Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN 3007-3138

Print ISSN 3007-312X

Investigating the Scalability of FFT Algorithms in Contemporary Parallel Computing Environments

Salahuddin[1*]
Department of Computer Science, NFC Institute of Engineering and Technology, Multan, Pakistan.
Corresponding Author: Email: msalahuddin8612@gmail.com

Muhammad Faran Aslam[2]
Department of Artificial Intelligence, School of Systems and Technology, University of Management and Technology, Lahore, Pakistan

Ayesha Yasin[3]
Department of Computer Science, School of Systems and Technology, University of Management and Technology, Lahore, Pakistan

Meiraj Aslam[4]
Department of Computer Science, NFC Institute of Engineering and technology, Multan, Pakistan.

Assad Latif[5]
School of Management and Engineering, North China University of Water Resources and Electric Power Zhengzhou Henan, China

Abstract

Parallel programming models are quite challenging and emerging topic in the parallel computing era. These models allow a developer to port a sequential application onto a platform with number of processors so that the problem or application can be figured out easily. Adapting the applications in this mode using the Parallel programming models is often influenced by the type of the application, the type of the platform and many others. There are several parallel programming models developed and two main

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

variants of parallel programming models classified are shared and distributed memory based parallel programming models. This thesis compares various techniques for the fast evaluation of Fast Fourier transform on parallel machines. In this work we present a model covering the essential features of communication systems for discussing and comparing their operational semantics. Our access is based on parallel FFT algorithms. Currently, many cores are the most suitable for the deployment of HPC (High performance Computing) infrastructures, due to their performance over cost ratio and scalability. These systems can be programmed using OpenMP(For Shared memory) and MPI(For distributed memory) and their hybrid model MPI+OpenMP (for cluster of shared memory) and the recent variation PGAS (Partitioned Global Address Space) languages, such as UPC (Unified parallel C), promises more productivity and execution, providing support for shared, distributed and their hybrid model in efficient manners. PGAS languages demonstrate very little operating cost as compared with MPI for problems that are inadequately parallel. Evaluation of sequential and MP based implementation of FFT is desirable, because FFT is one of the seven benchmarks for measuring performance of HPCC.

Keywords: Unified parallel C, Parallel programming, Partitioned Global Address Space, Shared memory

Introduction

The Fourier transform has long been used as an important analytical tool in many fields of scince and engineering . The advent of digital computers provided the fast mean to compute the discrete Fourier transform (DFT) on one hand and the fast Fourier transform algorithm boosted this speed by reducing the number of operations required to compute the DFT on the other hand. While the FFT algorithm transform the data from time domain to frequency domain and vice versa in O(N2), where FFT save great time by reducing complexity to O(N lg N)[1]. The FFT

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN
3007-3138
Print ISSN
3007-312X

(Fast Fourier Transform) was developed in 1965, widely used in many field of science and engineering, is considered as one of the most prolific and useful algorithms of the last century. Having reached a linit with a serial computer can compute the FFT , one is naturally led to think of some other means of fast computation of FFT [1]. The concept of computing FFT in Parallel for time saving is not new and can be trace back in 80's. The current architecture of computing machine many core clusters that need parallelism. While current and conventional parallel programming paradigm supporting these proposed systems more efficiently. Currently many core are the most suitable for the deployment of HPC (High performance Computing) infrastructures, due to their performance over cost ratio and scalability. These system can be programmed using OpenMP(For Shared memory) and MPI(For distributed memory) and their hybrid model MPI+OpenMP [2](for cluster of shared memory) and the recent variation PGAS (Partitioned Global Address Space) languages, such as UPC (Unified parallel C), promises more productivity and performance, providing support for shared, distributed and their hybrid model in efficient manners[3]. PGAS languages demonstrate very little operating cost as compared with MPI for problems that are inadequately parallel[4]. Hand tuned code, with obstruct moves, is still considerably simpler than message passing code. Evaluation of Sequential and parallel FFTin MPI is desirable , because FFT is one of the seven benchmark for measuring performance of HPCC.

The message passing (MPI) is most widely used programming model as it is scalable, maintainable, portable and for a wide verity of platform it provides excellent performance[5]. It is the proper choice for parallel programming on distributed memory systems, such as multi-core clusters. The message-passing provides process communication with other process by explicitly calling library routines to send and receive messages. The key attractive features of MPI include the entire control over data

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

Online ISSN
3007-3138
Print ISSN
3007-312X

SPECTRUM OF ENGINEERING SCIENCES

distribution, explicit communication, data locality optimization, and process synchronization. Due to the above mentioned features MPI programs provides scalability and high performance; However, it also suffers the limitation that is MPI program difficult to construct debug[6].

The shared memory model provides a simpler programming for parallel applications, as here data location control is not required. OMP (OpenMP) is the preferred choice for shared memory programming, as it provides compiler directives to develop parallel application. However, as this model provides support only for shared memory architectures, the performance is limited to computational performance of single computer system. To overcome this limitation, hybrid systems, with both distributed/shared memory, such as multi/many-core clusters, can be programmed using MPI+OMP[6]. However, this model can make the parallelization more complicated and performance gains might not reimburse for the exertion.

The PGAS (Partitioned Global Address Space) is best alternate of conventional programming paradigms. Due to its scalability and performance on large scale clusters. In this paradigm concurrent threads of process that use shared partitioned space that is in actual global arrays have partitioned in multiple positions. The PGAS model has the main features of Shared-Programming-Model and Message-passing. Each Thread in PGAS Model has its Own separate memory space and an associated shared memory of the global address space that can be accessed by other threads. PGAS languages allow shared-memory-like programming on distributed-memory systems and also have a mechanism of exploitation of data locality, because shared memory is partitioned between threads in regions and each one with affinity to related threads. These features make PGAS more important for modern Multi/Many-core architectures[4].

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

Main Contribution of this Work as Follows

The programming model is an Interface to the underlying architecture. Programming paradigms allow applications to utilize the full performance of the underlying architecture. There are many parallel programming paradigms already exist and claims more productivity and performance. Due to continuous advancement in hardware level still there is a need of enhancement and researchers proposed new models to exploit the full performance of the underlying architecture. In this work we will perform comparative and analysis of conventional parallel programming with some recent variation such as PGAS. PGAS languages demonstrate very little operating cost as compared with MPI for problems that are inadequately parallel. Hand tuned code, with obstruct moves, is still considerably simpler than message passing code. Evaluation parallel implementation of FFT is desirable by using MPI and MPJ Express, because the FFT is one of the seven benchmarks for measuring performance of HPCC[1].

Related Work

In [13]proposed one dimensional FFT algorithms for distributed-memory parallel computers with vector symmetric multiprocessor nodes. After alternating four step FFT into five step FFT algorithm we can use to implement the parallel one-dimensional FFT algorithms. We succeeded in obtaining performance of about 38 GFLOPS on a 16-node HITACHI SR8000 which sows low communication cost and long vector length of the proposed algorithm. Implementation of the GPFA which has a lower operation count than conventional FFT algorithms on distributed-memory parallel computers with vector SMP nodes is one of the important problems for future.

The Hierarchical FFT ASIP design is flexible and efficient to meet the requirements of contemporary digital communication standards. In [14] developed their FFTASIP based on Xtensa core LX2.0 and extended the instruction set with four custom

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

instructions to accelerate the FFT computations and data communications. The overall performance is greatly improved by parallel computation and utilization of on-chip custom registers. The hierarchical structure provides good scalability to any point FFT. Both the custom hardware cost and power consumption are acceptable.

Multidimensional high performance parallel FFT algorithm which is an extension of the approach of Agarwal and Cooley[15] . This new algorithm was used to compute a commonly encountered FFT based kernel on the ~BM SP1. We showed that the multi-dimensional formulation helps in reducing the inter-processor communication and also provides an efficient mechanism for blocking for cache of a single node of a parallel machine. They implemented the kernel on the IBM SP1, and observed a performance of 1.25 GFLOP/Seconds on a 64-node system. The performance results demonstrate that the proposed algorithm has low communication cost and utilizes cache effectively

FFT is a widely used algorithm, of which parallelization is a very important topic and many parallel algorithms were published in several decades. In [10]propose COPF, an implement of Parallel FFTs with Inter-Processor Permutations. COPF reserves the features of PFFT with IPP, balances overloads and optimizes communication. COPF can be used widely without updating current hardware since the architecture in which COPF suits are still butterfly. COPF focuses on the communication between processors, so it will have good performance in distributed memory computers. The only flaw is that the results in COPF are re-ranged,and it may limit the utilization.

Depicts the operation of parallel algorithm performance evaluation in PIE, an Environments geared toward performance, efficient parallel programming and the prediction, implementation, measurement and evaluation of parallel Fast

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

Online ISSN 3007-3138

Print ISSN 3007-312X

Fourier Transform algorithms. Measurements indicated that the Cooley-Tukey (shuffle) algorithm is the quickest of the three algorithms [16].The contribution of this study is two fold; first provides an exemplar of a mature technology for evaluating parallel applications. The method employed to underscore the demand for integration between modelling and measurements. Second, we have studied an important application (FFT) and gave relevant results and considerations.

