Introduction to Practical FFT and NFFT

Michael Pippig and Daniel Potts

Department of Mathematics
Chemnitz University of Technology

September 14, 2011

supported by BMBF grant 01IH08001B

Table of Contents

Outline

1. Serial FFT Algorithms
2. Parallel FFT Algorithms
3. Parallel Fast Summation

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

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

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

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

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

FFT Software Libraries

Examples of FFT implementations

- IBM ESSL
- Intel MKL
- FFTW

Features of FFTW [Frigo, Johnson]

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

Available at
`http://www.fftw.org`

Using FFTW

FFTW

Plan - only once

- hardware adaptive
- time consuming

⇓

Execute - several times

- fast transform

⇓

Finalize - only once

- free memory

FFTW_ESTIMATE

- heuristic choice of algorithm

FFTW_MEASURE

- compare different algorithms

FFTW_PATIENT

- compare more algorithms

FFTW_EXHAUSTIVE

- compare all available algorithms

FFTW Interface

Basic Interface

- simple
- do a single transform

Advanced Interface

- do a transform on multiple datasets by one call
- supports strided input and output
- use a plan on different datasets

Guru Interface

- most powerful and most complicated
- combine transform and data permutation
- 64 bit compatible

Nonequispaced 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 $0 \le l_t < N$ ($\Rightarrow 0 \le \frac{l_t}{N} < 1$), $t = 0, 1, 2$, 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 $\mathcal{O}(N^3 M)$

NFFT

Matrix-Vector-Notation

Compute $\boldsymbol{f} = (f_j)_j \in \mathbb{C}^M$ via

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

where $\boldsymbol{A} = (\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j})_{\boldsymbol{k},j} \in \mathbb{C}^{M \times N^3}$ and $\hat{\boldsymbol{f}} = (\hat{f}_{k_0 k_1 k_2})_{\boldsymbol{k}} \in \mathbb{C}^{N^3}$.

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

$$\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{F}\boldsymbol{D}, \qquad \boldsymbol{A}^{\mathsf{H}} \approx \boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{B}^T$$

where

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

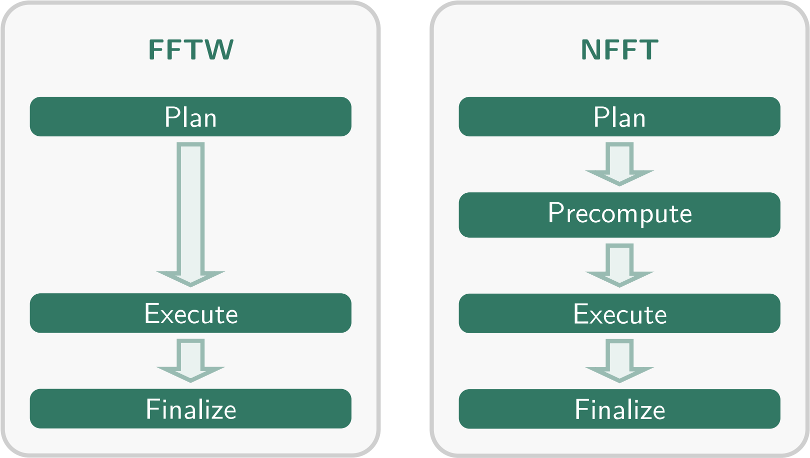
NFFT Software Library

NFFT Software Library [Keiner, Kunis, Potts]

Available at

`http://www.tu-chemnitz.de/~potts/nfft`

Using NFFT


FFTW
Plan
Execute
Finalize
NFFT
Plan
Precompute
Execute
Finalize

NFFT Precompute

PRE_FULL_PSI

- fully precomputed window function
- Storage: $(2m+2)^d M$, Computation: None

PRE_PSI

- tensor product based precomputation
- Storage: $d(2m+2)M$, Computation: $(2m+2)^d M$

PRE_LIN_PSI

- linear interpolation from lookup table
- Storage: dK, Computation: $2(2m+2)^d M$

PRE_FG_PSI

- fast Gaussian gridding
- Storage: $2dM$, Computation: $(2m+2)^d M$

Outline

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

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

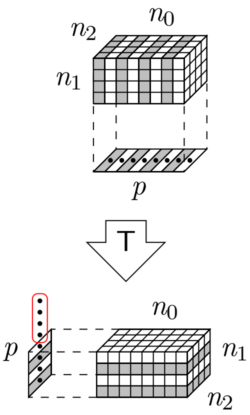
$$= \sum_{k_0=0}^{n_0-1} \left(\sum_{k_1=0}^{n_1-1} \sum_{k_2=0}^{n_2-1} \hat{g}_{k_0 k_1 k_2} \, \mathrm{e}^{-2\pi \mathrm{i} (k_2 \frac{l_2}{n_2} + k_1 \frac{l_1}{n_1})} \right) \mathrm{e}^{-2\pi \mathrm{i} k_0 \frac{l_0}{n_0}}$$

