A software framework for the portable parallelization of particle-mesh simulations

I. F. Sbalzarini, J. H. Walther†, B. Polasek, P. Chatelain, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos

Institute of Computational Science, ETH Zürich, CH-8092, Switzerland
† also at: DTU, Mechanical Engineering, DK-2800 Lyngby, Denmark.

Abstract. We present a software framework for the transparent and portable parallelization of simulations using particle-mesh methods. Particles are used to transport physical properties and a mesh is required in order to reinitialize the distorted particle locations, ensuring the convergence of the method. Field quantities are computed on the particles using fast multipole methods or by discretizing and solving the governing equations on the mesh. This combination of meshes and particles presents a challenging set of parallelization issues. The present library addresses these issues for a wide range of applications, and it enables orders of magnitude increase in the number of computational elements employed in particle methods. We demonstrate the performance and scalability of the library on several problems, including the first-ever billion particle simulation of diffusion in real biological cell geometries.

1 Introduction

A large number of problems in physics and engineering can be described by particles. Examples include Molecular Dynamics (MD) simulations of nano-devices, Smooth Particle Hydrodynamics (SPH) simulations of astrophysics, and Vortex Methods (VM) for fluid dynamics. The dynamics of particle methods are governed by the interactions of the N computational elements, resulting in an N-body problem with a nominal computational cost of $\mathcal{O}(N^2)$. For short-ranged particle interactions, as in simulations of diffusion [1], the computational cost scales linearly with the number of particles. In the case of long-range interaction potentials such as the Coulomb potential in MD or the Biot-Savart law in VM, Fast Multipole Methods (FMM) [2] reduce the computational cost to $\mathcal{O}(N)$. Alternatively, these long-range interactions can be described by the Poisson equation which, when solved on meshes, results in the hybrid Particle-Mesh (PM) algorithms as pioneered by Harlow [3, 4]. The computational cost of hybrid methods scales as $\mathcal{O}(M)$, where M denotes the number of mesh points used for resolving the field equations. Due to the regularity of the mesh, PM methods are one to two orders of magnitude faster than FMM [5].

The parallelization of PM and FMM techniques is complicated by several factors:

- the simultaneous presence and spatial distribution of particles and meshes prohibits a single optimal way of parallelization,
- complex-shaped computational domains and strong particle inhomogeneities require spatially adaptive domain decompositions,
- particle motion may invalidate the existing domain decomposition, causing rising load imbalance and requiring additional communication in multi-stage ODE integration schemes,
- exploiting the symmetry of the particle interactions requires sending back of ghost contributions to the corresponding real particle,
- inter-particle relations constrain decompositions and data assignment,
- the global nature of the tree data structure hampers the parallel implementation of FMM methods.

Many of the available domain decomposition, load balancing, solver, interpolation, and data communication methods are applicable to a wide range of particle or PM algorithms, regardless of the specific physics that are being simulated [6]. In this paper we present the newly developed Parallel Particle Mesh library PPM [5] and its extension to FMM. The PPM library provides a generic, physics-independent infrastructure for simulating discrete and continuum systems, and it bridges the gap between general libraries and application-specific simulation codes.

The design goals of PPM include ease of use, flexibility, state-of-the-art parallel scaling, good vectorization, and platform portability. The library is portable through the use of standard languages (Fortran 90 and C) and libraries (MPI) and it was successfully compiled and used on distributed memory, shared memory, and vector architectures on 1 to 512 processors. Computational efficiency is achieved by dynamic load balancing, dynamic particle re-distribution, explicit message passing, and the use of simple data structures.

2 Particle Concepts and Particle-Mesh techniques

The use of the PPM library requires that the simulated systems are formulated in the framework of PM algorithms [6]. The field equations are solved using structured or uniform Cartesian meshes. As a result, the physical and computational domains are rectangular or cuboidal in two or three dimensions. Complex geometries are handled by immersed boundaries, through the use of source terms in the corresponding field equations, or through boundary element techniques. Adaptive multi-resolution capabilities are possible using mapping concepts as adapted to particle methods [7].

