Predictive Model for FFT Scalability Performance

K.M. Mostafa#, M. B. Abdelhalim#, Mohamed Waleed Fakhr##

#College of Computing and Information Technology (CCIT)
Arab Academy of Science and Technology and Maritime Transport (AASTMT)
Cairo, Egypt

1kziad@live.com
2mbakr@ieee.org
3waleedf@aast.edu

Abstract—In order to introduce more efficient high performance computing (HPC) applications we need to construct a performance model for the complex heterogeneous systems.
Most of the HPC applications, as in the molecular dynamics simulation implement the FFT parallel algorithm that consumes a major portion of the application execution time.
This work constructs a model that can be used to decide the execution plan scalability and highlight major factors that impact directly the algorithm performance.
The parallel FFT algorithm is used as a case study for our modeling procedure.
This paper aims to explore the most impacting parameters of the parallel FFT algorithms execution where other platform and hardware-dependent factors are not included.

Keywords: FFT, Performance, model, parallel, decomposition

1 Introduction

High Performance Computing (HPC) allows scientists and engineers to solve complex science and engineering problems using parallel algorithms on a large number of computing processors and high bandwidth network.

As the scalability is a very important measurement of the applications performance, predicting the algorithm scalability is required in the pre-deploying phase.
The peta-scale computers are the new generation of the HPC platforms as in blue waters project. It delivers 1 peta fops performance [1], the lack of scalability of the parallel algorithms forms a step challenging issue in the new Peta-scale computers.
The scalability issues may prevent the applications from achieving the desired performance hence rise up the importance of the prediction model.
The performance prediction model should address the major factors that affect the performance, however in order to implement a generic model it should not be very detailed to a specific hardware or a specific platform.

The Fast Fourier Transform (FFT) is an enhanced method for calculating the Discrete Fourier Transform (DFT). As it is more efficient, often reducing the computation steps by hundreds. The 3D FFT can be expressed as [2]

\[ f(k_x, k_y, k_z) = \sum_s \sum_y \sum_z f(x, y, z) e^{ik_x x} e^{ik_y y} e^{ik_z z} \]

(1)

FFT has been one of the most popular and widely used numerical methods in many areas of scientific computing, including digital speech and signal processing, solving partial differential equations, molecular dynamics many-body simulations and Monte Carlo simulations.

Given its importance, there have been a large number of libraries that provide different implementations of FFT aimed at achieving high-performance in various environments.

The outline of this paper is organized as follows. Section 2 shows the most relevant work related to our work. Section 3 describes the different decomposition algorithms. The experiments test-bed and the analytical model described in section 4 and 5 respectively, section 6 illustrate the experiments results and finally paper conclusion is section 7.

2 Related Work

The steps required to calculate 3DFFT with 2D decomposition add extra computation and communication step while they extend the algorithm scalability [3].

Previous work introduced an analysis of the 3DFFT decompositions methodologies and the performance measurements of each decomposition method [3], the experimental results shown in Table 1 shows a speedup of the 2D decomposition [4], the 2D decomposition theoretical analysis proves a better scalability performance up to the product of the widest two dimensions of the data mesh however it is proven in this work that other factors have a major impact on the algorithm scalability.
The major FFT parallel algorithm cost introduced as a summation of the communication and computation tasks

\[ T_{fft}(N; P; Chunks) = T_{comp}(N; P) + T_{comm}(Chunks; P) \]
Where Tcomp(N; P) is the Computation Time, Tcomm(Chunks; P) is the Communication Time [5]. In this paper the analysis model cost equation introduced with more details.

Many intensive studies addressed the FFT algorithm communication cost highlighting BlueGene/L torus and mesh network topologies [6], in our work both computation and communication factors considered along with the different decomposition algorithms and the grid size, the model addressed only the major network parameters aiming to be independent of the network topologies.

A comparison between different FFT libraries and the performance measurements through each decomposition method of the libraries predicts the theoretical scalability enhancements of the 2D decomposition [4]. The FFTW [7] is a popular implementation of the 3DFFT; however the FFTW does not support the 2D decomposition however some modifications added to enable the 2D decomposition more details in section IV.

Message Passing Interface (MPI) is a language-independent communications protocol used to program parallel computers especially with the distributed system. MPI communication protocols include both point-to-point and collective communication [8].

There are various models to evaluate the MPI communication performance from the hardware perspective as LogP [9], LogGP [10] and pLogP [11] where parameters as in Table 2 [12].

TACC ranger network parameters measured [12] for the blocking and the non-blocking communication (discussed in the analytical model section) as shown in Table 3.

