Improving Protein Docking Efficiency with General Purpose Computation for Graphics Processing Units – Customer Needs and Target Specifications

Project Team E-51:
Timothy Blattner
David Hartman
Andrew Kiel
Adam Mallen
Andrew St. Jean

Faculty Advisor:
Dr. Craig Struble

Table of Contents

I. Introduction

The Customer Needs and Target Specification Document provides a layout of the problem, the needs provided by the customers for improving the AutoDock software, and our specific quantifiable goals for satisfying these needs.

The following sections are included in this document:

- Project Background
- Project Objective Statement
- Customer Needs
- Customer Needs Classification
- Target Specifications

II. Background

AutoDock

AutoDock can perform a docking simulation in as little as a minute and although this is quite fast, a drug company can go through millions of individual simulations before having the requisite data to start physically synthesizing a new drug compound [1]. Since time is a valuable commodity, we will be using parallel computing techniques to further speed up AutoDock.

Parallel Processing

Parallel processing uses multiple processors acting simultaneously in order to solve a computational problem [2]. By running a process in parallel, problems are solved faster because different pieces of the problem can be solved at the same time [2]. The algorithms used in the AutoDock software never take advantage of this strategy, though examination of these algorithms shows potential for optimization using parallel processing techniques.

Graphics Processing Units

Graphics Processing Units (GPUs) provide parallel processing over numerous processor cores [3]. When programming for the GPUs, it is useful to use their multi-core architecture by modifying pieces of the code to run in parallel over their multiple processors. A downside on programming on GPUs is the fact that modern GPUs only support single-precision floating point values [4] in comparrison to programming on double-precision floating point CPUs.

CUDA

One can send instructions directly to NVIDIA graphic cards using the CUDA Application Programming Interface (API) [5]. The CUDA API will be used to access the parallel processing power of the GPUs. In this way, we can modify the existing AutoDock C++ code to harness the power of parallel computing and improve the efficiency of the program.

III. Project Objective Statement

Problem Statement

The popular AutoDock software uses serial processing techniques to simulate protein docking. AutoDock does not take advantage of the parallel processing power of modern computer architecture such as multi-core processor chips and graphical processing units. Due to this limitation, AutoDock runs slower than it could on machines that are capable of parallel processing.

Project Objective

Reduce docking time in the AutoDock simulation by utilizing parallel processing on GPUs, which will be tested and implemented by April 29, 2009 without funding.

IV. Customer Needs

To establish the following customer needs for a CUDA capable AutoDock the following information was obtained through an interveiw with Dr. Struble. We intended to obtain up to two additional interviews although scheduling confilcts prevented these interviews from taking place before the completion of this document. An interview with Dr. Sem and his assistant Andrew Olson will still take place to further define the requirements of our final product.

Customer:	Craig A. Struble, Ph.D.	**Interviewers:**	Timothy Blattner
Address:	1313 W. Wisconsin Ave. Room 368 Milwaukee, WI 53233		Adam Mallen Andrew St. Jean
Telephone:	414-288-3783		
Email:	craig.struble@marquette.edu	**Date:**	October 10, 2008
Type of User:	Uses AutoDock as part of his research on predictive models for the interaction of different ligands with Cytochrome P450 2D6 (CYP2D6).		

Question/Prompt	**Customer Statement**	**Interpreted Needs**
Significance of AutoDock Results	AutoDock currently has about 10 – 15 significant figures, but Dr. Struble only uses 4 or 5 of them.	The precision of the restults of our modified AutoDock should contain at least 5 significant figures
	1% - 5% are realistic goals for a difference in accuracy between our program and the original program.	Results from our modified AutoDock can only differ by 1% - 5% from the original AutoDock results.

