JÜLICH
FORSCHUNGSZENTRUM

Development of a parallel, tree-based neighbour-search algorithm

for the tree-code PEPC

28.09.2010 | Andreas Breslau

Outline

1. Motivation
2. Short introduction to tree-codes
3. The tree-based neighbour search
4. Validation
5. Benchmarking
6. Summary and Outlook

Outline

1. Motivation
2. Short introduction to tree-codes
3. The tree-based neighbour search
4. Validation
5. Benchmarking
6. Summary and Outlook

Motivation

for simulations

- In astrophysics the usual time-scales are much bigger than in other fields of physics

Motivation

for simulations

- In astrophysics the usual time-scales are much bigger than in other fields of physics
  - Formation of a star $\approx$ 10 mio years

Motivation

for simulations

- In astrophysics the usual time-scales are much bigger than in other fields of physics
  - Formation of a star ≈ 10 mio years
  - Revolution of the sun around the center of the Milky Way ≈ 200 mio years

Motivation

for simulations

- In astrophysics the usual time-scales are much bigger than in other fields of physics
  - Formation of a star $\approx$ 10 mio years
  - Revolution of the sun around the center of the Milky Way $\approx$ 200 mio years
  - dynamical timescale of superclusters $\approx$ few billion years

Motivation

for simulations

- In astrophysics the usual time-scales are much bigger than in other fields of physics
    - Formation of a star $\approx$ 10 mio years
    - Revolution of the sun around the center of the Milky Way $\approx$ 200 mio years
    - dynamical timescale of superclusters $\approx$ few billion years
- Objects of interest (stars, starclusters, galaxies) are very big

Motivation

for simulations

- In astrophysics the usual time-scales are much bigger than in other fields of physics
  - Formation of a star $\approx$ 10 mio years
  - Revolution of the sun around the center of the Milky Way $\approx$ 200 mio years
  - dynamical timescale of superclusters $\approx$ few billion years
- Objects of interest (stars, starclusters, galaxies) are very big
- that makes laboratory experiments difficult

Motivation

for simulations

Apart from observation, simulations are the only way to test theories

Motivation

for neighbour search

- Often huge amounts of self-gravitating matter are simulated

Motivation

for neighbour search

- Often huge amounts of self-gravitating matter are simulated
- e.g. starclusters can be simulated as n bodies (only attracting forces)

[NGC 3603 from hubblesite.org]

Motivation

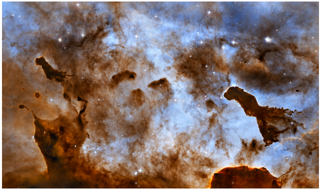
for neighbour search

- Often huge amounts of self-gravitating matter are simulated
- e.g. starclusters can be simulated as n bodies (only attracting forces)
- gravitation only first approximation (radiation, magnetism)

Motivation

for neighbour search

- Often huge amounts of self-gravitating matter are simulated
- e.g. starclusters can be simulated as n bodies (only attracting forces)
- gravitation only first approximation (radiation, magnetism)
- repulsing force from pressure-gradient for gas



[Carina Nebula from hubblesite.org]

Motivation

for neighbour search

To simulate self-gravitating gas:

- ⇒ simulate gravitational force as usual

Motivation

for neighbour search

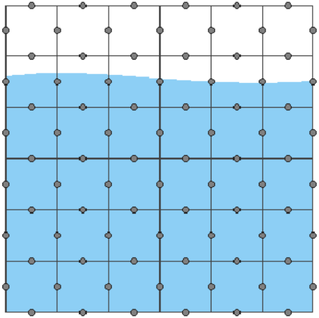
To simulate self-gravitating gas:

- ⇒ simulate gravitational force as usual
- ⇒ add thermodynamic forces from fluid simulation

Motivation

for neighbour search

- fluid-codes based on a fixed mesh would waste resources computing empty regions

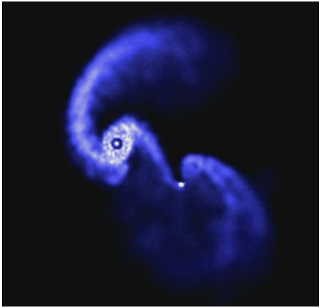


[Kelager, M., 2006]

Motivation

for neighbour search

- fluid-codes based on a fixed mesh would waste resources computing empty regions
- often matter is highly clustered within the simulation box

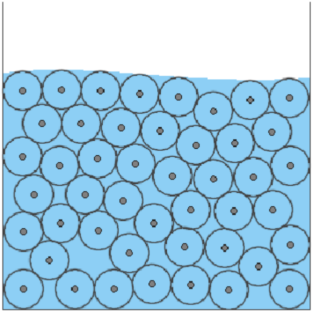


[http://www.astro.uni-koeln.de/movies]

Motivation

for neighbour search

- fluid-codes based on a fixed mesh would waste resources computing empty regions
- often matter is highly clustered within the simulation box
- ⇒ use Smoothed Particle Hydrodynamics (SPH)



[Kelager, M., 2006]

Motivation

for neighbour search

- mesh-based fluid codes compute thermodynamic properties (temperature, density, pressure) locally using input from neighboring cells

Motivation

for neighbour search

- mesh-based fluid codes compute thermodynamic properties (temperature, density, pressure) locally using input from neighboring cells
- In SPH, fluid properties are computed from averages over neighboring particles

Motivation

for neighbour search

- mesh-based fluid codes compute thermodynamic properties (temperature, density, pressure) locally using input from neighboring cells
- In SPH, fluid properties are computed from averages over neighboring particles
- need to know the next neighbours of a particle

Outline

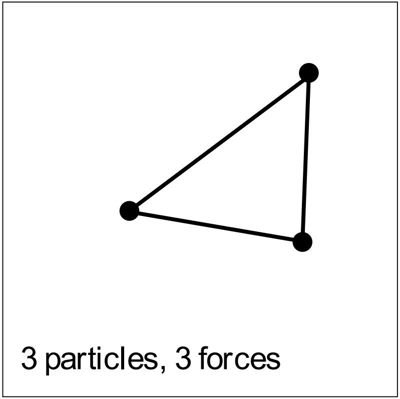
1. Motivation
2. Short introduction to tree-codes
3. The tree-based neighbour search
4. Validation
5. Benchmarking
6. Summary and Outlook

The n-body problem

- n bodies interacting with each other ($n(n-1)$ interactions)

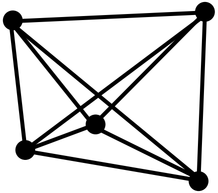
The n-body problem

- n bodies interacting with each other ($n(n-1)$ interactions)



The n-body problem

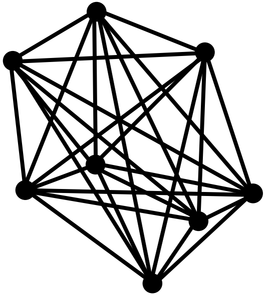
- n bodies interacting with each other ($n(n-1)$ interactions)



