PARALLEL THREE-DIMENSIONAL NONEQUISPACED FAST FOURIER TRANSFORMS AND THEIR APPLICATION TO PARTICLE SIMULATION

MICHAEL PIPPIG AND DANIEL POTTS

Abstract. In this paper we describe a parallel algorithm for calculating nonequispaced fast Fourier transforms on massively parallel distributed memory architectures. These algorithms are implemented in an open source software library called PNFFT. Furthermore, we derive a parallel fast algorithm for the computation of the Coulomb potentials and forces in a charged particle system, which is based on the parallel nonequispaced fast Fourier transform. To prove the high scalability of our algorithms we provide performance results on a BlueGene/P system using up to 65536 cores.

Key words and phrases: parallel nonequispaced fast Fourier transform, parallel fast summation, parallel particle mesh methods, NFFT
2000 AMS Mathematics Subject Classification : 65T50 65Y05

1. Introduction. A broad variety of mathematical algorithms and applications depends on the calculation of the nonequispaced discrete Fourier transform (NDFT), which is a generalization of the discrete Fourier transform to nonequispaced nodes. Especially its fast approximate realization called nonequispaced fast Fourier transform (NFFT) [7, 2, 43, 46, 42, 18, 27] multiplied the application areas, since it led to fast algorithms in computerized tomography [14, 8], particle simulation [40, 21] and spectral methods on adaptive grids, just to name a few examples. An extensive list of applications can be found e.g. in [18].

Roughly speaking, the nonequispaced fast Fourier transform consists of three steps. First, a deconvolution in Fourier space. Second, a fast Fourier transform (FFT) and, finally, a discrete convolution in spatial domain. The deconvolution and convolution is performed with a window function that is well localized in Fourier and spatial domain. Therefore, the convolution steps can be performed approximately in a fast way. Another advantage of the good localization is, that parallel implementations of the convolution steps only require next neighbor communication.

The FFT plays central role in the modular structure of the NFFT algorithm and is a perfect example for the important interplay between the development of fast algorithms and sustainable software engineering in order to produce high performance software. Without a doubt, the FFTW software library [16, 17] is an outstanding implementation of the fast Fourier transform and one of the most important software packages in scientific computing. It offers support of shared memory parallelism and also distributed memory parallelism based on a one-dimensional decomposition of the input array. However, it has been argued that the one-dimensional data decomposition is not scalable enough for modern massive parallel distributed memory architectures [5, 10, 12, 44]. Whenever the dimensionality of the input array is at least three, a more scalable two-dimensional domain decomposition can be applied. Several publicly available parallel FFT software libraries [39, 34, 32, 31, 37, 35] based on this approach have been proposed during the last years.

Following the example of the FFTW software library, the NFFT algorithm has been implemented in the publicly available NFFT software library [27, 26], which also offers support of shared memory parallelism [45] and a parallel implementation for

{michael.pippig,potts}@mathematik.tu-chemnitz.de,
Chemnitz University of Technology, Department of Mathematics, 09107 Chemnitz, Germany

graphic processing units [28]. However, to our knowledge there is no publicly available implementation of the NFFT algorithm based on distributed memory parallelism. The algorithms in this paper and their publicly available implementations are intended to close the gap between NFFT and modern distributed memory architectures.

In this paper we propose a parallel algorithm for computing the NFFT on massively parallel distributed memory architectures. This algorithm strongly requires the parallel pruned FFT [37] in order to overcome severe load balancing problems. Our highly scalable implementation is based on the Message Passing Interface [33] and utilizes the PFFT software library [35] for computing the pruned FFT in parallel. Furthermore, we describe a massively parallel fast summation algorithm based on the parallel NFFT. The fast summation algorithm [40, 41] deals with the computation of Coulomb interactions in charged particle systems with non-periodic boundary conditions. This is similar to Ewald-like particle-mesh algorithms, which only work for periodic boundary conditions, see e.g. [19, Ch. 7] for an overview. Indeed, in [38] we point out that the building blocks of the fast summation [40, 41] for non-periodic boundary conditions and the fast Ewald summation [21] for periodic boundary conditions are very similar. Especially, both algorithms employ the NFFT in order to achieve a fast algorithm. To our knowledge, this is the first paper that presents results on distributed memory parallelization of particle-mesh algorithms with non-periodic boundary conditions. A numerical comparison with parallel implementation of other methods, e.g. FMM [25], will be published elsewhere.

The outline of this paper is as follows: We start in Section 2 with the introduction of the notation and definitions that we will use in the remainder of the paper. Next, we give the definition of the nonequispaced discrete Fourier transform and explain the basic principles of the NFFT in Section 3. In Section 4 we present our parallel NFFT algorithm, which we apply in Section 5 to develop a parallel algorithm for computing the Coulomb potentials and fields of a charged particle system. Section 6 contains performance evaluations of our publicly available, parallel implementation. Finally, we conclude the paper in Section 7.

2. Notation, Definitions and Assumptions.

In this section we introduce the notation, basic definitions and assumptions that are used throughout the entire paper. Assume the multi-bandwidth $\boldsymbol{N} = (N_0, N_1, N_2)^\top \in 2\mathbb{N}^3$. We define the multi-index set of possible frequencies $\mathcal{I}_{\boldsymbol{N}} := \left\{-\frac{N_0}{2}, \dots, \frac{N_0}{2} - 1\right\} \times \left\{-\frac{N_1}{2}, \dots, \frac{N_1}{2} - 1\right\} \times \left\{-\frac{N_2}{2}, \dots, \frac{N_2}{2} - 1\right\}$, the total number of frequencies $|\mathcal{I}_{\boldsymbol{N}}| = N_0 \cdot N_1 \cdot N_2$ and the three-dimensional torus $\mathbb{T}^3 := \mathbb{R}^3/\mathbb{Z}^3 \sim [-\frac{1}{2}, \frac{1}{2})^3$. For $|\mathcal{I}_{\boldsymbol{N}}|$ complex numbers $\hat{f}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N}}$, the trigonometric polynomial $f \colon \mathbb{T}^3 \to \mathbb{C}$ is given by

$$f(\boldsymbol{x}) = \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N}}} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}}. \tag{2.1}$$

The fast evaluation of f at arbitrarily chosen nodes $\boldsymbol{x}_j \in \mathbb{T}^3$, $j = 1, \dots, M$, with $M \in \mathbb{N}$, i.e.,

$$f_j := f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N}}} \hat{f}_{\boldsymbol{k}} \mathrm{e}^{-2\pi \mathrm{i} \boldsymbol{k}\boldsymbol{x}_j}, \quad j = 1, \dots, M, \tag{2.2}$$

is known as three-dimensional nonequispaced fast Fourier transform (NFFT). Equation (2.2) can be written as a matrix-vector product,

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

with the vectors $\boldsymbol{f} := (f_j)_{j=1,\dots,M} \in \mathbb{C}^M$, $\hat{\boldsymbol{f}} := (\hat{f}_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}} \in \mathbb{C}^{|\mathcal{I}_{\boldsymbol{N}}|}$, and the nonequispaced Fourier matrix $\boldsymbol{A} := \left(\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\right)_{j=1,\dots,M;\, \boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}} \in \mathbb{C}^{M\times|\mathcal{I}_{\boldsymbol{N}}|}$. For clarity of presentation the multi-index $\boldsymbol{k}$ addresses elements of vectors and matrices as well. In general, the matrix $\boldsymbol{A}$ is not square. Even for the square case, it is usually not orthogonal. Therefore, the definition of an inverse NFFT is not canonical, but can be realized by an iterative method, see e.g. [29]. Instead, it is customary to define the *adjoint transform* by the matrix-vector product

$$\hat{\boldsymbol{h}} = \boldsymbol{A}^{\mathsf{H}}\boldsymbol{f},$$

that is equivalent to the sums

$$\hat{h}_{\boldsymbol{k}} = \sum_{j=1}^{M} f_j \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}, \qquad \boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}, \tag{2.3}$$

with the vector $\hat{\boldsymbol{h}} := (\hat{h}_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}}$. In addition, we are interested in the fast calculation of the gradients

$$\nabla f_j := \nabla f(\boldsymbol{x}_j) = \sum_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}} \hat{f}_{\boldsymbol{k}} \nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}, \quad j = 1,\dots,M. \tag{2.4}$$

Equation (2.4) can be written as a matrix-vector product,

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

with the vectors $\nabla\boldsymbol{f} := (\nabla f_j)_{j=1,\dots,M} \in \mathbb{C}^{3M}$, $\hat{\boldsymbol{f}} := (\hat{f}_{\boldsymbol{k}})_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}} \in \mathbb{C}^{|\mathcal{I}_{\boldsymbol{N}}|}$, and the matrix $\nabla\boldsymbol{A} := \left(\nabla\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\right)_{j=1,\dots,M;\, \boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}} \in \mathbb{C}^{3M\times|\mathcal{I}_{\boldsymbol{N}}|}$.

The parallel NFFT (PNFFT) algorithms, implemented in our library [36], are fast approximate algorithms to compute the sums in (2.2) and the adjoint transform (2.3). These both transforms are also the cornerstone for the nonequispaced convolution, see e.g. [30]. In addition, we implemented a fast approximate algorithm for computing the gradients (2.4).

