Parallel Summation of Inter-Particle Forces in SPH

Johannes Willkomm

Fifth International Workshop on Meshfree Methods for Partial Differential Equations

17.-19. August 2009

Bonn

Overview

- Smoothed particle hydrodynamics (SPH)
- Grid data structure
- Summation exploiting symmetry
- Performance results
- Conclusion

SPH Particles

- In SPH in d dimensions, a particle is a tupel $p = (\rho, \mathbf{v}, \mathbf{x})$
  - Density ρ, scalar
  - Velocity $\mathbf{v}$, d dimensional vector
  - Position $\mathbf{x}$, d dimensional vector
- Particle interaction for a particle p defined by ODE

$$\frac{\mathrm{d}\,p}{\mathrm{d}\,t} = \sum_{q \in Particles} f(p,q) + f_{\mathrm{G}}(p)$$

- Discretization of PDE with particles as interpolation points
  - Using kernel functions with local support

Force Summation Function

- Force summation function

$$\mathbf{u} = F(\delta t, \mathbf{p})$$

- Given vector of N particles $\mathbf{p}$ and time step δt
  - Compute vector of updates $\mathbf{u}$
    - Such that new state is $\mathbf{p} = \mathbf{p} + \mathbf{u}$, in the case of explicit Euler time integration
    - Other time integrators (Runge-Kutta, Predictor-Corrector) may call F several times with intermediate states $\mathbf{p}'$
  - Compute $\mathrm{d}p_i/\mathrm{d}t$ for all particles p_i

$$u_i = \delta t \frac{\mathrm{d}\, p_i}{\mathrm{d}\, t} = \delta t \sum_{q \in Particles} f(p_i, q) + f_{\mathrm{G}}(p_i)$$

Locality and Symmetry

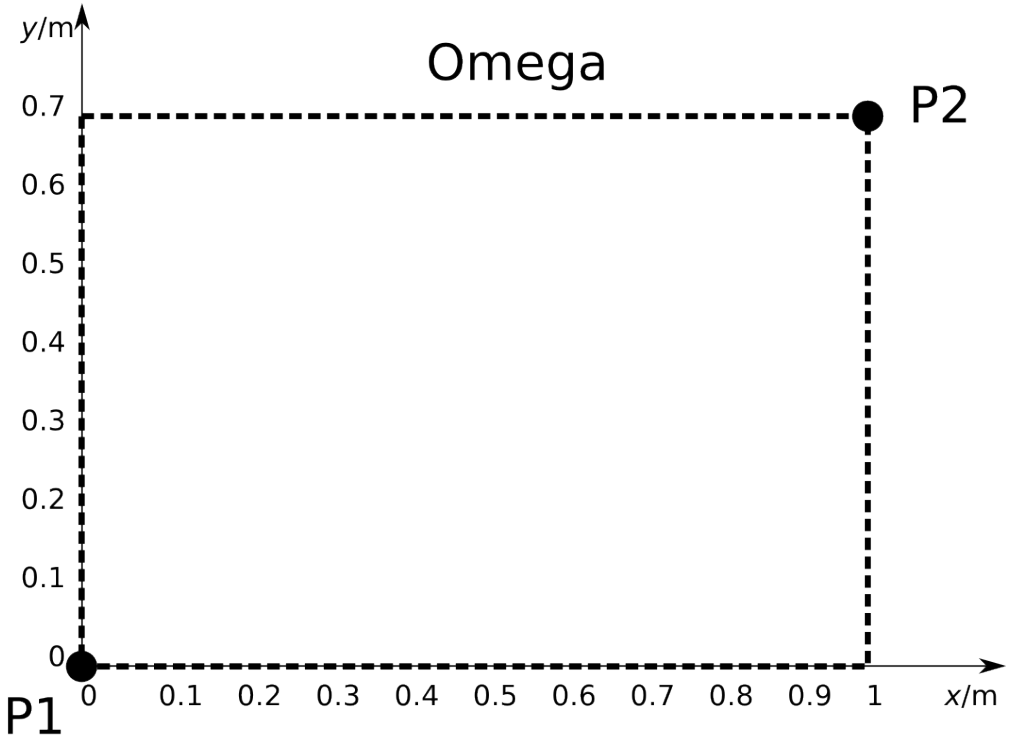
- The particle interactions are local
  $f(p_i, p_j) = 0$ if $|\mathbf{x}_j - \mathbf{x}_i| >= H$, the kernel support radius
- Symmetry of classic particles [Monaghan94]
  - f is symmetric wrt. density and anti-symmetric wrt. velocity, if particles have the same mass
- Compute $c = f(p_i, p_j)$ and update both u_i and u_j

```
u[i].velocity += c.velocity; u[j].velocity -= c.velocity
u[i].density += c.density;   u[j].density += c.density
```

- In [Ferrari09] f is only "almost symmetric"
  - $f(p_i, p_j)$ and $f(p_j, p_i)$ still share many common subexpressions

