An Efficient and Flexible Parallel FFT Implementation Based on FFTW

Michael Pippig

Faculty of Mathematics
Chemnitz University of Technology

22.06.2010

supported by BMBF grant 01IH08001B

Table of Contents

Discrete Fast Fourier Transform

Task of 3d-DFT

Consider a three-dimensional dataset of $n_0 \times n_1 \times n_2$ complex numbers $\hat{g}_{k_0 k_1 k_2} \in \mathbb{C}$. Compute

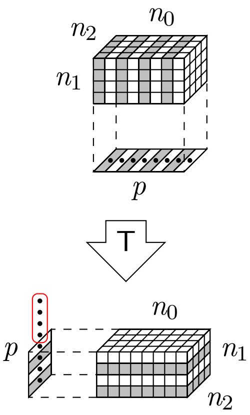
$$\begin{aligned} g_{l_0 l_1 l_2} &:= \sum_{k_0=0}^{n_0-1} \sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} \hat{g}_{k_0 k_1 k_2} \, \mathrm{e}^{-2\pi\mathrm{i}(k_2 \frac{l_2}{n_2} + k_1 \frac{l_1}{n_1} + k_0 \frac{l_0}{n_0})} \\ &= \sum_{k_0=0}^{n_0-1} \left(\sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} \hat{g}_{k_0 k_1 k_2} \, \mathrm{e}^{-2\pi\mathrm{i}(k_2 \frac{l_2}{n_2} + k_1 \frac{l_1}{n_1})} \right) \mathrm{e}^{-2\pi\mathrm{i} k_0 \frac{l_0}{n_0}} \end{aligned}$$

for all $l_t = 0, \ldots, n_t - 1$ $(t = 0, 1, 2)$.

Realized by 3d-FFT ($n_0 = n_1 = n_2 = n$)

$\Rightarrow \mathcal{O}(n^3 \log n)$ instead of $\mathcal{O}(n^6)$

One-Dimensional Data Distribution



p - number of processors

Features of FFTW [Frigo, Johnson]

- open source
- easy interface
- communicator
- arbitrary size
- d-dim. FFT
- in place FFT
- high performance
- many transforms
- adjust blocksize
- adjust planning
- collect wisdom

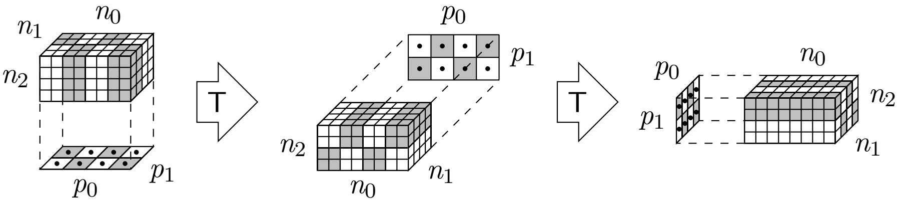
Maximum Number of Processors $p_{\max}^{1\mathrm{D}}$ $(n_0 = n_1 = n_2 = n)$

$$p_{\max}^{1\mathrm{D}} = n$$

FFTW combines portable performance and good usability, but is not scalable enough.

Two-Dimensional Data Distribution

[Ding, Eleftheriou et al. 03, Plimpton, Pekurovsky - P3DFFT]



$p_0 \times p_1$ - size of processor grid

Maximum Number of Processors p^{2D}_{max} $(n_0 = n_1 = n_2 = n)$

$$p^{2D}_{max} = n^2$$

n	$p^{1D}_{max} = n$	$p^{2D}_{max} = n^2$
64	64	4096
128	128	16384
256	256	65536
512	512	262144
1024	1024	1048576

Algorithms Supported by FFTW3.3alpha1

Aim

Implement a new parallel FFT sofware library (PFFT) based on FFTW and the highly scalable two-dimensional data distribution.

1d-FFT Combined with Local Transposition

$$
\begin{aligned}
\hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 \quad &\underset{012}{\overset{\text{FFT2}}{\longrightarrow}} \quad \hat{n}_0 \times \hat{n}_1 \times n_2 \\
\hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 \quad &\underset{102}{\overset{\text{FFT2}}{\longrightarrow}} \quad \hat{n}_1 \times \hat{n}_0 \times n_2 \\
\hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 \quad &\underset{021}{\overset{\text{FFT2}}{\longrightarrow}} \quad \hat{n}_0 \times n_2 \times \hat{n}_1 \\
(\hat{n}_0 \times \hat{n}_1) \times \hat{n}_2 \quad &\underset{201}{\overset{\text{FFT2}}{\longrightarrow}} \quad n_2 \times (\hat{n}_0 \times \hat{n}_1)
\end{aligned}
$$

Algorithms Supported by FFTW3.3alpha1

Transposition of One-Dimensional Distributed Data

$$N_0 \times \frac{N_1}{P} \quad \xrightarrow{\mathsf{T}} \quad \frac{N_0}{P} \times N_1$$

Group two of the three dimensions to use FFTWs matrix transposition on two-dimensional decomposed data, e.g. $N_0 = n_2, N_1 = n_0 \times \frac{n_1}{p_1}, P = p_0$.

