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

Kun Li

Yan Li, Ting Cao, Haipeng Jia, Yunquan Zhang

SKL of Computer Architecture, ICT, CAS

Background

- **Basics**
  - FFT(Fast Fourier Transform)

    A fast algorithm to compute DFT，widely applied

    An important algorithm of data processing in SKA project

    Bandwidth-intensive task

    Considering the characteristic of matrix W_n caused by the periodicity of ω_N, the basic idea of FFT algorithm is to decompose matrix into many small matrix to calculate

Background

(a): $$y_k = \sum_{j=0}^{N-1} \omega_n^{jk} \cdot x_j \ (0 \le k < n) \quad \omega_n = e^{-\frac{2\pi i}{n}} (i = \sqrt{-1})$$

$$DFT_n = \begin{bmatrix} 1 & 1 & 1 & & 1 \\ 1 & \omega_n^1 & \omega_n^2 & \cdots & \omega_n^{n-1} \\ 1 & \omega_n^2 & \omega_n^4 & & \omega_n^{2(n-1)} \\ & \vdots & & \ddots & \vdots \\ 1 & \omega_n^{n-1} & \omega_n^{2(n-1)} & \cdots & \omega_n^{(n-1)(n-1)} \end{bmatrix}, y = DFT_n \cdot x$$

(b): $$j = (j_1, j_2) = j_1 \cdot r + j_2 \quad (0 \le j_1 < q, \ 0 \le j_2 < r)$$
$$k = (k_2, k_1) = k_1 + k_2 \cdot q \quad (0 \le k_1 < q, 0 \le k_2 < r)$$
$$n = q \cdot r$$

(c): $$y_k = \sum_{j_2=0}^{r-1} [(\sum_{j_1=0}^{q-1} \omega_q^{j_1 k_1} \cdot x_{j_1 \cdot r + j_2}) \cdot \omega_n^{j_2 k_1}] \cdot \omega_r^{j_2 k_2}$$

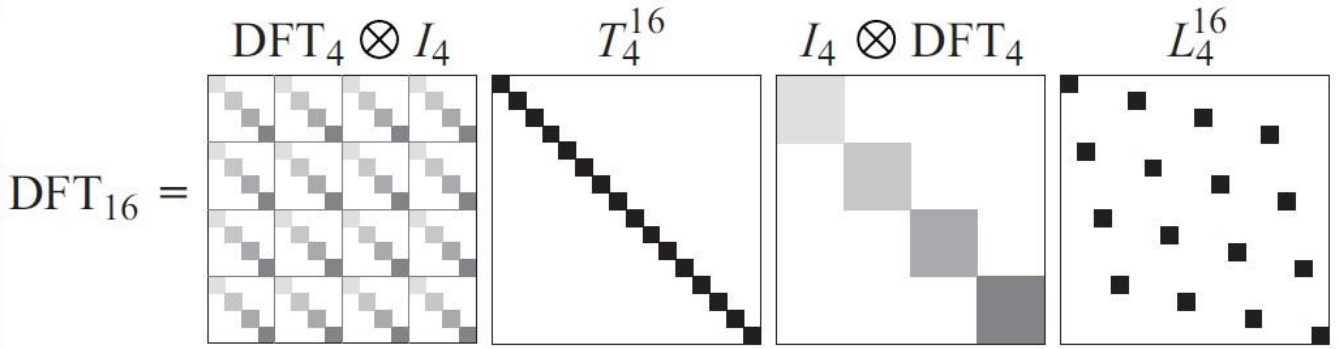
Background

(c): $$y_k = \sum_{j_2=0}^{r-1} [(\sum_{j_1=0}^{q-1} \omega_q^{j_1 k_1} \cdot x_{j_1 \cdot r + j_2}) \cdot \omega_n^{j_2 k_1}] \cdot \omega_r^{j_2 k_2}$$

$$A \otimes B = [a_{k,\ell} B], \quad \text{for } A = [a_{k,\ell}].$$

$$A \otimes B = \begin{pmatrix} a_{0,0}B & \cdots & a_{0,m-1}B \\ \vdots & \ddots & \vdots \\ a_{m-1,0}B & \cdots & a_{m-1,m-1}B \end{pmatrix}.$$

