more readily trusted.
[9]Cactus has a convention of placing thorns inside an `arrangements` directory with the structure `arrangements/<arrangement>/<thorn>`.

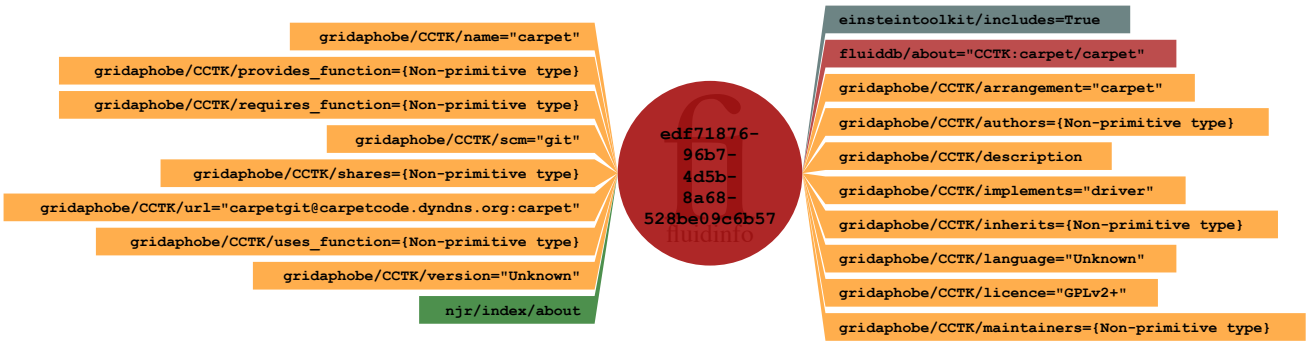


**Figure 4: Visual representation of the Fluidinfo object for the Carpet module in the Einstein Toolkit.**

| Fully-qualified Tag | Description |
|---|---|
| `gridaphobe/CCTK/arrangement` | The *arrangement* the thorn belongs to. |
| `gridaphobe/CCTK/authors` | A list of all authors of the thorn. |
| `gridaphobe/CCTK/description` | The description of the thorn as found in the Readme. |
| `gridaphobe/CCTK/implements` | A list of *interfaces* the thorn implements. |
| `gridaphobe/CCTK/inherits` | The thorn (if any) inherited from. |
| `gridaphobe/CCTK/name` | The name of the thorn. |
| `gridaphobe/CCTK/scm` | The version control system used for the thorn's source code. |
| `gridaphobe/CCTK/url` | The URL where the thorn's source code is located. |

**Table 1: A sample of the tags used to describe Cactus thorns in Fluidinfo. The tag names are fully-qualified and assume the current user's name is `gridaphobe`.**

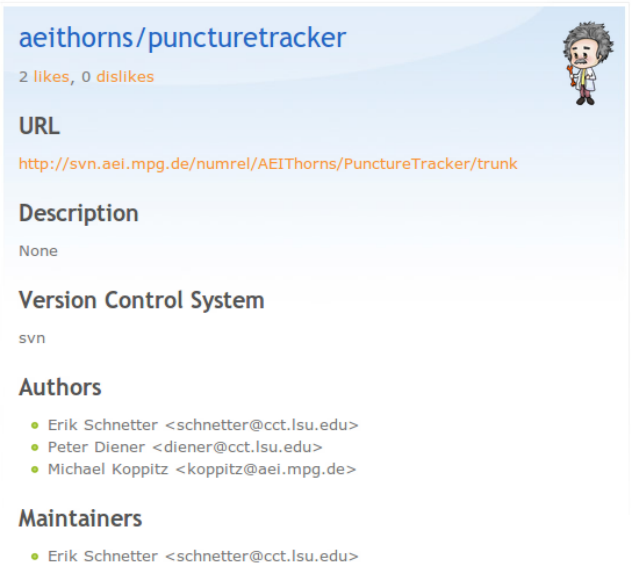


**Figure 5: A prototype of a web application that dynamically displays thorn metadata based on the tags stored in Fluidinfo. The Einstein logo in the top-right corner indicates that this thorn is part of the Einstein Toolkit.**

- `gridaphobe/CCTK/scm`

The returned data could then be reformatted into a CRL file, and GetComponents invoked to automatically retrieve the requested thorns[10].

This is already a significant improvement over the current system of creating and distributing a thornlist, which is both tedious and error-prone, but we can go further and solve a problem that was previously unsolvable. The Einstein Toolkit thorns can all be compiled together; however, they are not all needed to run individual simulations. Researchers will generally only compile a subset of the Einstein Toolkit, including just the thorns needed to model their particular system. In this case downloading the entire Einstein Toolkit is superfluous, we would like to simply download the thorns that we *actually need*. Using the thorn configuration files, we can construct a list of the thorns we will need to download in order to use a specific *base set* of thorns, providing initial data, drivers, and other components of a simulation [1]. We can then dynamically retrieve the tags mentioned above for only this subset of thorns, and provide GetComponents with a much smaller list of thorns to download. This also has the benefit of isolating the code in the source tree of any simulation to only that which is necessary.

If we wanted to implement a system like this on our own, we would have to setup a new webserver and database, define a schema to contain the data, create a REST API, and

[10]There are some issues not covered by this example, e.g. the directory structure of different git repositories, but none that could not be resolved by adding a few extra tags

then assign someone to maintain the database and server. If we additionally wanted the system to be writeable (or at least have individual thorns managed by their authors), we would then have to implement an authentication system as well, and our data would still be limited to some pre-defined schema. Fluidinfo allows others to add to our data, and we can *choose* whether to ignore it or to begin incorporating pieces into our applications.

# 6. FUTURE WORK

In the previous section we saw how to use Fluidinfo to store the metadata of Cactus thorns in a *writeable* format, add tags to those thorns from a different source, and then use tags from both sources to solve a problem that previously could not be solved without setting up our own web server. We did, however, ignore one issue; the example only dealt with thorns uploaded by one user, whereas the Einstein Toolkit is comprised of thorns written by many different authors. Suppose we don't know who all of the authors are, how will we know which tags to retrieve? For example, the Carpet thorns are written by Dr. Erik Schnetter, but unless we know his Fluidinfo username, we won't know how to retrieve his tags. Fluidinfo does not currently support wildcards in the list of tags to return, so we *must* explicitly list the tags we want. So how can we best adapt our solution to the actual problem? There are two possible solutions:

1. Instead of using tags in the author's namespace, we could take advantage of Fluidinfo's permissions system to give all authors write permission to tags in a `cactuscode.org/CCTK` namespace. This way we would always retrieve tags from the trusted domain user. This solution detracts from the personalization of Fluidinfo though, since the tags are coming from a domain user instead of the author himself. In a sense this represents how we might solve the metadata problem on our own, but with the extra downside that we can no longer prevent authors from modifying each other's tags! Fluidinfo does not allow separate permissions per tag-instance, and this would become far too complex to manage regardless.

2. Create a `cactuscode.org/author` tag that would be applied to the objects representing the *users* in Fluidinfo who are authors of Cactus thorns. This way we can query Fluidinfo for the objects with the tag, and ask it to return the `fluiddb/users/username` tag, giving us a list of all Fluidinfo users who are also Cactus authors. Then we can proceed with the process described in Section 5. This solution has several advantages: (1) authors cannot modify each other's tags without explicit permission, (2) in the event of a *tag collision* (where more than one author has tagged a thorn) we can apply some filtering condition based on the thorn's own author list to determine which tags are most authoritative, (3) we are actually adding more data to the ecosystem by *tagging* the users as Cactus authors.

## 6.1 Other Datasets

Supercomputers are generating massive amounts of data on a daily basis, data which must be stored efficiently and then classified so that it can be referred to and even cited. Our strategy in Section 5 can easily be adapted to solve this problem. Suppose we run a simulation of two colliding neutron stars and store the resulting dataset. We can now create an object in Fluidinfo to represent this simulation, and tag it with the machine used, number of cores, initial values, duration, and any number of other relevant statistics about both the simulation and the output. Then a PhD student uses our dataset in her thesis; she can tag the dataset in Fluidinfo with a `<student>/cited` tag whose value would be a list of all papers in which she cited our dataset (likely using DOIs). If she is consistent in tagging the datasets she has cited, we could perform interesting queries using Fluidinfo, i.e. we could quickly determine which supercomputers had contributed most to her work. Other researchers might tag the datasets with specific situations where they proved useful, or perhaps related datasets. With a writeable, schemaless system, the datasets may be augmented in any fashion deemed suitable by users. This allows for use-cases the original publisher could not have conceived of to arise organically.

It is becoming clear that citing datasets produced by simulations will be essential for continued scientific progress, one need look no further than the NSF's Computational and Data-Enabled Science and Engineering[11] program. Ball and Duke have raised some important questions that will have to be answered for data citation to become widespread [2]. We would like to address the question of how the metadata can be stored in a manner accessible both to humans and automated scripts. By storing the metadata in a shared datastore like Fluidinfo, it is immediately available for consumption by scripts, and by extension easily converted into a human-readable page as we have demonstrated in this paper. We also gain the advantage of not being tied to any schema, allowing us to freely add more metadata whenever necessary. Finally, the writable nature of Fluidinfo removes the author's responsibility of linking to all papers that have cited the dataset. The author of a paper can simply tag the dataset in Fluidinfo!

# 7. CONCLUSION

Scientific research is increasingly dependent on the simulation of complex processes and, by extension, on the ability to organize, search, and refer to the datasets generated by simulations. We propose using writable metadata to distribute and maintain scientific metadata, and have shown one possible method of implementing such a system. More work will be required to investigate alternative systems, schemas, and interfaces, as well as to determine what would be an optimal solution. We hope that the scientific community will take this opportunity to start a conversation about how to manage the large amounts of data currently being generated by our research on a daily basis.

# 8. EDUCATIONAL EXPERIENCE

The research presented in this paper was performed as part of a year-long internship sponsored by the Shodor Educational Foundation[12]. The program began with a two-week

[11] http://www.nsf.gov/mps/cds-e/
[12] http://www.shodor.org

intensive introduction to HPC, covering parallelization issues, N-Body problems, MPI, and other computational science topics. Following the introductory session, the interns split up to work with individual mentors for the rest of the year. While not strictly related to Computational Science, the research presented in this paper was strongly supported and enhanced by the Blue Waters Petascale Internship, especially the focus on solving *real* problems.

## Acknowledgments

This work was supported by the Blue Waters Undergraduate Petascale Education Program, as well as Fluidinfo, Inc. The initial work relating to the Cactus Configuration Language was supported by NSF REU program (#1005165). We would like to thank Steven Brandt, Frank Löffler, and Erik Schnetter for their mentorship in the Cactus group, and Gabrielle Allen for suggesting the use of Fluidinfo for storing the thorn metadata. We acknowledge Nicholas J. Radcliffe, who created http://abouttag.com to generate visuals of Fluidinfo objects.