3. The Three-Dimensional NFFT Algorithm.

This section summarizes the mathematical theory and ideas behind the NFFT based on [42, 26, 27]. For further NFFT approaches see [27, Appendix C]. Assume $\boldsymbol{n} = (n_0, n_1, n_2)^\top \in 2\mathbb{N}^3$, with $\boldsymbol{N} \le \boldsymbol{n}$. Again, we use the multi-index set $\mathcal{I}_{\boldsymbol{n}} := \left\{-\frac{n_0}{2},\dots,\frac{n_0}{2}-1\right\} \times \left\{-\frac{n_1}{2},\dots,\frac{n_1}{2}-1\right\} \times \left\{-\frac{n_2}{2},\dots,\frac{n_2}{2}-1\right\}$ and the total number of frequencies $|\mathcal{I}_{\boldsymbol{n}}| = n_0 \cdot n_1 \cdot n_2$. Let $\psi\colon \mathbb{T} \to \mathbb{R}$ be a smooth window function, i.e., a function that is well localized in spatial domain and in frequency domain. We denote its Fourier coefficients by $\hat{\psi}_k$, $k \in \mathbb{Z}$. Furthermore, we define a multivariate window function $\varphi\colon \mathbb{T}^3 \to \mathbb{R}$ by the tensor product $\varphi(\boldsymbol{x}) := \psi(x_0)\,\psi(x_1)\,\psi(x_2)$. A simple consequence is that its Fourier coefficients

$$\hat{\varphi}_{\boldsymbol{k}} := \int_{\mathbb{T}^3} \varphi(\boldsymbol{x}) \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \mathrm{d}\boldsymbol{x}$$

are given by $\hat{\varphi}_{\boldsymbol{k}} = \hat{\psi}_{k_0}\hat{\psi}_{k_1}\hat{\psi}_{k_2}$, $\boldsymbol{k} = (k_0, k_1, k_2)^\top \in \mathbb{Z}^3$ with $\hat{\psi}_k := \int_{\mathbb{T}} \psi(x)\mathrm{e}^{+2\pi\mathrm{i}kx}\mathrm{d}x$ and the gradient $\nabla\varphi(\boldsymbol{x})$ can be easily computed by

$$\nabla\varphi(\boldsymbol{x}) = \left(\psi'(x_0)\psi(x_1)\psi(x_2),\, \psi(x_0)\psi'(x_1)\psi(x_2),\, \psi(x_0)\psi(x_1)\psi'(x_2)\right)^\top .$$

We follow the general approach of [43, 42] and approximate the complex exponentials in the trigonometric polynomial (2.1) by

$$\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}} \approx \frac{1}{|\mathcal{I}_{\boldsymbol{n}}|\hat{\varphi}_{\boldsymbol{k}}} \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x})} \varphi\left(\boldsymbol{x}-\boldsymbol{l}\odot\boldsymbol{n}^{-1}\right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\left(\boldsymbol{l}\odot\boldsymbol{n}^{-1}\right)},$$

where the multi-index set

$$\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}) := \{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n}} \colon \boldsymbol{n}\odot\boldsymbol{x}-m\boldsymbol{1} \le \boldsymbol{l} \le \boldsymbol{n}\odot\boldsymbol{x}+m\boldsymbol{1}\}$$

collects these indexes where the window function φ is mostly concentrated. Here, $m\in\mathbb{N}$ is a small window cutoff parameter, which depends on the particular choice of the window function. We use the vector $\boldsymbol{1} := (1,1,1)^\top$, the component-wise vector product $\boldsymbol{n}\odot\boldsymbol{x} := (n_0x_0, n_1x_1, n_2x_2)^\top$, the reciprocal of a vector $\boldsymbol{n}$ with nonzero components $\boldsymbol{n}^{-1} := \left({n_0}^{-1}, {n_1}^{-1}, n_2^{-1}\right)^\top$ and the inequality between two vectors holds component-wise.

After changing the order of summation in (2.1) we obtain for $\boldsymbol{x}_j\in\mathbb{T}^3$, $j=1,\ldots,M$, the approximation

$$f(\boldsymbol{x}_j) \approx \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)} \left(\sum_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}} \frac{\hat{f}_{\boldsymbol{k}}}{|\mathcal{I}_{\boldsymbol{n}}|\hat{\varphi}_{\boldsymbol{k}}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\left(\boldsymbol{l}\odot\boldsymbol{n}^{-1}\right)}\right) \varphi\left(\boldsymbol{x}_j-\boldsymbol{l}\odot\boldsymbol{n}^{-1}\right),$$

which causes a truncation error and an aliasing error, see [42, 27] for details. As can be readily seen, after an initial deconvolution step,

$$\hat{g}_{\boldsymbol{k}} = \frac{\hat{f}_{\boldsymbol{k}}}{|\mathcal{I}_{\boldsymbol{n}}|\hat{\varphi}_{\boldsymbol{k}}}, \quad \boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}, \tag{3.1}$$

the expression in brackets can be computed via a three-dimensional oversampled FFT of total size $|\mathcal{I}_{\boldsymbol{n}}|$,

$$g_{\boldsymbol{l}} = \sum_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}} \hat{g}_{\boldsymbol{k}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\left(\boldsymbol{l}\odot\boldsymbol{n}^{-1}\right)}, \quad \boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n}}. \tag{3.2}$$

The final step consists of the evaluation of sums having at most $(2m+1)^3$ terms where the window function φ is sampled only in the neighborhood of the node $\boldsymbol{x}_j$, i.e.,

$$f(\boldsymbol{x}_j) \approx s_j := \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}\, \varphi\left(\boldsymbol{x}_j-\boldsymbol{l}\odot\boldsymbol{n}^{-1}\right), \tag{3.3}$$

and

$$\nabla f(\boldsymbol{x}_j) \approx \nabla s_j := \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}\, \nabla\varphi\left(\boldsymbol{x}_j-\boldsymbol{l}\odot\boldsymbol{n}^{-1}\right).$$

In addition to the evaluation of the window function φ, it requires roughly $|\mathcal{I}_{\boldsymbol{N}}| + |\mathcal{I}_{\boldsymbol{n}}|\log|\mathcal{I}_{\boldsymbol{n}}| + (2m+1)^3M$ floating point operations. In matrix-vector notation, the NFFT Algorithm can be written as $\boldsymbol{A}\hat{\boldsymbol{f}} \approx \boldsymbol{B}\boldsymbol{F}\boldsymbol{D}\hat{\boldsymbol{f}}$, where $\boldsymbol{D}$ is a real $|\mathcal{I}_{\boldsymbol{N}}|\times|\mathcal{I}_{\boldsymbol{N}}|$ diagonal matrix defined by

$$\boldsymbol{D} := \operatorname{diag}\left(1/\hat{\varphi}_{\boldsymbol{k}}\right)_{\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}}.$$

The matrix $\boldsymbol{F} := (\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}(\boldsymbol{n}^{-1}\odot\boldsymbol{l})})_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n}},\boldsymbol{k}\in\mathcal{I}_{\boldsymbol{N}}}$ is a truncated Fourier matrix of size $|\mathcal{I}_{\boldsymbol{n}}| \times |\mathcal{I}_{\boldsymbol{N}}|$ and $\boldsymbol{B}$ denotes the sparse real $M \times |\mathcal{I}_{\boldsymbol{n}}|$ matrix

$$\boldsymbol{B} := (b_{j\boldsymbol{l}})_{j=1,\ldots,M;\,\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n}}} , \quad b_{j\boldsymbol{l}} := \begin{cases} \varphi\left(\boldsymbol{x}_j - \boldsymbol{n}^{-1}\odot\boldsymbol{l}\right) & : \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j) \\ 0 & : \boldsymbol{l} \notin \mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j) \end{cases} .$$

An approximation of the adjoint transform is given by $\boldsymbol{A}^{\mathsf{H}}\boldsymbol{f} \approx \boldsymbol{D}\boldsymbol{F}^{\mathsf{H}}\boldsymbol{B}^{\top}\boldsymbol{f}$. The gradients (2.4) can be approximated by means of the analytic derivative of the window function [11], i.e., $\nabla\boldsymbol{A}\hat{\boldsymbol{f}} \approx \nabla\boldsymbol{B}\boldsymbol{F}\boldsymbol{D}\hat{\boldsymbol{f}}$ with

$$\nabla\boldsymbol{B} := (\nabla b_{j\boldsymbol{l}})_{j=1,\ldots,M;\,\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n}}} , \quad \nabla b_{j\boldsymbol{l}} := \begin{cases} \nabla\varphi\left(\boldsymbol{x}_j - \boldsymbol{n}^{-1}\odot\boldsymbol{l}\right) & : \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j) \\ 0 & : \boldsymbol{l} \notin \mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j) \end{cases} .$$

Note that the NFFT and gradient NFFT only differ in the multiplication with the last matrix $\boldsymbol{B}$ and $\nabla\boldsymbol{B}$, respectively. Since the window function φ is defined as a tensor product, the evaluation of function values for both matrices can be easily combined. For a given node $\boldsymbol{x} = (x_0, x_1, x_2)^{\top} \in \mathbb{T}^3$ it is sufficient to evaluate the one-dimensional window function ψ and its derivative ψ' at the three coordinates x_0, x_1, x_2.

To keep the approximation error small, several functions φ with good localization in time and frequency domain have been proposed. In our parallel NFFT implementation the user is free to choose between the (dilated) *Gaussian* [7, 43, 6], (dilated) *cardinal central B–splines* [2, 43], and (dilated) *Kaiser–Bessel functions* [24, 15]. We point out that the approximation error introduced by the NFFT decays exponentially with the number of summands m. Error estimates for the multivariate case were presented in [9], see also [27, Appendix C]. In the case of the Gaussian window function, the evaluations of the exponential function `exp()` can be reduced substantially, see [18] and [27, Appendix C].

4. The Parallel Three-Dimensional NFFT Algorithm.

In this section we describe a parallel algorithm for computing the three-dimensional NFFT on massively parallel, distributed memory architectures. The implementation of this algorithm is based on the Massage Passing Interface [33]. Our parallel NFFT (PNFFT) algorithm combines the serial three-dimensional NFFT algorithm from Section 3 with a three-dimensional block domain decomposition. In addition, we pay special attention to the case where all the nonequispaced nodes $\boldsymbol{x}_j$ are contained in a special subset of the torus $\mathbb{T}^3$. For $\boldsymbol{C} = (C_0, C_1, C_2)^{\top} \in \mathbb{R}^3$ with $0 < C_0, C_1, C_2 \leq 1$ we define the truncated torus $\mathbb{T}^3_{\boldsymbol{C}} := [-\frac{C_0}{2}, \frac{C_0}{2}) \times [-\frac{C_1}{2}, \frac{C_1}{2}) \times [-\frac{C_2}{2}, \frac{C_2}{2})$. For the parallel NFFT we assume $\boldsymbol{x}_j \in \mathbb{T}^3_{\boldsymbol{C}}$ for every $j = 1, \ldots, M$. Obviously, for $C_0 = C_1 = C_2 = 1$ this corresponds to the serial NFFT, where the nodes $\boldsymbol{x}_j$ are contained in the whole three-dimensional torus $\mathbb{T}^3$. This slight generalization is necessary in order to assure a load balanced distribution of nodes $\boldsymbol{x}_j$ whenever the nodes are concentrated in the center of the box.

