January 2017

Volume 8 Issue 1

JoCsE

Journal Of Computational Science Education

Promoting the Use of Computational Science Through Education

ISSN 2153-4136 (online)

Volume 8, Issue 1 **January 2017**

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 8 Issue 1

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

Forward

This issue begins with an article by Shiflet et.al. on using HPC for genomic sequence alignment. They present an overview of an online educational module that employs a sequential algorithm to determine the alignments of two DNA sequences. The module also describes several approaches to parallelization and speedup. The module is evaluated based on its use in a bioinformatics course at University "Magna Græcia" of Catanzaro, Italy.

The article by Jung, Zirpoli, and Slick provides an overview of a computational chemistry module that helps students to visualize a complex organic chemistry reaction. The module was used in an undergraduate course to help students understand the thermodynamics and other aspects of the reaction.

The three student articles detail the findings of several intern experiences. Collins discusses the use of augmented reality to visualize the molecular structures from the Protein Data Bank. Maringanti describes the creation of an algorithm to create a graph for a complex integrated circuit design. Finally, Nguyen presents an approach for the parallelization of database queries for large-scale distributed systems.

All of the student articles are a product of their participation in the Blue Waters Student Internship program.

Implementation of Computational Aids in Diels-Alder Reactions: Regioselectivity and Stereochemistry of Adduct Formation

Jiyoung Jung
Department of Science,
Penn State University
Worthington Scranton, PA 18512
1-570-963-2559
juj23@psu.edu

Susan Zirpoli
Department of Chemistry,
Slippery Rock University
Slippery Rock, PA 16057
1-724-738-2716
susan.zirpoli@sru.edu

Glenn Slick
Department of Science,
Penn State University
Worthington Scranton, PA 18512
1-570-963-2559
gss5127@psu.edu

ABSTRACT
The Diels-Alder reaction is one of the most well-known organic reactions and is widely used for six-membered ring formation. Regio- and stereo-selective Diels-Alder reactions have been emphasized in various areas including pharmaceutical and polymer industries. However, covering the theoretical background of such reactions in an undergraduate class is challenging because the interactions between molecular orbitals is poorly visualized for students. Especially when dealing with polycyclic aromatic hydrocarbons (PAHs) and asymmetric compounds, the complexity of regio- and stereo-selectivity becomes more pronounced. Herein we utilized web-based computational tools (WebMO) to visualize the HOMO-LUMO of each reaction component and their interaction to form chemical bonds. In this study we demonstrated the incorporation of computational aids into a Diels-Alder laboratory class dramatically facilitates students' understanding of several important concepts including frontier orbital theory, thermodynamics of the reaction, three-dimensional visualization, and so on. The assessment of teaching effectiveness prior to and after implementation of computational aids into Diels-Alder reactions will also be discussed in this manuscript.

CCS Concepts
• Social and Professional Topics → Computational Science and Engineering Education.

Keywords
Diels-Alder reaction, Molecular orbital, Geometry optimization, Computational calculation, WebMO, Education, Undergraduate laboratory.

DOI: https://doi.org/10.22369/issn.2153-4136/8/1/1

1. INTRODUCTION
As computational chemistry has significantly contributed to chemical research, many chemistry programs across the country now incorporate computational chemistry in their undergraduate curriculum [1-4]. In the last decade, leveraging such modern technology including open-source computational tools into actual undergraduate laboratories has been published in journals. Inspired by such efforts [5, 6], we recently implemented a web-based computational aid (WebMO, MOPAC with PM3) in our organic chemistry laboratory course where students have difficulty in understanding physical organic chemistry concepts.

Diels-Alder cycloaddition reactions are widely used in synthetic organic chemistry since it was first described by Otto Diels and Kurt Alder in 1928 [7, 8]. Due to its interesting mechanism and accessibility to upper-level undergraduate coursework, there have been a variety of reports which involve Diels-Alder reaction in undergraduate laboratory classes [9-12]. The challenges of effective teaching the Diels-Alder reaction, however, stem from understanding the three-dimensional perspective of the molecules and molecular orbitals in an adduct formation which most undergraduate students have poorly understood in class. Cycloadditions such as the Diels-Alder reaction involve the concerted forming and breaking of bonds within a closed ring. In order to understand the reaction mechanism, conservation of orbital symmetry has been used to predict how cycloaddition can occur and what adducts will be produced. Due to complexity including prediction of regioselectivity, stereochemistry and reactive positions, these concepts tend to be one of the most challenging sections to comprehend for undergraduate students. Such complexity becomes more pronounced when students deal with polycyclic aromatic hydrocarbons such as anthracene which is a well-known diene in undergraduate laboratory classes [12]. In this manuscript the integration of web-based computational aids with the hands-on synthetic experiment and its effectiveness in facilitating students' understanding will be discussed.

2. EXPERIMENTAL DESIGN
The computational procedure of this study is based on the exemplified lab exercises which can be found in the website of *Computational Chemistry for Chemistry Educators* (CCCE) and the CCCE workshop [13, 14]. To evaluate the effectiveness of computational aids in understanding a Diels-Alder reaction in the sophomore level class, the study was conducted for eight semesters and observed over eighty students who were enrolled in both the lecture and the corresponding laboratory sections in each semester.

This particular lab was achieved over a series of five assignments which consisted of one class lecture and two lab periods as outlined in Figure 1.

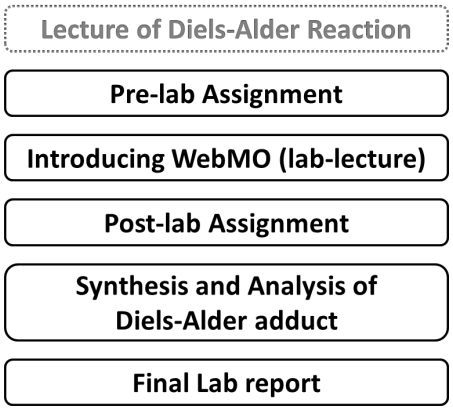

Figure 1. Class design for computational-aid of Diels-Alder reaction experiment. Gray color indicates that Diels-Alder reaction is covered in a regular organic chemistry class.

The class design consists of two homework assignments (pre-lab and post-lab assignments) and one lab-lecture and the actual performance of Diels-Alder reaction. Students were first introduced to Diels-Alder reactions in a regular organic chemistry lecture prior to the pre-lab assignment. The assignment asked a series of fundamental questions pertaining to the identification of the diene and dienophile, location of reactive sites, predicting products and sketching reactants' molecular orbitals. After completing the initial assignment, students were introduced to computational calculations using WebMO software which can calculate geometric optimization, molecular orbitals, and heats of formation, and etc. Using the information collected by computational aids in hand, the post-lab assignment addressed the similar set of questions with anthracene and maleic anhydride which are used as reactants in the actual Diels-Alder reaction experiment. (Table 2; The actual pre-lab and post-lab assignments can also be found in Supporting Material). After completing the computational calculations in the post-lab assignment, reaction between anthracene and maleic anhydride was performed in the laboratory. Therefore students have much better understanding about the Diels-Alder reaction by studying the theoretical calculations before they actually conduct the synthetic experiment. The product of the reaction was then identified using a melting point determination and NMR spectroscopy analysis. Experimental detail including synthetic procedure and discussion points is attached in Supporting Material.

3. COMPUTATIONAL TOOLS

WebMO is a freely available web-based tool which provides an access to multiple computational engines such as MOPAC, Gaussian, and DFT [15]. In this experiment, the MOPAC engine and PM3 theory was used to obtain molecular orbitals and geometric optimization of the proposed dienes/dienophiles and products. Once a calculation is completed, optimized geometry, heat of formation energy, energy of molecular orbital, and visualized molecular orbitals become available in the viewer window. Detailed tutorial and troubleshooting of using WebMO is also available in the website (http://www.webmo.net).

4. RESULTS AND DISCUSSIONS

4.1 Diels-Alder Reaction

Predicting regioselectivity and stereochemistry of asymmetric dienes-dienophiles is commonly performed by identifying electron withdrawing and donating groups followed by a drawing of the respective resonance structures (Figure 2a). In addition to this traditional approach, the molecular orbital calculation was conducted on the asymmetric Diels-Alder reactions. In this study, acrylaldehyde and 2-methoxy cyclohexa-1,3-diene were used as a dienophile and a diene, respectively. The frontier orbital approach is a good way to understand these [4+2] cycloaddition reactions. The highest occupied molecular orbital (HOMO) of 2-methoxy cyclohexa-1,3-diene and the lowest unoccupied molecular orbital (LUMO) of acrylaldehyde were identified by evaluating electron occupancy in each orbital. Matching orbitals with the largest lobes in the $HOMO_{diene}$ and $LUMO_{dienophile}$ can aid in visualizing the formation of more favorable transition state with a six-membered ring which, in turn, provides the major product. As shown in Figure 2b, the phases of the interacting orbitals are also color coded to help students to match lobes with the same sign and interact to form chemical bonds.

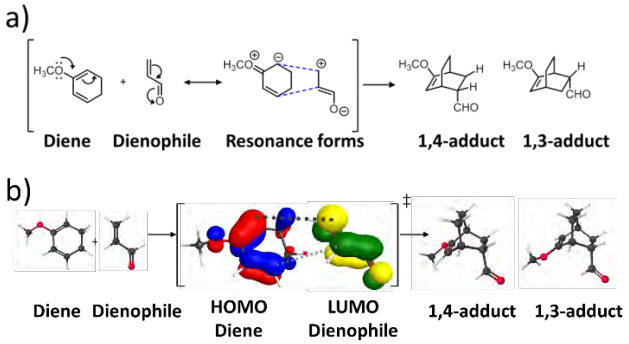

Figure 2. Reaction scheme between acrylaldehyde and 2-methoxy cyclohexa-1,3-diene, a) Regio-selective adduct prediction by charge-separated resonance structures, b) HOMO-LUMO interaction and three-dimensional view of both adducts calculated by MOPAC with PM3.

After deducing which orbitals were most likely to form a bond during the Diels-Alder reaction, both potential adducts were geometrically optimized. The enthalpy changes for the reactions were calculated to determine which adduct would be enthalpically favored (Table 1). The heat of reaction revealed that the 1,4-adduct is slightly favored by 0.3 kcal/mol which is consistent with experimental results [16].

Table 1. Enthalpy change for Diels-Alder reaction‡

Name	Diene	Dienophile	Adduct	ΔH_{rxn} (kcal/mol)*
1,4-adduct	-17.26	-18.31	-68.45	-32.89
1,3-adduct	-17.26	-18.31	-68.17	-32.61

‡Only enthalpic contribution was considered.

* Data is calculated by MOPAC with PM3.

Although the favorable formation of 1,4-adduct is largely governed by the transition state interaction as shown in the resonance structures of diene-dienophile (Figure 2a), students can readily evaluate the regioselectivity between two potential adducts through visually evaluating the HOMO-LUMO interaction. Using the molecular orbital diagrams in conjunction with the thermodynamic results allowed not only a visual evaluation but also a quantitative approach in determining a regioselective product.

4.2 Cycloaddition of Polycyclic Aromatic Hydrocarbons

Although the Diels-Alder reaction between anthracene and maleic anhydride is a well-known reaction, often rationalizing how to predict the reaction sites on fused-ring systems is challenging in undergraduate laboratory classes (Figure 3a). The regioselectivity of this reaction can best be understood by using aromaticity, so the reaction occurs at C9 and C10 site where the weakest benzenoid character exists. The structures with more fused rings tend to have lower resonance energies per π-electron compared to benzene. Because the structures with fewer rings are more stable, cycloaddition reaction of anthracene where three rings are fused together occurs at an internal ring over a terminal ring to give a more stable product. With computational aids, however, students can visually approach which carbon atoms in the diene are involved in the cycloaddition. On the HOMO of anthracene, the carbon C9 and C10 on the middle cycle possess the largest lobes indicating the most reactive site (Figure 3b). Connecting with the largest lobes designated in maleic anhydride gives the anticipated product, 9-10-dihydroanthracene-9,10-α-β succinic anhydride.

Another challenge for students to understand would be predicting the three-dimensional geometry of the final adduct. As shown in Figure 3c, the optimized geometry of the adduct was obtained by MOPAC, and clearly displays that the two hydrogen atoms in maleic anhydride retain cis-position. Additionally, the puckered structure of middle cycle indicates the change from sp^2 to sp^3 hybridization upon the formation of the adduct.

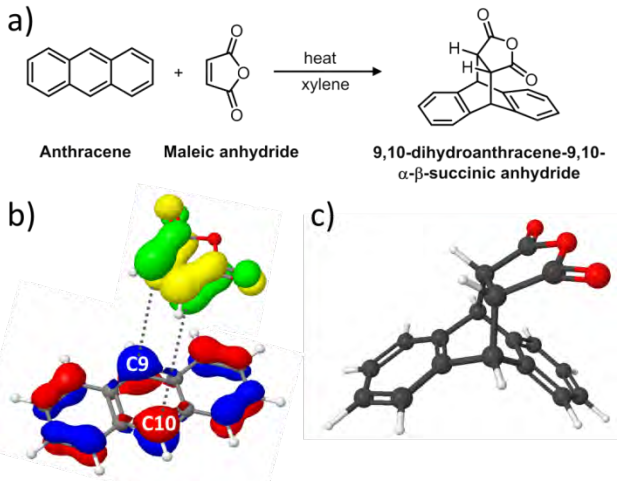

Figure 3. a) Diels-Alder Reaction between maleic anhydride (dienophile) and anthracene (diene), b) Schematic HOMO-LUMO interaction, c) Optimized geometry of the Diels-Alder adduct.

Generating an energy level diagram with calculated molecular orbital energies of the diene and dienophile facilitated further visualization of the HOMO-LUMO interaction. As taught in class, the strongest interaction should be between the $HOMO_{diene}$ and the $LUMO_{dienophile}$ because of the smaller energy gap compared to the other interaction between the $LUMO_{diene}$ and $HOMO_{dienophile}$. With a smaller HOMO-LUMO energy gap, the orbitals interact to a great extent in the transition state, which reduces the energy barrier between the reactants and the product. Consistent with the lesson from the lecture, the energy gap between $HOMO_{diene} - LUMO_{dienophile}$ (154.53 /mol) is unambiguously smaller than $HOMO_{dienophile} - LUMO_{diene}$ (247.65 kcal/mol), which points out another important concept of frontier orbital interaction upon the cyclization (Figure 4).

