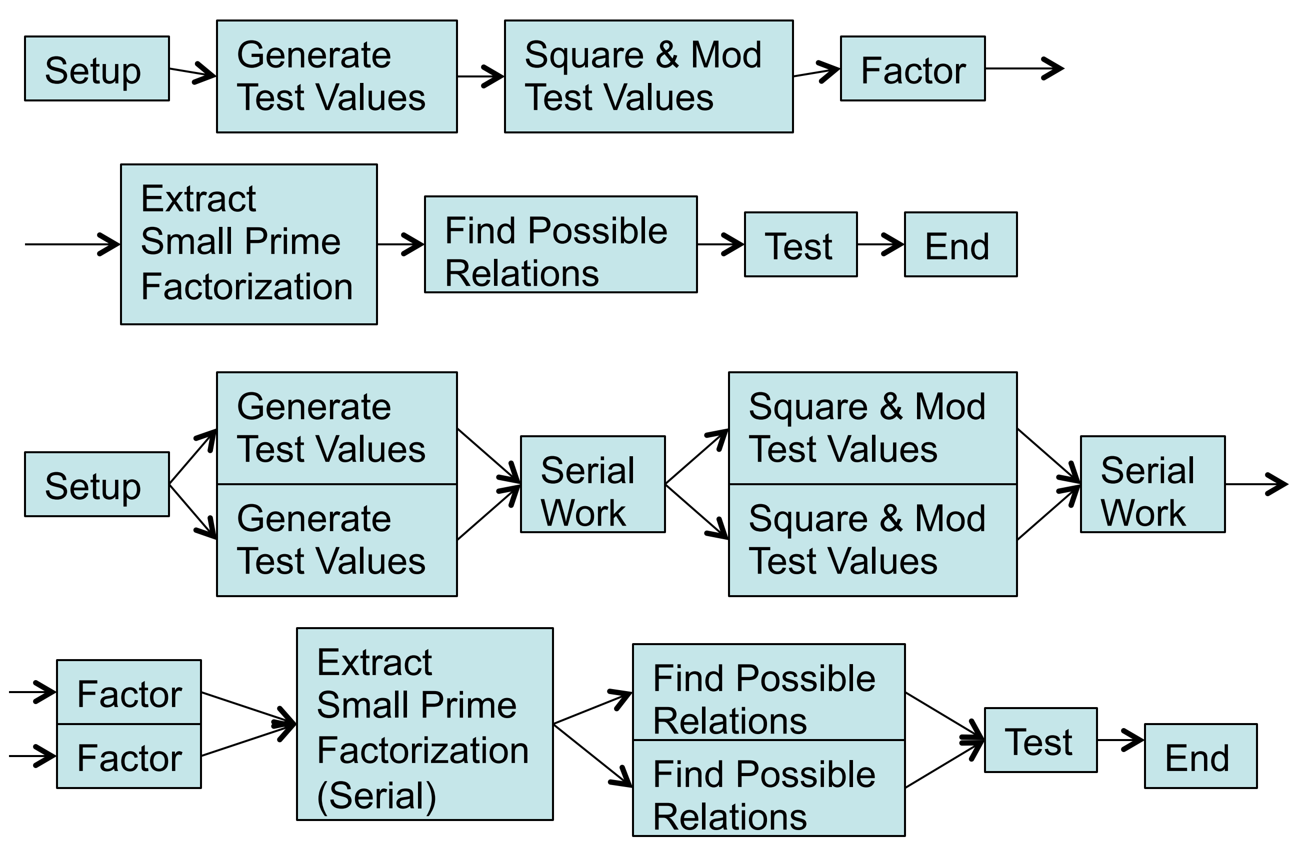
CPU/GPU Workload Distributions for Sieve Factoring Methods