5 particles, 20 forces

The n-body problem

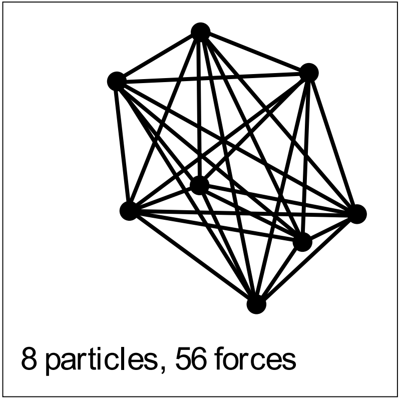
- n bodies interacting with each other ($n(n-1)$ interactions)



8 particles, 56 forces

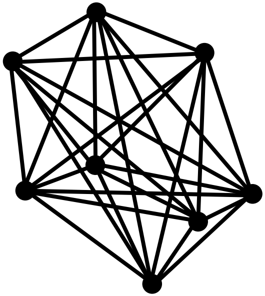
The n-body problem

- n bodies interacting with each other ($n(n-1)$ interactions)
- → runtime $O(n^2)$



The n-body problem

- n bodies interacting with each other ($n(n-1)$ interactions)
- $\rightarrow$ runtime $O(n^2)$
- bad computation time for big systems



8 particles, 56 forces

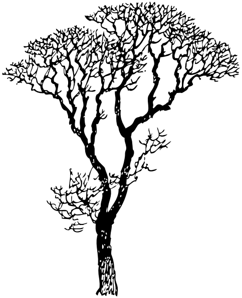
A better solution

A better solution

Use a tree

A better solution

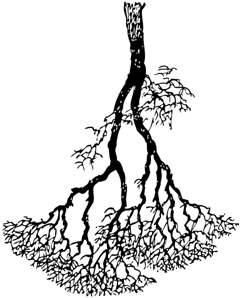
Use a tree



[openclipart.org]

A better solution

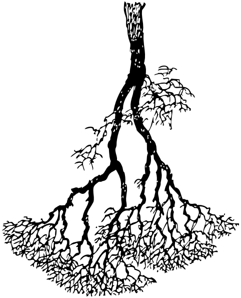
Use a tree



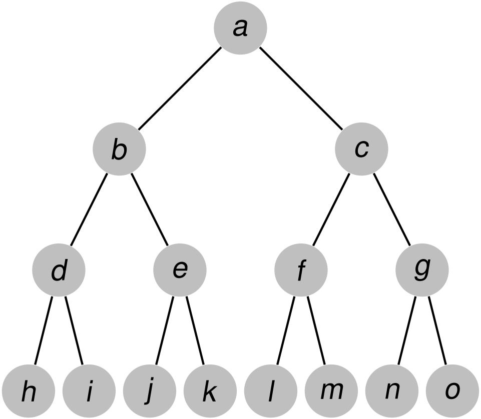
[openclipart.org]

A better solution

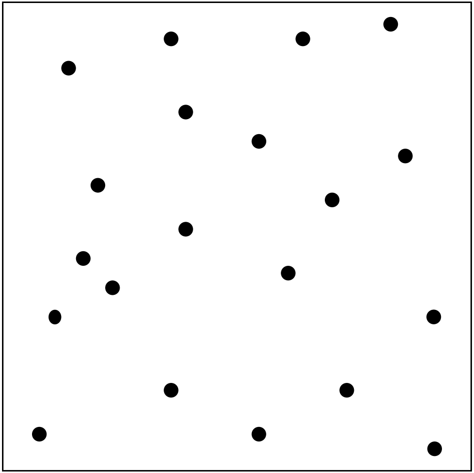
Use a tree



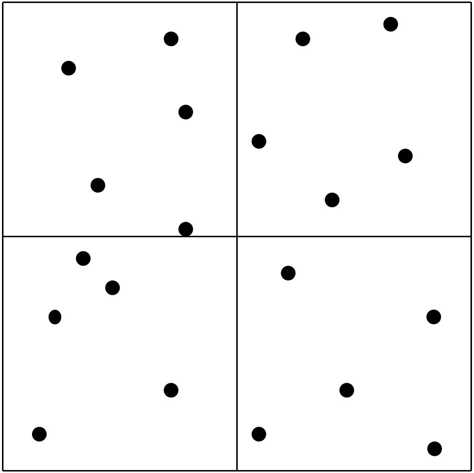
[openclipart.org]



