An MPI-based 3D FFT Implementation on CPU+GPU Heterogeneous Clusters

Kun Li, Yan Li, Ting Cao, Haipeng Jia, and Yunquan Zhang

SKL of Computer Architecture, ICT, CAS, Beijing 100190, China

Abstract

FFT(Fast Fourier Transform) is a fast algorithm to compute DFT，and it is also an important algorithm of data processing in SKA project. In the poster, the Auto-tuning FFT Framework on GPU and the FFT parallel library based on MPI are both introduced, and different FFT algorithms are evaluated on different platforms including TianHe-2.

1. Introduction

FFT is one of the key algorithm in SDP image processing. CPU+GPU heterogeneous clusters (such as Tianhe-IA and Dawning Nebulae) is a promising hardware platform for SKA project. So in our work, we prototyped an auto-tuning FFT Framework on GPU, and based on that, the parallel 3D FFT algorithm on CPU+GPU clusters through 2D grid decomposition are introduced. We will first introduce 1D decomposition and its deficiencies, and then focus on 2D decomposition and our implementation results.

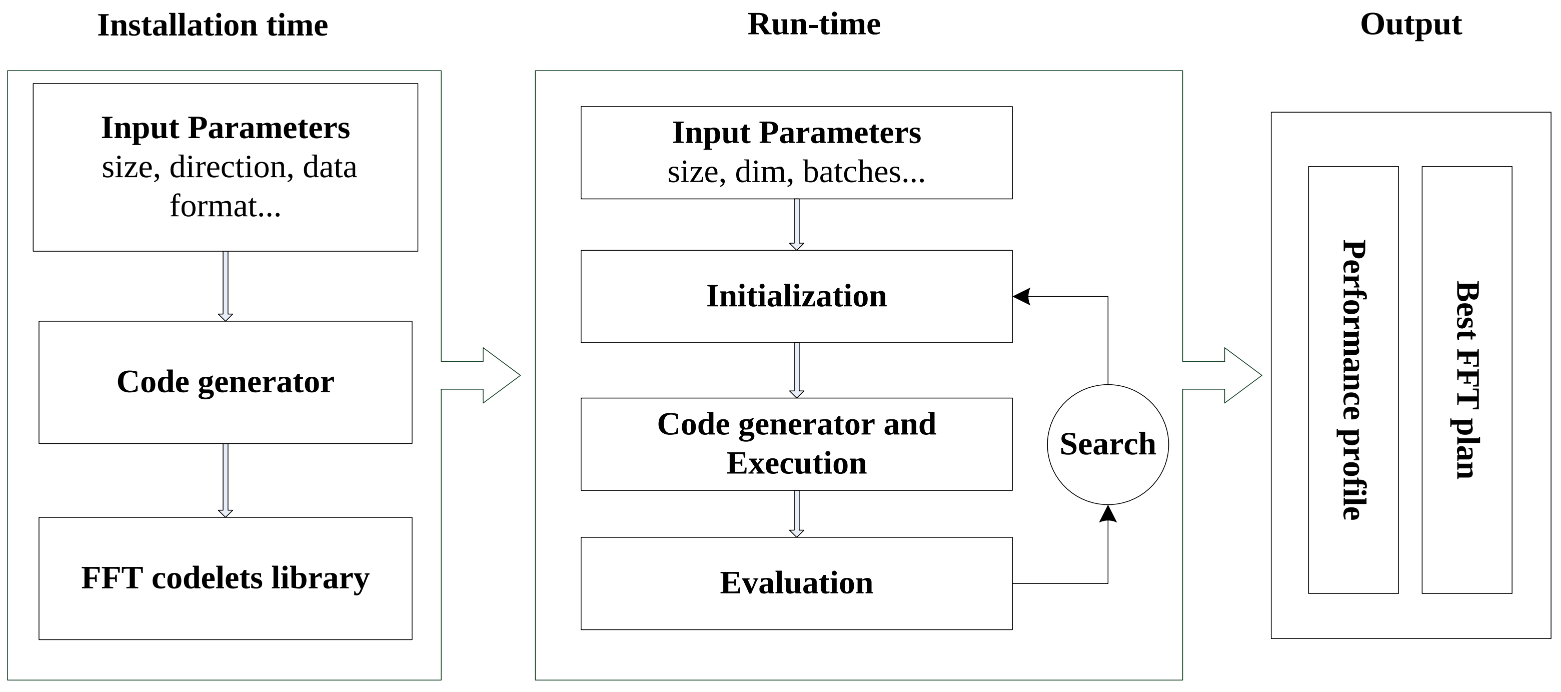
2. Auto-tuning FFT Framework on GPU

Currently, the general GPU FFT library is mostly NVIDIA CUFFT, and it couldn’t be portable on different platforms. The optimization work for Auto-tuning of Fast Fourier Transform based on CUDA is mainly on the following points:

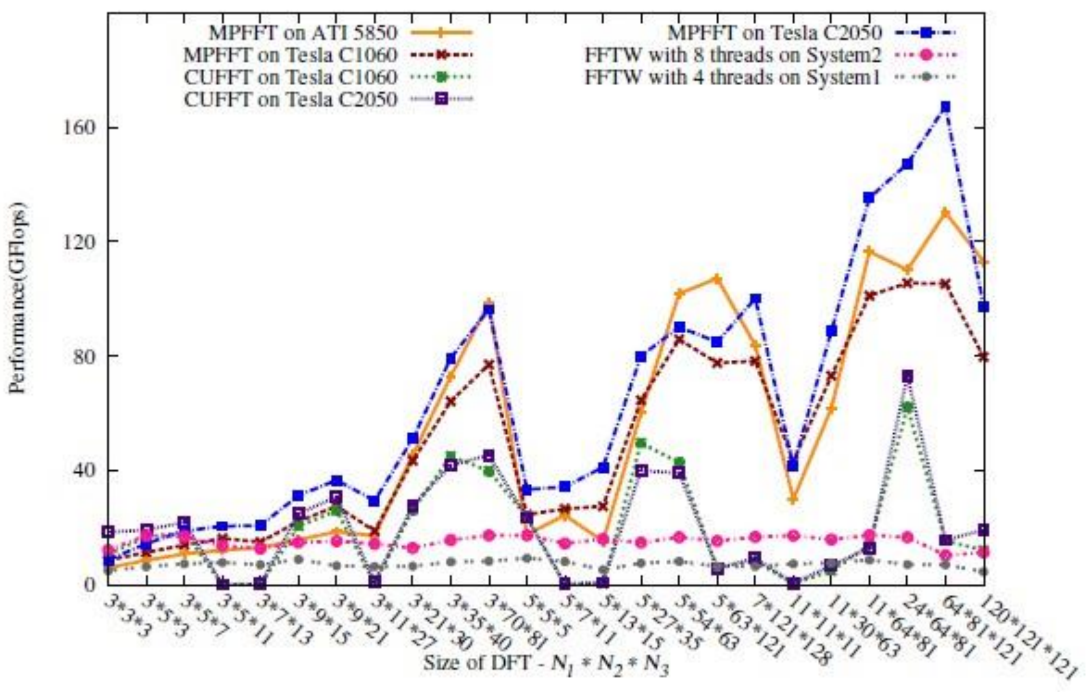
(1) Support the run-time optimization partly by cutting the thread blocks of heuristic search strategy.

(2) Combining with the characteristics of the GPU storage level to highly optimize one-dimensional small-scale FFT, the YZ dimensions of 3D are treated as one dimension to optimize respectively, and the main optimization work is focus on solving the access conflicts on shared memory of GPU. The Auto-tuning strategies are as follows:

a. Choose different decomposition bases and sequence
b. Memory-access optimizations
c. Adjust the thread blocks number


Installation time
Input Parameters
size, direction, data format...
Code generator
FFT codelets library
Run-time
Input Parameters
size, dim, batches...
Initialization
Code generator and Execution
Evaluation
Search
Output
Performance profile
Best FFT plan

(a) Auto-tuning FFT Framework



(b) The comparison result on different platforms

3. The FFT parallel library based on MPI

The research of 3D FFT on clusters is mainly based on 1D and 2D decomposition. We assume that the input size is $n_0 \times n_1 \times n_2$ ($n_0 \geq n_1 \geq n_2$), and the computing sequence will be n_2 , n_1, n_0.