Zachary Johnson   Salisbury University
Mentor: Dr. Don Spickler

Abstract

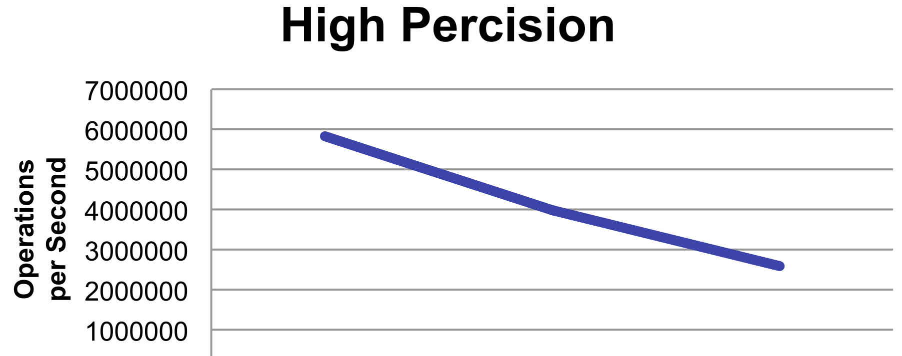
The Quadratic Sieve is used to increase the efficiency of factoring semi-primes that are usually used in cryptography. The algorithm for the sieve has many steps that can be parallelized. Due to this, efficiency can be increased if these steps are computed on the GPU.



Raw Speeds

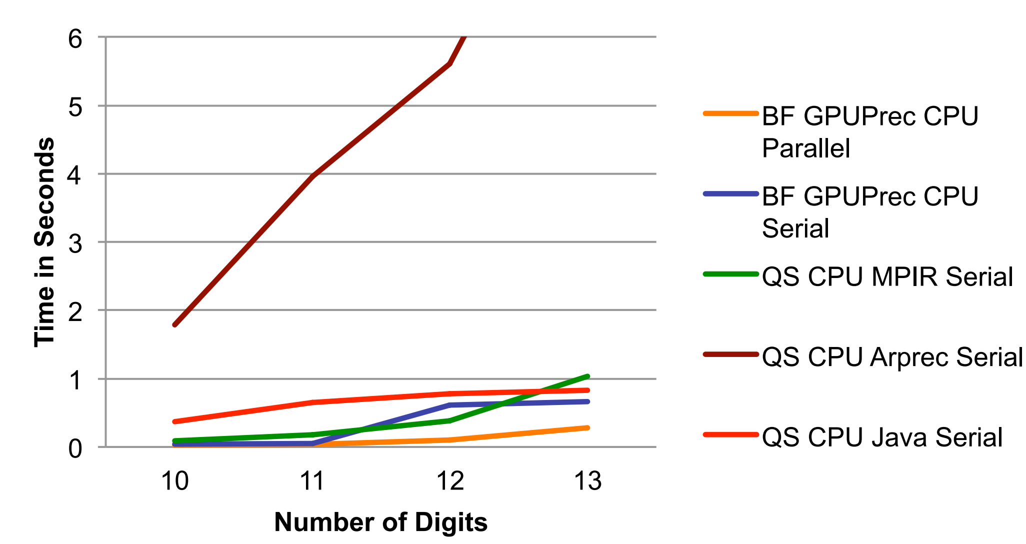
	Additions	Multiplications	Divisions
GPUPrec	153320.6179	8694.285281	7281.838309
Integers	12470424905	12484855489	9358545651
Long	12412168421	12302482887	9358541167
Float	12506417114	12443488914	10541142006
Double	6384091008	6493728873	6499560587
MPIR	5839187.357	3994582.448	2587401.584