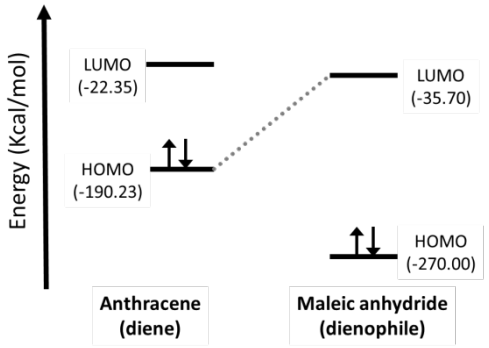

Figure 4. Frontier orbital energy diagram of maleic anhydride and anthracene.

Furthermore, the overall ΔH_{rxn} was calculated to be -32.9524 kcal/mol which is the energy difference between anthracene (61.5035 kcal/mol)/ maleic anhydride (-90.1541 kcal/mol) and adduct (-61.6030 kcal/mol). The heat of reaction between reactants and product through the calculations allowed students to predict that the product will be enthalpically favored in the cycloaddition prior to the hands-on experiment.

4.3 Assessment of Effectiveness of Computational Aids

As described in the Class design (Figure 1), the effectiveness of computational aids was assessed by comparison between pre-lab assignment and post-lab assignment, which is tabulated in Table 2. Overall the post-lab assignment results showed markedly improved scores compared to the results in all questions prior to the implementation of computational visualization.

Table 2. Comparison between Pre-lab and Post-lab assignment results (prior to and after implementation of computational aids).

Question	Percentage Correct Response	
	Pre-lab assignment	Post-lab assignment
Identify the Diene and Dienophile	98	100
Predict reactive site in Dienophile	88	89
Predict reactive site in Diene	27	86
Sketch HOMO-LUMO energy diagram	34	73
Draw Diels-Alder product (including stereochemistry)	23	71

Regardless of the computational analysis, most students correctly identified which components of a reaction were either diene or dienophile. Once diene and dienophile had been assigned, predicting the reaction sites of the dienophile was straightforward. It is presumably due to the fact that only one carbon-carbon double bond is present in maleic anhydride.

Finding the reactive carbons in the diene, however, becomes more challenging because of the highly conjugated structure of anthracene. The structure of anthracene gave mainly two choices of reaction sites, the middle cycle vs outer cycle. The majority of incorrect answers (73%) in pre-lab assignment were assigning the reactive site at the outer cycle of anthracene because they considered less steric hindrance for the cyclization at the outer cycle as shown in students' final reports.

In the post-lab assignment, after utilizing visual aids from MOPAC, the percentage of correct response was dramatically increased (27% to 86%). It is because modeling the HOMO of anthracene clearly provides more information of which atoms have the right phase and largest lobe to interact with the LUMO of maleic anhydride. Therefore, it is evident that computational aids facilitated the students in properly identifying the most reactive carbons of fused ring systems.

Understanding the interaction between HOMO and LUMO during a Diels-Alder reaction was another challenging concept which resulted in only 34% of students correctly drawing the energy diagrams in the pre-lab assignment. Many students had difficulty in understanding why the interaction between $HOMO_{diene}$-$LUMO_{dienophile}$ is favorable. It is presumably because generating a molecular orbital diagram was not a topic heavily emphasized in an undergraduate level class so that students are less familiar with molecular orbital or frontier orbital concepts. After using computational calculations which give molecular orbital energy levels in the numerical values (Figure 4) and visualized images of HOMO/LUMO with specific phases and sizes (Figure 3), students had a much better understanding of how the molecular orbitals were interacting and forming chemical bonds. This visual comprehension of molecular orbital theory transitioned into a more accurate molecular orbital diagram in the post-lab assignment.

Finally, only 23% of students were able to draw a correct chemical structure of the Diels-Alder product in the pre-lab assignment. There were mainly three categories of incorrect structures; i) wrong regioselectivity, i.e., bicycle formation with outer cycle of anthracene, ii) incorrect or no stereochemistry (cis- vs trans-hydrogen of maleic anhydride, iii) flat (unpuckered) structure of the fused bicycle. The majority of incorrect answers stemmed from the incorrectly assigned regioselectivity. It can be understood that students would not be able to draw the final product structure correctly once they identified incorrect reaction sites. However, it should be noted that besides the adduct formation with wrong regioselectivity, we found other two types of errors which are from the lack of three-dimensional understanding. In other words, students found a difficulty in assigning stereochemistry and structural transition from sp^2 to sp^3 hybridization. After obtaining the geometrically optimized adduct structure by molecular modeling, students had a better sense of a molecules geometric configuration in a three dimensional space, which translated into highly improved scores of the adduct stereochemistry in the post-lab assignment. Furthermore, such structural information was also confirmed by NMR spectroscopy after the synthesis of the Diels-Alder adduct. Therefore the results indicate that the implementation of computational analysis leverages students' understanding of many important concepts of physical organic chemistry.

5. CONCLUSION

In this study, we demonstrated that computational aids such as WebMO can be introduced into undergraduate level classes and facilitate students' better understanding of Diels-Alder reactions. Combinations of multiple concepts in chemistry often require quite complicated design of classes with multiple separated sections. The Diels-Alder reaction is a well-established example utilized in the undergraduate organic chemistry laboratory. By incorporating computational aids into this particular laboratory class, students can be exposed to several important concepts including thermodynamics, frontier orbital theory, stereochemistry, and so on. We also observed such visualization increased students' engagement in the course, which is presumably because they are more familiarized with today's technology. Harmonization of synthetic chemistry lab with computational analysis to demonstrate how physical chemistry can provide "visible" understanding is very important. It is, however, equally important for students to realize that computational results cannot solely be used to answer the questions. Further efforts to develop classes with highly efficient and effective teaching tools associated with computational analysis are ongoing in our department.

6. ACKNOWLEDGMENTS

We gratefully acknowledge the Shodor Education Foundation for the assistance of utilizing WebMO. And we specially acknowledge to the students who participated in this study at Penn State University at Worthington Scranton and Slippery Rock University.

7. REFERENCES

[1] Nassabeh, N.; Tran, M., and Fleming, P. E. 2014. Dissociation of the Ethyl Radical: An Exercise in Computational Chemistry. *J. Chem. Edu.* 91, 1248–1253. http://pubs.acs.org/doi/pdf/10.1021/ed4007748

[2] Perri, M. J. and Weber, S. H. 2014. Web-Based Job Submission Interface for the GAMESS Computational Chemistry Program. *J. Chem. Edu. 91,* 2206–2208. http://pubs.acs.org/doi/pdf/10.1021/ed5004228

[3] Albrecht, B. 2014. Computational Chemistry in the Undergraduate Laboratory: A Mechanistic Study of the Wittig Reaction. *J. Chem. Edu.* 91, 2182–2185. http://pubs.acs.org/doi/pdf/10.1021/ed400008d

[4] McNaught, I. J. 2011. Testing and Extending VSEPR with WebMO and MOPAC or GAMESS. *J. Chem. Edu.* 88, 421–425. http://pubs.acs.org/doi/pdf/10.1021/ed900038a

[5] Rowley, C. N.; Woo, T. K., and Mosey, N. J. 2009. A Computational Experiment of the Endo versus Exo Preference in a Diels–Alder Reaction. *J. Chem. Edu.* 86, 199–201. http://pubs.acs.org/doi/pdf/10.1021/ed086p199.

[6] Palmer, D. R. J. 2004. Integration of Computational and Preparative Techniques To Demonstrate Physical Organic Concepts in Synthetic Organic Chemistry: An Example Using Diels-Alder Reactions. J. *Chem. Edu.* 81, 1633–1635. http://pubs.acs.org/doi/pdf/10.1021/ed081p1633

[7] Diels, O., and Alder, K. 1928. Synthesen in der hydroaromatischen Reihe. *Justus Liebigs Ann. Chem*. 460, 98–122. http://onlinelibrary.wiley.com/doi/10.1002/jlac.19284600106/pdf

[8] Takao, K.-i.; Munakata, R., and Tadano, K.-i. 2005. Recent Advances in Natural Product Synthesis by Using Intramolecular Diels−Alder Reactions. *Chem. Rev.* 105, 4779–4807. http://pubs.acs.org/doi/abs/10.1021/cr040632u

[9] Celius, T. C. 2010. Fast Hetero-Diels−Alder Reactions Using 4-Phenyl-1,2,4-triazoline-3,5-dione (PTAD) as the Dienophile. *J. Chem. Edu.* 87, 1236–1237. http://pubs.acs.org/doi/pdf/10.1021/ed100344g

[10] Coleman, W. F. 2010. Diels−Alder Reactions and the Structure of Transition States. *J. Chem. Edu.* 87, 1278–1279. http://pubs.acs.org/doi/pdf/10.1021/ed1009078

[11] Weizman, H.; Nielsen, C.; Weizman, O. S., and Nemat-Nasser, S. 2011. Synthesis of a Self-Healing Polymer Based on Reversible Diels–Alder Reaction: An Advanced Undergraduate Laboratory at the Interface of Organic Chemistry and Materials Science. *J. Chem. Edu.* 88, 1137–1140. http://pubs.acs.org/doi/pdf/10.1021/ed101109f

[12] Amin, S.; Barnes, A.; Buckner, C.; Jones, J.; Monroe, M.; Nurmomade, L.; Pinto, T.; Starkey, S.; Agee, B. M.; Crouse, D. J., and Swartling, D. J. 2015. Diels–Alder Reaction Using a Solar Irradiation Heat Source Designed for Undergraduate Organic Chemistry Laboratories. *J. Chem. Edu.* 92, 767–770. http://pubs.acs.org/doi/pdf/10.1021/ed500850c

[13] Sendlinger, S. C.; Metz C. R. 2010. Computational Chemistry for Chemistry Educators. *J. Comput. Sci. Educ.* 1, 28–32. http://www.shodor.org/media/content//jocse/submissions/sendlinger2010/sendlinger2010_pdf

[14] More information of workshop, and detailed lectures and lab exercises can be found in the Computational Chemistry for Chemistry Educators website. http://www.computationalscience.org/ccce/

[15] Schmidt, J. R., and Polik, W. F. 2015 WebMO Version 16. *http://www.webmo.net* (Accessed 2013–2015).

[16] L. G. Wade, J.: *Organic Chemistry*; 8th Ed. ed.; Person Education, Inc., 2013.

Educational Module on Genomic Sequence Alignment Using HPC

Angela B. Shiflet
George W. Shiflet
Wofford College
Department of Computer Science
Department of Biology
Spartanburg, S. C. 29303 USA
+01 (864) 909-5396
shifletab@wofford.edu
shifletgw@wofford.edu

Daniel S. Couch
Wofford College Student
Blue Waters Intern
Spartanburg, S. C. 29303 USA
+01 (864) 597-4000
couchds@email.wofford.edu

Pietro Hiram Guzzi
Mario Cannataro
University "Magna Græcia" of Catanzaro
Department of Medical and Surgical Sciences
Catanzaro, Italy
+39 0961-369 4100
hguzzi@unicz.it
cannataro@unicz.it

ABSTRACT

"Aligning Sequences–Sequentially and Concurrently," an educational computational science module by the authors and available online, develops a sequential algorithm to determine the highest similarity score and the alignments that yield this score for two DNA sequences. Moreover, the module considers several approaches to parallelization and speedup. Besides a serial implementation in C, a parallel program in C/MPI is available. This paper describes the module and details experiences using the material in a bioinformatics course at University "Magna Græcia" of Catanzaro, Italy. Besides being appropriate for such a course, the module can provide a meaningful application for a high performance computing or a data structures class.

CCS Concepts

• Social and professional topics~Computing education • Theory of computation~Parallel algorithms • Theory of computation~Dynamic programming • Applied computing~Bioinformatics

Keywords

Computational Science; High-Performance Computing; Educational Modules; Blue Waters; Fulbright.

1. INTRODUCTION

In a Fulbright Specialist visit to University "Magna Græcia" of Catanzaro, Italy, in January 2015, Angela Shiflet and George Shiflet initiated a project with Mario Cannataro and Pietro Guzzi to develop educational modules on high-performance-computing bioinformatics algorithms. Wofford College student Daniel Couch, supported by a one-year internship with the Blue Waters Student Program, implemented the sequential and high-performance computing algorithms associated with the first two of the resulting modules.

DOI: https://doi.org/10.22369/issn.2153-4136/8/1/2

The first product of this collaboration is a module that uses algorithms to determine the highest similarity score and the alignments that yield this score for two DNA sequences. Sequence comparison to determine the similarity or difference of two sequences is a fundamental operation of the interdisciplinary field of bioinformatics. Bioinformatics, which relies on mathematics, statistics, and computer science, provides tools to compile, organize, and analyze the overwhelming volumes of data that are being generated from genomic studies. Because of the enormous quantity of data involved, high performance computing is an essential tool in bioinformatics. Similarities in gene sequences from different organisms, such as human and mouse, can help establish the function of the gene, and through sequence alignment, scientists can help establish the genetic causes of certain diseases. Moreover, sequence alignment is used to study protein functions and as a basis to predict protein structure.

The educational module, "Aligning Sequences–Sequentially and Concurrently," available at [6], develops the sequential Needleman-Wunsch Algorithm for sequence alignment and two parallel versions of the algorithm, the Pipeline Algorithm and the Block-and-Band Version of the Pipeline Algorithm. Also included in the module are the necessary biological background, quick review questions, exercises, and projects. The module was class tested in the course Advanced Techniques for Bioinformatics at University "Magna Græcia" in Spring, 2016, under the direction of Dr. Pietro Guzzi. This paper describes and examines the module and our experiences using it.

2. Module

2.1 Pedagogy

A variety of courses can incorporate the educational module "Aligning Sequences–Sequentially and Concurrently." For instance, a bioinformatics course can encompass the concepts and algorithms and consider or omit the programming components. The module can also be useful in an entry-level programming or data structures course to show an application of two-dimensional arrays. Moreover, a high-performance computing class can cover the module emphasizing the HPC algorithms and concepts.