The simultaneous presence of particles and meshes requires different concurrent domain decompositions. These decompositions divide the computational domain into a minimum number of *sub-domains* with sufficient granularity to provide adequate load balancing. The concurrent presence of different decompositions allows to perform each step of the computational algorithm in its optimal environment with respect to load balance and the computation-to-communication ratio. For the actual computations, the individual sub-domains

are treated as independent problems and extended with *ghost mesh layers* and *ghost particles* to allow for communication between them. *Ghosts* are copies of true mesh points or particles that either reside on a neighboring processor or account for periodic boundary conditions.

The PPM library supports *connections* and *relations* between particles, such as particle pairs, triplets, quadruplets, etc. These relations may describe a physical interaction, such as chemical bonds in molecular systems, or a spatial coherence, such as a triangulation of an immersed boundary or an unstructured mesh.

2.1 Topologies

A *topology* is defined by the decomposition of space into sub-domains with the corresponding boundary conditions, and the assignment of these sub-domains onto processors. Multiple topologies may coexist and library routines are provided to *map* particle and field data between them as described below. Fields are defined on *meshes*, which in turn are associated with topologies. Every topology can hold several meshes.

In order to achieve good load balance, the *SAR heuristic* [8] is used in the PPM library to decide when problem re-decomposition is advised, i.e. when the cost of topology re-definition is amortized by the gain in load balance. Moreover, all topology definition routines can account for the true computational cost of each particle, for example defined by the actual number of its interactions, and the effective speeds of all processors.

The PPM library provides a number of different adaptive domain decomposition techniques for particles, meshes, and volumes, the latter defining geometric sub-domains with neither meshes nor particles present [5]. The sub-domains can be assigned to the processors in various ways. The PPM-internal method assigns contiguous blocks of sub-domains to processors until the accumulated cost for a processor is greater than the theoretical average cost under uniform load distribution. The average is weighted with the relative processor speeds to account for heterogeneous machine architectures. In addition, four different Metis-based [9] and a user-defined assignment are available.

At the external boundaries of the computational domain, Neumann, Dirichlet, free space, symmetric, and periodic boundary conditions are generally supported, but depend on the particular mesh-based solver that is being employed. More involved boundary conditions and complex boundary shapes are represented inside the computational domain by defining connections among the particles, or by using immersed interfaces.

2.2 Data mapping

PPM topologies implicitly define a data-to-processor assignment. Mapping routines provide the functionality of sending particles and field blocks to the proper processor, i.e. to the one that "owns" the corresponding sub-domain(s) of the computational space. Three different mapping types are provided for both particles and field data:

– a *global mapping*, involving an all-to-all communication,
– a *local mapping* for neighborhood communication, and
– *ghost mappings* to update the ghost layers.

The global mapping is used to perform the initial data-to-processor assignment or to switch from one topology to another, whereas the local mapping is mainly used to account for particle motion during a simulation. Communication is scheduled by solving the minimal edge coloring problem using the efficient approximation algorithm by Vizing [10], and connections between particles are appropriately accounted for. Ghost mappings are provided to receive ghost particles or ghost mesh points, and to send ghost contributions back to the corresponding real element, for example after a symmetric particle-particle interaction or a particle-to-mesh interpolation.

All mapping types are organized as stacks. A mapping operation consists of four steps: (1) defining the mapping, (2) pushing data onto the send stack, (3) performing the actual send and receive operations, and (4) popping the data from the receive stack. This architecture allows data stored in different arrays to be sent together to minimize network latency, and mapping definitions to be reused by repeatedly calling the push/send/pop sequence for the same, persisting mapping definition.

3 Particle-Particle interactions

The evaluation of Particle-Particle (PP) interactions is a key component of PM algorithms. The overall dynamics of the system may be governed by local particle interactions, sub-grid scale phenomena may require local particle-based corrections [11], or differential operators can be evaluated on irregular locations [12]. The PPM library implements symmetric and non-symmetric PP computations using a novel type of cell lists [5], Verlet lists [13], and the full $\mathcal{O}(N^2)$ direct method. The last is based on distributing the particles among processors in equal portions, irrespective of their location in space.

