Integrating an N-body problem with SDC and PFASST

Robert Speck[1], Daniel Ruprecht[1,5], Rolf Krause[1], Matthew Emmett[2], Michael Minion[3], Mathias Winkel[4], and Paul Gibbon[4]

1 Introduction

Particle methods are an attractive approach for solving complex three-dimensional flow problems since they are naturally adaptive [4]. In this work, we utilize a particle description based on vorticity to discretize the Navier-Stokes equations in space, which results in a first-order initial value ODE for the particles' positions and vorticities. When highly accurate solutions to the initial value problem are required, it is usually more efficient to use higher-order temporal integration schemes. Spectral Deferred Correction (SDC) methods [6] are an elegant way to achieve high-order time integration by using simple low-order schemes in an iterative fashion. While a single time-step of an SDC method is usually more expensive in terms of computation time than a step of a classical Runge-Kutta scheme, SDC can be competitive in terms of time-to-solution required for a fixed accuracy [3].

The temporal integration in vortex methods requires the evaluation of an N-body problem for each function evaluation. This evaluation can be parallelized efficiently by a spatial distribution of the particles over multiple cores. However, the strong scalability of this approach is limited when the number of degrees-of-freedom per core becomes too small (similar to domain decomposition techniques for mesh-based methods). Time-parallel methods are one possible approach to speed up simulations beyond this saturation point, with early approaches dating back to [18]. A very general scheme is Parareal [15], which allows arbitrary integration schemes to be used in a black-box fashion. A detailed mathematical analysis of Parareal is conducted in [8] and comprehensive lists of references can be found e. g. in [17, 20]. The drawback of Parareal is that the parallel efficiency is formally bounded by $1/K$ where K is the number of iterations required for convergence. The *Parallel Full Approximation Scheme in Space and Time* (PFASST) method for parallelizing SDC methods in time is introduced in [7, 17]. By combining the iterations of SDC with the iterations of Parareal, it significantly relaxes the efficiency bound of Parareal and further enhances the competitiveness of integration schemes based on SDC.

[1]Institute of Computational Science, Università della Svizzera italiana, Lugano, Switzerland, {robert.speck,daniel.ruprecht,rolf.krause}@usi.ch · [2]Lawrence Berkeley National Laboratory, Berkeley (USA), mwemmett@lbl.gov · [3]Institute for Computational and Mathematical Engineering, Stanford University, Stanford (USA), mlminion@gmail.com · [4]Jülich Supercomputing Centre, Jülich (Germany), {m.winkel,p.gibbon}@fz-juelich.de · [5]Mathematisches Institut, Heinrich-Heine-Universität Düsseldorf, Düsseldorf (Germany).

The present paper investigates the accuracy of integrating a particle-based discretization of the 3D Navier-Stokes equations in time using SDC and PFASST. We are not aware of any other studies that investigate SDC integration methods in conjunction with particle-based spatial solvers aside from a small, one-dimensional N-body example in [2] for *Revisionist Integral Deferred Corrections* (RIDC).

Since this work focuses on the accuracy of the temporal discretization, the N-body problem is solved directly with $\mathscr{O}(N^2)$-complexity, limiting the presented studies to rather small numbers of particles. The unfavorable quadratic complexity can be overcome by computing approximate interactions using e. g. Barnes-Hut tree codes [1] or the Fast Multipole Method [11]. Results on the strong scaling of PFASST on extreme scales, simulating merely 4 million particles on up to 262,144 cores, are reported in [26], where the massively parallel Barnes-Hut tree code PEPC [9, 10, 23, 24, 27] is applied. There, however, only a very brief discussion of accuracy is given, aiming solely at identifying parameters that generate time-parallel and time-serial solutions of comparable quality that allow for a meaningful comparison in terms of runtimes. Here, accuracy of the method is addressed in more detail, including a comparison with a standard Runge-Kutta scheme.

We briefly describe SDC and PFASST in Section 2 and present accuracy studies in Section 3. In Section 4 we summarize our results and comment on how further efficiency can be achieved for particle-based methods as a prelude to the large-scale simulations performed in [26].