The module provides the biological background necessary to understand the applications and references for further study. Eighteen (18) multi-part quick review questions throughout the module, with answers at the end of the module, provide immediate feedback. Nine (9) exercises give additional practice to

aid understanding of various aspects of the algorithms. The module also provides five (5) project assignments for further exploration using sequential and/or parallel programming. Instructors can obtain implementations of the sequential algorithm in C and the parallel algorithms in C with MPI from [5] or the authors.

2.2 Biological Background

The introduction begins with a story of a woman diagnosed with breast cancer, possibly caused by inherited, mutated genes, and a general discussion of genes. Subsequent background sections are on "Nucleic Acids," "Proteins," "Connecting DNA Code to Protein Sequence," "Mutations and Cancer," and "Genomics and Bioinformatics." The latter section indicates the importance of computation to biology by emphasizing that bioinformatics employs mathematics, statistics, and computer science to organize and store in databases vast amounts of data generated from genomic studies and to analyze that data.

Some of the biological material in the module will be familiar to some students but is included for students with minimal science backgrounds. Crucial to the understanding of the material is a basic understanding of deoxyribonucleic acid (DNA). DNA is a long chain of molecules, each containing a nitrogenous base, adenine (A), guanine (G), cytosine (C), or thymine (T).

Among the different bioinformatics algorithms, pairwise sequence alignment was chosen because it is largely used in various fields of biology, e.g. to highlight conserved DNA sequences or protein motifs along evolution, or as a basis of the protein structure prediction algorithms used to predict secondary and tertiary structure of proteins.

In particular, pairwise sequence alignment algorithms arrange the two input sequences (e.g. representing DNA, RNA, or proteins) to discover similar regions that may be due to functional, structural or evolutionary relations among the sequences. On the other hand

Moreover, sequence alignment algorithms use a very simple and intuitive metrics, i.e. the similarity among sequences, as a criterion to evaluate the quality of alignment. Finally, the chosen Needleman-Wunsch algorithm works on tabular data that is a data format very familiar to the students to whom the educational module is addressed.

2.3 Sequential Algorithm

Using bioinformatics, we can align DNA sequences to identify regions that are similar. Such a similarity might indicate that the two regions have the same function or evolve from a common ancestor in a sequence of mutations. In comparing two sequences, such as ATGAC and ACGC, we can employ a metric, called a similarity score, or score, to rate various alignments. For a scoring scheme, the highest possible similarity score indicates the best alignment(s). As the module discusses, an alignment of two DNA sequences has spaces in the sequences so that they are of the same length but so that a space in one sequence is not in the same position as a space in the other sequence. For example, with a dash (-) indicating a space, one alignment of s = ATGAC and t = ACGC follows:

s:	A	T	G	A	C
t:	A	-	C	G	C

Another possible alignment is as follows:

s:	A	T	G	A	C	-	-
t:	-	-	-	A	C	G	C

Although we can rate the quality of an alignment in many ways, this example in the module defines the score for an alignment as the total of column, or position, scores, where the column scores have the following values: +1 for a match, -1 for a mismatch, and -2 for a space in one of the corresponding positions. Adding all the position scores, the following alignment, with a dash (-) indicating a space, has a score of 1 + (-2) + (-1) + (-1) + 1 = -2:

s:	A	T	G	A	C
t:	A	-	C	G	C
column scores:	1	-2	-1	-1	1

The Needleman-Wunsch Algorithm is a technique to determine the similarity and the alignments that yield this score [4]. The algorithm employs dynamic programming, which divides a problem into a collection of smaller problems and uses the solutions to these smaller problems to solve the larger problem. The Needleman-Wunsch Algorithm makes the best decision for prefixes, or subsequences from the start of the sequences (the smaller problems), as it iterates over the length of those prefixes. The module used the notation $s[i..j]$ to indicate the subsequence from position i to position j, where the first position number is 0. For example, in s = ATGAC, $s[1..2]$ is TG.

We write the developing intermediate similarity scores in a two-dimensional array, or matrix, a. A blank (dash) and the bases of one sequence, such as s, are row labels, while a blank and the bases of the other sequence, such as t, label the columns. As indicated in Figure 1, in row 0 and column 0, we write the on-going scores for matching all spaces with prefixes of sequence s and t, respectively.

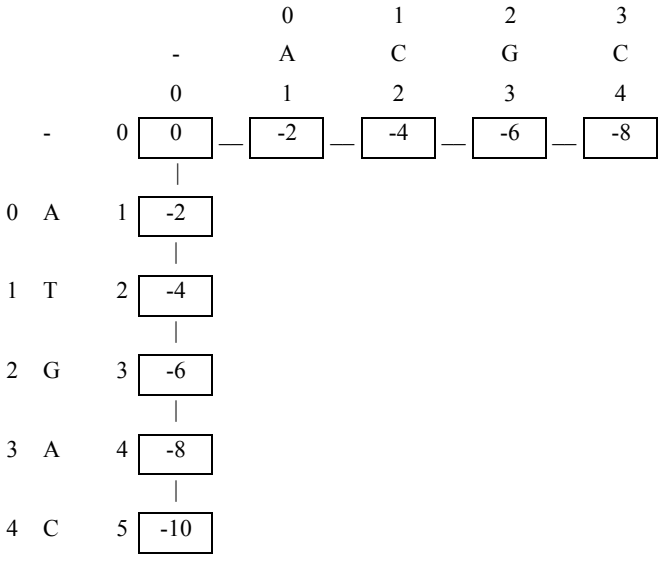

Figure 1. Initial values in similarity matrix

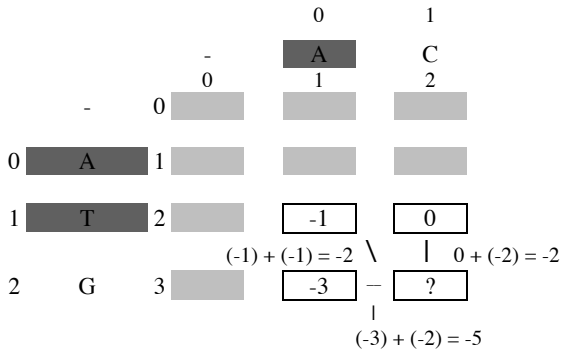

Figure 2. Determine *a*[3][2] from *a*[2][2], *a*[3][1], and *a*[2][1]

To determine the matrix scoring values, we proceed row by row, from left to right, calculating elements. Figure 3 contains the entire similarity matrix, with line segments marking the paths from the maximum element(s). The value in the bottom, right corner, 0, is the similarity of ATGAC and ACGC. Following line segments from that corner backward to *a*[0][0], we obtain a corresponding alignment for the sequences, such as the following optimal alignment:

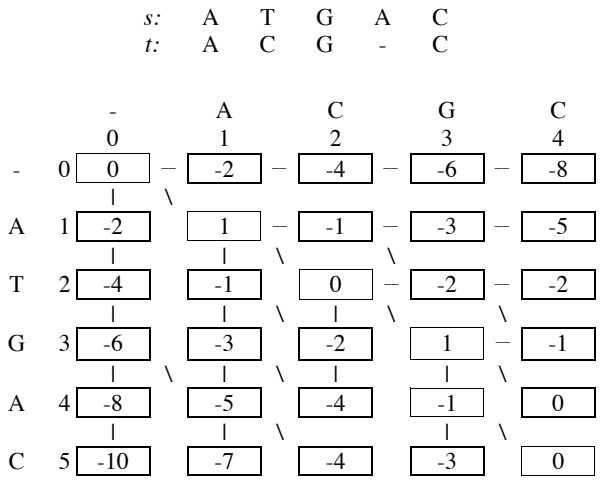

Figure 3. Array of similarity values for ATGAC and ACGC

2.4 HPC in Module

After covering this material, the module shows that employing a two-dimensional array, the complexity of the sequential algorithm is on the order of n^2, $O(n^2)$, where n is the length of a sequence. To illustrate the problem, the module displays a graph of timings using a C implementation of the Needleman-Wunsch Algorithm with an increasing number of nucleotides (Figure 4). When we are trying to match a sequence to multiple sequences in a database, the timing challenge of employing an $O(n^2)$ algorithm so often becomes evident.

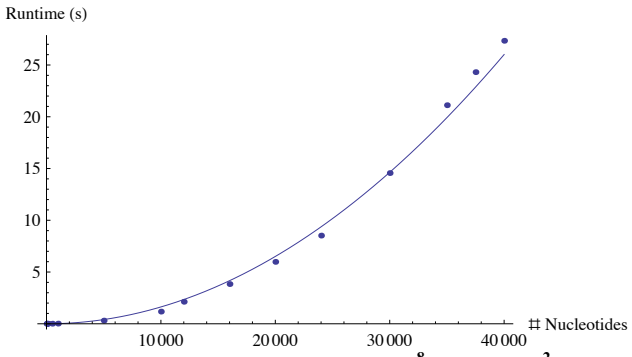

Figure 4 *runtime* = 1.62585 × 10^{-8} *nucleotides*^{2}

Thus, discussion of complexity motivates the need for parallel processing. We can have different processes aligning the search sequence to different database sequences in a way called "embarrassingly parallel," and/or we can have a parallel alignment algorithm, such as the Pipeline Algorithm, to operate on each pair of sequences.

In the Pipeline Algorithm, for simplicity, the module assumes that the number of processes equals the number of rows, n, in the similarity matrix and that Process j is responsible for making the calculations on row j. The processes can simultaneously compute their first column elements without communication with the other processes using the formula j * *spacePenalty* for $a[0][j]$. Then, Process j, for $j = 0, 1, \ldots, n-2$, can send $a[0][j]$ to Process $(j + 1)$. Similarly, Process 0 can compute the ith element in row 0 as i * *spacePenalty*. Immediately after calculation of $a[i][0]$, Process 0 can communicate the value to Process 1. Now, knowing the crucial values on the row above, $a[0][0]$ and $a[0][1]$, and the value to the left, $a[1][0]$, Process 1 can calculate $a[1][1]$. Moreover, while this calculation is occurring, Process 0 can be calculating its next value, $a[0][2]$. Then, Process 0 sends $a[0][2]$ to Process 1, and Process 1 sends $a[1][1]$ to Process 2, so that enough information will be available to occupy the first three processes. With each step, an additional process is drafted to work. Figure 5 depicts the progress of this pipelining system for sequences of length $n = 5$ and $m = 8$. After receiving communication of the value above, a process can start computation of its element in darker outline. Thus, calculation of values on this anti-diagonal can proceed in parallel [2].

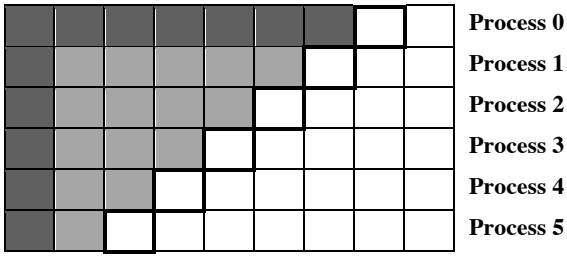

Figure 5 Pipelining the similarity matrix for sequences of length $n = 5$ and $m = 8$

After discussing the algorithm, the module considers the number of steps for n processes to calculate the similarity matrix for sequences of length n, $O(n)$. However, as the module illustrates, one disadvantage is the amount of communication, which is $O(n)$, too. Besides a theoretical discussion in the module, a table

presents runtime and speedup results versus number of nucleotides for sequential and pipeline C implementations of the Needleman-Wunsch Algorithm. The table shows that with more than 10,000 nucleotides the pipeline algorithm is faster than the sequential one, but speedup is not linear. Increasing communication between processes dampens the runtime of the HPC version.

The problem of communication motivates consideration of the block-and-band version of the Pipeline Algorithm. To reduce communication, each process calculates a block of several column values. Moreover, as Figure 6 illustrates, we can make a process responsible for a band, or several rows, of elements. After calculating a submatrix, a process sends the block of elements in the last submatrix row to the next process so that the latter can start evaluation of a submatrix. Not only does a process transmit fewer elements, communication can involve a block of elements instead of multiple separate *send* operations, which is slower. One of the module's projects has the students implementing the algorithm and determining optimal block and band sizes. Consideration of the results can lead to a class discussion of scaling and load balancing.

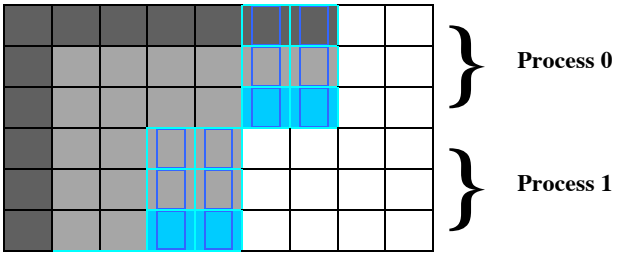

Figure 6 Pipelining a similarity matrix with block size of 2 and band size of 3

2.5 Reinforcement of Material

Eighteen, often multipart, Quick Review Questions throughout the module provide an assessment of the student's comprehension of the material. For example, one question has the students calculating the value of a scoring matrix element by hand. Another 17-part question has the student trace through the scoring algorithm using particular sequences. Answers at the end of the module provide immediate feedback to determine if the student is understanding the concepts.

Nine exercises provide additional reinforcement. For example, three exercises ask the students to develop entire scoring matrices for particular sequences, two involve complexity, and two consider modifications to the sequential algorithm.

Five projects have the students developing various sequential and/or parallel algorithms, performing timings, calculating speedup, and determining advantageous band and block sizes.

3. Class Testing

3.1 Class

In Spring, 2016, the course Advanced Techniques for Bioinformatics at the University "Magna Græcia" of Catanzaro in Italy covered the module. The course is a requirement during the last semester of their Master's Degree in Biomedical Engineering. The thirty (30) students in the class had BS degrees in biomedical engineering and, thus, had more skills on the biological side than in computer science. The students covered the module in one week with four class contact hours. A final assignment, done individually, was to implement the sequential version in Python and to compare their times to those in the module for the sequential and parallel implementations.

3.2 Results

After coverage of the material, seventeen (17) students completed a survey about the module. Table 1 gives the list of survey questions, eliciting a response from 1 for "strongly disagree" to 5 for "strongly agree," with the average scores. The responses were mostly very favorable but indicated some challenges with the parallel algorithms and programming.

Score	Statement
4.47	I understood the science in the module.
4.29	I understood the sequential algorithms in the module.
3.76	I understood the parallel algorithms in the module.
4.53	I understood the importance of using high performance computing.
4.47	The module was readable.
4.59	The Quick Review Questions helped me understand the material.
4.25	The exercises helped me understand the material.
3.13	The project helped me understand the material.