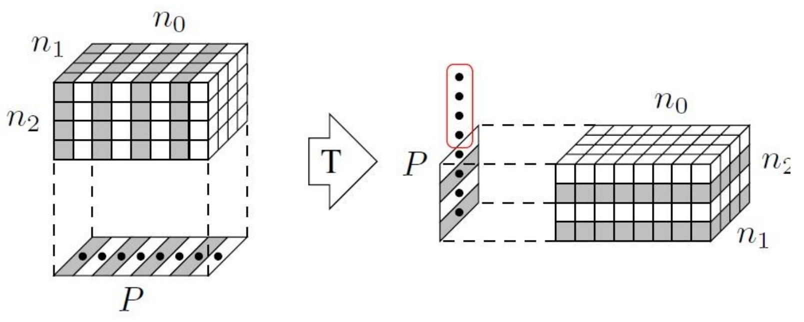
(1) 1D decomposition

a. Strategy

- Along n_0 dimension, divide data into P ($P \leq n$) processes.
- Secondly，P processes computes 2D FFT (size= $n_1 \times n_2$) of n_0/p batches parallelly.
- Because the data needed for the n_0 dimension are distributed in all processes, We need to do a full exchange of data on all processes to complete the computation of n_0 dimension.

b. bottleneck

- The scale of parallel computing is limited to n_1 or n_2.

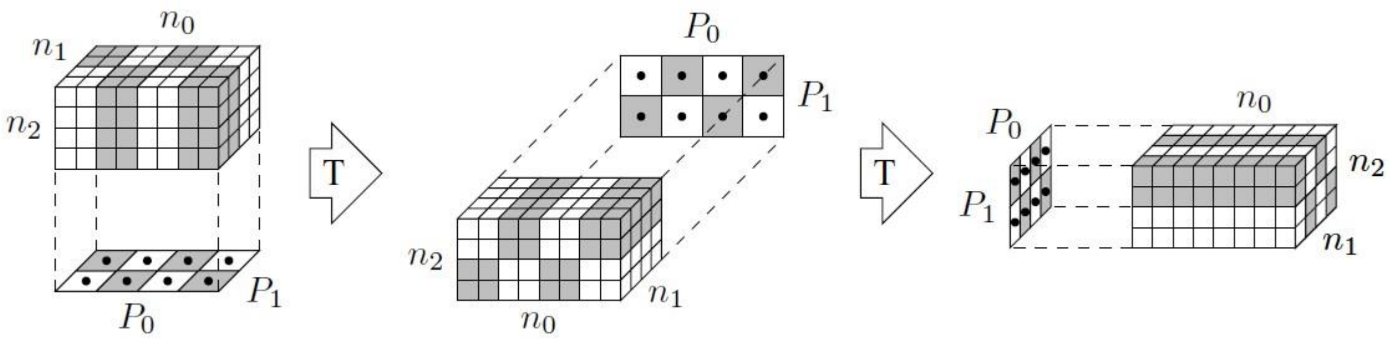


(c) 1D decomposition

(2) 2D decomposition

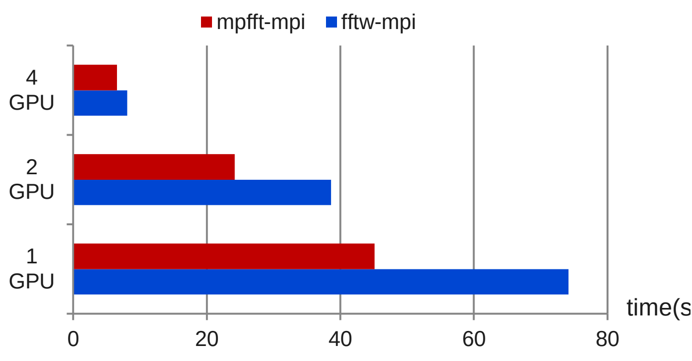
a. Strategy

- As Figure (d) shows, the decomposition is firstly along n_1 and n_0 to compute the data along n_2 parallelly. Then the decomposition is along n_0 and n2 to compute n_1. Finally the compute will be along n_0.