High Percision

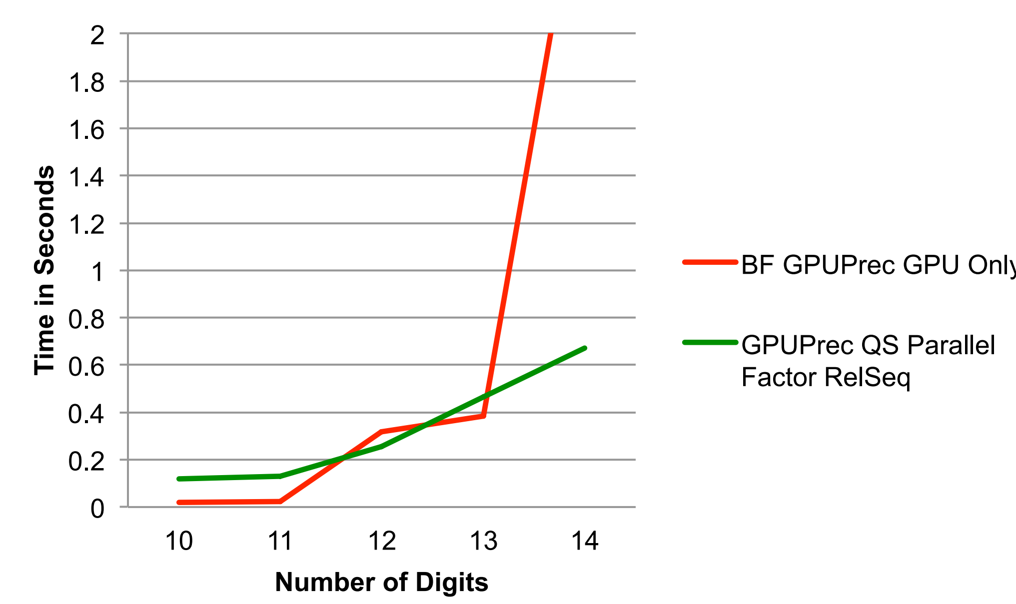


	Additions	Multiplications	Divisions
GPUPrec	153320.6179	8694.285281	7281.838309
MPIR	5839187.357	3994582.448	2587401.584