## 9. REFERENCES

[1] Gabrielle Allen, Tom Goodale, Frank Löffler, David Rideout, Erik Schnetter, and Eric L. Seidel. Component specification in the cactus framework: The cactus configuration language. In *CBHPC '10*, New York, NY, USA, 2010. ACM.

[2] A. Ball and M. Duke. Data citation and linking. In *DCC Briefing Papers*. Digital Curation Centre, 2011.

[3] Min Cai, Martin Frank, Baoshi Yan, and Robert MacGregor. A subscribable peer-to-peer rdf repository for distributed metadata management. *Web Semantics: Science, Services and Agents on the World Wide Web*, 2(2):109 – 130, 2004.

[4] Mesh Refinement with Carpet.

[5] Dublin Core Metadata Initiative.

[6] Fluidinfo Documentation.

[7] T. Goodale, G. Allen, G. Lanfermann, J. Massó, T. Radke, E. Seidel, and J. Shalf. The Cactus framework and toolkit: Design and applications. In *Vector and Parallel Processing – VECPAR'2002, 5th International Conference, Lecture Notes in Computer Science*, Berlin, 2003. Springer.

[8] Python Programming Language.

[9] Erik Schnetter, Peter Diener, Nils Dorband, and Manuel Tiglio. A multi-block infrastructure for three-dimensional time-dependent numerical relativity. *Class. Quantum Grav.*, 23:S553–S578, 2006.

[10] Erik Schnetter, Scott H. Hawley, and Ian Hawke. Evolutions in 3D numerical relativity using fixed mesh refinement. *Class. Quantum Grav.*, 21(6):1465–1488, 21 March 2004.

[11] Eric L. Seidel, Gabrielle Allen, Steven Brandt, Frank Löffler, and Erik Schnetter. Simplifying complex software assembly: the component retrieval language and implementation. In *TG '10: Proceedings of the 2010 TeraGrid Conference*, pages 1–8, New York, NY, USA, 2010. ACM.

[12] SimFactory: Herding Numerical Simulations.

[13] Hironobu Takagi, Shinya Kawanaka, Masatomo Kobayashi, Takashi Itoh, and Chieko Asakawa. Social accessibility: achieving accessibility through collaborative metadata authoring. In *Proceedings of the 10th international ACM SIGACCESS conference on Computers and accessibility*, Assets '08, pages 193–200, New York, NY, USA, 2008. ACM.

[14] Michael Thomas and Erik Schnetter. Simulation factory: Taming application configuration and workflow on high-end resources. In *CBHPC '10*, New York, NY, USA, 2010. ACM.

[15] Cactus Computational Toolkit.

[16] The Einstein Toolkit.

# Bringing *ab initio* Electronic Structure Calculations to the Nano Scale through High Performance Computing

James Currie
University of Missouri – Kansas City
Department of Physics and Astronomy
5110 Rockhill Road
Kansas City, MO, 64110
1-816-235-2501
jecyrd@mail.umkc.edu

Rachel Cramm Horn
University of Missouri – Kansas City
Department of Physics and Astronomy
5110 Rockhill Road
Kansas City, MO, 64110
1-816-235-2501
rec44f@mail.umkc.edu

Paul Rulis
University of Missouri – Kansas City
Department of Physics and Astronomy
5110 Rockhill Road
Kansas City, MO, 64110
1-816-235-5945
rulisp@umkc.edu

## ABSTRACT
An *ab initio* density functional theory based method that has a long history of dealing with large complex systems is the Orthogonalized Linear Combination of Atomic Orbitals (OLCAO) method, but it does not operate in parallel and, while the program is empirically observed to be fast, many components of its source code have not been analyzed for efficiency. This paper describes the beginnings of a concerted effort to modernize, parallelize, and functionally extend the OLCAO program so that it can be better applied to the complex and challenging problems of materials design. Specifically, profiling data were collected and analyzed using the popular performance monitoring tools TAU and PAPI as well as standard UNIX time commands. Each of the major components of the program was studied so that parallel algorithms that either modified or replaced the serial algorithm could be suggested. The program was run for a collection of different input parameters to observe trends in compute time. Additionally, the algorithm for computing interatomic interaction integrals was restructured and its performance was measured. The results indicate that a fair degree of speed-up of even the serial version of the program could be achieved rather easily, but that implementation of a parallel version of the program will require more substantial consideration.

## Categories and Subject Descriptors
D.2.8 [**Software Engineering**]: Metrics – *performance measures.*

## General Terms
Algorithms, Measurement, Performance.

## Keywords
Density functional theory, atomic orbitals



## 1. INTRODUCTION
Advanced materials have played a pivotal role in recent technological progress, often causing the demand for designer characteristics or novel properties to outpace our ability to understand these complex materials at a fundamental level. This pressure to master materials at the nanoscale has pushed forward the development of many theoretical approaches and the implementation of many computational methods. A particular area of interest includes structures with defects that are on the order of 10 nanometers in size because many bulk structural and electronic properties of materials are dominated by the properties of the defect. Density functional theory (DFT) based approaches represent the current state of the art for the application of theory to materials problems that require both high accuracy and high efficiency. This is a position somewhat between the larger scale molecular dynamics methods and the smaller scale but often more accurate quantum chemical methods. Some of the issues that DFT is well suited to deal with include catalysis processes [15], configuration of ultra-dilute dopants in crystal structures [27], and the determination of the tensile strength of bioceramics [5]. DFT is a computational, quantum mechanical framework for the modeling of materials and it is being actively applied with much success across a wide breadth of fields within a growing number of scientific domains. Density functional theory was created in a sequence of two papers by Hohenberg and Kohn [14] and Kohn and Sham [17]. They presented the method as one that reduces the problem of determining the many-body ground state wave function to one of determining only the charge density. For a system of N interacting electrons this reduces the problem from a space of 3N dimensions down to a space of just three. A type of mean-field approach is used that solves a one-electron problem where the potential is derived from the charge density distribution of all the electrons in the system. A self-consistent field (SCF) cycle iterates through determination of the wave-function, the charge density distribution, and the potential until there is no change in these terms. Since its original inception, much work has been done to enhance this method, and a review of the theory can be found in a paper written by Peter Blöchl [1]. Presently, there exist a variety of different implementations of DFT that can be divided into a few prominent camps based on the choice of basis functions used to expand the system wave function, the representation of the potential function, and the representation of the charge density [20]. Each of the main approaches to DFT has its own set of advantages and disadvantages that make it particularly applicable to one range of materials and problems or another.

Although DFT provides an incredible simplification of the quantum mechanical many-body problem while still retaining excellent accuracy, the types of problems that are of the most interest still require prohibitive amounts of time for even the fastest computer processor to solve. Hence, parallel processing has become an invaluable tool and many DFT program codes have been adapted to take advantage of this high performance computing (HPC) capability. Interestingly though, the programming style required to develop parallel algorithms is significantly different from the approach of serial algorithms which sometimes makes it difficult to parallelize an existing code and gain as much efficiency as is desired. Therefore, when parallelizing an existing serial application, performance analysis of the existing algorithm lends some helpful insight about which sections of a program are the most computationally expensive and why. This could be used to determine whether the algorithm should be simply modified for a parallel execution environment or if it needs to be totally rewritten.

This paper describes the beginnings of a concerted effort to modernize, parallelize, and functionally extend a particular DFT based program so that it can be applied to the complex and challenging problems of materials design. Specifically, profiling data was collected and analyzed using widely available and portable external libraries and standard UNIX/Fortran time commands. Then, each of the major components of the program was studied so that parallel algorithms could be suggested that either modified or replaced the serial algorithm.

## 2. METHODS

The primary focus of this development work is the Orthogonalized Linear Combination of Atomic Orbitals (OLCAO) method [29]. This is a density functional theory based method that uses Gaussian based atomic orbitals in the solid state wave function expansion and atom centered Gaussian functions for an analytical description of the potential and charge density distribution functions. OLCAO has found particular application in the study of the electronic structure, bonding, and spectroscopic properties of large and complex materials systems ranging from amorphous solids [9–11, 16] and complex crystals [3, 18, 19, 30] to those containing large scale structures such as grain boundaries (GBs) [6, 21, 23], intergranular glassy films (IGFs) [7, 8, 25], and passive defects [4, 24]. Of particular interest to the development work that was started as a part of the Blue Waters – Undergraduate Petascale Education Program (BW-UPEP) Internship is the capability of the OLCAO method with regard to performing core level spectroscopic calculations such as x-ray absorption near edge structure or electron energy loss near edge structure (XANES/ELNES) [12, 13, 24, 27]. An extension of the normal spectral calculation is being developed within OLCAO whereby spectra are computed for every atom in a model and then brought together to form an image that correlates spectral features with atomic structure [24]. This spectral imaging technique will be quite computationally intensive and while the OLCAO program is efficient and capable of being used to compute the XANES/ELNES spectra of rather large systems it is still a serial application and thus the calculation times can be quite lengthy, sometimes lasting more than a few days. There are two key mathematical operations that are performed in OLCAO. The first is the analytic calculation of a set of integrals between atomic orbitals in various forms given for s-type orbital in Equations 1-4.

$$\langle s_A | s_B \rangle = \int e^{-\alpha_1 \vec{r}_A^{\,2}} e^{-\alpha_2 \vec{r}_B^{\,2}} d\vec{r} \quad (1)$$

$$\langle s_A | -\nabla^2 | s_B \rangle = \int e^{-\alpha_1 \vec{r}_A^{\,2}} \left(-\nabla^2\right) e^{-\alpha_2 \vec{r}_B^{\,2}} d\vec{r} \quad (2)$$

$$\left\langle s_A \left| \frac{1}{\vec{r}_C} e^{-\alpha \vec{r}_C^{\,2}} \right| s_B \right\rangle = \int e^{-\alpha_1 \vec{r}_A^{\,2}} \frac{1}{\vec{r}_C} e^{-\alpha_3 \vec{r}_C^{\,2}} e^{-\alpha_2 \vec{r}_B^{\,2}} d\vec{r} \quad (3)$$

$$\left\langle s_A \left| e^{-\alpha \vec{r}_C^{\,2}} \right| s_B \right\rangle = \int e^{-\alpha_1 \vec{r}_A^{\,2}} e^{-\alpha_3 \vec{r}_C^{\,2}} e^{-\alpha_2 \vec{r}_B^{\,2}} d\vec{r} \quad (4)$$

Equation 1 represents the overlap of two s-type Gaussian orbitals. Equation 2 also shows integration of s-type Gaussians but with the Laplacian operator for the computation of the kinetic energy. Equation 3 accounts for the contribution of the nuclear interaction to the total potential. Equation 4 is a three center integral between s-type Gaussians that is used for determining the Coulombic electron – electron contribution to the total potential. Higher angular momentum integrals can be derived from these equations via repeated differentiation making progressively more complicated formulas [29]. The second major operation is the processes of solving the eigenvalue problem that obtains the wave function expansion coefficients and associated energy eigenvalues. This requires the complete diagonalization of a large matrix. Some key parameters that affect the cost of these operations are the number of atoms in the system, the number of basis functions used for each atom (typically identified as either a minimal, full, or extended basis), the number of k-points (Brillouin zone integration sampling points), and the number of terms used to describe the potential function. Parallelization of these operations could significantly decrease the calculation time or, if the calculation time is maintained, it would allow for the study of much larger systems.