4.1. Description of the Algorithm.

Assume $\boldsymbol{P} = (P_0, P_1, P_2)^{\top} \in \mathbb{N}^3$. We identify every MPI process of a given parallel hardware architecture with a multi-index of the three-dimensional process mesh $\mathcal{P}_{\boldsymbol{P}} := \{0, \ldots, P_0 - 1\} \times \{0, \ldots, P_1 - 1\} \times \{0, \ldots, P_2 - 1\}$. For every process $\boldsymbol{r} = (r_0, r_1, r_2)^{\top} \in \mathcal{P}_{\boldsymbol{P}}$ we define the multi-index set

$$\mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}} = \left\{ (k_0, k_1, k_2)^{\top} \in \mathcal{I}_{\boldsymbol{N}} : -\frac{N_t}{2} + r_t\frac{N_t}{P_t} \leq k_t < -\frac{N_t}{2} + (r_t + 1)\frac{N_t}{P_t},\ t = 0, 1, 2 \right\} .$$

At the beginning of our parallel algorithm, we assume the NFFT input array of $|\mathcal{I}_{\boldsymbol{N}}|$ complex numbers to be distributed among the three-dimensional process mesh $\mathcal{P}_{\boldsymbol{P}}$ such that every process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$ holds the input data $\hat{f}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}}$ in its locally available memory. For the sake of simplicity, we assume that the input array sizes N_0, N_1, N_2 are divisible by the process mesh sizes P_0, P_1, P_2. Therefore, the input array is distributed in equal blocks of size $N_0/P_0 \cdot N_1/P_1 \cdot N_2/P_2$. This restriction serves to keep the notation simple. Nevertheless, our implementation supports arbitrary process mesh sizes $\boldsymbol{P} \in \mathbb{N}^3$.

The serial NFFT algorithm starts with the deconvolution step (3.1) that consists of an ordinary point wise multiplication. It can be calculated straight forward in parallel, i.e., every process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$ computes

$$\hat{g}_{\boldsymbol{k}} = \frac{\hat{f}_{\boldsymbol{k}}}{|\mathcal{I}_{\boldsymbol{n}}|\hat{\varphi}_{\boldsymbol{k}}}, \quad \boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}} .$$

In the second step (3.2) we compute a three-dimensional oversampled FFT. Before we look at the parallel counter part of this step, we need the following slight generalization. Similar to the truncated input data set of an oversampled FFT, we want to allow a truncated output data set. Therefore, we introduce the pruned FFT output size $\boldsymbol{L} \in 2\mathbb{N}^3$ with $\boldsymbol{L} \leq \boldsymbol{n}$. Hereby, the inequality holds component-wise. The pruned FFT is given by

$$g_{\boldsymbol{l}} = \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N}}} \hat{g}_{\boldsymbol{k}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\left(\boldsymbol{l} \odot \boldsymbol{n}^{-1}\right)}, \quad \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{L}} ,$$

with $\mathcal{I}_{\boldsymbol{L}} := \{-\frac{L_0}{2}, \ldots, \frac{L_0}{2}-1\} \times \{-\frac{L_1}{2}, \ldots, \frac{L_1}{2}-1\} \times \{-\frac{L_2}{2}, \ldots, \frac{L_2}{2}-1\}$. Obviously, for $\boldsymbol{L} = \boldsymbol{n}$ the pruned FFT coincides with the second step of the serial NFFT algorithm given by equation (3.2). The significance of the pruned FFT output size $\boldsymbol{L}$ becomes clear, if we have look at the third step of the NFFT algorithm shown in equation (3.3). There the summation runs over the multi-index sets $\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j) \subset \mathcal{I}_{\boldsymbol{n}}$, $j \in \mathcal{M}$. We want to take advantage of the fact that all the nodes $\boldsymbol{x}_j$ are contained in the truncated torus $\mathbb{T}^3_{\boldsymbol{C}}$. In order to avoid the computation of unneeded coefficients $g_{\boldsymbol{l}}$ we are looking for the smallest multi-index set $\mathcal{I}_{\boldsymbol{L}}$ that holds $\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j) \subset \mathcal{I}_{\boldsymbol{L}}$ for every $j \in \mathcal{M}$. The component-wise smallest $\boldsymbol{L} = (L_0, L_1, L_2)^\top \in 2\mathbb{N}^3$ that fulfills these conditions is given by

$$L_t := \min\left\{n_t, 2\left(\left\lceil C_t \frac{n_t}{2} \right\rceil + m\right)\right\}, \quad t = 0, 1, 2 .$$

In parallel we substitute the three-dimensional pruned FFT by a parallel one, i.e.,

$$g_{\boldsymbol{l}} = \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{s}}_{\boldsymbol{N},\boldsymbol{P}}} \hat{g}_{\boldsymbol{k}} \, \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}(\boldsymbol{n}^{-1} \odot \boldsymbol{l})}, \quad \boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{L},\boldsymbol{P}} .$$

The formal order of summation in this notation was chosen to symbolize the parallel data decomposition of a block distributed parallel three-dimensional FFT algorithm. In general, a parallel FFT algorithm may use a different order of summation or an approximate algorithm to calculate the Fourier transform. The inner sum reflects that every process $\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}$ starts with calculations on its locally available input data block of size $N_0/P_0 \cdot N_1/P_1 \cdot N_2/P_2$. The outer sums stands for the global communication that must be performed somehow within the parallel FFT algorithm. After the parallel

FFT the output data $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{L}}$, is distributed on the process mesh in a similar way as the input data set, i.e., every process owns a block of $L_0/P_0 \cdot L_1/P_1 \cdot L_2/P_2$ complex numbers. We assume that L_0, L_1, L_2 are divisible by the process mesh sizes P_0, P_1, P_2 in order to keep the notation simple. Again, for every process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$ the multi-index set

$$\mathcal{I}_{\boldsymbol{L},\boldsymbol{P}}^{\boldsymbol{r}} := \left\{ (k_0, k_1, k_2)^\top \in \mathcal{I}_{\boldsymbol{n}} : -\frac{L_t}{2} + r_t \frac{L_t}{P_t} \le k_t < -\frac{L_t}{2} + (r_t + 1) \frac{L_t}{P_t},\ t = 0, 1, 2 \right\}$$

collects all the multi-indexes of locally available data. We apply the PFFT software library [35] for computing the parallel pruned FFT. This library was developed for calculating parallel FFT on massively parallel architectures. It uses a transpose FFT algorithm that consist of successive one-dimensional FFT and global data transpositions, see [37] for details. We stress that PFFT is the only publicly available parallel FFT software library that pays special attention to the efficient parallel computation of pruned FFT. This feature is crucial in order to assure a good load balancing of our parallel NFFT algorithm. It is noteworthy to say that PFFT is based on a two-dimensional domain decomposition, i.e., the three-dimensional decomposed FFT input $\hat{g}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{r}}$, and output $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{L},\boldsymbol{P}}^{\boldsymbol{r}}$ is redistributed before and after every parallel FFT. Therefore, an upper limit for the number of processes is given by the two-dimensional decomposition of the parallel FFT.

The block data distribution of the FFT output $g_{\boldsymbol{l}}$, $\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{L}}$, naturally induces a block decomposition of the truncated Torus $\mathbb{T}_{\boldsymbol{C}}^3$. This motivates the definition of the index sets

$$\mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}} := \left\{ j = 1, \ldots, M : \exists \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{L},\boldsymbol{P}}^{\boldsymbol{r}} \text{ with } \boldsymbol{l} \le \boldsymbol{n} \odot \boldsymbol{x}_j < \boldsymbol{l} + \boldsymbol{1} \right\},$$

for every process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$. We assign all nodes $\boldsymbol{x}_j$, $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$, to a single process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$. As one can already see, heterogeneous distributions of the nodes $\boldsymbol{x}_j$, $j = 1, \ldots, M$, can lead to imbalances in memory consumption and workload between the processes, which is a typical problem of mesh based domain decompositions.

According to the discrete convolution step of the serial NFFT algorithm, we compute the sums (3.3), which run over the local multi-index sets $\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)$, $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$. Our choice of parameter $\boldsymbol{L}$ assures $\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j) \subset \mathcal{I}_{\boldsymbol{L}}$ for every $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$, i.e., the output of the pruned FFT is sufficient. But in general, not all sufficient data g_l, $l \in \mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)$, is located on a single process $\boldsymbol{r}$. Therefore, we perform a communication step in order to gather all the additionally needed data. However, this step equals to a mesh ghost cell communication [20, Ch. 5.6.1] and only involves nearest neighbor communication. With the definition of the multi-index sets

$$\begin{aligned}\mathcal{I}_{\boldsymbol{L},\boldsymbol{P},m}^{\boldsymbol{r}} := \Big\{ &(l_0, l_1, l_2) \in \mathcal{I}_{\boldsymbol{n}} : \\ &-\frac{L_t}{2} + r_t \frac{L_t}{P_t} - m \le l_t < -\frac{L_t}{2} + (r_t + 1) \frac{L_t}{P_t} + m,\ t = 0, 1, 2 \Big\},\end{aligned}$$

for all processes $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$, we symbolize the ghost cell communication by

$$g_{\boldsymbol{l}}^{\boldsymbol{r}} = g_{\boldsymbol{l}}, \quad \boldsymbol{l} \in \mathcal{I}_{\boldsymbol{L},\boldsymbol{P},m}^{\boldsymbol{r}} \setminus \mathcal{I}_{\boldsymbol{L},\boldsymbol{P}}^{\boldsymbol{r}}. \tag{4.1}$$

