Multiplication of Two Bignum Prime Numbers on a Nvidia Graphic Card

Johannes Häbe
jh-191149@hs-weingarten.de

Maximilian Nestle
mn-192181@hs-weingarten.de

Ravensburg-Weingarten University of Applied Sciences

June 5, 2020

Abstract

This paper is about testing the performance of the NVIDIA CUDA Fast Fourier Transform library (cuFFT) by multiplying bignum prime numbers on the graphics card. Several tests are presented, evaluated and compared.

1 Introduction

The fast multiplication of two large prime numbers is necessary in some procedures to hack asymmetrical encryption algorithms like the RSA encryption. These computations are normally made on the CPU. Within this paper, General Purpose Computing On GPUs (GPGPU) is used to multiply bignum prime numbers with the help of the NVIDIA CUDA Fast Fourier Transform library (cuFFT). For this purpose, the computing time of bignum multiplication on CPU and GPU is compared and evaluated in this paper.

2 Hardware and Environment

For the multiplications an Acer Aspire V3-772G is used. The Acer has got a NVIDIA GTX 850M graphics card and an Intel Core i5 4200M CPU. The graphics card disposes of 2004 MIB storage. The Acer is running the Linux distribution Ubuntu. The code that is used to multiply two bignums on the GPU is using the cuFFT library from NVIDIA. The multiplications on the CPU are done with the `BN_mul()` function which is based on the Karatsuba recursive multiplication algorithm. The `BN_mul()` function comes with the openssl library.

3 Comparison CPU with GPU

For the time measurement the C library function `clock(void)` is used. The legend of the graphics is explained in the following:

CPU

The time needed for the multiplications of the two bignums on the CPU using the `BN_mul()` function.

GPU_All

The sum of the times needed for GPU_Alloc, GPU_Calc, GPU_Clean, CUDA_Pre, and CUDA_Post.

GPU_Alloc

The time needed to allocate the amount of graphics card memory needed for the two bignums using `cudaMalloc()` and copying them to the graphics card memory by using `cudaMemcpy()`.

GPU_Calc

The amount of time needed for the calculation of the two bignums on the GPU. This includes converting the bignums to frequency domain, multiplying them with `ComplexPointwiseMulAndScale()` and converting them back to time domain.

GPU_Clean

`cufftDestroy()` `cudaFree()`

CUDA_Prep

CUDA_Post

4 Issues and Improvements

5 Conclusion

References