Table 1 Student survey averages (1 – strongly agree and 5 – strongly agree)

The survey also included the following questions that required free responses:

- Please elaborate about the above scores, particularly those below 4.
- What did you like best about the module?
- What did you find most difficult in the module?
- Please give corrections and suggestions for improvement.
- Please make further comments.

These responses indicated a desire by many students for the module to have more examples and explanation of the parallel algorithms. In response, the authors revised the sections on "Pipeline Algorithm" and "Block-and-Band Version of the Pipeline Algorithm" to include three specific, detailed examples and three additional multipart quick review questions with answers on the parallel versions (pipeline and block-and-band with bands of size 1 and greater than 1) of the algorithm.

The students' free responses contained numerous compliments. Students indicated that they liked the linkage between genomics and bioinformatics, the correlation between genetic mutations and cancer, and the discussion of bioinformatics. Several stated that the quick review questions, figures, tables, and pseudocode for the Needleman-Wunsch Algorithm were helpful. One student commented, "The way in which alignment algorithms were explained in the module is better than other articles I read, I really like it," and another stated, "The article was interesting, from which we learned new concepts." About one-third of the students volunteered that the module was "interesting" or "very interesting."

4. CONCLUSION

Based on the survey responses and the students' performance, the module, "Aligning Sequences–Sequentially and Concurrently," accomplishes conveying some of the basic principles of bioinformatics, the Needleman-Wunsch Algorithm, HPC versions of the algorithm (pipeline and block-and-band), and the utility of high performance computing. Moreover, the students understood and appreciated the material and successfully completed the assignment. As a result of their suggestions, the authors improved the module, making it an even better educational module incorporating HPC topics in the context of other applications learning.

5. ACKNOWLEDGMENTS

Our thanks go to the Fulbright Specialist Program, University "Magna Græcia" of Catanzaro, and Wofford College for funding the Shiflets' visit to the university and to the National Computational Science Institute Blue Waters Student Internship Program for funding Daniel Couch's internship.

6. REFERENCES

[1] Cannataro, M., and Guzzi, P. 2011. *Data Management of Protein Interaction Networks*, Wiley.

[2] Chen, Y., Yu, S., and Le, M.. 2006. "Parallel Sequence Alignment Algorithm for Clustering System" in *International Federation for Information Processing (IFIP)*, Volume 207, *Knowledge Enterprise: Intelligent Strategies In Product Design, Manufacturing, and Management*, eds. K. Wang, Kovacs G., Wozny M., Fang M., (Boston: Springer), pp. 311-321.

[3] National Computational Science Institute Blue Waters Student Internship Program. 2016. `http://computationalscience.org/bwsip/` Accessed April 6, 2016.

[4] Needleman, S. and Wunsch, C. 1970. "A General Method Applicable to the Search for Similarities in the Amino Acid Sequence of Two Proteins." *J. Molecular Biology*, vol. 48, pp. 443-453.

[5] Website associated with Shiflet, A. and Shiflet, G. 2014. *Introduction to Computational Science: Modeling and Simulation for the Sciences, 2nd ed.*, Princeton University Press `https://ics.wofford-ecs.org/` Accessed April 6, 2016.

[6] Shiflet, A., Shiflet, G., Couch, D., Guzzi, P., and Cannataro, M. 2016. "Aligning Sequences–Sequentially and Concurrently" `https://wofford-ecs.org/` Accessed April 6, 2016.

VisMo: Augmented Reality Visualization of Scientific Data and Molecular Structures

Max Collins
BWSIP Intern
University of Illinois at Urbana-Champaign (Undergraduate)
847-840-1020
macolli2@illinois.edu
truecollins@gmail.com

Dr. Alan B. Craig
Research Scientist
NCSA
University of Illinois at Urbana-Champaign
a-craig@illinois.edu

ABSTRACT
In this paper, we describe and detail our project that allows for augmented reality visualization of data produced using the Blue Waters supercomputer or other high performance computers. While molecular structures have been displayed using augmented reality before [1][6], we created a pipeline for using information from the Protein Data Bank and automatically loading it into an augmented reality scene for further display and interaction. We find it important to create an easy way for students, scientists, and anyone else to be able to visualize molecular structures using Augmented Reality because it offers an interactive three dimensional perspective that is typically not available in the classroom. Learning about molecular structures in 2D is much less comprehensive, and our technique for visualization will be free for the end user and offer a great deal of aid to the learning and teaching process. There is no separate purchase required as long as a user has a smart phone or tablet. This is a helpful addition to scientific papers which, if containing the right target image, can be used as the visualization "anchor." The Protein Data Bank (PDB) houses information about proteins, nucleic acids, and more to help scientists and students understand concepts and ideas in biology and chemistry [5]. Our project goal is to open the PDB up to students and people who are not familiar with augmented reality visualization and allow people to learn using the PDB by visualizing molecular structures in different representations, annotating and interacting with the structures, and offering learning modules for common molecular structures. We created a prototype mobile application allowing for molecular visualization of PDB structures, and are continuing to tweak our project for an eventual release to the public.

Keywords
Augmented Reality, Virtual Reality, Blue Waters, Unity 3D, PDB, Molecule, Visualization, Science

DOI: https://doi.org/10.22369/issn.2153-4136/8/1/3

1. INTRODUCTION
"Augmented Reality (AR) is a medium in which digital information is overlaid on the physical world that is in both physical and temporal registration with the physical world and that is interactive in real time," [2]. This is different than virtual reality, which allows us to enter an entirely digital world where our environment is generated by a computer. Our project focus is to allow users to, via a mobile application, display and interact with molecular structures using augmented reality. To do this, we used a game engine, in our case, Unity 3D, so that we did not have to create and develop capabilities that a game engine already has to offer. The game engine allows us to arrange digital information that will be overlaid onto the physical world and thus allow for augmented reality. We also have access to the Blue Waters supercomputer which made it efficient and quick to put structures in the PDB into a format that we could visualize and manipulate. The middle step from PDB to AR visualization is a program called VMD (Visual Molecular Dynamics) which takes structures from the PDB, visualizes them, and creates output in a format that we can make into an Augmented reality scene (a format readable by the game engine software). The end goal is to be able to open the application, which communicates with a server and requests a visualization in the desired format, have the server execute VMD to create a file with the correct specifications and send it back to the mobile application where it is displayed and can be interacted with via AR.

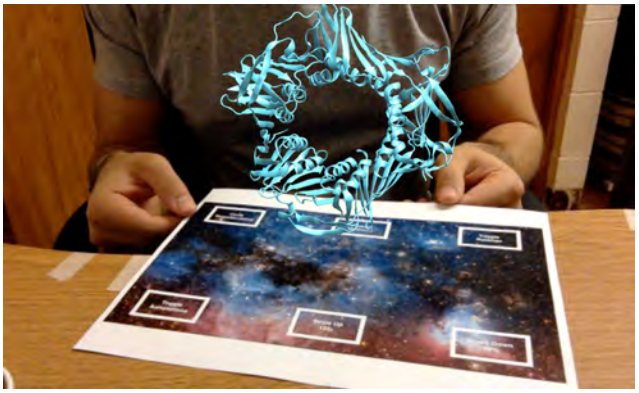

Figure 1: The view as seen from the camera of a computer that is using the VisMo application. You can see the target image, virtual buttons, and molecular structure.

2. Related Work

Work by Billinghurst, Poupyrev, and May about mixed reality environments and how augmented reality allows for collaborative computing is important and relates to concepts we are addressing with in our project [1].

A similar project at the HITLab at the University of Washington for augmented tangible molecular models created a molecular viewer where using virtual models highlighted primary, secondary, tertiary, and quaternary levels of structural organization and amino acid sequences [4].

Another related project is the NCSA Access magazine [5] created by the National Center for Supercomputing Applications (NCSA) at the University of Illinois at Urbana-Champaign. This issue of the magazine was a special augmented reality issue done as collaboration between Dr. Alan Craig, NCSA, and a team at Daqri, a company focused on augmented reality. The pages in the issue had unique content connected with them, and when users view the magazine through the mobile Daqri application, the augmented reality visualizations, including some example molecular visualizations, become visible.

3. Creating the Pipeline

3.1. From PDB to AR

In order to visualize structures from the PDB, we used Visual Molecular Dynamics (VMD) software. VMD retrieves the data for the requested molecule from the PDB and creates a 3D computer graphics representation of the desired visualization. In our case, VMD creates a .obj file which is then returned to the VisMo application and placed into our virtual scene which is overlaid on the real world via augmented reality. The user of VisMo requests a PDB file within the application by entering the desired molecule via a dialog box on the mobile application, VisMo passes that information along to the server, and VMD runs and produces the desired file with the user-selected specifications. VisMo then can place the visualization in the scene and the user can see and interact it. The user can now control the point of view, size, and other characteristics of the structure they are observing using on screen and virtual buttons.

3.2. Communicating With a Server

It was necessary to set up a server for this project because there is computing that must be done outside of the mobile application. Shodor was very helpful and set up a server with VMD on a virtual machine so that we could, from within our augmented reality application, request that a file be made using VMD and certain specifications, and then export the file back to users.

Storing all of the information from the PDB is not feasibly possible within a mobile application, which is why using a server is necessary for this project.

3.3. Live Annotations

Creating a system for live annotations within the scene during a visualization session was a goal of ours from the beginning; a way for the viewer to mark up the scene in order to help their comprehension and maybe help them teach a concept to other viewers. For now, we have implemented a “graffiti style” method of annotating the scene in real time. Essentially, when users press a button on the screen, a wall pops up behind the structure and allows the user to draw and write on it, acting as a worksheet of sorts.. In order to do this, we had help from Rodrigo Fernandez who allowed us to work using ideas and modified code from his project Texture Painter which can be found in the Unity 3D asset store [3]. Figure 2 shows a method of graffiti style annotation which could be implemented into the scene.

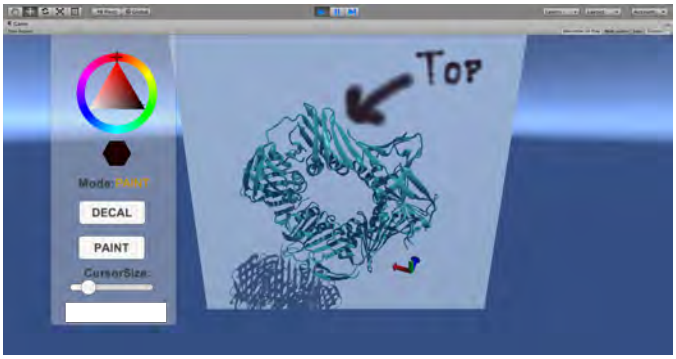

Figure 2: structure of human PCNA with a palette for graffiti style annotations behind it in Unity 3D.

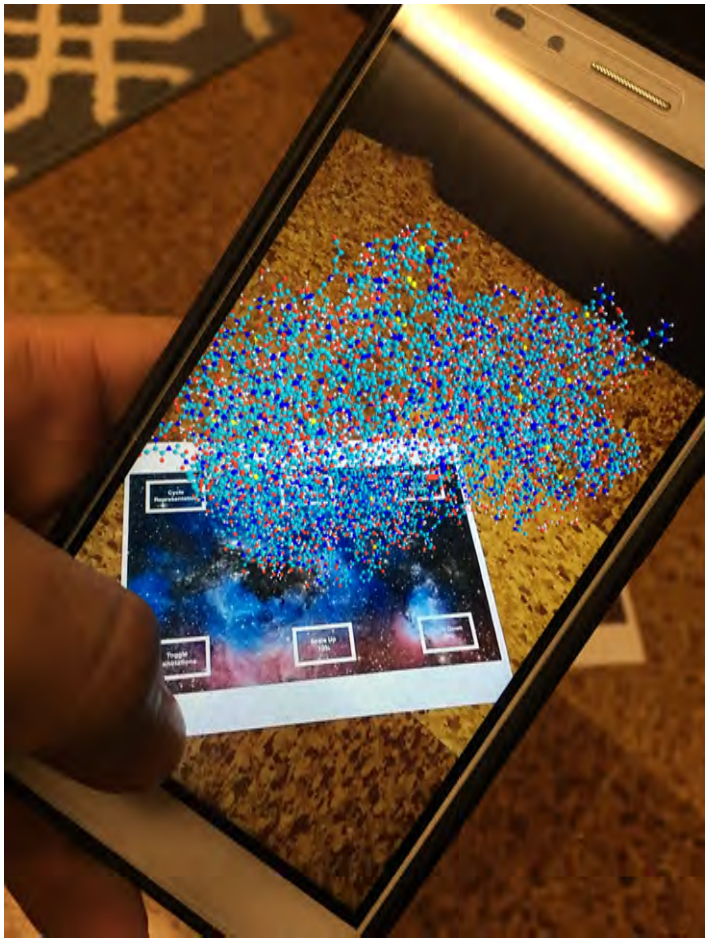

Figure 3: User has the application open in order to visualize a human PCNA structure using the ball and stick visualization technique.

4. Application

The applications for a project like this are vast and relatively unlimited; when we have the ability to control what digital content we are placing in physical and temporal registration with our real world, there is a great deal of versatility.

What we have created is a mobile application that we see as helpful for classroom use, lab use, and other scholarly settings. Our motivation for creating such an application was noticing how limited textbooks and worksheets are when teaching chemistry and biology. Learning about three dimensional structures by looking at them in two dimensional representation simply is not good enough. What we are doing is giving users the ability to manipulate and visualize structures from any angle with no more than a sheet of paper and their smartphone or tablet. It can be seen in Figure 2 that users working with this application will see through the lens of a camera on a tablet or smartphone as their looking glass into the augmented reality world that we have created.

In a lab, scientists often must communicate information with one another, and it is senseless to assume that each and every person they are working with understands exactly what they mean when trying to communicate certain information about certain molecular structures. With our application, scientists can more easily demonstrate concepts by being given the ability to show what they are working with more precisely, and being able to choose representation style, coloring style, etc. to better highlight certain structures or sections of a structure.

Finally, we can say with confidence that readers of scientific articles and papers are often bombarded with heavy text as the only means of information with intermittent pictures scattered throughout the page. Our application would allow users to insert pictures into scientific papers that could be used to display information in augmented reality without compromising the written content of the paper. This would offer a more rounded perspective to information given in a certain paper or article.