We use the ghost cell support of the PFFT software library for implementing the ghost cell communication. Finally, the sums

$$s_j = \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)} g_{\boldsymbol{l}}^{\boldsymbol{r}} \, \varphi(\boldsymbol{x}_j - \boldsymbol{n}^{-1} \odot \boldsymbol{l}), \quad j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}},$$

are calculated locally on all processes $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$. Alg. 1 summarizes the PNFFT algorithm in pseudo-code.

```
Input: $\boldsymbol{x}_j \in \mathbb{T}^3_{\boldsymbol{C}}$, $j \in \mathcal{M}^{\boldsymbol{r}}_{\boldsymbol{P}}$, and $\hat{f}_{\boldsymbol{k}} \in \mathbb{C}$, $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}}$.
1: For $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}}$ compute $\hat{g}_{\boldsymbol{k}} := |\mathcal{I}_{\boldsymbol{n}}|^{-1} \cdot \hat{f}_{\boldsymbol{k}}/\hat{\varphi}_{\boldsymbol{k}}$.
2: For $\boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{L},\boldsymbol{P}}$ compute $g_{\boldsymbol{l}} := \sum_{\boldsymbol{s}\in\mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k}\in\mathcal{I}^{\boldsymbol{s}}_{\boldsymbol{N},\boldsymbol{P}}} \hat{g}_{\boldsymbol{k}} \,\mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}(\boldsymbol{n}^{-1}\odot\boldsymbol{l})}$ by a parallel
   three-dimensional pruned FFT.
3: For $\boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{L},\boldsymbol{P},m} \setminus \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{L},\boldsymbol{P}}$ copy $g^{\boldsymbol{r}}_{\boldsymbol{l}} := g_{\boldsymbol{l}}$ by a ghost cell communication.
4: For $j \in \mathcal{M}^{\boldsymbol{r}}_{\boldsymbol{P}}$ compute $s_j := \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)} g^{\boldsymbol{r}}_{\boldsymbol{l}} \,\varphi(\boldsymbol{x}_j - \boldsymbol{n}^{-1}\odot\boldsymbol{l})$.
5: For $j \in \mathcal{M}^{\boldsymbol{r}}_{\boldsymbol{P}}$ compute $\nabla s_j := \sum_{\boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)} g^{\boldsymbol{r}}_{\boldsymbol{l}} \,\nabla\varphi(\boldsymbol{x}_j - \boldsymbol{n}^{-1}\odot\boldsymbol{l})$.
Output: Approximate function values $s_j \approx f_j$ and gradients $\nabla s_j \approx \nabla f_j$, $j \in \mathcal{M}^{\boldsymbol{r}}_{\boldsymbol{P}}$.
```

Alg. 1: Parallel, three-dimensional, nonequispaced fast Fourier transform (PNFFT) for each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$.

The adjoint PNFFT algorithm can be derived analogously from the serial adjoint NFFT algorithm [27]. Note that the adjoint counterpart of the ghost cell communication (4.1) is a sum over all ghost cells, i.e.,

$$g_{\boldsymbol{l}} = \sum_{\boldsymbol{s}\in\mathcal{P}_{\boldsymbol{P}}} g^{\boldsymbol{s}}_{\boldsymbol{l}}, \quad \boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{L},\boldsymbol{P}}\,.$$

The pseudo-code of the adjoint PNFFT is given by Alg. 2.

```
Input: $\boldsymbol{x}_j \in \mathbb{T}^3_{\boldsymbol{C}}$, and $f_j \in \mathbb{C}$, $j \in \mathcal{M}^{\boldsymbol{r}}_{\boldsymbol{P}}$.
1: For $\boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{L},\boldsymbol{P},m}$ compute $g^{\boldsymbol{r}}_{\boldsymbol{l}} := \sum_{\substack{\{j\in\mathcal{M}^{\boldsymbol{r}}_{\boldsymbol{P}}:\\ \boldsymbol{l}\in\mathcal{I}_{\boldsymbol{n},m}(\boldsymbol{x}_j)\}}} f_j \,\varphi(\boldsymbol{x}_j - \boldsymbol{n}^{-1}\odot\boldsymbol{l})$.
2: For $\boldsymbol{l} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{L},\boldsymbol{P}}$ accumulate $g_{\boldsymbol{l}} := \sum_{\boldsymbol{s}\in\mathcal{P}_{\boldsymbol{P}}} g^{\boldsymbol{s}}_{\boldsymbol{l}}$ by an adjoint ghost cell communication.
3: For $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}}$ compute $\hat{g}_{\boldsymbol{k}} := \sum_{\boldsymbol{s}\in\mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{l}\in\mathcal{I}^{\boldsymbol{s}}_{\boldsymbol{L},\boldsymbol{P}}} g_{\boldsymbol{l}} \,\mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}(\boldsymbol{n}^{-1}\odot\boldsymbol{l})}$ by an adjoint parallel
   three-dimensional pruned FFT.
4: For $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}}$ compute $\hat{s}_{\boldsymbol{k}} := |\mathcal{I}_{\boldsymbol{n}}|^{-1} \cdot \hat{g}_{\boldsymbol{k}}/\hat{\varphi}_{\boldsymbol{k}}$.
Output: Approximate coefficients $\hat{s}_{\boldsymbol{k}} \approx \hat{h}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}^{\boldsymbol{r}}_{\boldsymbol{N},\boldsymbol{P}}$.
```

Alg. 2: Adjoint, parallel, three-dimensional, nonequispaced fast Fourier transform (adjoint PNFFT) for each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$.

5. Application of the Parallel NFFT. In this section we demonstrate the application of our parallel NFFT algorithm in order to calculate the Coulomb potentials and forces of a charged particle system on massively parallel distributed memory architectures. We start with the outline of the serial fast summation algorithm [40, 41] and continue with its parallel counterpart.

5.1. Serial Fast Summation Algorithm. Assume M charged particles with charge $q_j \in \mathbb{R}$ at position $\boldsymbol{x}_j \in \mathbb{T}^3$, $j = 1\ldots, M$. We are interested in the fast evaluation of the potentials

$$\phi_j := \phi(\boldsymbol{x}_j) = \sum_{\substack{l=1\\l\neq j}}^{M} q_l \frac{1}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2}, \quad j = 1, \ldots, M, \tag{5.1}$$

and fields

$$\boldsymbol{E}_j := -\nabla\phi(\boldsymbol{x}_j) = -\sum_{\substack{l=1\\l\neq j}}^{M} q_l \frac{\boldsymbol{x}_j - \boldsymbol{x}_l}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2^3}, \quad j = 1, \ldots, M\,. \tag{5.2}$$

Hereby, $\|\boldsymbol{x}\|_2 := (x_0^2 + x_1^2 + x_2^2)^{1/2}$ denotes the Euclidean norm. Without loss of generality we may assume that the nodes are scaled, such that $\|\boldsymbol{x}_j\|_2 < \frac{1}{4} - \frac{\varepsilon_\mathrm{B}}{2}$ and consequently $\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2 < \frac{1}{2} - \varepsilon_\mathrm{B}$.

We outline the NFFT based fast summation algorithm. It requires $\mathcal{O}(M \log \sqrt[3]{M})$ arithmetic operations for uniformly distributed source nodes $\boldsymbol{x}_j$. This approach was suggested in [40, 41]. There, a regularization

$$R(r) := \begin{cases} T_\mathrm{I}(r) & \text{if} \quad r \le \varepsilon_\mathrm{I}, \\ T_\mathrm{B}(r) & \text{if} \quad \frac{1}{2} - \varepsilon_\mathrm{B} < r < \frac{1}{2}, \\ T_\mathrm{B}(\frac{1}{2}) & \text{if} \quad \frac{1}{2} \le r, \\ \frac{1}{r} & \text{otherwise}, \end{cases}$$

has been introduced. The functions T_I and T_B are chosen such that $R(\|\boldsymbol{x}\|_2)$ is in the Sobolev space $H^p(\mathbb{T}^3)$ for an appropriate degree of smoothness $p \in \mathbb{N}$. Several regularizations of $\frac{1}{r}$ are possible, e.g., by algebraic polynomials, splines, trigonometric polynomials or two point Taylor interpolation, see [13]. The potentials ϕ_j in equation (5.1) can be approximated by

$$\phi_j \approx h_j := \phi_j^\mathrm{NE} + \phi_j^\mathrm{RF},$$

where

$$\phi_j^\mathrm{NE} := R(0) + \sum_{l \in \mathcal{I}_{\varepsilon_\mathrm{I}}^\mathrm{NE}(j)} q_l \left(\frac{1}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2} - R(\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2) \right), \tag{5.3}$$

$$\phi_j^\mathrm{RF} := \sum_{\boldsymbol{k} \in I_{\boldsymbol{N}}} \hat{R}_{\boldsymbol{k}} \left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_l} \right) \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\,. \tag{5.4}$$

Hereby, the index set $\mathcal{I}_{\varepsilon_\mathrm{I}}^\mathrm{NE}(j) := \{l \in \{1, \ldots, M\} \setminus \{j\} \colon \|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2 < \varepsilon_\mathrm{I}\}$ collects all the indexes of $\boldsymbol{x}_j$ next neighbors and the Fourier coefficients of the regularized kernel function

$$\hat{R}_{\boldsymbol{k}} := \frac{1}{|\mathcal{I}_{\boldsymbol{N}}|} \sum_{\boldsymbol{l} \in I_{\boldsymbol{N}}} R(\|\boldsymbol{N}^{-1} \odot \boldsymbol{l}\|_2) \mathrm{e}^{+2\pi\mathrm{i}(\boldsymbol{N}^{-1} \odot \boldsymbol{l})\boldsymbol{k}}, \quad \boldsymbol{k} \in I_{\boldsymbol{N}}\,,$$

are precomputed by a three-dimensional discrete Fourier transform. Since the gradient of the regularization is given by $\nabla R(\|\boldsymbol{x}\|_2) = R'(\|\boldsymbol{x}\|_2)\frac{\boldsymbol{x}}{\|\boldsymbol{x}\|_2}$, we are able to approximate the fields $\boldsymbol{E}_j$ in equation (5.2) analogously to the potentials by

$$\boldsymbol{E}_j \approx \nabla h_j := \boldsymbol{E}_j^{\mathrm{NE}} + \boldsymbol{E}_j^{\mathrm{RF}},$$

where

$$\boldsymbol{E}_j^{\mathrm{NE}} := -\sum_{l\in\mathcal{I}_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)} q_l \frac{\boldsymbol{x}_j - \boldsymbol{x}_l}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2}\left(\frac{1}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2^2} - R'(\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2)\right), \tag{5.5}$$

$$\boldsymbol{E}_j^{\mathrm{RF}} := -\sum_{\boldsymbol{k}\in I_{\boldsymbol{N}}} \hat{R}_{\boldsymbol{k}}\left(\sum_{l=1}^{M} q_l \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_l}\right)\nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}. \tag{5.6}$$

If the nodes $\boldsymbol{x}_j$ are "sufficiently uniformly distributed" this can indeed be done in a fast way, namely:

Near field computation (5.3)**,** (5.5)**.** To achieve the desired complexity of our algorithm we suppose that there exists a small constant $\nu \in \mathbb{N}$ such that the near field index sets $\mathcal{I}_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)$ contain at most ν indexes for every node $\boldsymbol{x}_j$, $j = 1, \ldots, M$. This implies that ε_{I} depends linearly on $1/\sqrt[3]{M}$. Then for fixed $\boldsymbol{x}_j$ the sum (5.3) contains not more than ν terms so that its evaluation at all M nodes $\boldsymbol{x}_j$ requires only $\mathcal{O}(\nu M)$ arithmetic operations.