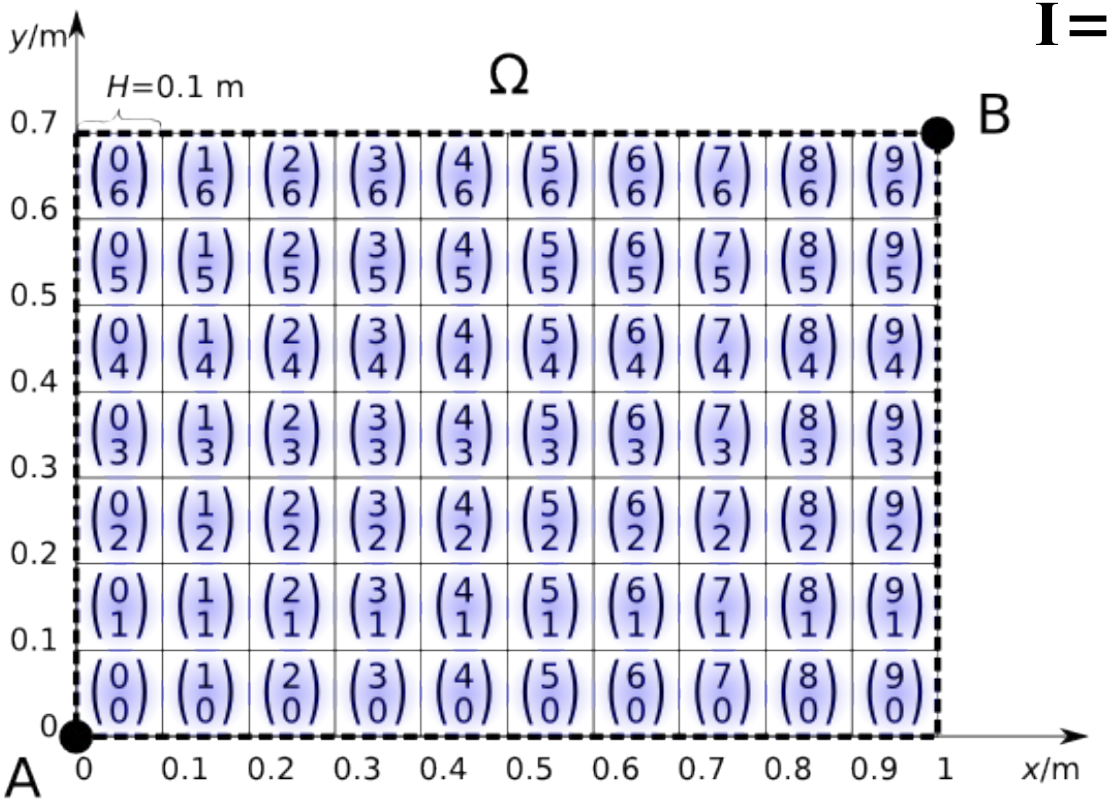
Computational Domain

- The d dimensional computational domain is the cuboid Ω spanned by two points A and B
- Example: $d = 2$, $A = (0, 0)$, $B = (1, 0.7)$



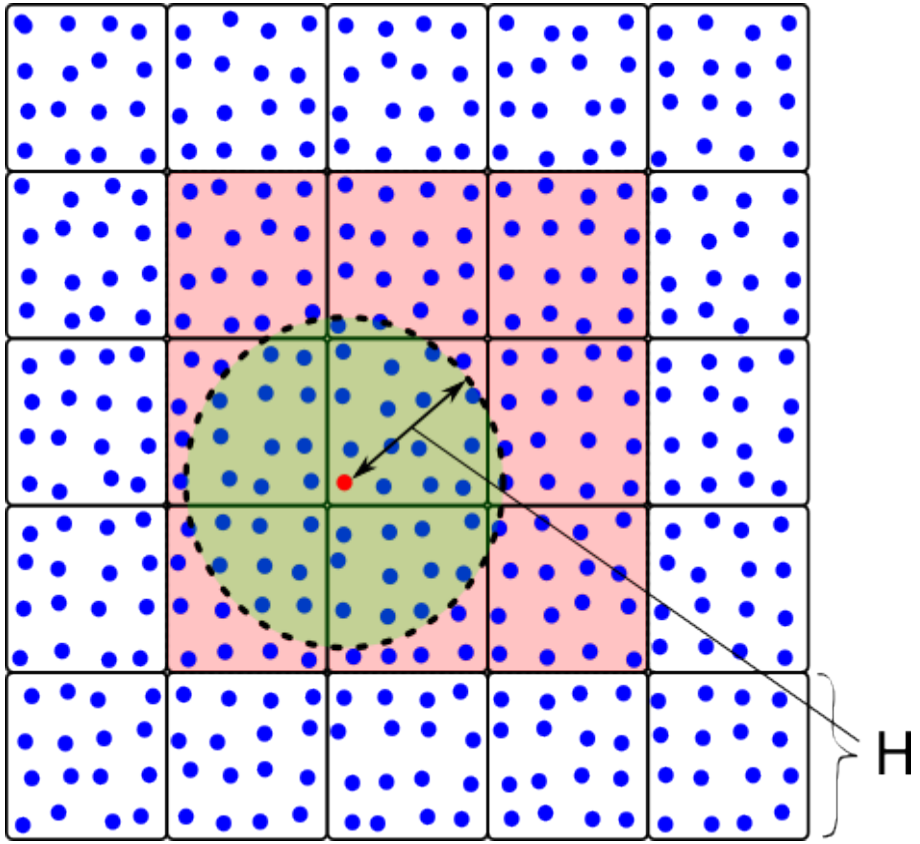
- Use grid-based lookup structure
  - d dimensional rectangular grid with cell width H
  - Define grid cell of particle at position $\mathbf{x}$ is defined by the integer vector $\mathbf{I}$

$$\mathbf{I} = \text{floor}\left(\frac{\mathbf{x} - A}{H}\right)$$



Finding Contributing Particles

- For p_i, look in own and surrounding cells for particles p_j with distance $< H$
  - All other p_k can be excluded because $f(p_i, p_k) = 0$
- 3^d cells
  - 1D: 3 cells
  - 2D: 9 cells
  - 3D: 27 cells



Evaluation of F

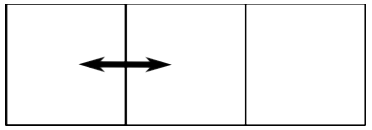
- Two serial algorithms to evaluate F
  - Naïve algorithm
  - Symmetry-exploiting algorithm
- Parallel evaluation of F
  - Naïve algorithm is easily parallelizable
  - How can symmetry be exploited in parallel?

Serial Summation of Forces

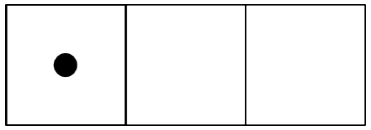
- Naïve algorithm
  - Consider each particle p_i in turn
    - Find neighbors and sum up contributions in u_i
  - Each pair of particles will be considered twice
    - First p_i is considered and neighbour p_j is found
    - Again when p_j is considered p_i will be found among neighbours
- Exploiting symmetry [LiuLiu04]
  - Construct index list of neighbouring particles
  - For each entry (i, j) compute $c = f(p_i, p_j)$ and update both u_i and u_j

Parallel Summation of Forces

- To exploit symmetry, find a method which
  - Considers each pair of particles (p_i, p_j) only once
  - Updates both u_i and u_j in a thread-safe manner
  - The list-based approach is not efficiently parallelizable
- Define two helper functions
  - sumF2$(\mathbf{I}, \mathbf{J})$
    - Compute all relations of particles p_i in cell $\mathbf{I}$ and p_j in cell $\mathbf{J}$, updating u_i and u_j



  - sumF1$(\mathbf{I})$
    - Compute all relations of particles p_i and p_j in cell $\mathbf{I}$, updating u_i and u_j