<table>
<thead>
<tr>
<th>Library</th>
<th>Time</th>
<th>Cores</th>
<th>Cores/Node</th>
<th>Decomposition</th>
<th>Nodes</th>
</tr>
</thead>
<tbody>
<tr>
<td>2DECOMP&amp;FFT</td>
<td>0.0045</td>
<td>8192</td>
<td>16</td>
<td>2D</td>
<td>512</td>
</tr>
<tr>
<td>2DECOMP&amp;FFT</td>
<td>0.0057</td>
<td>512</td>
<td>1</td>
<td>2D</td>
<td>512</td>
</tr>
<tr>
<td>P3DFFT</td>
<td>0.006</td>
<td>4096</td>
<td>16</td>
<td>2D</td>
<td>256</td>
</tr>
<tr>
<td>P3DFFT</td>
<td>0.0063</td>
<td>512</td>
<td>1</td>
<td>2D</td>
<td>512</td>
</tr>
<tr>
<td>FFTW</td>
<td>0.0124</td>
<td>256</td>
<td>1</td>
<td>1D</td>
<td>256</td>
</tr>
<tr>
<td>P3DFFT</td>
<td>0.013</td>
<td>256</td>
<td>1</td>
<td>1D</td>
<td>256</td>
</tr>
<tr>
<td>2DECOMP&amp;FFT</td>
<td>0.014</td>
<td>256</td>
<td>1</td>
<td>1D</td>
<td>256</td>
</tr>
<tr>
<td>P3DFFT</td>
<td>0.0233</td>
<td>256</td>
<td>16</td>
<td>1D</td>
<td>16</td>
</tr>
<tr>
<td>2DECOMP&amp;FFT</td>
<td>0.0244</td>
<td>256</td>
<td>16</td>
<td>1D</td>
<td>16</td>
</tr>
<tr>
<td>FFTW</td>
<td>0.0341</td>
<td>256</td>
<td>16</td>
<td>1D</td>
<td>16</td>
</tr>
</tbody>
</table>

1 Libraries sorted by time.
2 Total number of cores
+ Maximum number of total cores (1D decomposition).

Table 2: LogGP Parameters

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>m</td>
<td>Message size</td>
</tr>
<tr>
<td>L</td>
<td>Network latency</td>
</tr>
<tr>
<td>o</td>
<td>Overhead per message transm.</td>
</tr>
<tr>
<td>e</td>
<td>Gap between successive mes.</td>
</tr>
<tr>
<td>g</td>
<td>The reciprocal of network b.</td>
</tr>
<tr>
<td>P</td>
<td>Number of processors</td>
</tr>
</tbody>
</table>

3 3D FFT decompositions

FFT Decomposition is the process of distributing the 3D FFT data mesh and determine the steps of communication and computation across the number of processors grid.

The decomposition way beside the hardware architecture and specifications are the key factors of any FFT algorithm execution plan.

In 1D decomposition shown in Figure 1 which is also called slab decomposition, the 3D data grid is divided into a number of slabs across one dimension.

At the first step, the 1DFFT first calculated with respect to two dimensions let’s say the x and y direction, and along the third direction z then in the second step a global transpose with all to all communication takes place then the at the last step the 2DFFT is calculated across the last dimension z.

In the 1D decomposition, the global transpose (all-to-all communication) takes place only once, however the drawback of this algorithm is the scalability where the number of slabs is limited to the maximum number of processors in one dimension.

2D decomposition of the data, which sometimes called the pencil decomposition, is when the data is divided across two dimensions of the 3D data grid. In the first step, the 1DFFT is calculated for each pencil followed by a global transpose in the second step. At the third step, the 1DFFT is calculated along the second dimension then again in the forth step the global transpose takes place. At the last step, a final calculation of the 1DFFT along the last dimension is performed.

The 2D decomposition algorithm is shown in Figure 2 [6]. Two global all-to-all transpose takes place, however
the maximum scalability is up two the product of the widest two dimensions of the 3D data grid.

![3DFFT calculation steps for the 2D decomposition](image)

Figure 2: 3DFFT calculation steps for the 2D decomposition

4 Test-bed

Our target platform is the Texas Advanced Computing Centre – Ranger constellation – TACC Ranger. The TACC system consists of 62,976 nodes interconnected through InfiniBand technology providing a theoretical 1GB/sec point-to-point bandwidth, TACC provide a peak performance of 9.2 GFLOPS/core or 128 GFLOPS/node [13]. As shown in Figure 3, every four cores are attached to one socket. Such structure must be considered when measuring the communication performance specially when testing with little number of cores (e.g. 2, 4, 8, and 16) to avoid measuring the intra socket communication instead of the network communication.

![TACC Ranger Architecture](image)

Figure 3: TACC Ranger Architecture

Our experiments code was developed using C++, MPI routines version 1.2.7, PGI 7.1 compiler.

The Fastest Fourier Transform in the West (FFTW) [7] library is selected for computing multiple dimensions of the DFT, FFTW is a MIT developed C subroutine library for computing multiple dimensions of the DFT, FFTW supports only the 1D decomposition however some changes are added to our experimental code to enable the 2D decomposition testing.