Far field computation (5.4)**,** (5.6) **by NFFT.** The expression in the inner brackets of (5.4) can be computed by an adjoint parallel three-dimensional NFFT of total size $|\mathcal{I}_N|$. This is followed by $|\mathcal{I}_N|$ multiplications with the Fourier coefficients $\hat{R}_{\boldsymbol{k}}$ of the regularized kernel function and completed by a parallel three-dimensional NFFT of total size $|\mathcal{I}_N|$ to compute the outer most sum. If m is the cut-off parameter and $\boldsymbol{n}$ the FFT size of the (adjoint) NFFT, then the proposed evaluation of ϕ_j^{RF} at the nodes $\boldsymbol{x}_j$, $j = 1, \ldots, M$ requires $\mathcal{O}(m^3 M + |\mathcal{I}_n| \log |\mathcal{I}_n|))$ arithmetic operations. The relation between $M, \boldsymbol{N}$ and $\boldsymbol{n}$ is determined by the approximation error of the algorithm and is discussed in detail in [40, 41, 13].

5.2. Parallel Fast Summation Algorithm. After we have seen the highly modularized structure of the serial fast summation algorithm it is easy to derive a parallel fast summation algorithm by substituting every module with its parallel counterpart. Our parallel algorithm starts with a parallel forward sorting step that assures the following two conditions. First we need to distribute the nodes $\boldsymbol{x}_j$ according to our block decomposition such that every process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$ holds the nodes $\boldsymbol{x}_j$, $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$. Furthermore, every process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$ needs local copies of all the nodes that are involved in the calculation of the near field sums (5.1) and (5.5), i.e., all the nodes $\boldsymbol{x}_j$, $j \in \bigcup_{l\in\mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}} \mathcal{I}_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(l)$. We perform these two tasks at once using a fine-grained data distribution operation [23] that is implemented within a software library for parallel sorting algorithms [3].

Note that the serial fast summation algorithm requires the condition $\|\boldsymbol{x}_j\|_2 < \frac{1}{4} - \varepsilon_B$ for every node $\boldsymbol{x}_j$, $j = 1, \ldots, M$. Therefore, we set $\boldsymbol{C} = (\frac{1}{4} - \varepsilon_B, \frac{1}{4} - \varepsilon_B, \frac{1}{4} - \varepsilon_B)^\top$ and our parallel adjoint NFFT starts with a three-dimensional decomposition of the truncated torus $\mathbb{T}_{\boldsymbol{C}}^3$. We stress that the usage of the truncated torus $\mathbb{T}_{\boldsymbol{C}}^3$ is crucial in order to avoid severe load balancing problems in this case. If we use a block decomposition of the whole torus $\mathbb{T}^3$ instead, at most an eighth of the processes will receive a non empty block.

After the parallel forward sort every process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$ owns all the local particles $\boldsymbol{x}_j$, $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$, and the associated particles in the near field radii of these particles, i.e., $\boldsymbol{x}_j$, $j \in \bigcup_{l \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}} \mathcal{I}_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(l)$. Now, the local near field computations (5.3) and (5.5) are performed with a standard linked cell algorithm, see e.g. [22, Ch. 8.4] or [19, Ch. 3], and the far field computations are split into the following three steps.

At first we compute

$$\hat{a}_{\boldsymbol{k}} := \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{s}}} q_j \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\,, \quad \boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{r}}$$

by an adjoint parallel three-dimensional NFFT (Alg. 2). Again the formal order of summation was chosen to reflect the parallel data decomposition. The convolution in Fourier space is a simple point wise multiplication an is performed locally on each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$, i.e.,

$$\hat{d}_{\boldsymbol{k}} := \hat{a}_{\boldsymbol{k}} \hat{R}_{\boldsymbol{k}}\,, \quad \boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{r}}\,.$$

Hereby, the Fourier coefficients $\hat{R}_{\boldsymbol{k}}$ of the regularization R are precomputed by a parallel three-dimensional FFT

$$\hat{R}_{\boldsymbol{k}} = \frac{1}{|\mathcal{I}_{\boldsymbol{N}}|} \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{l} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{s}}} R(\|\boldsymbol{N}^{-1} \odot \boldsymbol{l}\|_2) \mathrm{e}^{+2\pi\mathrm{i}(\boldsymbol{N}^{-1}\odot\boldsymbol{l})\boldsymbol{k}}, \quad \boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{r}}\,. \tag{5.7}$$

The far field potentials and fields are computed by a parallel NFFT (Alg. 1)

$$\phi_j^{\mathrm{RF}} := \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{s}}} \hat{d}_{\boldsymbol{k}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\,, \quad j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$$

$$\boldsymbol{E}_j^{\mathrm{RF}} := \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{s}}} \hat{d}_{\boldsymbol{k}} \nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\,, \quad j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}\,.$$

Finally, we use the fine-grained data distribution operation from the parallel sorting library to restore the initial parallel distribution of the nodes $\boldsymbol{x}_j$ together with the computed Coulomb potentials and fields.

In summary we obtain Alg. 3 for the fast evaluation of the potentials (5.1) and the fields (5.2). The steps of this algorithm are very similar to P3M (Particle-Particle–Particle-Mesh) algorithms [22, Ch. 8], see also [4] for an overview of particle-mesh algorithms. Analogously, Alg. 3 is called P2NFFT (Particle-Particle–NFFT), since the short range particle-particle interactions are computed in the same way, while the long range particle-mesh part is computed by nonequispaced fast Fourier transforms. Note that this algorithm can be easily modified for other kernels frequently used in the approximation by radial basis functions, e.g., the Gaussian [30] or the (inverse) multiquadric[13] $(x^2 + c^2)^{\pm 1/2}$.

6. Numerical Results. We implemented Alg. 1 (PNFFT), Alg. 2 (adjoint PNFFT) and Alg. 3 (P2NFFT) and investigated the strong scaling behavior of our implementations on a BlueGene/P architecture. The parallel NFFT algorithms have been published in the PNFFT software library [36]. In this section we first present the parallel runtime measurements of the parallel NFFT and its adjoint. Secondly, we show the strong scaling of the P2NFFT. The software has been build with the IBM XL C/C++ compiler (Advanced Edition for Blue Gene/P, V9.0) and the compiler flags `CFLAGS="-O3 -qmaxmem=-1 -qarch=450 -qtune=450"`. As a test system

```
Input:  $\boldsymbol{N} \in 2\mathbb{N}^3$ multi degree,
        $\varepsilon_{\mathrm{I}} > 0$ nearfield size,
        $\varepsilon_{\mathrm{B}} > 0$ boundary size,
        nodes $\boldsymbol{x}_j \in \left\{\boldsymbol{x} \in \mathbb{T}^3 : \|\boldsymbol{x}\|_2 \leq \frac{1}{4} - \frac{\varepsilon_{\mathrm{B}}}{2}\right\}$, $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$,
        sources $q_j \in \mathbb{R},\ j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$.
Precomputation:  Compute the Fourier coefficients $\hat{R}_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{r}}$, given by Equation (5.7) by a parallel three-dimensional FFT.
1: Assign the local nodes $\boldsymbol{x}_j$, $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}} \cup \bigcup_{l \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}} \mathcal{I}_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(l)$, by a parallel forward sort.
2: For $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{r}}$ compute $\hat{a}_{\boldsymbol{k}} := \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{s}}} q_j \mathrm{e}^{+2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}$ by an adjoint parallel three-dimensional NFFT, see Alg. 2.
3: For $\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{r}}$ compute the products $\hat{d}_{\boldsymbol{k}} := \hat{a}_{\boldsymbol{k}} \hat{R}_{\boldsymbol{k}}$.
4: For $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$ compute the far field sums
       $$\phi_j^{\mathrm{RF}} := \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{s}}} \hat{d}_{\boldsymbol{k}} \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}\,,$$
       $$\boldsymbol{E}_j^{\mathrm{RF}} := \sum_{\boldsymbol{s} \in \mathcal{P}_{\boldsymbol{P}}} \sum_{\boldsymbol{k} \in \mathcal{I}_{\boldsymbol{N},\boldsymbol{P}}^{\boldsymbol{s}}} \hat{d}_{\boldsymbol{k}} \nabla \mathrm{e}^{-2\pi\mathrm{i}\boldsymbol{k}\boldsymbol{x}_j}$$
   by a parallel three-dimensional NFFT, see Alg. 1.
5: For $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$ compute the near field sums
       $$\phi_j^{\mathrm{NE}} = R(0) + \sum_{l \in \mathcal{I}_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)} q_l \left( \frac{1}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2} - R(\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2) \right)\,,$$
       $$\boldsymbol{E}_j^{\mathrm{NE}} = \sum_{l \in \mathcal{I}_{\varepsilon_{\mathrm{I}}}^{\mathrm{NE}}(j)} q_l \left( \frac{\boldsymbol{x}_j - \boldsymbol{x}_l}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2^3} - R'(\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2) \frac{\boldsymbol{x}_j - \boldsymbol{x}_l}{\|\boldsymbol{x}_j - \boldsymbol{x}_l\|_2} \right)\,.$$
   by a linked cell algorithm, see [22, Ch. 8.4] or [19, Ch. 3].
6: For $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$ compute the near field corrections $h_j = \phi_j^{\mathrm{NE}} + \phi_j^{\mathrm{RF}}$ and $\nabla h_j = \boldsymbol{E}_j^{\mathrm{NE}} + \boldsymbol{E}_j^{\mathrm{RF}}$.
7: Restore the initial data distribution by a parallel backward sort.
Output: Approximate potentials $h_j \approx \phi_j$ and fields $\nabla h_j \approx \boldsymbol{E}_j$, $j \in \mathcal{M}_{\boldsymbol{P}}^{\boldsymbol{r}}$.
```

Alg. 3: Parallel three-dimensional Particle-Particle NFFT (P2NFFT) for each process $\boldsymbol{r} \in \mathcal{P}_{\boldsymbol{P}}$.

we use a cubic box filled with 12960 particles of a silica melt. It was generated by a simulation of a melting silica crystal using the potential given in [1]. This particle system consist of positive and negative charged ions which are sufficiently homogeneously distributed. We duplicate the initial test system of 12960 particles for 4, 9 and 20 times in every direction of space in order to generate a cubic box filled with

829440, 9447840 and 103680000 particles, respectively. The sum of all charges is equal to zero since the initial sample of 12960 particles is neutrally charged.