2 Parallel in Time Integration using Spectral Deferred Corrections

Discretizing the vorticity-velocity formulation of the Navier-Stokes equations with N particles results in an initial value problem of the form (see e. g. [4])

$$\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{y}(t) = f(\mathbf{y}(t)), \quad \mathbf{y}(0) = \mathbf{y}_0 \in \mathbb{R}^{6N}, \quad t \in [0,T] \tag{1}$$

where the right-hand side f denotes the sum of all mutual particle interactions (commonly via regularized smoothing kernels) and $\mathbf{y} \in \mathbb{R}^{6N}$ is a vector containing $3D$ positions and vorticities of N particles. Without further approximation, directly evaluating f is of $\mathscr{O}(N^2)$-complexity. This can be significantly reduced using multipole approximations with either Barnes-Hut tree-codes [22] or the Fast Multipole Method [5] at the cost of increased spatial approximation errors. In this work, however, we focus on errors from temporal discretization, and hence f is evaluated exactly with full accuracy and quadratic complexity. To solve the initial value problem (1), classical explicit time-stepping algorithms such as fourth-order Runge-Kutta schemes are commonly used, see e. g. [19]. Here, we use SDC methods [6], which can easily produce high-order time integration schemes from simple low-order methods and which can be parallelized in time using PFASST [7, 17].

Let $0 = t_0 < t_1 < t_2 < \ldots < t_M = T$ denote a discretization of the time-interval $[0,T]$, and let $t_m \leq \tau_1^m < \tau_2^m < \ldots \tau_J^m \leq t_{m+1}$ denote a set of quadrature points in the interval $[t_m, t_{m+1}]$, see [14] for details on the choice of these nodes. For brevity, we fix m and write τ_j instead of τ_j^m for all $j = 0,\ldots,J$. Moreover, let $\mathbf{y}_j$ denote an approximation to $\mathbf{y}(t_j)$, $j = 0,\ldots,J$. Starting from the equivalent Picard formulation of Eq. (1), the key ingredient of SDC is the spectral approximation

$$\mathbf{S}_j^{j+1} f = \sum_{l=0}^{J} \alpha_{j,l} f(\mathbf{y}_l) \approx \int_{\tau_j}^{\tau_{j+1}} f(\mathbf{y}(\tau))\,\mathrm{d}\tau \tag{2}$$

with quadrature weights $\alpha_{j,l} \in \mathbb{R}$. Then, the $k+1$ explicit update for $\mathbf{y}_{j+1}$ at node $j+1$ using low-order explicit Euler is evaluated as

$$\mathbf{y}_{j+1}^{k+1} = \mathbf{y}_j^{k+1} + \Delta\tau_j \left[f(\mathbf{y}_j^{k+1}) - f(\mathbf{y}_j^k) \right] + \mathbf{S}_j^{j+1} f^k, \tag{3}$$

where $\Delta\tau_j = \tau_{j+1} - \tau_j$, k is the iteration index and $\mathbf{y}_j^0$ is some provisional solution computed at the nodes τ_j. For K iterations with a first-order propagator, SDC formally results in a Kth-order time integrator, provided the quadrature approximation is accurate enough. On the other hand, using M Gauss-Lobatto quadrature nodes yields a method of order $2M-2$, provided the number of iterations K is large enough. We refer to [6, 12] for more details and properties of this approach.

To introduce parallel time-stepping we briefly review the Parareal approach [15]. Here, temporal parallelization of (1) is imposed by iterating over two integration schemes, a fast and inaccurate one (the "coarse" propagator) denoted typically as $\mathscr{G}$ and a slow but accurate one (the "fine" propagator) labeled $\mathscr{F}$. While a classical time-marching scheme computes a sequence of solutions

$$\mathbf{y}_{m+1} = \mathscr{F}(\mathbf{y}_m),\ t_m \in [0,T], \tag{4}$$

Parareal replaces (4) by the iteration

$$\mathbf{y}_{m+1}^{k+1} = \mathscr{G}(\mathbf{y}_m^{k+1}) + \mathscr{F}(\mathbf{y}_m^k) - \mathscr{G}(\mathbf{y}_m^k),\ m = 0,\ldots,M-1 \tag{5}$$

where $k \geq 0$ is again the iteration index. The key here is that if the solution from iteration k is known, the expensive evaluation of the terms $\mathscr{F}(\mathbf{y}_m^k)$ can be done in parallel for multiple m. Then, a correction is propagated from t_0 to t_M through the cheap yet serial computation of the terms $\mathscr{G}(\mathbf{y}_m^{k+1})$. The Parareal iteration (5) converges to a solution of the same accuracy as obtained by running $\mathscr{F}$ in serial (i. e. by computing (4)). For N_{it} iterations of Parareal and N_{p} processors, the speedup achievable is bound by $N_{\mathrm{p}}/N_{\mathrm{it}}$, see [17].

