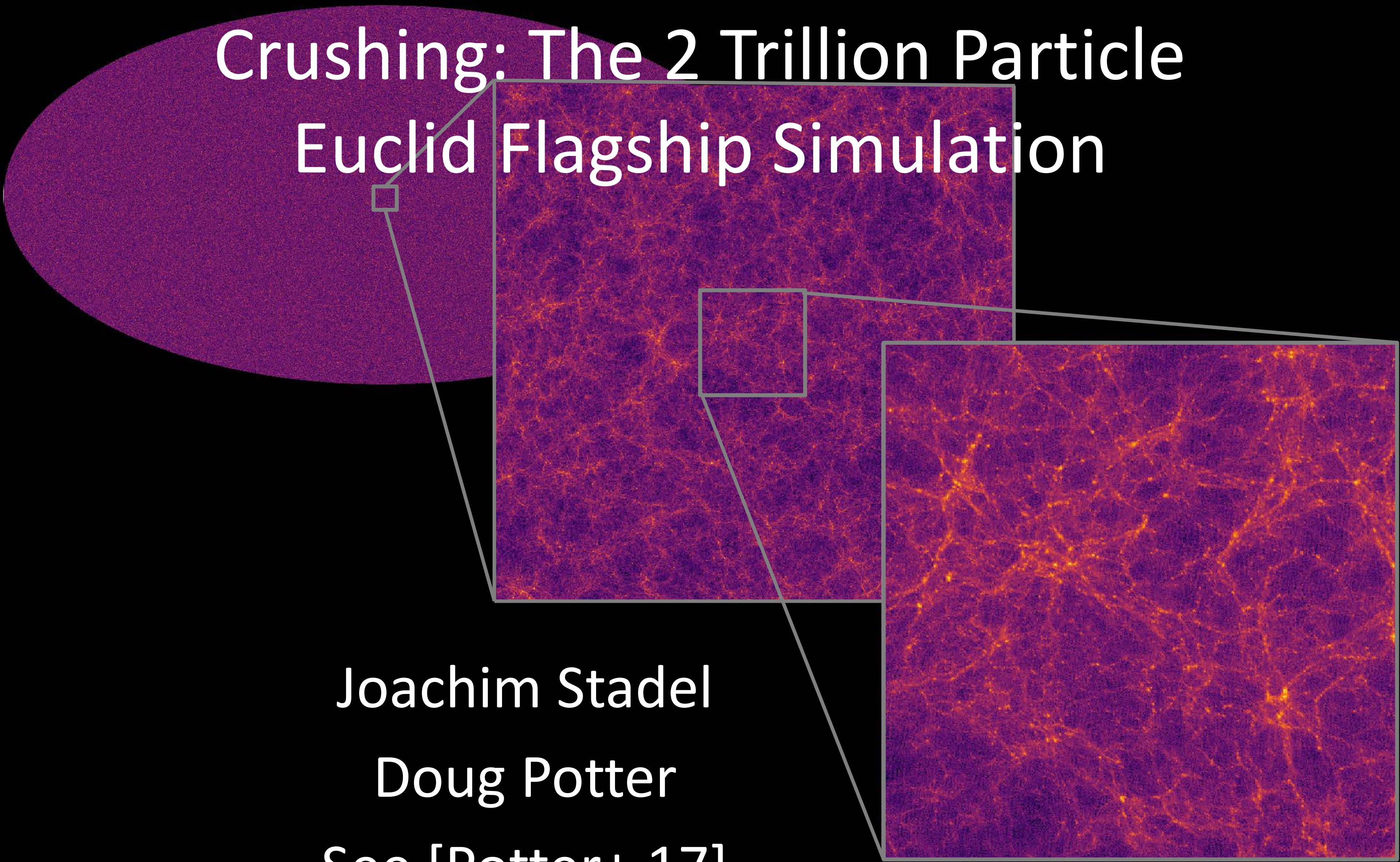
Fast Multipole, GPUs and Memory Crushing: The 2 Trillion Particle Euclid Flagship Simulation



Joachim Stadel

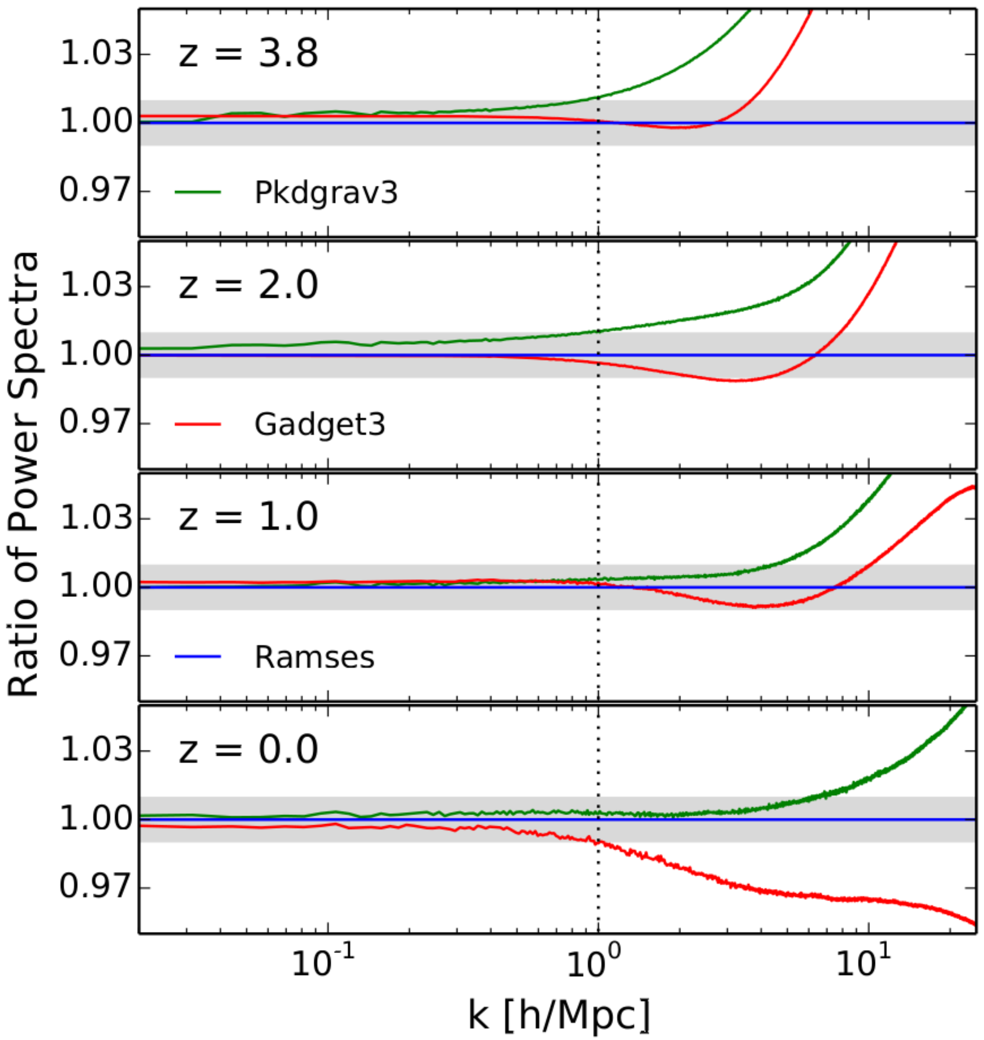
Doug Potter

See [Potter+ 17]

Outline

- Predictability
- The Euclid Flagship Light-cone and Mock
- Pkdgrav3: Fast Multipole Method
- Hybrid Computing
- Dual Tree Multistepping
- Performance and Scaling
- Light-cone Algorithm
- Dollars and Sense

Predictive?...

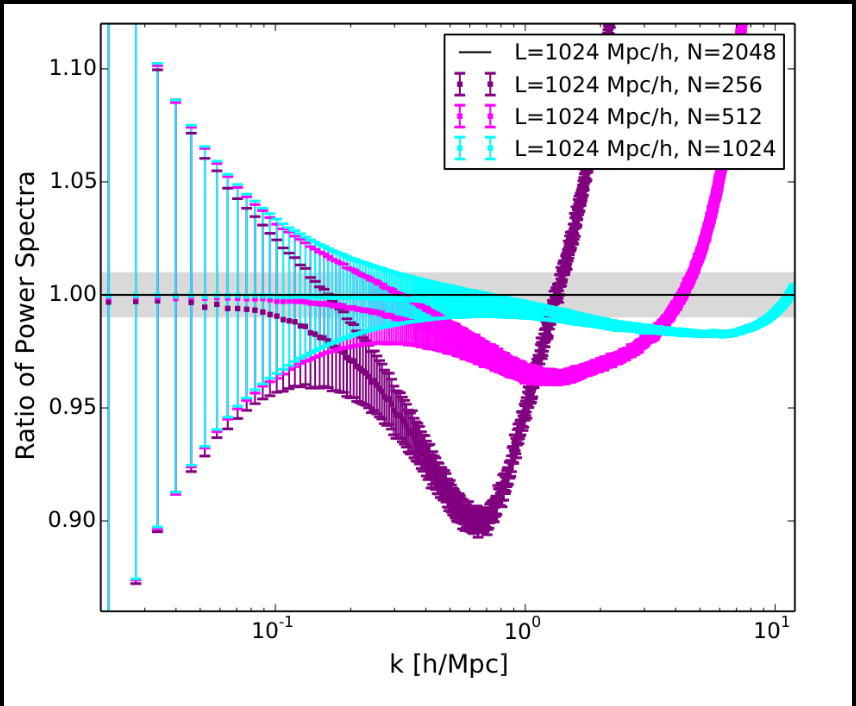


Note: all 3 codes have very different Poisson solvers and integration methods!