(d): $$DFT_n = (DFT_r \otimes I_q) T_q^n (I_r \otimes DFT_q) L_r^n$$



(a) Matrix factorization

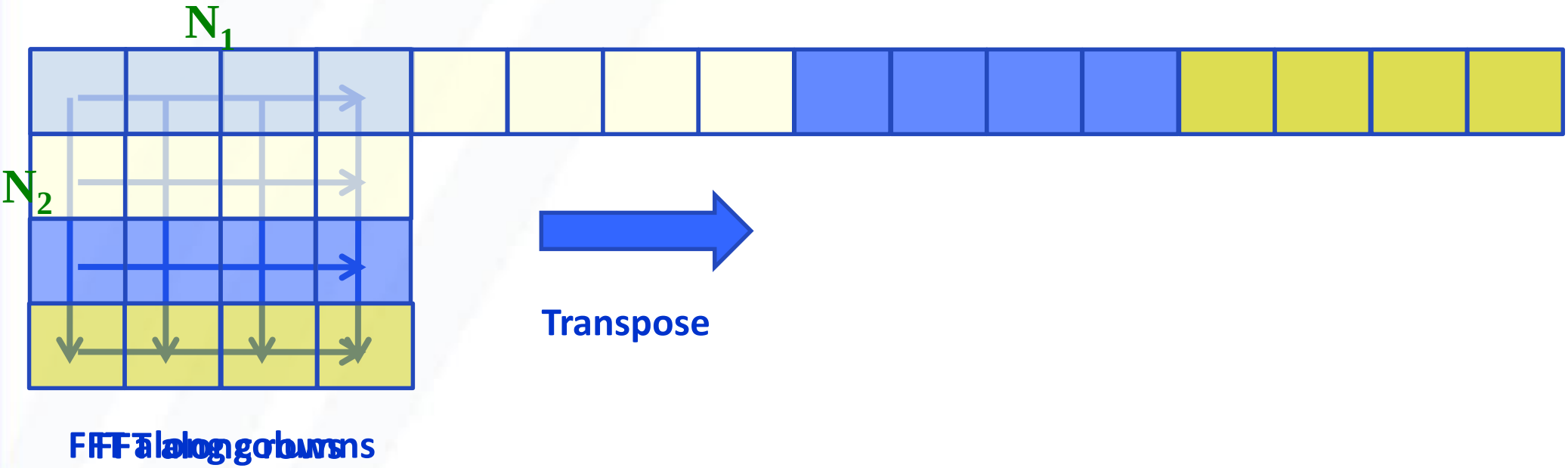
Background

- **General factorizations**
  - **FFT(Fast Fourier Transform)**

Example： $N=N_1 \times N_2 = 4 \times 4$

Background

- **General factorizations**
  - **FFT(Fast Fourier Transform)**



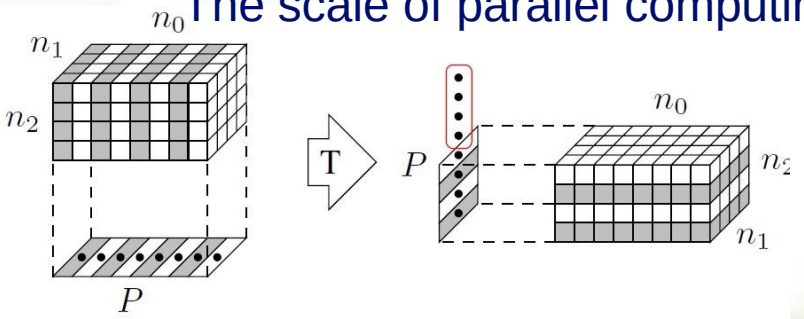
FFT on CPU+GPU Clusters

- **CPU+GPU** heterogeneous computing system has gradually become a hot research direction in the field of high performance computing.
  - Tianhe-IA
  - Dawning Nebulae
- **Possible platform candidate for SDP**
- Case study: 3D FFT on **heterogeneous clusters**
  - Application scenarios: astrophysical N-body simulations, blood flow simulations, etc.
  - Parallel algorithm: based on 1D and 2D decomposition