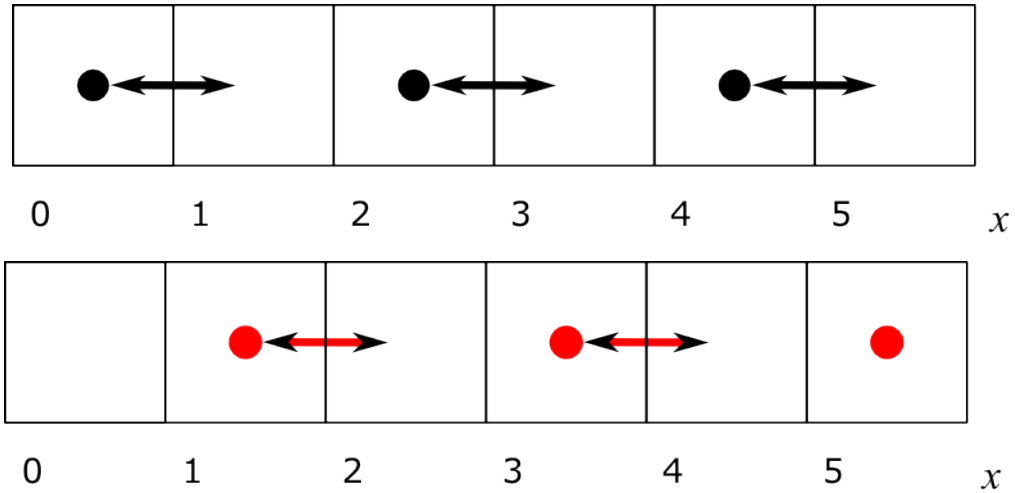
Parallel Algorithm in 1D

```
parallel for i in {0, 1, ..., numCells/2} do
   if both cells exist then
      sumF2(2*i, 2*i + 1)
   sumF1(2*i)
parallel for i in {0, 1, ..., numCells/2} do
   if both cells exist then
      sumF2(2*i + 1, 2*i + 2)
   sumF1(2*i + 1)
```

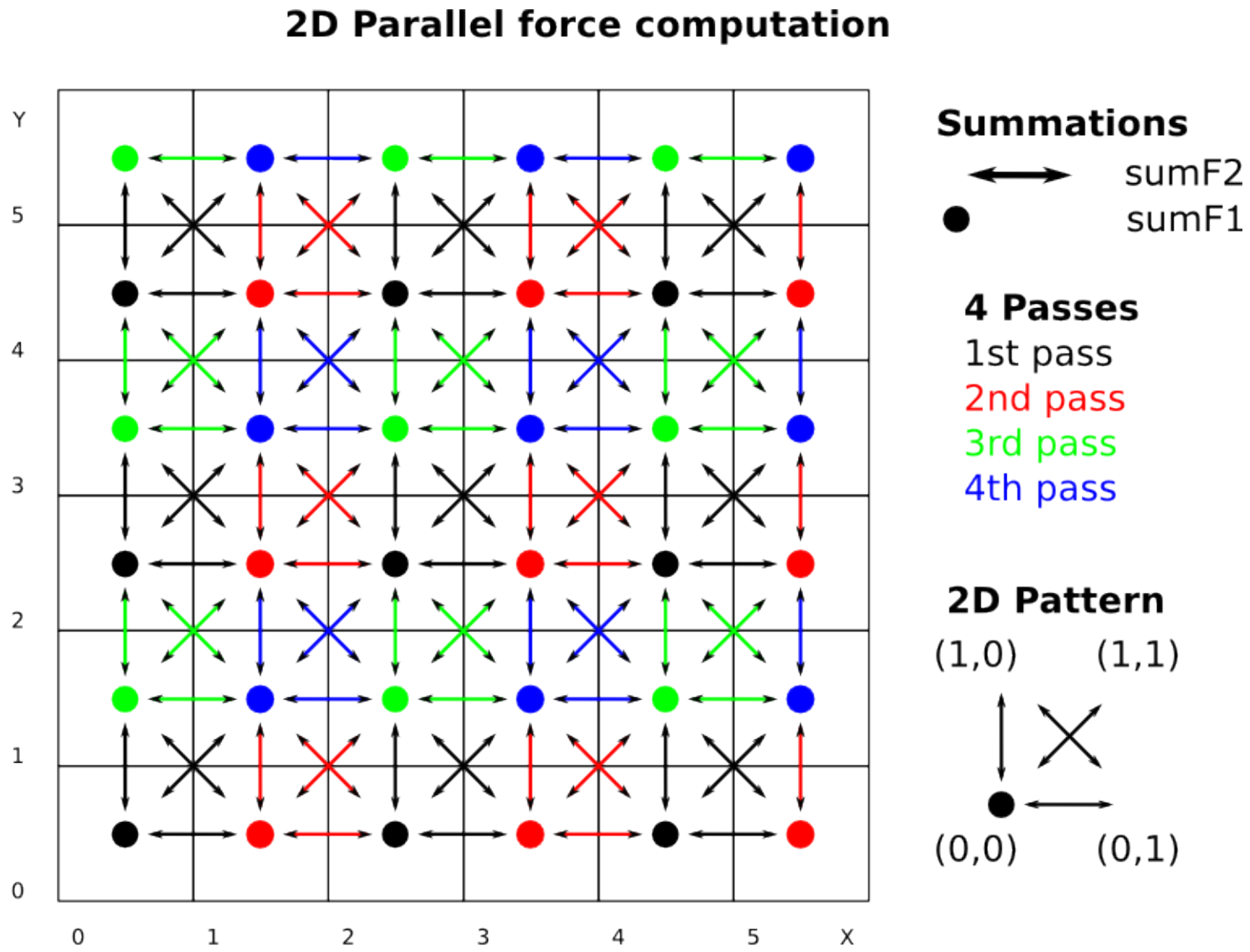
2 Passes
1st (even) pass
2nd (odd) pass

Summations

↔ sumF2
● sumF1



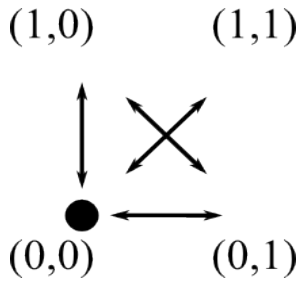
Graphical Solution in 2D



Generalization

- The version in 3D is now clear, but can this be to formulated as an algorithm for any d?
- In 2D the following pairs are considered for sumF2 of one pattern in local coordinates

