CS6068 Fall 2014

Final Project Design Document

31 October 2014

Connor Pike & Jacob Holbrook

Sieve of Sundaram

1. **Background and Motivation**

Prime numbers are useful in a wide variety of computer science problems, such as hashing and public key cryptography. Sieve theory is a concept of number theory used to find a specific set of numbers. In class we learned about the Sieve of Eratosthenes, which is a sieve used to find all prime numbers below a given maximum. The Sieve of Sundaram is a more modern approach to the same problem. We believe that this algorithm is open to significant parallelization as there are no serialized steps in the algorithm.

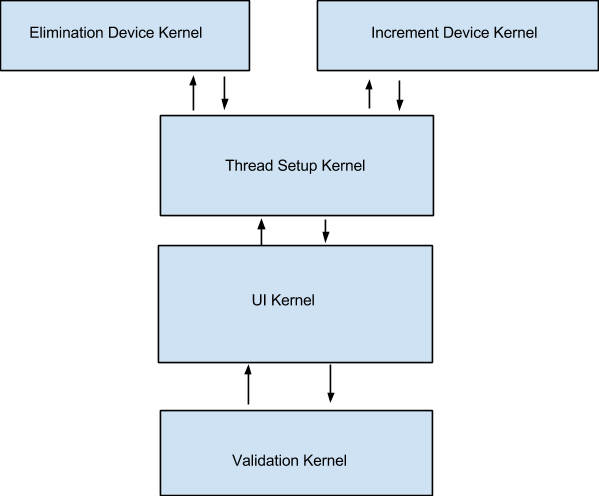
1. **Application-level Objectives**

Our main objective is to parallelize an algorithm which identifies probable prime numbers as quickly as possible. The program should work based on a dynamic input from the user with as little performance change as possible on different inputs. This will give other developers, whose projects rely on prime numbers, a way to quickly find as many as possible.

1. **Design Objectives**

Sieve of Sundaram Algorithm:

1. Start with a list of integers from 1 to n
2. remove all numbers of the form i + j + 2ij where:
   1. 1 <= i <= j
   2. i + j + 2ij <= n
3. the remaining numbers are doubled and incremented by one



Both steps two and three of this algorithm are readily exposed to parallelization as neither are dependent on serial steps. By parallelizing step two we can eliminate many numbers at once. While in step three we can find all of the odd prime numbers in parallel.

The program will have one host kernel to prompt the user for maximum and which version to use, then run the program, record the metrics, launch the validation kernel and finally report back to the user. The program will also have one host kernel that will set thread and block size, launch the device kernels, and return the list of all prime numbers. The program will have a device kernel that eliminates all numbers of the form i + j + 2ij and returns the modified array. We will have multiple device kernels that perform this step in order to compare and find the fastest. There will also be another device kernel to double and increment the remaining values in the array and return the modified array. Once again we will have multiple versions of this device kernel in order to find the fastest. Finally there will be a third device kernel to compare the output to our check array and guarantee it is valid.

The interface will be a simple text interface ran in the console. There will be inputs for the maximum as well as a list of the types of sieve the user wants to run. When complete the program will output the list of prime numbers, a yes/no on if this list is valid and the time it took to run the algorithm.

1. **Performance Goals**

Our performance should be faster than the serial implementation.

We should be able to perform the algorithm on large ranges of numbers.

1. **Verification**

We will compare the output of our program with a list of prime numbers to make sure we have found every prime in the range. We will also time our program and compare it to a serialized version as well as different parallel versions of the algorithm to find the fastest possible.

1. **Schedule and Division of Work**

Interface Module - Jacob

Verification Module - Connor

Device Modules - Both/Pair Programming

Due Date- Monday December 8th?

Final Modules Ready - Friday November 21st

Presentation Ready - Monday November 1st

The interface and Verification modules should be fairly straightforward serial modules and should be completed quickly. The device modules however will be more complex and the multiple versions we will develop for optimization will take up the bulk of the work. We will try to have the development of the program done with some buffer time before the due date. The remainder of the time before the due date will be used to prepare the final presentation.

1. **Sources**

Information on the math behind the algorithm:

<http://plus.maths.org/content/sundarams-sieve>

<http://en.wikipedia.org/wiki/Sieve_of_Sundaram>