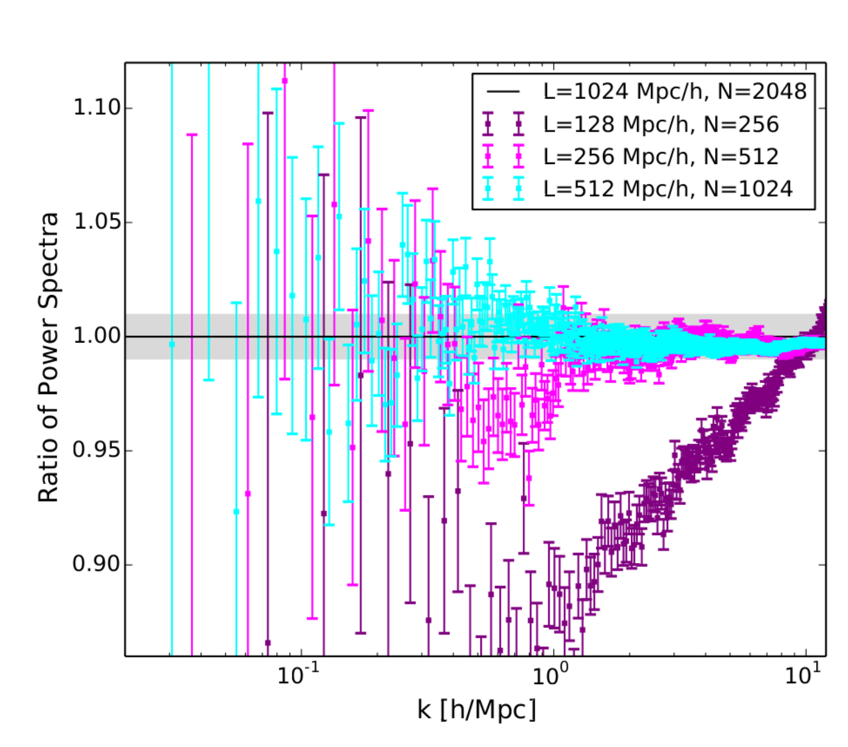
See
Schneider, et al. 2015

Quantify systematics

Convergence with resolution

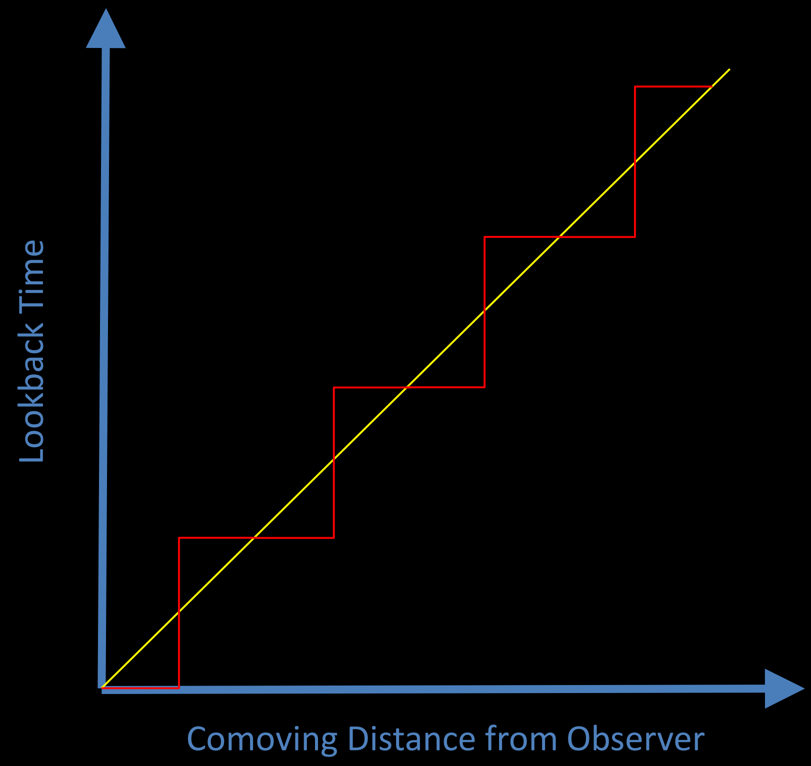


Convergence with box size



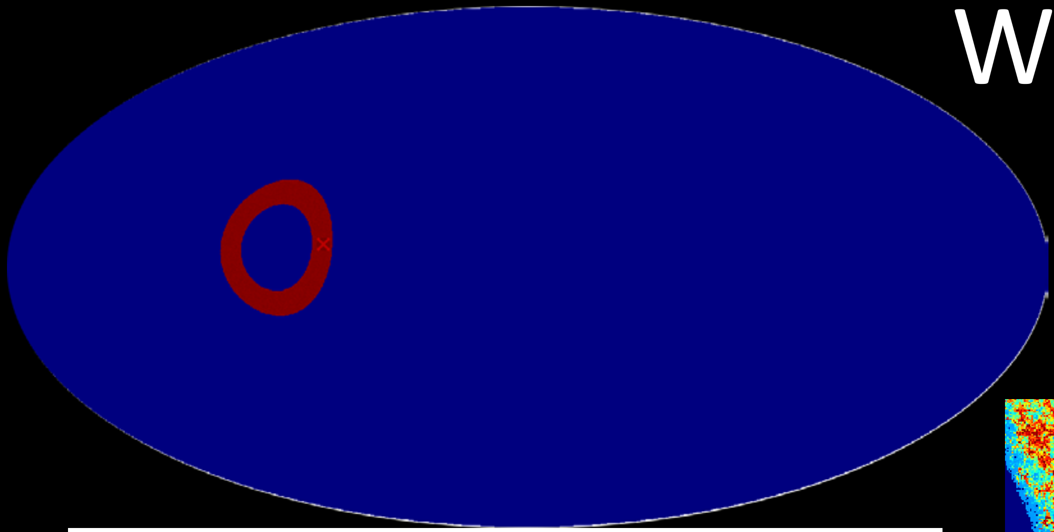
2 trillion particle values: L=3780 Mpc/h N=12'600 → N=3413 in 1024 Mpc/h

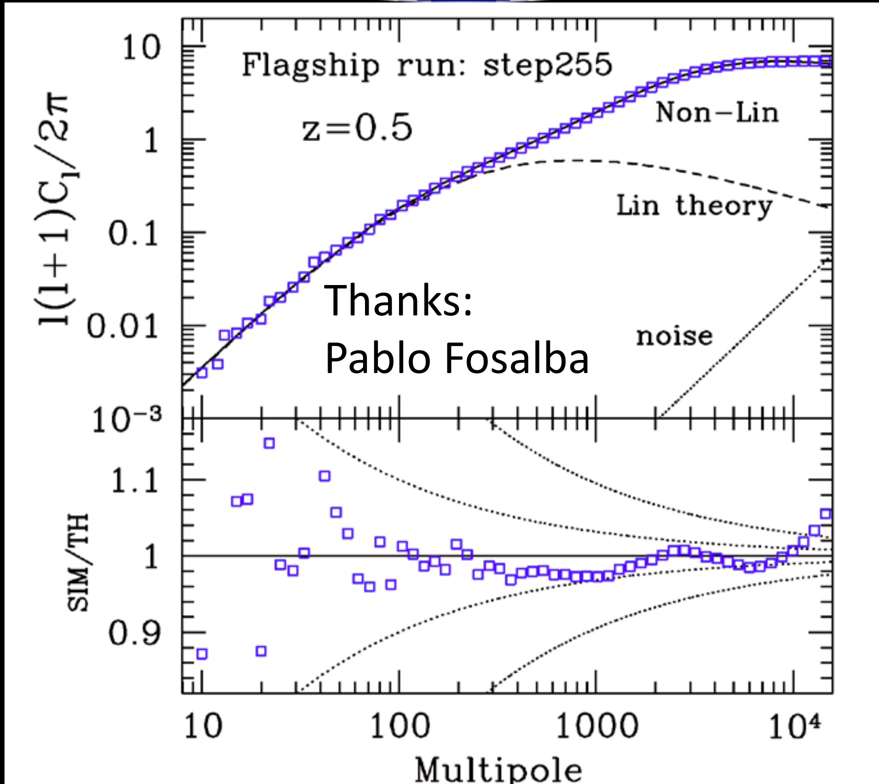
Produce the ideal of a light cone without discontinuous jumps in the observed structure.

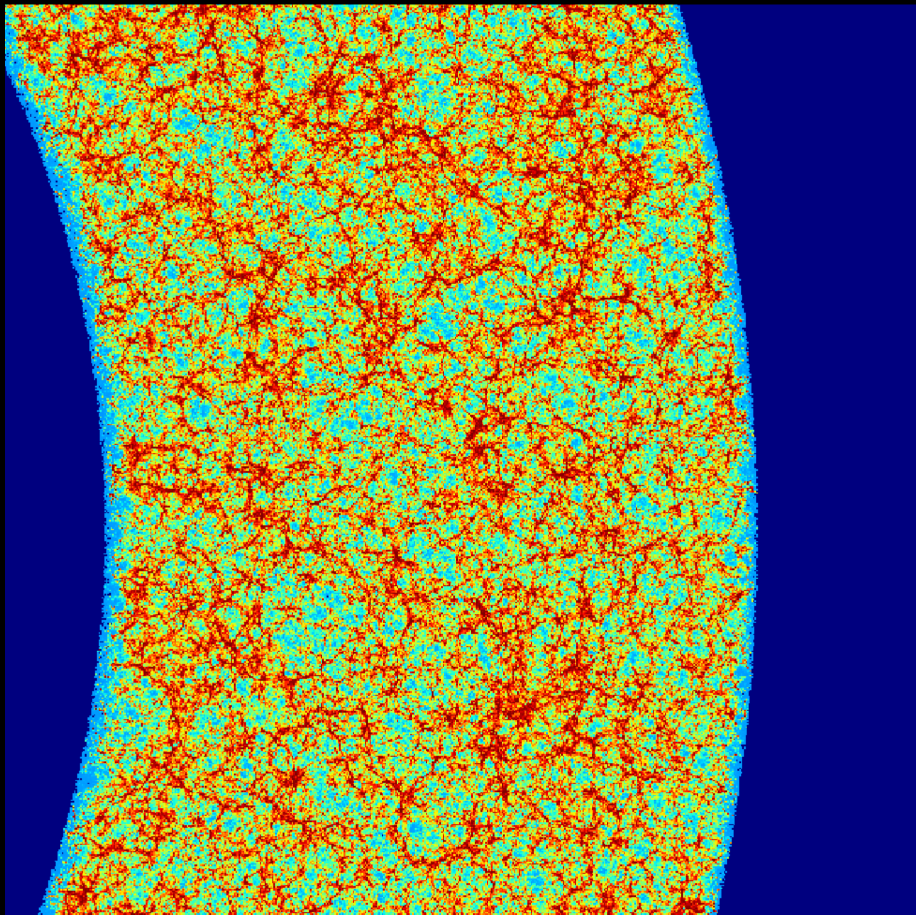


Weak lensing maps

There are O(400) such maps which form a set of spherical, concentric, lensing planes to distort the shapes of the background galaxies.





