REU Summary

Huston Rogers

August 2013

0.1 *The Overall Problem I was tasked with*

The focus of my internship was expanding my knowledege of parallel programming techniques and how they can be applied to next-generation hardware. To this extent, I was tasked with developing a method for computational integration and implementing it in a highly parallel fashion, that was based upon the Boltzmann integrals. I developed a hybrid parallel program for use with computational numerical integration in spherical coordinates that can be used on the Intel®Xeon Phi Coprocessor.

0.2 *What I accomplished in this program*

0.2.1 General Outline

Develop a parallel code for spherical integration using the trapezoidal rule

Implementation of the code

Testing against Volume Integrals

Applications of the code to the Boltzmann Problem

0.2.2 Implementation

What was implemented

In this REU, I developed a parallel code for numerically evaluating integrals in spherical coordinates. I used the trapezoidal rule to evaluate integrals. The

Spherical Conversion is shown below as well as an example of the trapezoidal rule usage.

$$\begin{aligned} \mathrm{d}x\mathrm{d}y\mathrm{d}z &= \rho^2 * sin(\phi)\mathrm{d}\rho\mathrm{d}theta\mathrm{d}\phi \\ x &= \rho * sin(\theta) * cos(\phi) \\ y &= \rho * sin(\theta) * sin(\phi) \\ z &= \rho * cos(\theta) \end{aligned} \tag{1}$$

$$\iiint_S f(\rho,\theta,\phi)\rho^2 sin(\phi)\mathrm{d}V = \sum_{\rho=0}^{N_\rho}\sum_{\theta=0}^{N_\theta}\sum_{\phi=0}^{N_\phi}\frac{f(\rho,\theta,\phi)+f(\rho+1,\theta+1,\phi+1)}{2}\rho^2 sin(\phi)\Delta\phi\Delta\theta\Delta\rho \tag{2}$$

How it was implemented (code)

The code was designed to run on Beacon, a next-generation supercomputer based on the Intel®Xeon Phi™coprocessor. The code is written in C++. The implementation in parallel begins with a parallelization across the Intel®Xeon host processors. OpenMP was used to map work to the cores of the host processor. Message Passing Interface (MPI) was used to distribute the problem among multiple processors, based upon a decomposition of a spherical grid into subdomains. The second step of implementation of this code includes using the Intel®Xeon Phi ™Coprocessor. The coprocessor can be used with more than 200 OpenMP threads to evaluate (**??**). I did not finish an implementation utilizing the Intel®Xeon Phi ™Coprocessor, but I do have results on the Intel®Xeon host processors.

Because of an inter-dependency among the integrals, the computations had to be split up into two major sections. OpenMP was used to parallelize the summations using nested for loops and the collapse argument. Some pseudocode of the problem is included below.

- Initialize Function and Grid
- Begin Density and Moment computation
- Sum and Reduce Density and Moments, share among processes

- Begin Temperature and Heat Flux computations
- Sum and Reduce Temperature and Heat Flux computations, end

The code will be available for download at https://github.com/snowbird294/CSURE.git

0.2.3 Testing Against Volume

This specific problem is based around how computational integration can be applied to the Boltzmann Integrals. This problem began with testing the code against volume integrals. In my learning, I cames across many methods for computing a volume including various geometrical decompositions and Monte Carlo methods. The integral used to evaluate the methods tried is below.

$$\iiint_S 1\rho^2 * sin(\phi)\mathrm{d}\rho\mathrm{d}\theta\mathrm{d}\phi \tag{3}$$

The method chosen uses a mesh decomposition in each of the three directions, ρ,θ,ϕ, which divides a sphere with size Radius$_{max}$,2π,π into volume elements with volume determined by the equation below. The total volume is equal to the sum of all the volume elements. This method was chosen for it's ability to approximate volumes very accurately and for the ability to be used in other spherical integrations.