Transposition of Two-Dimensional Distributed Data

$$n_2 \times \left(\frac{n_0}{p_0} \times \frac{n_1}{p_1}\right) \quad \xrightarrow{\mathsf{T}} \quad \frac{n_2}{p_0} \times \left(n_0 \times \frac{n_1}{p_1}\right)$$

Two-Dimensional Distributed FFT Based on FFTW

PFFT Forward Transform

$$\frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \times \hat{n}_2 \quad \underset{201}{\overset{\text{FFT2}}{\longrightarrow}} \quad n_2 \times \frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \quad \overset{\text{T}}{\longrightarrow}$$

$$\frac{n_2}{p_1} \times \frac{\hat{n}_0}{p_0} \times \hat{n}_1 \quad \underset{201}{\overset{\text{FFT2}}{\longrightarrow}} \quad n_1 \times \frac{n_2}{p_1} \times \frac{\hat{n}_0}{p_0} \quad \overset{\text{T}}{\longrightarrow}$$

$$\frac{n_1}{p_0} \times \frac{n_2}{p_1} \times \hat{n}_0 \quad \underset{102}{\overset{\text{FFT2}}{\longrightarrow}} \quad \frac{n_2}{p_1} \times \frac{n_1}{p_0} \times n_0$$

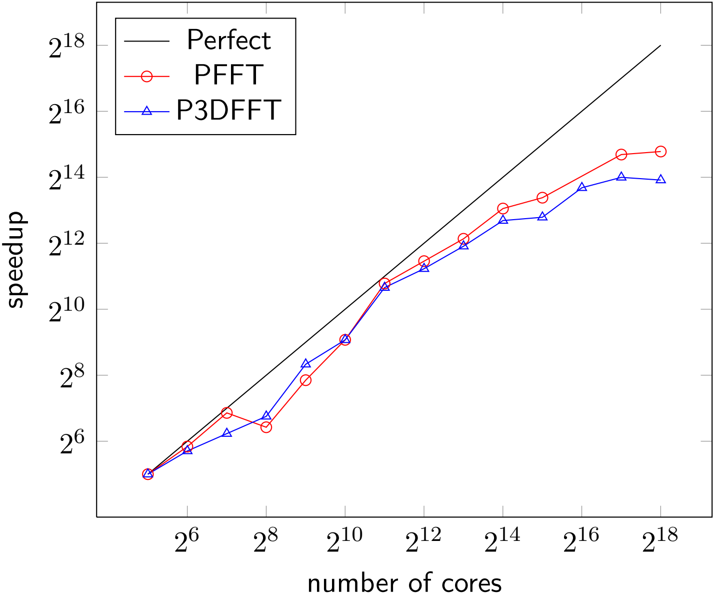
PFFT Backward Transform

$$\frac{n_2}{p_1} \times \frac{n_1}{p_0} \times n_0 \quad \underset{201}{\overset{\text{FFT2}}{\longrightarrow}} \quad \hat{n}_0 \times \frac{n_2}{p_1} \times \frac{n_1}{p_0} \quad \overset{\text{T}}{\longrightarrow}$$