In the 1D decomposition, the `fftw_mpi_local_size` function used to determine the local size for each processor, `fftw_plan_dft` function is used to initialize the plan with the local variables and the input and output data and `fftw_execute` function is used to perform the FFT calculation.

In the 1D decomposition, the local size of the processors that are out of boundaries (greater than the largest axis) is equal to zero. In our implementation of the 2D decomposition we overwrite the local size variables to scale it to the number of the product of the widest two dimensions.

In the transposing task, the `fftw_mpi_plan_transpose` function used to transpose the global 3D grid over the contributors processors, in the 2D decomposition the transposing action is modified to divide the contributing processors to a set of sub groups that share the required transposing axis as shown in Figure 2. Similar related work in [11] implement the same idea.

Performance Application Programming Interface (PAPI) [14] framework supports several events API, PAPI 3.6.0 is used to measure the number of the floating points instructions needed to calculate the FFT for each processor, however PAPI does not support MPI intercommunication profiling. Integrated Profiling Monitoring IPM [15] is a portable profiling infrastructure for parallel codes. It provides performance profile from several aspects of the computation communication, and IO of the parallel programs, IPM used to measure the communication time acquired, the message size transferred, and the number of messages for the transposing tasks.

5 The analytical model

This paper aims to explore the most impacting parameters of the parallel FFT algorithms execution to build a high level model for its timing behaviour. The model takes into consideration the major hardware system parameters. However it is not hardware-specific as the more generic parameters we achieve, the more platforms we can use.

For simplicity we can divide the FFT algorithm into two non overlapping stages:

First: the computation part where each processor computes the DFT for the local portion of the data grid, the number of steps per core based on the algorithm order time (N^2, N log N, etc) and data mesh size. As shown in equation (2) the FFTW algorithm uses \( O(N \log N) \) [7], the computation time per core depends on several parameters.

The \( FFC \) parameter is the floating point factor that reflects the number of flops required for calculating each point. \( FFC \) equals 5 in case of complex number transform as in our experiments and 2.5 in case of real number transform [7], \( f(N_{procs}) \) is the maximum number of processors scalability considering the FFT algorithm decomposition method where \( f(N_{procs}) \) in 1D decomposition equals the widest dimension of the data grid. In the 2D dimension, it is equal to the product of the widest two dimensions of the data grid as shown in
equation (3). Peak Performance is the peak floating-point operations performance per core, the value of the peak performance equal to 9.2 GFLOPS/core or 128 GFLOPS/node [13]. Peak Percentage is the average peak performance percentage measured in our experiments and is found to be around 950 mflops that equivalent to 10% of the theoretical peak performance as shown in Figure 4, the mflops is the rate of flops × 10⁶ per second where flips stands for the number of floating points measured, Ndata is the number of the data grid points.

\[
\text{computation time} = \frac{f(N\text{procs})}{\text{Peak performance} \times \text{Peak percentage}}
\]

\[
f(N\text{procs}) = \begin{cases} 
\text{Max}(N_x, N_y, N_z), & 1D \\
(N_i \times N_j \text{AND } N_i, N_j \neq \text{Min}(N_x, N_y, N_z)), & 2D
\end{cases}
\]

Second: The communication part that represents the time consumed in the transposing tasks of the algorithm.

Again, the transposing time can be divided into two parts, first the data transmission time, as we have first to figure out the data grid size that will be transposed by each core by dividing the total data size Ndata on the number of processors f(Nprocs) that depends directly on the decomposition way multiplied by the number of bytes in each data grid point FFM, divided on the network bandwidth BW. The second part is the overall transpose Message latency MSGlat that equal to the summation of the three overhead parameters multiplied by the number of messages f(Nprocs) as in the transposing task, an all-to-all communication takes place.

The overhead parameters derived from three parameters l, o, and g. l refers to network latency, o overhead per message transmission and g gap between the successive message transmissions. The three parameters have been measured in previous work [12] as in Table 2 and Table 3, the tables address the latency values for the MPI blocking and non-blocking communication.

The blocking MPI means that the program execution will be suspended till the message buffer is safe to use, however in the non-blocking MPI communication, the call returns immediately after the call is initiated. The FFTW is found to use the non blocking MPI_SENDRECV routine in the transposing communications.

It is worthy to mention that the communication performance has many other parameters that are more hardware-specific; it is out of the scope of this model as the model is designed to be algorithm-oriented.

communication time = MSGlat + \frac{N_{\text{data}} \times f(N\text{procs})}{BW}
\]

\[MSGlat = (L + o + g) \times f(N\text{procs}) - 1\]

The number of points handled by each core can be calculated as N_{\text{data}} / f(N\text{procs}) multiplied by the number of bytes FFM.

