Proceedings of Parallel Scientific Computing
D5130
Jan-May, 2021, IIT Madras, Chennai

PARALLELIZATION OF DFT FOR GENOMIC SIGNAL PROCESSING

Shreya Nema
BS17B033
Department of Biotechnology
Indian Institute of Technology Madras
Email: bs17b033@smail.iitm.ac.in

ABSTRACT

Genomic Signal Processing (GSP) refers to Digital Signal Processing (DSP) tools for analyzing genomic data such as DNA sequences. Genomic signal processing has a wide range of applications in comparative genomics, gene prediction, gene regulatory networks analysis, metagenomics, clustering, and feature selection & error estimation for classification [1] [2] [3]. DNA sequence comprises four nucleotides, namely, Adenine (A), Thymine(T), Guanine (G), and Cytosine (C) [4]. The string representing a DNA sequence can be further treated as a discrete signal wherein each nucleotide is assigned a numerical value using different encoding methods. These numerical representation methods include Chaos Game Representation (CGR), Purine-Pyrimidine (PP), Just-A, Just-C, Just-G, Just-T., etc.[2]. Frequency analysis of discrete genomic signals can then be performed using the Discrete Fourier Transform (DFT) to get magnitude spectra corresponding to each sequence. The straightforward implementation of the DFT is computationally less efficient. The use of faster DFT implementations like N/2 point DFT, FFT etc has been extensively covered in literature, with several pieces of research exploring its optimization and parallelization [5] [6] [7]. A genome processing algorithm based on the FFT can therefore yield good results.

Existing solutions in the literature are often limited to small DNA sequences of a few thousand base pairs. A parallel implementation of DFT can effectively process large sequences (complete genomes). Different numerical representations (CGR, PP, Just-A, etc.) can be used for genome sequences to convert them to discrete signals. A serial DFT is be applied on the discrete signals of large size to get the corresponding magnitude spectra in the first experiment. In the second experiment, a parallel code for N/2 point DFT using MPI and OpenMP+MPI is implemented, followed by a comparative analysis of the time taken by the two methods. The objective of this project is to make GSP fast, reliable and accurate such that it can work for all types of genomes.

INTRODUCTION

Alignment-based methods for DNA comparison have been used successfully, but these methods have certain limitations. First, the alignment-based methods assume that homologous sequences consist of linearly arranged and conserved stretches of sequences. However, this assumption of collinearity doesn't hold in the real world, especially in viral genomes. Viral genomes have high mutation rates, gene duplication/repeats, gene deletion, and frequent genetic recombination events. Second, when the sequence identity falls, the accuracy drops rapidly. Third, the alignment-based approaches are generally computationally expensive and have a time-complexity of the order of the product of the lengths of input sequences. Fourth, the multiple sequence alignment problem is an NP-hard problem, and finally, such alignments depend on many apriori assumptions about the sequence evolutionary trajectory. On the other hand, alignment-free approaches do not rely on sequence alignment and are computationally less expensive. These methods are resilient to recombination and shuffling events and can perform equally well on genomes with less sequence conservation.

Alignment-free methods have been proven successful and sufficient and they overcome all the challenges that the alignment-based techniques fail to address. A practical

approach is to use the Discrete Fourier Transform (DFT) for DNA comparison [4]. DFT is a well-established method in the fields like speech signal processing and image processing. The genomic signals obtained from DFT can then be processed using parametric and non-parametric approaches to perform a comparative genomic analysis. The genomic sequences, when numerically represented as discrete numerical sequences, can be treated as digital signals [8]. This process of applying DSP techniques on DNA sequences is termed Genomic Signal Processing(GSP) [9]. GSP has a wide range of applications ranging from classification of coding and non-coding regions in a genomic sequence [10-11], classification of biological sequences[2, 12], clustering[13], gene regulatory networks (control, inference and analysis) to proteomics. These techniques can be utilized for functional understanding of the disease and development of systems-based medical solutions and predict the origin of species and upcoming viral mutants potentially causing pandemics. DFT maps sequence over time to another sequence over frequency. However, the straightforward implementation of DFT is practically costly, having a time complexity of $O(n^2)$.