In addition to the time stamps given by OLCAO, we used tools such as the Tuning and Analysis Utilities (TAU) [26] and the Performance Application Protocol Interface (PAPI) [2]. TAU and PAPI exist as a set of library function calls that are instrumented into the source code of a program to track a wide variety of information ranging from simple subroutine runtimes to the number of times that specific CPU events, such as a cache miss or an execution branch, occur. The TAU package is described as a "portable profiling and tracing toolkit for performance analysis of parallel programs."[28] However, it does have particular uses in the analysis of serial programs as well. PAPI, similar to TAU, is a portable kit that accesses hardware performance counters physically present on modern microprocessors. The aim of PAPI is "to see, in near real time, the relation between software performance and processor events." [22] Memory access overhead and branching performance were of particular interest in this study to understand where bottlenecks in OLCAO may exist. Memory access overhead is a fundamental problem, especially for HPC, because the processors often operate at data rates that are much faster than the data transfer rates to memory so that if the CPU is not well supplied with data it will sit idle. Understanding the cache performance of a program can help the programmer reorganize data structures to provide more efficient memory access. Conditional branch prediction is another important area because modern CPUs maintain a pipeline of operations and when a branch is mispredicted the pipeline must be flushed and refilled at substantial cost to efficiency. These were the primary tools that were used to explore the characteristics and efficiency of the algorithms in the OLCAO program suite.

The first step to parallelizing OLCAO is to develop a base line understanding of its serial execution performance in terms of the

compute time for different sections of the code under different conditions. To do this, a series of different materials systems were selected and parameters that control the computational cost of the calculation were varied systematically. Two machines were used to collect the performance data, a local workstation using the commercially available AMD Phenom II X6 1090T processor and the Pittsburg Supercomputing Center's Blacklight machine which uses the Intel Xeon X7560 processor. Smaller system calculations were performed on the local workstation while the larger system calculations made use of Blacklight. The performance trends were observed using a variety of tools. Specifically, timing data for different sections of the OLCAO program were collected first by using Fortran write statements that recorded the current time. This gave an initial understanding of the effects of parameter changes and did not require any modifications to the program code because such write statements already existed. Then, particular sections of the code were identified and studied with TAU and PAPI because they are more fine grained tools for understanding the reason why a particular algorithm behaves as it does. The data that was collected using TAU and PAPI included the rate of branching misprediction for the combination of different compiler flags and changes in the algorithm for a particular section of code. The code in the select section was restructured with the intent of reducing the number of conditional statements encountered overall.

## 3. RESULTS AND DISCUSSION

The profiling techniques described in section 2 were applied to the OLCAO program for three different material systems with a systematic variation of parameters. The systems used in the study were a 907 atom model of an IGF within crystalline β-$Si_3N_4$, a ten base pair periodic model of DNA with 650 atoms (computed on Blacklight), and a series of supercells of pure Al (computed with a local workstation). Details of the key structural parameters of each system are provided in **Table 1** and an illustration of each system is provided in **Figure 1**.

**Table 1: Crystal Structure Details**

| | IGF | DNA | Al Full Cell |
|---|---|---|---|
| a (Å) | 14.533 | 30.000 | 4.050 |
| b (Å) | 15.225 | 30.000 | 4.050 |
| c (Å) | 47.420 | 39.208 | 4.050 |
| α (°) | 90 | 90 | 90 |
| β (°) | 90 | 90 | 90 |
| γ (°) | 90 | 90 | 90 |
| # of Atoms | 907 | 650 | 4 |
| # of Electrons | 4288 | 2220 | 12 |
| Matrix Dimension | 9111 (Full Basis) | 4740 (Full Basis) | 52 (Full Basis) |
| Elements | Si, N, O | C, H, N, Na, O, P | Al |

The IGF and DNA material systems were specifically chosen as representatives of particular classes of materials that are of current research interest and therefore represent the types of systems likely to be encountered by the OLCAO program in practice. Also, these systems possess specific features that can make their comparison helpful. The IGF and DNA models both have a relatively large number of atoms but the dimension of the interaction matrices for the IGF and the number of electrons is larger in it compared to the DNA model by a factor of about two. The dimension is larger because the Si in the IGF includes unfilled 3d atomic orbitals in its basis while the DNA model has no atoms with 3d orbitals in its basis. The number of electrons is larger because the IGF contains no H and thus every atom has more electrons. For the IGF and DNA models it is also possible to easily and realistically alter the number of independent terms in the potential function representation. This is done by changing the threshold criteria for how similar two atoms need to be with respect to their local environment before they can share the same potential function values. This "sharing of values" between potential sites means that fewer terms are used in the potential function and thus that fewer independent interaction matrices need to be created. This issue is a particular characteristic of the representation of the potential function in OLCAO and is not generally applicable to all DFT based methods.

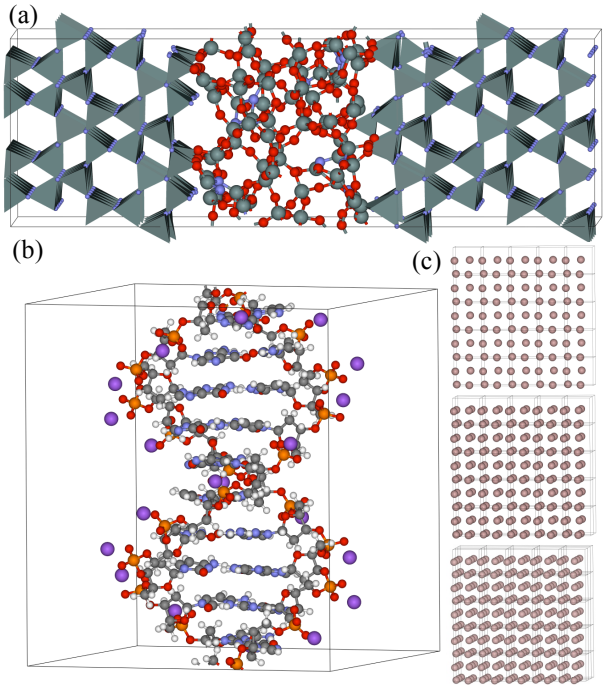


**Figure 1: Ball and stick models of the material systems studied. (a) An intergranular glassy film model in β-$Si_3N_4$; (b) A ten base pair periodic model of DNA with Na counter ions; (c) A sequence of three crystalline Al supercells 5×5×1, 5×5×2, and 5×5×3.**

The first level of analysis was done with a simple measurement of the amount of time that different segments of the OLCAO program took to run calculations on the IGF model and the DNA model when the number of terms in the potential function was modified. In particular, two key sections, identified as "Setup" and "Main," were analyzed to find the most time expensive parts of their code. The "Setup" (integrals, electrostatics, exchange-correlation) and "Main" (Secular Equation [preparation, solution], and everything else) programs comprise the self-consistent field (SCF) implementation of OLCAO. These programs and their components are illustrated schematically in **Figure 2**.

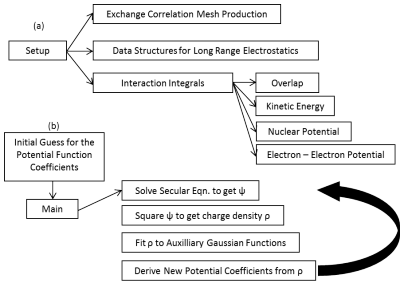


**Figure 2: Schematic of the components of the SCF portion of the OLCAO program. (a) Setup; (b) Main.**

The runtime analysis of these components is show in **Figure 3**. In the IGF system, the ratio of the calculation time for the integrals compared to the rest of the setup calculation is quite large (**Figure 3a**). We also see that the long-range coulomb and exchange correlation calculations are almost unaffected by the number of potential terms, with the majority of the time being spent with the interaction integrals. In the sections of main we see that as the number of terms increases the preparation time of the secular equation quickly grows to be much larger than the time it takes to solve the secular equation. In this case the preparation is simply the task of reading the packed matrix data stored on disk by setup, unpacking it, and applying a coefficient to each matrix before accumulating it.

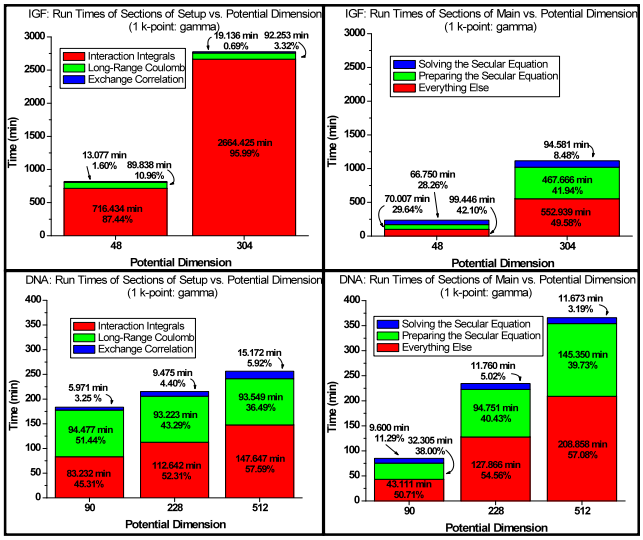


**Figure 3: Illustration of the execution time for the IGF and DNA models for various sections of the OLCAO Setup and Main programs and various number of potential terms as measured by Fortran time records. (a) Setup and IGF; (b) Main and IGF; (c) Setup and DNA; (d) Main and DNA.**

When the same type of analysis is done for the DNA system, as shown in **Figures 3c and 3d**, we see a similar trend to that of the IGF system, where the interaction integrals grow more costly as the number of potential terms increases. Also, the long-range Coulomb calculations have more of a presence here due to the larger cell size and thus a larger number of reciprocal space cells needed for the convergence of the Ewald summation series. Considering the run length of the program "Main", the trend is similar to the case of the IGF. The similarity of the trend is expected, but a comparison of the two systems presents some confusion. The total time of even the longest run of Setup for the DNA model (about 260 minutes with 512 terms in the potential function) is less than the shortest IGF model run with only 48 terms in the potential function (about 815 minutes). This is likely due in large part to the larger size of the interaction matrix for the IGF model (i.e. it has many more basis functions), but there may also be something more subtle at work. That is, the number of nearest neighbors for a given atom in the DNA model is, on average, less than that of the IGF model because DNA is a molecule with exposed surface boundaries and lots of H. This will substantially reduce the number of interaction integrals that will need to be calculated resulting in an interaction matrix that is not only smaller, but also sparser.