With the extensive applications of the FFT in digital and image signal processing which needs an extensive application of large-scale computing. In [9]basing on the traditional parallel FFT algorithm, the grid technology is introduced. At the same time a kind of grid structure based on center management is advanced. In this structure, the relationship table between the introduction of the node is introduced, which makes the data can pass each other between calculation nodes. Performance of Parallel FFT algorithms is improved when FFT applied to grid Environments. It reflects the computing power of the processing platform grid, and greatly increased the computational efficiency.

In [13] propose high-performance parallel one-dimensional fast Fourier transforms (FFT) algorithms for distributed-memory parallel computers with vector symmetric multiprocessor (SMP) nodes. To expand the innermost loop length alterned four-step into five-step FFT algoritm. We use the four-step and five-step algorithms to implement the parallel one-dimensional FFT algorithms. In our proposed parallel FFT algorithms, all-to-all communication takes place only once. Moreover, the input data and output data are both in natural order. We succeeded in obtaining performance of about 3’8 GFLOPS on a 16-node SR8000. The performance results demonstrate that the proposed algorithms have low communication cost and long vector length. Implementation of the GPFA on distributed-memory parallel

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

computers with vector SMP nodes is one of the important problems for the future.

This paper describes the process of parallel algorithm performance evaluation in PIE, an environment geared toward performance, efficient parallel programming and the prediction, implementation, measurement and evaluation of parallel Fast Fourier Transform algorithms[2]. The contribution of this work is two fold First provides an example of a mature technology for evaluating parallel applications. The method used to emphasize the need for integration between modelling and measurements Second, we have studied an important application (FFT) and presented relevant results and considerations for the parallelization of FFT.

An empirical comparison is made between two parallel implementations of a one-dimensional Fast Fourier transform (FFT) that is targeted for a symmetric multiprocessor (SMP)[17] . On SMP with gigaplane bus, the almost linear is effiecincy function can be achieved for transpose algorithm. Transpose algorithm has the same time complexity . The overhead associated of transpose algorithm is transposing the array three times. The efficiency function is defined as the rate at which the data should be increased with the number of processors to maintain constant efficiency. Tree algorithm is better than the transpose algorithm. However, transpose algorithm is better for all data sizes. Overlapping was used to reduce the effect of start up time when the array is transposed. Furthermore, caches and overlapping can significantly affect the performance of the FFT algorithm on SMP.

The Cray Gemini Interconnect has been recently introduced as the next generation network for building scalablemulti-petascale supercomputers [4].Th eobjective of our work is to design micro-benchmarks motivated from application case studies using the Cray DMAPP user space. The intended outcome of this study is to provide designers of one-sided communication runtime

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

systems with an in-depth performance analysis of performance parameters with the CrayGemini Interconnect. To meet this objective, our study includes designing micro-benchmarks for one-sided communication primitives. The Gemini Interconnect can achieve a peak bandwidth of 6911 MB/s and a latency of1s for get communication primitive. Scalability tests for atomic memory operations and shift communication operation up to 65536 processes shows the efficacy of the Cray Gemini Interconnect. We plan to use this study to design efficient communication protocols for one-sided communication runtime systems and the performance of these communication runtime systems with applications in computational chemistry.

Method and Materials

Implementation of Serial Algorithms

The in-place serial algorithms based upon the signal flow graph of the Cooley-Tukey are given as algorithm 3.1 and 3.2 . The implementation code of this algorithm in C and Java is attached as APPENDIX-A . For the sake of comparison of the relative speed and performance gain have been implemented in C and JAVA for single and double precision because there are many issue related to single and double precision computation have been arise. There are many reason to choose JAVA because it's native langauge of HPC like Fortran and C. The advantages of using JAVA are improved compile time and runtime checking, faster debuging and problem detection and automatic garbage collection. A most attractive feature of applications written in JAVA are portability to any hardware. In the following section Equation 1 is typical algorithm used to compute fourier transformation. Where Equation 2 compute the DFT using set of symetric points around a unit circle and Equation 3 shwo the decimation of DFT from Equation 1. In Equation 3 The FFT divides the DFT problem into two subproblems , each of which equals half the orignal sum.

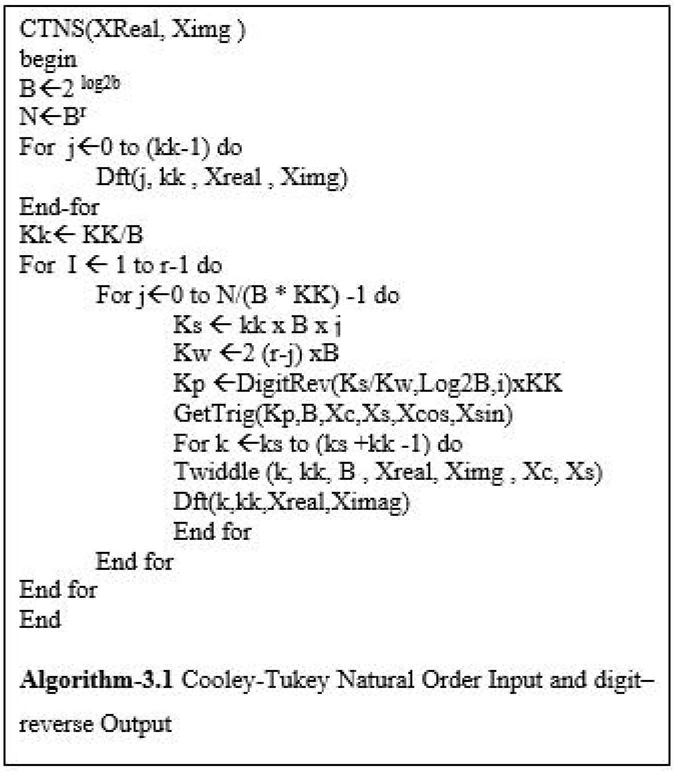
Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

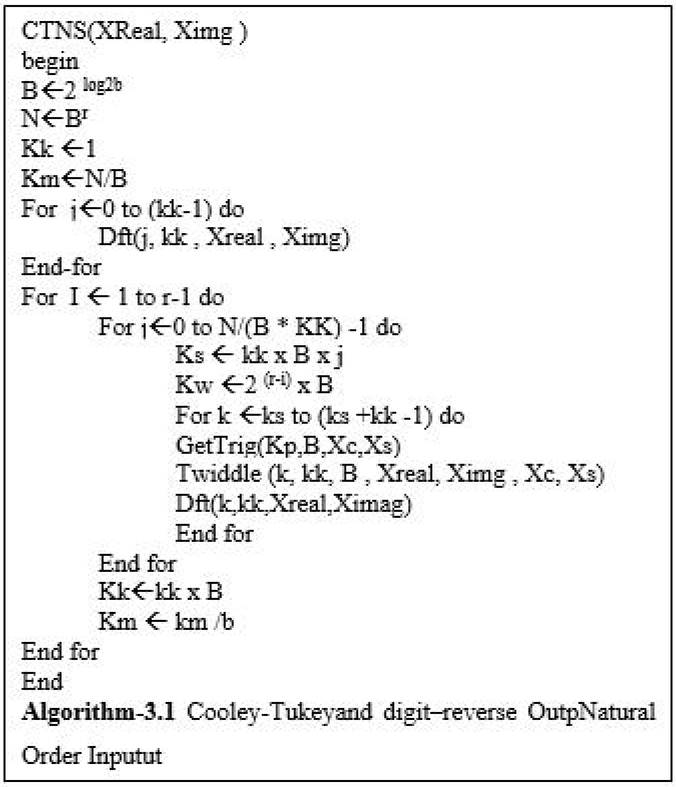
Online ISSN 3007-3138

Print ISSN 3007-312X



```
CTNS(XReal, Ximg )
begin
B←2 $^{log2b}$
N←B$^r$
For  j←0 to (kk-1) do
          Dft(j, kk , Xreal , Ximg)
End-for
Kk← KK/B
For  I ← 1 to r-1 do
          For j←0 to N/(B * KK) -1 do
                    Ks ← kk x B x j
                    Kw ←2 (r-j) xB
                    Kp ←DigitRev(Ks/Kw,Log2B,i)xKK
                    GetTrig(Kp,B,Xc,Xs,Xcos,Xsin)
                    For k ←ks to (ks +kk -1) do
                    Twiddle (k, kk, B , Xreal, Ximg , Xc, Xs)
                    Dft(k,kk,Xreal,Ximag)
                    End for
          End for
End for
End
```

Algorithm-3.1 Cooley-Tukey Natural Order Input and digit–reverse Output



```
CTNS(XReal, Ximg )
begin
B←2 $^{log2b}$
N←B$^r$
Kk ←1
Km←N/B
For  j←0 to (kk-1) do
          Dft(j, kk , Xreal , Ximg)
End-for
For  I ← 1 to r-1 do
          For j←0 to N/(B * KK) -1 do
                    Ks ← kk x B x j
                    Kw ←2 $^{(r-i)}$ x B
                    For k ←ks to (ks +kk -1) do
                    GetTrig(Kp,B,Xc,Xs)
                    Twiddle (k, kk, B , Xreal, Ximg , Xc, Xs)
                    Dft(k,kk,Xreal,Ximag)
                    End for
          End for
          Kk←kk x B
          Km ← km /b
End for
End
```