4 Particle-Mesh interpolation

All hybrid PM methods involve interpolation of irregularly distributed particle quantities from particle locations onto a regular mesh, and interpolation of field quantities from the grid points onto particle locations. These interpolations are utilized for two purposes, namely: (1) the communication of the particle solver with the field solver, and (2) the reinitialization of distorted particle locations ("remeshing"). The PPM library provides routines that perform these operations. The interpolation of mesh values onto particles readily vectorizes: interpolation is performed by looping over the particles and receiving values from mesh points within the support of the interpolation kernel. Therefore, the values of individual particles can be computed independently. The interpolation of particle values onto the mesh, however, leads to data dependencies as the interpolation

is still performed by looping over particles, but a mesh point may receive values from more than one particle. To circumvent this problem, the PPM library implements the following technique [14]: when new particles are created in the course of remeshing, we assign colors to the particles such that no two particles within the support of the interpolation kernel have the same color. Particle-to-mesh interpolation then visits the particles ordered by color to achieve data independence. In combination with appropriate directives, this coloring scheme enables the compiler to safely vectorize the loops, as confirmed by a test on a NEC SX-5 vector computer. The wall-clock time for interpolating 2 million particles onto a 128^3 mesh using the M_4' interpolation kernel [15] decreases from 30 s to 2.7 s when using the present coloring scheme, and the vector operation ratio increases from 0.4% to 99%.

5 Mesh-based solvers

In PPM, meshes can be used to solve the field equations associated with long-range particle interactions [4], or to discretize the differential operators in the governing equations of the simulated physical system. A large class of pair interaction potentials in particle methods can be described by the Poisson equation as it appears in MD of charged particles via electrostatics (Coulomb potential), fluid mechanics in stream-function/vorticity formulation (Biot-Savart potential), and cosmology (gravitational potential). The PPM library provides fast parallel Poisson solvers based on FFTs and geometric Multi-Grid (MG) in both two and three dimensions. The library architecture is however not limited to Poisson solvers.

6 ODE solvers

Simulations using particle methods entail the solution of systems of ODEs [6]. The PPM library provides a set of explicit integration schemes to solve these ODEs. Parallelism is achieved by mapping the integrator stages of multi-stage schemes – i.e., the intermediate function evaluations – along with the other particle quantities. The set of available integrators currently includes forward Euler with and without super time stepping [16], 2-stage and 4-stage standard Runge-Kutta schemes, Williamson's low-storage third order Runge-Kutta scheme [17], and 2-stage and 3-stage TVD Runge-Kutta schemes [18].

7 Parallel file I/O

File I/O in distributed parallel environments exist in *distributed* and *centralized* modes. By distributed we denote the situation where each processor writes its part of the data to its local file system. Centralized I/O on the other hand produces a single file on one of the nodes, where the data contributions from all processors are consolidated. The PPM library provides a parallel I/O module

which supports both binary and ASCII read and write operations in both modes. Write operations in the centralized mode can concatenate or reduce (sum, replace) the data from individual processors; read operations can transparently split the data in equal chunks among processors, or send an identical copy to each one. To improve performance of the centralized mode, network communication and file I/O are overlapped in time using non-blocking message passing.

8 Parallel FMM

The PPM library uses FMM to evaluate Dirichlet and free-space boundary conditions for the computationally more efficient MG solver. The implementation of the parallel FMM module is based on the topology, tree, and mapping routines provided by the library core. Hereby, the FMM module creates multiple temporary topologies. The first topology comprises the sub-domains defined at the level of the tree that holds at least as many sub-domains as there are processors. Subsequent levels of the FMM tree structure are also declared as PPM topologies. By means of the user-defined assignment scheme, individual processors operate on disjoint sub-trees to minimize the amount of communication. The particles are then mapped according to the finest topology, which contains all the leaf boxes as sub-domains. The PPM tree directly provides index lists to the particles in each box. This allows straightforward computation of the expansion coefficients on the finest level, without requiring communication. The computed leaf coefficients are shifted to parent boxes by recursively traversing the tree toward its root, cf., e.g., [19]. Since the topologies are defined such that each processor holds a disjoint subtree, the expansions can be shifted without communication.

To evaluate the potential at the locations of a set of target particles, these particles are first mapped onto the finest-level PPM topology. A pre-traversal of the tree then decides which expansion coefficients and source particles are needed for each target point. This is done by traversing the tree from the root down to the leafs using a stack data structure. On each level, we check if the corresponding box is already far enough away from the target particle. This is done by comparing the distance (D) between the target particle and the center of the box to the diameter (d) of the box. If the ratio $D/d > \theta$, the expansion of that box is used and the traversal stops.