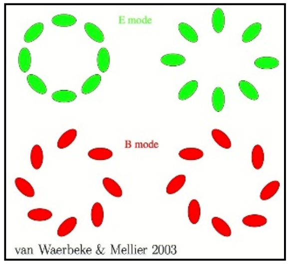


Euclid Validation: Weak Lensing

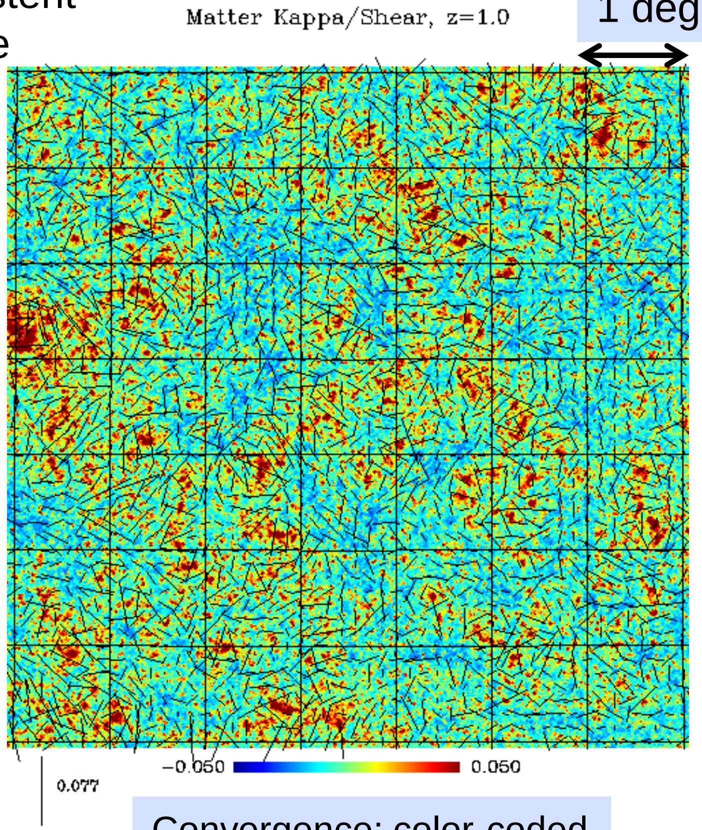
EUCLID CONSORTIUM

- Shear patterns and clustering show right behaviour down to small scales

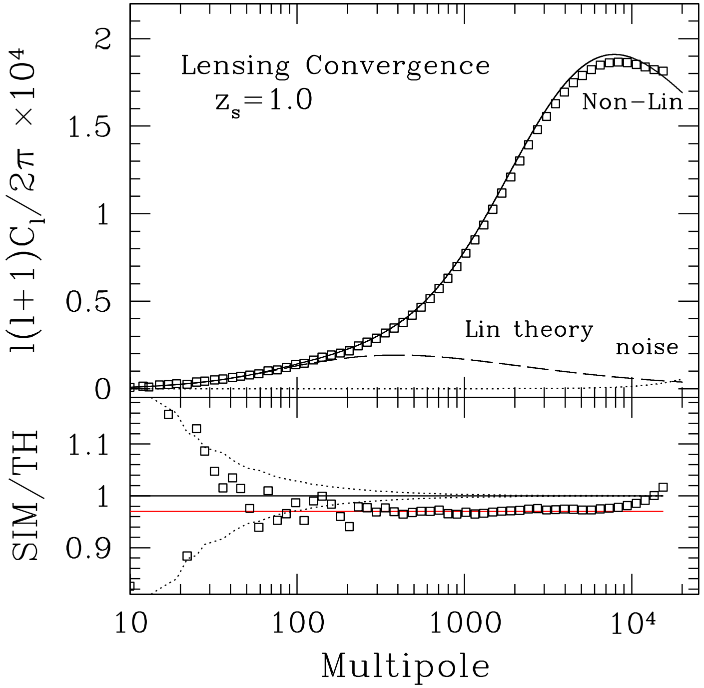
Patterns consistent with no B-mode



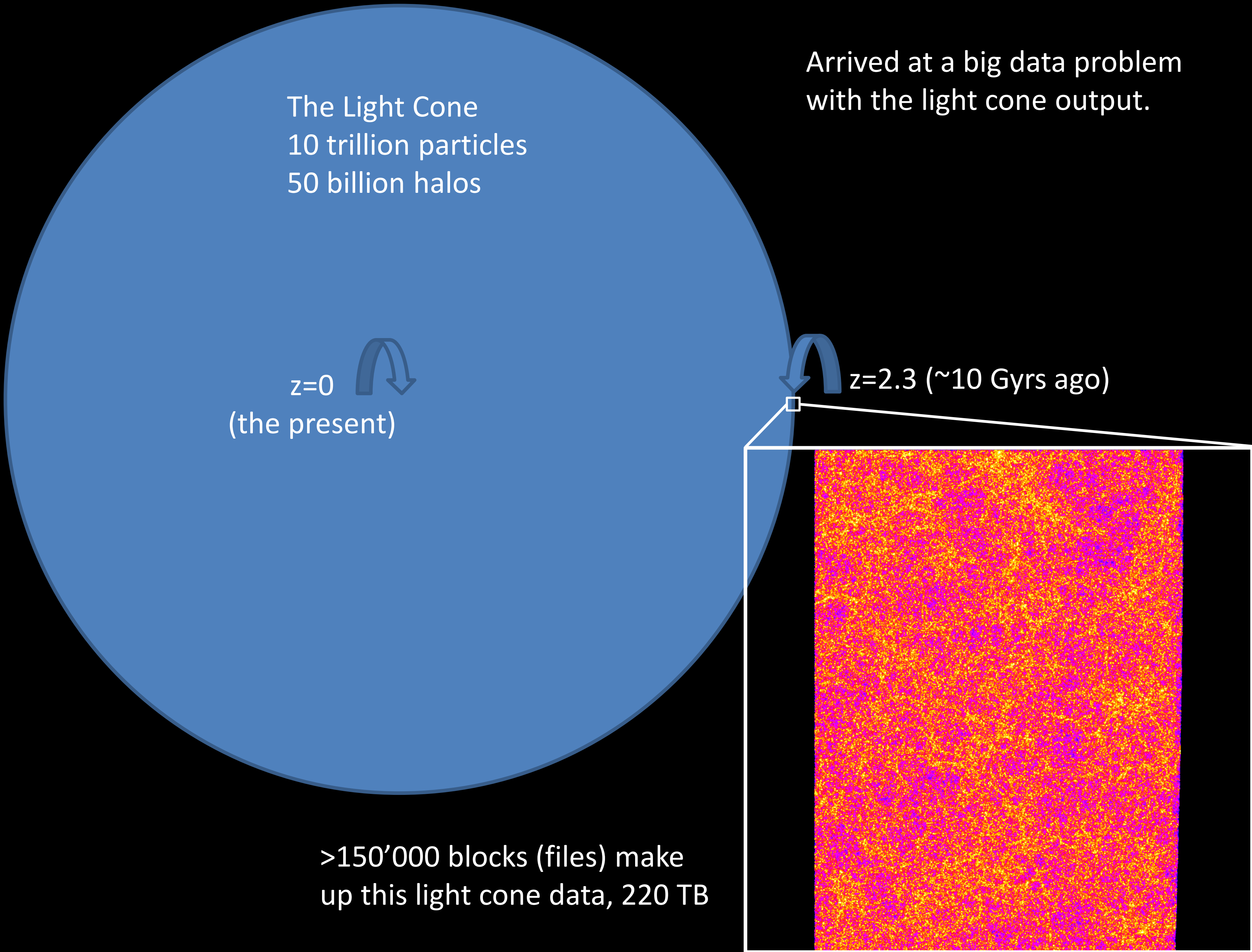
1 deg



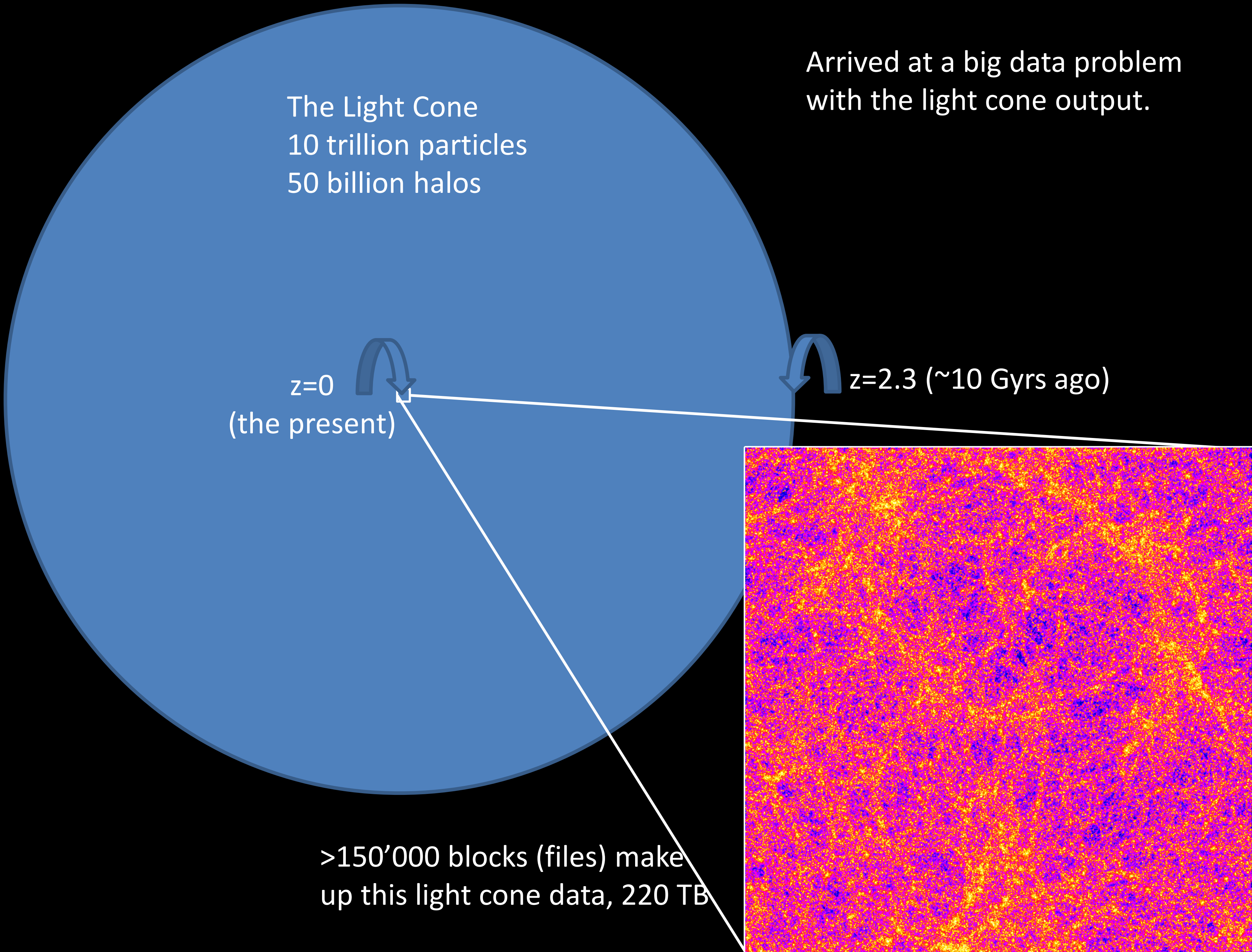
Convergence: color-coded
Shear: overlaid vectors