The pure Al model was chosen because of its simplicity and for the fact that we could vary specific parameters with less worry about how the innate characteristics of the model might mask the effect of a change in other parameters such as the number of k-points or the choice of a full, minimal, or extended basis. The results of the timing runs for a constant number of terms in the potential function are shown in **Figure 4**.

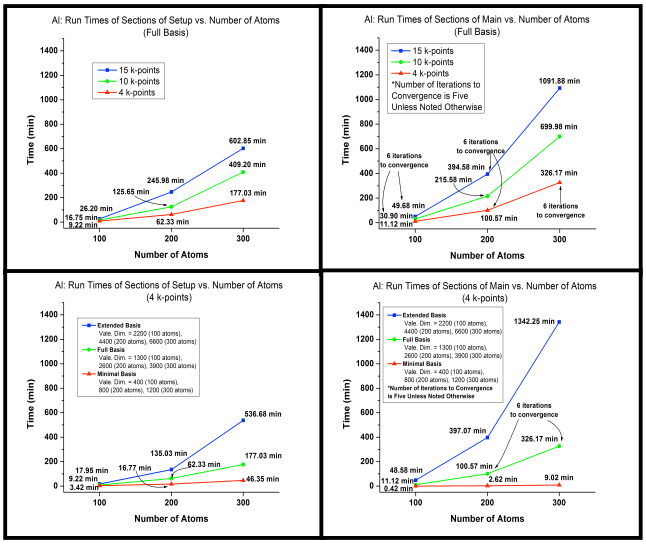


**Figure 4: Illustration of the execution time for calculations of various sections of the OLCAO Setup and Main programs for three Al supercell models with variation in the number of k-points and the size of the basis set. (a) Setup and k-points; (b) Main and k-points; (c) Setup and basis size; (d) Main and basis size.**

In the study of the Al runtime performance we see that OLCAO is more sensitive to changes in the basis than to increases in the number of k-points. When k-points are added the effect is multiplicative so that doubling the number of k-points effectively doubles the cost of the calculation while when the basis size is increased a non-linear trend is observed. For the 300 atom case, an approximate doubling of the basis size from full to extended more than doubles the cost of the calculation for both Setup and Main. This is expected because the basis size affects the interaction integrals matrix size which scales as the square of the dimension. This is clear in the "Main" section because it shows an increase in calculation time for the extended basis that is approximately four times greater than for the full basis. In setup this change was much less exaggerated with only about a three

times increase. One important note is that as the system size increases the number of k-points needed for a high resolution calculation decreases. Because OLCAO is typically used for large and complex systems, the number of k-points is often just equal to one.

Beyond the crude timing data, a series of calculations were performed to study the performance of some of the most costly components of the program using TAU and PAPI. The multicenter interaction integrals take a significant portion of the overall time which is typical for all atomic orbital based methods. Another component that is more specific to the OLCAO program is the orthogonalization procedure. This modifies the resultant interaction matrix to force the valence orbitals to be orthogonal to the core orbitals. For large systems this could be a costly step because a sequence of matrix-matrix multiplications is required. The program code for computing those integrals and for doing the orthogonalization has performed well for decades, but it has also not been evaluated for efficiency in just as long. Hence a concerted effort is underway to evaluate and possibly improve this aspect of the program.

For the analysis of the OLCAO program using TAU and PAPI a different set of Al supercells was used. They were 1×1×1, 2×2×2, and 3×3×3 supercells of the full cell so that the models had 4, 32, and 108 atoms respectively. The goal of the orthogonalization subroutine analysis was simply to count the number of branches encountered by the program for a given compiler optimization level. The collected data is shown in **Table 2**. For level two compiler optimization (obtained using the –O2 compiler flag) the branch misprediction rate was slightly better than that obtained with level three optimization. However the total number of branches encountered by the program under level two optimization was significantly greater than that encountered under level three optimization. This result may have been expected, but an unusual second result was that the misprediction rate for the 2×2×2 supercell for both levels of optimization was significantly higher than for both of the other two supercells. This exercise indicates that interpreting the results of higher level code analysis can have important subtleties that can be easily overlooked.

**Table 2: Al Supercell Orthogonalization Subroutine Branching Data**

| Cell and Optimization | Miss-Predicted (%) | Correctly Predicted (%) | Total |
|---|---|---|---|
| 1x1x1 -O2 | 3.92 | 96.06 | 7748064 |
| 1x1x1 -O3 | 3.97 | 96.03 | 7749479 |
| 2x2x2 -O2 | 6.52 | 93.48 | 62052410 |
| 2x2x2 -O3 | 6.81 | 93.19 | 59604903 |
| 3x3x3 -O2 | 3.01 | 96.99 | 1222164262 |
| 3x3x3 -O3 | 3.86 | 96.14 | 957394561 |

The analysis of the integration subroutine was also performed as a comparison between level two and level three compiler optimization, and it was also performed for the case of a modification in the algorithm versus the unmodified algorithm. The essential modification is that depending on the particular atoms it may be necessary to perform only s-type with s-type integration, or perhaps s-type with p-type or only up to p-type with p-type. In other cases the integral may need to include all the way up to the complicated d-type with d-type integral. The subroutine that performs the integrals checks along the way to determine which integrals to do, but it was observed that this sequence of "if-else" blocks in the code was repetitive such that if the section was rewritten one block could replace a sequence of three or four. This replacement option was present multiple times in the algorithm. The data from the sequence of TAU and PAPI runs are shown in **Tables 3 through 6**. The different types of integrals correspond to those given in Equations 1 through 4 plus all of the similar integrals of the higher angular momentum orbitals. Again, the trend is clear. The higher level of optimization had a higher percentage of branch miss-predictions, but the total number of branches was significantly less. When the comparison is between the old and the new algorithm the total number of branches drops in all cases, but again the branch miss-prediction percentage increases.

**Table 3: Original Integration Subroutine With –O2 Optimization.**

| Integral Type | Miss-Predicted (%) | Correctly Predicted (%) | Total |
|---|---|---|---|
| 1 | 1.00 | 99.00 | 53795827 |
| 2 | 1.09 | 98.91 | 67004191 |
| 3 | 1.44 | 98.58 | 208187973 |
| 4 (Avg.) | 1.35 | 98.65 | 310677930 |

**Table 4: Original Integration Subroutine With –O3 Optimization.**

| Integral Type | Miss-Predicted (%) | Correctly Predicted (%) | Total |
|---|---|---|---|
| 1 | 1.95 | 98.08 | 9767131 |
| 2 | 2.58 | 97.42 | 9766631 |
| 3 | 2.37 | 97.63 | 63389710 |
| 4 (Avg.) | 2.04 | 97.96 | 66161685 |

**Table 5: Modified Integration Subroutine With –O2 Optimization.**

| Integral Type | Miss-Predicted (%) | Correctly Predicted (%) | Total |
|---|---|---|---|
| 1 | 0.92 | 99.08 | 53415195 |
| 2 | 0.99 | 99.01 | 66378834 |
| 3 | 1.16 | 98.84 | 207903892 |
| 4 (Avg.) | 1.24 | 98.76 | 310464531 |

**Table 6: Modified Integration Subroutine With –O3 Optimization.**

| Integral Type | Miss-Predicted (%) | Correctly Predicted (%) | Total |
|---|---|---|---|
| 1 | 1.96 | 98.04 | 8988970 |
| 2 | 2.75 | 97.25 | 9140602 |
| 3 | 2.92 | 97.08 | 60653657 |
| 4 (Avg.) | 2.40 | 97.60 | 57420935 |

# 4. CONCLUSION

The sections of the OLCAO program associated with the SCF calculation were analyzed for their performance characteristics using both simple timers and more complicated instrumented performance monitoring tools. The results confirm some previously held beliefs about the relative computational cost of various components of the setup and main programs and also introduced some new and as yet unexplained results. Clear results include the fact that the computation of the interaction integrals is generally the most expensive but that preparing the Ewald summation data structures can become time consuming when the cell size becomes large. The overall cost of the interaction integrals calculation scales primarily with the number of terms in the potential function. The calculation cost scaled linearly with an increase in the number of k-points while it increased super-linearly (quadratically) with an increase in the size of the basis. Generally, the timing data was observed to be crude but it was also good at giving a big picture of the behavior of the program and so a key conclusion is that sometimes it is not necessary to apply highly sophisticated methods to get a good initial understanding. However, it is often difficult to totally isolate one particular variable, especially when attempting to model realistic calculation performance. This may be because multiple effects are correlated or because the model under study has characteristics that are not apparent simply in terms of the number of atoms, potential terms, matrix dimension, etc. This was exemplified by the smaller average atom density in the DNA model compared to the IGF model. The conclusion is that extreme care must be taken when interpreting any profiling results as it may be possible for subtle effects to skew the data.

Additional profiling data was obtained for the specific subroutines that involve the interaction integrals computed in OLCAO. The results indicated that even a simple attempt at code restructuring was able to produce noticeable results. We can conclude from this that while the algorithm may be fast because of its analytic nature, it is quite likely that significant improvements can be obtained by implementing a better approach to the calculation. The reasoning is that if even a straightforward modification can produce such dramatic results, then it is likely that a deeper degree of consideration will produce more substantial results. This also demonstrates that even though the profiling mission was to set the stage for later parallelization, it is entirely possible that the analysis will spark a deeper understanding of a subroutine or algorithm that will remain entirely serial but which may have a significant effect on the overall efficiency.

Another important conclusion that can be drawn from this study is that good profiling must come from the intelligent application of multiple tools, both crude and advanced. The fine grained advanced tools can often help provide the insight needed to understand why a particular algorithm behaves the way it does, but crude measurements can provide the appropriate context in which to interpret that data. It is not possible to naively apply a powerful tool kit and expect it to do all the work and provide clear results. Rather, deep consideration is required to sort out the significance of the results for the variety of probing techniques.

# 5. ACKNOWLEDGMENTS

Special thanks to SHODOR, the National Computational Science Institute (NCSI), the National Center for Supercomputing Applications (NCSA), and everyone else involved in the Blue Waters Undergraduate Petascale Education Program (BW-UPEP).

The BW-UPEP program supports a number of undergraduate students working on a diverse range of computational science problems. Support includes a two week intensive workshop at the NCSA facility on the University of Illinois – Urbana Champaign campus, pairing with a faculty advisor, a stipend, and a forum for the student to attain help if needed throughout the internship.

The BW-UPEP program has given me (J.C.) the chance to experience a number of things I might not otherwise have been exposed to. The two week workshop was particularly effective in teaching me the tools I would need to effectively carry out my research. Specifically, I learned how to develop parallel algorithms using MPI, OpenMP, and CUDA. Particular to my individual project, I learned how to use many technical tools like TAU and PAPI, and the important role that these tools can play while trying to improve a computationally intensive application. I was also given the opportunity to attend two conferences, Teragrid ’11 (now XSEDE) and SC ’11. I gave poster presentations at both conferences and this provided me with important experiences such as how to create and give scientific presentations. Furthermore, in attending the conferences I was able to meet and interact with members of the broader HPC community which was very inspirational. The experience as a whole has persuaded me to pursue a career in computational science.

