

Performance Analysis of Parallelization of Multiple Fast Fourier Transforms Using OpenMP and CUDA

Holly K. Williams
University of Alabama
at Birmingham

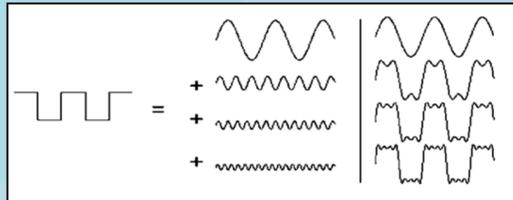
Research Alliance in Math and Science
Computer Science and Mathematics Division
<https://sites.google.com/site/hkwrams2011>

Mentor
Jacob Barhen
Oak Ridge
National Laboratory

Introduction

The Discrete Fourier Transform (DFT) decomposes a signal in the time domain into its frequency components and can be computed in $O(N^2)$ time. The Fast Fourier Transform (FFT) is an efficient algorithm that performs the DFT in $O(N \log_2 N)$ time. Many fields such as digital signal processing, image processing, and fluid dynamics employ this method.

The FFT of several vectors may require a large amount of computation time. FFT computation of multiple vectors can be performed in parallel reducing the execution time and allowing faster computations and applications that would be much slower or impossible without parallelism.



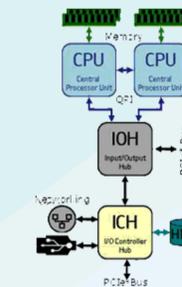
Research Objectives

- Use OpenMP and CUDA to parallelize the computation of multiple FFTs
 - Perform FFT computation using MKL library of vectors with 4K samples in parallel using shared memory and OpenMP on a multicore central processing unit (CPU)
 - Perform FFT computations in parallel using CUFFT library on a graphics processing unit (GPU)
- Gather and analyze performance results
 - Execution time
 - Speedup—speedup achieved with P threads $T(1) / T(P)$
 - Efficiency—determines efficiency of P threads based on the speedup and number of threads used $T(1) / (P \cdot T(P))$
 - Gflops— billions of floating point operations per second

Resources and Hardware

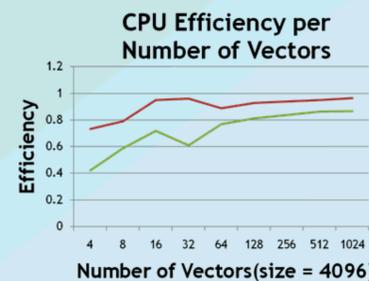
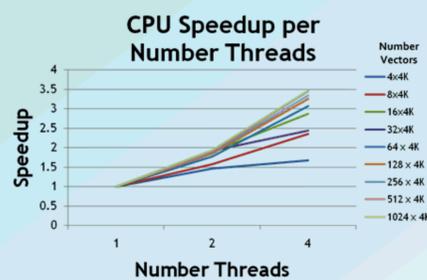
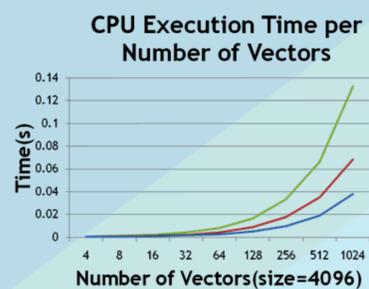
Intel Xeon E5530	
Cores	4
Clock speed	2.39GHz
Memory	6GB
Cache	8MB

NVIDIA Tesla C1060	
Cores	240
Clock speed	1.296 GHz
Memory	4GB



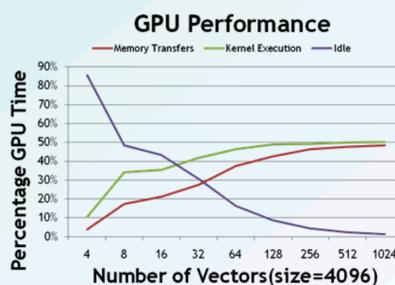
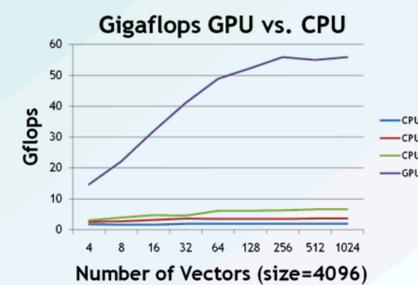
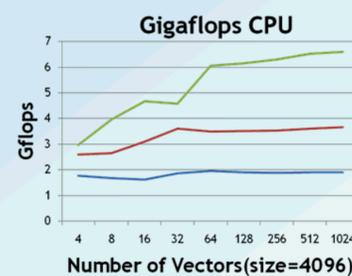
- NVIDIA Compute Visual Profiler
- PGI Visual Fortran Compiler
- Intel Math Kernel Library
- NVIDIA CUFFT library
- PGI CUDA Fortran

Results



- Using multiple threads does not decrease execution time significantly for smaller data sets
- Efficiency is almost one with larger datasets using two threads

Results



- Gigaflops for CPU are larger with larger number of vectors
- Gigaflops for GPU are much higher than CPU
- As data sizes increase, GPU spends as much time moving data as computing

Conclusion

Using multiple threads on the CPU decreases execution time. Efficiency and speedup results show that using multiple cores is beneficial. From the results, it can be seen that using multiple threads is not very efficient for small data sizes because the time to compute is small and thread creation adds overhead. As data sizes increase, using multiple threads is more useful.

Multicore computing does introduce speedup and reduced execution time, but the GPU implementation yields the best results with smaller execution times and more flops. Results show that the GPU spends time computing and also transferring data. With larger datasets, a higher percentage of time is spent transferring the larger data. Future work will involve using CUDA streams to compute the FFT.

References

Intel Xeon Processor E5530. ARK|Your Source for Information on Intel® Products
NVIDIA Tesla C1060 GPGPU Image Gallery. Rackmount Servers, Storage, and High Performance Computing | Silicon Mechanics.
Bores Signal Processing – Introduction to DSP – Frequency Analysis: Fourier Transforms. Bores Signal Processing – Training in DSP and Media Processing.

