Software challenges for Petaflop computing

Ivo F. Sbalzarini

Institute of Computational Science, ETH Zurich

Some of them...

- Programming languages and compilers (how much of the parallelism can be done on the language level?)
- Middleware (how much of the parallelism can be automated using libraries?)
- Ease of use (how do we enable a wider range of scientists to use HPC? In what form should they communicate to the machine?)
- Debugging and performance profiling tools
- Interactive simulations and on-line visualization
- Education

Language level

Vectoral: only by trying new things will we ever get computer languages right

Alan Wray
NASA Ames

- Ideas for language-level parallelism exist for a long time! (1978 Illiac IV)

But...

- We need a hierarchy of levels (language, loops, algorithms, numerical methods)
- Domain/Method-specific extensions are useful/needed
- A lot exists on the level of linear algebra (scaLAPACK, PETSc, ...)
- but what to do for other applications (agent-based models, particles, trees, ...)?

A middleware for the portable parallelization of particle-mesh simulations

Ivo F. Sbalzarini, J. H. Walther, B. Polasek, P. Chatelin, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, P. Koumoutsakos

Institute of Computational Science, ETH Zurich

Why middleware?

- **Goal:** make HPC systems easier to use and reduce code development time.
- Enable non-traditional HPC user fields (biology, social sciences, economics, psychology, ...).
- Re-usable code base: well tested, improvements immediately benefit all applications.
- Portable code across (inhomogeneous) architectures

Why hybrid particle-mesh?

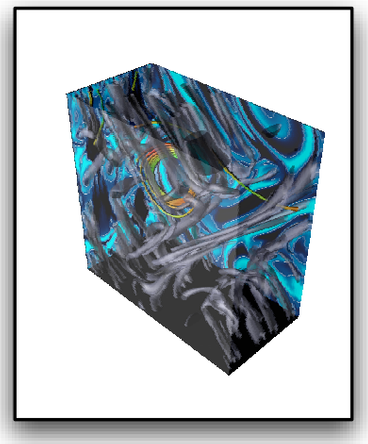
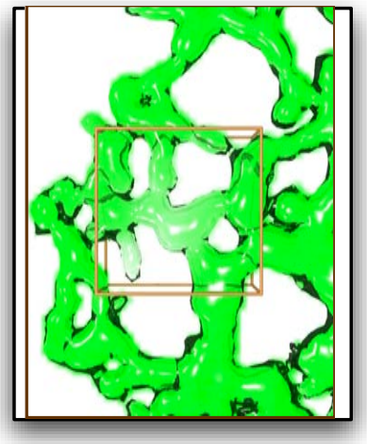
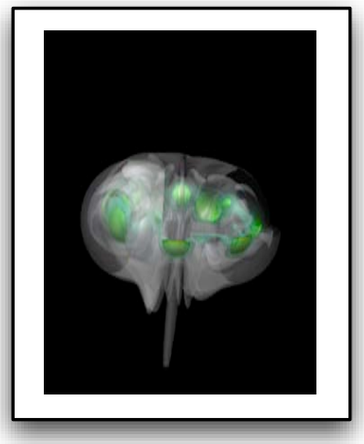
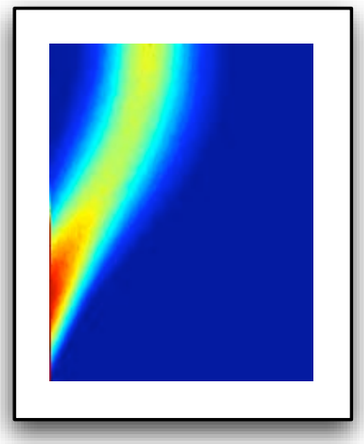
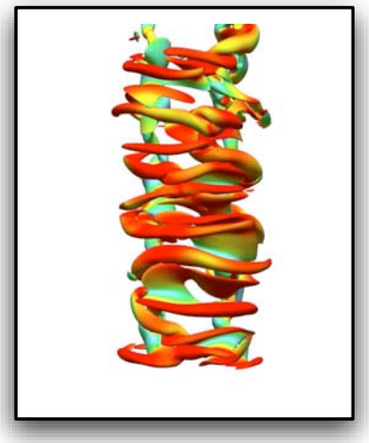
- Hybrid particle-mesh descriptions are common in many fields:
  - Plasma physics (charged particles, dynamics on mesh)
  - Molecular dynamics (fast electrostatics involve a mesh)
  - Astrophysics (planetary and gravitational systems)
  - Fluid mechanics (vortex methods and SPH)
  - Computational biology (reaction-diffusion simulations in complex geometries)
  - but also: Monte Carlo, Traffic Simulations, Optimal Control, Financial Mathematics, Granular Flows, DEM, ...