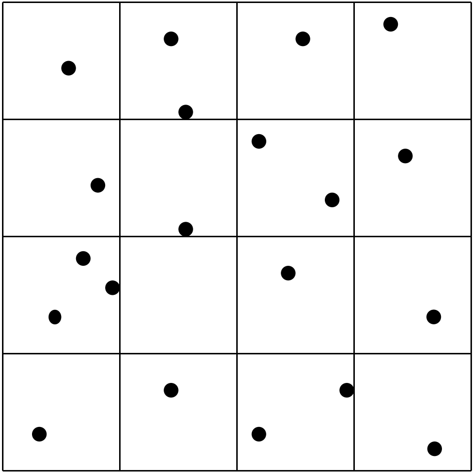
Using a tree



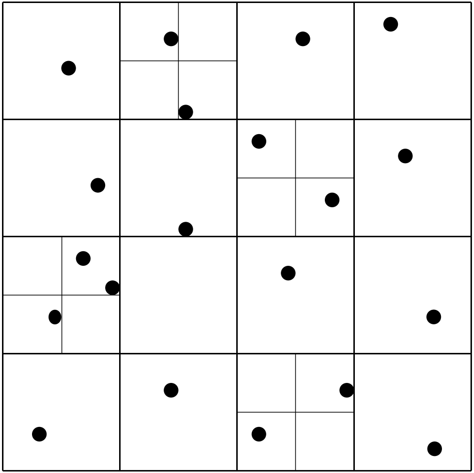
Using a tree



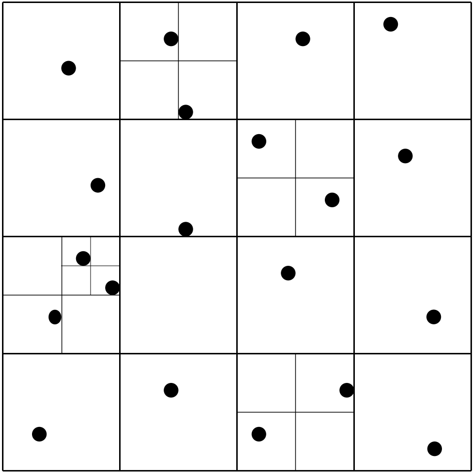
Using a tree



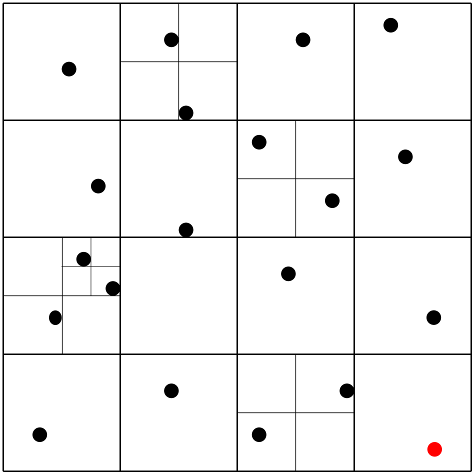
Using a tree



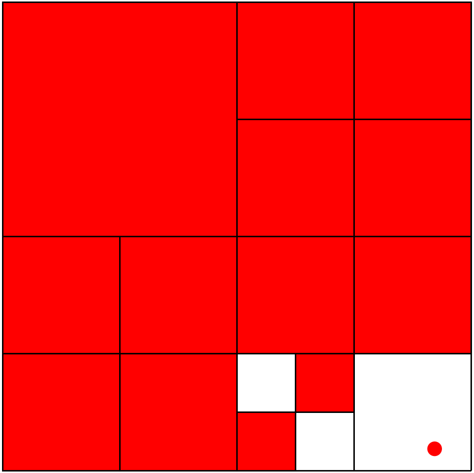
Using a tree



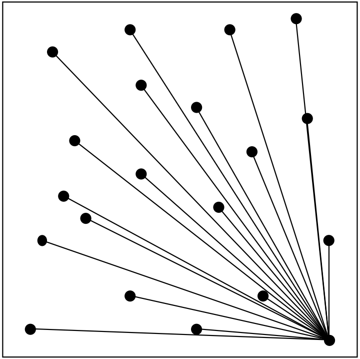
Using a tree



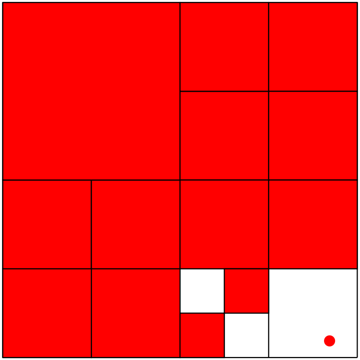
Using a tree



Direct summation vs. tree-code



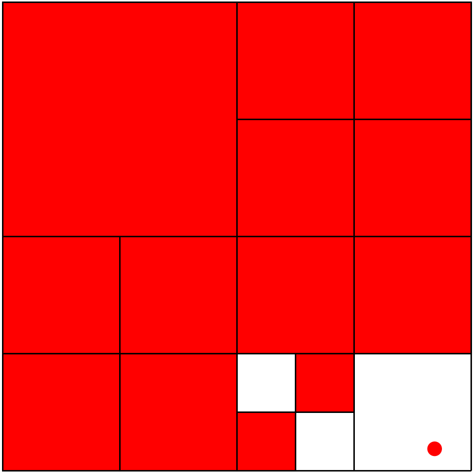
direct: 19



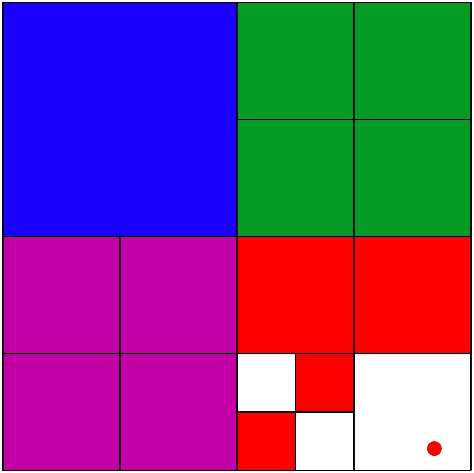
tree: 13

Parallelization of tree-codes

Parallelization of tree-codes



Parallelization of tree-codes



Outline

1. Motivation
2. Short introduction to tree-codes
3. The tree-based neighbour search
4. Validation
5. Benchmarking
6. Summary and Outlook

The search algorithm

- the search algorithm was integrated in the tree-code PEPC written by Dr. Paul Gibbon in 2003

The search algorithm

- the search algorithm was integrated in the tree-code PEPC written by Dr. Paul Gibbon in 2003
- PEPC is a tree-code following the tradition started in 1986 by Barnes and Hut

The search algorithm

- the search algorithm was integrated in the tree-code PEPC written by Dr. Paul Gibbon in 2003
- PEPC is a tree-code following the tradition started in 1986 by Barnes and Hut
- It uses a Hashed Oct-tree as described by Warren and Salmon (1993)

The search algorithm

- the search algorithm was integrated in the tree-code PEPC written by Dr. Paul Gibbon in 2003
- PEPC is a tree-code following the tradition started in 1986 by Barnes and Hut
- It uses a Hashed Oct-tree as described by Warren and Salmon (1993)
- the search algorithm also follows an idea of Warren and Salmon

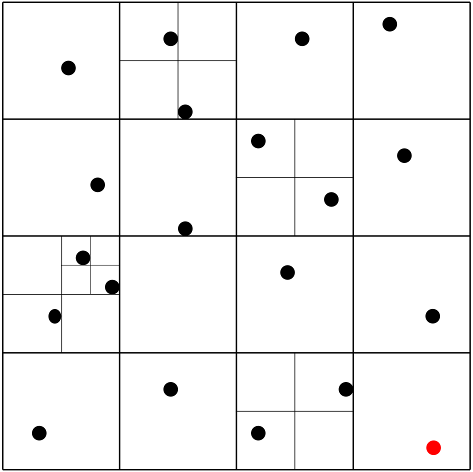
The search algorithm

```
while there are particles with less than N_nn
     found next neighbours
    search_neighbours_of_particle_i(r_i)

    if found next neighbours < N_nn
        increase r_i for this particle
        put particle on list to search
            neighbours again
    end
end
```

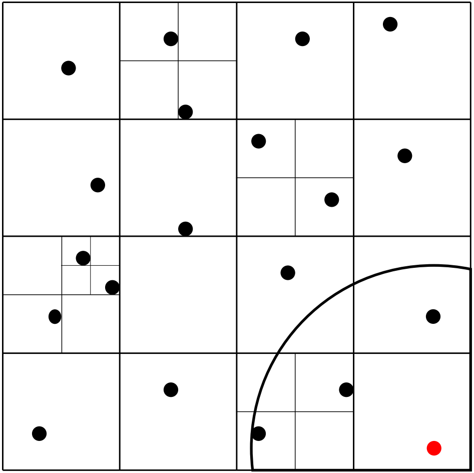
The search algorithm

```
search_neighbours_of_particle_i(r_i) {

    walk through tree from root to leaves
        1) particles within r_i put on next
           neighbour list
        2) ignore nodes/particles outside r_i
        3) resolve nodes with overlap with r_i
    end
}
```

