Highly scalable asynchronous parallelization of the Tree Code PEPC
n-body problem

- long-range interactions, i.e. unlimited range
- force depends on distances

- Coulomb / gravity potential equation

\[ \phi(\vec{r}_1, \vec{r}_2) \sim \frac{1}{|\vec{r}_1 - \vec{r}_2|} \]

- evaluation involves computing roots and reciprocal
- initial computational complexity \( O(n^2) \)
Scientific applications

Laser-cluster interaction

Planet creation

Vortex dynamics

Molecular dynamics
Complexity reduction

- Direct summation \(O(n^2)\)
- Tree code \(O(n \log n)\)
- Fast multipole method \(O(n)\)

- good load balancing
- kernel flexibility
- simple parallelization

- minimize FP operations
- more logistics
- scalability?
PEPC – Pretty Efficient Parallel Coulomb solver

Frontend Applications
- pepc-mini Demo Application
- pepc-b Laser-Plasma Int.
- pepc-s ScaFaCoS-Library
- pepc-v Vortex Dynamics
- pepc-g Stellar Disc Evol.

Tree-Code Algorithm
- Tree Data Structures
- Parallel Tree Buildup
- Parallel Tree Traversal
- Load Balancing

Utils:
- VTK-IO, Diagnostics, Benchmarking, ...

Interaction-Specific Modules
- reg. Coulomb
- reg. Algebraic Kernels
- Neighbour-search/SPH + Gravitation

17.05.2012, ScicomP 2012, Lukas Arnold
Tree Code (2D)

physical particle positions

space filling curves

tree construction
Do the interaction

- interact only with particle cluster, if possible
- multipole acceptance criterion (MAC), decides if multipole expansion is good enough
- Barnes-Hut MAC
- work is not equal for particles
Parallel Tree Code

- hash table
- multipole computation
- global communication of branch nodes (A2A)
Communication pattern

- nearly-all-2-nearly-all communication
- pattern is problem dependent

17.05.2012, ScicomP 2012, Lukas Arnold
Scalability of the pure MPI version

- run fine on up to 8k cores BG/P
- main issue: load balancing + collective operations
Trace of pure MPI version

- tree traversal and force calculation separated
- global load balancing: perfect, on chunk level: not so impressive
Why have a hybrid parallelization?

- **scalability**
  - if second level parallelization is perfect: push MPI scalability
  - some parts of the algorithm do not scale, inverse scaling

- **memory**
  - only one hash table per node needed
  - memory consumption per process grows with its number

- **communication**
  - less volume: more process local particles
  - hiding: potential for overlapping
General asynchronous idea

- why not issue communication when needed?
- why not overlap data exchange with computation?

- usage of PThreads + MPI gives a portable and flexible base
- main concept
  - one communicator thread and multiple worker
  - workers issue communication, communicator does the MPI
PThreads + MPI parallelization

17.05.2012, ScicomP 2012, Lukas Arnold
Hybrid scalability

- better scaling then the pure MPI version
- bottle necks: collective operations in the tree setup
New scalability bottle necks

- main phase (traversal) fine, now other parts pop up
- tree setup does not scale: neither in memory, nor in run time
- only algorithmic changes will help, aim: no collective operations
- works as designed
- the workload of communication thread may become a bottle neck
Load balancing

- number of interactions counts, not particle number
- total run time reduces significantly
- load balancing based on work load of last iteration
Comparing BG/P to BG/Q

- performance ratio BG/P to BG/Q ~ 8x on node level
- huge benefit of SMT (30%), no usage of Double/Quad-Hummer

17.05.2012, ScicomP 2012, Lukas Arnold
Task oriented implementation

- using BSC’s task oriented run time SMPSs (now: OMPSs)
- setup whole/main execution as a dependency graph
- easy setup of tasks (via pragmas)
- asynchronism comes for free (MPI, IO, ...)

- downside: SMPSs development status is still experimental
ScaFaCoS library

- ScaFaCoS = Scalable Fast Coulomb Solvers
- collection of 7 solvers for long-range interactions (+near field)
- same interface for all methods (easy method switching)
- grid based and mesh-free methods
- periodic and open systems
- current target architecture: BG/P

- open source, GPL/LGPL license
- first “official” release this summer
Conclusions

- PEPC – Pretty Efficient Parallel Coulomb solver
- General idea of a tree code
- Scalability of pure MPI version and the hybrid version

- PEPC benefits from an asynchronous execution
- We gain a factor of 8x on BG/Q architecture

- PEPC is freely available at www.fz-juelich.de/ias/jsc/pepc