Algorithm-3.1 Cooley-Tukeyand digit–reverse OutpNatural Order Inputut

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

Online ISSN 3007-3138

Print ISSN 3007-312X

$$X[k] = \sum_{n=0}^{N-1} \mathrm{x[n]}.\mathrm{W}_{\mathrm{N}}^{\mathrm{kn}} \qquad \text{Equation 1}$$

$$\mathrm{WN[n]} = e^{-j\left(\frac{2\pi}{N}\right)} \qquad \text{Equation 2}$$

$$X[\mathrm{k}] = \left(\sum_{n=0}^{\frac{N}{2}-1}\left(\mathrm{x[n]} + \left\langle n + \frac{N}{2}\right\rangle\right).\mathrm{W}_{\frac{\mathrm{N}}{2}}^{\mathrm{kn}}\right) + \left(\sum_{n=0}^{\frac{N}{2}-1}\left(\mathrm{x[n]} + \left\langle n + \frac{N}{2}\right\rangle.\mathrm{W}_{\mathrm{N}}^{\mathrm{n}}\right).\mathrm{W}_{\frac{\mathrm{N}}{2}}^{\mathrm{kn}}\right) \qquad \text{Equation 3}$$

Signal Flow Graph of FFT

When FFT coded , the SFG (Signal Flow Grapg) of the equations disscussed above resemble a butterfly . The butterfly of the equation 3 shown in Figure 1

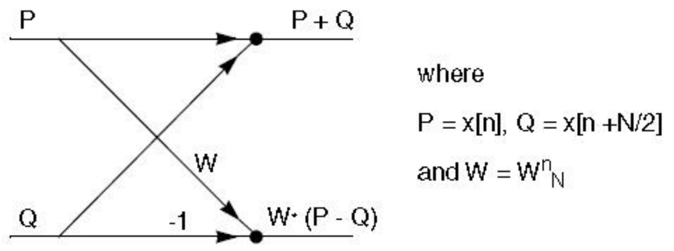


Figure 1 Single Butterfly Representation of a Signal Flow Graph

Where Figure 2 Shows an Signal Flow Graph for a data set of size N=8 . Calculated values are on Right side and Input values are on Right side of the figure. FFT takes natural order input and calculated values are bit reverse.In the Figure 3-2 bit-reverse order, each index of output is represented as a binary and indices bit are reversed. For example for eight points FFT , sequence of indices is 000, 001, 010, 011 .... Reversing these yeilds 000, 100, 010 , 110 . This sequence corresponds to the decimal notation, 0, 4, 2 ,6, which is shown in Figure 3-2 . Each butterfly invloves one complex addtion and one complex subtraction followed by a complex multiplication with W (The value W is called Twiddle value). The one of the most advantage of butterfly structure is that result values can overwritten in memory of input value. That why Radix-2

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN 3007-3138
Print ISSN 3007-312X

FFT is a complete in-place computation. A single itration of in place calculation forms a stage. As Figure 3-2 shown , wihin a stage , there are N/2 butterflies, There are N*log2(N) stages , therefor the Radix-2 FFT is and O(N*log2(N)) number of operations .

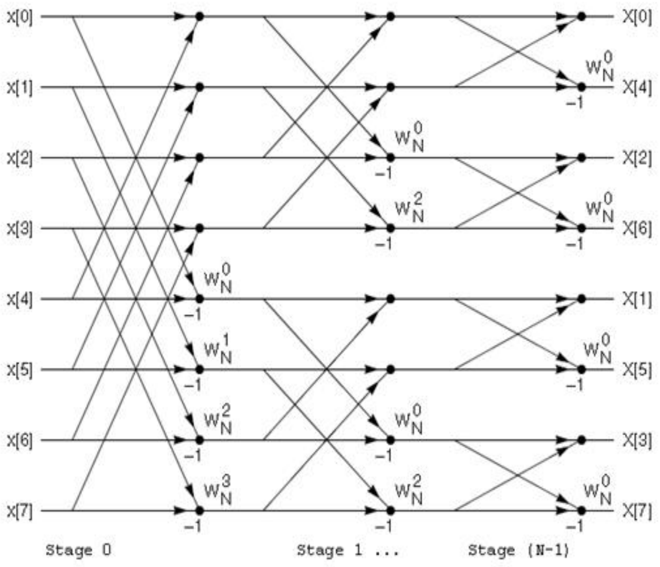


Figure 2 Radix 2 Cooley-Tukey FFT

Implementation of Parallel Algorithms

The parallel FFT algorithm is a divide and conqure data splitting scheme. This means that N/P parallelism of FFT can only be exploited if the data points can be effeciently placed exactly where they are needed and when they are needed. The implementation of FFT is actually data routing problem .

Double Track Implementation

The double track implementation of radix-2 FFT addresses the problems associated With the Single Track implementation of distributed butterflies namely the imbalanced and extra buffering and the problem of communicating twice in the case of Walton’s implementations. This scheme is the modification to the Walton’s implementation and is described as follows : Let us divide the data x[0:N] into two equal halves so that x0 = x[0:N/2] and x1=x[N/2:N] . Now distribute evenly the data x0 among P processor and similarly x1 evenly among P processors so that upper half of the data xi of ith node processor consist of data points x0i[I x N/(2 x P) :N/(2 x P)] and lower half consists of data points x1i[I x N/(2 x P) :N/(2 x P)] . In our case of Cooley-Tukey implementation of distributed FFT on a P=2^P processor system, when the data is in natural order , each processor computes , in parallel, its portion of distributed butterflies and then exchange its lower or upper half with the appropriate processor . After exchanging the data, the two processor again continue the processing independently. Since the communication occur after/before the computation depending upon the order of the data being processed, therefore , there is no need of extra buffering. Since each processor has to compute its own potion of distributed butterflies , this method is , therefore, naturally load balanced. Also , each node exchanges data once , at the end/start of a distributed recursion .

Figure 3 depicts in-Place SFG of 8 points FFT input to the butterflies and output from them for the 3 computational stages arecomputed on 2 processors. Now consider the first FFT computational stage, the upper half of the data points is held by the processor P0 and the data points belonging to lower half are held by the processor P1 . The obvious way to compute butterflies is that the processor P0 and P1 exchange data and then perform computation.

For the Second has 4 set of independent butterflies . Each processor can compute a set of butterflies without intervention of the other processors.

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN 3007-3138

Print ISSN 3007-312X

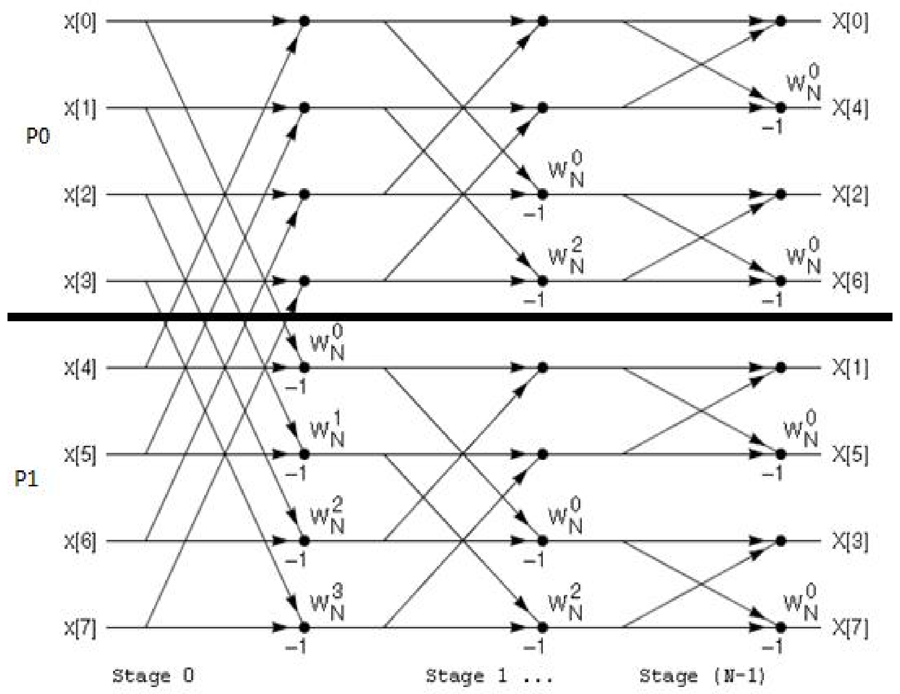


Figure 3 Radix 2 Cooley-Tukey FFT on 2 Processors

Figure 4 Shows that if 8 point FFT is computed on 4 processors so there are two communication steps required at satge 1 and stage 2. Stage 3 can be computed independently without communication. P0 needs x[0] , x[4] and x[1],x[5] for computation at stage 1.