6.1. Some Notes on Performance Optimization. Before we present our numerical results, we introduce some performance optimizations that we employed in order to improve the run times of Alg. 1, Alg. 2 and Alg. 3.

Precomputation of the window function. In order to reduce the computational cost of the evaluation of the window function in Alg. 1 and Alg. 2, we use tensor structure based precomputation and interpolation from lookup tables [27]. The flags `PRE_LIN_PSI`, `PRE_QUAD_PSI` and `PRE_KUB_PSI` enable linear, quadratic and cubic interpolation of the one-dimensional window function $\psi(x)$. Additionally, the flag `PRE_PHI_HAT` enables the precomputation of the Fourier coefficients $\varphi_{\boldsymbol{k}}$, $\boldsymbol{k} \in \mathcal{I}^{r}_{\boldsymbol{N},\boldsymbol{P}}$. Hereby, we do not store the full set of $|\mathcal{I}_{\boldsymbol{N}}|$ Fourier coefficients. Instead, we exploit the tensor structure of the three-dimensional window function $\hat{\varphi}_{\boldsymbol{k}} = \hat{\psi}_{k_0}\hat{\psi}_{k_1}\hat{\psi}_{k_2}$, $\boldsymbol{k} = (k_0, k_1, k_2)^\top \in \mathcal{I}_{\boldsymbol{N}}$ and store the precomputed $N_0 + N_1 + N_2$ Fourier coefficients of one-dimensional window functions $\hat{\psi}_{k_t}$, $k_t = -N_t/2, \ldots, N_t/2 - 1$, $t = 0, 1, 2$. Therefore, the evaluation of the three-dimensional Fourier coefficients requires $2 * |\mathcal{I}_{\boldsymbol{N}}|$ multiplications. For our numerical experiments we chose the Kaiser-Bessel window function with cubic interpolation from a precomputed lookup table.

Transposed FFT Output. The PFFT software library employs a parallel FFT algorithm that contains several data transpositions in order to enable the computation of one-dimensional local FFTs [37]. On default these transpositions are reverted after the computation of the FFT, which doubles the amount of global communication. However, a convolution in Fourier space corresponds to a simple point-wise multiplication of two arrays. As long as both array are distributed in the same way, a point-wise multiplication can be easily implemented for any parallel data decomposition. Therefore, we omit the additional communication and use the transpositions of the second FFT to restore the initial data decomposition in the following way. First, we compute the parallel adjoint FFT Step 3 of the adjoint PNFFT Alg. 2 with transposed output. Afterward, the convolutions in Fourier space of the adjoint PNFFT (Step 4 of Alg. 2), the P2NFFT (Step 3 of Alg. 3) and the PNFFT (Step 1 of Alg. 1) are computed on transposed arrays. Finally, we compute the parallel FFT Step 2 of Alg. 1 with transposed input.

Interpolation of the Regularization. The computation of the near field step 5 of the P2NFFT Alg. 3 requires the repetitive evaluation of the regularization $R(r)$ and its derivative $R'(r)$ for $0 \leq r \leq \varepsilon_I$. Since these two functions are smooth, we use cubic interpolation from precomputed lookup tables to speed up their evaluation.

6.2. Runtimes of PNFFT and adjoint PNFFT. In Figure 6.1 we show the wall clock time measurements of Alg. 1 (PNFFT) and Alg. 2 (adj. PNFFT) for 512^3 Fourier coefficients and 829440 nonequispaced nodes up to 16384 cores of a BlueGene/P architecture. For comparison purposes we show the perfect strong scaling times (perfect) of the first recorded time. In addition, we add the wall clock time of the most time consuming parts of these algorithms. These are the convolution Steps 4 and 5 (B, ∇B), the ghost cell communication Step 3 (ghost), the FFT Step 2 (F) and the deconvolution Step 1 (D) of Alg. 1 and their adjoint counterparts of Alg. 2, i.e., the adjoint convolution Step 1 (adj. B), the adjoint ghost cell communication Step 2 (adj, ghost), the adjoint FFT Step 3 (adj. F) and the adjoint deconvolution

Step 4 (adj. D). Both plots are scaled equally such that a direct comparison of the time measurement between the two algorithms is possible.

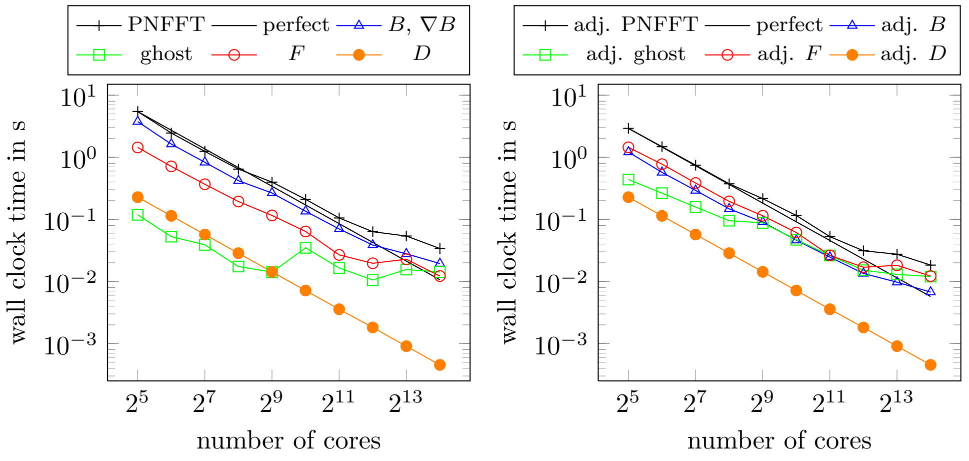

FIGURE 6.1. *Wall clock time measurements of Alg. 1 (PNFFT) on the left and Alg. 2 (adjoint PNFFT) on the right with number of nonequispaced nodes* $M = 829440$*, pruned input FFT size* $\boldsymbol{N} = (512, 512, 512)^\top$*, oversampled FFT size* $\boldsymbol{n} = (576, 576, 576)^\top$*, pruned output FFT size* $\boldsymbol{L} = (174, 174, 174)^\top$ *and window cutoff parameter* $m = 4$*.*

The deconvolution Step 1 of Alg. 1 and the adjoint counterpart Step 4 of Alg. 2 scale perfectly and only represent a small amount of the overall run times. This step is noted by matrix D. The fast Fourier transform (Step 2 of Alg. 1, matrix F) and its adjoint (Step 3 of Alg. 2) show good strong scaling up to 4096 cores, which corresponds to one rack of the BlueGene/P. We observe a performance penalty for computing the parallel FFT on more than one rack. However, please note that the pruned FFT output is only of size 174^3. The FFT-internal two-dimensional distribution of 174^3 complex numbers on 128^2 processes results in very small workload per process.

The discrete convolution step of Alg. 1 includes the calculation of the potentials (Step 4 of Alg. 1) and the calculation of the gradients (Step 5 of Alg. 1). Therefore, it is more time consuming than the corresponding adjoint Step 1 of Alg. 2. Both show good strong scaling behavior. Note that every process gets only 51 nodes $\boldsymbol{x}_j$ in average, if we use 16384 processes in total.

Instead, the ghost cell communication shows bad strong scaling for more than 512 processes. The adjoint ghost cell communication is more expensive than plain ghost cell communication since it involves the communication of all the summands to one process and the synchronization of the partial sums. For 16384 processes the adjoint ghost cell communication turns out to be the most time consuming part of the adjoint parallel NFFT. We observe that the ghost cell communication is the most limiting factor for strong scaling. This slightly improves for larger test cases, where the ration between the ghost cell communication and the overall computing time decreases.

In Figure 6.2 we show the wall clock time measurements of Alg. 1 (PNFFT) and Alg. 2 (adj. PNFFT) for 1024^3 Fourier coefficients and 9447840 nonequispaced nodes up to 65536 cores of a BlueGene/P architecture.

In Figure 6.3 we show the wall clock time measurements of Alg. 1 (PNFFT) and Alg. 2 (adj. PNFFT) for 2048^3 Fourier coefficients and 103680000 nonequispaced nodes up to 65536 cores of a BlueGene/P architecture. There the computing time of

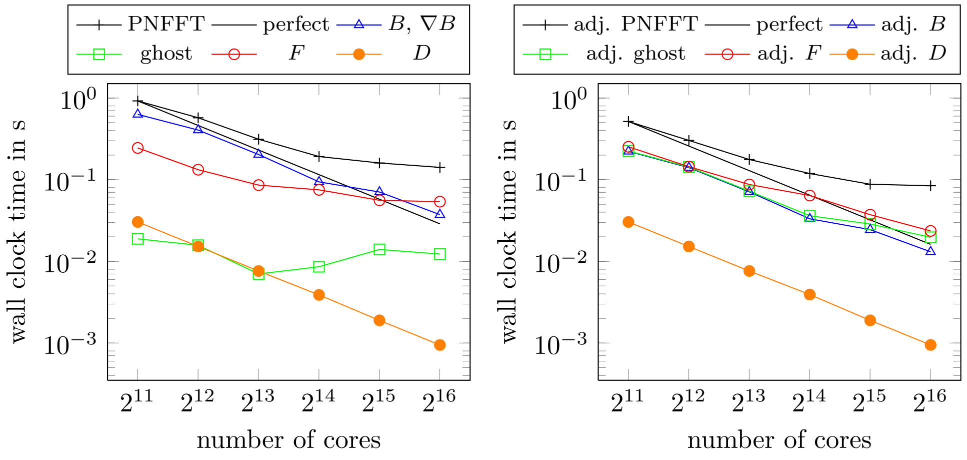

FIGURE 6.2. *Wall clock time measurements of Alg. 1 (PNFFT) on the left and Alg. 2 (adjoint PNFFT) on the right with number of nonequispaced nodes $M = 9447840$, pruned input FFT size $\boldsymbol{N} = (1024, 1024, 1024)^\top$, oversampled FFT size $\boldsymbol{n} = (1152, 1152, 1152)^\top$, pruned output FFT size $\boldsymbol{L} = (340, 340, 340)^\top$ and window cutoff parameter $m = 4$.*

the discrete convolution step, the adjoint FFT and the adjoint ghost cell communication are nearly the same. Also the strong scaling behavior of all three steps is good. Although the ghost cell communication does not provide good scaling behavior, it only takes 4% of the overall PNFFT run time with 65536 processes. Once more, the deconvolution steps show perfect strong scaling. The overall wall clock time of the parallel NFFT and the adjoint parallel NFFT reflect the good scaling of all their most time consuming steps.

