1D FFT Optimization on Machines with Memory Hierarchy

Wayne Anderson

(17/11/2023)

1. Introduction

Discrete Fourier transform (DFT) is widely used in aspects and fields, such as digital signal processing, computer vision, etc. However, the naive implementation from DFT has time complexity of $O(n^2)$, which is expensive for most of the computational devices for real-time processing. Fortunately, fast Fourier transform (FFT) is able to reduce the time complexity to $O(nlogn)$. Many famous algorithms, such as Cooley Tukey [1], StockHam [2], etc., are implemented.

However, the traditional FFT algorithms involve pitched memory access. Given a large discrete signal, the pitches are usually huge that forces the machines to load data from main memory each time instead of cache systems, wasting the advantages of caches. This article introduces a new method of dividing the process into multiple FFTs on much shorter signals, to optimize the memory access pattern on machines with memory hierarchy. It is also the implementation of FFT algorithms of DECX, a high-performance scientific computational library.

2. Method

Starting with the definition equation of discrete Fourier transform (DFT):

$$X(k) = \sum_{n=0}^{N-1} x(n) e^{-\frac{j2\pi nk}{N}} \tag{1}$$

Let $e^{-\frac{j2\pi nk}{N}} = W_N^{nk}$ and named as twiddle factor, equation can be written as:

$$X(k) = \sum_{n-0}^{N-1} x(n) W_N^{nk} \tag{2}$$

Traditional FFT can be performed to optimize this equation, reducing dozens of calculations. However, that involves pitched memory access, which fails to take the advantages of cached system. When the pitch is larger than L1 or L2 cache in CPU, accessing data needs transferring data between main memory and CPU for each time, wasting a lot of clock cycles.

The FFT process can be divided into multiple FFTs on much shorter data arrays [3]. Assuming that the signal length N can be divided into N_1 and N_2. In other words, $N = N_1 N_2$.

The equation can be written as:

$$X(k_1 + k_2 N_1) = \sum_{n_2=0}^{N_2-1} \sum_{n_2=0}^{N_1-1} x(n_2 + n_1 N_2) W_N^{(n_2+n_1N_2)(k_1+k_2N_1)} \tag{3}$$

The twiddle factor term can be simplified as

$$\begin{aligned} & W_N^{n_1N_2k_1+n_1N_2N_1k_2+n_2k_1+n_2k_2N_1} \\ &= W_N^{n_1N_2k_1} W_N^{n_1N_2N_1k_2} W_N^{n_2k_1} W_N^{n_2k_2N_1} \\ &= W_{N_1}^{n_1k_1} \cdot 1 \cdot W_N^{n_2k_1} W_{N_2}^{n_2k_2} \end{aligned} \tag{4}$$

Hence, the partitioned equation can be written as:

$$\begin{aligned} & X(k_1 + k_2 N_1) = \\ & \sum_{n_2=0}^{N_2-1} [(\sum_{n_1-1}^{N_1-1} x(n_1 N_2 + n_2) W_{N_1}^{n_1k_1}) W_N^{n_2k_1}] W_{N_2}^{n_2k_2} \\ &= \sum_{n_2=0}^{N_2-1} [(\sum_{n_1-1}^{N_1-1} x_{n_2}(n_1) W_{N_1}^{n_1k_1}) W_N^{n_2k_1}] W_{N_2}^{n_2k_2} \end{aligned} \tag{5}$$

Let

$$X_{n_2}(k_1) = \sum_{n_1-1}^{N_1-1} x_{n_2}(n_1) W_{N_1}^{n_1k_1} \tag{6}$$

Hence, equation (5) can be represented as:

$$X(k_1 + k_2 N_1) = \sum_{n_2=0}^{N_2-1} (X_{n_2}(k_1)\, W_N^{n_2k_1}) W_{N_2}^{n_2k_2} \tag{7}$$

Equation (6) indicates multiple N_1 point FFTs performing independently and parallelly, which is called stage 1 of the process.

N_1 and N_2 are small enough to fit in the cache of the CPUs. Therefore, each N_1 point FFT or N_2 point FFT is able to be performed within cache, without worrying about any cache miss caused by pitched memory access. In this context, traditional FFT algorithms like Cooley Tukey and StockHam can be considered.

The data access pattern can be illustrated as figure 1. Stage 1 shows the data flow pattern of rearranging the signal vector and performing the multiple-lane FFTs.