Question/Prompt	Customer Statement	Interpreted Needs
Typical Use of AutoDock	For his research he looks at 20 different “confirmations” of CYP2D6. This means they have to run the docking 20 times per ligand. He uses 81 different ligands as a data set. Approximately 2.5 million energy calculations are performed per docking. This results in approximately 4 billion energy calculations per experiment.	Modified AutoDock should run multiple calculations in parallel to decrease runtime. Currently Dr. Struble picked ‘20’ confirmations as an arbitrary value, but could run more if it didn’t take so long. Modified AutoDock should allow for more test runs for more robust results.
Typical Run Time of AutoDock	Each experiment takes about 1 to 2 weeks on a single machine. Would like/expect to see a speed up of an order of magnitude or more. Less than that is not worthwhile and certainly not worth purchasing new hardware. However, a smaller speed up on cheaper hardware would still be significant.	Speedup the original AutoDock runtime by at least 1000% on more expensive GPUs such that the cost of upgrading hardware is worthwhile.
		Speed up of less than 1000% is still worthwhile if the hardware is cheaper.
Miscellaneous	With our new AutoDock he wants to be able to run the AutoDock command in the same way as the original. As a user he doesn't want to see any differences.	Both our version and the original AutoDock should have an identical or nearly identical interface.

V. Customer Needs Classification

Introduction

Analyzing and setting priorities is crucial to the development of the project. Categorizing the customer needs into similar groups is a beneficial step in order to divide and conquer the problem. It is also important to prioritize the needs of the customer in order to identify the critical elements that should be the strongest focus of the project development process.

Customer Needs Categories and Details

1. Accuracy/Precision
    - AutoDock software should not cause a significant loss of accuracy in the program's outputted results.
    - Customers should be satisfied with the output despite the loss of precision from performing computations on single precision GPUs instead of double precision CPUs.
    - Provide results of CUDA AutoDock test data and differences in comparison to old AutoDock results for customer review.
2. Compatibility
    - Customers without the necessary CUDA capable GPU hardware will be able to compile and run AutoDock the same as before our changes to the code.
    - CUDA AutoDock should make changes transparent to the user.
    - Command line tools of AutoDock are consistent with prior version.
    - Code that we add/modify in the software should follow the same styles as the existing code and provide clear comments for the customer to understand.
    - Released as open source software.
3. Cost
    - We should provide a low cost mechanism for parallel processing in order to speed up the runtime of an AutoDock docking simulation. We plan on using both high-end and low-end GPUs to meet this need. So, since this design decision has already been made, the real customer need is making sure that the cost of the CUDA capable GPU hardware is worth the speed up trade-off.
    - Our modified AutoDock code should be released as open-source software.
4. Speed
    - CUDA AutoDock will provide a speedup of at least 1000%.
    - We should provide measurements of speedups achieved for customer review.

Prioritized Customer Needs

1. Has minimal loss of accuracy in results
2. Is precise enough for customer to use
3. Cost of CUDA capable hardware is worth the speed up our modifications provide
4. Runs as usual if customer does not have CUDA capable hardware
5. User interaction with AutoDock commands when running our modified program is identical (or at least similar) to previous AutoDock interface
6. Results in a speed up of an order of magnitude or more with high-end hardware
7. Released as open source software
8. Coding style remains consistent with previous AutoDock code and provides clear comments
9. Provides differences in accuracy and precision for customer review
10. Provide measurements of speed tests for customer review

VI. Benchmarking and Testing

Benchmarking

In our project, the competitive benchmark is the original AutoDock program. Our target specifications are almost exclusively relative to the this original program. In other words, the metrics we use in our target specifications are all relational, comparing the original program to what we want to see in our modifications of it.

Testing

In order to compare our modified AutoDock with the original program we must first have controlled test runs on the original program and compare these results with tests on the modified program. Even though we plan on using the same seed for random number generation—in an attempt to minimize the differences between results from separate runs with the same input—parts of the algorithm are still stochastic. This will yield some randomness in the results which makes it important to statistically compare the results of a number of test runs; this means statistically comparing the distribution of results from the controlled test runs with the distribution of results from our modified program. It is obvious that we must use the same set of inputs on both tests and run both tests on the same machine for comparisons. We will also want to run the same experiments on a machine with low-end GPU hardware so we can compare the speed up factors of low-end and high-end GPU hardware. For our project we plan on using Dr. Struble's gpgpu machine for tests on high-end GPU hardware and the Systems Lab's MicroWulf cluster for tests on low-end GPU hardware (specifications for both of these machines can be found in the appendix). For sample input we will be using two data sets from researchers here at Marquette which include sample ligands and proteins from Dr. Struble's Bio-Informatics lab and sample ligands and proteins from Dr. Sem and his assistant Andrew Olson's research.