5. Conclusion

We were able to create an innovative application for Android and iOS platforms which allows users to visualize structures of molecules which are typically only seen in 2D representations on paper in a 3D space. Our project allows for students and scientists to interact with their work in a way that can foster a greater understanding and sense of comprehension because it is, by nature, more wholesome to our senses. Seeing things from multiple perspectives and multiple representation styles while not being limited to how many physical balls or sticks a classroom has to build a structure is endlessly helpful.

6. Future Work

We are continuing to work on this project, and our goal is to familiarize people with augmented reality and allow people to simultaneously enjoy elements of the physical and virtual worlds without shutting out exposure to one or the other at any given time. Viewing digital content and seeing the world around should not be mutually exclusive. We would like to continue to add learning modules into the application which can help people learn scientific concepts and complete tutorials by visualizing scientifically curated structures and reactions using our applications and augmented reality. We will also continue to add features such as expanding our pool of curated structures and working with scientists and teachers to assess what needs we can work to fulfill in the scientific community. We would like to create tutorials where people can learn through pre-designed lessons in addition to the freeform use of visualizing any PDB structure.

7. Reflections

The Blue Waters Student Internship Program created an environment where students learned about high performance computing, parallelization, and in general doing research at university level. Students were given the opportunity to sit in on seminars and educational sessions for learning about parallel computing and how to implement ideas from this type of computing into the research and problems that we are working on.

This was a great experience that has molded the education and career path I am taking. As a result of this program, I have already begun taking more computer science courses and visualization courses at University of Illinois at Urbana-Champaign. This has helped me learn more about the basics of computing, and I build upon that more in my independent classes, and I can also tie in the information I have learned from this internship to do what I need to do. This internship has also inspired me to get involved with other departments on campus. I have presented some of my work from this internship at various talks, and as a result of that I have been asked to help with projects and lab studies around campus. I am currently participating in two projects where I am using programming and visualization skills enhanced by this internship.

I plan to do more research at the graduate level as well. I plan to get my PhD in Informatics or Computer Science, so this internship has definitely had a great influence on me. I now understand how important research is, and I do think that I can make an impact on the scientific community throughout my years of graduate school as I continue to learn and grow more, and participate in more great programs like the Blue Waters Student Internship Program.

8. ACKNOWLEDGMENTS

Our thanks to John Stone for helping us with VMD.

Thanks to Mark Van Moer for helping us find the right visualization software and helping and meeting with us as needed.

Thanks to Rodrigo Fernandez for his help with creating a painting unity asset and helping create a custom version for our scene [3].

Thanks to the Blue Waters Student Internship Program for the help and support whenever it was needed. This program gives students a chance to participate in important research and the staff is without question passionate, knowledgable, and driven to allow students in this program to be successful.

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications [7][8][9].

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

Thanks to NASA for supplying the image used as our target image in our figures [10]. This image “Hubble Peers into the Storm” shows a glimpse into the large Magellanic Cloud.

A great thanks to Aaron Weeden for helping with cgi scripting and setting up a server with SHODOR.

9. REFERENCES

1. Billinghurst, M., Poupyrev, I., Kato, H., & May, R. (n.d.). Mixing realities in Shared Space: An Augmented Reality Interface for Collaborative Computing. 2000 IEEE International Conference on Multimedia and Expo. ICME2000. Proceedings. Latest Advances in the Fast Changing World of Multimedia (Cat. No.00TH8532). doi: 10.1109/icme.2000.871085
2. Craig, A. B. (2013). Understanding Augmented Reality: Concepts and Applications. Morgan Kaufmann.
3. Fernandez, R. (2015). Texture Painting. Retrieved June 22, 2016, from http://codeartist.mx/tutorials/dynamic-texture-painting/
4. HITLab Projects : Augmented Tangible Molecular Models. (n.d.). Retrieved June 23, 2016, from http://www.hitl.washington.edu/projects/scripps/
5. RCSB Protein Data Bank - RCSB PDB. (n.d.). Retrieved July 09, 2016, from http://www.rcsb.org/pdb/home/home.do
6. University of Illinois: NCSA Access, 25(3), 1-32. (2012, Fall).
7. Brett Bode, Michelle Butler, Thom Dunning, William Gropp, Torsten Hoe- fler, Wen-mei Hwu, and William Kramer (alphabetical). The Blue Waters Super-System for Super-Science. Contemporary HPC Architectures, Jeffery Vetter editor. Sitka Publications, November 2012.Edited by Jeffrey S . Vetter, Chapman and Hall/CRC 2013, Print ISBN: 978-1-4665-6834-1, eBook ISBN: 978-1-4665-6835-8.
8. Kramer, William, Michelle Butler, Gregory Bauer, Kalyana Chadalavada, Celso Mendes, Blue Waters Parallel I/O Storage Sub-system, High Performance Parallel I/O, Prabhat and Quincey Koziol editors, CRC Publications, Taylor and Francis Group, Boca Raton FL, 2015, Hardback Print ISBN 13:978-1-4665-8234-7.
9. John Towns, Timothy Cockerill, Maytal Dahan, Ian Foster, Kelly Gaither, Andrew Grimshaw, Victor Hazlewood, Scott Lathrop, Dave Lifka, Gregory D. Peterson, Ralph Roskies, J. Ray Scott, Nancy Wilkins-Diehr, "XSEDE: Accelerating Scientific Discovery", Computing in Science & Engineering, vol.16, no. 5, pp. 62-74, Sept.-Oct. 2014, doi:10.1109/MCSE.2014.80.
10. Hubble Peers into the Storm. (2016, September 9). Retrieved January 16, 2017, from https://www.nasa.gov/image-feature/goddard/2016/hubble-peers-into-the-storm

GPU-Accelerated VLSI Routing using Group Steiner Trees

Venkata Suhas Maringanti
Department of Computer Science
Trinity College Hartford, CT 06106
venkatasuhas.maringanti@trincoll.edu

Basileal Imana
Department of Computer Science
Trinity College Hartford, CT 06106
basileal.imana@trincoll.edu

Peter Yoon
Department of Computer Science
Trinity College Hartford, CT 06106
peter.yoon@trincoll.edu

ABSTRACT
The problem of interconnecting nets with multi-port terminals in VLSI circuits is a direct generalization of the Group Steiner Problem (GSP). The GSP is a combinatorial optimization problem which arises in the routing phase of VLSI circuit design. This problem has been intractable, making it impractical to be used in real-world VLSI applications. This paper presents our work on designing and implementing a parallel approximation algorithm for the GSP based off an existing heuristic on a distributed architecture. Our implementation uses the CUDA-aware MPI approach to compute the approximate minimum-cost Group Steiner tree for several industry-standard VLSI graphs. Our implementation achieves up to 103x speedup compared to the best known serial work for the same graph. We present the speedup results for graphs up to 3k vertices. We also investigate some performance bottleneck issues by analyzing and interpreting the program performance data.

1. INTRODUCTION
A number of optimization problems with different application areas can be modeled by the GSP: given an undirected weighted graph $G = (V, E)$ and a family $N = \{N_1,...,N_k\}$ of k disjoint groups of nodes $N_i \subseteq V$, find a minimum-cost tree which contains at least one node from each group N_i. One of such problems is the global routing phase in VLSI design. The exponential increase in complexity of integrated circuits where tens and thousands of non-overlapping nets may need to be routed simultaneously makes VLSI design a broad area where combinatorial optimization methods can be applied. The problem of interconnecting a net with multi-port terminals is a direct generalization of the GSP.

The advent of modern petascale supercomputing architectures has enabled scientists and engineers to solve several complex problems. Today's supercomputers can not only perform calculations with blazing speed, but also process vast amounts of data in parallel by distributing computing tasks to thousands of processing elements. With portable Application Programming Interfaces (APIs) such as MPI (Message Passing Interface) and CUDA (Compute Unified Device Architecture), researchers can now exploit parallelism to not only solve bigger problems, but also solve more problems in shorter time. This paper presents our work on the design and implementation of a parallel approximation algorithm for the GSP that uses Depth Bounded Steiner Tree Approximation [1]. Our goal was to achieve a better run time to make the heuristic practical for very large scale problems.

DOI: https://doi.org/10.22369/issn.2153-4136/8/1/4

2. A GPU-BASED ALGORITHM
Given an instance of the GSP, our parallel implementation returns a minimum-cost group Steiner tree. Our parallel algorithm follows the following steps in order.

2.1 Metric Closure on GPU
In general, the given graph G may violate the triangle inequality, i.e., there may be edges in G whose cost is greater than the cost of the minimum u-v path in G. An optimal group Steiner tree will contain no such edges, since replacing such edges with the corresponding shortest paths will decrease the total tree cost. Therefore, without loss of generality, we replace G by its Metric Closure. The Metric Closure is defined as the complete graph where the cost of each edge (u, v) is equal to the cost of the minimum u-v path in G. In other words, our first task is to compute the All Pair Shortest Paths (APSP) for the given graph and replace every edge cost with the corresponding minimum u-v path cost. For the APSP, we use a highly efficient CUDA implementation for the Blocked Floyd-Warshall APSP algorithm from [2] on a GPU. After computing the metric closure and replacing the original graph with it, we modify G as follows. We duplicate and replace every port with a new node and add a zero-cost edge between the two. The original port is now a non-port and the newly added node is now a port. An optimal tree in the modified graph G' has the same cost as an optimal tree in the original graph. Hence, this transformation allows us to seek a near-optimal Steiner tree in the original graph.

2.2 Group Steiner Heuristic on CPU
Using G' from the previous step, we now construct a minimum-cost Group Steiner tree by launching multiple processes using CUDA-aware MPI. A *d-star* is defined to be a rooted tree of depth at most d. With every vertex as the potential root r, we follow the following steps in order to construct the final solution.

2.2.1 1-Star
We construct 1-star tree rooted at the root of the optimal solution tree, i.e. a tree of depth 1 where all leaves are ports, one from each group.

2.2.2 Minimum-Norm Partial Star
We then select the intermediate nodes and determine a set of groups that should be connected to each intermediate node. A root, an intermediate node and a set of groups together form a

Partial-Star. Each partial-star is a sub-tree of the solution tree. We compute all such partial stars until all the groups are spanned.

2.2.3 2-Star

We combine all the partial-stars computed in the previous step to form the 2-star solution tree for the given vertex.

2.2.4 Minimum-cost 2-Star

We then collect all such 2-star solution trees obtained from the previous step and select the one with the minimum cost as the final solution. This is the minimum cost Steiner tree that the algorithm is supposed to output.

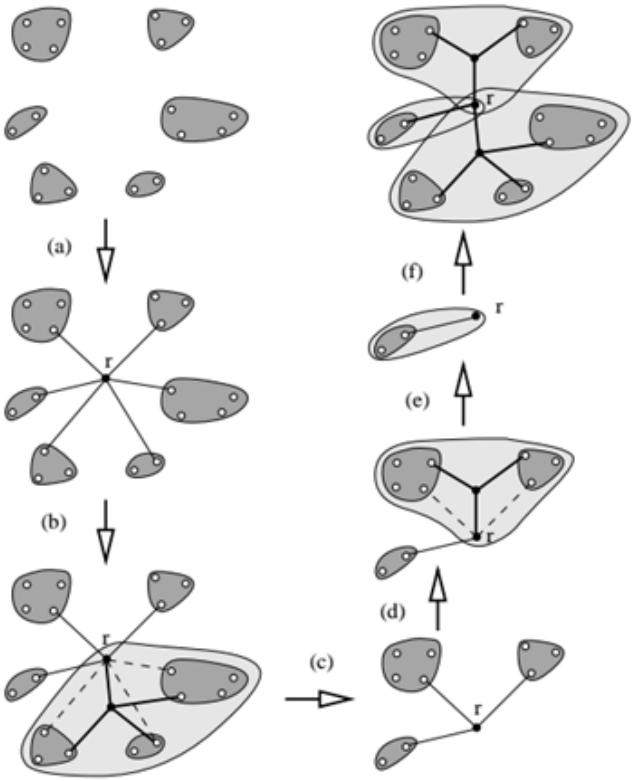

Figure 1. Steps (a) through (f) of the GSP Heuristic [1]

As shown in Figure 1, (a) constructs a rooted *1-star*, i.e. a tree of depth 1 where all leaves are ports, one from each group. A root, an intermediate node and a set of groups together form a *partial-star*. Each minimum-*norm* partial-star is a sub-tree of the solution tree [1]. Steps (c) though (e) compute such partial-stars until all groups are spanned. Step (f) combines all the partial-stars computed in the previous step to form a 2-star tree for the given root *r*. Out of all such 2-star trees obtained from the previous step, the one with the minimum cost is the final solution.

2.3 Work Distribution

Our Hybrid CPU-GPU approach uses CUDA-aware MPI as the standard for launching multiple processes on the Blue Waters supercomputer. In distributing work, the popular master-slave approach was used, wherein process 0 is the "master" process and the remaining ones are "slave" processes. The master performs step 2.1 and then broadcasts the modified graph to all the slaves. Each vertex (a potential root) is then mapped to a slave, whose task is to perform step 2.2 and communicate the result back to the master. After receiving all such results, the master then performs reduction to compute the overall solution.

2.4 NP-Hardness of GSP

The Group Steiner Problem (GSP) is a direct generalization of Classical Steiner Tree Problem, and has been known to be NP-hard. Hence it is not known whether an optimal solution to the GSP can be found by using a polynomial-time algorithm. The GPU-based algorithm as described above is a polynomial-time approximation scheme that efficiently outputs a near-optimal Group Steiner Tree.

The *performance ratio* is defined as the ratio of the approximate cost to the optimal cost for a given instance of the GSP. The Group Steiner Heuristic as described above returns a solution with a performance ratio no more than $2.\left(2+\ln(\frac{k}{2})\right).\sqrt{k}$ where k is the number of groups in the given instance of GSP [1].

The results in Table 1 show the comparison between the best known upper bound of the optimal cost (Opt. Cost) and the approximate cost(Approx. cost) returned by our GPU-based approach. Our results show that the GPU-based approach returns a nearly optimal solution with negligible cost error and a performance ratio within the given upper bound.