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

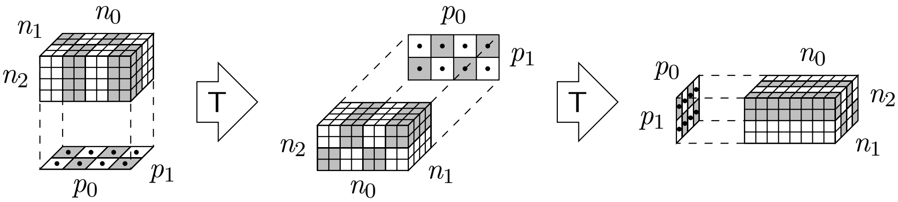
Maximum Number of Processors $p_{\max}^{\text{1D}}$ $(n_0 = n_1 = n_2 = n)$

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

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 $p_{\max}^{\mathbf{2D}}$ ($n_0 = n_1 = n_2 = n$)

$$p_{\max}^{2D} = n^2$$

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

Algorithms Supported by FFTW3.3

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

$$\hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 \quad \underset{012}{\overset{\text{FFT2}}{\longrightarrow}} \quad \hat{n}_0 \times \hat{n}_1 \times n_2$$

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

$$\hat{n}_0 \times \hat{n}_1 \times \hat{n}_2 \quad \underset{021}{\overset{\text{FFT2}}{\longrightarrow}} \quad \hat{n}_0 \times n_2 \times \hat{n}_1$$

$$(\hat{n}_0 \times \hat{n}_1) \times \hat{n}_2 \quad \underset{201}{\overset{\text{FFT2}}{\longrightarrow}} \quad n_2 \times (\hat{n}_0 \times \hat{n}_1)$$

Algorithms Supported by FFTW3.3

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

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

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

$$\frac{n_1}{p_0} \times \frac{n_2}{p_1} \times \hat{n}_0 \quad \underset{012}{\overset{\mathsf{FFT2}}{\longrightarrow}} \quad \frac{n_1}{p_0} \times \frac{n_2}{p_1} \times n_0$$

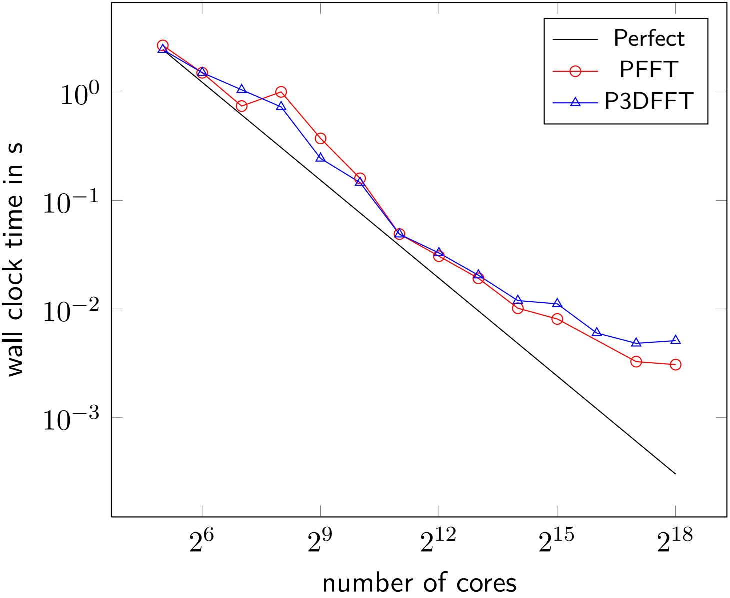
PFFT Backward Transform

$$\frac{n_1}{p_0} \times \frac{n_2}{p_1} \times n_0 \quad \underset{012}{\overset{\mathsf{FFT2}}{\longrightarrow}} \quad \frac{n_1}{p_0} \times \frac{n_2}{p_1} \times \hat{n}_0 \quad \overset{\mathsf{T}}{\longrightarrow}$$