Design goals

- use of **symmetry** in particle-particle interactions
- ease of use and flexibility
- parallel scaling in CPU time **and memory**
- independence from specific applications / physics
- **adaptive** domain decompositions
- good **portability** across platforms
- good **vectorization** of all major loops
- re-usable, well tested code base

Parallel Particle Mesh Library (PPM)

Easy to use and efficient infrastructure for
particle-mesh simulations on parallel computers

Parallel Particle Mesh Library (PPM)

Message Passing Interface (MPI) | METIS | FFTW

vector | shared memory | distributed memory | single processor

The PPM Library

The PPM Library provides:

An **easy to use** and **efficient infrastructure** to implement particle-mesh simulations on parallel machines

Based on the **abstractions**:

- Topologies (data partition onto processors)
- Mappings (communication operators to move data between processors)
- Particles (location in space and attributed properties)
- Connections (links between particles, hard or soft)
- Meshes (hierarchies of Cartesian meshes)

Write simulation sequentially in terms of these primitives!

The PPM Library

The Library then automatically (**transparently**) performs:

Adaptive domain decomposition:

- Generate >>Nproc sub-domains with associated cost
- Build smaller sub-domains in high-density regions

Dynamic load balancing:

- Probe and monitor individual processor speeds
- Dynamically assign sub-domains to processors (particles AND mesh)
- Re-decomposition if amortized, predict re-decomp time points

Communication scheduling:

- Probe and monitor communication speeds (empirical topology)
- Build graph of application communication need
- Minimal edge coloring (+1 bound) to determine near-optimal schedule

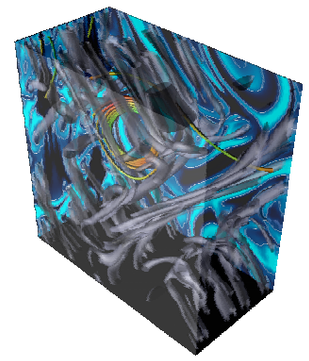
The PPM Library

The core has been *supplemented* with modules for:

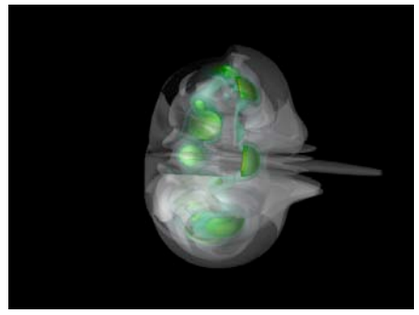
- Parallel Poisson solvers (multigrid and FFT)
- Parallel evaluation of differential operators on particles
- Parallel neighbor lists (cell lists and Verlet lists)
- Parallel multi-stage ODE integrators
- Parallel Fast Multipole Method (FMM) -- global distributed tree code!
- Boundary element solvers
- Parallel disk I/O
- Interfaces to external libraries (fftw, Hypre, HDF5, Metis)

Past applications of PPM

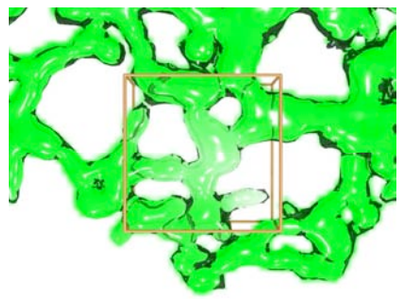
- **Vortex methods**
  for incompressible fluids
  268M particles, 512 processors, 76% efficiency



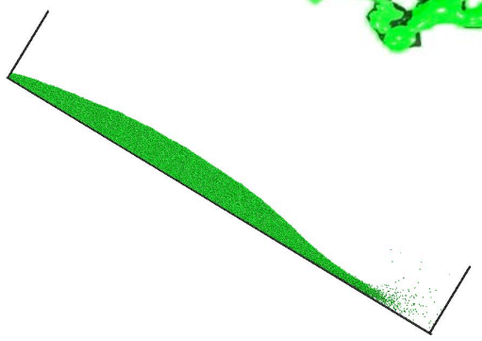
- **Smooth Particle Hydrodynamics**
  for compressible fluids
  268M particles, 128 processors, 91% efficiency