3. PERFORMANCE EVALUATION

We ran our performance tests on Blue Waters supercomputer which uses a Cray XE6/XK7 system. We tested both our serial and parallel implementations using several Wire Routing Problem (WRP) instances from industry. The instances are in a widely accepted standard STP format [3]. We compare our approximate solutions for WRP instances with optimal solutions from [4]. Our analysis shows the cost error is less than 1% for all the input graphs that we tested on.

Graph name	Opt. cost	Approx. cost	Error %
wrp3-11	1100361	1100427	0.006
wrp3-39	3900450	3900600	0.004
wrp3-96	96001172	96003009	0.002
wrp3-83	8300906	8302279	0.017

Table 1. Error of approximate cost

The graph below shows a comparison between the running times for the best known serial work [4] and our parallel implementation. We tested on several graph sizes ranging from 128 to 3168 vertices. Our analysis shows that our algorithm achieves speed-ups for bigger graph sizes (>600 vertices), with a maximum speed-up of 103x for the wrp3-83 graph with 3168 vertices.

Figure 2. Serial time, parallel time and speedup vs graph size

A common task in HPC is measuring the *scalability* (also referred to as the *scaling efficiency*) of an application. This measurement indicates how efficient an application is when using increasing numbers of parallel processing elements. The graph in figure 3 shows that our problem is highly scalable for a problem size of 2518 vertices (wrp3-96).

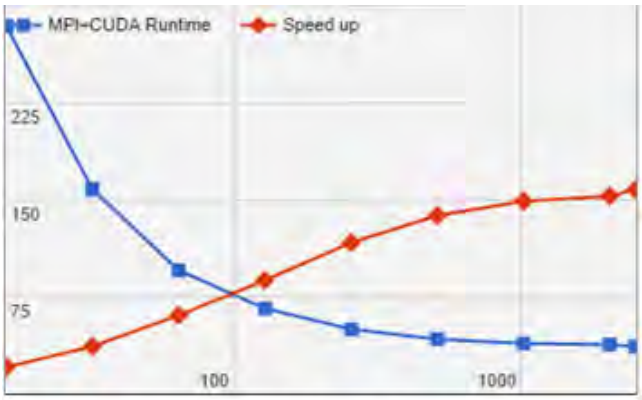

Figure 3. Parallel time and speed-up vs processes

4. CONCLUSION

After careful analysis, we have noticed some subtle yet interesting points about our algorithm. Our algorithm is highly dynamic in the sense that it is not possible to predict the size of the solution at any point before actually computing it. Because of this, the work needed to be done at every step, which is proportional to the output size, cannot be predicted. This means that work distribution is highly irregular and this leads to load imbalance among processes which inhibits performance. This uncertainty also contributes to a lot of irregular memory accesses which is also a performance bottleneck. Our algorithm is also adaptively refined, in the sense that it uses several steps to refine the solution for each vertex and then chooses the best solution among all the vertices. Algorithms that share the same characteristics as ours also share the problems of load imbalance and memory hierarchy.

5. FUTURE WORK

Our implementation suffers from the problems of load imbalance and irregular memory accesses due to the highly dynamic nature of our algorithm. Hence we wish to design a better load balancing mechanism and optimize the memory consumption of our implementation. Future work also includes making modifications to overlap more computation with data-communication.

6. REFLECTIONS

The project described in this paper was Venkata's Blue Waters Student Internship project where he learned to incorporate several principles of computation and high-performance computing into his research. This section presents Venkata's reflections about his internship and the impact that it has had on his current and future academic endeavors: My interest in computer science was ignited right from the introductory courses that I took my freshman year in college. I was fortunate to have received an opportunity to work with Prof. Yoon on this research project right from my freshman summer. At the end of the summer, we had the sequential and parallel versions of the code running on our local cluster. To our surprise, the parallel version was slower than its sequential counterpart in terms of run-time. After thorough investigation we concluded that our implementation had suffered from load balancing and thread divergence issues that hurt the performance a lot. We articulated that significant parts of our algorithm were more suitable to be handled by the CPU than the GPU and hence we started looking into distributed computing architectures like the Blue Waters Supercomputer. I then applied for the Blue Waters Internship Program and was fortunately accepted for the summer after my sophomore year in college. At the 2-week workshop, I learned parts of the C and FORTRAN programming languages in order to learn the basics of the parallel computing libraries OpenMP, CUDA, MPI, and OpenACC. I was taught how to use profiling and debugging tools like CPMAT and TAU. I was also exposed to parallel I/O libraries such as Lustre. Thanks to this experience, I am now confident using the Linux command line to navigate Blue Waters and write basic shell scripts to execute the code for my research. Having learned these skills, I am capable of using supercomputers for my research, which is at the intersection of engineering and computer science. This research experience has given me a glimpse of how computer scientists carry out research that continually shapes the world we live in. I am planning on pursuing a doctorate at a graduate school in the field of computational science, and I believe that these experiences will make me a good candidate in the application process. The Blue Waters Internship is definitely a turning point in my college career and my life in general, and having this opportunity to do cutting-edge research with real-world implications is an invaluable experience.

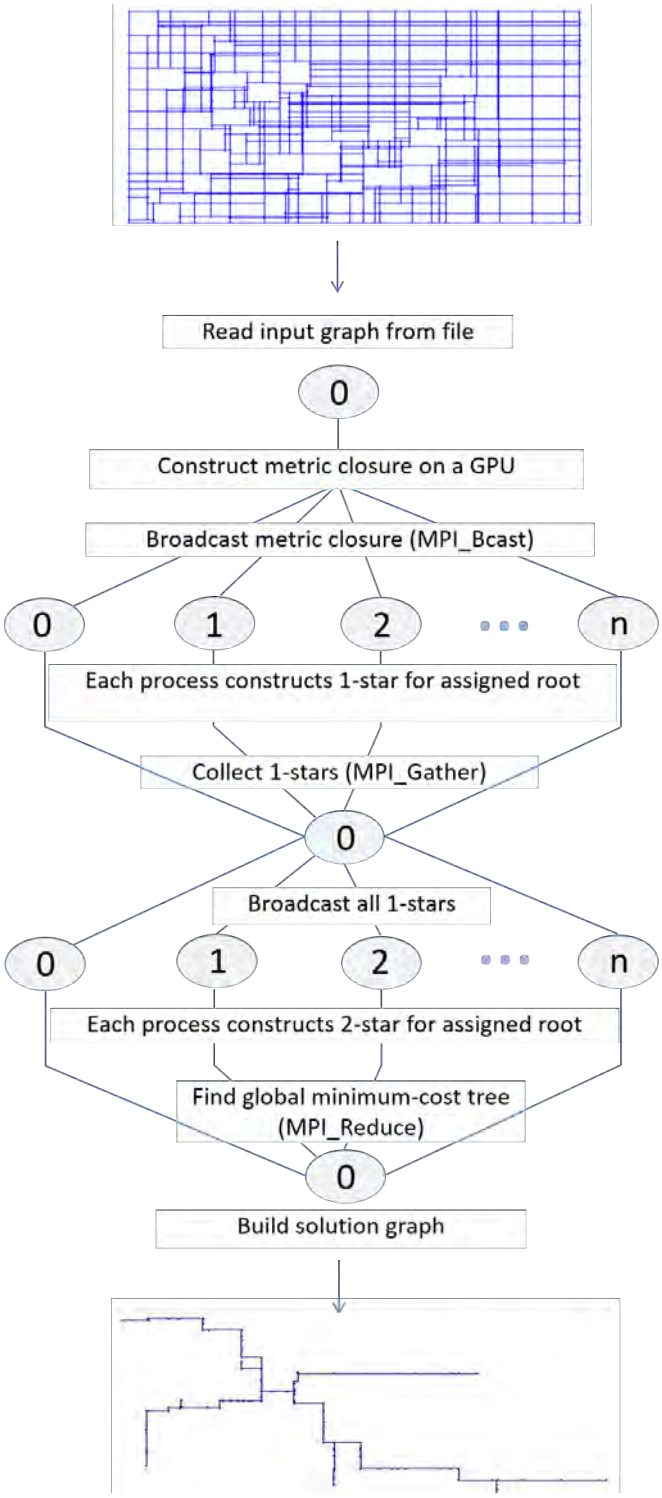

Figure 4. A graphical flow chart representation of the parallel algorithm.

```
IF master
    G ← read_graph(filename)
    (Mc, P) ← FLOYD_APSP_CUDA(G) /*get metric closure
    and predecessors matrix*/
ENDIF
BROADCAST(Mc) /*from master*/
onestar_all ← [ ]
WHILE there is a remaining root r
    onestar ← build_onestar(Mc, r)
    GATHER(onestar, onestar_all) /*to master*/
ENDWHILE
BROADCAST(onestar_all) //from master
global_min ← { }
local_min ← { }
WHILE there is a remaining root r
    twostar ← build_twostar(Mc, r ,onestar_all)
    IF cost(twostar) < cost(local_min)
        local_min ← twostar
    ENDIF
ENDWHILE
REDUCE(local_min, global_min) /*to master*/
IF master
    build_sol_graph (global_min, P, Mc)
ENDIF
```

Figure 5. A parallel approximation algorithm for GSP.

7. ACKNOWLEDGMENTS

This research is part of the Blue Waters sustained-petascale computing project, which is supported by the National Science Foundation (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. We thank the Blue Waters Student Internship Program for providing Venkata with this opportunity. We also thank the Summer Science program at Trinity College, which provided Venkata and Basileal with room and board for the summer.

8. REFERENCES

[1] Helvig C.S., Robins, G. and Zelikovsky, A. New Approximation Algorithms for Routing with Multi-Port Terminals. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *19*(10), 1118-1128.

[2] Lund, B. D., and Smith, J. W. A multi-stage CUDA Kernel for Floyd-Warshall. CoRR abs/1001.4108 (2010).

[3] Koch, T., Martin, A. and Voß, S. SteinLib: an updated library on Steiner tree problems in graphs. in Cheng, X. and Du, D.Z. eds. *Steiner Trees in Industry,* Springer US, Berlin, 2001, 285–326.

[4] Polzin, T., Vahdati, S. The Steiner tree challenge: An updated study, *11th DIMACS Implementation Challenge.* Retrieved July 06, 2015, from Princeton University: http://dimacs11.cs.princeton.edu/papers/PolzinVahdatiDIMACS.pdf.

STUDENT PAPER: GPU Acceleration for SQL Queries on Large-Scale Distributed Systems

Linh Nguyen
Hampden-Sydney College
1 College Road
Hampden-Sydney, VA 23943 USA
nguyenl16@hsc.edu

Paul Hemler
Hampden-Sydney College
1 College Road
Hampden-Sydney, VA 23943 USA
phemler@hsc.edu

ABSTRACT

General purpose GPUs are a powerful hardware with a number of applications in the realm of relational databases. We further extended a database framework designed to allow for GPU execution queries. Our technique is novel in that it implements Dynamic Parallelism, a new feature in recent hardware, to accelerate SQL JOINs. Query execution results in 1.25X speedup on average with respect to a previous method, also accelerated by GPUs, which employs a multi-dimensional CUDA Grid.

More importantly, we divided the queries to run on multiple BW nodes to investigate the scalability of both SELECT and JOIN.

Keywords

GPGPU, BWSIP, CUDA, SQL, Distributed Computing

1. INTRODUCTION

A relational database management system (RDBMS) manages the organization, storage, access, security, and integrity of data. Manipulating an RDBMS requires the use of Structured Query Language (SQL), an industry-standard declarative language capable of performing very complex queries and aggregations over data sets. In an RDBMS, these data sets are organized in structured tables, and SQL serves as an intermediary between a client program and its databases. The explosive growth of various types of unstructured data, has called for the next wave of innovation in data storage, management, and analysis, otherwise known as Big Data.

MapReduce presents one such innovation. Initially developed by Google, MapReduce is a programming paradigm widely used for data intensive and large-scale data analysis applications [10]. The architecture is simple abstraction that allows programmers to write a *map* function that processes key-value pair associated with the input data. The *reduce* function then merges all the intermediate values associated with the same intermediate key. The programming model involves three phases: *Map; Shuffle and Sort; Reduce.*

DOI: https://doi.org/10.22369/issn.2153-4136/8/1/5

In some cases, a MapReduce framework has replaced traditional SQL database, though the advantage of one over another remains a debated topic. Both approaches are very general methods through which data can be processed. In fact, there are efforts to combine the two in order to ease programmers' learning curve. In particular, Apache Hive allows programmers to write queries in SQL-like fashion and then converts the queries to map/reduce [1]. Regardless of the specifics, SQL remains highly applicable in the Big Data era thanks to the power of its declarative syntax. As such, significant efforts have been made in improving the performance of SQL, targeting parallel heterogeneous hardware architectures, which incorporates many processor types in a single machine [5, 6, 7, 8, 11, 12, 13, 14, 16].

The SQL model of data organization fits a parallel execution model extremely well since different processors can work on different portions of a table simultaneously. Among relevant attempts, the most promising approach currently utilizes the newest generation of Graphics Processing Units (GPUs), which are implemented as massively parallel architectures designed for rapidly rendering complex and realistic graphical scenes. More importantly, several general purpose software development frameworks for programming GPUs have become standard. Two such frameworks are Compute Unified Device Architecture (CUDA) [19] and the Open Compute Language (OpenCL) [17]. These frameworks allow applications to simply and effectively harness the power of the GPU. In addition, they have also become a common method of accelerating many compute-intensive applications [9].

This project aims to extend a GPU-based database to allow for the execution of SQL queries on multiple GPUs contained within the Blue Waters (BW) supercomputer system [18]. The specific focus of this project targets two common but extremely important SQL queries: SELECT and JOIN. While most previous papers utilize parallel primitives, our implementation executes with an opcode virtual machine model. In short, the key contributions of this paper include:

- **Dynamic Parallelism Approach** - Earlier GPU hardware did not include the ability to launch new GPU functions directly from threads. The BW compute nodes provide newer GPUs with the Kepler Architecture, which includes this important feature, referred to as Dynamic Parallelism (DP). We employ DP to enhance hardware utilization in accelerating JOIN.

- **Multi-GPU Configuration** - To the best of our knowledge, all previous work has shown speedups on a single-

GPU system. However, databases often reside on many compute nodes and querying data incurs additional expensive communication costs. Since the BW is a large-scale distributed system with thousands of nodes, it is particularly suitable to investigate this necessary aspect of both SELECT and JOIN queries.