6 Model Verification

This section discusses and compares the model and the algorithm performance and scalability.

In the following Figures, the performance results measured in seconds over 2 to 4096 cores, multiple runs executed to calculate the average execution time for 64³ and 256³ input grids, respectively.

As discussed in the previous Section, the model and the actual computation time are measured by calculating the number of flops of the FFT calculations for each core divided by the peak performance multiplied by %10 as the peak percentage.

The MPI_SENDRECV mpi routine used in point-to-point communication by the FFTW algorithm, the MPI Allreduce used to gather the results of the average number of flops through the running instances, the communication analysis include only the MPI_SENDRECV communication in the transposing tasks excluding any other communication overhead used in gathering or synchronizing the data.

The IPM used to profile the messages between the cores, as simple histogram method used to calculate the average message time between cores.

In our platform, TACC Ranger; some parameters should be addressed to submit a job, one of them is the wayness [13] as it determines the number of cores to be used through the whole node (16 cores), for example if the program will run on 32 core and 8 wayness method used, this means that the program will need 4 nodes and then only 8 cores will be used in each node.

As shown in Figure 3 each group of 4 cores are communicated with the other groups through one socket, the small number of processors (Nprocs < 16), the 1-way job scheduling wayness used to avoid measuring the inter-group communication instead of the InfiniBand socket communication.

6.1 1-D Decomposition

The model successfully predicts the overall trend changes in the FFT execution performance and scalability, Figure 5 and Figure 6 show that the time consumed by the computation part is dominant when the number of cores is less than 64, the communication and computation parts start to be very close when the number of cores are equal to the widest dimension over 2, the results confirms that algorithm scalability extent is equal to the widest dimension in the input 3D grid, when the number of cores is larger than the widest dimension the algorithm divides the data grid on a number of cores equals to the widest dimension and the rest of the cores is not involved in the execution plan.

6.2 2-D Decomposition

The model successfully predicts the overall performance behaviour and scalability, as shown in Figure 7 and Figure 8, the 2-D decomposition success to scale to the product of the widest two dimensions of the data grid as discussed in the previous Section, however the performance drops dramatically when the number of all-to-all messages increased to be more than 128×128 and
the size of the message became very small (less than 32 bytes) where the message latency time is the dominant. The message sizes transmitted for each scenario are shown in Figure 9 that shows the relationship between the message size of each scenario and the decomposition algorithm as well as the data grid size, message size is measured in bytes, the message size of the all-to-all communication step can be derived from the communication analytical equation as 

\[ \text{message size} = \frac{N_{\text{data}} + F_{\text{FM}}}{f(N_{\text{procs}})^2} \]

where the total amount of data required to be transposed on each processor is divided over the total number of the processors engaged in the all-to-all communication.

As shown in equation (5) the number of messages directly impact the time consumed by the messages latency, as shown in Figure 9 the dramatic increase of the messages number along the increase of the number of processors.

As shown in previous work [12] and in Figure 12 the relation between the message size and the G parameter - that introduced in Table 2 - has a Monotonic decrease relationship where the bandwidth affected badly by the message size.

To achieve the desired scalability and performance, the message size and the network latency factors should be considered.

In Figure 10 and Figure 11 the model accuracy for the gap between the predicted time and the measured time is shown for both 1-D and 2-D decompositions with 64\(^3\) and 256\(^3\) data grids inputs, the gap fluctuations may refer to change in the peak performance due to hardware considerations however the overall performance behaviour still predictable.

---

**Figure 4**: Average Number of mflops

**Figure 5**: 1D decomposition 256\(^3\) data grid size

**Figure 6**: 1D decomposition 64\(^3\) data grid size

**Figure 7**: 2D decomposition 64\(^3\) data grid size

**Figure 8**: 2D decomposition 256\(^3\) data grid size

**Figure 9**: Message size and number of messages
Figure 10: Gap between the Model and the Measured for 64³

Figure 11: Gap between the Model and the Measured for 256³

Figure 12: G for Various Message Sizes

7 Conclusions and future work

The paper presents a predictive model of scalability and performance of the parallel FFT algorithm. The model successfully predicts the overall scalability performance behavior, although there is a gap between the predicted and measured performance for the very large and the very small message size.

The model scope includes the execution of 3D FFT grid, 1D and 2D decomposition methods. A future work can be done against the 3D decomposition (volumetric decomposition).

The model can be enhanced to include the computation and communication overlapping probability.

The network parameters, as the bandwidth and the message latencies parameters should be known for the target platform; it can be done through 1 - System documentation 2 - profiling tools.

8 Acknowledgment

Our deep thanks go to Eric J. Bohm - Department of Computer Science - University of Illinois at Urbana-Champaign for his support and for providing HPC resources that have contributed to the research results reported within this paper.

9 References