To improve parallel efficiency, the PFASST algorithm intertwines SDC integrators of different accuracy for $\mathscr{G}$ and $\mathscr{F}$ with the iterations of Parareal. In addition to multiple levels in time, PFASST can benefit from spatial coarsening, as used in e. g. multi-grid techniques. Furthermore, PFASST employs Full Approximation Scheme (FAS) corrections to increase the accuracy of SDC iterations on coarse levels. Many details of SDC, Parareal and PFASST have been omitted here for brevity, and the reader is referred to the more detailed discussions in e. g. [6, 7, 15, 17, 20, 26].

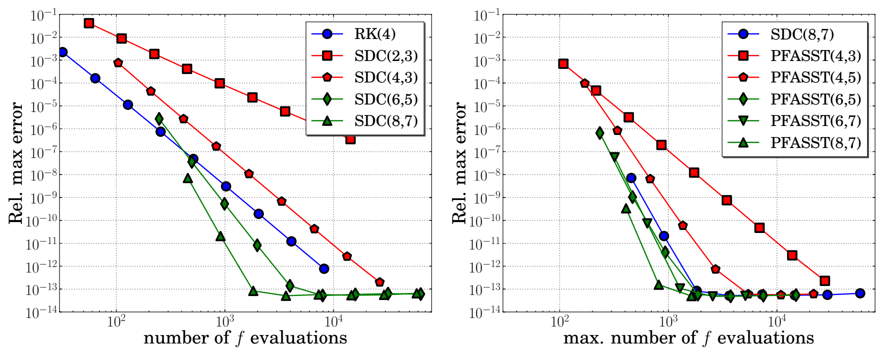
3 Numerical Results

To test SDC and PFASST in the framework of particle simulations we use a standard spherical vortex sheet setup as discussed e. g. in [21, 25] with $N = 10{,}000$ particles (i. e. 60,000 degrees-of-freedom), a sixth-order algebraic kernel for the regularization, and final time $T = 32$. For convenience, the direct evaluation of f has been parallelized in space using 64 processors of the Intel Cluster JUROPA at Jülich Supercomputing Centre [13]. Reported errors are computed using a reference solution generated by an 8th-ordered SDC method with 2,048 time-steps.

Figure 1(a) shows the relative maximum error against the number of evaluations of f for the standard RK4 method (denoted RK(4)) and SDC with different numbers of iterations and Gauss-Lobatto quadrature nodes (denoted SDC(X,Y) for X iterations and Y Gauss-Lobatto nodes). The left-most markers correspond to 8 time-steps, the rightmost to 2,048. Here, the order of convergence of SDC equals the number of performed iterations, since the number of quadrature nodes is high enough, see [6]. Increasing the order of SDC (i. e. increasing the number of iterations) reduces the error substantially for a given number of function evaluations. Hence, if solutions of moderate or high accuracy are sought, e. g. below 10^{-7}, higher-order SDC methods are more efficient in terms of f evaluations than RK4 or lower-order SDC methods.

Although fourth-order SDC (realized here by SDC$(4,3)$ in Figure 1(a)) is more expensive than the classical RK4, one advantage of SDC methods is that the order of convergence can be easily controlled by the interplay of quadrature nodes and iterations. Implementing Runge-Kutta schemes of higher-order, on the other hand, involves tedious and error-prone code re-implementations. Moreover, SDC can easily treat stiff and non-stiff parts of the right-hand side separately and/or with differing time-steps accuracy [16]. More importantly for the present study is that SDC methods can be parallelized in time using PFASST.