Finally thanks to XSEDE and the Pittsburg Supercomputing Center for providing access to the resources that were used to do this project.

# 6. REFERENCES

[1] Blöchl, P.E. 2011. Theory and Practice of Density-Functional Theory. *arXiv:1108.1104*. (Aug. 2011).

[2] Browne, S. et al. 2000. A Portable Programming Interface for Performance Evaluation on Modern Processors. *International Journal of High Performance Computing Applications*. 14, 3 (Aug. 2000), 189–204.

[3] Caruso, A.N. et al. 2009. Direct evidence of electron spin polarization from an organic-based magnet: [Fe[sup II](TCNE)(NCMe)[sub 2]][Fe[sup III]Cl[sub 4]]. *Physical Review B (Condensed Matter and Materials Physics)*. 79, 19 (2009), 195202–5.

[4] Chen, Y. et al. 2002. Ab-initio calculation of Si-K and Si-L ELNES edges in an extended inactive defect model of crystalline silicon. *Materials Transactions*. 43, 7 (2002), 1430–1434.

[5] Ching, W.Y. et al. 2009. Ab initio elastic properties and tensile strength of crystalline hydroxyapatite. *Acta Biomaterialia*. 5, 8 (2009), 3067–3075.

[6] Ching, W.Y. et al. 2006. Ab initio modeling of clean and Y-doped grain boundaries in alumina and intergranular glassy films (IGF) in β-$Si_3N_4$. *Journal of Materials Science*. 41, 16 (2006), 5061–5067.

[7] Ching, W.Y. et al. 2009. Ab initio tensile experiment on a model of an intergranular glassy film in β-$Si_3N_4$ with prismatic surfaces. *Applied Physics Letters*. 94, 5 (2009), 051907–3.

[8] Ching, W.Y. et al. 2010. Theoretical study of the elasticity, mechanical behavior, electronic structure, interatomic bonding, and dielectric function of an intergranular glassy film model in prismatic β-Si_{3}N_{4}. *Physical Review B*. 81, 21 (Jun. 2010), 214120.

[9] Ching, W.Y. 1982. Theory of amorphous $SiO_2$ and $SiO_x$. I. Atomic structural models. *Physical Review B*. 26, 12 (Dec. 1982), 6610–6621.

[10] Ching, W.Y. 1982. Theory of amorphous $SiO_2$ and $SiO_x$. II. Electron states in an intrinsic glass. *Physical Review B*. 26, 12 (Dec. 1982), 6622–6632.

[11] Ching, W.Y. 1982. Theory of amorphous $SiO_2$ and $SiO_x$. III. Electronic structures of $SiO_x$. *Physical Review B*. 26, 12 (Dec. 1982), 6633–6642.

[12] Ching, W.Y. and Rulis, P. 2008. Ab initio calculation of the O-K, N-K, Si-K, Si-L3, Y-K, Y-L3 edges in the Y-Si-O-N system: A strategy for ELNES/XANES spectral modeling in complex materials. *Physical Review B Condensed Matter and Materials Physics*. 77, 3 (2008), 035125/1–035125/17.

[13] Ching, W.Y. and Rulis, P. 2008. Large differences in the electronic structure and spectroscopic properties of three phases of AlPO4 from ab initio calculations. *Physical Review B Condensed Matter and Materials Physics*. 77, 12 (2008), 125116/1–125116/7.

[14] Hohenberg, P. and Kohn, W. 1964. Inhomogeneous Electron Gas. *Physical Review*. 136, 3B (1964), B864.

[15] Honkala, K. et al. 2005. Ammonia Synthesis from First-Principles Calculations. *Science*. 307, 5709 (Jan. 2005), 555–558.

[16] Huang, M.-Z. and Ching, W.Y. 1994. Electronic and transport properties of perfect $sp^2$-bonded amorphous graphitic carbon. *Physical Review B*. 49, 7 (1994), 4987.

[17] Kohn, W. and Sham, L.J. 1965. Self-Consistent Equations Including Exchange and Correlation Effects. *Physical Review*. 140, 4A (1965), A1133.

[18] Liang, L. et al. 2010. Mechanical properties, electronic structure and bonding of a- and b-tricalcium phosphates with surface characterization. *Acta Biomaterialia*. 6, 9 (2010), 3763–3771.

[19] Liang, L. et al. 2009. Theoretical study of the large linear dichroism of herapathite. *Physical Review B (Condensed Matter and Materials Physics)*. 80, 23 (2009), 235132–5.

[20] Martin, R.M. 2004. *Electronic Structure: Basic Theory and Practical Methods*. Cambridge University Press.

[21] Mo, S.-D. et al. 1999. Electronic structure of a grain-boundary model in SrTiO3. *Physical Review B*. 60, 4 (1999), 2416.

[22] PAPI: *http://icl.cs.utk.edu/papi/*. Accessed: 2012-05-30.

[23] Rulis, P. et al. 2004. Ab initio ELNES/XANES spectral calculation of polar and non-polar grain boundaries in β-SiC. *Acta Materialia*. 52, 10 (2004), 3009–3018.

[24] Rulis, P. et al. 2009. Spectroscopic imaging of electron energy loss spectra using *ab initio* data and function field visualization. *Ultramicroscopy*. 109, 12 (2009), 1472–1478.

[25] Rulis, P. and Ching, W. 2011. Theoretical ELNES spectra of Si-K, Si-L, N-K, and O-K edges of an intergranular glassy film model in β-Si3N4. *Journal of Materials Science*. 46, 12 (2011), 4191–4198.

[26] Shende, S.S. and Malony, A.D. 2006. The Tau Parallel Performance System. *International Journal of High Performance Computing Applications*. 20, 2 (May. 2006), 287–311.

[27] Tanaka, I. et al. 2003. Identification of ultradilute dopants in ceramics. *Nat Mater*. 2, 8 (2003), 541–545.

[28] TAU - Tuning and Analysis Utilities: *http://www.cs.uoregon.edu/Research/tau/home.php*. Accessed: 2012-05-30.

[29] Wai-Yim Ching and Paul Rulis 2012. *Electronic Structure Methods for Complex Materials: The Orthogonalized Linear Combination of Atomic Orbitals*. Oxford University Press.

[30] Zhong, X. and Ching, W.Y. 1990. Calculation of local orbital moments of conduction electrons in Nd2Fe14B. *Journal of Applied Physics*. 67, 9 (May. 1990), 4768–4770.

# A Performance Comparison of a Naïve Algorithm to Solve the Party Problem using GPUs

Michael V.E. Bryant
Merrimack College
315 Turnpike Street
North Andover, MA 01845

michael.bryant@merrimack.edu

David Toth
University of Mary Washington
1301 College Avenue
Fredericksburg, VA 22401

dtoth@umw.edu

## ABSTRACT
The R(m, n) instance of the party problem asks how many people must attend a party to guarantee that at the party, there is a group of *m* people who all know each other or a group of *n* people who are all complete strangers. GPUs have been shown to significantly decrease the running time of some mathematical and scientific applications that have embarrassingly parallel portions. A brute force algorithm to solve the R(5, 5) instance of the party problem can be parallelized to run on a number of processing cores many orders of magnitude greater than the number of cores in the fastest supercomputer today. Therefore, we believed that this currently unsolved problem is so computationally intensive that GPUs could significantly reduce the time needed to solve it. In this work, we compare the running time of a naïve algorithm to help make progress solving the R(5, 5) instance of the party problem on a CPU and on five different GPUs ranging from low-end consumer GPUs to a high-end GPU. Using just the GPUs computational capabilities, we observed speedups ranging from 1.9 to over 21 in comparison to our quad-core CPU system.

## General Terms
Parallel Programming, GPGPU, Ramsey Theory

## Keywords
Blue Waters Undergraduate Petascale Internship, CUDA, Party Problem, Performance Comparison



## 1. INTRODUCTION
Ramsey Theory is an area of mathematics concerned with "the mathematical study of combinatorial objects in which a certain degree of order must occur as the scale of the objects becomes large" [1]. Applications of Ramsey Theory include computational geometry, information theory, complexity, and parallelism [2]. The party problem is a problem in Ramsey Theory. The R(m, n) instance of the party problem asks how many people must be invited to a party (assuming that all the invitees will attend) to guarantee that at the party, there is a group of *m* people who all know each other or a group of *n* people who are all complete strangers. Thus, the solution to the R(5, 5) instance of the party problem indicates the fewest number of people required to attend a party to guarantee that at the party, there will be a group of 5 people who all know each other or a group of 5 people who are all complete strangers. While the party problem has been solved for some small values of *m* and *n*, it has yet to be solved for values of m and n that are both equal and at least 5 [3]. For several of these cases, the bounds on the answers to the problem have been established [3]. For example, it is known that $43 \leq R(5, 5) \leq 49$ [3].

To help visualize the party problem, 2-colored graphs are typically used to show the relationship between the people as either acquaintances or strangers. The graphs consist of vertices representing the people and edges that connect every vertex to every other vertex, forming a complete graph. A red edge connecting two vertices indicates that the vertices represent people who are strangers and a blue edge connecting two vertices indicates that the vertices represent people who are acquaintances. The graphs that are used to help visualize the problem can also be used to solve the problem. To solve R(m, n), one must find the smallest number of people *x* such that every complete graph on *x* vertices where the edges are colored blue and red contains a subgraph that is a red or blue complete graph on 5 vertices. Since the number of edges in a complete graph on *v* vertices is $(v^2\text{-}v)/2$ and each edge may be colored red or blue, there are $2^{((v\text{^}2)\text{-}v)/2}$ complete graphs on v vertices. Therefore, to increase the lower bound on R(5, 5) from the known lower bound of 43 to 44, one would need to demonstrate that at least one of the $2^{((43\text{^}2)\ -\ 43)/2}$ or $2^{903}$ complete graphs on 43 vertices does not contain a subgraph that is a red or blue complete graph on 5 vertices, denoted $K_5$. Alternatively, one could decrease the upper bound from the known value of 49 to *u* by demonstrating that every complete graph on *u* vertices where the edges are colored blue and red contains a subgraph that is a red or blue $K_5$. We do note that many of the $2^{903}$ graphs that are isomorphic to each other, and

thus testing one of those graphs eliminates the need to test the other graphs that are isomorphic to it. Unfortunately, it is often slower to test if a graph is isomorphic to another graph that has already been tested than to test if the graph contains a red or blue $K_5$. We also note that for every graph, another graph in the $2^{903}$ graphs can be obtained and if the original graph contains a red $K_5$, then the graph obtained by reversing the edge colors must contain a blue $K_5$. Therefore, only half of the $2^{903}$ graphs need to be tested and this can be accomplished relatively easily.