- **Educational Values** - Work on this project has exposed me to a multitude of areas in Computational Sciences, such as Compilers, Computer Architecture, and Parallel Computing. Most importantly, I have gained significant experiences in multi-GPU programming. Our specific implementation is under the MapReduce dwarf. The relevant characteristics will be discussed in subsequent sections.

In summary, we performed an in-depth investigation of dividing queries across multiple GPUs and the implications of novel hardware features to explore the potential benefits of GPU acceleration for SQL queries.

2. RELATED BACKGROUND

This section provides overviews of SQL queries, and MapReduce. It also discusses current solutions in parallel SQL execution on the GPUs

2.1 SQL SELECTs and JOINs

In the following discussion, we denote a table as T_i and each column in a table as c_i where i is their corresponding enumeration.

A SELECT statement extracts data from a database in an organized and readable format. A typical SELECT statement is accompanied by clauses such as FROM, WHERE, and ORDER BY. These clauses specify the conditions on which the data is filtered. A predicate includes one or more of these conditions.

T_1

c_1	c_2
1	w
2	z
3	z

$\Rightarrow$

c_1	c_2
2	z
3	z

Figure 1: The result rows of a SELECT query: **SELECT** $*$ **FROM** T_1 **WHERE** $T_1.c_1 \geq 2$

A JOIN query requires at least two tables that share a common attribute, referred to as a foreign key. While there are many types of JOINs, we focus only on the cross-join, denoted $\bowtie$, since it is the simplest JOIN query but still embodies all the computational characteristics of other JOINS. A cross-join computes the Cartesian product of all the keys in the relevant tables. Figure 2 highlights the result rows of a cross-join between T_1 and T_2 based on the predicate $T_1.c_2 = T_2.c_2$, where c_2 is the foreign key. While the result consists of nine rows in total, the predicate reduces the result to just three rows.

A SELECT query involves a loop that examines every row in a table. A JOIN entails a *nested loop*, where for each row in the first table, the corresponding loop iterates over the entire second table and emits rows that match the predicate. Clearly, this entire process is computationally demanding for very large tables with millions of rows.

T_1

c_1	c_2
1	w
2	z
3	z

$\bowtie$

T_2

c_2	c_1
x	5
z	6
w	7

$\Rightarrow$

$T_1 \bowtie T_2$

c_1	c_2	c_2	c_3
1	w	x	5
1	w	z	6
1	w	w	7
2	z	x	5
2	z	z	6
2	z	w	7
3	z	x	5
3	z	z	6
3	z	w	7

Figure 2: The cross join $T_1 \bowtie T_2$:
SELECT $*$ **FROM** T_1, T_2 **WHERE** $T_1.c_2 = T_2.c_2$

2.2 MapReduce

As previously mentioned, the MapReduce paradigm is an active research area. The three main phases are shown in Figure 3.

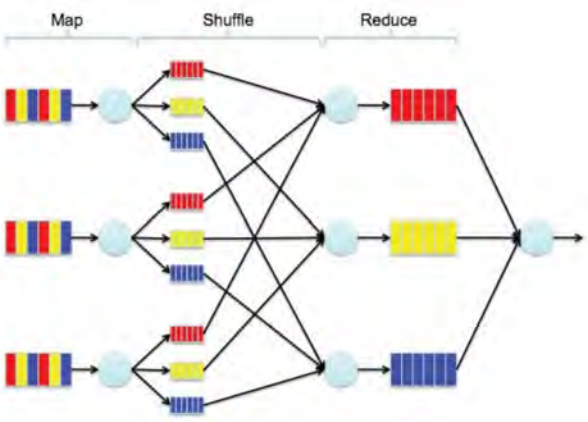

Figure 3: The three phases of MapReduce.

Map Phase: The input data is partitioned into splits and each is assigned to Map tasks to be processed in separate nodes in the cluster. These Map tasks perform computations on each input key-value pair from its assigned partition of data. Finally, they generate a set of intermediate results for each key.

Shuffle and Sort Phase: This phase sorts the data generated by the Map tasks from other nodes and divides this data into regions to be further processed by the Reduce task. Therefore, all Map tasks must complete prior to this phase. In addition, this phase distributes the data as necessary to nodes where the Reduce tasks will execute.

Reduce Phase: The Reduce tasks perform additional operations on the intermediate data by merging values associated with a particular key to a smaller set of values to produce the output. The number of Reduce tasks does not need to be equal to the number of Map tasks. For more complex data processing procedures, multiple MapReduce calls may be linked together in sequence.

MapReduce is a simple and scalable approach to achieve high speed and efficient parallel processing over a massive amount of data.

2.3 Parallel SQL Databases

There have been other works in the literature [7, 8, 11, 14] that employ the parallel primitives such as *scan, sort*, or *filter* to process queries on the GPUs. Each of these primitives is a kernel launched on a given set of data in an order specified by programmers. Every kernel then needs to retrieve relevant data from global memory. Fundamentally, this approach entails many memory accesses and consequently affects performance.

We build directly on the work presented in [3, 4, 5, 6]. This line of research proposes using a virtual machine (VM) instead. From a high-level point of view, a VM is a sequence of jump instructions inside the GPU. Each instruction is identified by an opcode and performs a certain operation. Each thread is mapped to a data point, small enough to be loaded directly into a register.

The source code of the Virginian database, which embodies this approach, is open source [4]. While inspired by SQLite [2] , the initial implementation presents a completely custom VM, which we will describe in Section 3. We modified this VM to accommodate the distributed architecture of the BW system.

3. IMPLEMENTATION

The program first builds an abstract syntax tree (AST) from the SQL query. The AST is then processed in several passes to generate a virtual program, namely a set of jump instructions that represents the query. The details of the parsing algorithm and code generations are documented in the Virginian database page [4].

3.1 Virtual Machine Infrastructure

A VM is implemented as a CUDA kernel that executes a sequence of instructions procedurally. Each instruction is identified by an opcode and contains four registers. Each opcode serves a certain functionality.

The aforementioned work partitions a SQL table into chunks of data, called the *tablet* [4, 5]. Each tablet is a fix-sized group of rows. As a result, accessing an entire table of data may involve multiple tablets, but accessing a single row of data involves only a single tablet. Partitioning is beneficial at two levels of parallelism, especially for SQL SELECT. First, for a heavily distributed table, each of this table's tablets resides on a different node and thus can be processed independently. Second, for each tablet, threads in a CUDA grid handle their corresponding rows.

Since SELECT operates only on single source table, an extension was developed to this model to support operations over two or more tables, namely JOIN [3]. CUDA's organization of threads into a grid of up to three dimensions coincides with the structure of a Cartesian product where a two-table join corresponds to a two dimensional grid. We recall the example from Figure 2, where two tables T_1 and T_2 are cross-joined on the same foreign key. Here, the x-dimension of the CUDA grid corresponds to the row index of T_1 while the y-dimension corresponds to row index of T_2. The shortcoming, however, is that this approach is limited to joining at most three tables since this is the maximum supported dimensions of a CUDA grid.

This method, which we will now refer to as the *Grid* method, is used as the baseline performance for our novel technique for accelerating JOIN, which will be discussed shortly. In addition, we scale both SELECT and JOIN to execute on multiple nodes and compare the performance with that of a single node.

3.2 Dynamic Parallelism for SQL JOINs

DP is a new functionality provided by the Kepler Architecture [19]. DP allows kernels to be launched from the device without going back to the host. The kernel, block or thread that initiates the launch is referred to as *parent*. Also, the kernel, block, or thread that is launched is referred to as the *child*. DP allows explicit synchronization between the parent and the child through a built-in device function. Launches can be nested from parent to child, then child to grandchild and so on. The deepest nesting level that requires explicit synchronization is the *synchronization depth*. Parent and child kernels have coherent access to global memory. Shared and local memories are exclusive for parent and child kernels.

The JOIN algorithm involves at least two data arrays, taken from the related data tables. Each thread takes one element from the first array in the JOIN predicate and finds the matching keys from the other array. DP implementation launches a kernel to gather the result elements in parallel for each thread.

3.3 Multi-GPU Configuration

To further improve the performance of large-scale data processing, it is essential to share workloads among distributed compute nodes equipped with GPUs. As discussed in Section 1, this project is the first to examine the possibility of breaking up a data set and running a query concurrently on multiple GPUs. Our custom data structure is useful in the context of a heavily distributed database by enabling efficient management of data between networked machines. In our implementation, tablets play a major role since they vastly simplify the process of moving data to and from different nodes. Tablets allow each node to operate exclusively on known-size records.

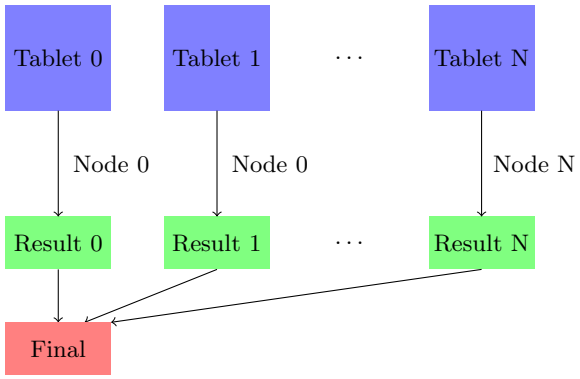

Figure 4: Query plan for SQL SELECT

A table is first divided into tablets. This step is trivial thanks to the partitioning scheme of a SQL table. Each tablet is sent to a XK compute node through MPI. On each node, each tablet is then processed by the same virtual machine row by row on the node's GPU. After producing its corresponding set of result rows, each node passes these rows back to the *master* node, which then reorganizes these rows and emits the final table to the user. Figure 4 illustrates this approach for the SELECT queries.

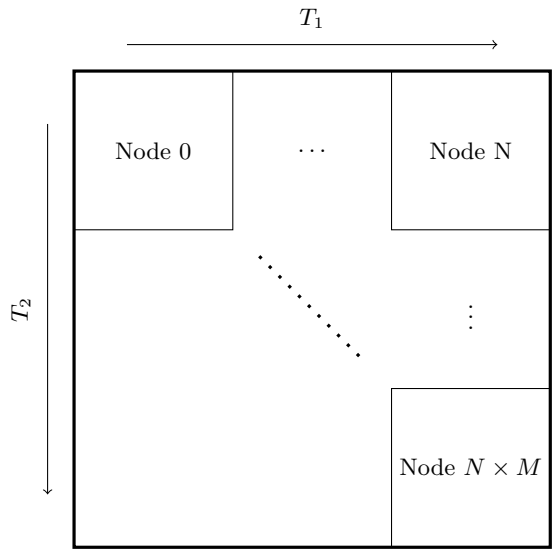

Figure 5: Query plan for SQL JOIN

Similarly, the two tables in a JOIN query are each divided into an array of tablets. Figure 5 demonstrates this plan. Table T_1 contains N tablets, and table T_2 contains M tablets. Since a JOIN statement is a Cartesian product of two arrays of data, each node is responsible for a different section of the cross product. For instance, *Node 0* computes $T_1.Tablet\_0 \bowtie T_2.Tablet\_0$. The results of these computations are again passed to the master node in the same procedure as in Figure 4.

Splitting data across multiple nodes has a two-fold advantage: data can be processed more quickly and a larger quantity of data can be processed. The database can now support much larger tables while retaining the same performance. For the same data set, more computing powers reduce the processing time.

4. RESULTS

We adopted the test data from [4, 3, 6], which includes 8 million rows randomly generated numerical values. The columns consists of an integer primary key and 2 columns each with distribution in [-100,100], a normal distribution with standard deviation 5, and another with standard deviation 20. Each of these distributions was generated once each for a 32-bit integer column and a IEEE 754 32-bit floating point column. The GNU Scientific Library was used to ensure the quality of the random distribution.

A suite of ten different queries were written to thoroughly test the different characteristics for both SELECT and JOIN. Hence, there are twenty queries in total. For SELECT, five of the queries for each suite involve integer values; the rest of the suite test floating point values. For JOIN, only one query operates exclusively on integer values; one query tests floating point exclusively; the rest of the suite test a arithmetic and conditional statements with a mixture of both integer and floating points.

We have little reason to believe results would change significantly with realistic data sets, since all rows are checked. Also, both textual and non-textual data are ultimately represented by numeric values. Besides, we have tested virtually all basic possible combinations of common case queries.

This section also highlights the educational values of our project.

4.1 Performance

We analyze in detail the runtime growth of our implementation and then discuss performance improvement relatively to the baseline as well as the scalability of our implementation as a function of the number of nodes.

4.1.1 SELECT

The characteristics of runtime growth on a single node has been discussed [5]. We therefore focus mainly on the scaling factor of the implementation. Figure 6 visually presents the speedups observed as we scale our program. Single-node execution was predictably slower by factors proportional to the number of nodes used in computation.

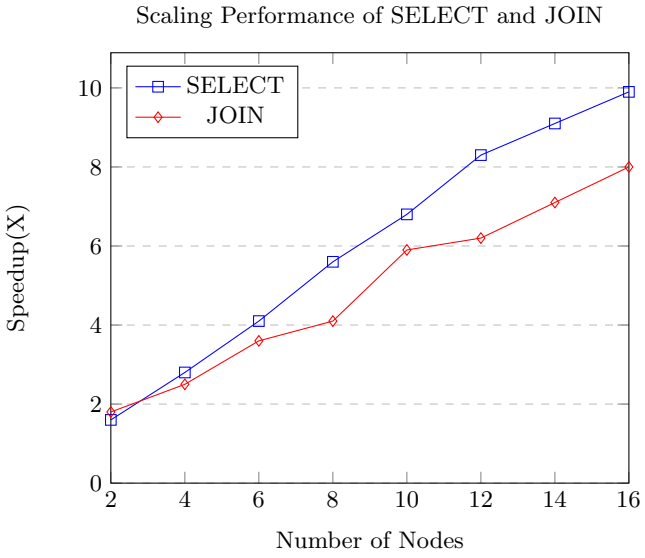

Figure 6: Scaling factors for SELECT and JOIN

4.1.2 JOIN

Test were performed using two 3500 row tables containing randomly generated values in the same layout as for SELECT. We chose this number of rows since the Cartesian product of two tables with 3500 rows each will space $3500 \cdot 3500 = 12,250,000$ rows, larger than the test number of rows for SELECT.