- **Particle diffusion**
  242 processors, 84%, up to 1 Billion particles



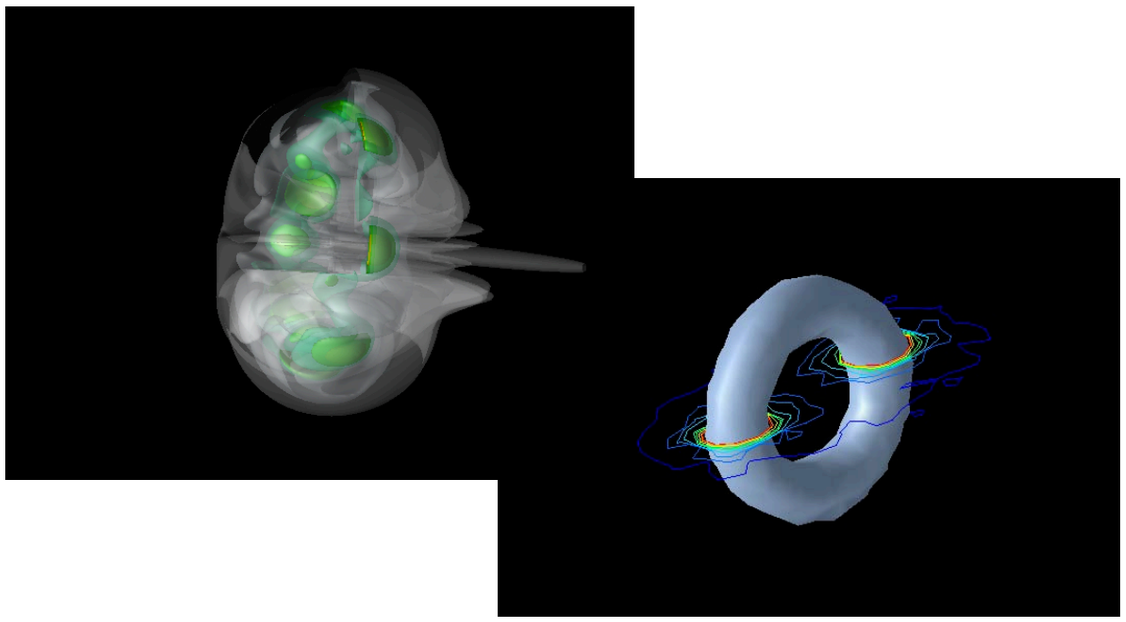
- **Discrete element methods**
  192 processors, 40%, up to 122M particles

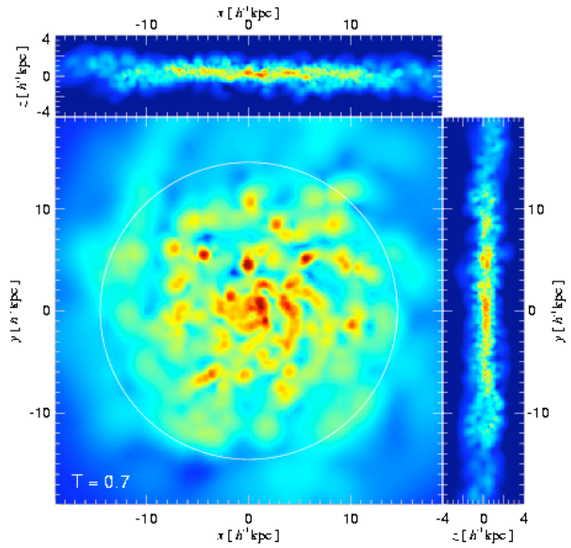


PPM-SPH

Remeshed Smoothed Particle Hydrodynamics method solving the 3D compressible Navier-Stokes equations.

Present	State of the art
• 1024×512^2 grid ($\sim 10^8$)	• Gadget (MPI Astrophysics)
• 128 CPUs (IBM p690, CSCS)	• 3082^3 grid ($\sim 10^{10}$)
• 91% Efficiency (on 128 CPU)	• 512 CPUs (IBM p690, MPI Garching)
	• 85% Efficiency (on 32 CPU)