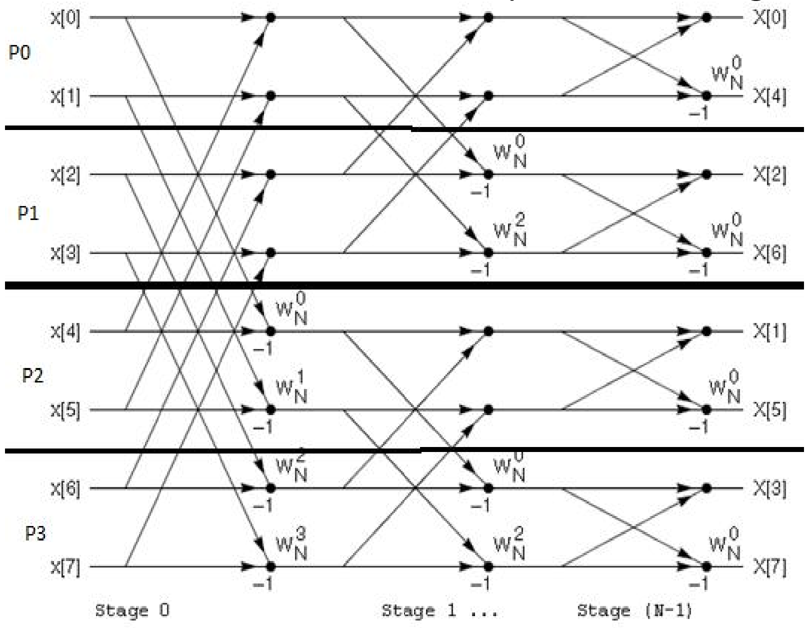


Figure 4 Radix 2 Cooley-Tukey FFT on 4 Processors

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN
3007-3138
Print ISSN
3007-312X

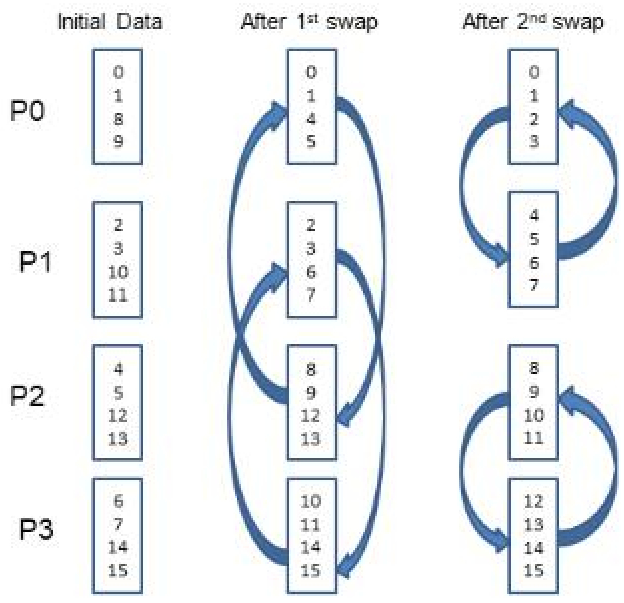


Figure 5 Shows Data swapping between 4 processors

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN 3007-3138

Print ISSN 3007-312X

```
CTN2D(Xr,Xi)
        Begin
        Ndp ←N/P
        kk←N
        kl←P
        Hs ←Ndp /$2^H$
        For i←0 to p-1 do
                ks ← (u/kl) x kk +(Ndo ) x mod(u, kl)
                kp ←DigitRev(ks/$2^{r-1}$ ,2,i) x (kk/2)
                GetTrig(kp, 2, Xc, Xs, Xcos, Xsin)
                For k ←0 to (Ndp/2) -1 do
                        J←k+(Ndp/2)
                        ComplexMul(Xr[j],Xi[j], Xc,Xs)
                        Dft(k,Ndp/2, Xr,Xi)
                End for
        ExchData(Xr,Xi,Ndp/2, Plane.no,Kn,False)
        kk←kk/2
        kl←kl/2
        endfor
end
```

Algorithm 3.3 Double Track Cooley-Tukey Radix-2 Distributed Butterflies, Natural Order Input , Bit Reversed Output

Performance Measurement

An obvious measure to evaluate an algorithm whether serial or parallel is its running time and is the time to taken by the algorithm to solve a problem on a computer that is the time elapsed from the moment the algorithm start to the moment it terminates. In the case of parallel computers , if all processors start and finish their computation simultaneously then the running time of the algorithm will be the running time of any processor. But it is not possible, in general , for all of the processors to begin and end their computation simultaneously . In such a case the running time of the parallel algorithm is equal to the time elapsed between the moment the first processor starts computing and the moment the last processor end computing.

Speed up

The speedup of a parallel algorithm for a problem is the ratio of worst case running time , say T1 of the fastest known sequential algorithm for the problem and worst case running time of the parallel algorithm running on P processors i.e.

Speed up = T1/Tp

Obviously, for a fixed P, the larger the speed up of the parallel algorithm, the better the algorithm.

Platform Overview

In this section we attempt to make a berief overview of plaform used to run programs. The platform used for expermental result of this research work includes *Intel Based deskptop systems model HP Compaq Elite 8300 MicroTower*. More detailed spesefication is disscussed below.

Cluster Details

All parallel experiments done on Rocks based cluster. Brief detail about rocks is given below and Figure shows the hardware perspective how they are connected. In this case Master node receives task and divide it into compute nodes. Compute nodes perform computation and send back results to master node.

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

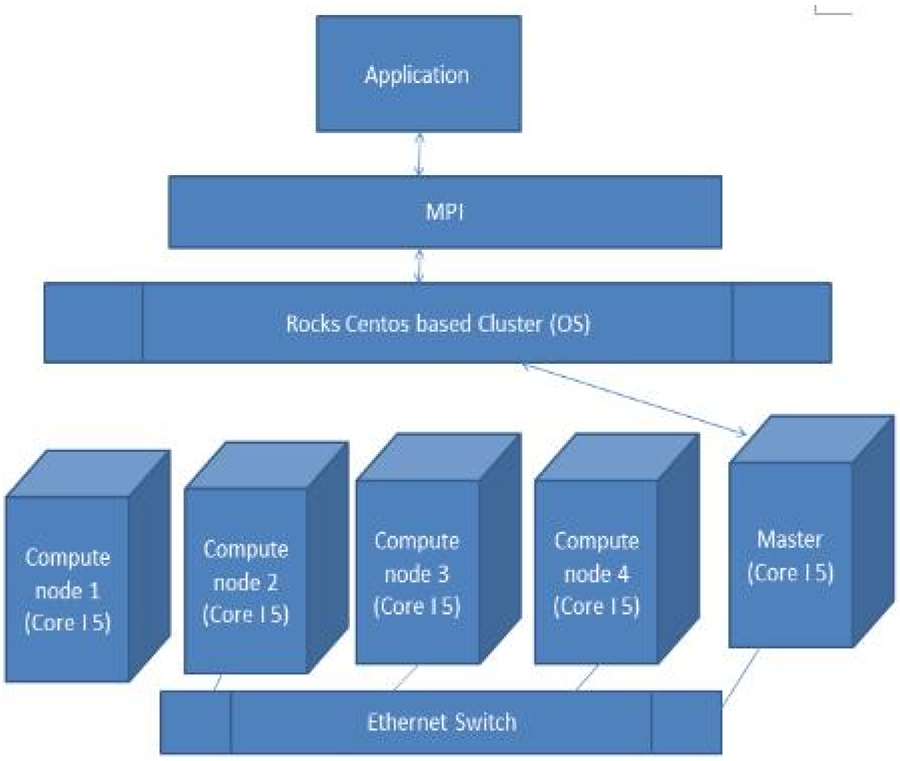

Application
MPI
Rocks Centos based Cluster (OS)
Compute node 1 (Core I 5)
Compute node 2 (Core I 5)
Compute node 3 (Core I 5)
Compute node 4 (Core I 5)
Master (Core I 5)
Ethernet Switch

Table 2: Hardware Specification

Rocks version	6.1
Linux version	2.6.32-279.14.1.el6.x86_64
Compute node	(CPU Core i 5 RAM 8 GB) X 5Nodes
GCC	version 4.4.6 20120305 (Red Hat 4.4.6-4)
MPI	(Open MPI) 1.6.2
Distributor ID	CentOS
Description	CentOS release 6.3 (Final)
Codename:	Final

Results and Implementation

The implementation fo 1D-FFT algorithms are presented in APPENDIX-A in C++ their timing resluts are given in Table 3. FFT algorithm is implemented using Float and Double Data Type using static and dynamic declaration. The reason for choosing dynamic