Fourier analysis is the representation of a continuous function by an infinite series of sin and cosine functions by making use of their orthogonality. The Fourier series is a function that can be expressed as the sum of complex exponentials (series of sins and cosines). A Fourier series can be represented as:

$$f(x) = \tfrac{1}{2}a_0 + \sum_{n=1}^{\infty} a_n \cos(nx) + \sum_{n=1}^{\infty} b_n \sin(nx)$$

where, n = 1,2,3…

$$a_o = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)dx$$

$$a_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\cos(nx)dx$$

$$b_n = \frac{1}{\pi}\int_{-\pi}^{\pi} f(x)\sin(nx)dx$$

Here, a_n and b_n are called Fourier coefficients of f. The infinite sum $f(x)$ is called the Fourier series of f.

FOURIER TRANSFORM

Fourier transform, unlike Fourier series is used to express non-periodic function by a continuous superposition or integral of complex exponentials. The Fourier Transform of $f(x)$ is defined as:

$$F(x) = \int_{-\infty}^{\infty} f(x)e^{-2\pi ikx}dx$$

where,

$$e^{xi} = \cos(x) + i * \sin(x)$$

$f(x)$ is a function of the variable time and $F(X)$ is a function of the variable frequency. Hence Fourier transform maps the time domain series into the frequency domain series.

DISCRETE FOURIER TRANSFOR

Discrete Fourier Transform (DFT) is a specific kind of Fourier transform where the input to the function is discrete values. It requires a discrete input function that can be a finite sequence of real or complex numbers. This makes DFT ideal for information processing on computers. [14]

DFT is one of the important methods used in digital signal processing. DFT, when applied on a temporal signal, gives its complex-valued frequency domain representation. The magnitude spectrum of the DFT of a signal shows each frequency component's power in the signal. The formula to calculate the N-point DFT of a discrete signal $x = \{x_0, x_1, x_2, \ldots, x_{N-1}\}$ is:

$$X_k = \sum_{n=0}^{N-1} x(n) * e^{\frac{-2\pi i}{N}kn}$$

where, $k \in [0, N-1]$, X_k is a complex number that contains the information of the phase and amplitude of a sinusoidal wave with frequency k/N cycles per unit time.

The time complexity of DFT for length n sequence is $O(n^2)$. Hence with a large N, using DFT might not be the best practice. We can use the divide and conquer strategy which gives the same result as DFT but improved performance. The DFT sequence of N samples can be divided into two subsequences of even and odd indices. This method is also called N/2 point DFT.

RADIX-2 DECIMAL-IN-TIME ALGORITHM

The radix-2 decimation-in-time algorithm [15] rearranges the discrete Fourier transform (DFT) equation into two parts: a

sum over the even-numbered discrete-time indices n=[0,2,4,…,N−2] and a sum over the odd-numbered indices n=[1,3,5,…,N−1] as in Equation:

$$X_k = \sum_{n=0}^{N-1} x(n) * e^{\frac{-2\pi i}{N} kn}$$

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x(2n) * e^{\frac{-2\pi i}{N} k(2n)} + \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) * e^{\frac{-2\pi i}{N} k(2n+1)}$$

$$X_k = \sum_{n=0}^{\frac{N}{2}-1} x(2n) * e^{\frac{-2\pi i}{\frac{N}{2}} kn} + e^{\frac{-2\pi i}{\frac{N}{2}} k} * \sum_{n=0}^{\frac{N}{2}-1} x(2n+1) * e^{\frac{-2\pi i}{\frac{N}{2}} kn}$$

$$X_k = DFT_{\frac{N}{2}}[x(0), x(2), ..., x(N-2)] + \omega_N{}^k * DFT_{\frac{N}{2}}[x(1), x(2), ..., x(N-1)]$$

where $\omega_N{}^k = e^{\frac{-2\pi i}{\frac{N}{2}} k}$

$X(k)$ can be computed as the sum of the outputs of two length-$\frac{N}{2}$ DFTs, of the even-indexed and odd-indexed discrete-time samples multiplied by $\omega_N{}^k = e^{\frac{-2\pi i}{\frac{N}{2}} k}$.

Because the time samples are rearranged in alternating groups of odd and even indices, this algorithm is called decimation in time algorithm and a radix-2 algorithm because there are two groups.