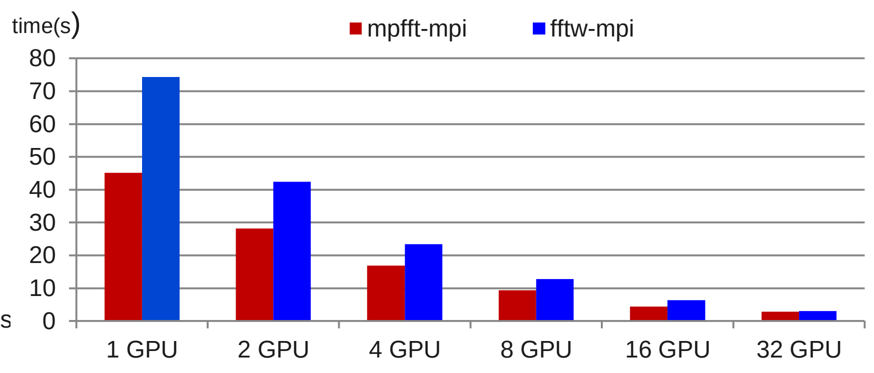


(d) 2D decomposition

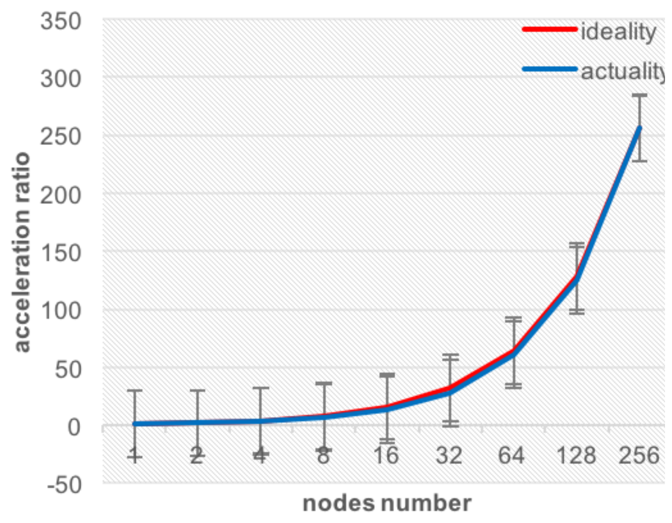
The parallel algorithm is evaluated in different environments, including TianHe-2. The comparison results are as follows:



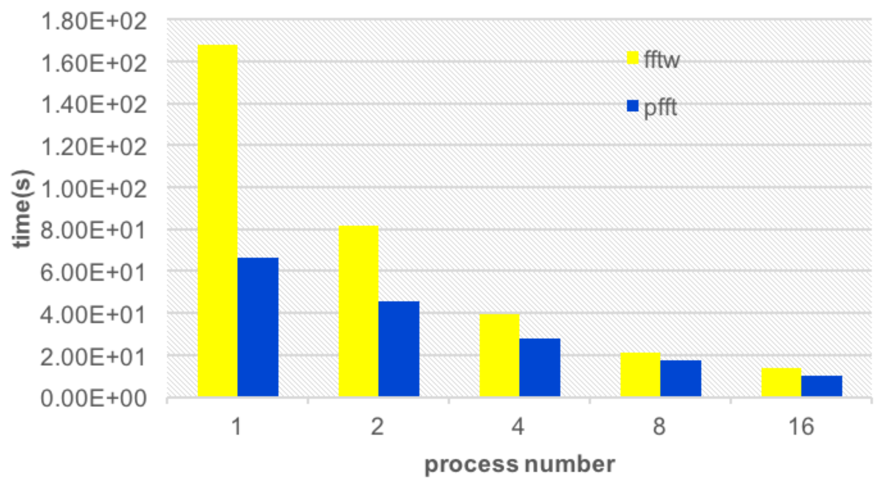
(e) QDR Infiniband



(f) Intel 82573 Gigabit Ethernet



(g) Tianhe2 (PFFT)



(h) QDR Infiniband

4. Conclusions and Future Work

The FFT parallel algorithms have been successfully evaluated in different platforms compared with FFTW. In the future, we will continue to enhance the scalability of FFT parallel algorithms to satisfy the higher requirement for data processing.

5. Reference

[1] Li Y, Zhang Y, Jia H, etal. Automatic FFT Performance Tuning on OpenCL GPUs.

[2] Li Y, Zhang Y. Auto-tuning FFT Optimization Framework on GPU and Virtualization Technology Research.

Shanghai, China , May 11-13, 2016
2016 SKA Workshop on SDP&HPC