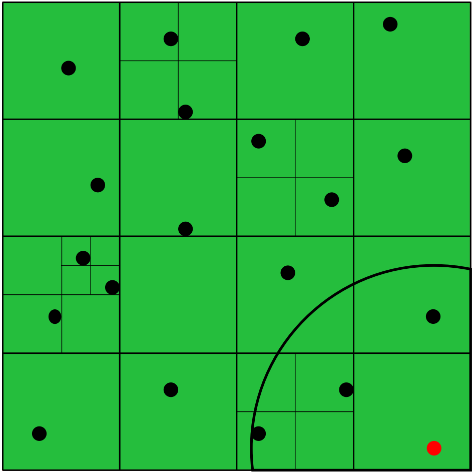
Tree based neighbour search



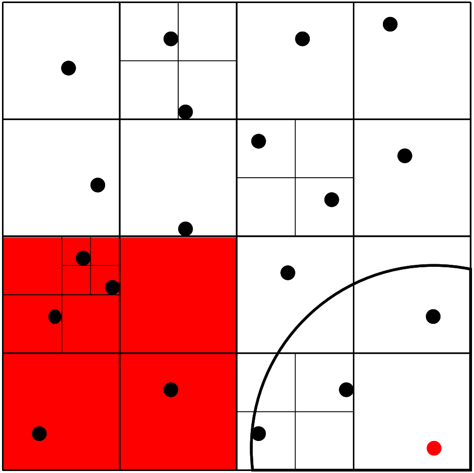
Tree based neighbour search



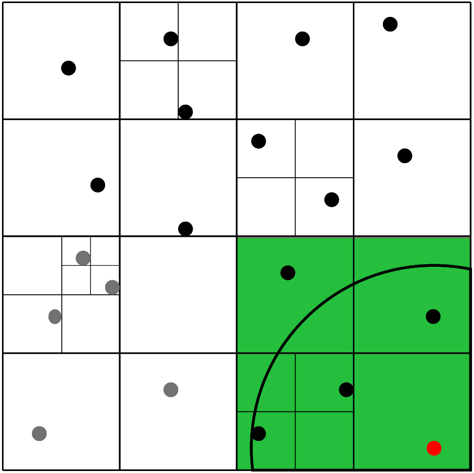
Tree based neighbour search



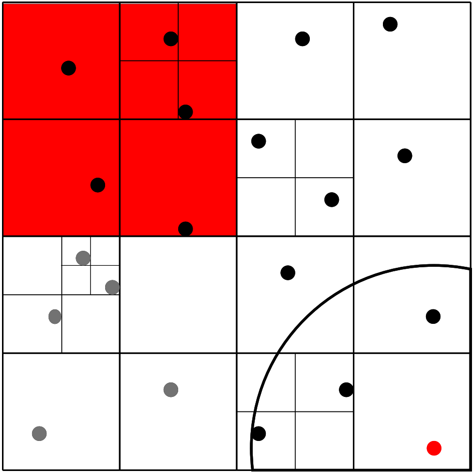
Tree based neighbour search



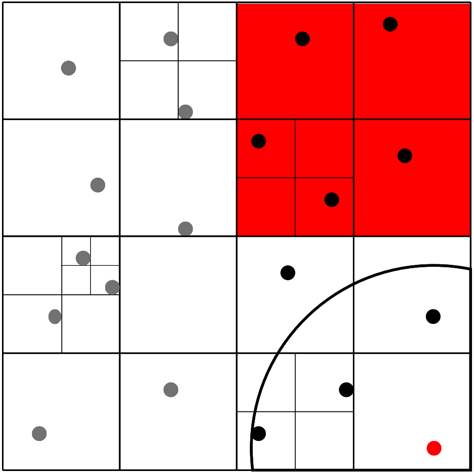
Tree based neighbour search



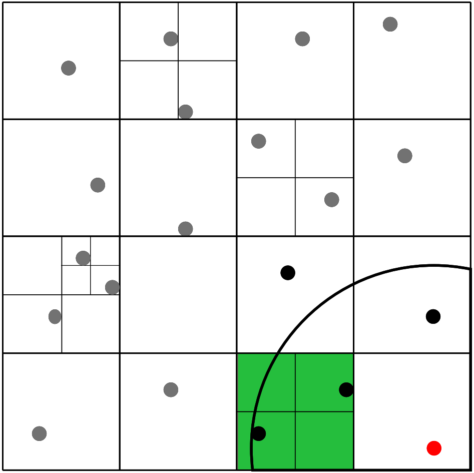
Tree based neighbour search



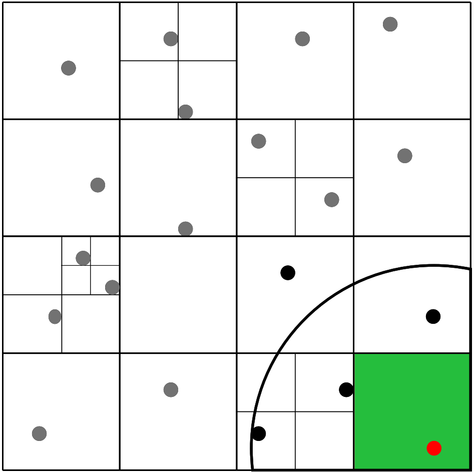
Tree based neighbour search



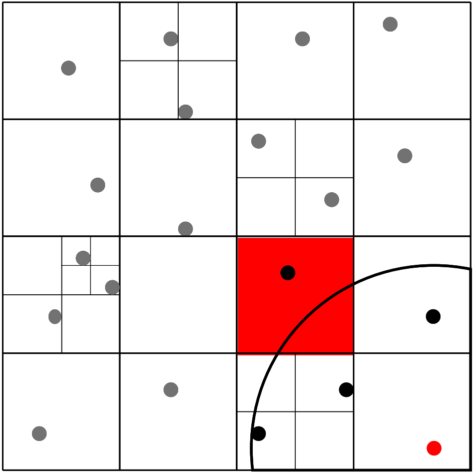
Tree based neighbour search



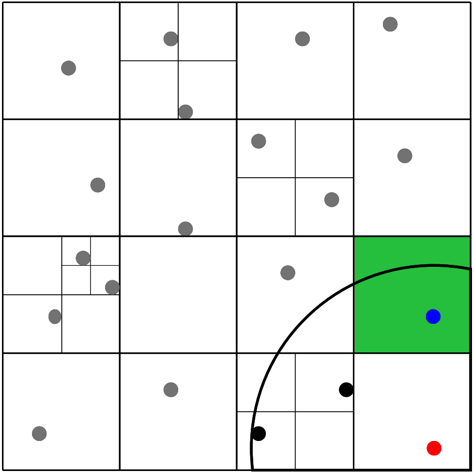
Tree based neighbour search



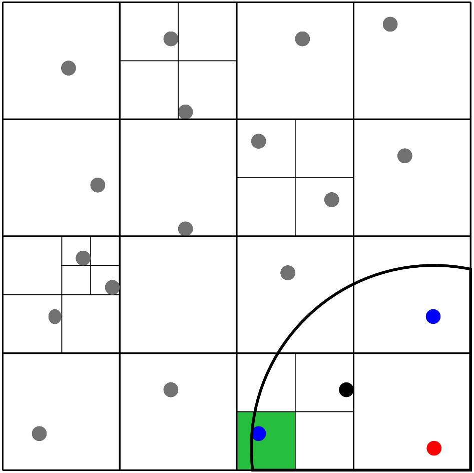
Tree based neighbour search



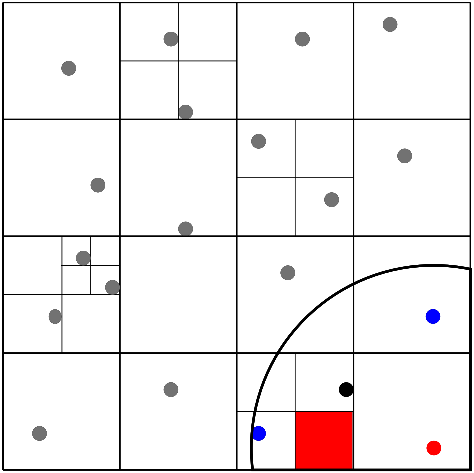
Tree based neighbour search



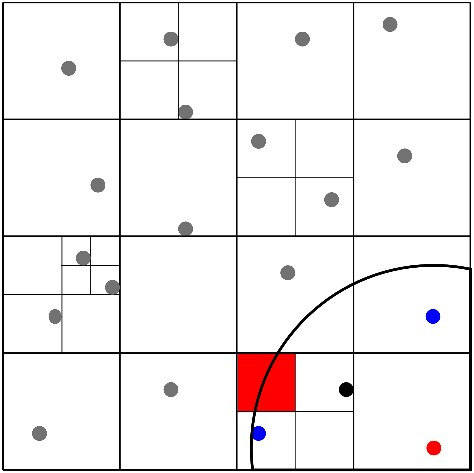
Tree based neighbour search