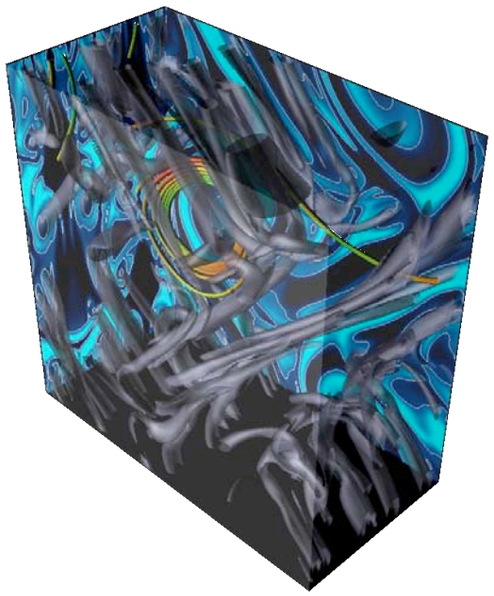


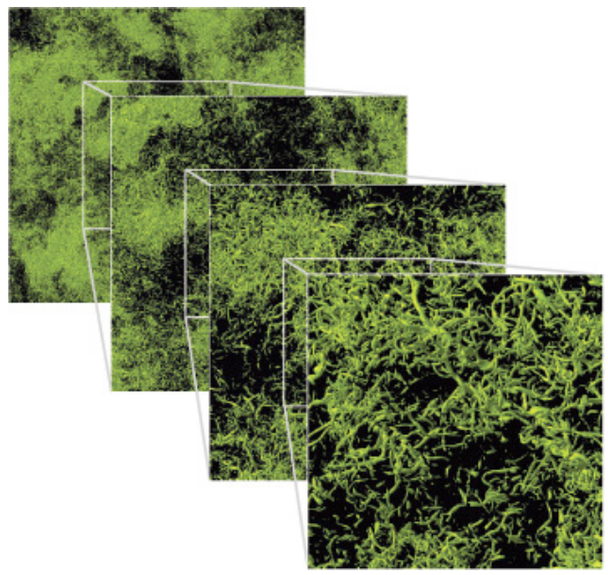


PPM-VM

Vortex-in-Cell particle method solving the 3D incompressible Navier-Stokes equations using multigrid as fast Poisson solver.

Present	State of the art
• double shear layer • $1024x512^2$ grid ($\sim 10^8$) • 512 CPUs (XT/3, CSCS) • 76% Efficiency	• Kuwahara Laboratory (Japan) • Isotropic turbulence • 4096^3 grid ($\sim 10^{10}$) • 512 CPUs (Earth Simulator) • 50% Efficiency