Algorithm Strategy

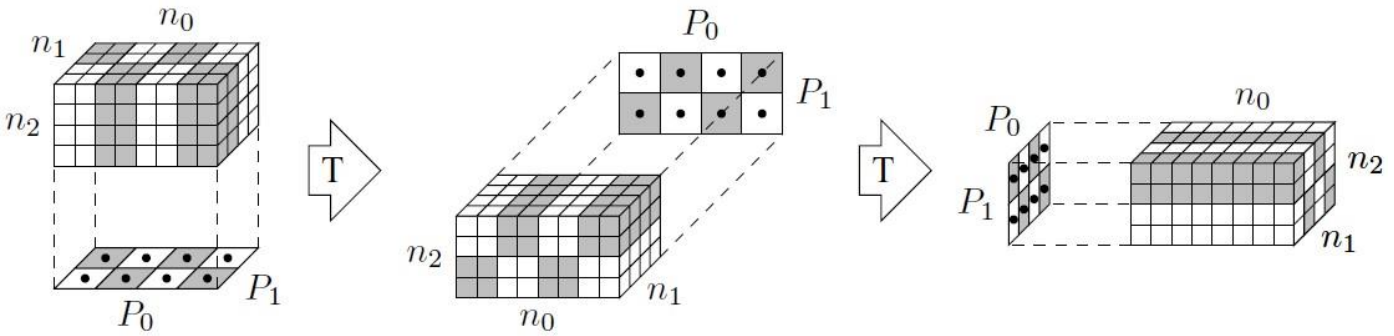
- **The research of 3D FFT on clusters is mainly based on 1D and 2D decomposition**
  - $n_0 \times n_1 \times n_2$ ($n_0 \geq n_1 \geq n_2$), Computing sequence： n_2 , n_1, n_0
  - **1D decomposition**
  - Strategy
    1. Along n_0 dimension, divide data into P($P \leq n$) processes.
    2. Secondly， P processes computes 2D FFT(size= $n_1 \times n_2$) of n_0/p batches parallelly.
    3. 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.
  - bottleneck

    The scale of parallel computing is limited to n_1 or n_2



Algorithm Strategy

- **The research of 3D FFT on clusters is mainly based on 1D and 2D decomposition**
  - $n_0 \times n_1 \times n_2$ $(n_0 \geq n_1 \geq n_2)$
  - **2D decomposition**
  - The number of processes allowed： $n_0 \times n_1$



Parallel Algorithm

According to the ratio of communication, algorithm is divided into 3 stages :

a: Initialize data distribution, and compute local multi-dimension FFT

for $t \leftarrow 0,\ldots,d\text{-}r\text{-}2$ **do**

$$h_0 \leftarrow \times_{s=0}^{r-1} \frac{N_s}{P_s} \times \times_{s=r}^{d-2-t} N_s$$

$$N \leftarrow N_{d-1-t}$$

$$h_1 \leftarrow \times_{s=d-t}^{d-1} \hat{N}_s$$

$$h_0 \times N \times h_1 \xrightarrow{\text{FFT}} h_0 \times \hat{N} \times h_1$$

end for (a)

Scale: $N = N_0 \times \ldots \times N_{d-1}$

Processes: $P = P_0 \times \ldots \times P_{r-1}$

$r < d$

$\times_{s=l}^{u} N_s = N_l \times \ldots \times N_u$

local FFT computation

- Call existing 1D FFT library

- **Example(using 2D process grid to compute 3D FFT)**
  - a. $\frac{n_0}{p_0} \times \frac{n_1}{p_1} \times n_2 \xrightarrow{FFT} \frac{n_0}{p_0} \times \frac{n_1}{p_1} \times \hat{n}_2$

