Midwestern State University GR GPU - Department of Computer Science NTASC Project Proposal

Team Name:

‘Two Loops’

Project Name:

Collatz Conjecture Verification

Steven Kundert: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Scott Gordon: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Summary Of Problem:

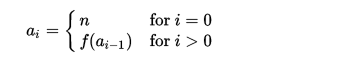
The Collatz conjecture is known by many names: the Ulam Conjecture, Kakutanl’s problem, the Thwaits conjecture, Hasse’s algorithm or the Syracuse Problem. The series of numbers is known as the hailstone sequence. It is a famous unsolved problem from mathematics and number theory, which has not been proven. Although no proof exists most mathematicians who looked into it believe it is true because of the large amount of experimental evidence to support it. Generally speaking, the conjecture has been verified for numbers as large as using computing machinery and all initial values tested so far eventually end in the cycle (4:2:1) So what is it?

The Problem can be stated Simply:

For any arbitrary positive integer, if the number is even, divide it by two. If the number is odd, triple it and add one. We could also define it formally as:

Macintosh HD:Users:scott:Desktop:Screen Shot 2017-02-17 at 8.15.22 PM.png

Now, we form the sequence by performing this action repeatedly using the result as the input for the next iteration:



a i {\displaystyle a\_{i}} is the value of f {\displaystyle f} applied to n {\displaystyle n} recursively i {\displaystyle i} times; a i = f i ( n ) {\displaystyle a\_{i}=f^{i}(n)} .

The Collatz Conjecture is: This process *eventually reaches the number 1 regardless of which integer was chosen initially.*

In Example:

*n* = 12, the sequence is 12, 6, 3, 10, 5, 16, 8, 4, 2, 1. And takes 10 steps

n = 13, the sequence is 13, 40, 20, 10, 5, 16, 8, 4, 2, 1. And also takes 10 steps

The longest progression for any number less then 10 billion is:

75,128,138,247 which has 1228 steps, yes, even this very large number eventually reaches 1.

Strategy:

1. Compute the problem sequentially on an array of X numbers, calculating the number of steps in the sequence for each number.
2. Then, we will compute the problem in parallel using X number of CUDA threads. we will exploit the potential of the NVIDIA GPU as much as possible here.
3. Conduct a performance analysis

Methodology:

Using C and CUDA we will implement a struct, known as ‘CollatzNum’, which contains a number and a step count. We will then create an array of ‘CollatzNums’ of a certain size in both main memory and global memory. Then we will load the array into Global memory, we will call our CUDA threads to compute the hailstone sequence for each number in the array ‘CollatzNums’, and we will copy the contents of global memory back into Main Memory where we will handle the results. Meanwhile on the CPU we will calculate the Hailstone sequence in linear style execution to double check our GPU bound results, and in order to perform time analysis. We will then do a Boolean comparison to verify our results. Finally, Our Main(); function will calculate the total time of both methods and will perform a simple performance analysis.

Presentation:

If possible we will attempt to create a simple graph or visualization of our data as well as a PowerPoint and a presentation along with our analysis to be presented at the north Texas area student conference 2017.

Expectations:

We expect to observe significant speedup between the two methods, even with large sets of data or data with large number of steps.