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN

3007-3138

Print ISSN

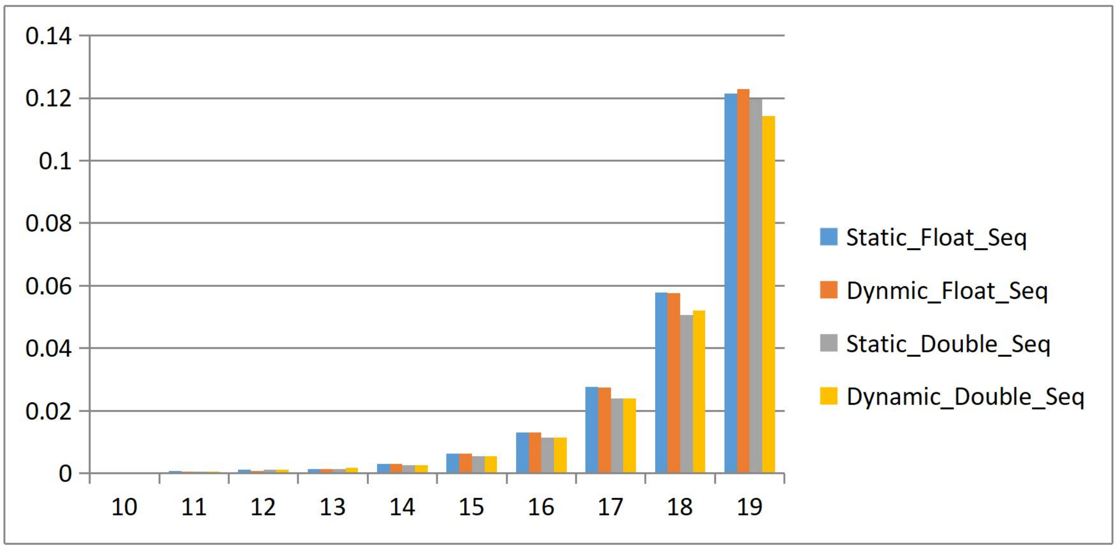
3007-312X

declaration is that while in experiments when Cooley-Tukey serial algorithm is implemented in C++ using static declaration generate segmentation fault at size of 220. That's why we move towards dynamic declaration for efficient use of memory . Static declaration uses stack where dynamic declaration use heap and size of stack is limited that's why segmentation fault occurs.Graph 1 shows running time of Serial Cooley-Tukey algorithm in C language.Result shows that algorithms using static and dynamic declaration remain same but the difference lies between float and double data type. Because algorithm is implemented on 64Bit architecture system and they claim that 64bit-cpu's are natively double precision they need to convert float data into double and after performing computation convert back to float that's why float take more time than double data type on 64 bit architecture.

Table 3: Running Time of FFT

Input	Static_Float_Seq	Dynmic_Float_Seq	Static_Double_Seq	Dynamic_Double_Seq
10	0.000291745	0.000356038	0.000320912	0.000319719
11	0.000721216	0.000588973	0.000524998	0.000528256
12	0.001224918	0.000666936	0.001122157	0.001089096
13	0.00140659	0.00139896	0.001267277	0.001707
14	0.0029517	0.00295655	0.002594867	0.00260957
15	0.00634694	0.006203177	0.005461853	0.005457403
16	0.013139733	0.013077	0.011425633	0.011422133
17	0.027710033	0.0274205	0.0239811	0.023937367
18	0.0578668	0.057536433	0.050629067	0.0520792
19	0.121383333	0.12297	0.119681333	0.114302333

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X



Graph 1: Running Times of Serial Cooley Tukey FFT Algorithm

Optimization

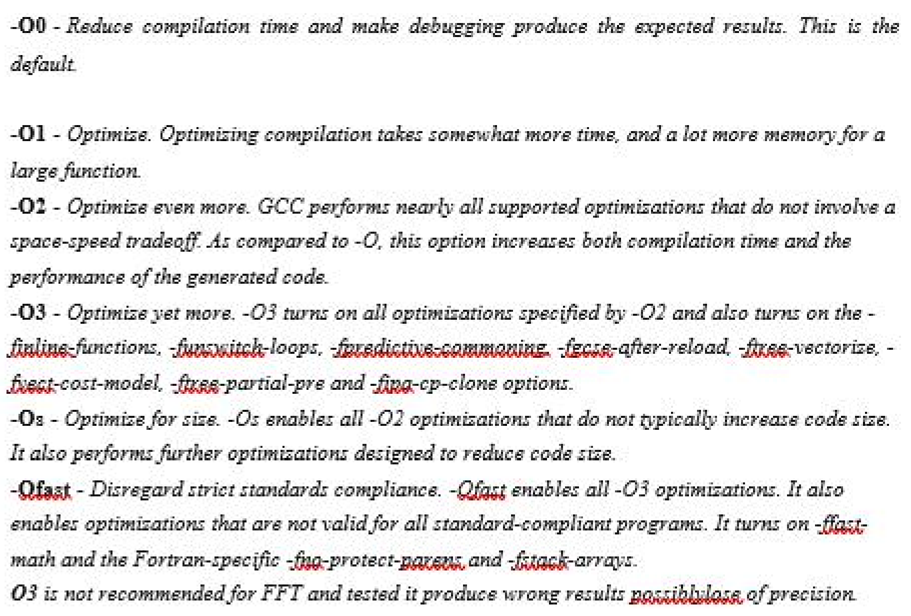
There is another way to increase the performance and that is optimization at compile time. GCC provide many optimization options by using these options performance can be increase if no optimization option is used then compiler's goal is to reduce the cost of compilation and to make debugging produce the expected results. Turning on optimization flags makes the compiler attempt to improve the performance and/or code size at the expense of compilation time and possibly the ability to debug the program. Not all optimization is available using flags and not all flags are recommended for al type of application because some time it produce unexpected results. GNU-GCC provide many levels of optimization which are briefly discussed below.

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN
3007-3138
Print ISSN
3007-312X



-O0 - *Reduce compilation time and make debugging produce the expected results. This is the default.*

-O1 - *Optimize. Optimizing compilation takes somewhat more time, and a lot more memory for a large function.*

-O2 - *Optimize even more. GCC performs nearly all supported optimizations that do not involve a space-speed tradeoff. As compared to -O, this option increases both compilation time and the performance of the generated code.*

-O3 - *Optimize yet more. -O3 turns on all optimizations specified by -O2 and also turns on the -finline-functions, -funswitch-loops, -fpredictive-commoning, -fgcse-after-reload, -ftree-vectorize, -fvect-cost-model, -ftree-partial-pre and -fipa-cp-clone options.*

-Os - *Optimize for size. -Os enables all -O2 optimizations that do not typically increase code size. It also performs further optimizations designed to reduce code size.*

-Ofast - *Disregard strict standards compliance. -Ofast enables all -O3 optimizations. It also enables optimizations that are not valid for all standard-compliant programs. It turns on -ffast-math and the Fortran-specific -fno-protect-parens and -fstack-arrays.*

O3 is not recommended for FFT and tested it produce wrong results possiblylose of precision.

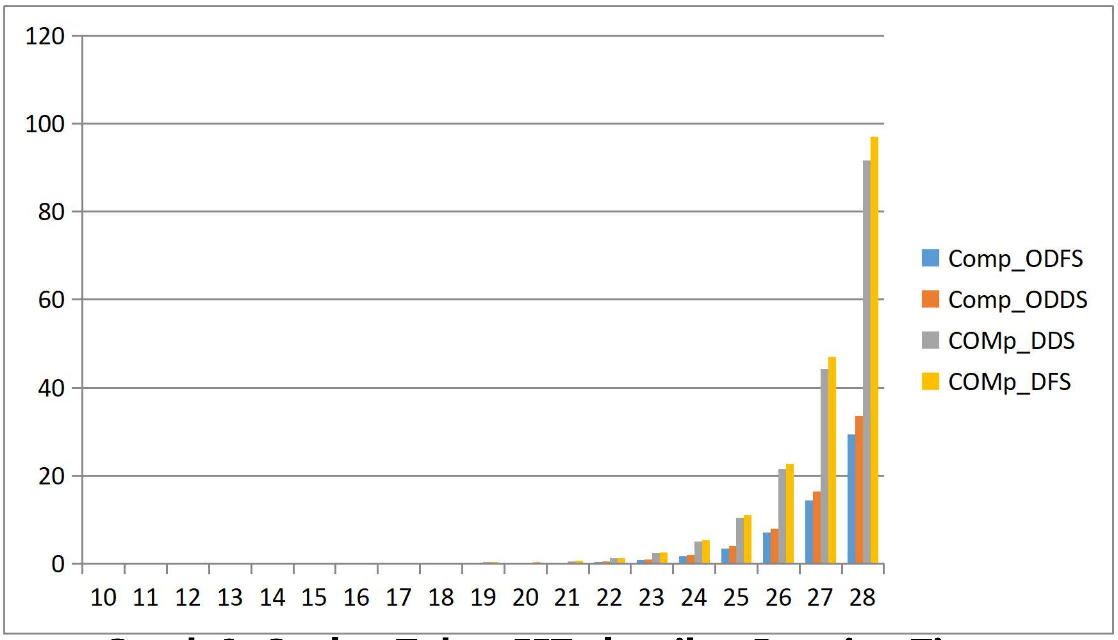
After applying o2 level optimization on the sequential code that generate efficient code with expected results. Here is the implementations result of both strategies. GCC would look for the fastest floating point behavior by default on higher optimization. There is another flag of GCC –ffast-math it can result incorrect output for program which depend on an exact implementation of IEE specification for mathematical calculations. By applying floating point optimization flags -float-store flag the time difference that discussed in section 4.1 is improved and as result float take minimum time as compared to double . In Table 4 shows improved running time after optimization. Running time result shows that after optimization performance increase as it save time 50% .