When evaluating the potential, expansion coefficients or particles from other processors may be needed. Before evaluating the potentials, the expansion coefficients from all processors are thus globally communicated. This can be done since the data volume of the coefficients is much smaller than the original particle data. Required particles are received on demand as additional ghosts using the regular PPM ghost mapping routines.

9 Benchmarks results

Benchmark results are presented for the PPM FMM, for a remeshed SPH (rSPH) client application [20, 5] for the simulation of compressible fluid dynamics, and

for simulations of diffusion in the Endoplasmic Reticulum (ER) of live cells [21, 22]. In addition, preliminary results with a simple PPM MD client have shown it to reach the same performance as the dedicated MD program FASTTUBE [23].

The benchmarks for the FMM are collected on a Linux cluster of 16 2.2 GHz AMD Opteron 248 processors, connected by standard gigabit Ethernet. This architecture is chosen since it provides the most stringent test for the communication-intense tree data structure of the FMM. To have access to larger numbers of processors, the subsequent application benchmarks are preformed on the IBM p690 computer of the Swiss National Supercomputing Centre (CSCS). The machine consists of 32 Regatta nodes with 8 1.3 GHz Power4 processors per node. The nodes are connected by a 3-way Colony switch system. Furthermore, code vectorization is assessed on the NEC SX-5 computer of CSCS.

9.1 The Fast Multipole Method

The test cases for the PPM FMM involve 10^5 source points with a uniformly random distribution in a cubic box. The potential induced by these points is computed at the locations of 10^5 target points, also uniformly randomly distributed in the same cube. Fig. 1(a) shows the wall-clock time as a function of the number of particles. The acceptance factor θ for the tree traversal is set to 1.5, and we vary the expansion order (l). The scaling of the FMM is compared to the $\mathcal{O}(N^2)$ scaling of the direct evaluation method. Fig. 1(b) shows the parallel speedup of the PPM FMM on up to 16 processors of the Linux cluster. The wall-clock time is 452 seconds on 1 processor and 36.8 seconds on 16. The observed loss in efficiency is mainly caused by the global communication of the expansion coefficients.

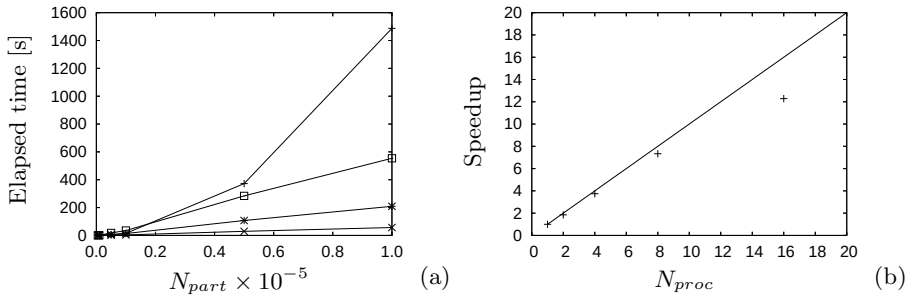


Fig. 1. Performance of the Fast Multipole Method (FMM) implementation in the PPM library. (a) Serial performance of the FMM as function of number of particles and the order (l) of the expansion: -+-: direct calculation; -□-: $l = 9$; -∗-: $l = 5$; -×-: $l = 1$. (b) Parallel speedup using 10^5 particles and $l = 5$ on up to 16 nodes of the 2.2 GHz Opteron Linux cluster. —: linear scaling; +: measurement.

9.2 A Remeshed Smooth Particle Hydrodynamics Application

The first application benchmark considers an rSPH client for the simulation of three-dimensional compressible flows. For the parallel performance to be independent of the particular flow problem, we consider a computational domain fully populated with particles. The speedup and parallel efficiency of the rSPH client are shown in Fig. 2. The maximum number of particles considered for this test case is 268 million with a parallel efficiency of 91% on 128 processors and also 91% on 32 processors. This compares well with the 85% efficiency of the GADGET SPH code by Springel et al. [24] on 32 processors of the same computer model (IBM p690). One time step of a fixed-size simulation using 16.8 million particles takes 196.9 seconds on 1 processor and 7.3 seconds on 128 processors.