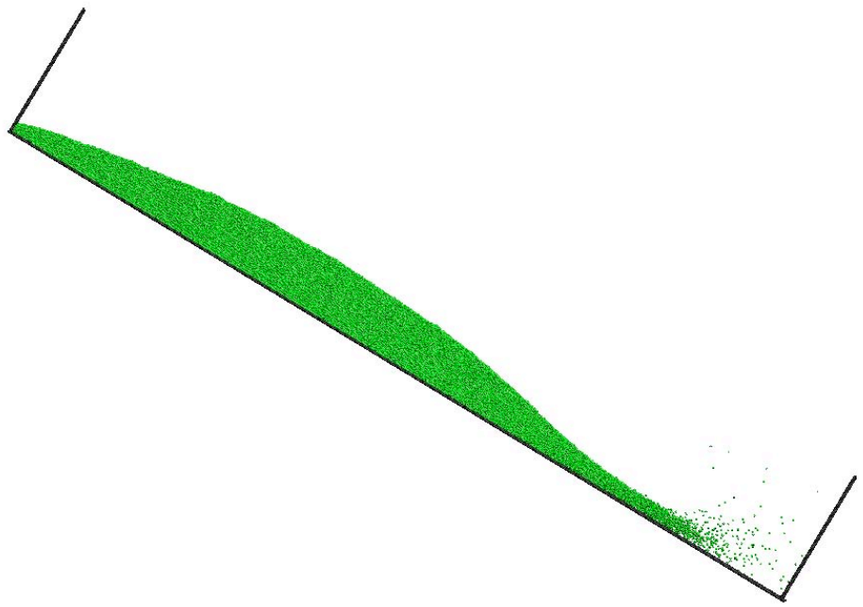


PPM-DEM

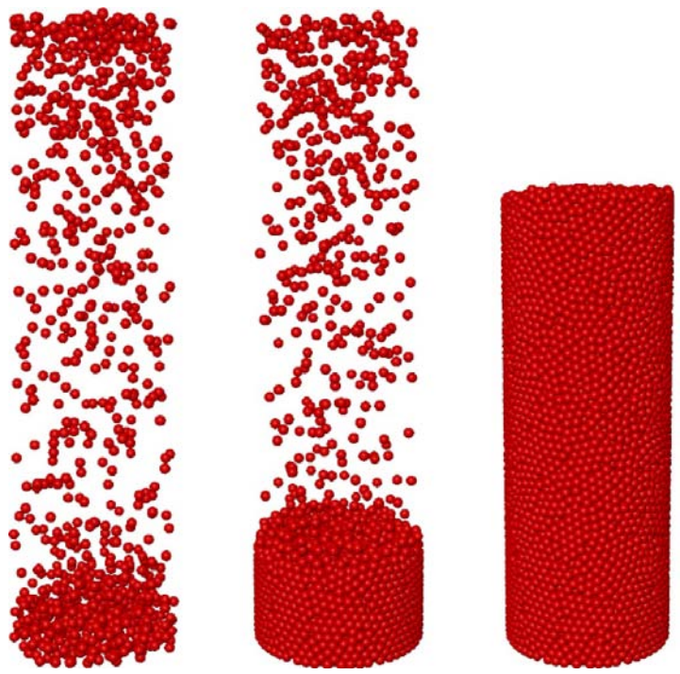
Discrete Element Method for granular flow with fully resolved visco-elastic colliding particles.

Present	State of the art

- sand avalanche
- 122M particles
- 192 CPUs (XT/3, CSCS)
- 40% Efficiency (strong)



- Landry et al. (SNL)
- same particle model (Silbert)
- 200k particle

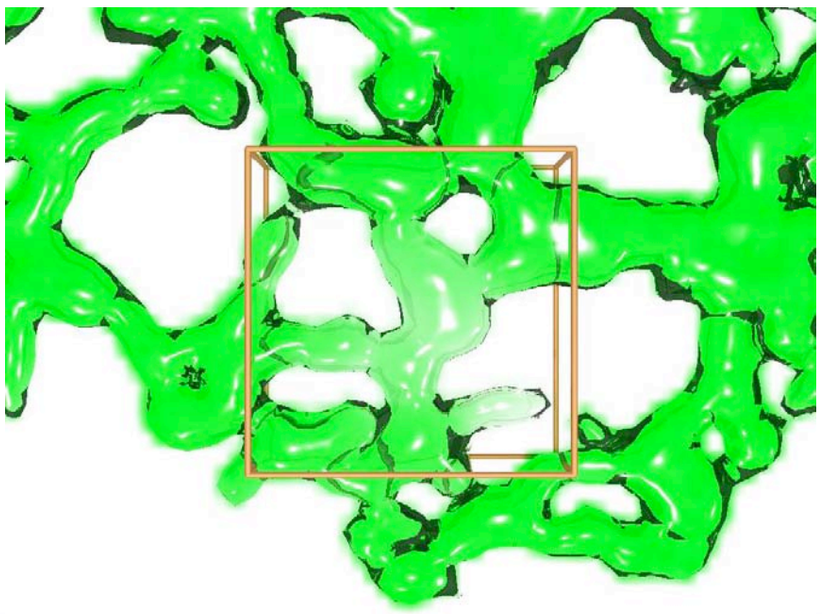


PPM-PSE

Particle Strength Exchange method solving the 3D diffusion equations in the geometry of a real cellular organelle (ER).

Present

- ER diffusion
- up to 1 billion particles
- 4…242 CPUs (IBM p690, CSCS)
- 84% Efficiency (on 242 CPU)