Table 4: Cooley-Tukey FFT Algorihm Running Time

Inputsize	Comp_ODFS	Comp_ODDS	COMp_DDS	COMp_DFS
10	0.00016308	0.000138154	0.00031972	0.000356
11	0.00030082	0.000255738	0.00052826	0.000589
12	0.00058969	0.000217742	0.0010891	0.0006669
13	0.0005437	0.000431467	0.001707	0.001399
14	0.00106953	0.000887269	0.00260957	0.0029566

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN 3007-3138
Print ISSN 3007-312X

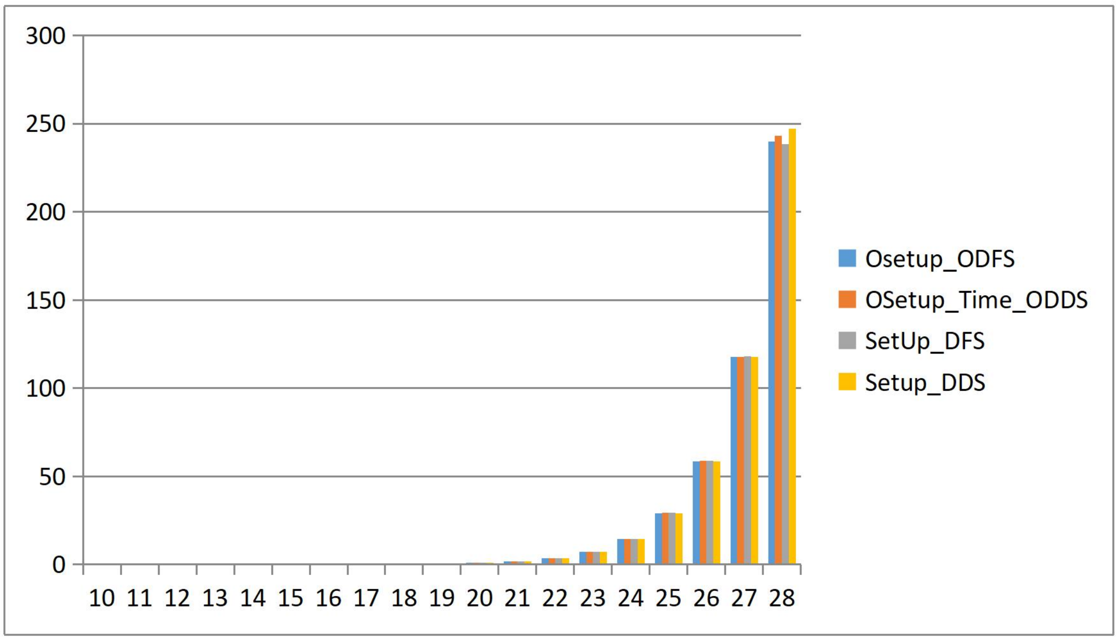
15	0.0021523	0.001885357	0.0054574	0.0062032
16	0.00439188	0.004043327	0.01142213	0.013077
17	0.00906784	0.008636407	0.02393737	0.0274205
18	0.0189455	0.018991833	0.0520792	0.0575364
19	0.03923783	0.043737033	0.11430233	0.12297
20	0.08639157	0.100507333	0.249809	0.2656687
21	0.18560267	0.213011667	0.53574633	0.570782
22	0.39575633	0.439708333	1.13711333	1.2066767
23	0.81875733	0.913025	2.38899667	2.5316267
24	1.67069667	1.901106667	5.00104	5.26563
25	3.43102667	3.934206667	10.3859	10.965767
26	7.0073	7.982743333	21.4294333	22.650833
27	14.3644333	16.31426667	44.2091667	46.943733
28	29.3463	33.63316667	91.6926333	97.096767



Graph 2: Cooley-Tukey FFT algorihm Running Time

There is no impact of optimization and float double precision on setup time as shown in Graph-4.3 which depict same setup time for all cases.

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN 3007-3138
Print ISSN 3007-312X



Graph 4-3 Cooley-Tukey FFT algorihm Running Time

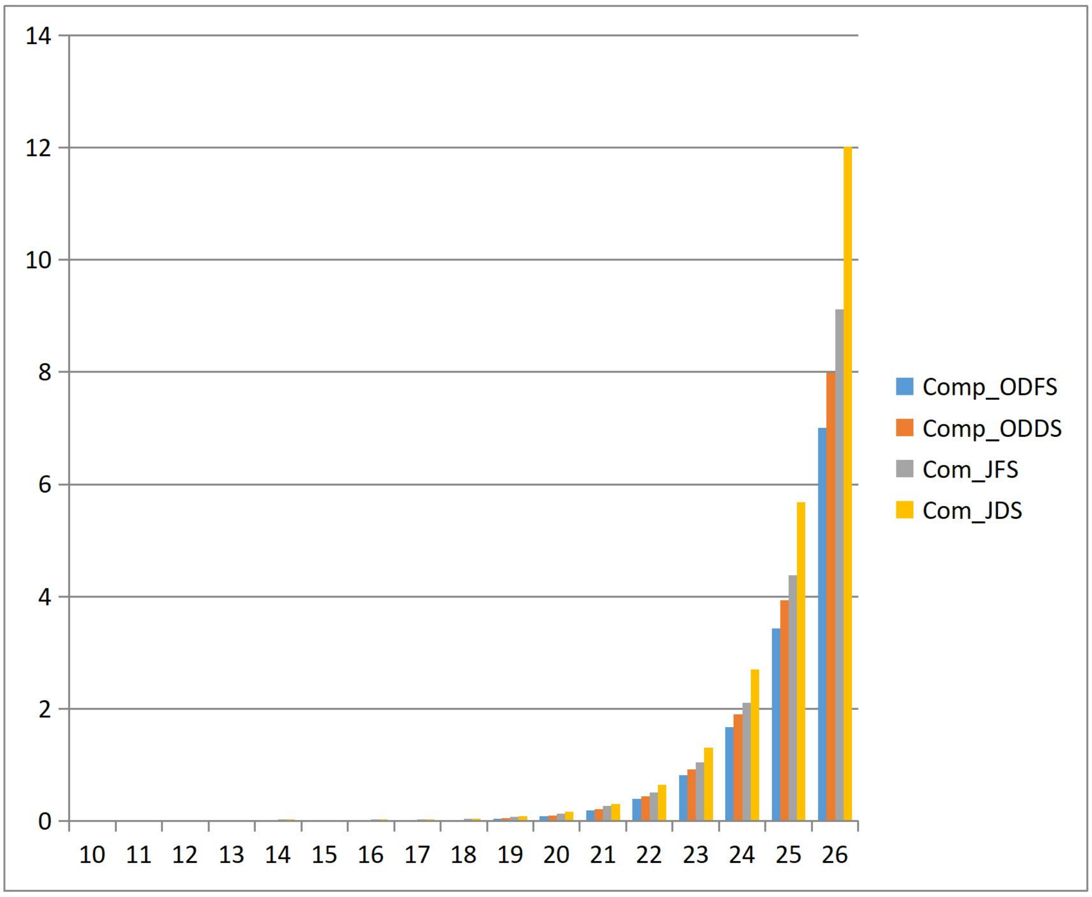
As discussed early in the section 3. 1 JAVA have some attractive advantages over C language so here is the running time comparison with optimized Cooley-Tukey FFT algorithm . As result shown in Table 5 and in Graph 4 JAVA take more time then C because of JVM . JVM provide more portability then C but it never promises of performance in term of time.

Table 5: Cooley-Tukey FFT algorihm Running Time in JAVA

Inputsize	Comp_ODFS	Comp_ODDS	Com_JFS	Com_JDS
10	0.00016308	0.000138154	0.001	0.001
11	0.00030082	0.000255738	0.002	0.003
12	0.00058969	0.000217742	0.004	0.005
13	0.0005437	0.000431467	0.01233333	0.015
14	0.00106953	0.000887269	0.02966667	0.031
15	0.0021523	0.001885357	0.018	0.018
16	0.00439188	0.004043327	0.02133333	0.0213333
17	0.00906784	0.008636407	0.02733333	0.0286667
18	0.0189455	0.018991833	0.041	0.0416667
19	0.03923783	0.043737033	0.06833333	0.0783333
20	0.08639157	0.100507333	0.131	0.1666667

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN 3007-3138
Print ISSN 3007-312X

21	0.18560267	0.213011667	0.26466667	0.295
22	0.39575633	0.439708333	0.51	0.6483333
23	0.81875733	0.913025	1.047	1.311
24	1.67069667	1.901106667	2.10633333	2.694
25	3.43102667	3.934206667	4.374	5.6783333
26	7.0073	7.982743333	9.12266667	12.022333



Graph 4 Cooley-Tukey FFT algorithm Running Time With JAVA

The third proposed strategy in pure MPI using C++. For this strategy the massage passing model is used one way using MPI_SEND() ,MPI_RCV and two way using MPi_SENDRCV () is implemented .On the first step Master Node of HPC cluster take input of complex number generated by random() function of C++ and stored in file size of 230. Then Master node distribute the data

Vol. 3 No. 1 (2025)