In Figure 1(b), we show the relative maximum error of two-level PFASST runs on four time-processes, i. e. with four times more processors than the space-parallel/time-serial SDC runs. Here, the x-axis depicts the maximum number of f evaluations performed by one process (maximum is typically attained on the last time-rank), counting evaluations of f in $\mathscr{F}$ as well as in $\mathscr{G}$, see also the discussion on Figure 2 below. We do not employ spatial coarsening, i. e. the propagators $\mathscr{F}$ and $\mathscr{G}$ differ only in the number of temporal quadrature nodes. We also compare the PFASST runs to a serial SDC run with 8 iterations on 7 Gauss-Lobatto nodes. PFASST, while being more expensive in terms of f evaluations, also yields a higher accuracy for a given number of iterations and quadrature nodes in our case. This is due to the fact that each iteration contains sweeps at both fine and coarse levels, so



(a) SDC vs. RK(4) (b) PFASST vs. SDC$(8,7)$

Fig. 1 Error versus number of evaluations of the right-hand side f for SDC(X,Y) and the maximum number of f evaluations performed by one process (typically on the last time-rank) for PFASST(X,Y) on 4 time-processes. Direct particle simulation of a spherical vortex sheet with 10,000 particles, sixth-order algebraic kernel, and up to 2,048 time-steps.

that effective number of SDC sweeps is higher than the number of iterations. Note again that higher-order schemes show much better efficiency in terms of accuracy versus f evaluations.

Besides providing higher accuracy, PFASST also introduces an additional layer of parallelism. Provided that the spatial parallelization is already saturated, the application of PFASST can push the strong-scaling limit further by distributing the temporal integration across multiple time-processes, as shown in [26]. To shed more light on this concept, Figure 2 shows the number of f evaluations required by PFASST$(8,7)$ on one to eight time-processors with 16 time-steps. As a reference, we choose SDC with 12 iterations on 7 quadrature nodes to obtain a comparable accuracy to PFASST$(8,7)$ when tested against a very fine resolved reference solution: Both schemes provide an accuracy of approx. 10^{-13} with 16 time-steps. The number of f evaluations in $\mathscr{G}$ are highlighted in blue and hatched, the ones in $\mathscr{F}$ in red. Note that there is no coarse propagator in the single-level SDC scheme. The second bar, denoted $P(8,7,1)$, corresponds to PFASST with 8 iterations and 7 quadrature nodes on one time-processor. When run on one time-processors, PFASST reduces to a multi-level SDC scheme. Comparing SDC$(12,7)$ and PFASST$(8,7,1)$ we note that by switching from a single-level SDC scheme of a given order to a multi-level SDC scheme with comparable accuracy, the number of iterations are reduced (from 12 to 8 in our case). On the other hand, significant additional costs are introduced due to the additional f evaluations required by the coarse step and the transfer operations between fine and coarse quadrature nodes. However, the extra computational work of the multi-level SDC scheme can be distributed across multiple processors as demonstrated in the three remaining bars, which correspond to PFASST$(8,7)$ on two, four and eight processors. Hence, if the workload of PFASST$(8,7)$ is distributed across sufficiently many processors, then the total runtime becomes smaller than the time-to-solution of the serial SDC$(12,7)$ method. This is highlighted by

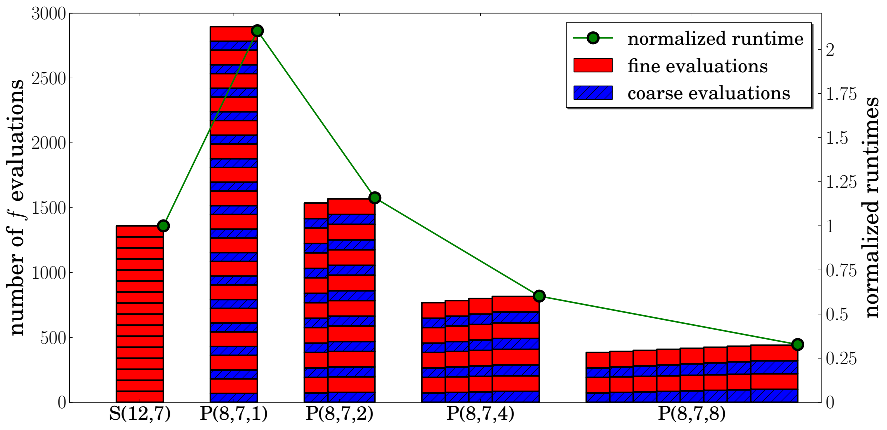


Fig. 2 Distribution of f evaluations on the coarse and fine level for $\mathrm{SDC}(12,7)$ (abbreviated by $\mathrm{S}(12,7)$) and $\mathrm{PFASST}(8,7,Z)$ with $Z = 1,2,4,8$ time-processes (abbrev. by $\mathrm{P}(8,7,Z)$). One block of coarse and fine evaluations corresponds to one time-step on one time-process (SDC is serial in time and evaluates only on the fine level). Depicted runtimes are normalized with respect to the SDC runtime (418 sec. for our test setup with with 10,000 particles, sixth-order algebraic kernel). All runs yield comparable accuracy for 16 time-steps.