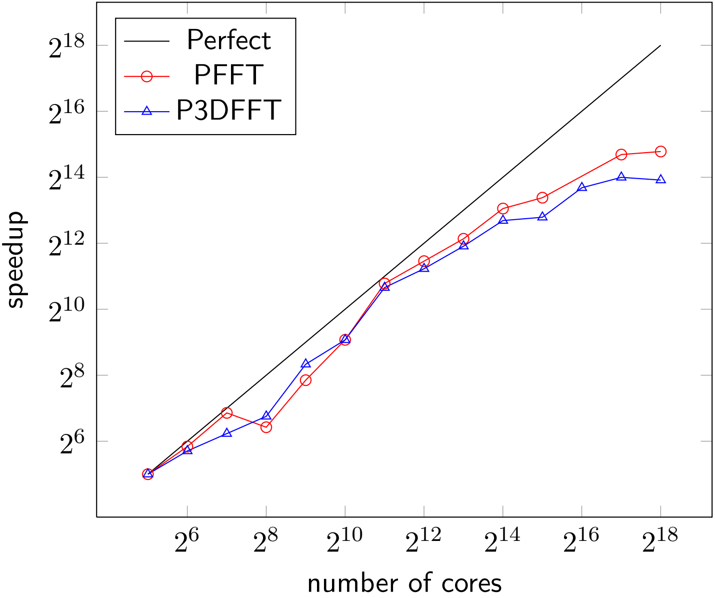
$$n_1 \times \frac{n_2}{p_1} \times \frac{\hat{n}_0}{p_0} \quad \underset{120}{\overset{\mathsf{FFT0}}{\longrightarrow}} \quad \frac{n_2}{p_1} \times \frac{\hat{n}_0}{p_0} \times \hat{n}_1 \quad \overset{\mathsf{T}}{\longrightarrow}$$

$$n_2 \times \frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \quad \underset{120}{\overset{\mathsf{FFT0}}{\longrightarrow}} \quad \frac{\hat{n}_0}{p_0} \times \frac{\hat{n}_1}{p_1} \times \hat{n}_2$$