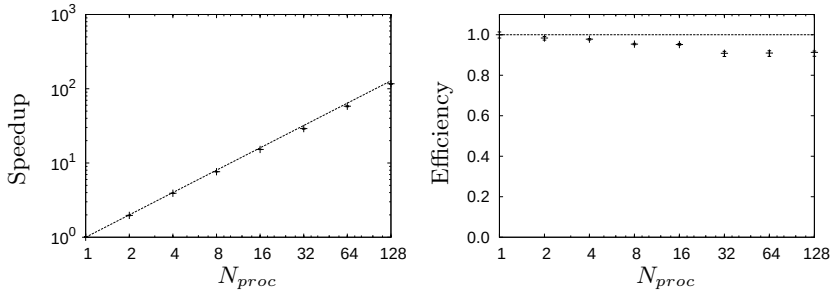
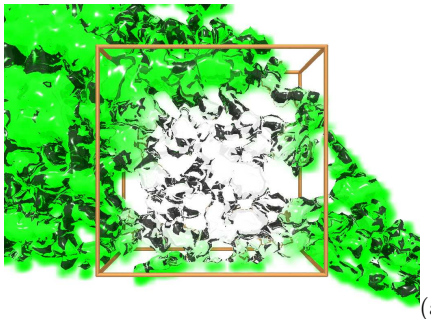


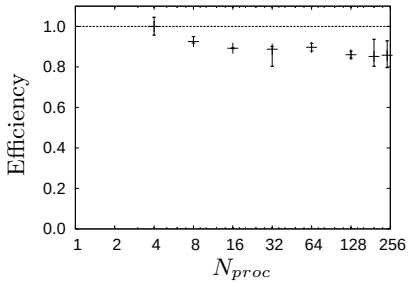
Fig. 2. Parallel speedup and efficiency of the PPM rSPH client for a scaled-size problem starting with 2 million particles on one processor. Each point is averaged from 5 samples, error bars indicate min-max span. Timings are performed on the IBM p690.

9.3 Diffusion in the ER

We present a client application for the simulation of three-dimensional diffusion in the ER, an organelle of live cells. This test demonstrates the capability of the library in handling complex-shaped domains with irregular load distribution. The problem size is fixed at 3.4 million particles, distributed inside the ER. Using an adaptive recursive orthogonal bisection, the complex ER geometry is decomposed, and the sub-domains are distributed among 4 to 242 processors. Fig. 3(a) shows a visualization of the solution, and Fig. 3(b) shows the parallel efficiency of the present PPM client. The simulations sustain 20% of the peak performance of the IBM p690, reaching 250 GFlop/s on 242 processors at 84% efficiency. The load balance is 90 to 95% in all cases, and one time step takes 14 seconds on 4 processors. This client was used to perform simulations of diffusion using up to 1 billion particles on 64 processors, thus demonstrating the good scaling in memory of the PPM library.



(a)



(b)

Fig. 3. (a) Visualization of the simulated concentration distribution inside a real ER as reconstructed from a live cell. The volume indicated by the cube is initially empty and we simulate the diffusive influx. The solution at time 0.25 is shown. The edge length of the box is 50, and the computational diffusion constant is 7.5. (b) Parallel efficiency of the PPM diffusion client for a fixed-size problem with 3.4 million particles distributed inside the ER. Each point is averaged from 5 samples, error bars indicate min-max span. Timings are performed on the IBM p690.

10 Summary

The lack of efficiently parallelized and user friendly software libraries has hindered the wide-spread use of particle methods. We have initiated the development of a generic software framework for hybrid Particle-Mesh simulations. The PPM library described in this paper provides a complete infrastructure for parallel particle and hybrid Particle-Mesh simulations for both discrete and continuum systems. It includes adaptive domain decompositions, load balancing, optimized communication scheduling, parallel file I/O, interpolation, data communication, and a set of commonly used numerical solvers, including a parallel FMM.

We have demonstrated the library's parallel efficiency and versatility on a number of different problems on up to 242 processors. All applications showed parallel efficiencies reaching or exceeding the present state of the art, and favorable run-times on large systems.