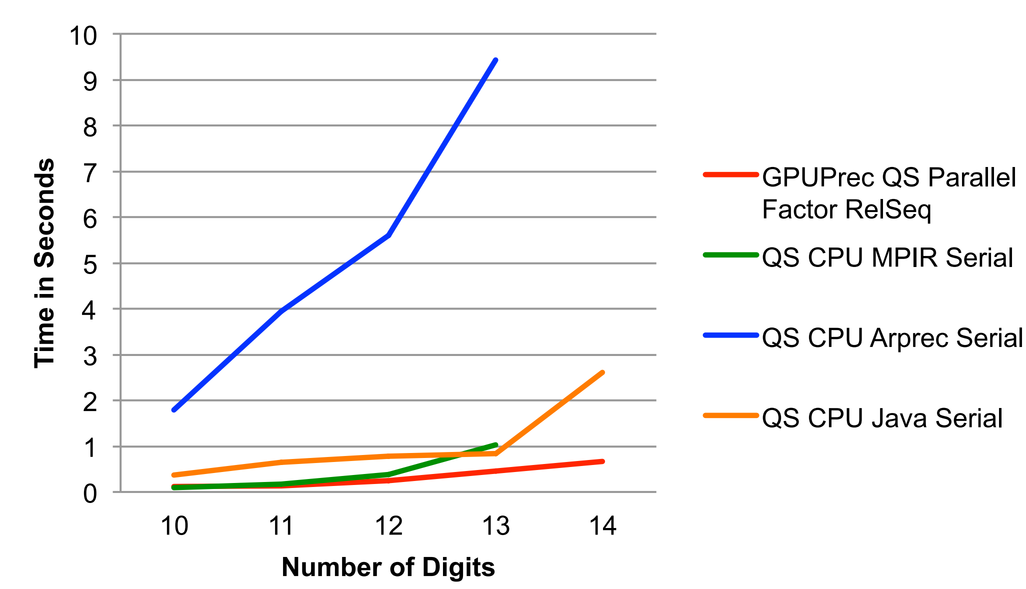
CPU



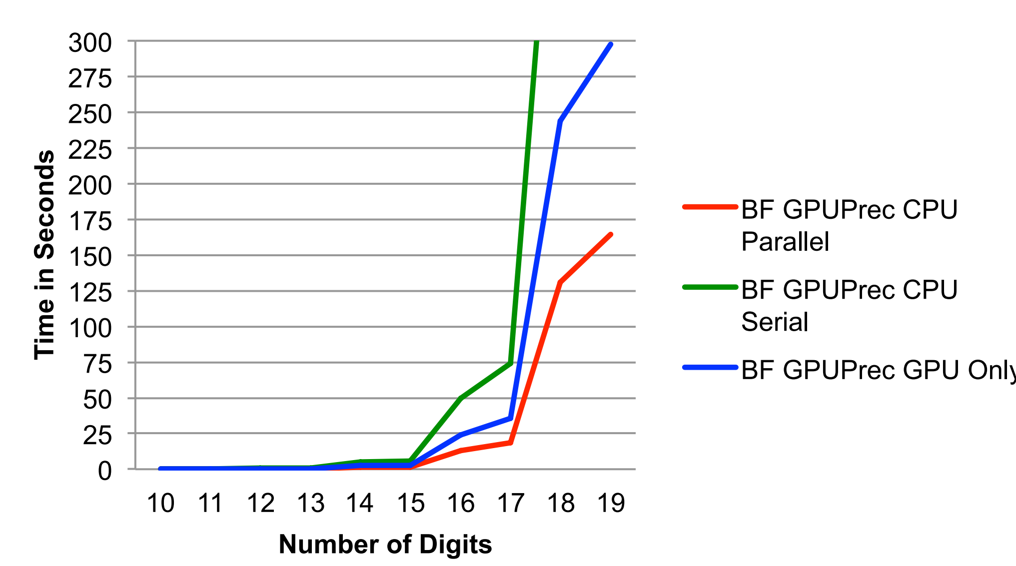
GPU



Quadratic Sieve



Brute Force



Goals

- Learn CUDA
- Implement the Quadratic Sieve on the CPU
- Implement the Quadratic Sieve on the GPU
- Implement the Quadratic Sieve as a combination of CPU and GPU

Results

Brute Force did jumps on time on even numbers when both factors had high digit counts. The Quadratic Sieve started slower because it had to generate test values then compare them. As the digits count increased the Quadratic Sieve became quicker compared to Brute Force.

Conclusions

Parallelizing the Quadratic Sieve on the GPU became faster as the digit count increased. Initially, the parallelized Brute Force on the CPU was faster since it takes time to transfer data to the GPU.

Future Work

- Generate a library for the GPUPrec code
- Continue to improve the efficiency of the Quadratic Sieve on the GPU
- Other factoring methods such as Elliptic Curve and other sieving methods

References

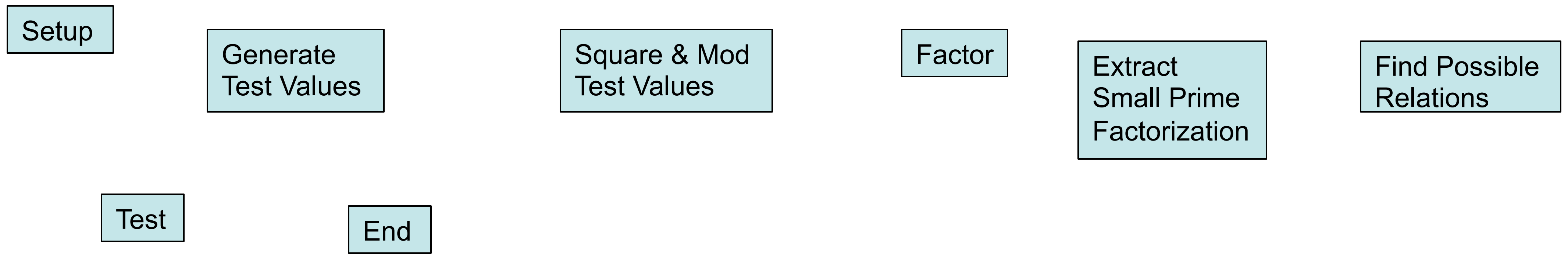
- GPUPrec Source Code and Documentation.
- MPIR Source Code and Documentation.
- Wade Trappe and Lawrence C.Washington, Introduction to Cryptography with Code Theory, 2 ed., Pearson Education, 2006.
- CUDA by Example: An Introduction to General-Purpose GPU Programming (29 July 2010) by Jason Sanders, Edward Kandrot

Algorithm

- $x^2 \cong y^2 \pmod{n}$
- $x \not\equiv y \pmod{n}$
- $GCD(x - y, n) = p$ so $p|n$

Procedure

- $x^2 = z \pmod{n}$ with $z = p_1^{\alpha 1} p_2^{\alpha 2} p_3^{\alpha 3} \ldots p_k^{\alpha k}$ such that $p_1, p_2, p_3, \ldots, p_k \leq$ Upper Prime Bound
- $Floor(\sqrt{(n*i)} + j) = x$
- i is moderate in size
- j is small in size ($1 \leq j \leq 100$)


Setup
Generate Test Values
Square & Mod Test Values
Factor
Extract Small Prime Factorization
Find Possible Relations
Test
End