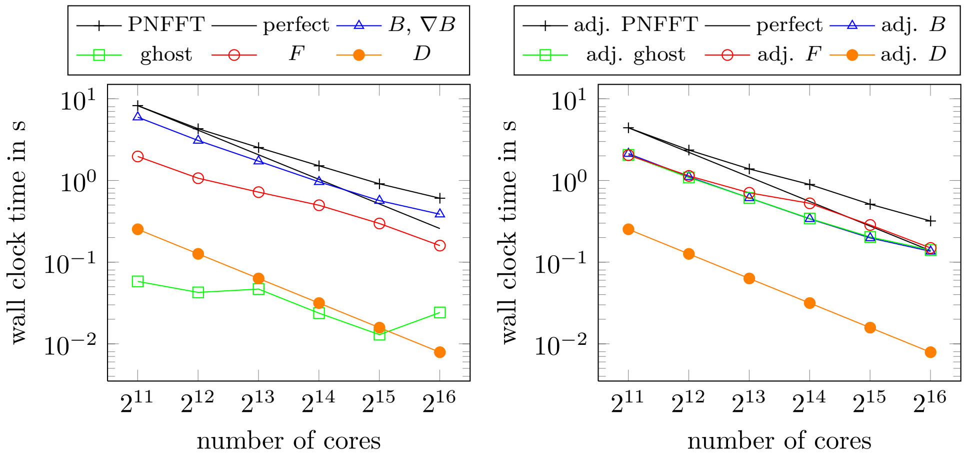

FIGURE 6.3. *Wall clock time measurements of Alg. 1 (PNFFT) on the left and Alg. 2 (adjoint PNFFT) on the right with number of nonequispaced nodes $M = 103680000$, pruned input FFT size $\boldsymbol{N} = (2048, 2048, 2048)^\top$, oversampled FFT size $\boldsymbol{n} = (2304, 2304, 2304)^\top$, pruned output FFT size $\boldsymbol{L} = (674, 674, 674)^\top$ and window cutoff parameter $m = 4$.*

6.3. Runtimes of P2NFFT. Let $\phi_{\text{P2NFFT}}(\boldsymbol{x}_j)$, $j = 1, \ldots, M$ denote the potentials that are calculated by Alg. 3 and $\phi_{\text{REF}}(\boldsymbol{x}_j)$, $j = 1, \ldots, M$ the potentials that are calculated by any high accuracy reference method. For small numbers of particles we chose the direct summation as reference method, while a fast method with higher accuracy was used to calculate the reference potentials for large systems. The absolute RMS potential error is given by

$$\varepsilon_{\text{pot}} := \frac{1}{M}\sqrt{\sum_{j=1}^{M}\left(\phi_{\text{REF}}(\boldsymbol{x}_j) - \phi_{\text{P2NFFT}}(\boldsymbol{x}_j)\right)^2}\,.$$

For the following run time measurements we tuned the parameters of P2NFFT in order to hold $\varepsilon_{\text{pot}} < 10^{-5}$.

In Figure 6.4 we show the wall clock time measurements of Alg. 3 (P2NFFT) for the silica melt test case with 829440 particles on a BlueGene/P architecture using up to 16384 cores. For comparison purposes we show the perfect strong scaling time

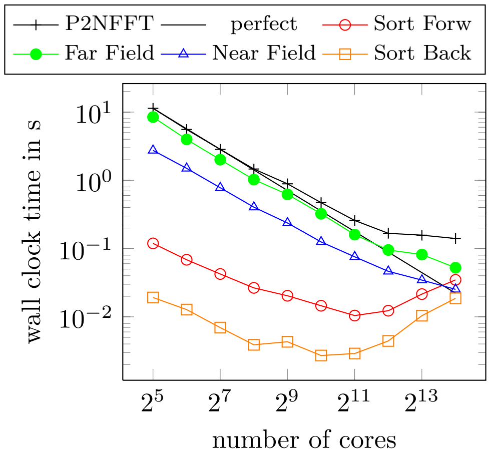

FIGURE 6.4. *Alg. 3 (P2NFFT) for a silica melt with number of nonequispaced nodes* $M = 829440$*, RMS-potential error* $\varepsilon_{\text{pot}} = 10^{-5}$*, regularization parameters* $\varepsilon_I = \varepsilon_B = 0.0078$*, pruned input FFT size* $\boldsymbol{N} = (512, 512, 512)^\top$*, oversampled FFT size* $\boldsymbol{n} = (576, 576, 576)^\top$*, pruned output FFT size* $\boldsymbol{L} = (174, 174, 174)^\top$ *and window cutoff parameter* $m = 4$.

(perfect) of the first recorded time. In addition, we add the wall clock time of the most time consuming parts of this algorithm. These are the forward sorting Step 1 (Forw Sort), the far field computation Steps 2, 3 and 4 (Far Field), the near field computation Step 5 (Near Field) and the backward sorting Step 7 (Back Sort) of the P2NFFT Alg. 3. Note that the individual wall clock time measurements of the adjoint PNFFT Step 2 (adj. PNFFT) and the PNFFT Step 4 (PNFFT) are given in Figure 6.1. We stress that the pruned FFT output size $\boldsymbol{L} = (174, 174, 174)^\top$ is significantly smaller than the oversampled FFT size $\boldsymbol{n} = (576, 576, 576)^\top$ for this test case, i.e., only 2.8% of the oversampled FFT output are necessary to compute the convolution Steps of the PNFFT and its adjoint. This problem arises from the fact that we must scale all the nonequispaced nodes in order to fulfill $\|\boldsymbol{x}_j\|_2 < \frac{1}{4} - \varepsilon_B$ for all $j = 1, \dots, M$. However, our algorithm takes care of this fact since we apply parallel pruned FFT and use the data decomposition of the truncated Torus $\mathbb{T}^3_{\boldsymbol{C}}$.

Although the near field and far field computations show good strong scaling, we observe that the sorting steps are the most limiting factor for strong scaling of the P2NFFT algorithm. Since the parallel sorting algorithm calls a `MPI_Alltoallv` we see the typical almost linear increase of sorting time from 2048 to 16384 processes. Note that using 16384 cores implies a very small number of 51 local nodes for every process. However, our P2NFFT algorithm uses a common three-dimensional data decomposition. Therefore, for several applications, e.g. molecular dynamics, the forward and backward sort can be omitted if the nodes are given in the correct data decomposition at the beginning of the P2NFFT algorithm.

In Figure 6.5 we show the wall clock time measurements of Alg. 3 (P2NFFT) for the silica melt test case with 9447840 particles on a BlueGene/P architecture using up to 65536 cores. Finally, in Figure 6.6 we see the wall clock time measurements of Alg. 3

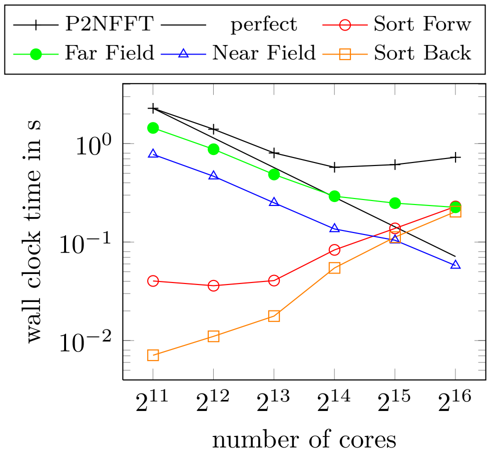

FIGURE 6.5. *Alg. 3 (P2NFFT) for a silica melt with number of nonequispaced nodes* $M = 9447840$, *RMS-potential error* $\varepsilon_{\text{pot}} = 10^{-5}$, *regularization parameters* $\varepsilon_I = \varepsilon_B = 0.0039$, *pruned input FFT size* $\boldsymbol{N} = (1024, 1024, 1024)^\top$, *oversampled FFT size* $\boldsymbol{n} = (1152, 1152, 1152)^\top$, *pruned output FFT size* $\boldsymbol{L} = (340, 340, 340)^\top$ *and window cutoff parameter* $m = 4$.

(P2NFFT) for the silica melt test case with 103680000 particles on a BlueGene/P architecture using up to 65536 cores. A comparison of Figure 6.5 and Figure 6.6 yields a better scaling of the larger test case since the quota of sorting decreases. In both test cases the wall clock time of the sorting steps increases almost linearly with the number of processes and prevents a good scaling of the over all run time.

7. Conclusions. In this paper we have presented new parallel algorithms for computing the nonequispaced fast Fourier transform and its adjoint on distributed memory architectures. We implemented and published these algorithms in the PNFFT software library [36]. To our knowledge this is the first publicly available massively parallel NFFT implementation. Run time tests on a BlueGene/P system using up to 65536 cores showed a good scalability of our implementation. Furthermore, we applied the parallel nonequispaced Fourier transform to derive a parallel fast summation algorithm. As we tested our parallel fast summation algorithm on a BlueGene/P system using up to 65536 cores it turned out that the sorting of particles is the most limiting factor for strong scaling.