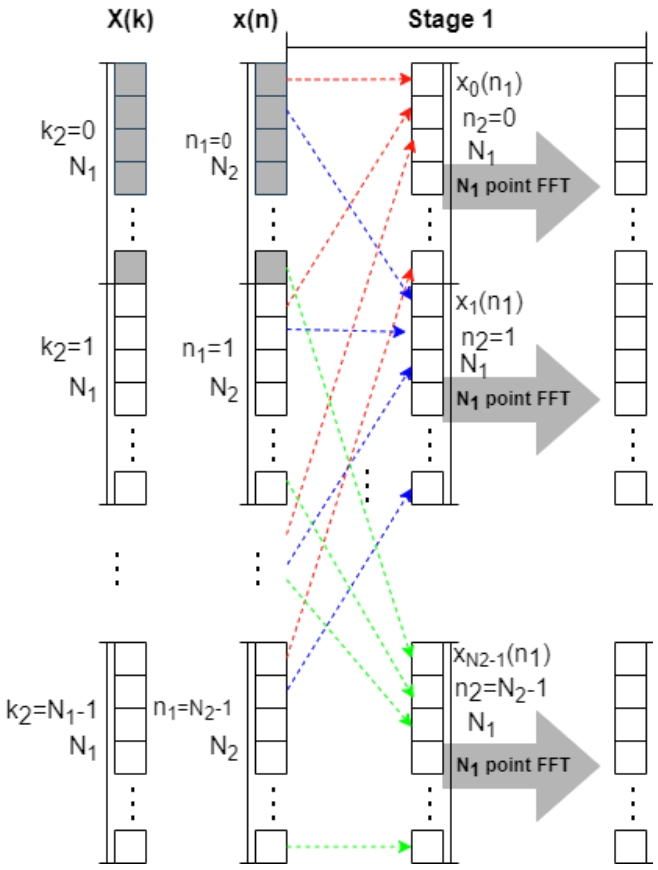


Figure 1 Data access pattern of stage 1

Similarly, stage 2 represents multiple-lane N_2-point FFT based on $x_{n_2}(k_1)W_N^{n_2k_1}$. $W_N^{n_2k_1}$ is noticeable in equation (7). In the gap of each stage, an additional group of twiddle factors should be multiplied with the result of the last stage.

The distribution of twiddle factors can be shown as figure 2.

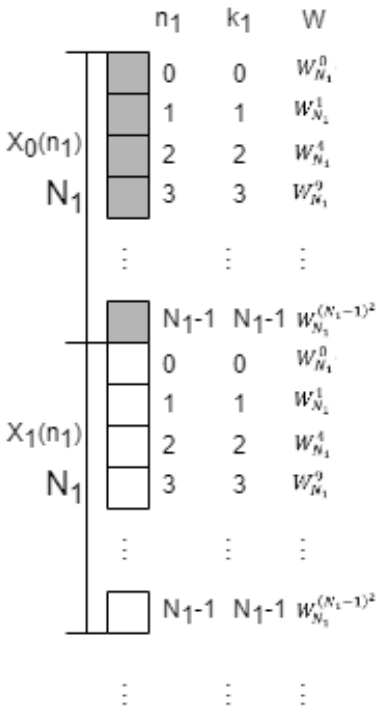
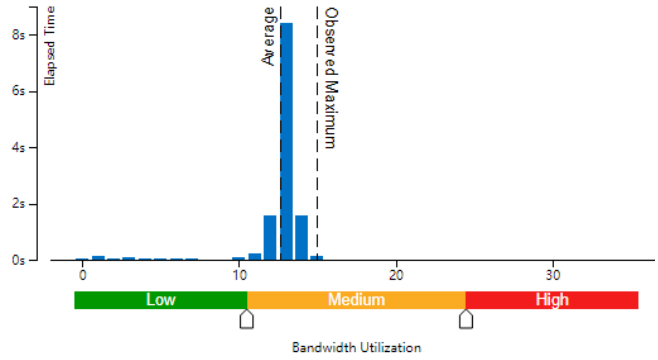


Figure 2 Twiddle factors

3. Performance

The design of optimal 1D FFT algorithms are implemented and available in DECX. The test is based on a PC equipped with an Intel® x86 architecture CPU, Intel Core I7 9th generation, and a DDR4 based DRAM, with theoretical bandwidth of 35.0 GB/s. The operating system is Microsoft® Windows 10. The profile data was obtained from Intel® VTune Profiler, profiling on local machine.