We chose to write our algorithm to attempt to increase the lower bound from the known value of 43 to 46 because of a conjecture that R(5, 5) = 46 [4]. In the computationally worst case scenario, our algorithm would test every graph and determine either that R(5, 5) > 45 if all the graphs other than the last one contained a red or blue $K_5$ or that R(5, 5) $\leq$ 46 if every complete graph on 45 vertices with red and blue edges contained a red or blue $K_5$. If we instead tried to decrease the upper bound on R(5, 5) by one, the worst case scenario would involve testing more graphs.

## 2. RELATED WORK

Over the past several years, many researchers have begun to use GPUs to do a wide variety of mathematical and scientific calculations that have traditionally been done on CPUs. GPUs have been used to run "space and time discrete simulations" called stencil codes [5]. They have been used for problems in physics such as solving Boltzmann equations for gas flow applications [6]. Scientists have used GPUs in bioinformatics for doing sequence alignment [7], for correcting errors in DNA sequencing [8], and weather forecasting [9]. GPUs are also being used for colorectal cancer research [10], cheminformatics [11], finite element numerical integration [12], n-body problems [13], dynamic programming applications [14], and many other scientific and mathematical applications. GPUs are being used as components in some of the newest and fastest supercomputers because of their ability to provide significant computational capability, while using less power than some CPU alternatives. The success that researchers have had with a wide variety of applications motivated us to evaluate whether GPUs would decrease the amount of time to solve the party problem.

## 3. DATA STRUCTURES AND ALGORITHM

As we designed our application, we were mindful of the idea that it would need to run efficiently in parallel on both multicore systems using OpenMP and on GPUs using CUDA with minimal revision to the code to port it from the CPU to the GPUs. The key implication of this is that the code had to run with a minimal amount of communication between processing cores when running on the CPU only and when running on the GPUs only. Because our application just requires us to test a large number of graphs, it is really just an embarrassingly parallel application. Therefore, we realized that we could simply divide the number of graphs evenly by the number of cores (CPU or GPU as appropriate), have each core report the results of testing its share of graphs to a master processor, and have that processor output the results. As long as each core would be able to efficiently generate the next graph that the core was supposed to test, this design would work well. This requirement affected the data structure we used to represent the graphs.

### 3.1 Data Structures

As discussed in section 1, the party problem can be solved using graphs. In computer programs, a graph is typically represented using adjacency lists or an adjacency matrix. For our purposes, an adjacency matrix is preferable to adjacency lists for both performance and ease of implementation reasons. While a natural representation of a graph using an adjacency matrix is a two-dimensional array, for our application, a single-dimensional array was more useful. To help illustrate this, we show a simple graph with five vertices and the corresponding two-dimensional adjacency matrix in Figure 1 where a red edge is represented as a zero, a blue edge is represented as a one, and no edge is represented as -9. The adjacency matrix in Figure 1 indicates there are no edges from a vertex to itself, because in the party problem, a person is assumed to know himself/herself and thus there is no need for an edge from a vertex to itself. Therefore, the information on the diagonal of the matrix from the upper left entry to the lower right entry does not need to be stored. Because the relationships between two people are symmetric (if person 1 knows person 2, then person 2 knows person 1), the information above the diagonal containing -9s is the same as the information below the diagonal and is therefore redundant. By only storing the information above the diagonal, we can save a little more than half the memory that would be required to store the information in the graph. Because many GPUs have large numbers of processing cores, each with fast access to a limited amount of the typically small quantity of memory on the GPU, this optimization was important to allow all the cores to evaluate different graphs in a single instant. In order to save memory, we "flatten" the matrix and store it as a one-dimensional array as shown in Figure 2. While this introduced some complexity during development, the advantages it provided far outweighed the added complexity. Since the flattened array contains only zeros and ones, the digits it contains may be thought of as a binary number representing the graph. By adding one to the rightmost element of the array and making any carries if appropriate to the elements to the left, one has adjusted the array to contain the binary representation of the next graph. By continuing to add one to the digit in the rightmost array slot and making the appropriate carries, one can generate all of the possible graphs, one at a time. This process is a very efficient way for a processing core to generate the next graph to test, and the primary advantage of using the one- dimensional array representation of the graph. By assigning equally sized sets of graphs to each core where the graphs in a set can be represented as consecutive binary numbers, the cores were able to efficiently generate each subsequent graph that they needed to test, making the one-dimensional array an excellent data structure for storing the graphs.

### 3.2 Algorithm

The algorithm we used was a naïve algorithm. The algorithm tested the first 335,544,320,000 (which is between $2^{38}$ and $2^{39}$) graphs of the $2^{990}$ graphs that need to be tested to determine if R(5, 5) $\geq$ 46. We refer to the algorithm as naïve for two reasons. The first reason is that the algorithm is a brute force approach to tightening the lower bound on the value of R(5, 5). Even more importantly, the algorithm was not optimized for running on the GPUs. For many applications, it is possible to get significantly better performance on algorithms that are optimized to run on GPUs. However, we were more concerned with the speedup that could be obtained with minimal effort.

In the version of the code that the CPU ran and the version that the GPUs ran, each of the $n$ processing cores was given the value of $n$ and assigned $1/n^{th}$ of the graphs to test by being given a

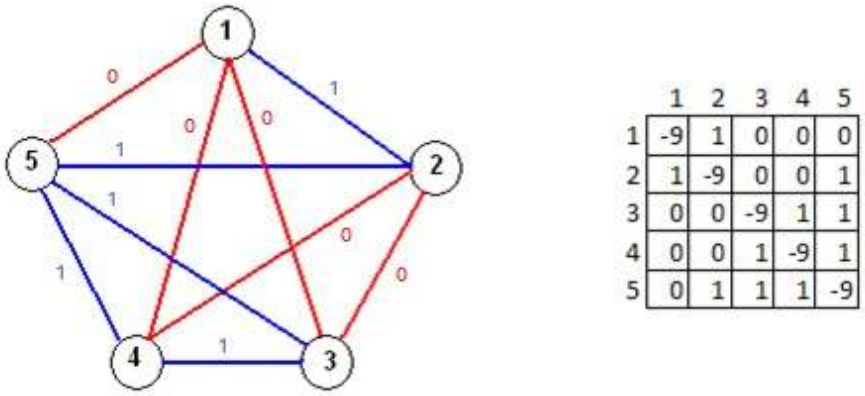


**Figure 1. A Five Vertex Graph and Its Adjacency Matrix**

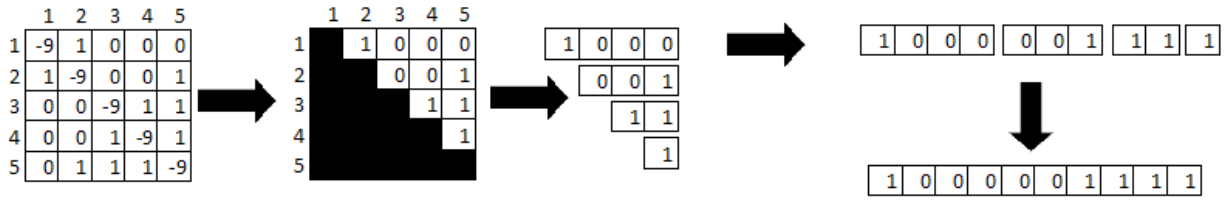


**Figure 2. Flattening the Two-Dimensional Array to a One-Dimensional Array Keeping Only the Necessary Information**

decimal value that the core converted into a binary number, corresponding to the first graph in the range of graphs to test. The cores were then able to calculate the number of graphs they needed to test. The algorithm we used to test a graph to see if it contains a red or blue $K_5$ cycles through all of the possible sets of five vertices until it finds a set that forms a red or blue $K_5$ or until it has determined that no set of five vertices can contain a red or blue $K_5$. Once a $K_5$ is found, the graph is discarded and the next graph is generated by adding 1 to the binary representation of the discarded graph to obtain the next graph to test. In the OpenMP version of the code for the CPU, whenever a core found a graph with neither a red or blue $K_5$, it was supposed to print out the graph, but this situation never arose. In the CUDA version of the code, whenever a core found a graph with neither a red or blue $K_5$, it was supposed to write the graph to a shared memory location and set a flag in shared memory. When the cores were all finished, the flag was copied to the CPU and if a graph with no $K_5$ had been found, then the shared memory with the graph's information would be copied to the host system and printed out.

## 4. METHODOLOGY

We did our testing on an upgraded Gateway GT5674 computer with an AMD Phenom 9500 2.2 GHz quad-core CPU, 4 GB of RAM, and a 650 watt power supply. The computer ran the Windows Vista operating system. The GPUs we used ranged from a low end home user card (GeForce 9500 GT) to a high end home user card (GeForce GTX 480) to a high end workstation card (Quadro FX 5800). The GPUs had varying numbers of CUDA cores, multiprocessors, memory, and different compute capabilities, as shown in Table 1.

In order to determine the performance improvement we could obtain by using the GPUs, we first ran a version of the code using OpenMP on the system using 1, 2, and 4 cores. We then ported the code to CUDA such that it would not use the CPU cores for testing any graphs, thus using the CPU minimally, and ran it on each GPU shown in Table 1.

For the performance testing, we chose to have the program test 335,544,320,000 graphs. That number was chosen because it was a multiple of the number of cores in every one of the GPUs we used and the algorithm took about an hour to test that many graphs using all 4 CPU cores. By picking a value that took the CPU a long time, we expected to see what, if any, performance gains we could reasonably expect under normal conditions of the cores having a huge number of graphs to test, which is what we would encounter if we tried to solve the problem with GPUs or CPUs.

Our only effort to tune the CUDA version of the application in an attempt to achieve better performance was to run the tests on each card using several different numbers of threads per block. In order to achieve high performance and scalability, NVIDIA GPUs divide the threads for a multithreaded CUDA program into groups of threads called blocks [16]. Each block runs on a streaming multiprocessor, so the number of blocks that can run simultaneously and thus the performance of an NVIDIA GPU

**Table 1. Specifications for GPUs Used**

| GPU | CUDA Cores | Streaming Multiprocessors | Memory (GB) | Compute Capability |
|---|---|---|---|---|
| GeForce 9500 GT | 32 | 4 | 1 | 1.1 [15] |
| GeForce GT 240 | 96 | 12 | 0.5 | 1.2 [15] |
| GeForce GTS 450 | 192 | 4 | 1 | 2.1 [15] |
| Quadro FX 5800 | 240 | 30 | 4 | 1.3 [15] |
| GeForce GTX 480 | 480 | 15 | 1.5 | 2.0 [15] |

scales with the number of streaming multiprocessors [16]. One can adjust the number of threads that run in each block to try to obtain better performance by using more blocks, to a maximum of 512 threads per block on GPUs with compute capability 1.x and 1024 threads per block on GPUs with compute capability 2.x [16]. Since we had GPUs with both 1.x and 2.x compute capabilities, we chose to limit the threads per block to 512 for all of our testing for consistency. Because some people have suggested that the number of threads per block should be at least 64 as well as being a multiple of 64 [17] and others have found that 128 to 256 threads per block resulted in the best performance for their applications [18], we also ran tests with 64, 128, and 256 threads per block in addition to 512 threads per block. The GeForce 9500 GT which was unable to run the code using 512 threads per block due to hardware limitations, so we were only able to test it with 64, 128, and 256 threads per block on that GPU. We discuss the results of these tests in Section 5.