Acknowledgments. This work was partly supported by the BMBF under grant 01IH08001B. We are grateful to the Jülich Supercomputing Center for providing the computational resources on Jülich BlueGene/P (JuGene). We wish to thank Dr. Franz Gähler, who supplied the silica melt test case. Furthermore, we gratefully

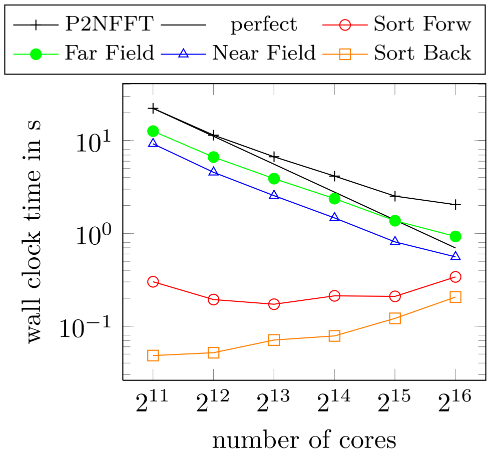

FIGURE 6.6. *Alg. 3 (P2NFFT) for a silica melt with number of nonequispaced nodes* $M = 103680000$*, RMS-potential error* $\varepsilon_{\text{pot}} = 10^{-5}$*, regularization parameters* $\varepsilon_I = \varepsilon_B = 0.002$*, pruned input FFT size* $\boldsymbol{N} = (2048, 2048, 2048)^\top$*, oversampled FFT size* $\boldsymbol{n} = (2304, 2304, 2304)^\top$*, pruned output FFT size* $\boldsymbol{L} = (674, 674, 674)^\top$ *and window cutoff parameter* $m = 4$*.*

acknowledge the help of Dr. Michael Hofmann on the parallel sorting algorithms and the help of Rene Halver on the implementation of the parallel linked cell algorithm.

REFERENCES

[1] B.W.H. van Beest and G.J. Kramer: *Force fields for silicas and aluminophosphates based on ab initio calculations.* Physical Review Letters, 64(16):1955–1958, apr 1990, ISSN 0031-9007.
[2] G. Beylkin: *On the fast Fourier transform of functions with singularities.* Appl. Comput. Harmon. Anal., 2:363 – 381, 1995.
[3] H. Dachsel, M. Hofmann, and G. Rünger: *Library Support for Parallel Sorting in Scientific Computations.* In *Proc. of the 13th International Euro-Par Conference*, vol. 4641 of *LNCS*, pp. 695–704. Springer, 2007, ISBN 978-3-540-74465-8.
[4] M. Deserno and C. Holm: *How to mesh up Ewald sums. I. A theoretical and numerical comparison of various particle mesh routines.* J. Chem. Phys., 109:7678 – 7693, 1998.
[5] H.Q. Ding, R.D. Ferraro, and D.B. Gennery: *A portable 3d FFT package for distributed-memory parallel architectures.* In *PPSC*, pp. 70 – 71, 1995.
[6] A.J.W. Duijndam and M.A. Schonewille: *Nonuniform fast Fourier transform.* Geophysics, 64:539 – 551, 1999.
[7] A. Dutt and V. Rokhlin: *Fast Fourier transforms for nonequispaced data.* SIAM J. Sci. Stat. Comput., 14:1368 – 1393, 1993.
[8] H. Eggers, T. Knopp, and D. Potts: *Field inhomogeneity correction based on gridding reconstruction.* IEEE Trans. Med. Imag., 26:374 – 384, 2007.
[9] B. Elbel and G. Steidl: *Fast Fourier transform for nonequispaced data.* In C.K. Chui and L.L. Schumaker (eds.): *Approximation Theory IX*, pp. 39 – 46, Nashville, 1998. Vanderbilt University Press.
[10] M. Eleftheriou, J.E. Moreira, B.G. Fitch, and R.S. Germain: *A volumetric FFT for BlueGene/L.* In T.M. Pinkston and V.K. Prasanna (eds.): *HiPC*, vol. 2913 of *Lecture Notes in Computer Science*, pp. 194 – 203. Springer, 2003, ISBN 3-540-20626-4.
[11] U. Essmann, L. Perera, M.L. Berkowitz, T. Darden, H. Lee, and L.G. Pedersen: *A smooth particle mesh Ewald method.* J. Chem. Phys., 103:8577 – 8593, 1995.
[12] B. Fang, Y. Deng, and G. Martyna: *Performance of the 3D FFT on the 6D network torus QCDOC parallel supercomputer.* Computer Physics Communications, 176(8):531 – 538, apr 2007, ISSN 00104655.
[13] M. Fenn and G. Steidl: *Fast NFFT based summation of radial functions.* Sampl. Theory Signal Image Process., 3:1 – 28, 2004.
[14] J.A. Fessler and B.P. Sutton: *Nonuniform fast Fourier transforms using min-max interpolation.*

IEEE Trans. Signal Process., 51:560 – 574, 2003.

[15] K. Fourmont: *Non equispaced fast Fourier transforms with applications to tomography.* J. Fourier Anal. Appl., 9:431 – 450, 2003.

[16] M. Frigo and S.G. Johnson: *The design and implementation of FFTW3.* Proceedings of the IEEE, 93:216 – 231, 2005.

[17] M. Frigo and S.G. Johnson: *FFTW, C subroutine library*, 2009. http://www.fftw.org.

[18] L. Greengard and J.Y. Lee: *Accelerating the nonuniform fast Fourier transform.* SIAM Rev., 46:443 – 454, 2004.

[19] M. Griebel, S. Knapek, and G. Zumbusch: *Numerical simulation in molecular dynamics*, vol. 5 of *Texts in Computational Science and Engineering.* Springer, Berlin, 2007.

[20] W. Gropp, E. Lusk, and R. Thakur: *Using MPI-2: Advanced Features of the Message-Passing Interface.* MIT Press, Cambridge, MA, USA, 1999, ISBN 0262571331.

[21] F. Hedman and A. Laaksonen: *Ewald summation based on nonuniform fast Fourier transform.* Chem. Phys. Lett., 425:142 – 147, 2006.

[22] R.W. Hockney and J.W. Eastwood: *Computer simulation using particles.* Taylor & Francis, Inc., Bristol, PA, USA, 1988.

[23] M. Hofmann and G. Rünger: *Fine-grained Data Distribution Operations for Particle Codes.* In M. Ropo, J. Westerholm, and J. Dongarra (eds.): *Recent Advances in Parallel Virtual Machine and Message Passing Interface, 16th European PVM/MPI Users Group Meeting*, vol. 5759 of *LNCS*, pp. 54–63. Springer, 2009, ISBN 978-3-642-03769-6.

[24] J.I. Jackson, C.H. Meyer, D.G. Nishimura, and A. Macovski: *Selection of a convolution function for Fourier inversion using gridding.* IEEE Trans. Med. Imag., 10:473 – 478, 1991.

[25] I. Kabadshow and H. Dachsel: *The Error-Controlled Fast Multipole Method for Open and Periodic Boundary Conditions.* In G. Sutmann, P. Gibbon, and T. Lippert (eds.): *Fast Methods for Long-Range Interactions in Complex Systems*, IAS-Series, pp. 85 – 113, Jülich, 2011. Forschungszentrum Jülich, ISBN 9783893367146.

[26] J. Keiner, S. Kunis, and D. Potts: *NFFT 3.0, C subroutine library.* http://www.tu-chemnitz.de/~potts/nfft.

[27] J. Keiner, S. Kunis, and D. Potts: *Using NFFT3 - a software library for various nonequispaced fast Fourier transforms.* ACM Trans. Math. Software, 36:Article 19, 1 – 30, 2009.

[28] S. Kunis and S. Kunis: *The nonequispaced FFT on graphics processing units.* PAMM, Proc. Appl. Math. Mech., 12, 2012.

[29] S. Kunis and D. Potts: *Stability results for scattered data interpolation by trigonometric polynomials.* SIAM J. Sci. Comput., 29:1403 – 1419, 2007.

[30] S. Kunis, D. Potts, and G. Steidl: *Fast Gauss transform with complex parameters using NFFTs.* J. Numer. Math., 14:295 – 303, 2006.

[31] N. Li: *2DECOMP&FFT, Parallel FFT subroutine library.* http://www.2decomp.org.

[32] N. Li and S. Laizet: *2DECOMP & FFT - A Highly Scalable 2D Decomposition Library and FFT Interface.* In *Cray User Group 2010 conference*, pp. 1 – 13, Edinburgh, 2010.

[33] MPI Forum: *MPI: A Message-Passing Interface Standard. Version 2.2*, September 2009. http://www.mpi-forum.org.

[34] D. Pekurovsky: *P3DFFT, Parallel FFT subroutine library.* http://code.google.com/p/p3dfft.

[35] M. Pippig: *PFFT, Parallel FFT subroutine library*, 2011. http://www.tu-chemnitz.de/~mpip/software.

[36] M. Pippig: *PNFFT, Parallel Nonequispaced FFT subroutine library*, 2011. http://www.tu-chemnitz.de/~mpip/software.

[37] M. Pippig: *PFFT - An extension of FFTW to massively parallel architectures.* Preprint TU Chemnitz, Preprint 6, 2012.

[38] M. Pippig and D. Potts: *Particle simulation based on nonequispaced fast Fourier transforms.* In G. Sutmann, P. Gibbon, and T. Lippert (eds.): *Fast Methods for Long-Range Interactions in Complex Systems*, IAS-Series, pp. 131 – 158, Jülich, 2011. Forschungszentrum Jülich, ISBN 9783893367146.

[39] S. Plimpton: *Parallel FFT subroutine library.* http://www.sandia.gov/~sjplimp/docs/fft/README.html.

[40] D. Potts and G. Steidl: *Fast summation at nonequispaced knots by NFFTs.* SIAM J. Sci. Comput., 24:2013 – 2037, 2003.

[41] D. Potts, G. Steidl, and A. Nieslony: *Fast convolution with radial kernels at nonequispaced knots.* Numer. Math., 98:329 – 351, 2004.

[42] D. Potts, G. Steidl, and M. Tasche: *Fast Fourier transforms for nonequispaced data: A tutorial.* In J.J. Benedetto and P.J.S.G. Ferreira (eds.): *Modern Sampling Theory: Mathematics and Applications*, pp. 247 – 270, Boston, MA, USA, 2001. Birkhäuser.

[43] G. Steidl: *A note on fast Fourier transforms for nonequispaced grids.* Adv. Comput. Math., 9:337 – 353, 1998.

[44] D. Takahashi: *An Implementation of Parallel 3-D FFT with 2-D Decomposition on a Massively Parallel Cluster of Multi-core Processors.* In R. Wyrzykowski, J. Dongarra, K. Karczewski, and J. Wasniewski (eds.): *Parallel Processing and Applied Mathematics*, vol. 6067 of *Lecture Notes in Computer Science*, pp. 606 – 614. Springer Berlin / Heidelberg, 2010, ISBN 978-3-642-14389-2.

[45] T. Volkmer: *OpenMP parallelization in the NFFT software library.* Preprint TU Chemnitz, Preprint 7, 2012.

[46] A.F. Ware: *Fast approximate Fourier transforms for irregularly spaced data.* SIAM Rev., 40:838 – 856, 1998.