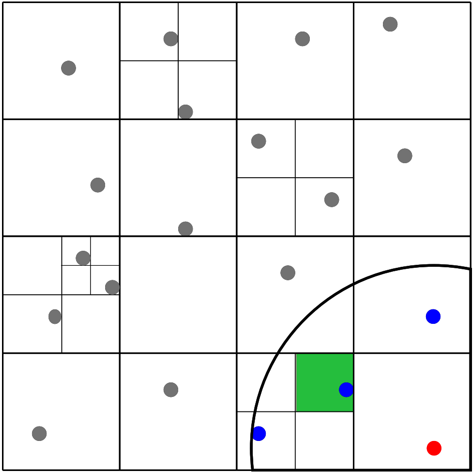
Tree based neighbour search



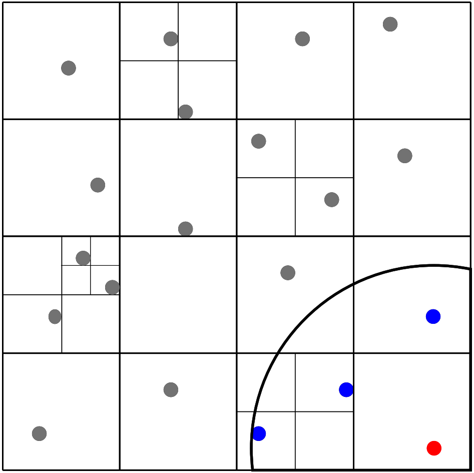
Tree based neighbour search



Tree based neighbour search



Tree based neighbour search



Outline

1. Motivation
2. Short introduction to tree-codes
3. The tree-based neighbour search
4. Validation
5. Benchmarking
6. Summary and Outlook

Validation

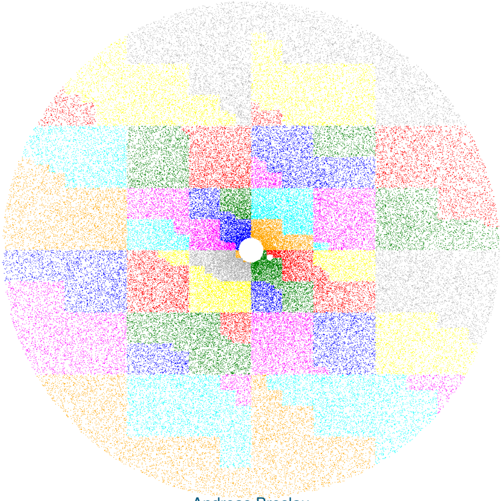
- check found neighbours manually with plots

Validation

- check found neighbours manually with plots
- write validation tool

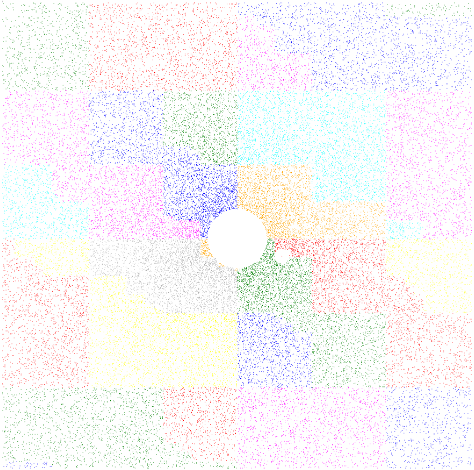
Validation

with plots



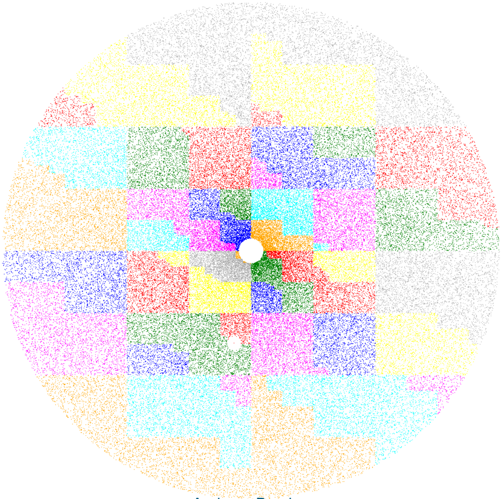
Validation

with plots



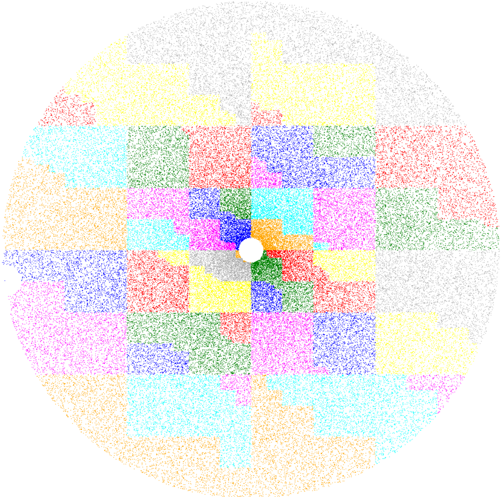
Validation

with plots



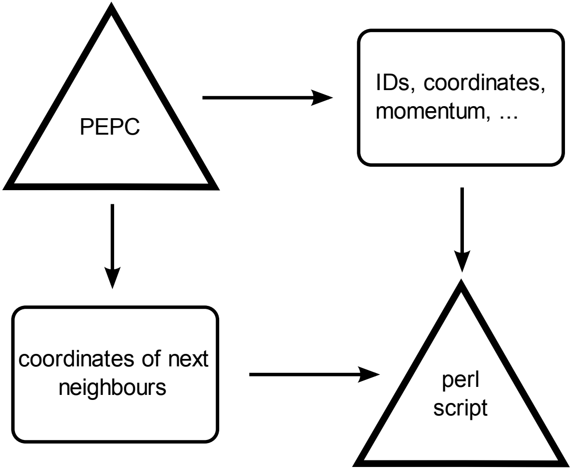
Validation

with plots



Validation

with validation tool



PEPC

IDs, coordinates, momentum, ...