In order to try to take full advantage of the GPU for computing rather than also for updating the computer's display, we explored the possibility of using a second graphics card dedicated to the display in our test system. The GPUs with compute capability 1.x were compatible with a second NVIDIA graphics card, an NVIDIA GeForce 6200 that is not CUDA-enabled. This allowed us to use the GeForce 6200 for the display, freeing up the CUDA cards to do only the computations. The cards with compute capability 2.x did not work with the second graphics card. Therefore, we tested all the GPUs without the GeForce 6200. We also tested cards with compute capability 1.x while using the GeForce 6200 for the display. This allowed us to determine if using a GPU for both computations and to run the display had a significant impact on the performance of the cards. We discuss the results of these tests in Section 5.

# 5. RESULTS

We conducted a test consisting of 10 trials where each of the devices (CPU-only using 1, 2, and 4 cores and GPU-only with each GPU) tested 335,544,320,000 graphs. Each test was repeated for each GPU using each number of threads per block except 512 threads per block for the GeForce 9500 GT. For the GPUs that were compatible with the second graphics card, we conducted 10 trials using the second graphics card to drive the display with each number of threads per block other than 512 blocks for the GeForce 9500 GT. After analyzing the results of 10 trials for each test, we concluded that the range of the results for each GPU and for each number of CPU cores used was sufficiently small to conclude that 10 trials was sufficient to be confident that we were getting accurate results. In 24 of the 30 GPU tests, the range between the values obtained from the 10 trials was 2 seconds or less. In the remaining 6 tests, the ranges were one of 3 seconds (test average 603 seconds), one of 5 seconds (test average 2241 seconds), two of 6 seconds (test averages 634 seconds and 3167 seconds), one of 10 seconds (test average 2258 seconds), and one of 13 seconds (test average 637 seconds). The only GPUs that had a range of more than 1 second for the 10 trials in any given test were the GeForce 9500 GT and the GeForce GT 240, which are the lower end home GPUs. For the CPU core tests, the ranges for the tests were 36 seconds (test average 3254 seconds) for 4 cores, 88 seconds (test average 6846 seconds) for 2 cores, and 130 seconds (test average 13,226 seconds) for 1 core.

## 5.1 Effects of Using the GPU to Drive Display

We observed that using the GPU to drive the system's display in addition to testing the graphs did increase the amount of time taken to finish testing the graphs, as shown in Table 2. The least powerful GPU (the GeForce 9500 GT) consistently suffered the biggest performance loss in terms of additional time to finish testing the graphs. However, when taking the additional time as a percentage of the time to test the graphs when not using the GPU, the most powerful GPU (the Quadro FX 5800) suffered the biggest performance loss. The increase in time required to test the graphs was only an additional 2.5% or less of the time taken to complete the computations without using the GPU for the system's video output. Therefore, we do not believe using the GPUs to run the systems video is a significant detriment to the performance of the GPU on the computations.

## 5.2 Effect of Using Different Numbers of Threads Per Block

As discussed in the Section 4, the number of threads per block can have an impact on the performance of the GPUs, so we ran the tests using 64, 128, 256, and 512 threads per block for each card except 512 for the GeForce 9500 GT. We found that there was a significant variation on the running time of the algorithm using different numbers of threads per block, as shown in Figure 3. The data shown in Figure 3 only shows the variation based on threads per block for the GPUs when the second graphics card was not in the system for consistency, but we observed that the data from the tests where the second graphics card was used demonstrated analogous results. In Table 3, we show the percent increase in time required to test the graphs from the tests using the number of threads per block that produced the fastest time to the tests using the number of threads per block that produced the slowest time. The increase in the time required to test all the graphs ranged from 8% to 54% based on the GPU. Although we

observed the least impact on the GPU with the most memory, the GPU with the least memory had the second smallest impact which was significantly less than the other GPUs, so the magnitude of the impact was not simply based on the amount of memory of the GPU. 128 threads per block produced the best results with the GeForce GT 240, Quadro FX 5800, and GeForce GTX 480. The GeForce 9500 GT performed best with 64 threads per block, but was only 4 seconds faster than it was with 128 threads per block. The GeForce GTS 450 performed best with 256 threads per block, but only 10 seconds better than with 128 threads per block. Therefore, when comparing the performance of the GPUs, we use the performance from the test with the number of threads per block that was most commonly the best for the GPUs. We note the difference between the speedup obtained by using the performance when the GPUs used 128 threads per block and the performance from the best number of threads per block would be negligible.

**Table 2. Performance Comparison of GPUs When Used for Video vs. When Not Used for Video**

| GPU | Threads Used Per Block | Average Time (seconds) When Using GPU for Video | Average Time (seconds) When Not Using GPU for Video | Additional Time Taken (seconds) | Additional Time as Percent Difference from Average Time When Not Using GPU for Video |
|---|---|---|---|---|---|
| GeForce 9500 GT | 512 | ------ | ------ | ------ | ------ |
| GeForce 9500 GT | 256 | 3167.4 | 3156.3 | 11.1 | 0.4 |
| GeForce 9500 GT | 128 | 2257.8 | 2242.4 | 15.4 | 0.7 |
| GeForce 9500 GT | 64 | 2253.9 | 2241.3 | 12.6 | 0.6 |
| GeForce GT 240 | 512 | 636.9 | 634.0 | 2.9 | 0.5 |
| GeForce GT 240 | 256 | 538.0 | 531.2 | 6.8 | 1.3 |
| GeForce GT 240 | 128 | 524.3 | 521.4 | 2.9 | 0.6 |
| GeForce GT 240 | 64 | 604.7 | 604.1 | 0.6 | 0.1 |
| Quadro FX 5800 | 512 | 297.6 | 292.4 | 5.2 | 1.8 |
| Quadro FX 5800 | 256 | 289.0 | 283.8 | 5.2 | 1.8 |
| Quadro FX 5800 | 128 | 275.8 | 269.1 | 6.7 | 2.5 |
| Quadro FX 5800 | 64 | 277.1 | 272.3 | 4.8 | 1.7 |

## 5.3 Speedups

The speedups attained by using the GPUs over using one, two, and four cores of the CPU in our quad core system are shown in Table 4. We note that it is strange that slightly greater than linear speedup was observed when running the program on the CPU only with all 4 cores. One would expect that there would be a small performance loss, rather than performance gain in that situation. We suspect that there might have been some process running on the computer such as antivirus software or a regularly scheduled task from the operating system that did not occur during the time the program was run with 4 cores, but did occur when the program was run with one core and two cores. The GPU with the fewest cores attained a small speedup of 1.44 over using all of the cores of the CPU, meaning it would take about 1.44 identical quad core systems to test the graphs in the same amount of time that GPU did. In contrast, a high-end GPU (GeForce GTX 480) attained a speedup of greater than 21 over the quad-core system using all of the cores of the CPU, meaning it would take over 21 identical quad core systems to test the graphs in the same amount of time that GPU did.

# 6. CONCLUSIONS

Our results have shown that a brute force solution to the party problem would be greatly speeded up running on GPUs, as our fastest GPU did the same amount of work as our host system in a fraction of the time. It would take more than 21 of our quad core systems or about 88 of the CPU cores in our host computer to do the same amount of work as the GPU in the same amount of time. We note that although faster CPUs are now available, they should still be significantly outperformed by the single GPU. Also, with systems available that can run up to 8 GPUs and with faster GPUs available, it appears that using GPUs would be a good way to make progress on the party problem. However, while we have seen the great potential of GPUs to speed up the process of making progress on the party problem, we have two concerns. Our program only tested graphs that allowed the cores to run continuously in parallel. When certain graphs much further along in the set of graphs to test begin to be tested, due to the GPU architecture, the code of our algorithm will have to branch in various places, which will force it to run sequentially at times, decreasing the performance gains we got from the GPUs. An issue that is much more problematic than the branching is that the brute force solution requires us to test $2^{990}$ graphs to raise the lower bound on R(5, 5) to 46 or to move the upper bound to 46. Our idea of using GPUs does not scale sufficiently by itself to accomplish this in a reasonable amount of time. While we were able to test more than $2^{38}$ graphs in 2.5 minutes using a small quantity of hardware available to any consumer, a back of the envelope calculation suggests that it would take more than a year on the Titan supercomputer which has 18,688 GPUs to test just the first $2^{66}$ graphs, leaving us too far from completing the tests we need to solve the problem [19]. Even a new generation of supercomputers with many times the number of GPUs as current supercomputers where the GPUs were also many times faster and had many times the number of cores as the current GPUs would still not let us solve the problem. Therefore, to devise a workable method to solve the party problem in a