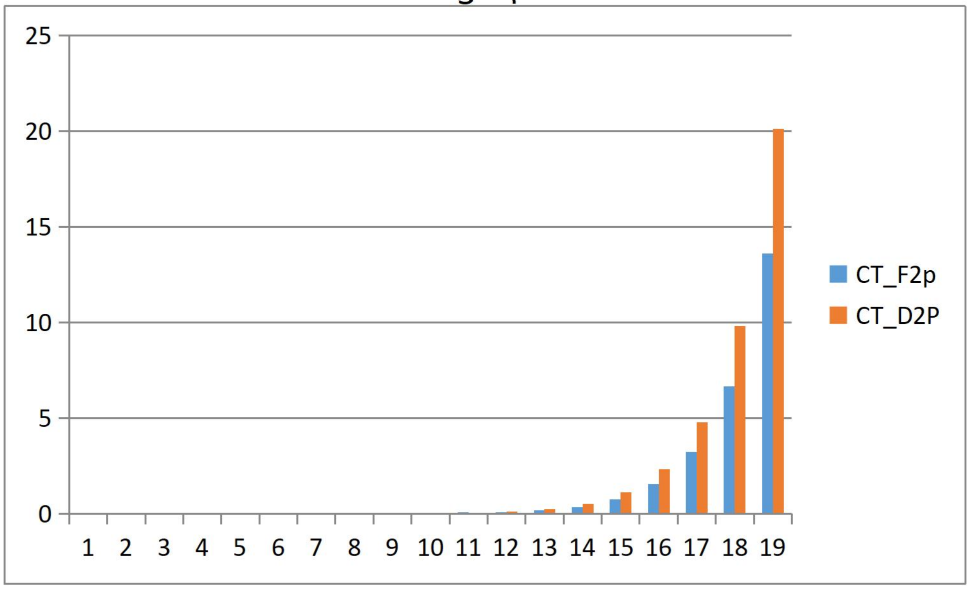
Spectrum of Engineering Sciences

Online ISSN 3007-3138

Print ISSN 3007-312X

among compute node as shown in figure 4.1. when compute node recived data then start computation and when the communication required they exchange their data as discussed earlier in the Section 3 . The procedure of initializing the input data is not included in our timing measurements. Hence, we simulate an environment where the input data is already distributed on the processor grid. In order to avoid any possible distortion, the phase of initialization is encountered within a global barrier. Once again, in order to allow a detailed performance analysis of the runtime, we introduced a set of timers to measure the time taken for each step of the whole procedure. The MPI's function MPI Wtime() was used for all the measurements made in the code.

Initially double track Cooley-Tukey algorithm implemented in C++ using MPI with float and double data types for and optimized using o2 level using different flags after result analysis and verification. As results shown in graph 5.



Graph 5: Running Time With Float and Double on 2 Processors

The send-receive operations combine in one call the sending of a message to one destination and the receiving of another message, from another process. The two (source and destination) are possibly the same. A send-receive operation is very useful for

Vol. 3 No. 1 (2025)

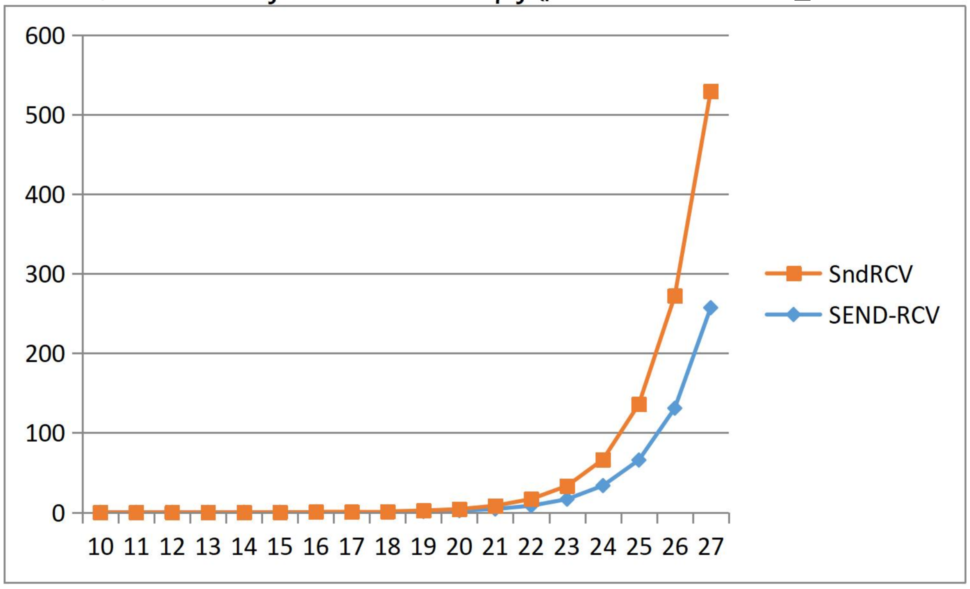
Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN 3007-3138

Print ISSN 3007-312X

executing a shift operation across a chain of processes. If blocking sends and receives are used for such a shift, then one needs to order the sends and receives correctly (for example, even processes send, then receive, odd processes receive first, then send) so as to prevent cyclic dependencies that may lead to deadlock. When a send-receive operation is used, the communication subsystem takes care of these issues. MPI_SENDRCV is more attractive than simple MPI_SEND and MPI_RCV but some in place algorithms there is a need of memory buffer because send and receive starts at same type so we cannot send and receive from same buffer that why temporary buffer is needed once send receive completed then temporary buffer loaded back to actual memory its take more time than simple MPI_SEND and MPI_RCV. As depict Graph.6 MPI_SENDRCV low performance due to memory overhead on FFT. Exprementel resluts computed on ROCKS based SMP cluster on 16 cores. As the number of processors increases communication time incereases, that's why more memcpy() is used in MPI_SENDRCV() .



Graph 6: Running Time With Float and Double on 2 Processors

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN
3007-3138
Print ISSN
3007-312X

Conclusion

We next evaluate the performance of our implementation. The performance results were obtained for different FFT sizes, from 2^{10} to 2^{30} points. In section 4 presents the data throughput results. For FFT computation implemented on Rocks, the data throughput shown in the Graphs and produce linear speed up. Following observations made during experiments. The larger the problem size is, the longer it takes to be computed for all cases using serial implementation the only way to minimize time is parallelism. Float data type take more time than Double data type on 64 Bit architecture on default configuration of GCC. Level 3 optimization is not suitable for double precision data computation. Level 2 Optimization using some appropriate flags according to the nature of a problem can save time up to 30%. MPI_SENDRCV shows low performance for in place algorithms. There is a still need of improvement by improving network. Main target to reduce time by implementing parallelism is achieved.

Reference

1. Bilal, O., Asif, S., Zhao, M., Li, Y., Tang, F., & Zhu, Y. (2024). Differential evolution optimization based ensemble framework for accurate cervical cancer diagnosis. Applied Soft Computing, 167, 112366.

2. Khan, S. R., Raza, A., Shahzad, I., & Ijaz, H. M. (2024). Deep transfer CNNs models performance evaluation using unbalanced histopathological breast cancer dataset. Lahore Garrison University Research Journal of Computer Science and Information Technology, 8(1).

3. Bilal, Omair, Asif Raza, and Ghazanfar Ali. "A Contemporary Secure Microservices Discovery Architecture with Service Tags for Smart City Infrastructures." VFAST Transactions on Software Engineering 12, no. 1 (2024): 79-92.

4. Waqas, M., Ahmed, S. U., Tahir, M. A., Wu, J., & Qureshi, R. (2024). Exploring Multiple Instance Learning (MIL): A brief survey.

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

Expert Systems with Applications, 123893.

5. S. Abhari, S. R. N. Kalhori, M. Ebrahimi, and H. Hasannejadasl, "Artificial Intelligence Applications in Type 2 Diabetes Mellitus Care: Focus on Machine Learning Methods," vol. 25, no. 4, pp. 248–261, 2019.

6. Mahmood, F., Abbas, K., Raza, A., Khan,M.A., & Khan, P.W. (2019). Three Dimensional Agricultural Land Modeling using Unmanned Aerial System (UAS). International Journal of Advanced Computer Science and Applications (IJACSA) [p-ISSN : 2158-107X, e-ISSN : 2156-5570], 10(1).

7. Waqas, M., Tahir, M. A., & Qureshi, R. (2023). Deep Gaussian mixture model based instance relevance estimation for multiple instance learning applications. Applied intelligence, 53(9), 10310-10325.

8. H. Kaur and V. Kumari, "Predictive Modelling and Analytics for Diabetes using a Machine Learning Approach," *Appl. Comput. Informatics*, no. December, 2018, doi: 10.1016/j.aci.2018.12.004.

9. I. M. Ibrahim and A. M. Abdulazeez, "The Role of Machine Learning Algorithms for Diagnosing Diseases," vol. 02, no. 01, pp. 10–19, 2021, doi: 10.38094/jastt20179.

10. M. Wajid, M. K. Abid, A. Asif Raza, M. Haroon, and A. Q. Mudasar, "Flood Prediction System Using IOT & Artificial Neural Network", VFAST trans. softw. eng., vol. 12, no. 1, pp. 210–224, Mar. 2024.