coordinates of next neighbours

perl script

Validation

Validation

- manual checking plots proofed that the algorithm works correct for 2D

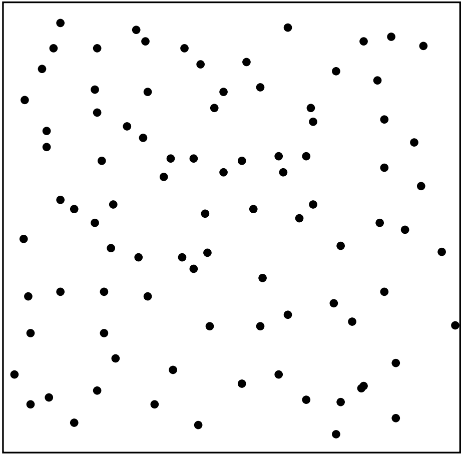
Validation

- manual checking plots proofed that the algorithm works correct for 2D
- the validation tool proofed that it works correct for 3D

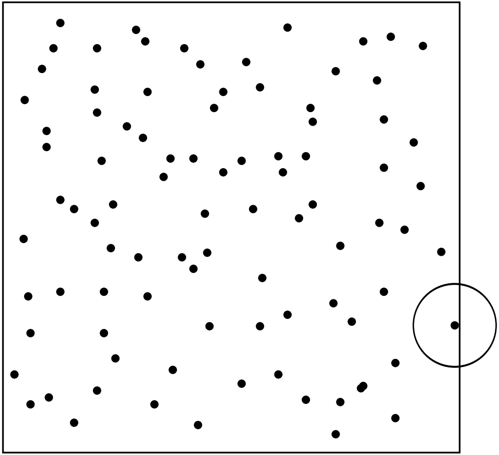
Outline

1. Motivation
2. Short introduction to tree-codes
3. The tree-based neighbour search
4. Validation
5. Benchmarking
6. Summary and Outlook

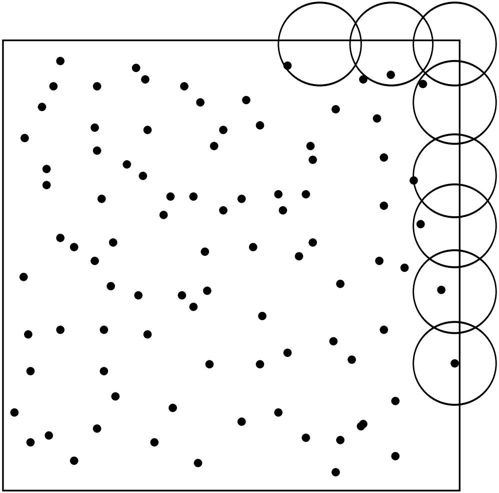
An estimation



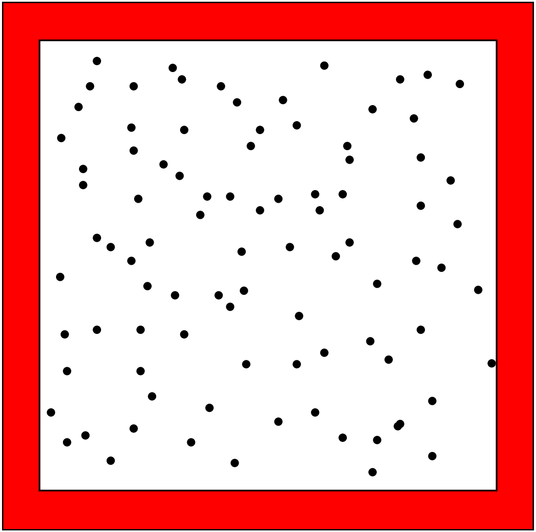
An estimation



An estimation



An estimation



An estimation

for the network and memory usage

N, N_{nn}, p

$$\rho = \frac{N}{V}, \quad N_{nn} = \frac{4}{3}\pi r_{search}^3 \rho, \quad \frac{V}{p} = \frac{4}{3}\pi R_{domain}^3$$

$$\begin{aligned} N_{fetch} &= \frac{\frac{4}{3}\pi\rho\left[(R+r)^3 - R^3\right]}{} \\ &= \sqrt[3]{27\frac{N_{nn}N^2}{p^2} + 27\frac{N_{nn}^2 N}{p} + N_{nn}^3} \end{aligned}$$

$\Rightarrow O(N_{nn}), O(N^{2/3}), O(p^{-2/3})$

An estimation

for the network and memory usage

N_p, N_{nn}, p

$$N_{fetch} = \sqrt[3]{27 N_{nn} N_p^2 + 27 N_{nn}^2 N_p + N_{nn}^3}$$

$$\Rightarrow O(N_{nn}), O(N_p^{2/3})$$

An estimation

for the network and memory usage

N_{nn}	N_p	N_{fetch}	$N_{fetch}[\%]$
50	10000	6000	61
50	50000	17000	33
50	200000	40000	20

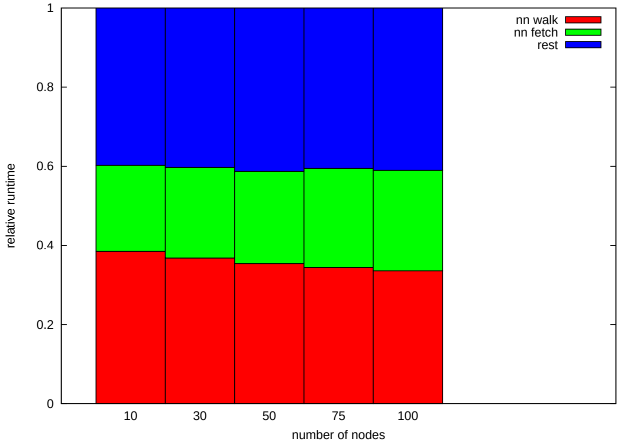
Juropa

- 2208 compute nodes
  - Compute node: 2 Intel Xeon quad-core processors at 2.93 GHz
  - Total cores: 17664
- Overall peak performance: 207 Teraflops
- Main memory: 24 GB per node / 51.75 TB total
- Networks: Infiniband Fat Tree

[http://www.fz-juelich.de/jsc/juropa/]