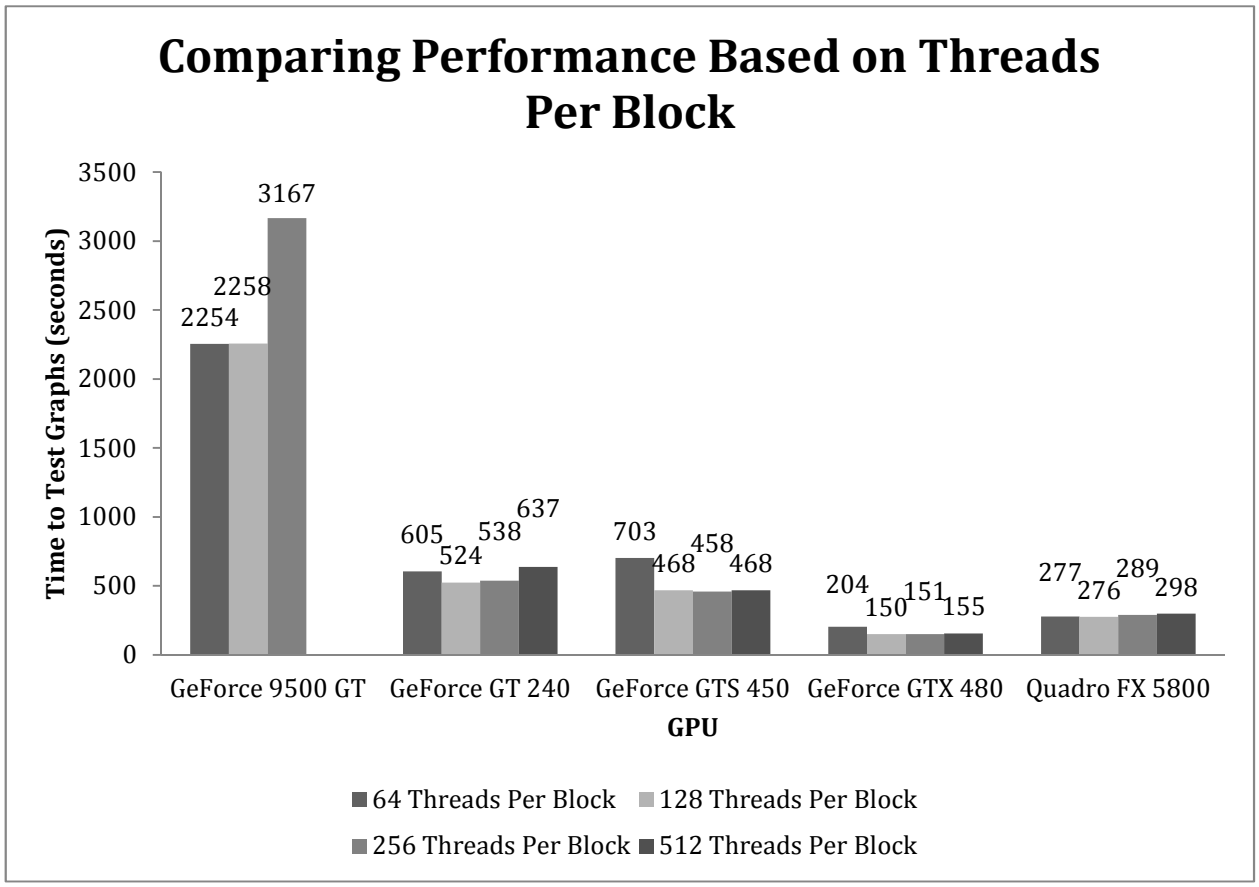


**Figure 3. Comparison of Running Time of GPUs Using Various Numbers of Threads per Block**

**Table 3. Comparison of Percent Increased Time Required by GPUs to Test Graphs with Best and Worst Numbers of Threads Per Block**

| GPU | Percent Time Increase |
|---|---|
| Quadro FX 5800 | 8 |
| GeForce GT 240 | 21 |
| GeForce GTX 480 | 36 |
| GeForce GTS 450 | 41 |
| GeForce 9500 GT | 54 |

reasonable amount of time, we will need to devise a better algorithm than just the naïve brute force algorithm. If the algorithm is amenable to running on GPUs, perhaps using GPUs will help us solve the problem in the future.

## 7. FUTURE WORK

There are several possible opportunities for future work related to this research. The first strategy we could try is to use a more intelligent algorithm than the brute force algorithm our program employed. Using mathematics, we could decrease the set of graphs to test by eliminating classes of graphs that can be shown will contain a red $K_5$ or blue $K_5$ subgraph. This may require us to devise a significantly more complicated algorithm to run on the GPUs, as the GPU cores may no longer be able to generate the next graph to test using the efficient method our current algorithm employed. We could also expand upon work that for a CPU-based system, used a breadth first search technique that employed pruning to try to search the tree of all possible graphs to generate only graphs with at least 43 vertices that contain neither a red or blue $K_5$ [20].

**Table 4. Speedup of Devices**

| Device | Average Time (seconds) | Speedup Over Single Core | Speedup Over Quad Core System |
|---|---|---|---|
| CPU using 1 core | 13226 | -------- | -------- |
| CPU using 2 cores | 6846 | 1.93 | -------- |
| CPU using 4 cores | 3254 | 4.06 | -------- |
| GeForce 9500 GT | 2258 | 5.86 | 1.44 |
| GeForce GT 240 | 524 | 25.22 | 6.20 |
| GeForce GTS 450 | 468 | 28.26 | 6.95 |
| Quadro FX 5800 | 276 | 47.92 | 11.79 |
| GeForce GTX 480 | 150 | 88.18 | 21.69 |

Another opportunity to explore this problem in more depth is to determine the binary representation of the first graph that causes the CUDA code we wrote to branch and thus run sequentially. We also would like to measure the performance of the cards during the period when the graphs cause the code to branch and compare it to the performance of the OpenMP version of the code running on just the CPU cores for the same set of graphs.

# 8. REFLECTIONS

This research project was a result of The Blue Waters Undergraduate Petascale Education Program with which I participated in as a student intern. The BWUPEP is an educational program that gives students from colleges and universities around the country the opportunity to learn about the high performance computing discipline and apply what they learn in specific research projects at their home institutions. Previous to my experience with the Blue Waters Undergraduate Petascale Education Program internship, I had limited knowledge about the field of computer science and parallel computing. I knew this internship would be the opportunity of a lifetime for me to get my foot into the door of the computer science world. Over the course of the internship, I learned more than I ever imagined and have gained experience that will be valuable to me for years to come. I am now confident in my abilities to handle issues related to HPC and computationally intense problems.

My Blue Waters Undergraduate Petascale Education Program internship began with a two-week trip to attend Blue Waters Undergraduate Petascale Institute at NCSA at the University of Illinois Urbana-Champaign to learn about High Performance and Parallel Computing. There I met new students from around the country who were also awarded the internship to conduct research at their respective universities and colleges. During those two weeks we underwent intense educational sessions everyday learning the methodologies and ideologies of High Performance Computing and methods used to develop and debug code on parallel computers and clusters. From this boot camp experience, I took back with me the knowledge and ability to conduct my research for this project and future work, as well as new friendships I had formed.

The bonus of my BWUPEP internship and the culmination of all my hard work involved a free trip to SC11 in Seattle, Washington to work as a student volunteer and participant in the education program with the other student interns and volunteers. I created a poster to be presented at the resource fair about my research experience as well as the research I conducted over the summer for this project. Going to SC11 was an eye-opening experience and I hope to attend more of the SC conferences in the future. I got the chance to meet many students in multiple disciplines of computation as well as network with professors and major companies who are involved in supercomputing.

This internship has provided me with the tools and knowledge I need to move forward and continue work and research in the field of high performance and parallel computing. Over the course of the internship, I have learned the basics of a computer's hardware and architecture and how the hardware works along with the operating system to conduct multiprocessing and parallel computing. I have been exposed to multiple ways to parallelize code using OpenMP, MPI, and CUDA. The mentors at the boot camp in Illinois instilled in me the ideologies of high performance computing and taught me the systematic methods to successfully debug parallelized code. I experienced hands on learning by creating several small-scale local clusters on my own with the resources I had available at my home institution. I also learned about Graph Theory, Ramsey Theory, and the mathematics they involve, catching a glimpse of the numerous computationally intense problems of today.

This internship has given me the opportunity to experience first hand the process of research in an academic environment, which I will continue to use in my future work. This experience as a whole has provided me with the knowledge, tools, and experience to be successful in future endeavors. My research work with my mentor, Dr. David Toth, has opened doors for me that I never knew existed. The work I have done for this project has sparked my interest for more computationally intense problems that might hold some potential for me to work on using HPC in the future.

# 9. ACKNOWLEDGEMENTS

This work was supported by the Blue Waters Undergraduate Petascale Education Program funded by the National Science Foundation's Office of CyberInfrastructure, which provided support for one of the authors in the form of a Blue Waters Undergraduate Petascale Research Internship. This work was also supported by NVIDIA through their Academic Partnership

Program, which provided the GeForce GTX 480 and Quadro FX 5800 GPUs we used.

## 10. REFERENCES

[1] ramsey theory - Wolfram|Alpha. (2012). http://www.wolframalpha.com/input/?i=ramsey+theory.

[2] Vera Rosta, Ramsey Theory Applications, The Electronic Journal of Combinatorics. December 7, 2004, http://www.combinatorics.org/ojs/index.php/eljc/article/view/ds13/pdf.

[3] S. P. Radziszowski, Small Ramsey Numbers, The Electronic Journal of Combinatorics. DS1.10. (originally published July 3, 1994, last updated August 4, 2009), http://www.combinatorics.org/Surveys/ds1/sur.pdf.

[4] Personal communication with Peter Christopher, Ph.D., January 2005.

[5] A. Schafer, D. Fey, High Performance Stencil Code Algorithms for GPGPUs, Procedia Computer Science 4 (2011) 2027-2036.

[6] Y. Y. Kloss, P. V. Shuvalov, F. G. Tcheremissine, Solving Boltzmann equation on GPU, Procedia Computer Science 1 (2010), 1083-1091.

[7] C. Ling, K. Benkrid, Design and Implementation of a CUDA-Compatible GPU-based Core for Gapped BLAST Algorithm, Procedia Computer Science 1 (2010) 495-504.

[8] H. Shi, B. Schmidt, W. Liu, W. Muller-Wittig, Quality-score guided error correction for short-read sequencing data using CUDA, Procedia Computer Science 1 (2010) 1129-1138.

[9] T. Shimokawabe, T. Aoki, J. Ishida, K. Kawano, C. Muroi, 145 TFlops Performance on 3990 GPUs of TSUBAME 2.0 Supercomputer for an Operational Weather Prediction, Procedia Computer Science 4 (2011) 1535-1544.

[10] GPGPUGRID.net, News archive (2011), http://www.gpugrid.net/old_news.php.

[11] M. Maggioni, M. D. Santambrogio, J. Liang, GPU-accelerated Chemical Similarity Assessment for Large Scale Databases, Procedia Computer Science 4 (2011), 2007-2016.

[12] P. Maciol, P. Plaszewski, K. Banas, 3D finite element numerical integration on GPUs, Procedia Computer Science 1 (2010), 1093-1100.

[13] L. Nyland, M. Harris, J. Prins, Fast N-Body simulation with CUDA, in: GPU Gems, vol. 3, Addison Wesley, 2007, 677-795.

[14] S. Che, M. Boyer, J. Meng, D. Tarjan, J. Scheaffer, K. Skadron, A performance study of general-purpose applications on graphics processors using CUDA, J. Parallel Distrib. Comput. 68 (2008), 1370-1380.

[15] NVIDIA, CUDA GPUs | NVIDIA Developer Zone (August 2011) - http://developer.nvidia.com/cuda-gpus.

[16] NVIDIA CUDA C Programming Guide 3.2 (November 2010) – http://developer.nvidia.com/object/cuda_3_2_downloads.html.

[17] S. Walkowiak, K. Wawruch, M. Nowotka, L. Ligowski, W. Rudnicki, Exploring utilization of GPU for database applications, Procedia Computer Science 1(2010) 505-513.

[18] V. W. Lee, C. Kim, J. Chhugani, M. Desiher, D. Kim, A. D. Nguyen, N. Satish, M. Smelyanskiy, S. Chennupaty, P. Hammarlund, R. Singhal, P. Dubey, Debunking the 100X GPU vs. CPU Myth: An Evaluation of Throughput Computing on CPU and GPU, Proceedings of the 37th annual international symposium on Computer architecture (2010) 451-454.

[19] Introducing Titan | The World's #1 Open Science Supercomputer (2012), http://www.olcf.ornl.gov/titan/.

[20] S. Krach, A High Performance Computing Approach to Ramsey Theory, undergraduate thesis, Department of Computer Science, Merrimack College, 2011.

December 2012

Volume 3 Issue 2

# TABLE OF CONTENTS



ISSN 2153-4136 (online)