Because of the periodicity with N/2 frequency samples, the DFT of even and odd indexed time samples can be used to compute X(k) and X(k+N/2), but with a different twiddle factor ($\omega_N{}^k$). This reuse of these short-length DFT outputs reduces the time complexity by a factor of 2. The new time complexity of this algorithm becomes $O(\frac{N^2}{2})$.

NUMERICAL ENCODING OF DNA SEQUENCE

The two DNA sequences used for the project are Bat SARS-like coronavirus (CoVZC45) and Coliphage phi-X174 (Coliphage) of length 29802 and 5386 base pair (bp) respectively.

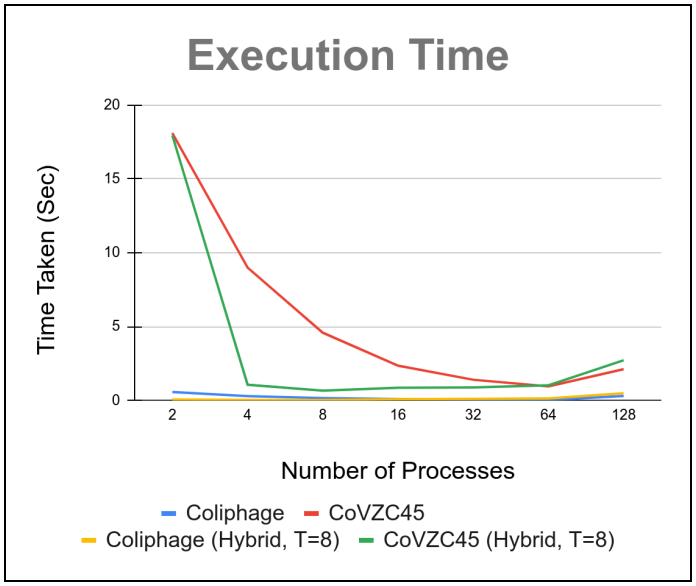
To apply digital signal processing techniques to genomic data, first, we need to map genomic sequences to some discrete numerical representations [2,8].

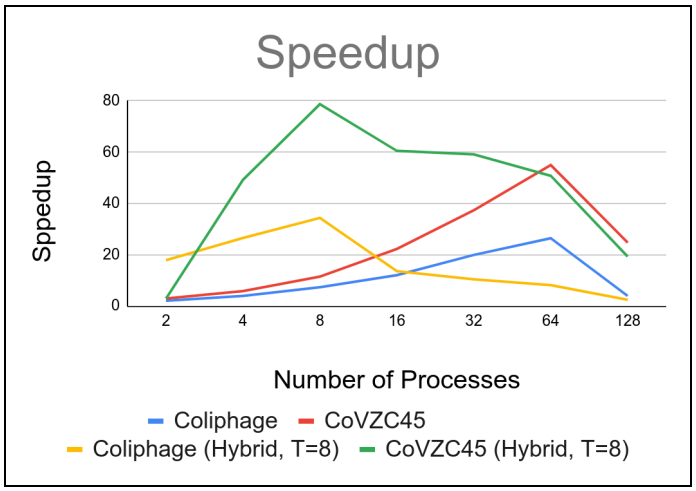
Different DNA sequence numerical encoding approaches, as shown in the Table, can be used to produce an indicator sequence (discrete numerical representation) for a given DNA sequence.

PARALLEL IMPLEMENTATION WITH PURE MPI

Each process controls N/p elements of numerically encoded input DNA sequence. After processing even and odd elements separately the final even indexed DFT and odd indexed DFT are obtained by summing even and odd part results from all the processes. Hence the overall time complexity of parallel implementation with pure MPI becomes $O(\frac{N^2}{2*p^2})$ [16].

Numerical Representation	Rule
Purine/Pyrimidine (PP)	A, G = -1; T, C = 1
Just-A	A = 1; T, G, C = 0
Just-T	T = 1; A, G, C = 0
Just-G	G = 1; T, A, C = 0
Just-C	C = 1; T, G, A = 0
Chaos Game Representation (CGR)	Corners of the square labelled as ‘A’, ‘T’, ‘G’ and ‘C’. Use the next base to pick the next point on the plane, halfway between the current point and the next base corner.





