GPU Accelerated Fast Fourier Transform

Matt McCarthy

Christopher Newport University
matthew.mccarthy.12@cnu.edu

December 2015

Abstract We empirically investigate the performance benefits of parallel fast Fourier transform running on the GPU over a sequential version running on the GPU.

1 Background

1.1 Discrete Fourier Transform

The discrete Fourier transform is a mathematical transformation that takes a set of Complex-valued signals and outputs a set of Complex-valued frequencies. For an n-dimensional Complex-valued vector $\mathbf{X}$, the discrete Fourier transform $\mathbf{Y} = \mathcal{F}(X)$ is given by

$$Y_j = \sum_{k=0}^{n} x_k \omega^{jk}$$

where ω is the n-th root of unity, $e^{2\pi i/n}$. Since $\mathbf{Y}$ is an n-dimensional, Complex-valued vector, we can see that the discrete Fourier transform has a complexity of $\Theta(n^2)$.

1.2 Fast Fourier Transform

Furthermore, we can split the discrete Fourier transform into even and odd sums for $n = 2m$, yielding

$$Y_j = \sum_{k=0}^{m} x_{2k} \omega^{2jk} + \omega^j \sum_{k=0}^{m} x_{2k+1} \omega^{2jk}$$

which is two seperate discrete Fourier transforms. Suppose $n = 2^k$. If we iterate this process, we get the following algorithm called the one-dimensional, unordered radix 2, fast Fourier transform.

function R-FFT($\mathbf{X}$,$\mathbf{Y}$,n,ω)
  if n=1 **then**
    $y_0 = x_0$
  else
    Let $\mathbf{Q} = \mathbf{0}, \mathbf{T} = \mathbf{0} \in \mathbb{C}^n$
    Let $\mathbf{X_e} = (x_0, x_2, \ldots, x_{n-2})$
    Let $\mathbf{X_o} = (x_1, x_3, \ldots, x_{n-1})$
    R-FFT($\mathbf{X_e}$,$\mathbf{Q_e}$,$n/2$,ω^2)
    R-FFT($\mathbf{X_o}$,$\mathbf{T_o}$,$n/2$,ω^2)
    for all $j \in \{0, 1, \ldots, n-1\}$ **do**
      $y_j = q_{j \mod n/2} + \omega^i t_{j \mod n/2}$
    end for
  end if
end function

1.2.1 Cooley Tukey

1.3 Parallelization

2 Experimental Design

3 Test Environment

3.1 Test System

3.2 Test Program

4 Results

4.1 Linear Speedup

5 Conclusion

Appendix