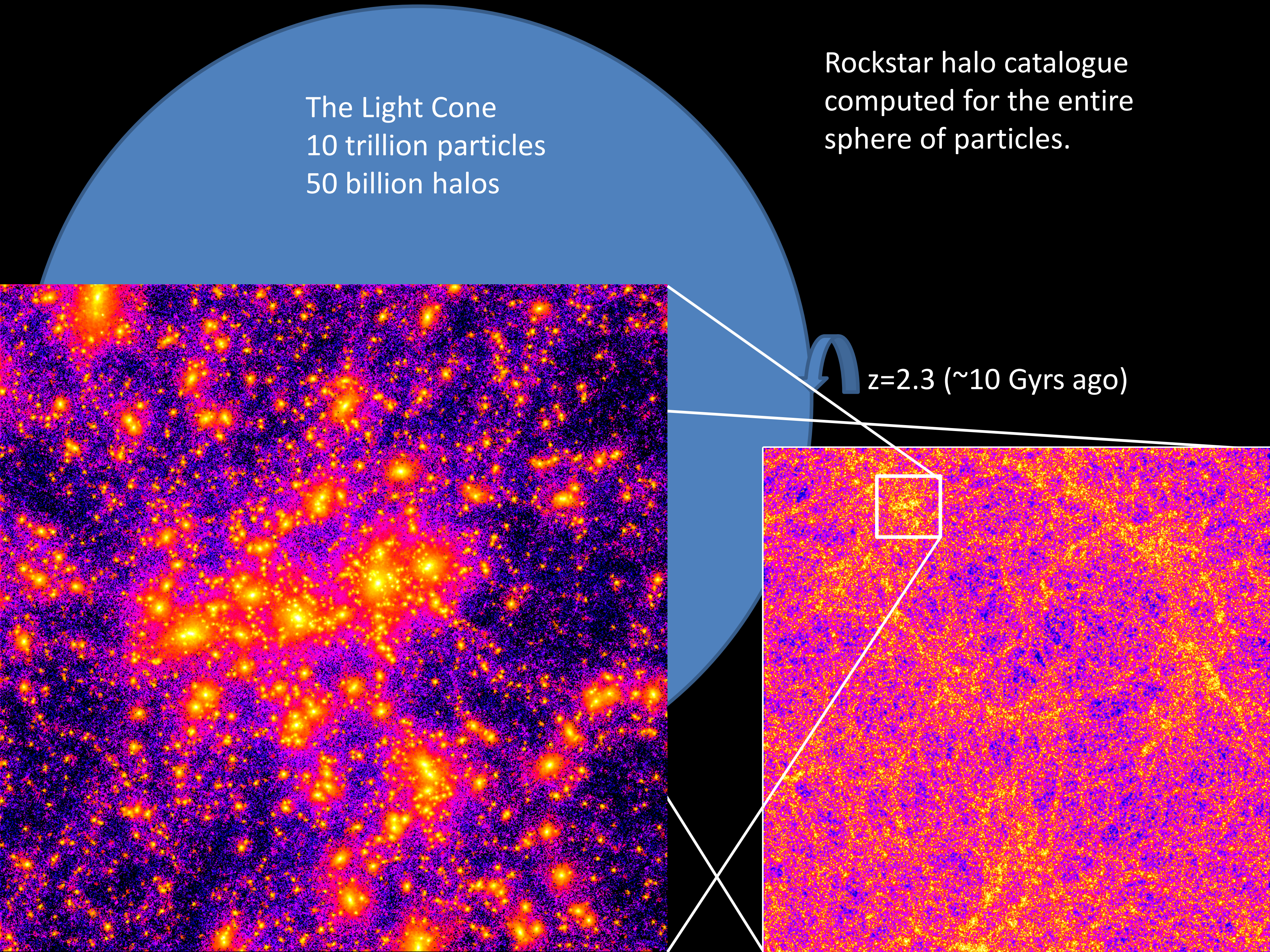


Arrived at a big data problem with the light cone output.



Arrived at a big data problem with the light cone output.

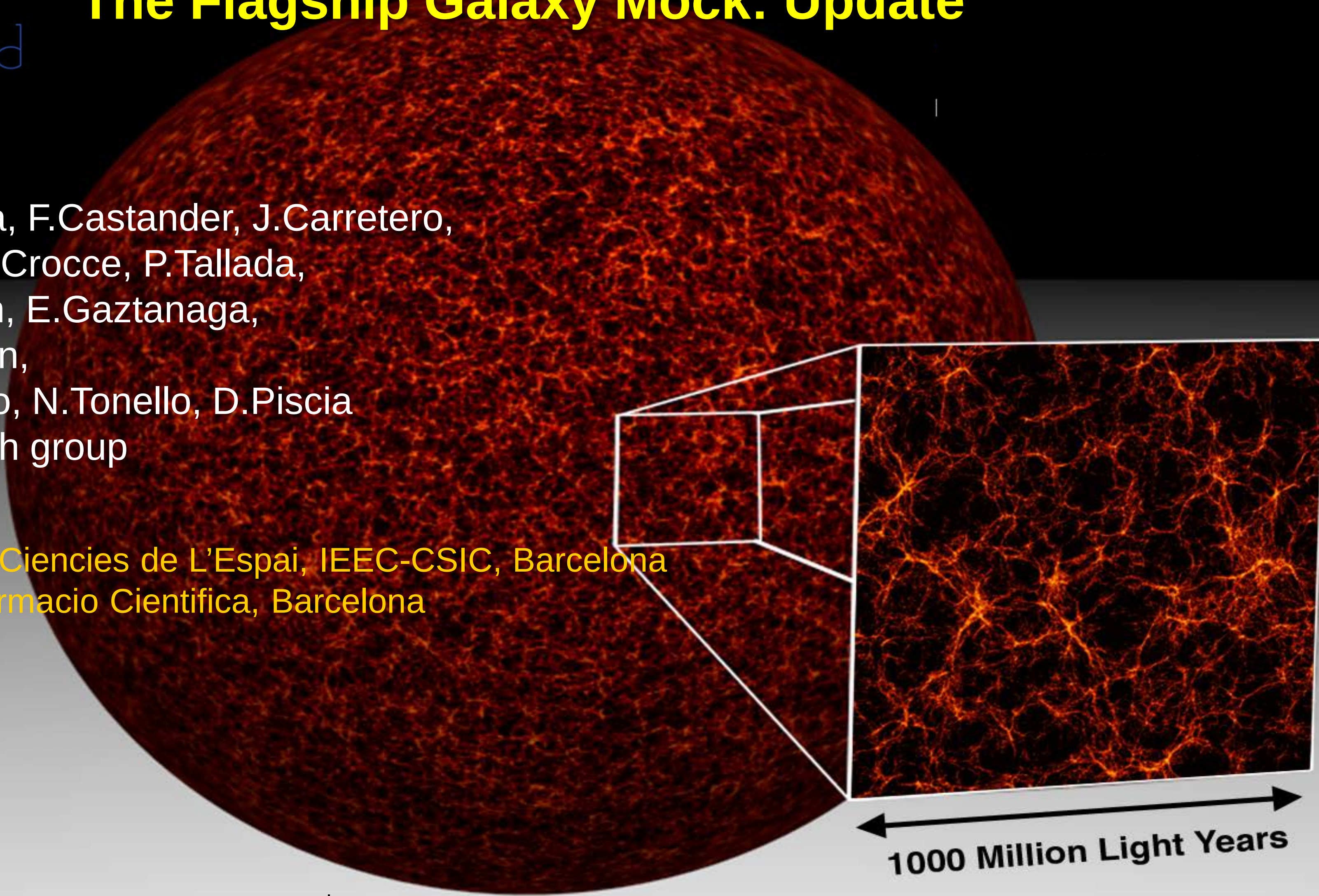



The Light Cone
10 trillion particles
50 billion halos
Rockstar halo catalogue
computed for the entire
sphere of particles.
z=2.3 (~10 Gyrs ago)