(0,0) (0,1)
(0,0) (1,0)
(0,0) (1,1)
(0,1) (1,0)



- These pairs of tupels have in common
  - their scalar product is zero and
  - these are all possibilities where this is the case
    - Considering only I, J with $I < J$ in lexicographic order

Algorithm in d dimensions

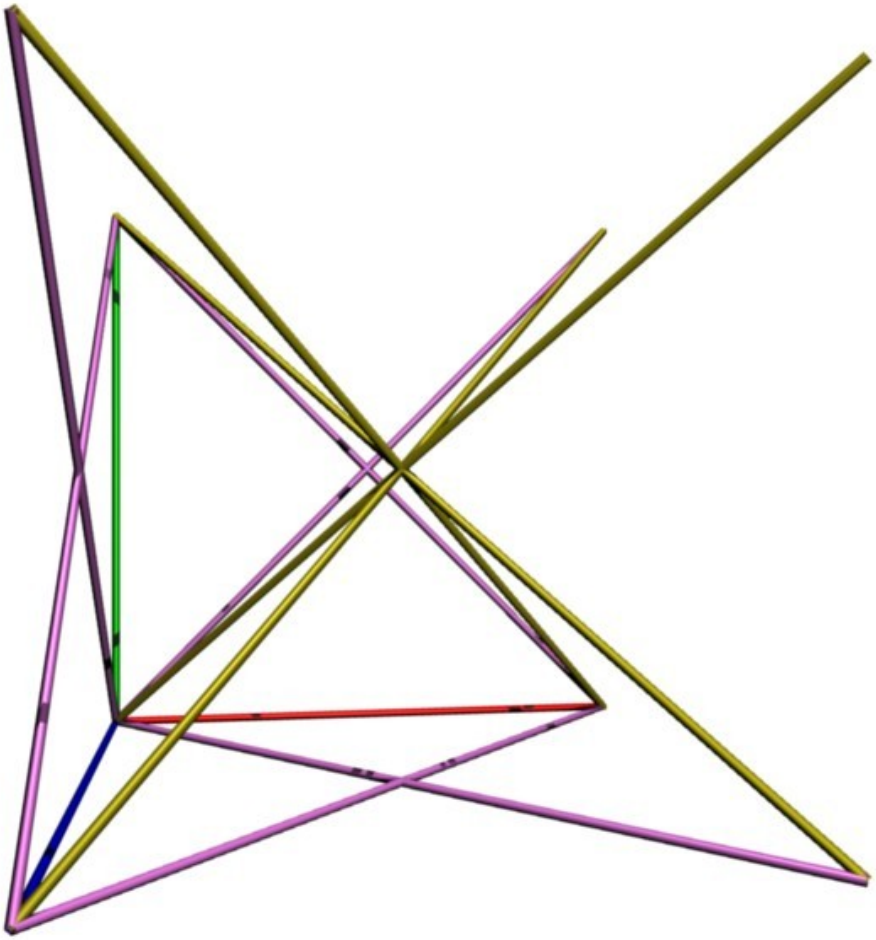
- Let o, i, j, k be integer vectors of dimension d

```
for o in {0,1}^d
   parallel for i in even grid cells
      sumF1(2*i + o)
      for j in {0,1}^d
         for k in {0,1}^d
            if k<j and j*k = 0 and both cells exist
            then
              sumF2(2*i + o + j, 2*i + o + k)
```

3D Pattern

- The 3D pattern looks like this

It consists of:
- 3 1D patterns (red, green, blue) which are part of
- 3 2D patterns (violet)
- 4 lines connecting the 8 cells (gold)

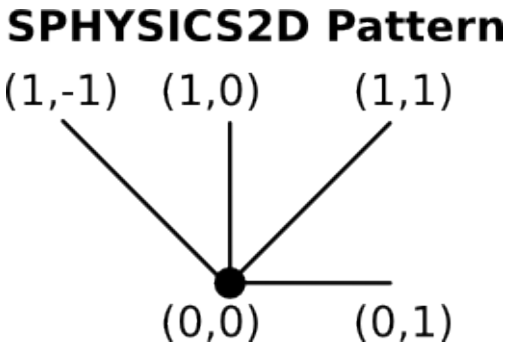


Related Ideas

- In SPHYSICS2D [Dalrymple08] is a similar idea
  - **Serial** algorithm exploiting symmetry
    - The following pairs of cells are considered

      (0,0) (0,1)
      (0,0) (1,1)
      (0,0) (1,0)
      (0,0) (-1,1)

      Special case (0,0) (0,0)



  - Complete but not thread-safe, patterns from one sweep overlap
- In [Sbalzarini06] the 2D and 3D patterns are given by listing the pairs of cell indices
  - There is also a drawing of the 2D pattern