the green line, which shows the runtime normalized by the runtime of $\mathrm{SDC}(12,7)$. While $\mathrm{PFASST}(8,7,2)$ is still slightly slower than $\mathrm{SDC}(12,7)$, $\mathrm{PFASST}(8,7,4)$ and $\mathrm{PFASST}(8,7,8)$ show significant speedup. The cost of enlarging the problem, i. e. switching from SDC to a multi-level scheme, is compensated by the fact that this multi-level approach is amenable to parallelization while SDC itself is not.

4 Conclusion and Outlook

In this work, we have investigated the accuracy and convergence order of Spectral Deferred Correction (SDC) methods and their parallelization using the PFASST method. SDC provides a reliable, flexible, and generic mechanism to generate high-order and high-accuracy time integrators. We have shown that SDC and its time-parallel variant PFASST provide the theoretically expected convergence orders and accuracies on an example particle problem. In contrast to classical Runge-Kutta schemes, the convergence order and/or accuracy of SDC methods can easily be controlled by changing the number of iterations and/or quadrature points used, and the use of higher-order SDC methods allows much larger time-steps and hence fewer evaluations of the right-hand side. This is consistent with the increase of accuracy and also stability regions observed in [6].

Another key advantage of SDC methods is that that they can be parallelized in time with PFASST. Here, the careful union of fine and coarse SDC iterations leads to a high-order parallel-in-time integration scheme which relaxes Parareal's bound on parallel efficiency and can provide significant speedup beyond space-only parallelization. In our test case, PFASST is more accurate for a given number of quadrature nodes and iterations, although for enough iterations both SDC and PFASST eventually provide the same solution. Moreover, we have demonstrated how the principle of “doing more to be faster” paves the way for temporal parallelism: the introduction of (possibly multi-level) coarsening in space and time increases the number of f evaluations significantly but also allows work to be distributed across many time-processors.

Here PFASST is used with temporal coarsening only, while considerably more parallel efficiency can be obtained by introducing both spatial and temporal coarsening. While grid-based spatial coarsening by multi-grid techniques is well understood, spatial coarsening of particles systems is less straightforward. One possibility is to control the quality of the approximation of f using multipole methods instead of direct summation [26]. Thus, the use of fast summation algorithms not only allows extreme-scale simulations as demonstrated in [27], but also introduces a promising way of particle-based spatial “coarsening”.

Acknowledgements This research is partly funded by the Swiss “High Performance and High Productivity Computing” initiative HP2C; the Director, DOE Office of Science, Office of Advanced Scientific Computing Research, Office of Mathematics, Information, and Computational Sciences, Applied Mathematical Sciences Program, under contract DE-SC0004011; and the ExtreMe Matter Institute (EMMI) in the framework of the German Helmholtz Alliance HA216. Computing resources were provided by Jülich Supercomputing Centre under project JZAM04.

References

1. Barnes, J.E., Hut, P.: A hierarchical $\mathscr{O}(N \log N)$ force-calculation algorithm. Nature **324**(6096), 446–449 (1986)
2. Christlieb, A., Macdonald, C., Ong, B.: Parallel high-order integrators. SIAM Journal on Scientific Computing **32**(2), 818 (2010)
3. Christlieb, A., Ong, B., Qiu, J.: Comments on high order integrators embedded within integral deferred correction methods. Comm. Appl. Math. Comput. Sci **4**(1), 27–56 (2009)
4. Cottet, G.H., Koumoutsakos, P.: Vortex Methods: Theory and Applications, 2nd edn. Cambridge University Press (2000)
5. Cruz, F.A., Knepley, M.G., Barba, L.A.: PetFMM-A dynamically load-balancing parallel fast multipole library. International Journal for Numerical Methods in Engineering **79**(13), 1577–1604 (2010)
6. Dutt, A., Greengard, L., Rokhlin, V.: Spectral deferred correction methods for ordinary differential equations. BIT Numerical Mathematics **40**(2), 241–266 (2000)
7. Emmett, M., Minion, M.: Toward an efficient parallel in time method for partial differential equations. Comm. App. Math. and Comp. Sci. **7**(1), 105–132 (2012)
8. Gander, M.J., Vandewalle, S.: Analysis of the parareal time-parallel time-integration method. SIAM J. Sci. Comp. **29**(2), 556–578 (2007)
9. Gibbon, P., Speck, R., Karmakar, A., Arnold, L., Frings, W., Berberich, B., Reiter, D., Masek, M.: Progress in mesh-free plasma simulation with parallel tree codes. IEEE Transactions on Plasma Science **38**(9), 2367–2376 (2010). DOI 10.1109/tps.2010.2055165
10. Gibbon, P., Winkel, M., Arnold, L., Speck, R.: PEPC website (2012). URL `http://www.fz-juelich.de/ias/jsc/pepc`

