Tuesday, April 07, 2009

Dr. Craig Struble
Dept. of Mathematics, Statistics and Computer Science
Cudahy Hall 369
Marquette University
Craig.Struble@marquette.edu

Dear Dr. Struble,

This report contains an update of our progress on this project, a breakdown of the completed and remaining tasks, and a final updated schedule of the remaining weeks until completion of the project. This document also contains some preliminary results via inspection of the data as well as a description of the intensive testing that will be the major focus of the final iterations which will ensure that our version of AutoDock provides reliable results.

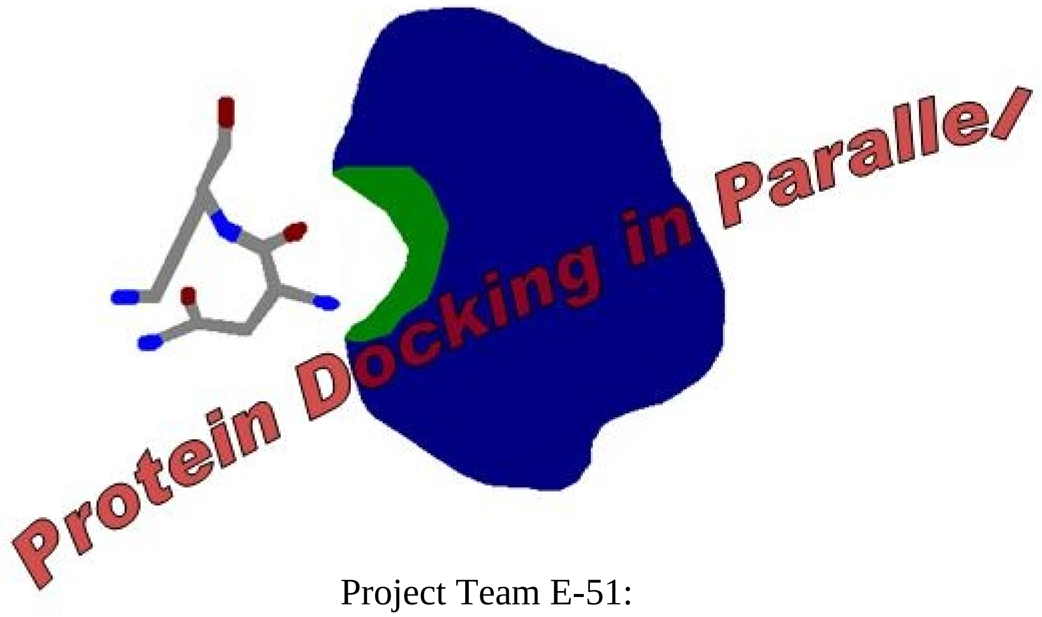
Sincerely,

Adam Mallen	David Hartman
Andrew St. Jean	Andrew Kiel
Timothy Blattner	

CC:
Dr. George Corliss
Dept. of Electrical and Computer Engineering
Haggerty Engineering 296
Marquette University

Improving Protein Docking Efficiency with General Purpose Computation for Graphics Processing Units –

Formal Proposal



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

Faculty Advisor:
Dr. Craig Struble

I. Executive Summary

Dr. Struble has commissioned our team to increase the efficiency of protein docking experiments by implementing parallel processing techniques in the existing AutoDock docking software. Currently AutoDock uses serial processing only and a significant increase in speed is expected with the implementation of parallel computation. With our plan, parallelization is facilitated by the CUDA API designed by NVIDIA which allows a program written in C++ to be automatically compiled in a suitable manner to execute pieces of the program in parallel on a NVIDIA GPU [1]. GPUs are inherently suited to the task of parallel processing due to their many individual cores which act independently [2].

GPUs are an attractive solution to the problem of cost and time effective parallel processing because they are common, thus cheap, and the CUDA API allows for an easy transition into learning to program for them [1]. Upon completion of this project any current AutoDock user with capable hardware (many workstations already exist that meet minimum specifications) will be able to download and compile our new code and use AutoDock as they always have with no apparent change in functionality to the user except for faster run time for experiments. The value of this optimization becomes apparent when one considers the fact that users typically run huge numbers of docking simulations within one experiment and the cost of using powerful supercomputers or large clusters is expensive.

With the release of our final product, as well as our testing results, we are confident that users whom already have supported hardware will have no hesitation in adopting our version. While we hope our improvements will be significant enough to warrant hardware upgrades, the average user is expected to be able to take advantage of our product with no additional cost.