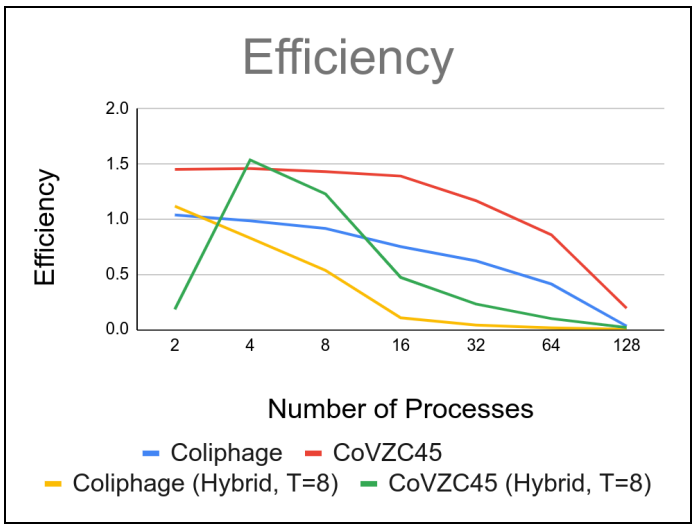


Fig: Plot of Execution time, speedup, and efficiency for Coliphage and CoVZC45 sequences processed using pure MPI and using Hybrid parallelization approach with fixed 8 threads per node.

PARALLEL IMPLEMENTATION WITH MPI+OPENMP HYBRID

As there is no MPI call within the parallel regions of the code, there is no overlap of communication. Hence we can use hybrid parallel programming using MPI + OpenMP, such that MPI is responsible for inter-node communications and OpenMP inside each SMP (symmetric multiprocessing) node parallelizes for loops (Fig). In MPI+threads hybrid programming, there can be multiple threads executing simultaneously. All these threads share all MPI objects.

The hybrid parallelization of DFT further reduces the time complexity of the algorithm by a factor of T (thread count). Hence, for hybrid implementation, the overall time complexity becomes $O(\frac{N^2}{2*p^2*T^2})$.

PERFORMANCE ANALYSIS

A performance analysis was performed to display execution time, speed-up, and efficiency. The code records the time at each iteration of the DFT algorithm, for both parallelized and sequential implementations. An average of multiple iterations is taken to get an average estimate of execution time. Further, to measure the relative performance of serial and parallel codes, the speedup is calculated [16]. Speed up is a measure of the relation between the serial and the parallel run-times and it is just the ratio of the serial run-time to the parallel run-time, defined as:

$$S(n,p) = \frac{T_{serial}(n)}{T_{parallel}(n,p)}$$

Ideally, the speedup should be equal to p (number of processes). Another measure of parallel performance is efficiency, which can be defined as speedup per processor [16]. Efficiency measure is defined as:

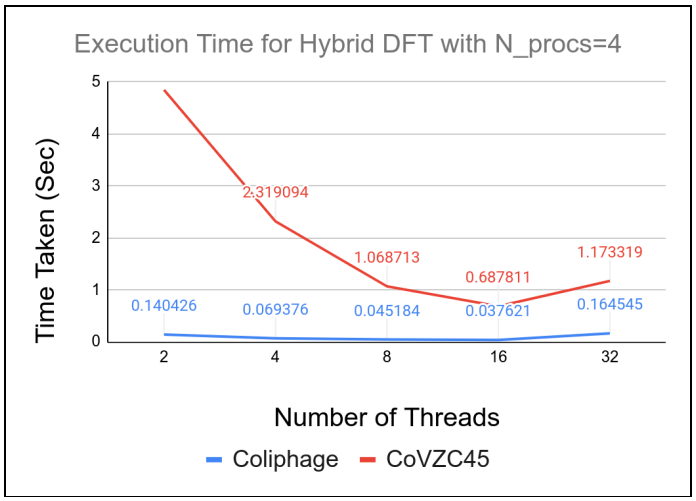
$$E(n,p) = \frac{S(n,p)}{p} = \frac{T_{serial}(n)}{p*T_{parallel}(n,p)}$$