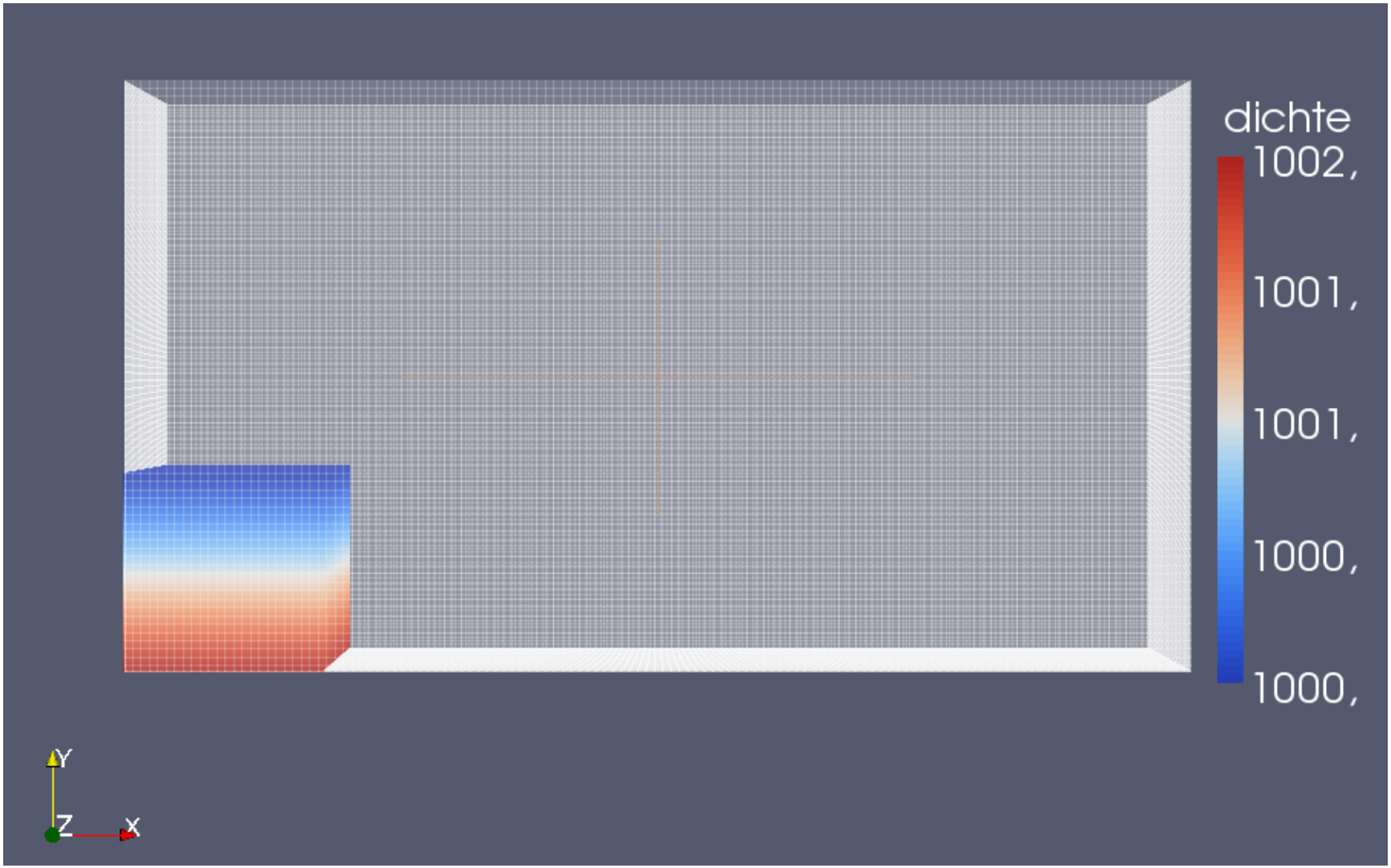
Performance Test

- 3D test case "dambreak" [Ferrari09] with
  - $A = (0, 0, 0), B = (3.3, 1.8, 0.61)$
  - Explicit time stepping, 5 steps, Ferrari particles, wendland2 (quintic) kernel, $\delta x = 0.00594$ m, $H = 0.01188$ m, boundary resolution $\delta x/1.5$
  - 10^6 water and 1014966 boundary particles
  - 278 x 152 x $52 = 2197312$ grid positions
- Use hashing on grid cells
  - 2^{dM} hashed grid positions
    - M=6 $\rightarrow 2^{18} = 262144$
    - M=7 $\rightarrow 2^{21} = 2097152$

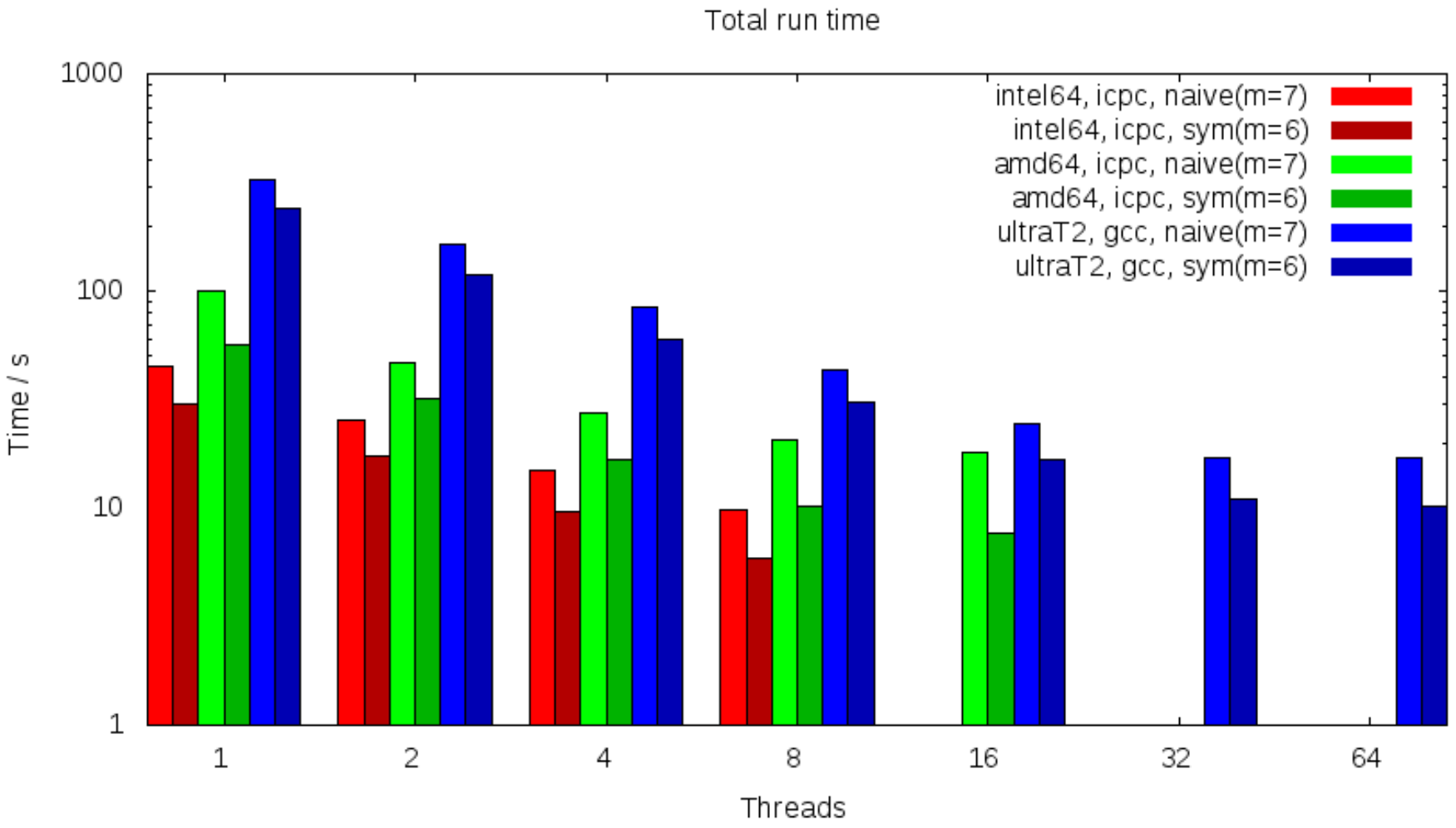
$$\tilde{\mathbf{I}} = \text{floor}\left(\frac{\mathbf{x} - A}{H}\right) \bmod 2^M$$