11. Greengard, L., Rokhlin, V.: A fast algorithm for particle simulations. J. Comp. Phys. **73**(2), 325–348 (1987)
12. Huang, J., Jia, J., Minion, M.: Accelerating the convergence of spectral deferred correction methods. Journal of Computational Physics **214**(2), 633–656 (2006)
13. Jülich Supercomputing Centre: JUROPA/HPC-FF website (2012). URL `http://www.fz-juelich.de/jsc/juropa`
14. Layton, A.T., Minion, M.L.: Implications of the choice of quadrature nodes for picard integral deferred corrections methods for ordinary differential equations. BIT Numerical Mathematics **45**(2), 341–373 (2005)
15. Lions, J.L., Maday, Y., Turinici, G.: A ”parareal” in time discretization of PDE’s. C. R. Acad. Sci. – Ser. I – Math. **332**, 661–668 (2001)
16. Minion, M.L.: Semi-implicit spectral deferred correction methods for ordinary differential equations. Communications in Mathematical Sciences **1**(3), 471–500 (2003)
17. Minion, M.L.: A hybrid parareal spectral deferred corrections method. Comm. App. Math. and Comp. Sci. **5**(2), 265–301 (2010)
18. Nievergelt, J.: Parallel methods for integrating ordinary differential equations. Commun. ACM **7**(12), 731–733 (1964)
19. van Rees, W.M., Leonard, A., Pullin, D.I., Koumoutsakos, P.: A comparison of vortex and pseudo-spectral methods for the simulation of periodic vortical flows at high Reynolds numbers. J. Comp. Phys. **230**, 2794–2805 (2011)
20. Ruprecht, D., Krause, R.: Explicit parallel-in-time integration of a linear acoustic-advection system. Computers & Fluids **59**, 72–83 (2012)
21. Salmon, J.K., Warren, M.S., Winckelmans, G.: Fast parallel tree codes for gravitational and fluid dynamical *N*-body problems. Int. J. Supercomp. App. **8**, 129–142 (1994)
22. Speck, R.: Generalized algebraic kernels and multipole expansions for massively parallel vortex particle methods. Ph.D. thesis, Universität Wuppertal (2011)
23. Speck, R., Arnold, L., Gibbon, P.: Towards a Petascale Tree Code: Scaling and Efficiency of the PEPC Library. J. Comp. Sci. **2**, 137–142 (2011)
24. Speck, R., Gibbon, P., Hofmann, M.: Efficiency and scalability of the parallel Barnes-Hut tree code PEPC. In: B. Chapman, F. Desprez, G.R. Joubert, A. Lichnewsky, F.J. Peters, T. Priol (eds.) Parallel Computing: From Multicores and GPU’s to Petascale, *Adv. in Parallel Comp.*, vol. 19, pp. 35–42. IOS Press (2010)
25. Speck, R., Krause, R., Gibbon, P.: Parallel remeshing in tree codes for vortex particle methods. In: K.D. Bosschere, E.H. D’Hollander, G.R. Joubert, D. Padua, F. Peters, M. Sawyer (eds.) Applications, Tools and Techniques on the Road to Exascale Computing, *Advances in Parallel Computing*, vol. 22 (2012)
26. Speck, R., Ruprecht, D., Krause, R., Emmett, M., Minion, M., Winkel, M., Gibbon, P.: A massively space-time parallel N-body solver. In: Proceedings of the SC’12 International Conference for High Performance Computing, Networking, Storage and Analysis (2012). (Accepted)
27. Winkel, M., Speck, R., Hübner, H., Arnold, L., Krause, R., Gibbon, P.: A massively parallel, multi-disciplinary Barnes-Hut tree code for extreme-scale N-body simulations. Comp. Phys. Comm. **183**(4), 880–889 (2012)