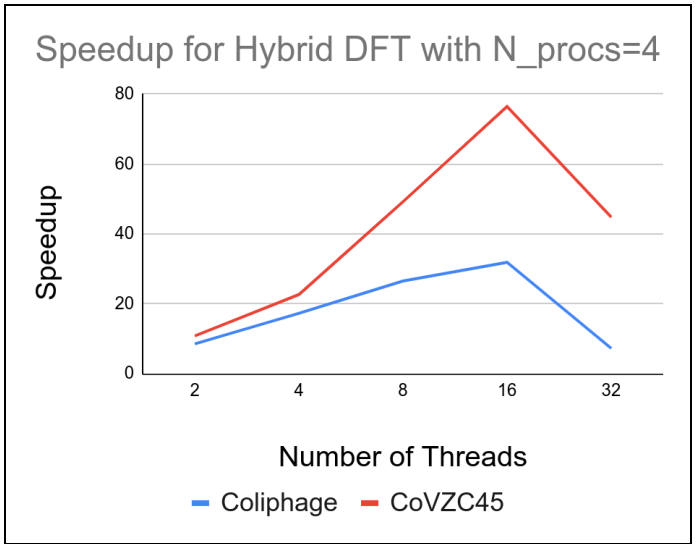
RESULT AND DISCUSSION

The motivation to use DFT based alignment-free approaches for comparative genomics comes from the fact that alignment-free approaches have an added advantage over alignment-based approaches. They do not require any

specialized biological knowledge, genome annotation, or training and are more time-efficient than alignment-based approaches. To apply DFT to DNA sequence we first need to encode DNA sequences into numerical representations. The "Purine-Pyrimidine" representation is used for this project.

To speed up the DFT application on discrete genomic sequence, N/2 point DFT (Radix-2 Decimal in Time Algorithm), is used for both serial and parallel programs. The algorithm is further parallelized using pure MPI and hybrid MPI+OpenMP, to enhance the execution speed. To compare and analyse the performance of serial and parallel codes two genomic sequences are picked. One with a large genomic size of around 29802 base pairs (Bat SARS-like coronavirus) and the other with a smaller genome size of 5386 base pairs (Coliphage phi-X174).





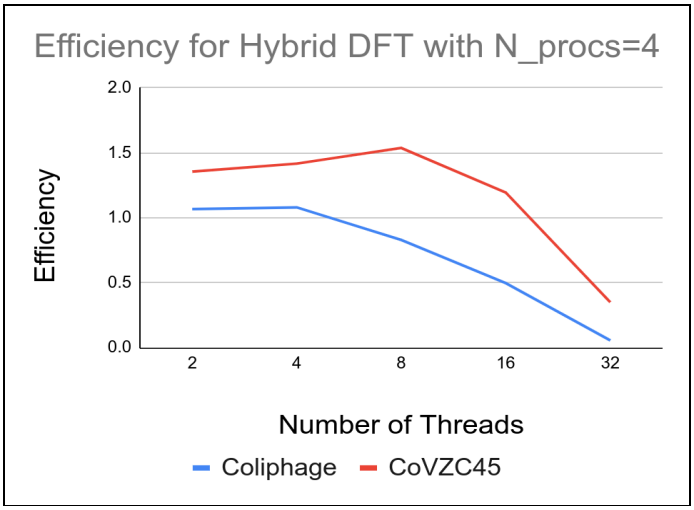


Fig: Plot of Execution time, Speedup, and Efficiency for Coliphage and CoVZC45 sequences processed using Hybrid parallelization approach with fixed processor count of 4.

The serial implementation of N/2 point DFT takes 1.199311 sec for Coliphage and 52.591436 sec for CoVZC45. The hybrid DFT parallelization approach performs best with significant speedup for both the genomic sequences. The execution time decreases drastically with parallelization.

For pure MPI based parallelization, the execution time decreases with an increase in the number of processes. The speedup increases with an increase in the number of processes and is greatest using 64 processes. The exception is when the number of processors is > 64. This might be because of increased internode communication and a relatively smaller sample size. As the number of processes increases the speedup deviates from its ideal value of p (number of processes).

For the hybrid parallel program, the highest speedup is obtained with processes count and thread count 8 each. This suggests the best configuration for the hybrid program. For a fixed process count of 4, the least execution time and the highest speedup is obtained with a thread count of 16.