$$\rho^2 sin(\phi)\Delta\rho\Delta\theta\Delta\phi \tag{4}$$

0.2.4 Applying the Quadrature code in the context of the Boltzmann Program.

The next step of the project came with converting the Boltzmann integrals from a cartesian coordinate system to spherical. Using the relationship between volume integrals where $\mathrm{d}x\mathrm{d}y\mathrm{d}z = \rho^2 sin(\phi)\mathrm{d}\rho\mathrm{d}\theta\mathrm{d}\phi$, the Boltzmann integrals were converted for use in the new grid, as shown below. The gas constant is not needed in these integrals because the equations have been non-dimensionalized. A method was also provided for determining the c^2 value in the temperature integral in cartesian coordinates. A spherical conversion, where θ is the azimuth angle and ranges from 0 to 2π, and ϕ is the inclination angle and ranges from 0 to π, was used to provide the values needed in the calculations for c^2. Verification of the code that involved the Boltzmann Integrals was performed

by using the known bulk values that were used in the Maxwellian Distribution function, which is also shown below, calculating any non-integrated values, and knowning that, for this situation, the heat flux should be 0 because the gas is in equilibrium.

$$
\begin{aligned}
f &= \frac{Density_0}{(\pi * Temp_0)^{3/2}} * e^{-c^2/Temp_0} \\
&\iiint_S f\rho^2 sin(\phi) \mathrm{d}V = Density \\
&\frac{\iiint_S f V_i \rho^2 sin(\phi) \mathrm{d}V}{Density} = U_i \\
&\frac{\iiint_S f c^2 \rho^2 sin(\phi) \mathrm{d}V}{3 * Density} = Temperature \\
&Density * Temperature = Pressure \\
&\iiint_S f c^3 \rho^2 sin(\phi) \mathrm{d}V = HeatFlux \\
&c^2 = (V_x - U_x)^2 + (V_y - U_y)^2 + (V_z - U_z)^2 \\
&x = \rho * sin(\theta) * cos(\phi) \\
&y = \rho * sin(\theta) * sin(\phi) \\
&z = \rho * cos(\theta)
\end{aligned}
\tag{5}
$$

The Boltzmann integrals were rewritten as sums so that they could be better represented in the code, similar to the example below.

$$
\sum_{i=1}^{N_\rho} \sum_{j=1}^{N_\theta} \sum_{k=1}^{N_\phi} V_{\rho_i}^2 * sin(V_{\phi_k}) * \Delta\phi\Delta\theta\Delta\rho \tag{6}
$$

0.2.5 Unfinished Tasks

Code verification needs to be performed to prove that the code gets the correct order of error. That can be measured with a refinement study. Any arbitrary grid can be used initally in the verification, and refinement in each of the three directions (ρ,θ,ϕ) can be done until the error values for each are relatively close together, then refine the entirety of the grid, keeping the proportinality of the three directions the same in order to keep the levels of error close to the same. Ideally, the level of error should be of an order of magnitude near 10^{-12}. This

level of error is determined by the maximum precision that can be held by a variable of type double.

After creating the program and designing it to run in a hybrid, parallel fashion, the next goal is utilizing Many Integrated Core (MIC) architectures. Using offload directives, determine a method for utilizing the MIC architecture that is included on each node of Beacon. This would require the transfer of any arrays used as well as any variable to the MIC card, then the transfer back of the values computed on the card. Additionally, each processor should only use the MIC cards associated with it.

0.3 *Reflections on Missed Engagement Opportunities*

In retrospect, i missed several engagement opportunities in this REU program. My approach to this REU program was too relaxed for the level of work being done. I came in with a mindset that was not professional and the work I accomplished reflects that. I should have applied due diligence to the work at hand, and approached my mentors with more questions to better illustrate to them my progress and to also further my understanding of the exact problem I was tasked with.

My experience in this REU has been a very humbling learning experience. From seeing the quality and amount of work put forth by the other students, I see what I should have achieved and will endeavor in the future to bring the appropriate level of professionalism to any REU, and to the CSURE program if I am granted an opportunity to come back. I would like to thank Dr. Brook, Dr Peterson, and Dr. Wong for the opportunity to be part of the CSURE program. I will endeavor to be a better representative of any professional internship or REU program in the future.