II. 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 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.	Metric	Units	Marginal Value	Ideal Value	Weight
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	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	Precision of our AutoDock results, measured as significant digits	Significant Digits	>= 5	>= 10	5
4	On high-end hardware, factor of speed up	Percentage	> 1000%	> 1500%	4
5	On low-end hardware, factor of speed up	Percentage	> 200%	> 1000%	1
6	Runs old AutoDock as normal on machines without supported hardware	Binary	Yes	Yes	5
7	Interface is identical (or similar)	Binary	Yes	Yes	4
8	Released as open source	Binary	Yes	Yes	5
9	Coding style is consistent with proper comments	Binary	Yes	Yes	2
10	Provide measurements of speed tests for customer review	Binary	Yes	Yes	2
11	Provides differences in accuracy and precision for customer review	Binary	Yes	Yes	2

III. Schedule

We have adopted an Agile Software Development plan for our project. This involves an iterative and adaptive approach to addressing and completing the tasks we have identified in our work break down structure. Notice that in our schedule we have separated the remaining weeks of the semester into separate two week sections. Each of these will be a single iteration according to the agile software development plan. Our understanding of this system led us to a plan where each iteration ends with an analysis of the results of that iteration followed by a decision of the specific goals for the upcoming one. With this approach we will follow the Agile Software Development's Manifesto by reflecting after each iteration on how to become more effective in the next iteration and then adjust our methods accordingly [4]. In this way we will also be embracing the Agile Software Development's Manifesto by staying flexible in order to welcome and adapt to changing requirements in our project [4]. This means, though we have a rough schedule of the order in which we will approach the tasks of our project, we leave our plan open to the possibility of adapting to unforeseen issues as they show up over the course of the project. However, in order to avoid a loss of efficiency and lack of motivation from the removal of hard time constraints on tasks, we are also placing emphasis on concrete results at the end of each iteration showing progress in at least one of the major sections identified in our work breakdown structure. Attached to this document is an updated schedule which details what was accomplished in the last iteration, and what our team plans to accomplish in the upcoming iterations before the end of the semester.

IV. Experimental Validation

Because we have opted for an agile and test driven approach, validation and evaluation are an ongoing process for this project. However, most of these evaluations are simply a series of test cases to check that a specific iterations modifications performed as expected. This way we can move on to further modifications without running the week long tests required for a complete statistical validation that our results are not different that the original program's. Only a few of the evaluations already performed were designed to validate the target specifications laid out earlier in this document. Furthermore, following our updated schedule we had not begun the bulk of our serious and thorough statistical analysis of the differences between our version and the original version's results until two weeks ago. This means that this document is not, in any way, a comprehensive report on all our experimental analysis of the project. The following description of our validation experiments includes four main parts. The first is an outline of the simple tests already performed to validate the binary target specifications. Next, is an outline of the small experiment performed during the last iteration to convince us that our modified program does not yield wildly different results and to convince us that is appropriate to move on to the final in-depth analysis on this version of the program. The third part describes the final and thorough statistical analysis we have begun to perform in order to provide conclusive empirical evidence that our modifications to the program have not altered the reliability of the results. The fourth and last part describes all other tests we still plan on performing but have not been able to yet because we lacked a completed version of our modifications.

Part (1)

Metrics 6, 7, and 9 are binary criteria and thus can be analyzed by simple inspection to determine whether the team has met the requirement or not. The option to compile the original program or our modified version is made at compile time, so users who want to run the original code can do so in exactly the same way they always have. Users who want to run our modified program need only enter a simple command line option during compilation to run our version. After this small difference, running our version of AutoDock is identical to the original. In this way, our changes to the program are transparent to the user.

Metric 3, similarly can be validated through simple inspection. It turns out that the original AutoDock program's log files store the results of the free binding energy with only 2 decimal places. Since all our preliminary trial runs of our version of the program also produce results with 2 decimal places, we consider this target specification validated.

Part (2)

The past iteration has yielded a working modified version of AutoDock which runs most of the free binding energy calculations on the GPU. In order to test that these modifications have not decreased the reliability of the results, we have performed a small experiment with a simple statistical analysis of the results of these experiments. This small exploratory experiment was simply to convince the team and our advisor that our modifications have not ruined the program's reliability and convince us that we should proceed to the next step and begin our large-scale, in-depth, statistical analysis described in part 3 of this section of the document.

This experiment consisted of 200 runs of the original program on the CPU and 200 runs of our modified program which ran most of the energy calculations on the GPU using CUDA. Each of these used the exact same input ligand, protein, and performance parameters (e.g. random number seed, number of individuals in the population, maximum number of energy calculations, maximum number of generations). We performed a two-tailed two-sample T-test comparing the means of the distributions of the resulting free binding energies of each set of 200 runs. This test assumes normal distributions but does not assume equal variances between the distributions.