However, because of the complexity and lengthy run times of these experiments we have not yet performed them in order to acquire benchmark values. This means that the specifications listed in the following Target Specifications section are measured in one of two ways. Either they are measured in proportions (percentages) of results from our modified program and results of the original program, or they are measured in statistical scores which reflect the statistical comparisons of results from our modified program and the results of the original program. Because we have not run the controlled tests yet, we are assuming that the distributions of results will be normal. The specifications in the following section include T-test and F-test statistical analyses which assume normal distributions. However, we still plan on running normal probability tests on the distributions first, to make sure that our assumptions are correct. If they are not, then we plan on recomputing the statistical specifications listed in the next section to be measured using non-parametric statistical analyses instead of the traditional T-test and F-test analyses.

VII. Target Specifications

The following metrics table contains a list of all specifications formulated by our list of different customer needs. Each entry contains the metric number, the associated customer need, the name of the specification, the type of units used to measure the specification, a pair of marginal and ideal values which represent our project's goal, and a relative weight of the importance of meeting each goal (1 is the least important and 5 is the most important).

Metric No.	Need	Metric	Units	Marginal Value	Ideal Value	Weight
1	1	Difference in mean value between our AutoDock results and original AutoDock results	P-value of T-statistic	(1-P) < 0.05	(1-P) < 0.01	5
2	1	Difference in variance between our AutoDock results and original AutoDock results	P-value of F-statistic	(1-P) < 0.05	(1-P) < 0.01	5
3	2	Precision of our AutoDock results, measured as significant digits	Significant Digits	>= 5	>= 10	5
4	3,6	On high-end hardware, factor of speed up	Percentage	> 1000%	> 1500%	4
5	3	On low-end hardware, factor of speed up	Percentage	> 200%	> 1000%	1
6	4	Runs old AutoDock as normal on machines without supported hardware	Binary	Yes	Yes	5
7	5	Interface is identical (or similar)	Binary	Yes	Yes	4
8	7	Released as open source	Binary	Yes	Yes	5
9	8	Coding style is consistent with proper comments	Binary	Yes	Yes	2
10	9	Provide measurements of speed tests for customer review	Binary	Yes	Yes	2
11	10	Provides differences in accuracy and precision for customer review	Binary	Yes	Yes	2

VIII. Appendix

System Specifications for the gpgpu.mscs.mu.edu Machine

Processor	Dell Precision T3400 Convertible MiniTower Processor E4500, 2.20GHz, 800 2MB L2, 525W
Video Card	Dual nVidia Quadro FX3700 512MB dual DVI
Ram	2GB, 667MHz, DDR2 NECC SDRAM Memory, 2X1GB
Hard Drive	80GB SATA 3.0Gb/s with NCQ and 8MB DataBurst Cache

System Specifications for MicroWulf

Processor	Four AMD Phenom 64 9500 Quad-Core 2.2 GHz
Video Card	Four MSI NX8400GS nVidia GeForce
Ram	Four 1GB, 800MHz, DDR2 x2
Hard Drive	500GB Western Digital 7200 RPM

IX. References

[cover sheet] The Patel Group@UDEL.EDU. University of Delaware. 24 September 2008 <http://patelgroup.chem.udel.edu/joomla/images/stories/frontpagefigs/1rdswligand.png>.

[1] AutoDock. 24 September 2008. <http://AutoDock.scripps.edu>

[2] Blaise Barney., “Introduction to Parallel Computing.” 2007 https://computing.llnl.gov/tutorials/parallel_comp/#Whatis

[3] “What is CUDA?” CUDA Zone. NVIDIA. 14 October 2008. <http://www.nvidia.com/object/cuda_what_is.html>

[4] Halfhill, Tom R. “Parallel Processing with CUDA, Nvidia's High-Performance Computing Platform Uses Massive Multithreading.” 1 January 2008. NVIDIA. 14 October 2008. <http://www.nvidia.com/docs/IO/55972/220401_Reprint.pdf>

[5] “CUDA education.” CUDA Zone. NVIDIA. 14 October 2008. <http://www.nvidia.com/object/cuda_education.html>

[6] “Random seed.” Wikipedia.org. 14 October 2008. <http://en.wikipedia.org/wiki/Random_seed>