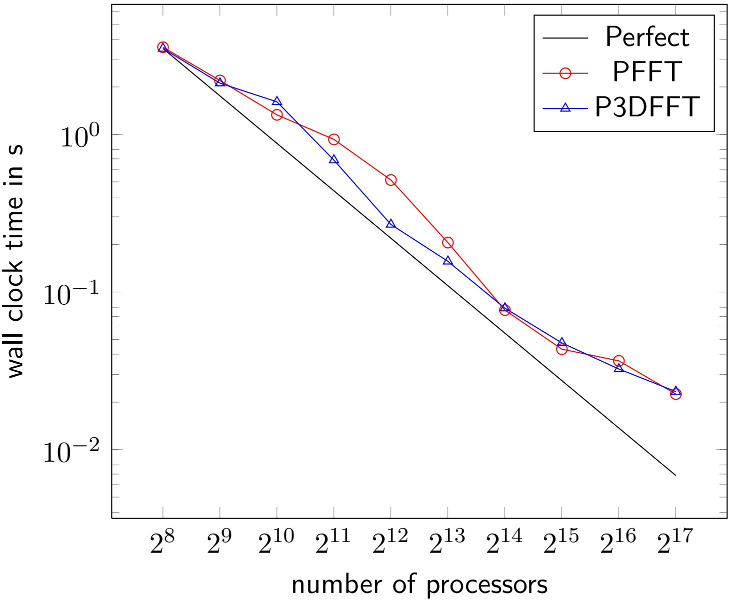
Scaling FFT of Size 512^3 on BlueGene/P



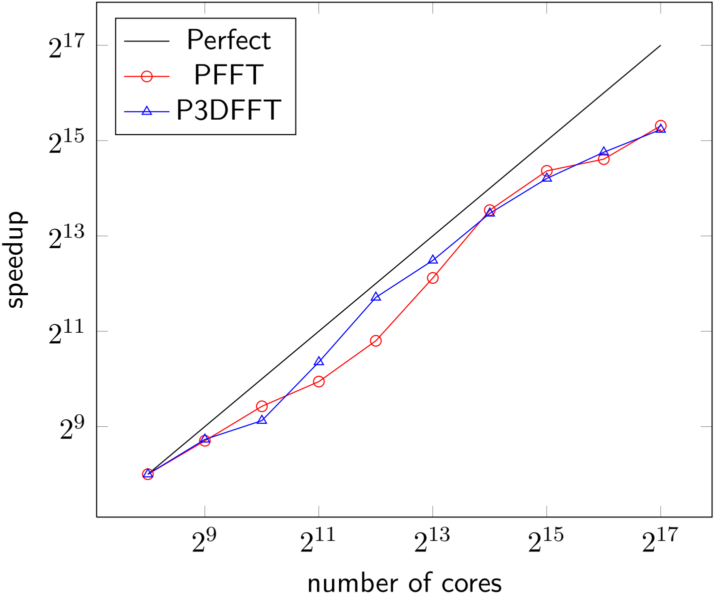
Scaling FFT of Size 512^3 on BlueGene/P



Scaling FFT of Size 1024^3 on BlueGene/P



Scaling FFT of Size 1024^3 on BlueGene/P



Comparison of PFFT and P3DFFT

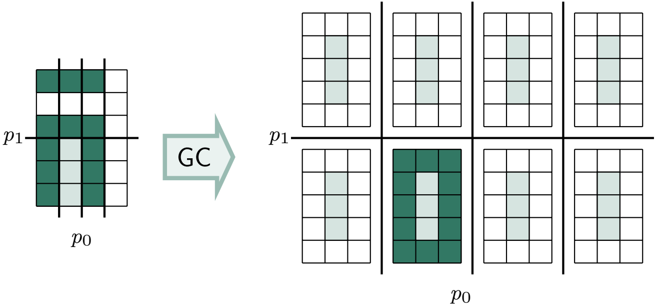
Common Features

- open source
- high scalability
- portability
- multiple precisions
- Fortran interface
- C interface
- r2c FFT
- ghost cell support

PFFT Unique Features

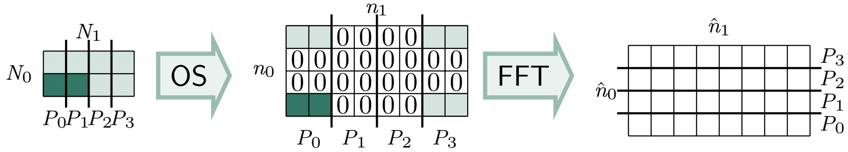
- c2c FFT
- completely in place FFT
- FFTW like interface (basic, advanced and guru)
- adjustable blocksize
- separate communicator
- accumulated wisdom
- change of planning effort without recompilation
- d-dimensional parallel FFT
- over- and downsampling

Ghost Cell Support

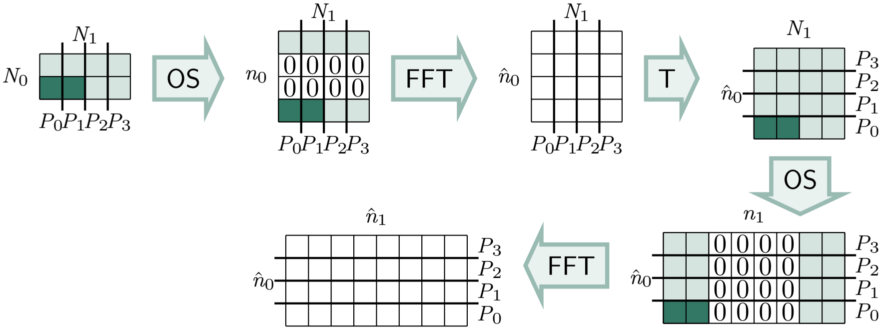


Oversampled & Downsampled FFT

Without Library Support



PFFT Library Support



PFFT Software Library

PFFT Software Library [Pippig]

Available at

`http://www.tu-chemnitz.de/~mpip/software`

Parallel NFFT

Matrix-Vector-Notation

Compute $\boldsymbol{f} = (f_j)_j \in \mathbb{C}^M$ via

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

where $\boldsymbol{A} = (\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j})_{\boldsymbol{k},j} \in \mathbb{C}^{M\times N^3}$ and $\hat{\boldsymbol{f}} = (\hat{f}_{k_0 k_1 k_2})_{\boldsymbol{k}} \in \mathbb{C}^{N^3}$.

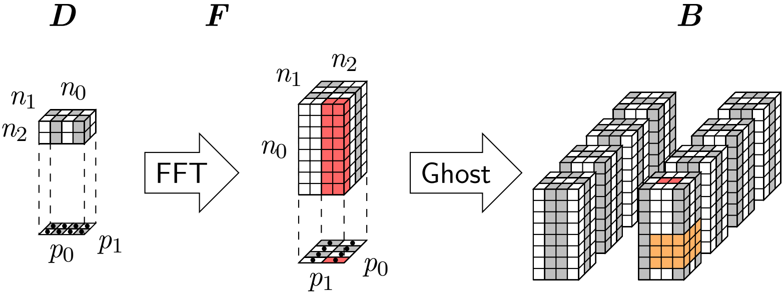
Approximation [Dutt, Rohklin 93, Beylkin 95, Steidl 96, ...]

$$\boldsymbol{A} \approx \boldsymbol{B}\boldsymbol{F}\boldsymbol{D}, \qquad \boldsymbol{A}^{\mathsf{H}} \approx \boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{B}^{T}$$

where

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

PNFFT in Pictures



PNFFT Software Library

PNFFT Software Library [Pippig]