The Flagship Galaxy Mock: Update

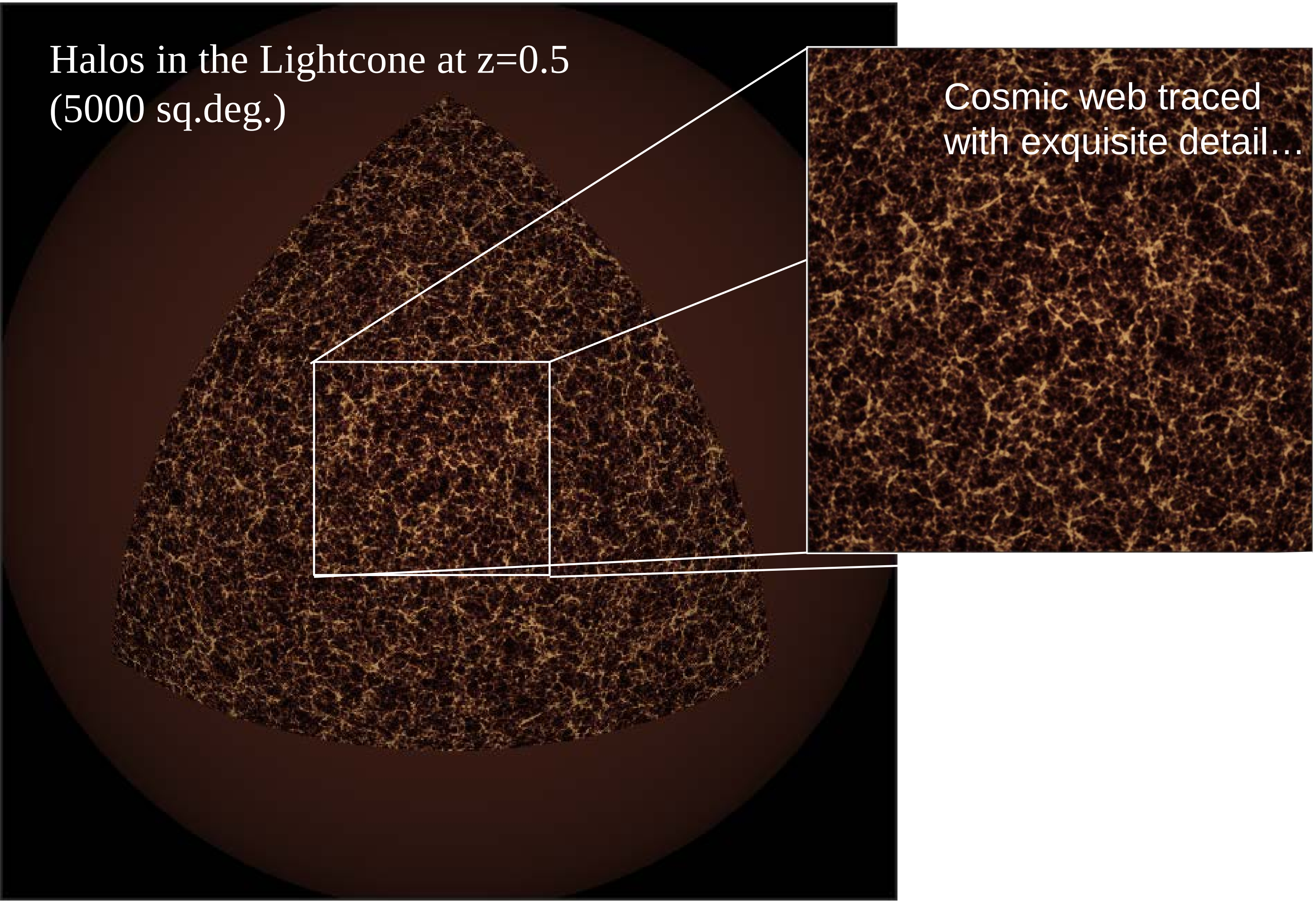
P.Fosalba, F.Castander, J.Carretero,
L.Blot, M.Crocce, P.Tallada,
A.Alarcon, E.Gaztanaga,
K.Hoffman,
S.Serrano, N.Tonello, D.Piscia
& U.Zurich group

Institut de Ciencies de L'Espai, IEEC-CSIC, Barcelona
Port d'Informacio Cientifica, Barcelona



SPV Kick-off meeting, IAP 25th Jan 2017

Validation: Halos



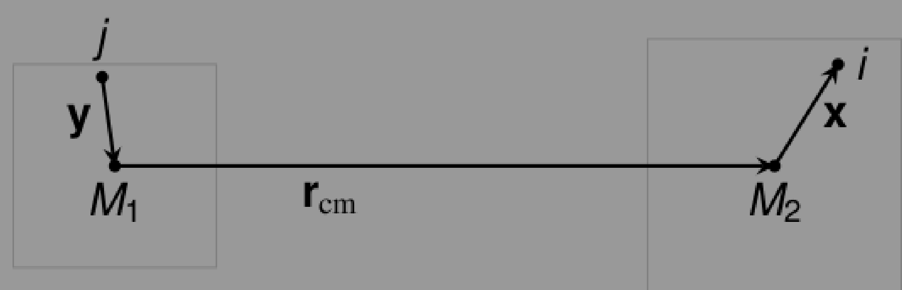
The pkdgrav3 N-Body Code

1. Fast Multipole Method, O(N), 5^{th} order in Φ
2. GPU Acceleration (Hybrid Computing)
3. Hierarchical Block Time-Stepping
4. Dual tree gravity calculation for very active particles
5. Very efficient memory usage per particle
6. On-the-fly analysis
7. Asynchronous direct I/O for checkpoints, the light cone data and halo catalogs.
8. Available on www.pkdgrav.org (bitbucket.org)

pkgdrav3 and Fast Multipole

Quick explanation of FMM

O($\mathbf{10^6}$) particles   **O**($\mathbf{10^6}$) particles

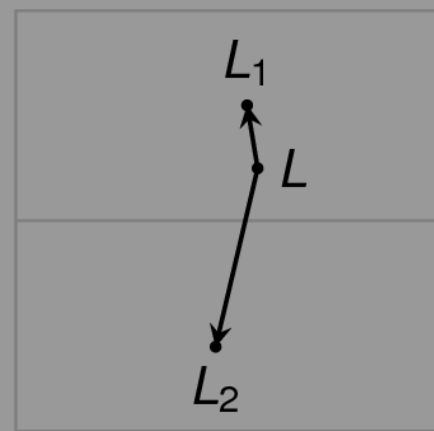


Direct **O**($\mathbf{10^{12}}$) interactions to calculate! $O(N^2)$ code.

Tree Use a multipole approximation for the mass at M_2 to calculate the force at each j: **O**($\mathbf{10^6}$) interactions to calculate. $O(N \log N)$ code.

FMM Use a multipole approx for the mass at M_2 to approximate the "potential landscape" at M_1 (n^{th} order gradients of the potential): **O**(**1**) interaction to calculate. $O(N)$ code!

Quick explanation of FMM



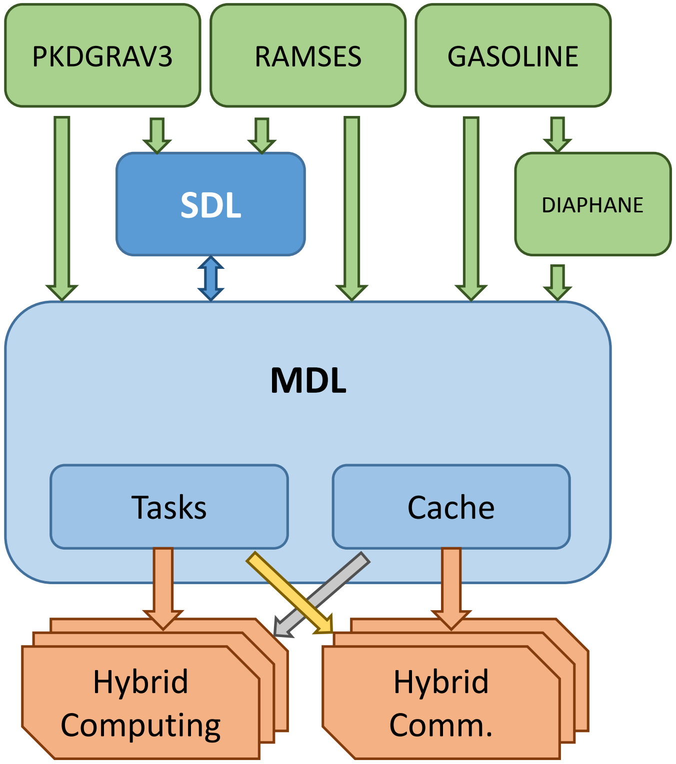
- We always open the "larger" of a pair of cells if they are too close!
- When we open the cell we shift the current local expansion to the child cells.
- When we reach the leaf cell, we can apply the local expansion (1 interaction) to each particle plus evaluate all remaining Particle-Particle interactions, typically O(500).
- This takes O(500 N), and O(N) shifts and O(N) L(M)-calculations.

Data Locality in pkdgrav3

- Note that as we proceed deeper in the tree, the data we need to fetch becomes ever more local! As long as data is stored in a kind of "tree order".
- This is what makes FMM very efficient on systems with many cache levels in the memory hierarchy (slowest: off-node mem).
- FMM algorithm achieves a kind of minimal amount of data movement within the entire computing architecture.
- The periodic BCs are handled by multipole Ewald summation technique. Instead of 4 transposes, 3 FFTs, 3 IFFTs, a single independent (GPU) calculation for each particle is done.

Piz Daint – over 5000 GPU Nodes

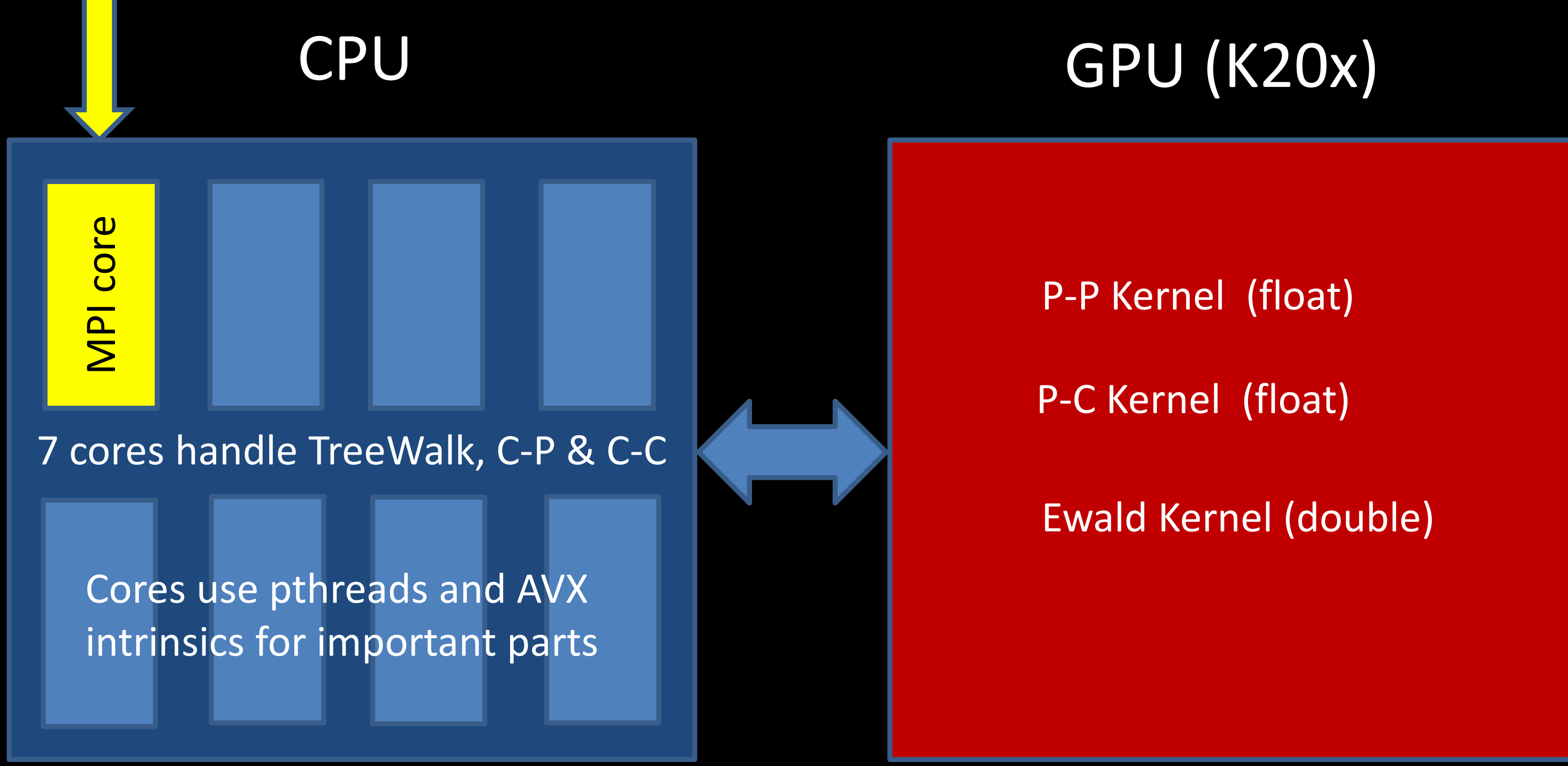
6^{th} Fastest Computer in the World. Upgrade to Haswell & P100 (now)…