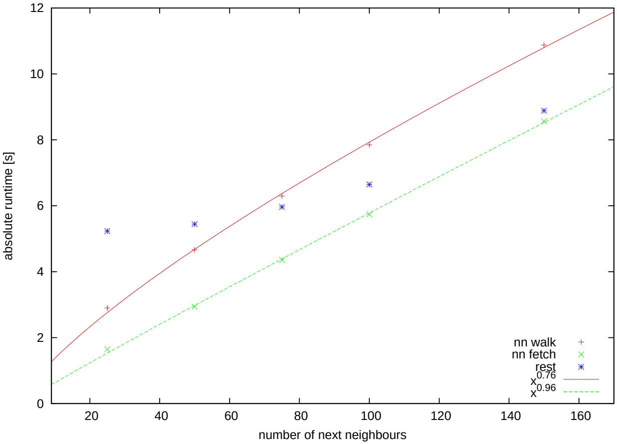
Weak scaling on Juropa

50 next neighbours, 150000 particles per process, relative



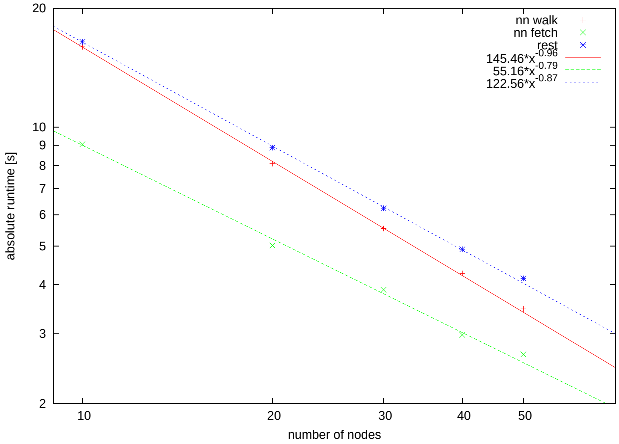
N_{nn} scaling on Juropa

10 nodes, 50000 particles per process, absolute



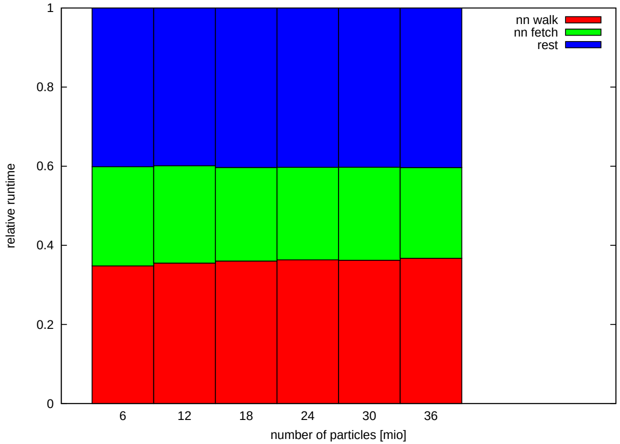
Strong scaling on Juropa

50 next neighbours, 12 mio particles, absolute, logscale



N scaling on Juropa

50 next neighbours, 30 nodes, relative



Outline

1. Motivation
2. Short introduction to tree-codes
3. The tree-based neighbour search
4. Validation
5. Benchmarking
6. Summary and Outlook

Summary

Summary

- Parallel tree-based neighbour-search successfully implemented

Summary

- Parallel tree-based neighbour-search successfully implemented
- Validation tool implemented (further versions can easily be tested)

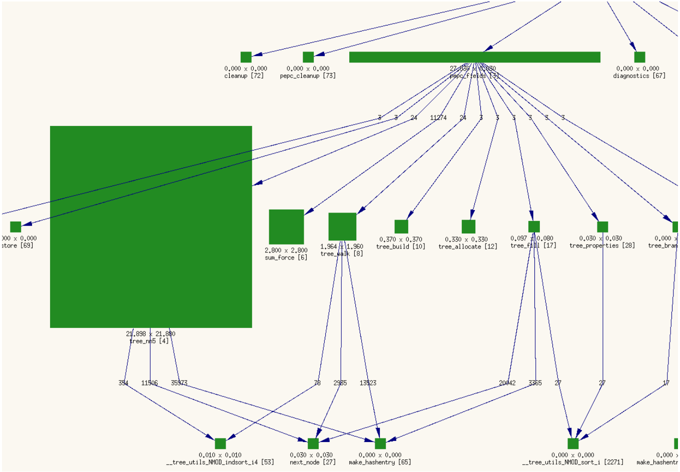
Summary

- Parallel tree-based neighbour-search successfully implemented
- Validation tool implemented (further versions can easily be tested)
- Weak and strong scaling at least as good as gravitational force computation

Summary

- Parallel tree-based neighbour-search successfully implemented
- Validation tool implemented (further versions can easily be tested)
- Weak and strong scaling at least as good as gravitational force computation
- Overhead currently $\approx 60\%$ total iteration time, but ...

Optimization needed



[Xprofiler screen-shot]

Outlook

- find out, what uses so much time

Outlook

- find out, what uses so much time
- domain decomposition balanced with nn_{search} work load

Outlook

- find out, what uses so much time
- domain decomposition balanced with nn_{search} work load
- compute nn lists only every t timesteps

Outlook

- find out, what uses so much time
- domain decomposition balanced with nn_{search} work load
- compute nn lists only every t timesteps
- neighbour search for symmetric SPH

Thank you for your attention.

Any Questions?

References

- Barnes, J., Hut, P., 1986: A hierarchical O(N log N) force-calculation algorithm
- Gibbon, P., et. al., 2010: Progress in Mesh-Free Plasma Simulation With Parallel Tree Codes
- Kelager, M., 2006: Lagrangian Fluid Dynamics Using Smoothed Particle Hydrodynamics
- Warren, M. S., Salmon, J. K., 1993: A parallel hashed oct-tree n-body algorithm
- Warren, M. S., Salmon, J. K., 1995: A portable parallel particle program

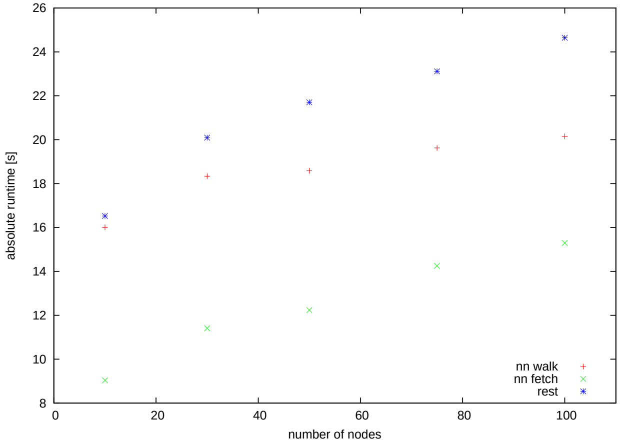
Some images used in this talk are intellectual property of other authors and may not be distributed or reused without their explicit approval.