As the number of processes increases the efficiency of parallelization decreases. Efficiency decreases because each process is now spending more time waiting and communicating than computing. If the problem size was bigger then efficiency would increase due to each process spending more time computing than waiting. This might also be the reason for an efficiency greater than 1 for a lower processes count. With increasing problem size, the memory requirement will also increase.

The N/2 point algorithm can be further expanded to implement Fast Fourier Transform (FFT). The same Radix-2

Decimal in Time algorithm can be applied recursively to $\frac{N}{2}$ length DFT at each iteration until we reach length-2 DFT.

ACKNOWLEDGMENTS

Thanks to Prof. Kameswararao Anupindi and Prof. Rupesh Nasre for their valuable guidance. I appreciate the organizations: GISAID, and NCBI, for making the data available free of cost that made this project possible.

REFERENCES

[1] Borrayo, Ernesto, et al. ”Genomic signal processing methods for computation of alignment-free distances from DNA sequences.” PloS one 9.11(2014): e110954.

[2] Randhawa, Gurjit S., Kathleen A. Hill, and Lila Kari. ”ML-DSP: MachineLearning with Digital Signal Processing for ultrafast, accurate, and scalable genome classification at all taxonomic levels.” BMC Genomics 20.1 (2019):267.

[3] Rivard, Sylvain Robert, et al. ”Design of high-performance parallelized gene predictors in MATLAB.” BMC research notes 5.1 (2012): 1-10.

[4] Cheever, E. A., et al. ”Using signal processing techniques for DNA sequence comparison.” Proceedings of the Fifteenth Annual Northeast Bioengineering Conference. IEEE, 1989.

[5] Takahashi, Daisuke. Fast Fourier Transform Algorithms for Parallel Com-puters. Springer Singapore, 2019.

[6] Plimpton, Steve, Roy Pollock, and Mark Stevens. ”Particle-Mesh Ewaldand rRESPA for Parallel Molecular Dynamics Simulations.” PPSC. 1997.

[7] Arbenz, Peter. Introduction to Parallel Computing: A Practical Guide with examples in C. Oxford Texts in Applied and Engineering Mathematics.Oxford University Press, USA, 2004.

[8] Kwan HK, Arniker SB. Numerical representation of DNA sequences. In2009IEEE International Conference on Electro/Information Technology 2009 Jun 7(pp. 307-310). IEEE.

[9] Anastassiou D. Genomic signal processing. IEEE signal processing magazine. 2001Jul;18(4):8-20.

[10] Roy M, Biswas S, Barman S. Identification and analysis of coding and non-coding regions of a DNA sequence by the positional frequency distribution of nucleotides(PFDN) algorithm. In2009 4th International Conference on Computers andDevices for Communication (CODEC) 2009 Dec 14 (pp. 1-4). IEEE.

[11] Lorenzo-Ginori JV, Rodr ́ıguez-Fuentes A, Abalo RG, Rodriguez RS. Digital signal processing in the analysis of genomic sequences. Current Bioinformatics.2009 Jan 1;4(1):28-40.

[12] Akhtar M, Ambikairajah E, Epps J. GMM-based classification of genomic sequences. In2007 15th International Conference on Digital Signal Processing2007 Jul 1 (pp. 103-106). IEEE.

[13] Mendizabal-Ruiz G, Rom ́an-God ́ınez I, Torres-Ramos S, Salido-Ruiz RA,V ́elez-P ́erez H, Morales JA. Genomic signal processing for DNA sequenceclustering. PeerJ. 2018 Jan 24;6:e4264.

[14] Smith, Julius O. "Mathematics of the discrete fourier transform (dft)." Center for Computer Research in Music and Acoustics (CCRMA), Department of Music, Stanford University, Stanford, California (2002).

[15] Montoya-Andrade, D. A., J. A. Rosendo-Macías, and A. Gómez-Expósito. "Efficient computation of the short-time DFT based on a modified radix-2 decimation-in-frequency algorithm." *Signal processing* 92.10 (2012): 2525-2531.

[16] Pacheco, Peter. An introduction to parallel programming. Elsevier, 2011.