Parallel Algorithm

According to the ratio of communication, algorithm is divided into 3 stages :

b: 1) local transposition

2) global transposition

3)local computation

T_0 :local transposition

T_1 :global transposition

MPI_AlltoAll

MPI_Alltoallv

for $t \leftarrow 0,\ldots,r\text{-}1$ **do**

$$h_0 \leftarrow \times_{s=r-t}^{r-1} \frac{\hat{N}_{s+1}}{P_s} \times \times_{s=r}^{r-t-1} \frac{N_s}{P_r}$$

$$N \leftarrow N_{r-t}$$

$$h_1 \leftarrow \times_{s=r+1}^{d-1} \hat{N}_s$$

$$h_0 \times N \times h_1 \xrightarrow{\text{FFT}} h_0 \times \hat{N} \times h_1$$

$$h_0 \times \hat{N} \times h_1 \xrightarrow{\text{T}} \hat{N} \times h_0 \times h_1$$

$$L_0 \leftarrow \hat{N}_{r-t}$$

$$h_0 \leftarrow \times_{s=r-t}^{r-1} \frac{\hat{N}_{s+1}}{P_s} \times \times_{s=0}^{r-t-2} \frac{N_s}{P_s}$$

$$L_1 = N_{r-t-1}$$

$$h_1 \leftarrow \times_{s=r+1}^{d-1} \hat{N}_s$$

$$P \leftarrow P_{r-t-1}$$

$$L_1 \times h_0 \times \frac{L_1}{P} \times h_1 \xrightarrow{\text{T}} \frac{L_0}{P} \times h_0 \times L_1 \times h_1$$

end for (b)

- **Example(using 2D process grid to compute 3D FFT)**

❖ b.

$$\frac{n_0}{p_0} \times \frac{n_1}{p_1} \times \hat{n}_2 \xrightarrow{T_0} \hat{n}_2 \times \frac{n_0}{p_0} \times \frac{n_1}{p_1} \xrightarrow{T_1} \frac{\hat{n}_2}{p_1} \times \frac{n_0}{p_0} \times n_1 \xrightarrow{FFT} \frac{\hat{n}_2}{p_1} \times \frac{n_0}{p_0} \times \hat{n}_1$$

$$\frac{\hat{n}_2}{p_1} \times \frac{n_0}{p_0} \times \hat{n}_1 \xrightarrow{T_0} \hat{n}_1 \times \frac{n_2}{p_1} \times \frac{n_0}{p_0} \xrightarrow{T_1} \frac{\hat{n}_1}{p_0} \times \frac{\hat{n}_2}{p_1} \times n_0 \xrightarrow{FFT} \frac{\hat{n}_1}{p_0} \times \frac{\hat{n}_2}{p_1} \times \hat{n}_0$$

Parallel Algorithm

According to the ratio of communication, algorithm is divided into 3 stages :

c: Do *r* local transposition and global transposition on the outputs of B to get the correct outputs.

$$h_1 \leftarrow \times_{s=r+1}^{d-1} \hat{N}$$

$$h_0 \leftarrow \times_{s=0}^{r-1} \frac{\hat{N}_{s+1}}{P_s}$$

$$N \leftarrow N_0$$

$$h_0 \times N \times h_1 \xrightarrow{\text{FFT}} h_0 \times \hat{N} \times h_1$$

for $t \leftarrow 0,\ldots,r\text{-}1$ **do**

$$L_1 = N_{r-t}$$

$$h_1 \leftarrow \times_{s=r-t}^{r-1} \frac{N_{s+1}}{P_s} \times \times_{s=0}^{r-t-2} \frac{N_s}{P_s}$$

$$L_0 \leftarrow N_{r-t-1}$$

$$h_0 \leftarrow \times_{s=r+1}^{d-1} N_s$$

$$P \leftarrow P_{r-t-1}$$

$$\frac{L_1}{P} \times h_1 \times L_0 \times h_0 \xrightarrow{\text{T}} L_1 \times h_1 \times \frac{L_0}{P} \times h_0$$

$$N \leftarrow N_{r-t}$$

$$N \times h_0 \times h_1 \xrightarrow{\text{T}} h_0 \times N \times h_1$$

end for

(c)