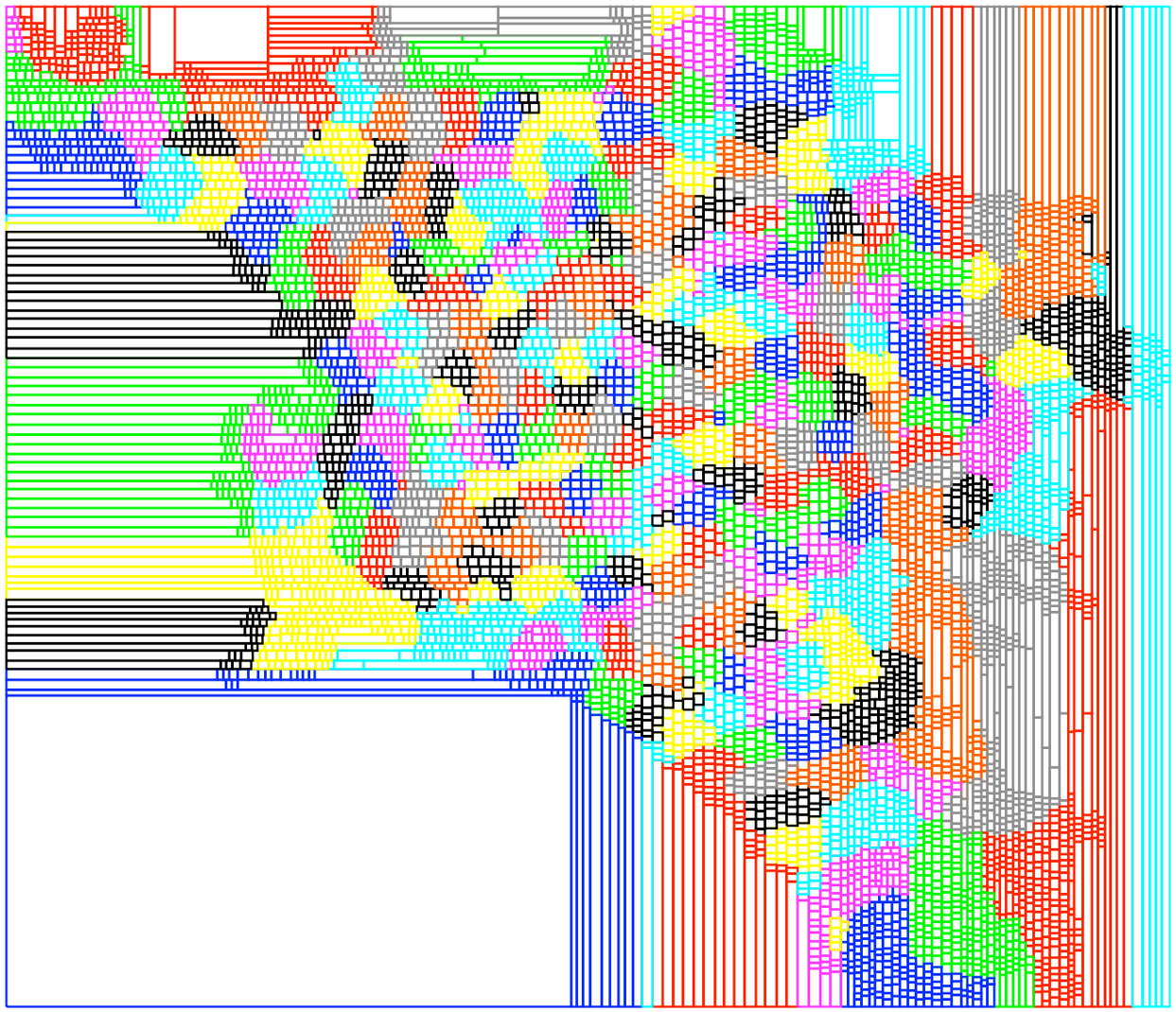
State of the art

- Winckelmans (Belgium)
- PSE for viscous Vortex Methods
- 2.4M particles (2002)
- 32 CPUs (HP Superdome)

- Sakaguchi et al. (Japan)
- Diffusion using random walk (2004)
- 6648 x 6648 x 6656 (294G)
- 4096 CPUs (NEX SX/5, Earth Sim.)

PPM-PSE Benchmark

Domain decomposition



Balanced decomposition on 242 processors

PPM-MD

Lennard-Jones Molecular Dynamics of Argon

Present

- Argon gas
- 8 million atoms
- 256 CPUs (XT/3, CSCS)
- 63% Efficiency
- 0.25 seconds/timestep

State of the art

- Dedicated MD package FASTTUBE: 2-fold slower than ppm-md. Took 6 years to develop, ppm-md 3 months.

Werder et al., ETH Zurich

Acknowledgements

PPM core development team:

Prof. Dr. Jens Walther
Prof. Dr. Ivo Sbalzarini
Dr. Philippe Chatelin
Michael Bergdorf
Evangelos Kotsalis
Simone Hieber

More information (reference manual) and PPM source:

http://minion.inf.ethz.ch/ppm

I. F. Sbalzarini, J. H. Walther, M. Bergdorf, S. E. Hieber, E. M. Kotsalis, and P. Koumoutsakos. PPM – a highly efficient parallel particle-mesh library for the simulation of continuum systems. *J. Comput. Phys.*, 215(2):566–588, 2006.