References

1. Degond, P., Mas-Gallic, S.: The weighted particle method for convection-diffusion equations. Part 1: The case of an isotropic viscosity. Math. Comput. **53**(188) (1989) 485–507
2. Greengard, L., Rokhlin, V.: The rapid evaluation of potential fields in three dimensions. Lect. Notes Math. **1360** (1988) 121–141
3. Harlow, F.H.: Particle-in-cell computing method for fluid dynamics. Methods Comput. Phys. **3** (1964) 319–343
4. Hockney, R.W., Eastwood, J.W.: Computer Simulation using Particles. Institute of Physics Publishing (1988)

5. Sbalzarini, I.F., Walther, J.H., Bergdorf, M., Hieber, S.E., Kotsalis, E.M., Koumoutsakos, P.: PPM – a highly efficient parallel particle-mesh library for the simulation of continuum systems. J. Comput. Phys. **215**(2) (2006) 566–588
6. Koumoutsakos, P.: Multiscale flow simulations using particles. Annu. Rev. Fluid Mech. **37** (2005) 457–487
7. Bergdorf, M., Cottet, G.H., Koumoutsakos, P.: Multilevel adaptive particle methods for convection-diffusion equations. Multiscale Model. Simul. **4**(1) (2005) 328–357
8. Moon, B., Saltz, J.: Adaptive runtime support for direct simulation Monte Carlo methods on distributed memory architectures. In: Proceedings of the IEEE Scalable High-Performance Computing Conference, IEEE (1994) 176–183
9. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20**(1) (1998) 359–392
10. Vizing, V.G.: On an estimate of the chromatic class of a p-graph. Diskret. Anal. **3** (1964) 25–30 in Russian.
11. Walther, J.H.: An influence matrix particle-particle particle-mesh algorithm with exact particle-particle correction. J. Comput. Phys. **184** (2003) 670–678
12. Eldredge, J.D., Leonard, A., Colonius, T.: A general deterministic treatment of derivatives in particle methods. J. Comput. Phys. **180** (2002) 686–709
13. Verlet, L.: Computer experiments on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. Phys. Rev. **159**(1) (1967) 98–103
14. Walther, J.H., Koumoutsakos, P.: Three-dimensional particle methods for particle laden flows with two-way coupling. J. Comput. Phys. **167** (2001) 39–71
15. Monaghan, J.J.: Extrapolating B splines for interpolation. J. Comput. Phys. **60** (1985) 253–262
16. Alexiades, V., Amiez, G., Gremaud, P.A.: Super-time-stepping acceleration of explicit schemes for parabolic problems. Comm. Numer. Meth. Eng. **12**(1) (1996) 31–42
17. Williamson, J.H.: Low-storage Runge-Kutta schemes. J. Comput. Phys. **35** (1980) 48–56
18. Shu, C.W., Osher, S.: Efficient implementation of essentially nonoscillatory shock-capturing schemes. J. Comput. Phys. **77**(2) (1988) 439–471
19. Cheng, H., Greengard, L., Rokhlin, V.: A fast adaptive multipole algorithm in three dimensions. J. Comput. Phys. **155** (1999) 468–498
20. Chaniotis, A.K., Poulikakos, D., Koumoutsakos, P.: Remeshed smoothed particle hydrodynamics for the simulation of viscous and heat conducting flows. J. Comput. Phys. **182**(1) (2002) 67–90
21. Sbalzarini, I.F., Mezzacasa, A., Helenius, A., Koumoutsakos, P.: Effects of organelle shape on fluorescence recovery after photobleaching. Biophys. J. **89**(3) (2005) 1482–1492
22. Sbalzarini, I.F., Hayer, A., Helenius, A., Koumoutsakos, P.: Simulations of (an)isotropic diffusion on curved biological surfaces. Biophys. J. **90**(3) (2006) 878–885
23. Werder, T., Walther, J.H., Jaffe, R.L., Halicioglu, T., P., K.: On the water–carbon interaction for use in molecular dynamics simulations of graphite and carbon nanotubes. J. Phys. Chem. B **107** (2003) 1345–1352
24. Springel, V., Yoshida, N., White, S.D.M.: GADGET: a code for collisionless and gasdynamical cosmological simulations. New Astron. **6** (2001) 79–117