Figure 8(a) graphically demonstrates the differences in running times using DP and Grid methods. The mean execution time for all ten queries for Grid is 0.2526 seconds and 0.233 seconds for DP. We noticed that floating points are slightly faster and reported the average between integer and floating points. Figure 8(b) depicts the speedups of DP with respect to Grid, which averaged to be 1.245X.

Scaling SQL JOINs on distributed is less straightforward than with SELECT. Since the Cartesian product is a two-dimensional product, we chose the sizes of the two tables in Figure 5 as $\{1 \times 2, 2 \times 2, 2 \times 3, 2 \times 4, 3 \times 4, 2 \times 7, 4 \times 4\}$. These numbers correspond to the same number of nodes for SELECT and their speedups are also displayed in Figure 6. We chose these dimensions to examine the effects of the ra-

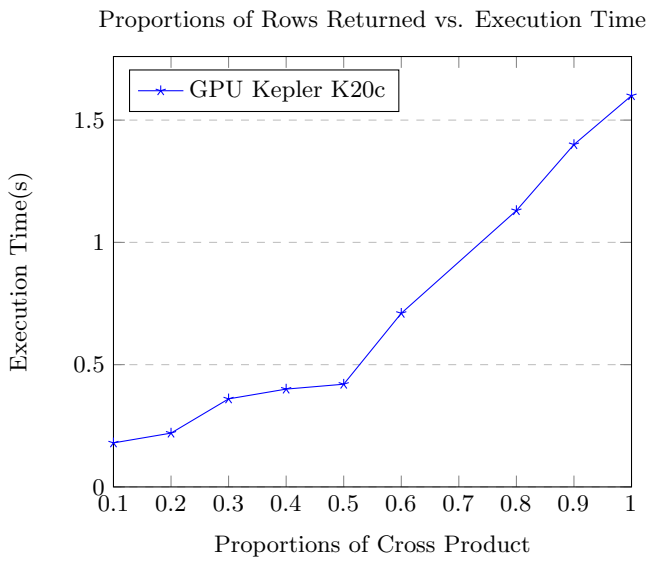

Figure 7: As fewer rows in the Cartesian product are filtered, the GPU becomes less efficient

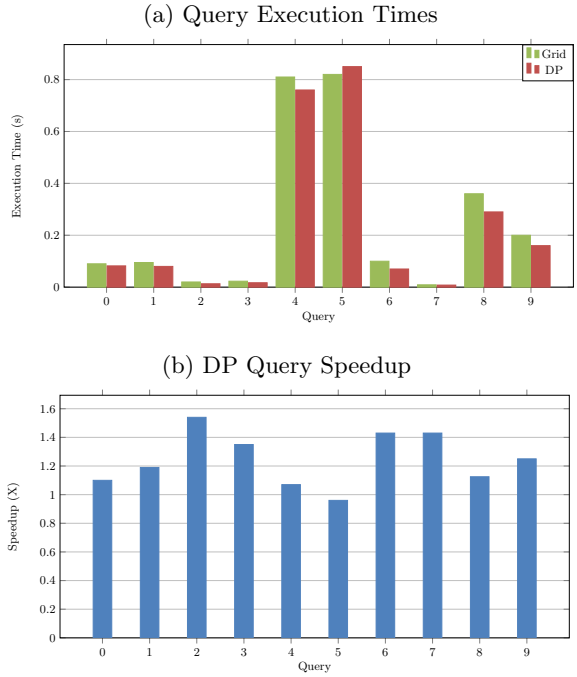

Figure 8: Variations of two different methods for accelerating JOINs

tio $\frac{N}{M}$ have on the scalability of JOINs. The graph showed a weaker scalability for JOINs due to a more complex distribution plan. The mean percentage to theoretical speedup is 54.4%.

Figure 9 provides the latency breakdown as percentages of critical parts in the execution cycle of a program. Predictably, as the number of nodes increases, the system incurs much more overhead to organize data to the assigned nodes. We notice a gradual decline in the role actual computation plays with respect to total runtime. In contrast, the node-to-node communication overhead increases linearly, up to 75.2% at 16 nodes.

4.2 Educational Values

The use of GPU in HPC is becoming extremely popular due to the high computational power and high bandwidth coupled with the availability of (*de facto*) frameworks. However, effectively programming a hybrid system with MPI and CUDA is especially difficult as the growing complexities of applications require more and more processors. Because the communication patterns and resource scheduling are common to a wide range of domains, we generalize our experiences and hope that they could be helpful to future programmers faced with similar problems.

4.2.1 Multi-GPU Programming

MPI and CUDA combine the two parallel programming frameworks enables solving problems with a data size too large to fit into the memory of a single GPU, or that would require an unreasonably long compute time on a single node. Also, this combination allows programmers to efficiently accelerate existing MPI applications with GPUs.

In multi-GPU programming, an MPI launcher starts multiple instances of an application and distributes these instances across the nodes of the BW. Each instance then launches its own CUDA kernel. A major problem arises when each node does not receive an even amount of data or does not return the same amount of data, referred to as workload imbalance. Our implementation automatically solves the first problem since data is evenly divided into tablets. The latter is harder to address since imbalance may occur due to the specific characteristics of an application. For instance, we chose a uniform distribution for our test data since the distribution has equal probability for any random region of data, thus minimizes the chance for load imbalance. It is critical for the programmer to be aware of these challenges in order to address them properly.

4.2.2 MapReduce Introduction

MapReduce has been shown to work well on the BW [15], which gives a compelling reason to develop a section on how to effectively utilize this paradigm. Database aggregation falls into the MapReduce dwarf since each stage is directly mapped to a phase in MapReduce, as described in Figure 3. In our case, the input data is the original tables and the output data is the result table.

Map: Each of the partitioned tablets in a table is *mapped* to a node. The node's number and its corresponding tablet form a key/value pair. Each node then executes the VM generated and then produces its own set of result rows. These are the intermediate results to be passed to next phase.

Shuffle and Sort: Each node sorts its own result table based on programmer's specification. For instance, these nodes arrange the numeric values in ascending/descending order.

Reduce: The master node is responsible for this phase, collecting other nodes' intermediate tables based on their keys and then rearrange these tables to into one single final table to display.

5. CONCLUSION

The Virginian database was used as a platform for the

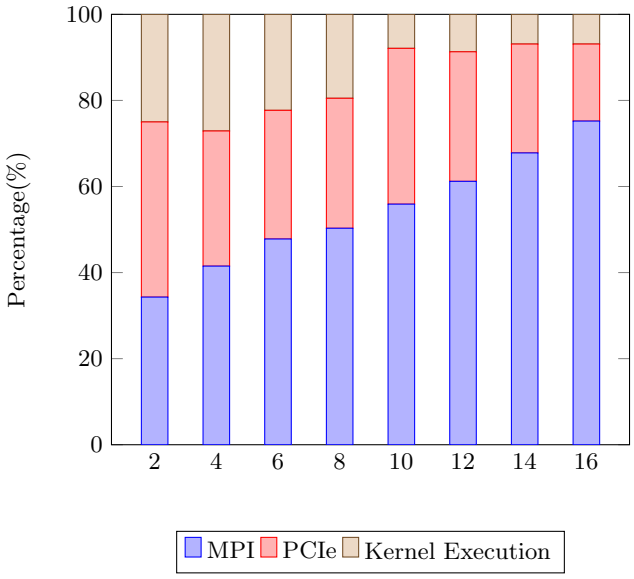

Figure 9: The fraction of each query runtime due to core parts of computation

project, enabling the use of an existing SQL parsing mechanism and switching between host and device. Execution on the GPU was supported by a completely custom VM implemented as a CUDA kernel. Our DP approach shows an average of 1.245X faster than the previously implemented Grid method. The characteristics of each query, the type of data being queried, the size of the result set, and the numbers of nodes involved were all significant factors in the performance of GPU-enabled database. Despite these variation, the minimum speedup for DP was 1.1X.

SQL is an excellent interface through which the GPU can be accessed: it is much simpler and more widely used than many alternatives. Using SQL represents a break from the paradigm of previous research which drove GPU queries through the use of operational primitives. Additionally, SQL dramatically reduces the effort required to employ GPUs for database acceleration. While still enjoying the simplicity of SQL, users can now benefit from a much reduced runtime. All the details are hidden so that users do not need to learn any additional materials besides their existing SQL knowledge.

Our opcode model allows the programmer to choose any granularity for database operation in conjunction with a relatively simple VM, while also enabling efficient data handling. Programmers can simply add or modify opcodes to support more complex queries. We also generalize the module to help with the learning curve by presenting the general challenges associated with multi-GPU programming as well as the project's relation to MapReduce.

6. REFLECTIONS

The Blue Waters Student Internship Program (BWISP) impacts me significantly on many aspects of my personal and professional developments.

The Petascale Institute (PI) allowed me the opportunity to learn how to use the Blue Waters supercomputer, using the various parallel programming frameworks it provides. This experience and exchanges with my fellow students and the instructors improved my programming skills greatly. Also, the exposure to other disciplines though these conversations opened my eyes to the vast application of Computer Science (CS). The areas I was introduced to include biomathematics, physical simulations, and financial modelling.

After the PI, working on my own project helped me apply these learned skills in practice. I gained invaluable experiences working with large software project that utilizes many software tools and involves various areas of CS, such as Compilers, Computer Architecture, and Software Development. More importantly, the experience provides me with the materials to discuss in my graduate school applications. As a result, I was accepted to all programs I applied to, including well-known institutions in High Performance Computing such as University of Michigan and Georgia Tech.

All in all, the BWISP was an immense success and equipped me with the necessary skills to succeed in my career as a Computer Science researcher.

7. FUTURE WORK

Our implementation has been designed partly to demonstrate a very general framework for GPU data processing. Using this framework, a next step is to implement and test additional database features such as other types of join, including *inner,outer*, and *natural* Joins. Modern RDBMSs are extremely complex, and much more work in this area is required to fully replicate this functionality in a GPU-friendly manner. We have shown that our opcode framework will be adaptable to this additional functionality, with modifications to our VM as appropriate to facilitate inter-opcode communication [4, 3].

Regardless of the direction, the general area of GPU application development is ripe for future research.

8. ACKNOWLEDGMENTS

This work is supported by a grant from the Shodor Education Foundataion through the Blue Waters Student Internship Program (BWSIP) and by the National Center for Supercomputing Applications, who provides the hardware used in this project.

9. REFERENCES

[1] apache hive. https://hive.apache.org/. Date accessed: 2016-04-11.

[2] the sqlite virtual machine. https://www.sqlite.org/opcode.html. Date accessed: 2016-04-06.

[3] K. Angstadt and E. Harcourt. A virtual machine model for accelerating relational database joins using a general purpose gpu. In *Proceedings of the Symposium on High Performance Computing*, HPC '15, pages 127–134, San Diego, CA, USA, 2015. Society for Computer Simulation International.

[4] P. Bakkum. The Virginian Database. https://github.com/bakks/virginian. Date accessed: 2015-05-10.

[5] P. Bakkum and S. Chakradhar. Efficient data management for gpu databases. Technical report, NEC Laboratories America, Princeton, NJ, 2013.

[6] P. Bakkum and K. Skadron. Accelerating sql Database Operations on a GPU with CUDA. In *Proceedings of the 3rd Workshop on General-Purpose Computation on Graphics Processing Units (GPGPU' 10)*, pages 94–103, New York, NY, 2010. ACM.

[7] S. Baxter. Relational Joins. https://nvlabs.github.io/moderngpu/join.html. Date accessed: 2016-03-19.

[8] S. Bress, M. Heimel, N. Siegmund, L. Bellatreche, and G. Saake. Gpu-accelerated database systems: Survey and open challenges. 2014.

[9] S. Che, J. W. Sheaffer, M. Boyer, L. G. Szafaryn, L. Wang, and K. Skadron. A characterization of the rodinia benchmark suite with comparison to contemporary cmp workloads. In *Workload Characterization (IISWC), 2010 IEEE International Symposium on*, pages 1–11, Dec 2010.

[10] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, Jan. 2008.

[11] N. K. Govindaraju, B. Lloyd, W. Wang, M. Lin, and D. Manocha. Fast computation of database operations using graphics processors. In *Proceedings of the 2004 ACM SIGMOD International Conference on Management of Data*, SIGMOD '04, pages 215–226, New York, NY, USA, 2004. ACM.

[12] R. J. Halstead, I. Absalyamov, W. A. Najjar, and V. J. Tsotras. Fpga-based multithreading for in-memory hash joins. In *Conference on Innovative Data System Research (CIDR' 15)*, 2015.

[13] R. J. Halstead, B. Sukhwani, H. Min, M. Thoennes, P. Dube, S. Asaad, and B. Iyer. Accelerating join operation for relational databases with fpgas. In *Field-Programmable Custom Computing Machines (FCCM), 2013 IEEE 21st Annual International Symposium on*, pages 17–20, April 2013.

[14] B. He, M. Lu, K. Yang, R. Fang, N. K. Govindaraju, Q. Luo, and P. V. Sander. Relational query coprocesing on graphics processors. *ACM Trans. Database Syst.*, 34(4):21:1–21:39, Dec. 2009.

[15] G. B. W. K. Manisha Gajbe, Kalyana Chadalavada. Benchmarking and performance studies of mapreduce / hadoop framework on blue waters supercomputer. In *WorldComp 15 - ABDA'15 International Conference on Advances in Big Data Analytics*. ISBN, 2015.

[16] S. Meki and Y. Kambayashi. Acceleration of relational database operations on vector processors. *Systems and Computers in Japan*, 31(8):79–88, 2000.

[17] Advanced Micro Devices. Opencl programming guide. http://developer.amd.com/wordpress/media/2013/07/AMD_Accelerated_Parallel_Processing_OpenCL_Programming_Guide-rev-2.7.pdf. Date accessed: 2015-05-10.

[18] National Center for Supercomputing Applications. Brief blue waters system overview. https://bluewaters.ncsa.illinois.edu/user-guide. Date accessed: 2016-03-19.

[19] NVIDIA Corporation. Nvidia cuda programming guide. http://docs.nvidia.com/cuda/pdf/CUDA_C_Programming_Guide.pdf. Date accessed: 2015-05-10.

January 2017

TABLE OF CONTENTS

ISSN 2153-4136 (online)