The resulting P-value of the test was 0.7647. This high value signifies extremely low statistical confidence in the claim that the mean of the two distributions are different. Other statistics from the results can be found in the following table:

	200 Runs of the Original Program	200 Runs of our Modified Program
Mean	-6.8497	-6.8658
Minimum	-8.5300	-8.7500
Maximum	-5.8600	-5.8600
St. Deviation	0.5212	0.5501
Confidence Interval on the difference between means		[-0.1226, 0.1547]

From this test we reached several conclusions. The first and most important is that our modified program passes this preliminary test and that we should move on to conducting our more thorough experiments and in-depth statistical analysis on this modified version of the program. However, a few things to note are the fact that these two distributions do not appear to be normal (one of the assumptions in our two sample T-test) and that this experiment only tested the results of one set of inputs. These issues will be addressed in the upcoming weeks when our team's efforts are devoted to running the validation experiments described in part 3 of this section of the document.

Part (3)

The bulk of our validation is yet to be completed and lies in the task of thoroughly analyzing the difference in the results of the original and modified versions of the program over a variety of input ligands-protein pairs and performance parameters. These tests are designed to validate metrics 1 and 2. The length of time necessary for performing test runs of the original program has prevented us from completing this crucial step of validation, but our updated schedule has accounted for the lengthy experiments. In fact, our control experiments have been running for the past three weeks because they do not require a working modified version of the program. The following outlines the major steps of the final experiments and thorough analysis we will perform by the end of the semester. These results will be included as the final results of our project and will serve as final validation that we have satisfied our major specification: our modified program yields equally reliable free binding energy results as the original program.

<u>Experiment Outline:</u> 3 experiments with different ligand/protein inputs that came as examples with the AutoDock 4 download. Each will consist of 150 control runs using the default parameter files that came with the examples with the following changes:

ga_pop_size set to 1500
ga_num_evals set to 250000000
ga_num_generations set to 270000
ga_elitism set to 10

Also, with each experiment we will run 150 runs using our modified program with the same ligand/protein inputs and parameter files as the control runs. Then, for each experiment, we will carry out the following analysis on the results of these 300 runs:

<u>Statistical Analysis Outline:</u>

We will perform a Wilcoxon rank-sum test on the two different distributions to check if the two samples are drawn from the same distributions. We chose this nonparametric test because it does not assume that the distributions generating the samples are normal. After performing our exploratory two-sample T-test on the preliminary results described in part 2, it was obvious that the distributions were not normal and that a T-test was not appropriate. The P-values returned by this test are similar to those returned by the T-test in that high P-values denote high confidence in the hypothesis that the two samples came from the same distribution and low P-values denote low confidence in the same hypothesis. This means that we can keep metrics 1 and 2 of our target specifications even though we have decided to run rank-sum tests instead of T-tests.

Part (4)

Part 3 described the upcoming validation for metrics 1 and 2, but this section will describe the validation experiment plans for metrics 4, 5, 8, 10, and 11 which have also not been evaluated.

Metrics 4 and 5 involve measuring the speed up of our modified program on both low-end and high-end graphics cards. However, due to the length of time that these docking simulations take we have decided not to run any speed tests on low-end graphics hardware. We will still, of course, be running the speed tests on the high-end graphics hardware because this kind of speed up is the goal of the project. These speed tests will consist of 10 benchmark runs on the GPGPU.mscs.mu.edu machine and 10 runs of our modified program for each of the 3 input ligand/protein combinations and parameter files described in part 3. Then the mean time taken for each of the corresponding sets of benchmark and experimental runs will be compared. This comparison will consist of calculating the percent of speed up that our version achieved over the benchmark runs using the same inputs. These values are what will be used to validate metric 5.

Metrics 8, 10, and 11 involve the way we release our program to the public. Obviously we can not validate these until we are finished with all other parts of the project. To validate these we simply need to observe that we release our software as open source and release a description of our validation tests and their results with our program.

VI. 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

VII. References

[1] "CUDA education." CUDA Zone. NVIDIA. 14 October 2008.

<http://www.nvidia.com/object/cuda_education.html>

[2] "What is CUDA?" CUDA Zone. NVIDIA. 14 October 2008.

<http://www.nvidia.com/object/cuda_what_is.html>

[3] Struble, Craig Ph.D. Personal interview. 14 Nov. 2008.

[4] "Principles behind the Agile Manifesto" Manifesto for Agile Software Development. 31 Jan. 2009.

<http://agilemanifesto.org/principles.html>