  - Naïve version works better with M=7: fewer collisions
  - Symmetric version is better with M=6: smaller world

3D Test Case Dambreak

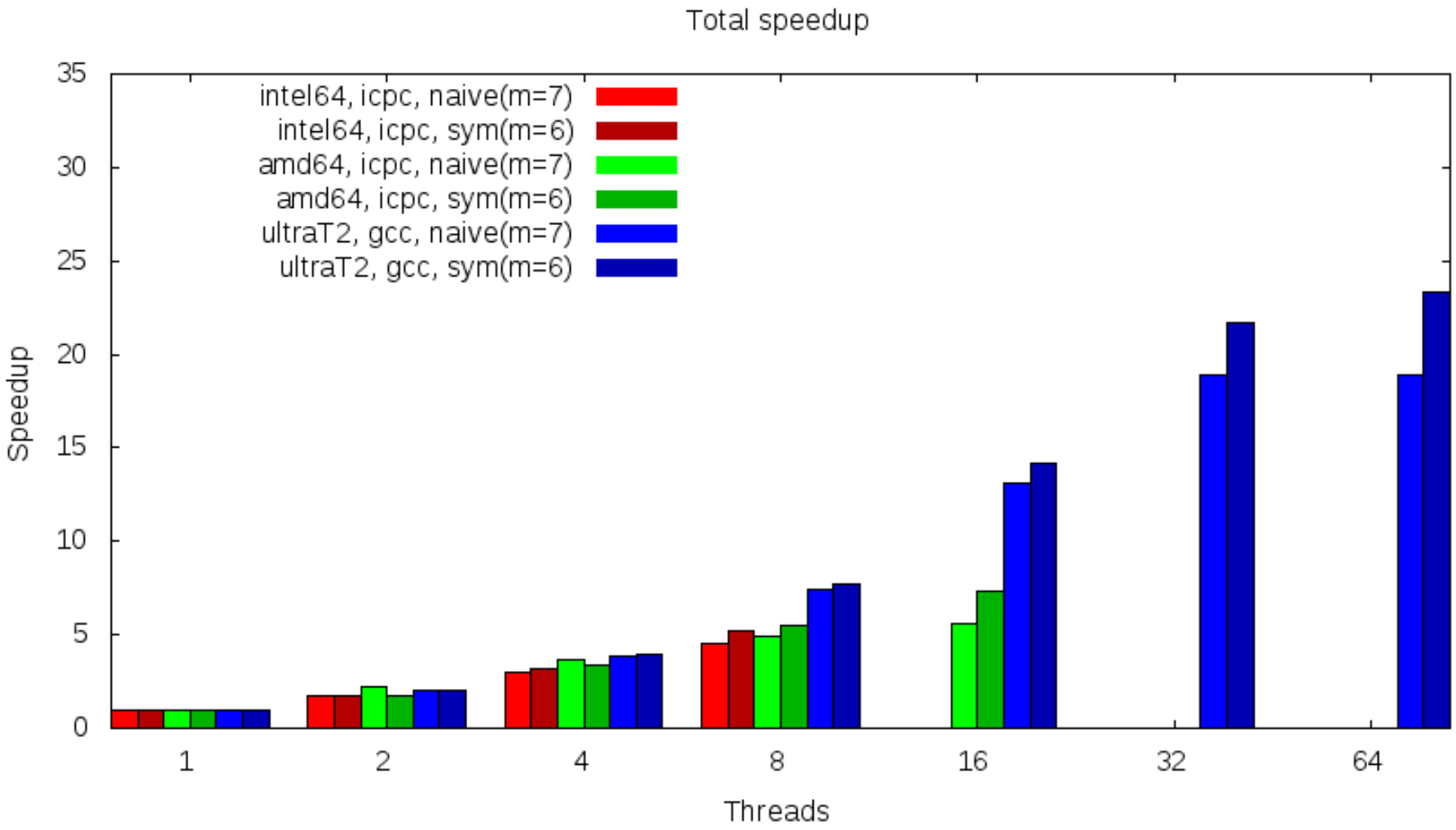


Total Runtime



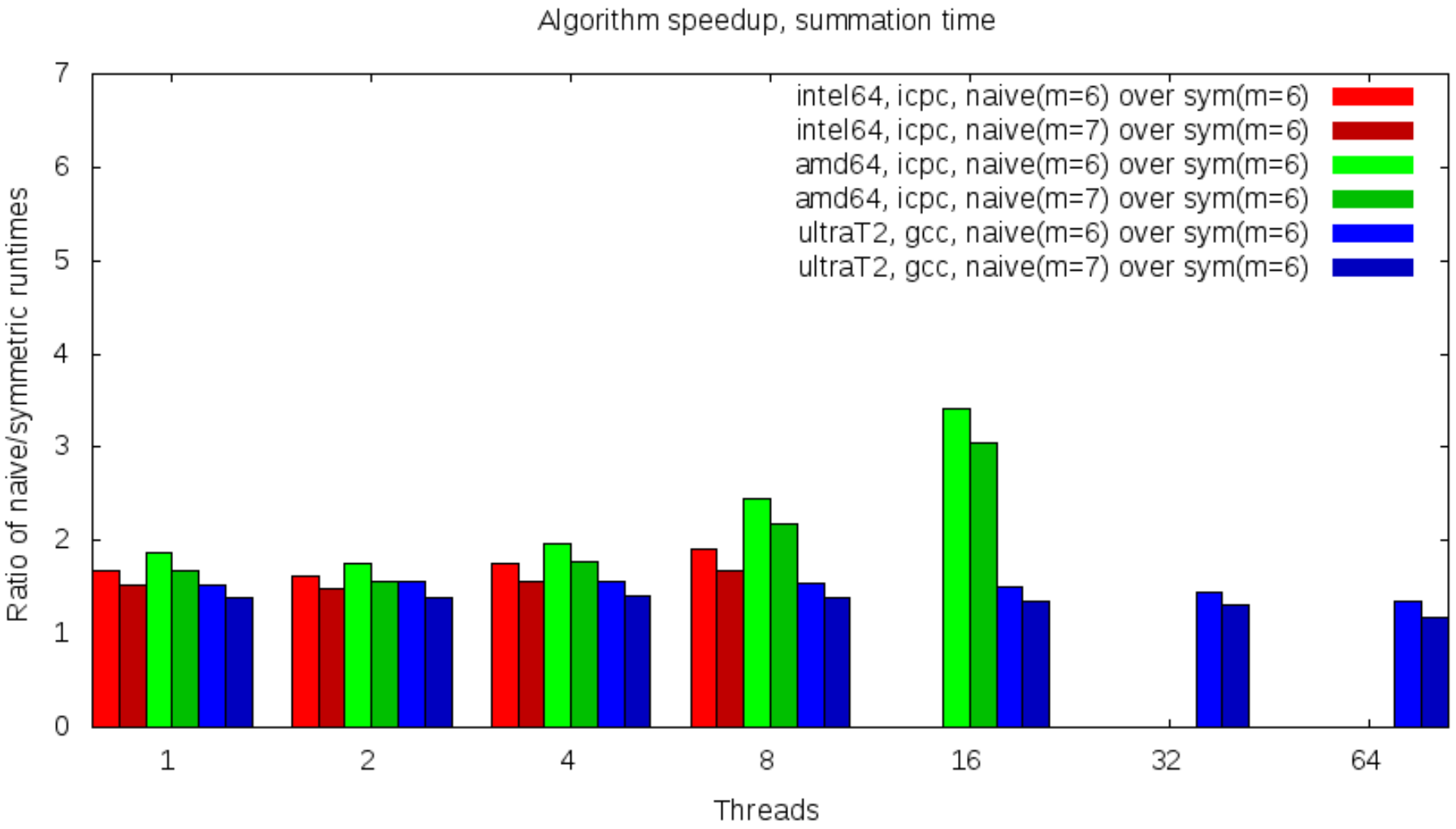
Intel64: Intel Xeon X7350 @ 2925 MHz running Linux 2.6.18-92.1.18.el5.sunhpc1, 1, 2, 4 and 8 threads
AMD64: AMD Opteron 8356 @ 2300 MHz running the same kernel, 1, 2, 4, 8 and 16 threads
UltraT2: SUN Niagara II T5120 running SunOS 5.10, Test with 1, 2, 4, 8, 16, 32 and 64 threads

Total Runtime Speedup



icpc: Intel C++ 11.0, GCC 4.4.1 headers, CXXFLAGS="-m64 -O3 -axSWT -fp-model fast=2 -ip"
gcc: GNU GCC 4.3.1, CXXFLAGS=" -m64 -O3 -ffast-math -mcpu=ultrasparc3"

How much faster is it?



- This is only the time spend in the summation algorithm.
- The symmetric parallel summation is about 2 times faster.