11. A. Mujumdar and V. Vaidehi, "ScienceDirect ScienceDirect ScienceDirect ScienceDirect Diabetes Prediction using Machine Learning Aishwarya Mujumdar Diabetes Prediction using Machine Learning Aishwarya Mujumdar Aishwarya," *Procedia Comput. Sci.*, vol. 165, pp. 292–299, 2019, doi: 10.1016/j.procs.2020.01.047.

12. HUSSAIN, S., Raza, A., MEERAN, M. T., IJAZ, H. M., & JAMALI, S. (2020). Domain Ontology Based Similarity and Analysis in Higher Education. IEEEP New Horizons Journal, 102(1), 11-16.

13. S. Brian and R. R. B. Pharmd, "Prediction of Nephropathy in

Vol. 3 No. 1 (2025)

Spectrum of Engineering Sciences

SPECTRUM OF ENGINEERING SCIENCES

Online ISSN 3007-3138

Print ISSN 3007-312X

Type 2 Diabetes: An Analysis of the ACCORD Trial applying Machine Learning Techniques," no. 317, doi: 10.1111/cts.12647.

14. Faruque, "Performance Analysis of Machine Learning Techniques to Predict Diabetes Mellitus," *2019 Int. Conf. Electr. Comput. Commun. Eng.*, pp. 1–4, 2019.

15. Raza, A., Soomro, M. H., Shahzad, I., & Batool, S. (2024). Abstractive Text Summarization for Urdu Language. Journal of Computing & Biomedical Informatics, 7(02).

16. Asif, S., Wenhui, Y., ur-Rehman, S., ul-ain, Q., Amjad, K., Yueyang, Y., ... & Awais, M. (2024). Advancements and Prospects of Machine Learning in Medical Diagnostics: Unveiling the Future of Diagnostic Precision. Archives of Computational Methods in Engineering, 1-31.

17. Asif, S., Zhao, M., Li, Y., Tang, F., Ur Rehman Khan, S., & Zhu, Y. (2024). AI-Based Approaches for the Diagnosis of Mpox: Challenges and Future Prospects. Archives of Computational Methods in Engineering, 1-33.

18. Waqas, M., Tahir, M. A., & Khan, S. A. (2023). Robust bag classification approach for multi-instance learning via subspace fuzzy clustering. Expert Systems with Applications, 214, 119113.

19. J. Li *et al.*, "International Journal of Medical Informatics Establishment of noninvasive diabetes risk prediction model based on tongue features and machine learning techniques," *Int. J. Med. Inform.*, vol. 149, no. August 2020, p. 104429, 2021, doi: 10.1016/j.ijmedinf.2021.104429.

20. Waqas, M., Tahir, M. A., Al-Maadeed, S., Bouridane, A., & Wu, J. (2024). Simultaneous instance pooling and bag representation selection approach for multiple-instance learning (MIL) using vision transformer. Neural Computing and Applications, 36(12), 6659-6680.

21. S. G. Azevedo *et al.*, "System-Independent Characterization of Materials Using Dual-Energy Computed Tomography," *IEEE Trans. Nucl. Sci.*, vol. 63, no. 2, pp. 341–350, 2016, doi:

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

10.1109/TNS.2016.2514364.

22. Khan, S. U. R., Asif, S., Zhao, M., Zou, W., Li, Y., & Li, X. (2024). Optimized Deep Learning Model for Comprehensive Medical Image Analysis Across Multiple Modalities. Neurocomputing, 129182.

23. Shahzad, I., Khan, S. U. R., Waseem, A., Abideen, Z. U., & Liu, J. (2024). Enhancing ASD classification through hybrid attention-based learning of facial features. Signal, Image and Video Processing, 1-14.

24. B. G. Choi, S. Rha, S. W. Kim, J. H. Kang, J. Y. Park, and Y. Noh, "Machine Learning for the Prediction of New-Onset Diabetes Mellitus during 5-Year Follow-up in Non-Diabetic Patients with Cardiovascular Risks," vol. 60, no. 2, pp. 191–199, 2019.

25. Raza, A., Salahuddin, & Inzamam Shahzad. (2024). Residual Learning Model-Based Classification of COVID-19 Using Chest Radiographs. Spectrum of Engineering Sciences, 2(3), 367–396.

26. Khan, S.U.R.; Raza, A.;Waqas, M.; Zia, M.A.R. Efficient and Accurate Image Classification Via Spatial Pyramid Matching and SURF Sparse Coding. Lahore Garrison Univ. Res. J. Comput. Sci. Inf. Technol. 2023, 7, 10–23.

27. Farooq, M.U.; Beg, M.O. Bigdata analysis of stack overflow for energy consumption of android framework. In Proceedings of the 2019 International Conference on Innovative Computing (ICIC), Lahore, Pakistan, 1–2 November 2019; pp. 1–9.

28. Farooq, M. U., & Beg, M. O. (2019, November). Bigdata analysis of stack overflow for energy consumption of android framework. In 2019 International Conference on Innovative Computing (ICIC) (pp. 1-9). IEEE.

29. Farooq, M. U., Khan, S. U. R., & Beg, M. O. (2019, November). Melta: A method level energy estimation technique for android development. In 2019 International Conference on Innovative Computing (ICIC) (pp. 1-10). IEEE.

30. Khan, S. U. R., & Asif, S. (2024). Oral cancer detection using

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X
SES

feature-level fusion and novel self-attention mechanisms. Biomedical Signal Processing and Control, 95, 106437.
31. Raza, A.; Meeran, M.T.; Bilhaj, U. Enhancing Breast Cancer Detection through Thermal Imaging and Customized 2D CNN Classifiers. VFAST Trans. Softw. Eng. 2023, 11, 80–92.
32. Dai, Q., Ishfaque, M., Khan, S. U. R., Luo, Y. L., Lei, Y., Zhang, B., & Zhou, W. (2024). Image classification for sub-surface crack identification in concrete dam based on borehole CCTV images using deep dense hybrid model. Stochastic Environmental Research and Risk Assessment, 1-18.
33. Khan, S.U.R.; Asif, S.; Bilal, O.; Ali, S. Deep hybrid model for Mpox disease diagnosis from skin lesion images. Int. J. Imaging Syst. Technol. 2024, 34, e23044.
34. Khan, S.U.R.; Zhao, M.; Asif, S.; Chen, X.; Zhu, Y. GLNET: Global–local CNN's-based informed model for detection of breast cancer categories from histopathological slides. J. Supercomput. 2023, 80, 7316–7348.
35. Khan, S.U.R.; Zhao, M.; Asif, S.; Chen, X. Hybrid-NET: A fusion of DenseNet169 and advanced machine learning classifiers for enhanced brain tumor diagnosis. Int. J. Imaging Syst. Technol. 2024, 34, e22975.
36. IoannisKavakiotis, Olga Tsave, Athanasios Salifoglou, and NicosMaglaveras, "Machine Learning and Data Mining Methods in Diabetes Research", Computational and Structural Biotechnology Journal, vol. 15, pp. 104– 116, 2017
37. Khan, U. S., & Khan, S. U. R. (2024). Boost diagnostic performance in retinal disease classification utilizing deep ensemble classifiers based on OCT. Multimedia Tools and Applications, 1-21.
38. Khan, M. A., Khan, S. U. R., Haider, S. Z. Q., Khan, S. A., & Bilal, O. (2024). Evolving knowledge representation learning with the dynamic asymmetric embedding model. Evolving Systems, 1-16.
39. Raza, A., & Meeran, M. T. (2019). Routine of encryption in

Spectrum of Engineering Sciences
SPECTRUM OF ENGINEERING SCIENCES
Online ISSN
3007-3138
Print ISSN
3007-312X

cognitive radio network. Mehran University Research Journal of Engineering & Technology, 38(3), 609-618.

40. Al-Khasawneh, M. A., Raza, A., Khan, S. U. R., & Khan, Z. (2024). Stock Market Trend Prediction Using Deep Learning Approach. Computational Economics, 1-32.

41. Khan, U. S., Ishfaque, M., Khan, S. U. R., Xu, F., Chen, L., & Lei, Y. (2024). Comparative analysis of twelve transfer learning models for the prediction and crack detection in concrete dams, based on borehole images. Frontiers of Structural and Civil Engineering, 1-17.

42. Hekmat, A., Zhang, Z., Ur Rehman Khan, S., Shad, I., & Bilal, O. (2025). An attention-fused architecture for brain tumor diagnosis. Biomedical Signal Processing and Control, 101, 107221. https://doi.org/https://doi.org/10.1016/j.bspc.2024.107221

43. Khan, S. U. R., Raza, A., Shahzad, I., & Ali, G. (2024, October). Enhancing Concrete and Pavement Crack Prediction through Hierarchical Feature Integration with VGG16 and Triple Classifier Ensemble. In 2024 Horizons of Information Technology and Engineering (HITE) (pp. 1-6). IEEE.

44. M. Waqas, Z. Khan, S. U. Ahmed and A. Raza, "MIL-Mixer: A Robust Bag Encoding Strategy for Multiple Instance Learning (MIL) using MLP-Mixer," 2023 18th International Conference on Emerging Technologies (ICET), Peshawar, Pakistan, 2023, pp. 22-26.