I continue this section with a discussion of questions I should have asked during the course of the REU.

[1]When implementing the trapezoidal rule, a rule had to be made for the maximum boundary conditions, otherwise the program would attempt to access unallocated space. Question 1: A simple if loop works by determining the function value at the boundary conditions, but does it need a condition for the two angular directions? Question 2: What method could be used in conjunction with other quadratures, or is there a method that would allow that? Question 3: is the rule being implemented correctly through the entire code?

[2]When changing the code to parallel, the values gotten back are clearly wrong and inconsistent with what they should be. Question 1: In the OpenMP directive, have all variables that are private to each iteration of the loop been declared correctly, and have all variable that are changed with each iteration

been declared correctly? Question 2: With the current MPI implementation, is the work being divided up correctly, where each process gets their fair share of work, and is not doing extra work? Question 3: In the offload sections, are all variables being offloaded correctly, and are calculated variables being passed back correctly?

[3]Scaling the grid should result in a smaller error, but the error does not decrease. Question 1: Are any of the calculations for the variables clearly written wrong. Question 2: The calculations are based on other calculated values; do those calculations look correct, and are the values needed in between correct. Question 3: The distribution function has an initialization along with the grid, are either or those incorrect?

[4]Examining the calculated volume, it was incorrect. It should have been equivalent to $\frac{4}{3} * \pi * 10^3$ since the radius is 10. Question 1: Is the formula for volume correct? Question 2: The volume is calculated with specific values, are any of those values coming out wrong? Question 3: Are all the values initialized in parallel correctly?

[5]Scaling of the grid still shows errors where an increase of grid size takes the calculated values to zero. Question 1: Are the variables being used precise enough to hold such small values? Question 2: In the reduction processes, are any variables being accidentally truncated or otherwise affected? Question 3: Is some value being cast as an integer in an important calculation, or is it being cast as a 0?

0.4 *Moving Forward*

This section adresses the questions asked in the previous section.

[1]For the trapezoidal rule, the if statement should accomodate each boundary condition differently. The Velocity arrays were coded in such a way that the boundary conditions were all satisfied and could all be accomodated with one if statement. Changing the quadrature would not require drastic changes, but

would likely require a change in the way the boundaries are treated. A change would require a more in depth look before determining what other changes would need to be made.

[2] In the OpenMP Directives, occasional vairable were in the wrong sections, or were forgotten entirely, leading to incorrect results. In the MPI implementation, a slight change was forgotten, so the values overall were directly proportional to the number of processes which ran. In the offload sections, the statements need to be designated correctly when offloaded. The offload sections also need to have allocated space before transferring needed variables.

[3] When reviewing the calculations, a mistake was made where phi had been replaced by theta in many instances including volume, density, and average bulk values. The neccesary changes were made. The probability function was examined against the function provided in Dr. Brook's thesis, and was verified.

[4] As mentioned previously, an error in the volume formula was found and corrected. The values being used in the volume calculation were examined and proved to be correct. An issue was found where the volume came out differently between runs. The OpenMP directives were examined first, and showed that the variable was declared incorrectly. A change solved the issue.

[5] All variables being used are integers for counters and boundaries, and doubles are being used in the intermediate steps. The values being used do not exceed the memory size of a double. No variables appear to be truncated in any process. as an extra thought, the size of the grid in physical memory was computed, and was determined to be more than the memory available to any single processor. Meaning the error in the code is not in the grid size implementation, because a small amount of error should be easily found in a less fine grid.

0.5 *A Valuable Learning Experience*

In this REU, I learned valuable lessons about parallel programming. I extended my programming abilities in C++ and learned how to develop programs in parallel using both OpenMP and MPI. I also learned about the Many Integrated

Core architecture and studied, but did not implement methods for utilizing the architecture. My greatest takeaway from the program was the opportunity to learn what scientific research demands with regards to the level of professionalism required.