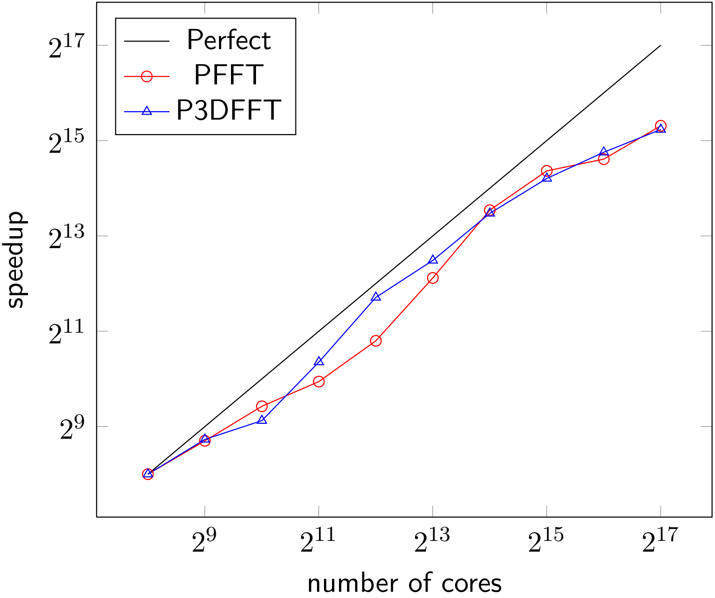
$$\frac{\hat{n}_0}{p_0} \times \frac{n_2}{p_1} \times n_1 \quad \underset{201}{\overset{\text{FFT2}}{\longrightarrow}} \quad \hat{n}_1 \times \frac{\hat{n}_0}{p_0} \times \frac{n_2}{p_1} \quad \overset{\text{T}}{\longrightarrow}$$

$$\frac{\hat{n}_1}{p_1} \times \frac{\hat{n}_0}{p_0} \times n_2 \quad \underset{102}{\overset{\text{FFT2}}{\longrightarrow}} \quad \frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \times \hat{n}_2$$

Scaling FFT of Size 512^3 on BlueGene/P



Scaling FFT of Size 1024^3 on BlueGene/P



Comparison of PFFT and P3DFFT

P3DFFT Unique Features [Pekurovsky]

- r2c FFT
- Fortran interface

Common Features

- open source
- high scalability
- portability
- multiple precisions
- C interface
- ghost cell support

PFFT Unique Features

- c2c FFT
- completely in place FFT
- FFTW like interface
- basic, advanced and guru interface
- adjustable blocksize
- separate communicator
- accumulated wisdom
- change of planning effort without recompilation
- d-dimensional parallel FFT
- truncated FFT support

Non-Equispaced Discrete Fourier Transform

Task of 3d-DFT and 3d-NDFT

For $\hat{f}_{k_0 k_1 k_2} \in \mathbb{C}$ compute

$$f_{l_0 l_1 l_2} := \sum_{k_0=0}^{N-1} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \hat{f}_{k_0 k_1 k_2} \, \mathrm{e}^{-2\pi \mathrm{i}(k_2 \frac{l_2}{N} + k_1 \frac{l_1}{N} + k_0 \frac{l_0}{N})} \qquad \text{(DFT)}$$

for all $0 \leq l_t < N$ ($\Rightarrow 0 \leq \frac{l_t}{N} < 1$), $t = 0, 1, 2$, and compute

$$f_j := \sum_{k_0=0}^{N-1} \sum_{k_1=0}^{N-1} \sum_{k_2=0}^{N-1} \hat{f}_{k_0 k_1 k_2} \, \mathrm{e}^{-2\pi \mathrm{i}(k_2 x_j^{(2)} + k_1 x_j^{(1)} + k_0 x_j^{(0)})} \qquad \text{(NDFT)}$$

for $x_j^{(t)} \in [0, 1)$ $(t = 0, 1, 2)$, $j = 1, \ldots, M$.

Realized by 3d-NFFT [NFFT software library]

$\Rightarrow \mathcal{O}(N^3 \log N + \log^3(\frac{1}{\varepsilon})M)$ instead of $\mathcal{O}(N^3 M)$

Parallel NFFT

Matrix-Vector-Notation

$$\boldsymbol{f} = \boldsymbol{A}\hat{\boldsymbol{f}},$$

where $\boldsymbol{f} = (f_j)_j$, $\boldsymbol{A} = (\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j})_{\boldsymbol{k},j}$, $\hat{\boldsymbol{f}} = (\hat{f}_{k_0 k_1 k_2})_{\boldsymbol{k}}$

Approximation [Dutt, Rohklin 93, Beylkin 95, Steidl 96, ...]

$$\boldsymbol{f} = \boldsymbol{A}\hat{\boldsymbol{f}} \approx \boldsymbol{B}\boldsymbol{F}\boldsymbol{D}\hat{\boldsymbol{f}},$$

where

- $\boldsymbol{D} \in \mathbb{C}^{N^3 \times N^3}$ diagonal matrix,
- $\boldsymbol{F} \in \mathbb{C}^{n^3 \times N^3}$ truncated Fourier matrix ($n \geq N$)
- $\boldsymbol{B} \in \mathbb{C}^{M \times n^3}$ sparse matrix

Summary

High Scalability

- 2d data distribution

FFTW Features

- performance
- interface
- portability

Additional Features

- truncated FFT
- ghost cells

PFFT

PNFFT