The test transformed 1024×1024 point FFT 1000 times based on a 32-bit floating point real signal using APIs provided by DECX. In the internal design of DECX digital signal processing (DSP) module, the maximum partition length is set to 1024. Hence, in this case, both N_1 and N_2 reached their maximum of 1024 points. The percentage of memory bound, bandwidth utilization, and latency histogram is shown in figure 3.



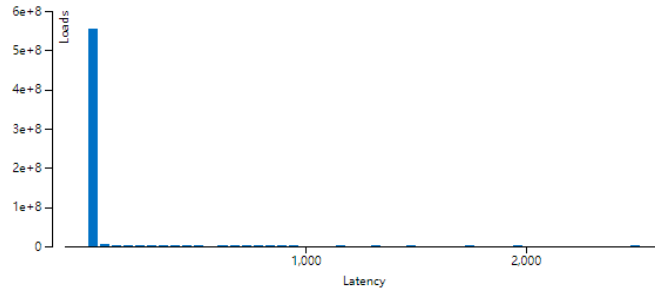
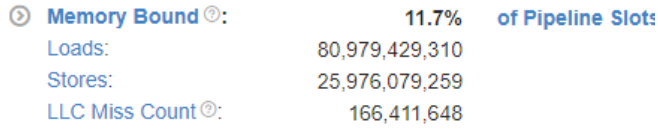
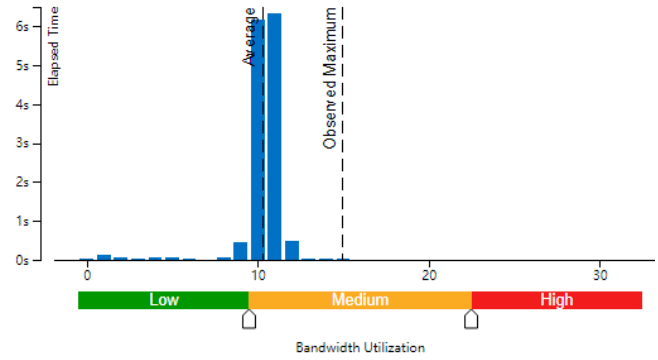


Figure 3 Profiling data of 1024×1024 point FFT

The memory bound, especially L1 bound is still observed a bit high. Since both N_1 and N_2 had reached 1024, imposing huge pressure to L1 cache.

In test 2, the length of the signal is changed to $625 \times 625 \times 3$. In this case, the signal was divided into 3 parts, with lengths $N_1 = 625, N_2 = 625, N_3 = 3$, reducing occupancy of L1 cache. Expectedly, the result became better, shown in figure 4.





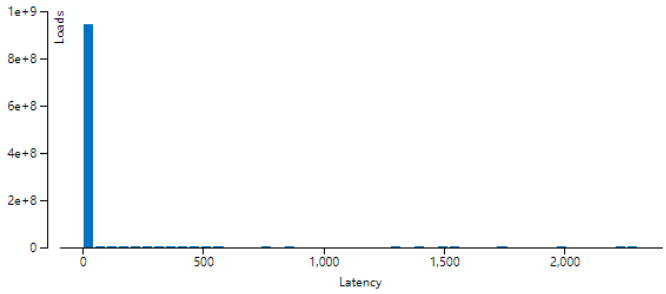


Figure 4 Profiling data of $625 \times 625 \times 3$ point FFT

References

[1] J. Kim, J. Lee, and S. Kim, "Stockham FFT Acceleration with Processing-in-Memory," in Proc. ITC-CSCC 2019, Jeju, Korea, Jul. 2019, pp. 1-4.

[2] F. Franchetti and M. Püschel, "FFT (Fast Fourier Transform)," in Encyclopedia of Parallel Computing, D. Padua, Ed. New York, NY, USA: Springer, 2011, pp. 844-854, doi: 10.1007/978-0-387-09766-4_243.

[3] Y. Zhao, Y. Ao, C. Yang, F. Liu, W. Yin, and R. Lin, "General Implementation of 1-D FFT on the Sunway 26010 Processor," J. Softw., vol. 31, no. 10, pp. 3184-3196, Oct. 2020, doi: 10.13328/j.cnki.jos.005848.