Available at

`http://www.tu-chemnitz.de/~mpip/software`

Outline

Fast Summation

Task

Fast computation of

$$h_j = \sum_{l=1}^{L} a_l K(\|\boldsymbol{y}_j - \boldsymbol{x}_l\|_2), \quad j = 1, \dots, M, \quad \boldsymbol{y}_j, \boldsymbol{x}_l \in \mathbb{R}^3$$

Example of Radial Kernel Function

$$K(\|\boldsymbol{x}\|_2) = \frac{1}{\|\boldsymbol{x}\|_2}, \dots$$

Realized with 3d-NFFT [NFFT Software Library]

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

Parallel Fast Summation

Matrix-Vector-Notation

Compute $\boldsymbol{h} = (h_j)_{j=1}^{M}$ via

$$\boldsymbol{h} = \boldsymbol{K}\boldsymbol{a},$$

where $\boldsymbol{K} = (K(\|\boldsymbol{y}_j - \boldsymbol{x}_l\|))_{j,l=1}^{M,M}$ and $\boldsymbol{a} = (a_l)_{l=1}^{M} \in \mathbb{C}^M$.

Standard Algorithm for Equispaced Nodes

$$\boldsymbol{K} = \boldsymbol{F}\boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}$$

where

- $\boldsymbol{F} \in \mathbb{C}^{M \times M}$ equispaced Fourier matrix
- $\boldsymbol{D} \in \mathbb{C}^{M \times M}$ diagonal matrix

Parallel Fast Summation

Matrix-Vector-Notation

Compute $\boldsymbol{h} = (h_j)_{j=1}^{M}$ via

$$\boldsymbol{h} = \boldsymbol{K}\boldsymbol{a},$$

where $\boldsymbol{K} = (K(\|\boldsymbol{y}_j - \boldsymbol{x}_l\|))_{j,l=1}^{M,L}$ and $\boldsymbol{a} = (a_l)_{l=1}^{L} \in \mathbb{C}^L$.

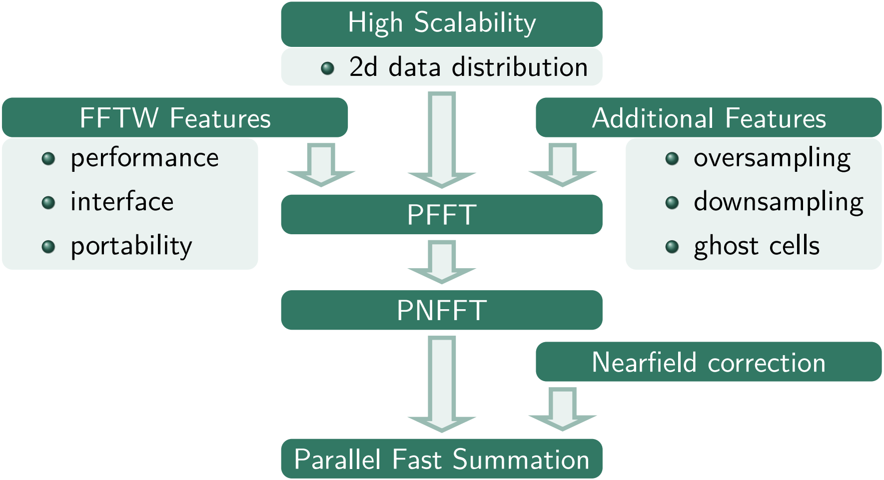
Approximation [Potts, Steidl, Nieslony 2004]

$$\boldsymbol{K} \approx \boldsymbol{A}_2 \boldsymbol{D} \boldsymbol{A}_1^{\mathsf{H}} + \boldsymbol{K}_{NF}$$

where

- $\boldsymbol{A}_2 = \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{y}_j}\right)_{j,\boldsymbol{k}} \in \mathbb{C}^{M \times N^3}$ nonequispaced Fourier matrix
- $\boldsymbol{D} \in \mathbb{C}^{N^3 \times N^3}$ diagonal matrix
- $\boldsymbol{A}_1 = \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_l}\right)_{l,\boldsymbol{k}} \in \mathbb{C}^{L \times N^3}$ nonequispaced Fourier matrix
- $\boldsymbol{K}_{NF} \in \mathbb{C}^{M \times L}$ sparse near field correction

Summary



High Scalability
2d data distribution
FFTW Features
performance
interface
portability
Additional Features
oversampling
downsampling
ghost cells
PFFT
PNFFT
Nearfield correction
Parallel Fast Summation