PKDGRAV3
RAMSES
GASOLINE
SDL
DIAPHANE
MDL
Tasks
Cache
Hybrid
Computing
Hybrid
Comm.

GPU Hybrid Computing

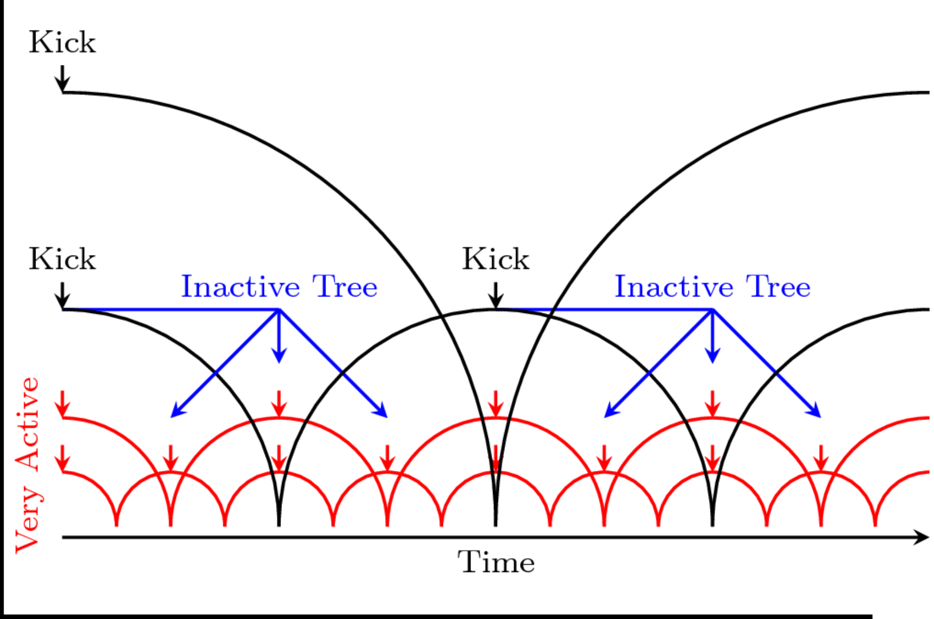
Piz Daint example

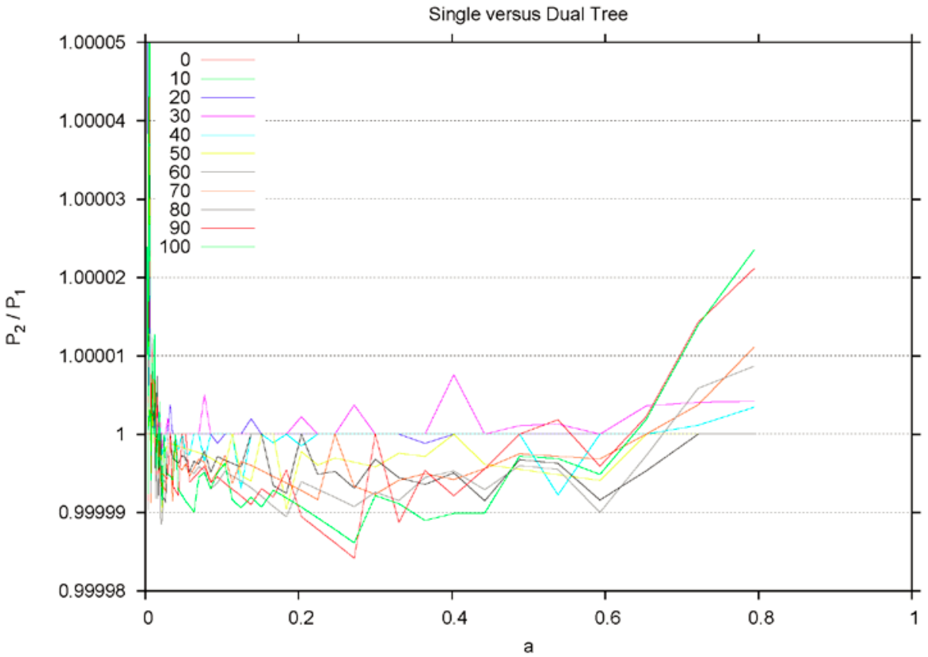
We don't directly use MPI or pthreads in the main code, but our own MDL library which implements a software cache to access data on remote nodes or threads.



Dual Tree Implementation

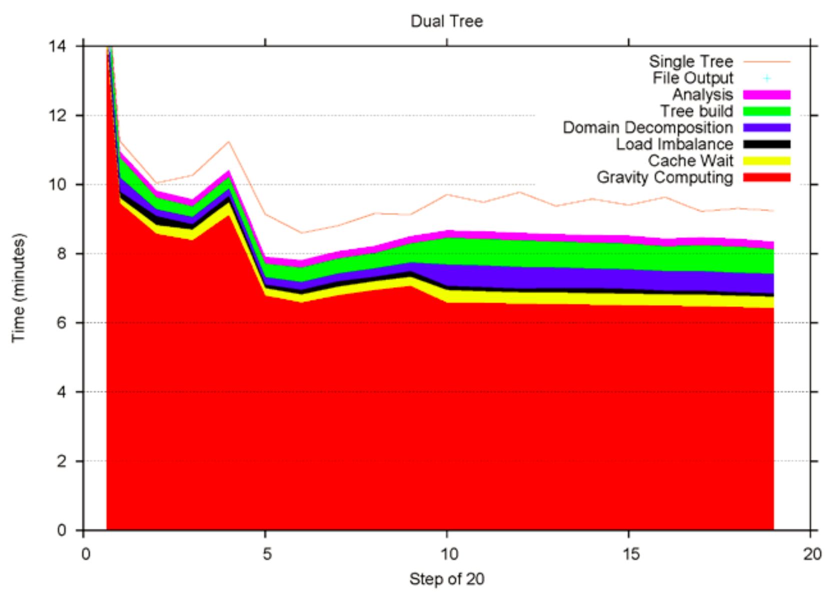
- We create a fixed tree of all particles which are on longer timescales.
- This *fixed* inactive tree is built **time centered** for the very active time-steps.
- Both trees are walked to obtain the force.
- Typically we define the very active rungs as <5% of the most active particles.



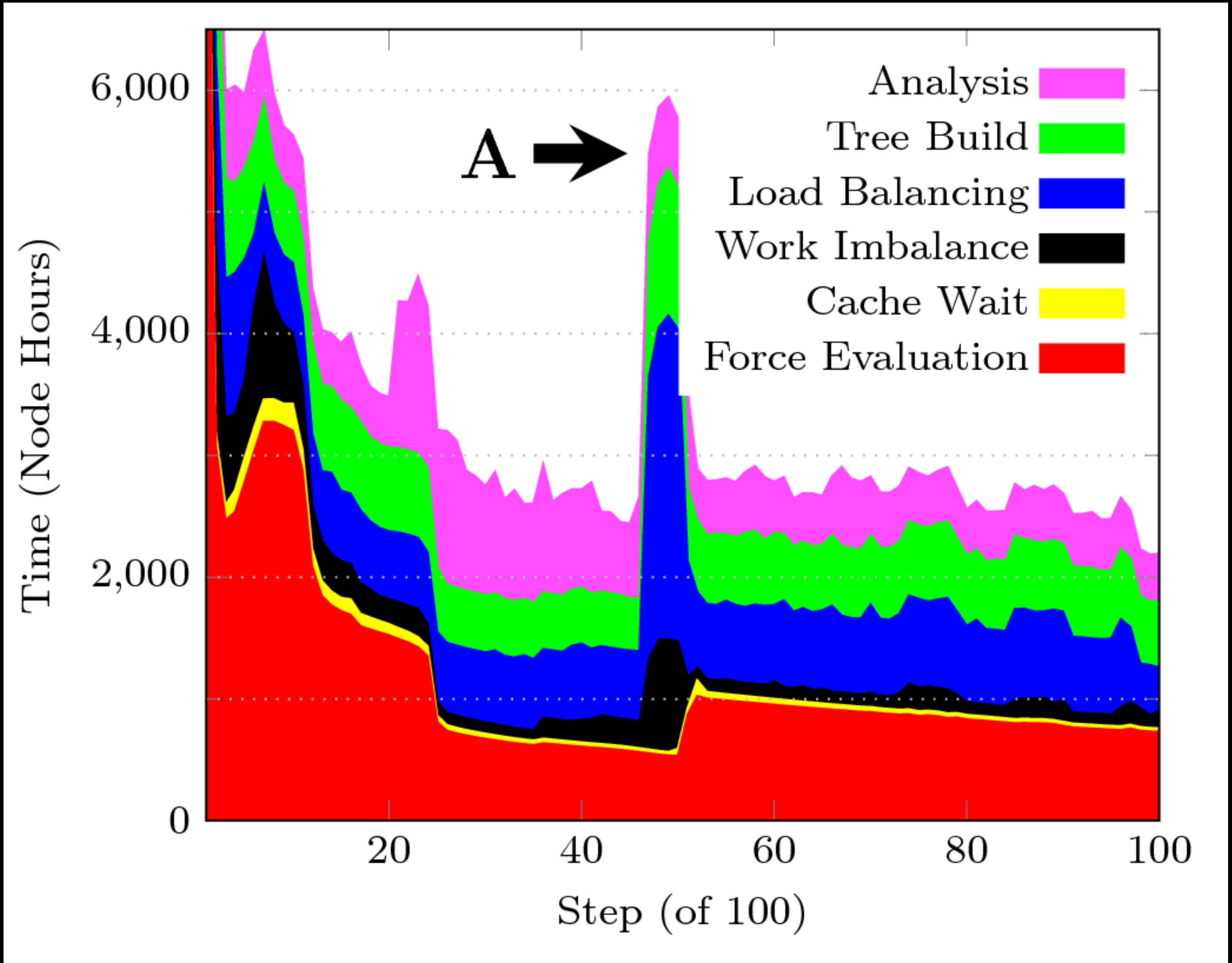


Dual Tree Performance

Note: without GPU here!