Appendix

- More scaling plots
- About the speaker

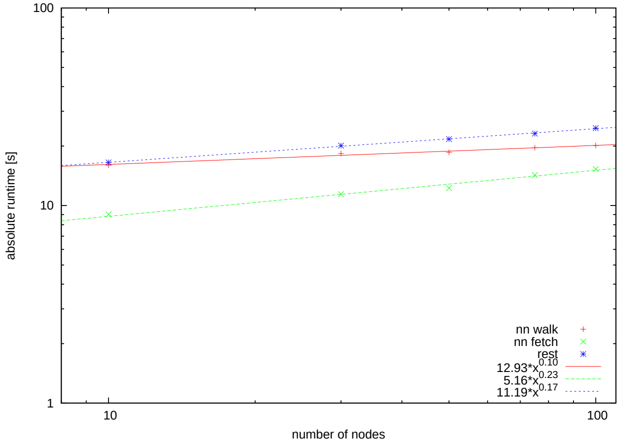
Weak scaling on Juropa

50 next neighbours, 150000 particles per process, absolute



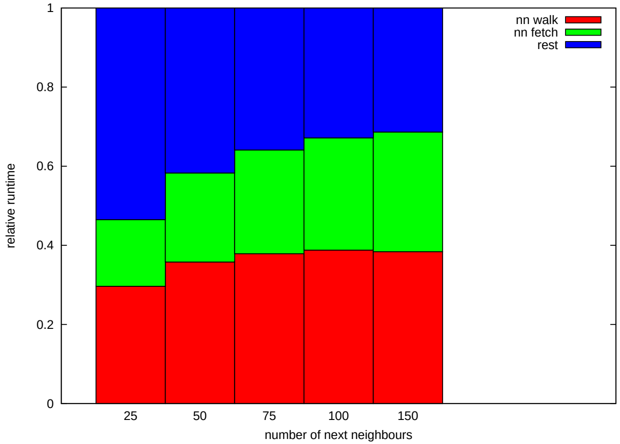
Weak scaling on Juropa

50 next neighbours, 150000 particles per process, absolute, logscale



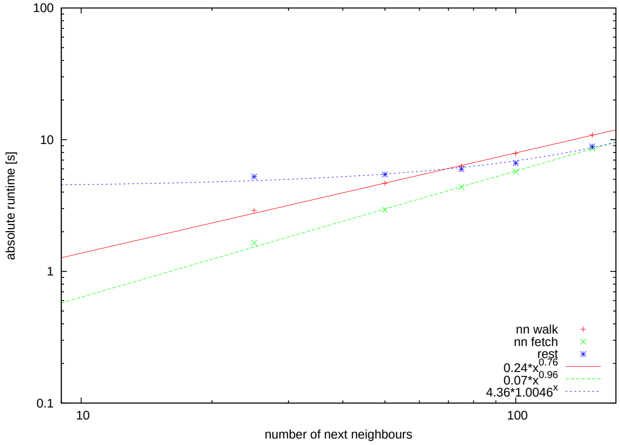
nn scaling on Juropa

10 nodes, 50000 particles per process, relative



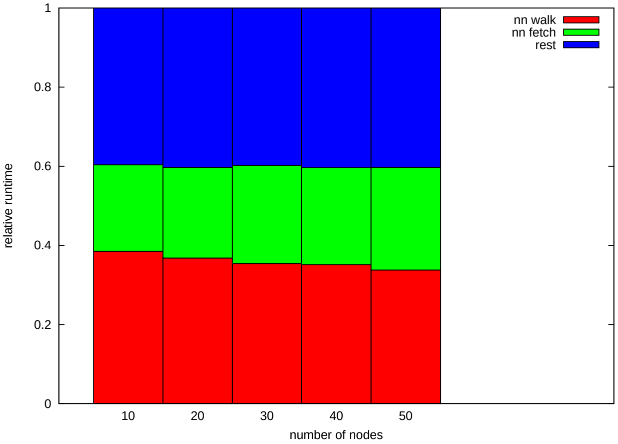
nn scaling on Juropa

10 nodes, 50000 particles per process, absolute, logscale



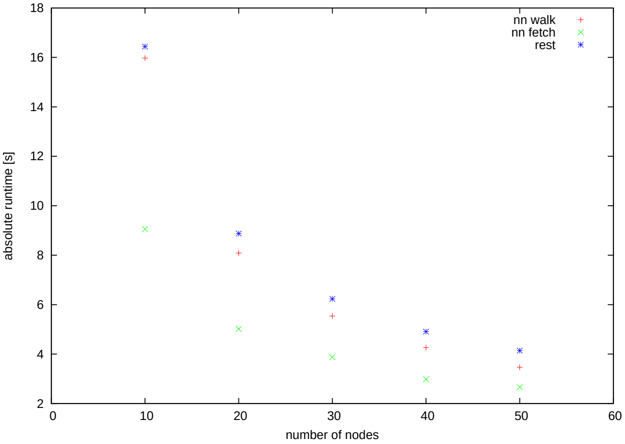
Strong scaling on Juropa

50 next neighbours, 12 mio particles, relative



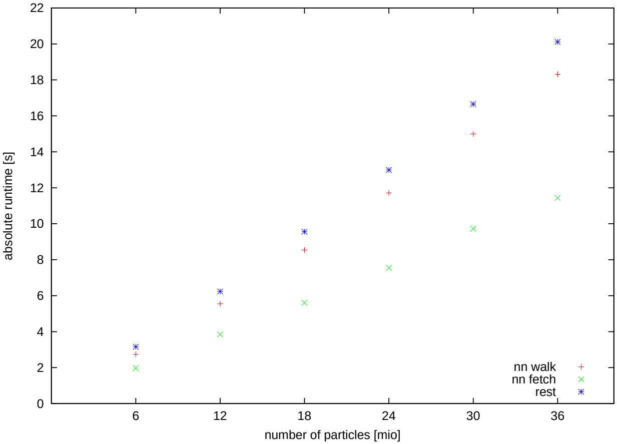
Strong scaling on Juropa

50 next neighbours, 12 mio particles, absolute



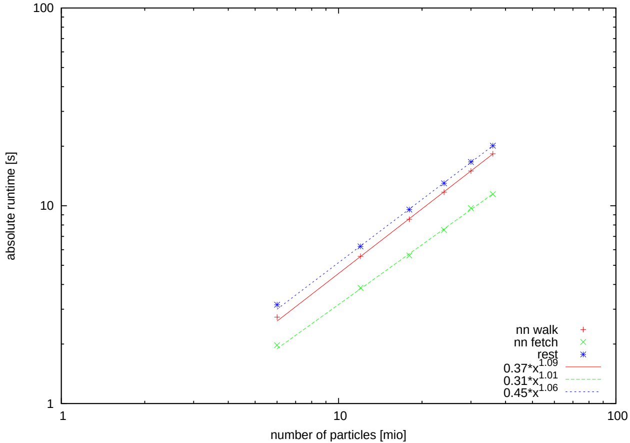
N scaling on Juropa

50 next neighbours, 30 nodes, absolute



N scaling on Juropa

50 next neighbours, 30 nodes, absolute, logscale



About the speaker

Andreas Breslau

JÜLICH SUPERCOMPUTING CENTRE (JSC)
Forschungszentrum Jülich GmbH

http://www.fz-juelich.de/jsc/JSCPeople/breslau