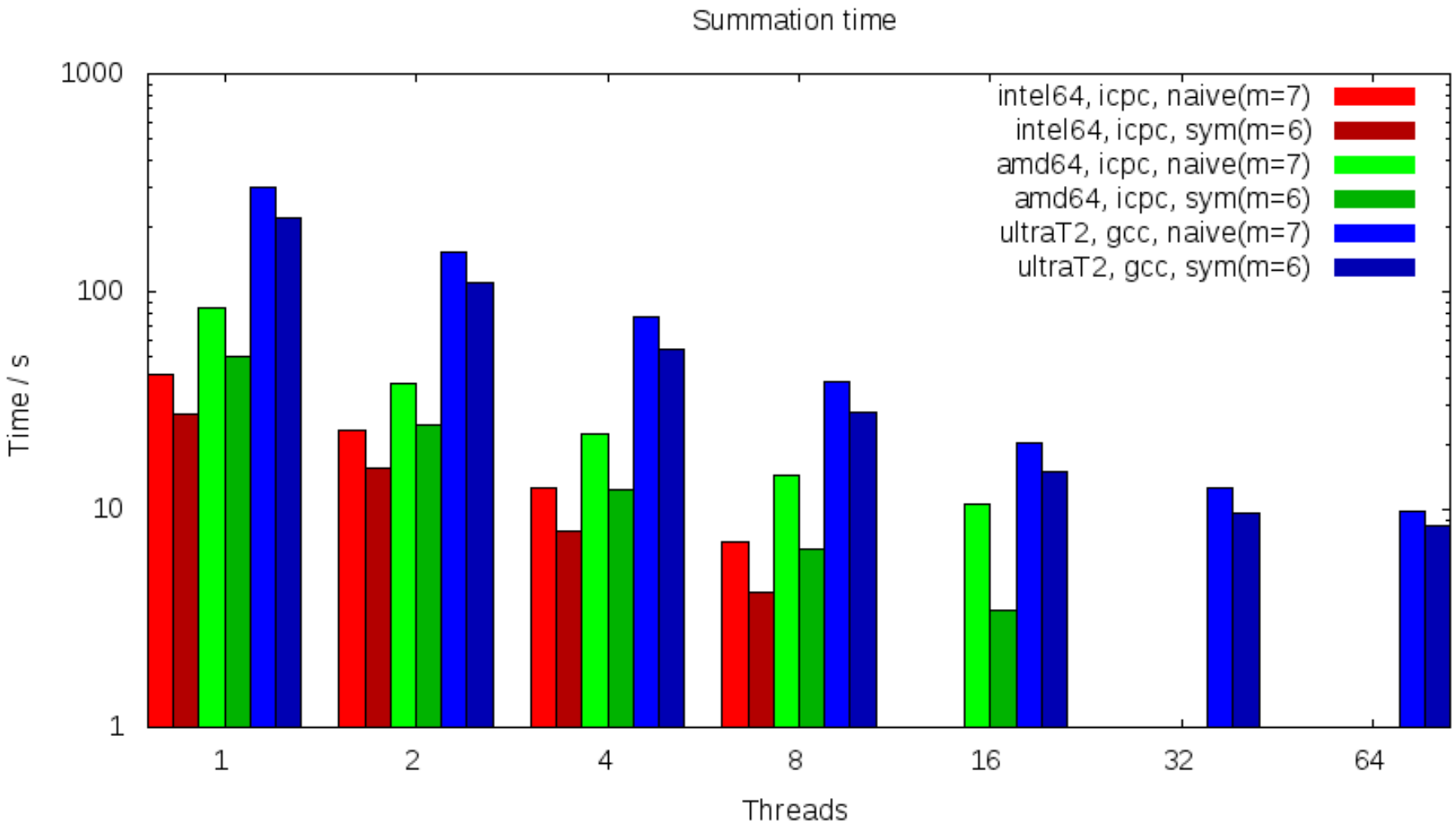
Conclusion

- Parallel and symmetry exploiting algorithm for summation of interpolation functions with local support
  - Iterate over the world not the particles
  - Bottom up binary domain decomposition
- Advantages
  - Is about twice as fast as naïve version
  - Scales as good as naïve version
- Disadvantages
  - Parameter M has to be configured correctly
  - No decomposition of memory access patterns, each sweep touches all grid cells

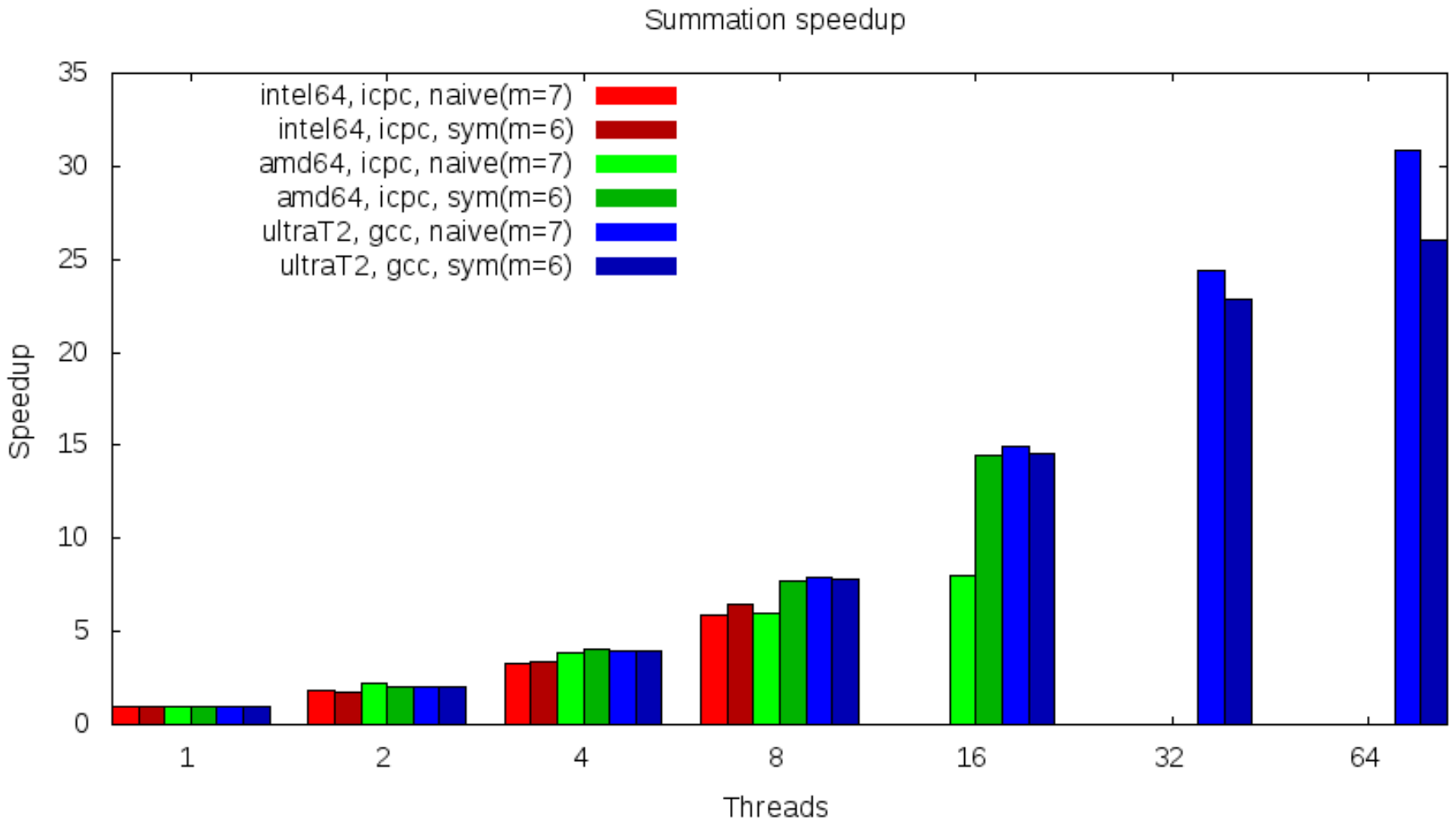
Literature

- [Monaghan94] – J. J. Monaghan, “Simulating free surface flows with SPH,” J. Comp. Physics, vol. 110, no. 2, pp. 399–406, 1994
- [LiuLiu03] – G. R. Liu and M. B. Liu, “Smoothed Particle Hydrodynamics.” Singapore, World Scientific Publishing Co. Pte. Ltd, 2003.
- [Dalrymple08] – User Guide for the SPHysics Code v1.2, http://wiki.manchester.ac.uk/sphysics, 2008
- [Sbalzarini06] – 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., vol. 215, no. 2, pp. 566–588, 2006.
- [Ferrari09] – A. Ferrari, M. Dumbser, E. F. Toro, and A. Armanini, “A new 3D parallel SPH scheme for free surface flows,” Computers & Fluids, vol. 38, no. 6, pp. 1203–1217, 2009.

Times, just the summation



Speedup, just the summation



Ratio of Hits to Combinations

- a(d) – Number of calls to sumF1 and sumF2 in d dimensions
  - 1,2,5,14,41,122,365,1094, … A007051
  - a(0) = 1, a(d) = 3*a(d-1) - 1
- b(d) – Number of calls to sumF2
  - 0, 1, 4, 13, 40, 121, 364, 1093, … A003462
  - $b(d) = 3^{d-1}/2$
- t(d) – Number of tests j*k = 0 in loop
  - $t(d) = 2^{2d} / 2$