Profile of the 2 trillion particle production simulation (Piz Daint)



Memory Usage in pkdgrav3

0.5 billion particles can fit on a 32 Gbyte Node like Piz Daint

28 bytes persistent

6 bits: old rung 24: group id	
pos[0]	int32_t
pos[1]	int32_t
pos[2]	int32_t
vel[0]	float
vel[1]	float
vel[2]	float

<28 bytes / particle

Tree Cells
Binary Tree

4th order
Multipoles
(float prec)

~5 bytes / particle

Cache/Buffers

0-8 bytes ephemeral

Group finding
Other analysis

ClAoS is used for the particle and cell memory which makes moving particles around simple

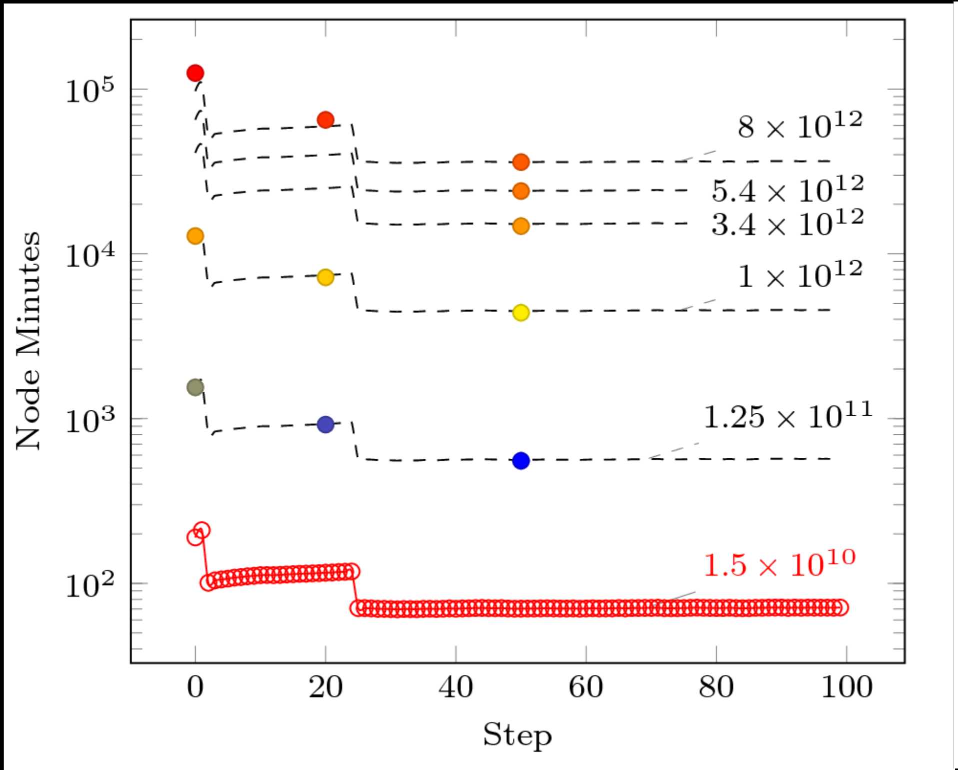
AoSoA is used for all interaction lists which are built by the TreeWalk algorithm.

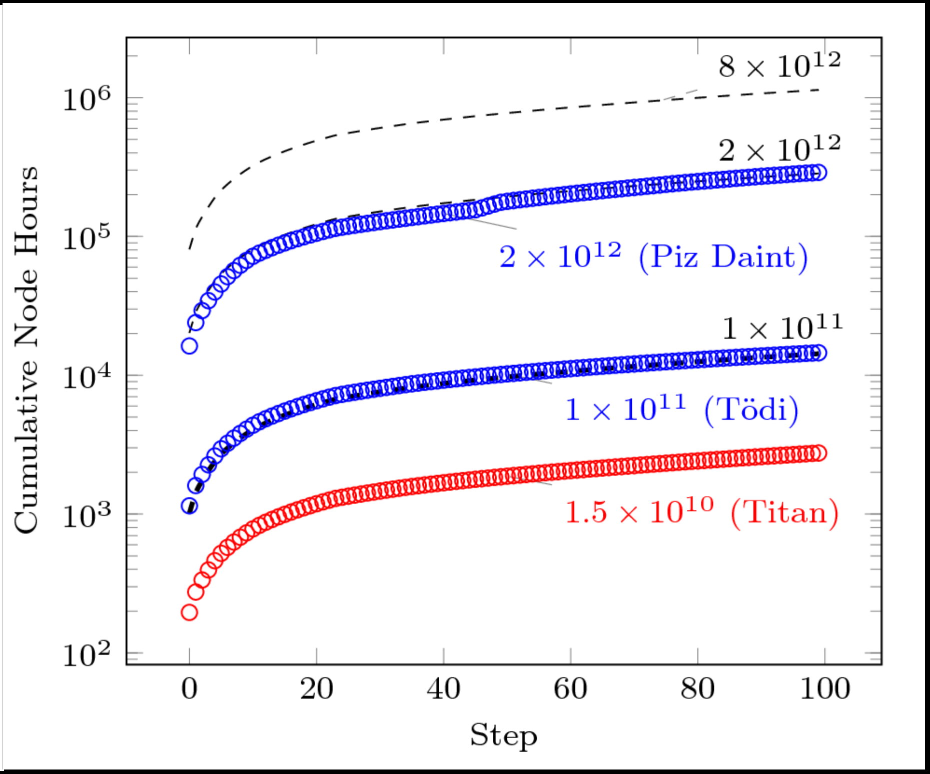
Reducing memory usage increases the capability of existing machines, but also increases performance somewhat. Simulations are limited more by memory footprint.

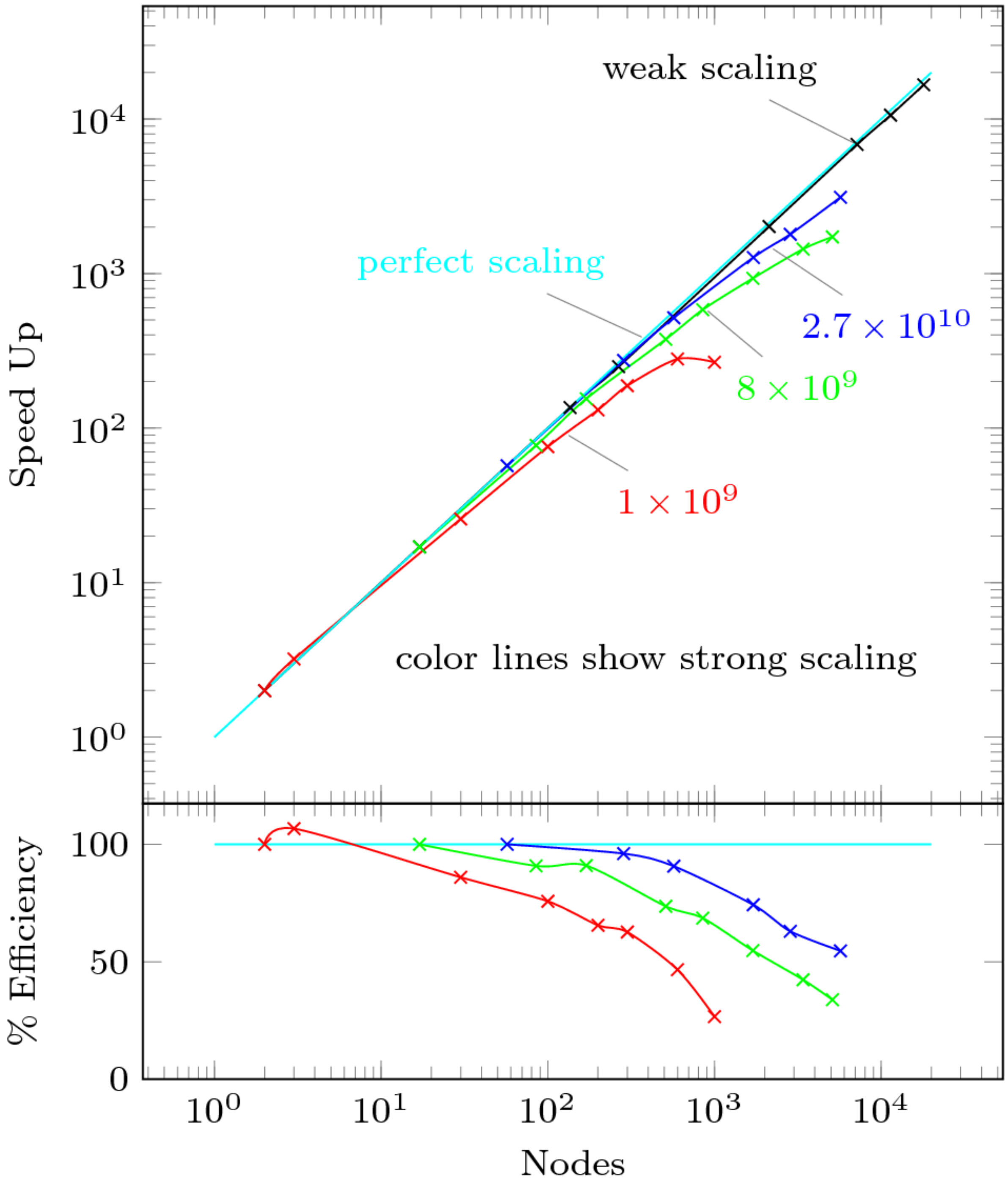
Benchmarking on Titan and Piz Daint

Nearly Perfect Weak Scaling makes performance prediction very accurate for these simulations.
120 seconds for an all N gravity solve!