- **Example(using 2D process grid to compute 3D FFT)**
  - c.

$$\frac{n_1}{p_0} \times \frac{n_2}{p_1} \times n_0 \xrightarrow{T_1} n_1 \times \frac{n_2}{p_1} \times \frac{n_0}{p_1} \xrightarrow{T_0} \frac{n_2}{p_1} \times \frac{n_0}{p_1} \times n_1 \xrightarrow{T_1} n_2 \times \frac{n_0}{p_0} \times \frac{n_1}{p_1} \xrightarrow{T_0} \frac{n_0}{p_0} \times \frac{n_1}{p_1} \times n_2$$

Experimental Result

- **Implementation Strategy**
  - Local computation calls NVIDIA CUFFT.
  - Local transposition computes on GPU.

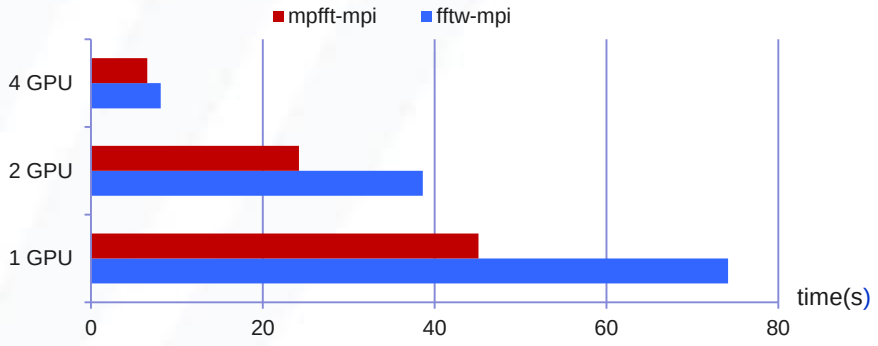
    When GPU computation completed, do transposition immediately to make use of high bandwidth of GPU effectively.
  - Global transposition uses encapsulated FFTW transposition functions.

Experimental Result

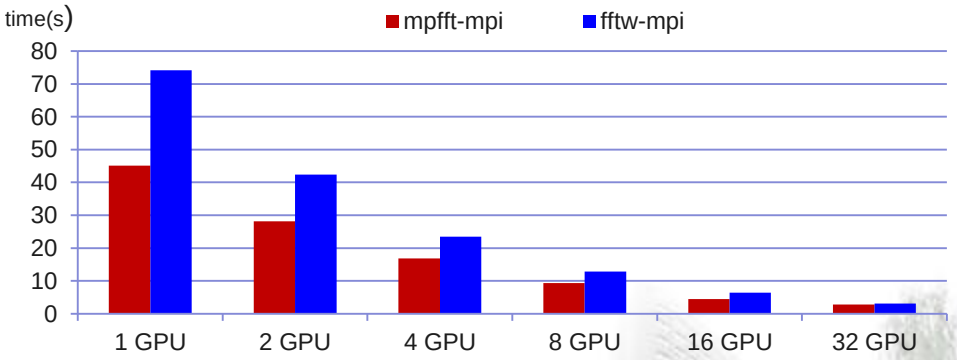
Experimental Platform

Platform	nodes	CPU	RAM	GCC	GPU	GPU SDK
A	2	Dual Intel E5430 2.66GHz	24G	4.4.3	Tesla C1060	CUDA 4.2
B	16	Dual Intel E5430 2.66GHz	24G	4.4.3	Tesla C1060	CUDA 4.2
network	A: QDR Infiniband			B: Intel 82573 Gigabit Ethernet		

Experimental Result



A: QDR Infiniband



B: Intel 82573 Gigabit Ethernet

Summary

- The basics of FFT
- FFT algorithm strategy：1D decomposition and 2D decomposition, and the analysis of scalability
- Illuminate the MPI parallel FFT algorithm
- FFT algorithm performance evaluation

Summary

Thank you!