We show that it is quite feasible to run 8 trillion particles on Titan with a little over 1 million node hours. **10 PFlops**

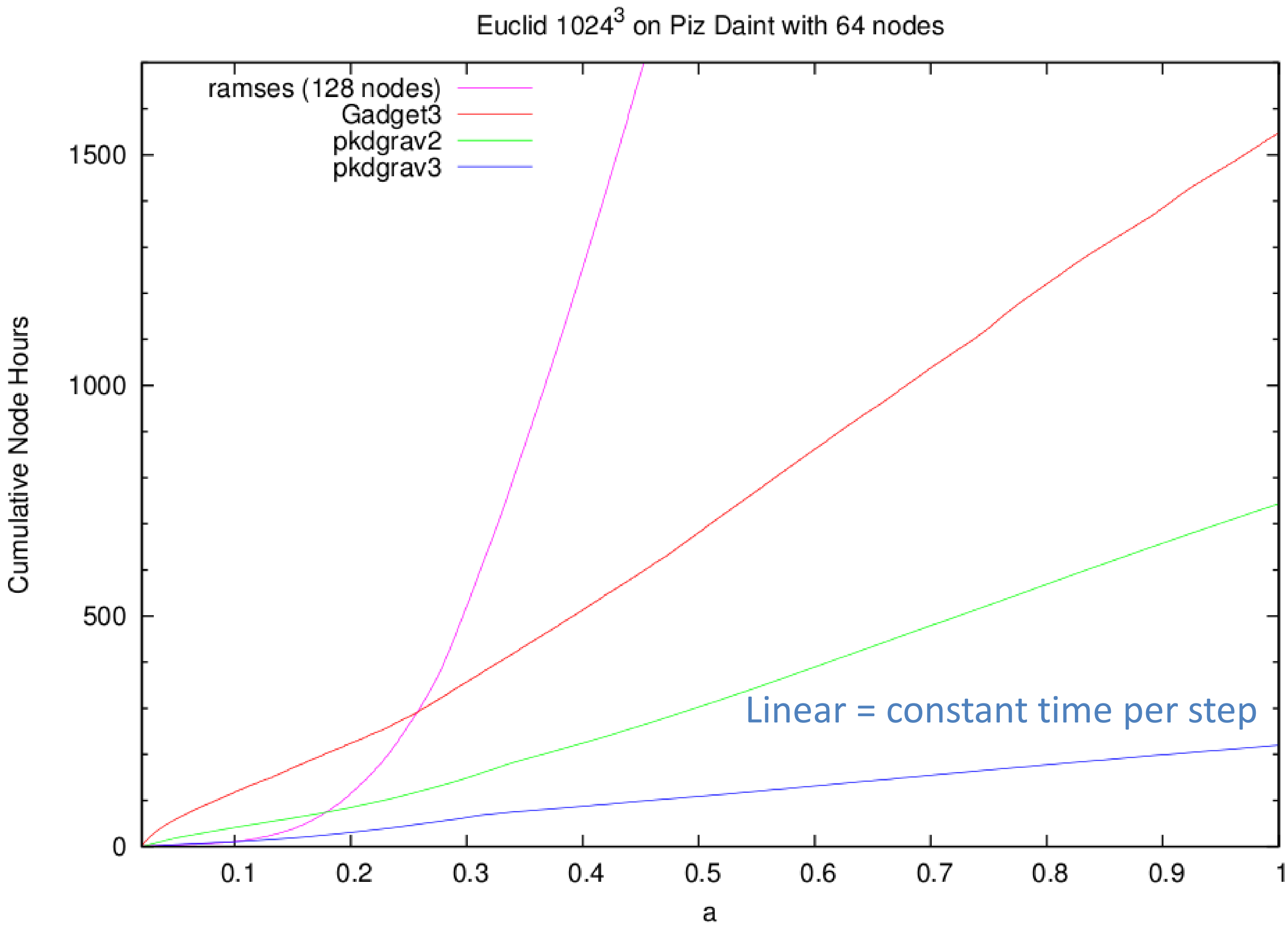




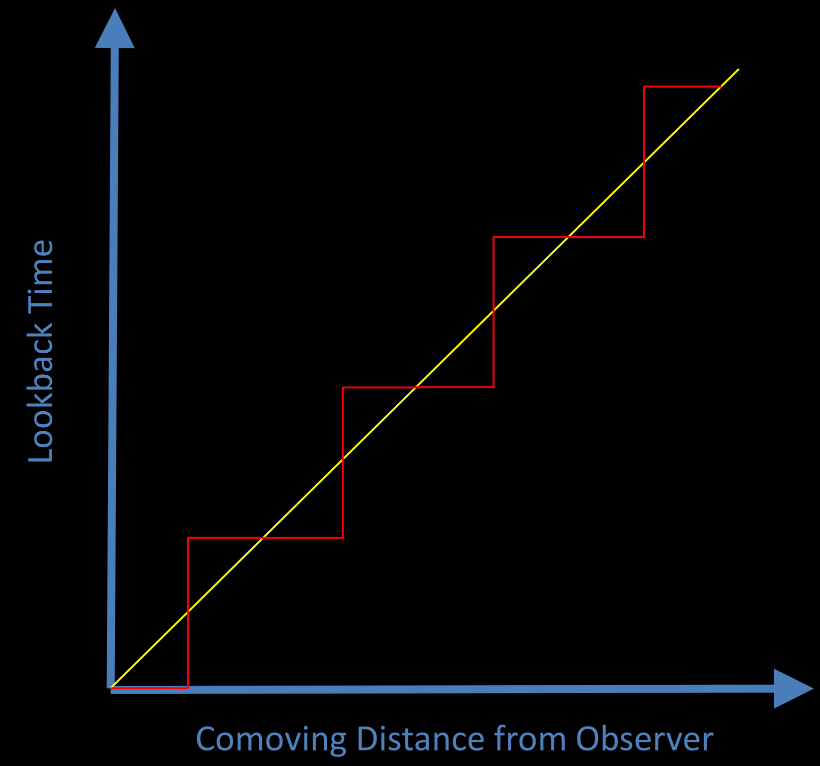


weak scaling
perfect scaling
2.7×10^{10}
8×10^9
1×10^9
color lines show strong scaling
Speed Up
% Efficiency
Nodes
10^0
10^1
10^2
10^3
10^4
0
50
100

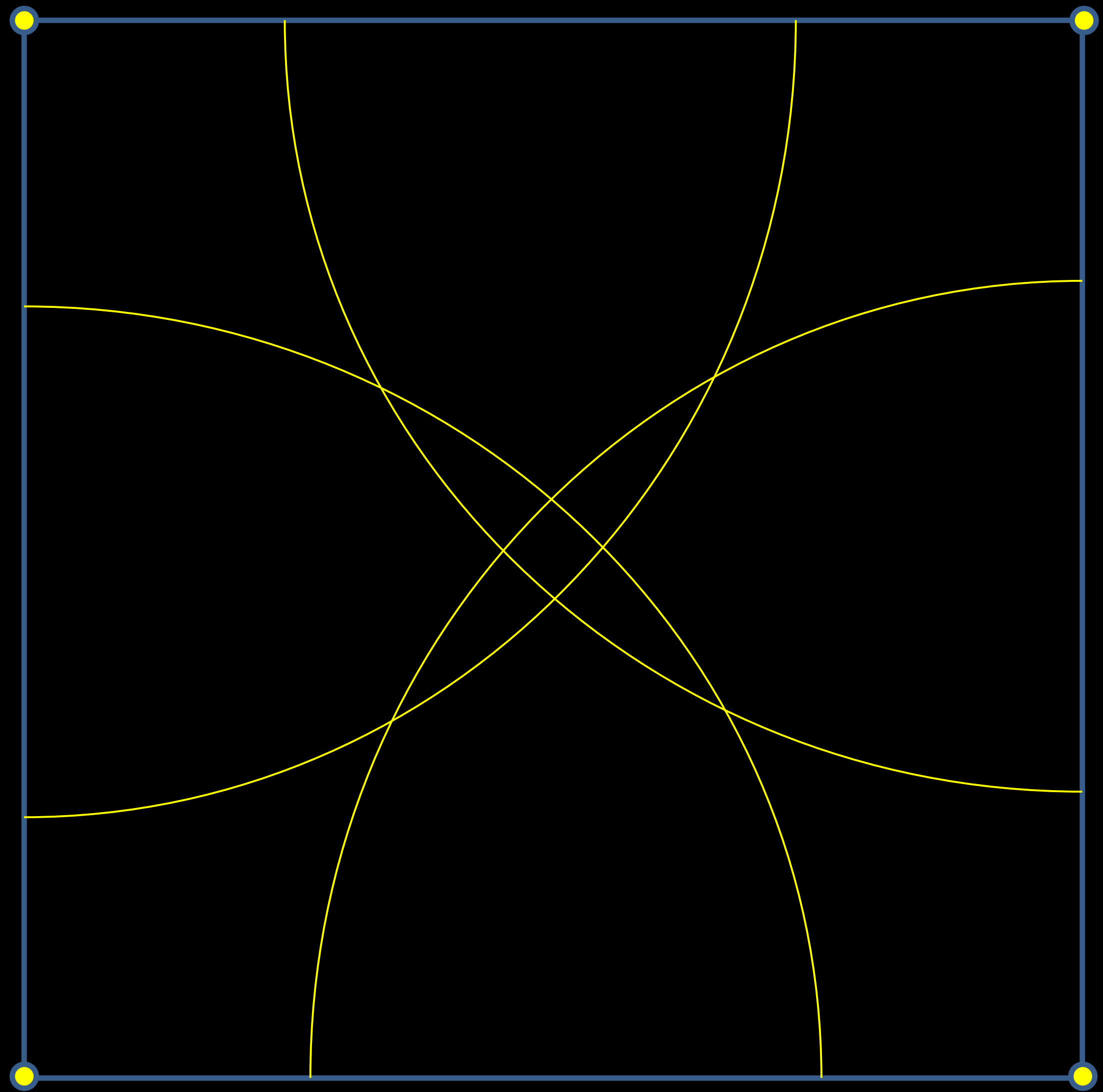
Speed of the Codes



Produce the ideal of a light cone without discontinuous jumps in the observed structure.



Continuous Light Cone Generation

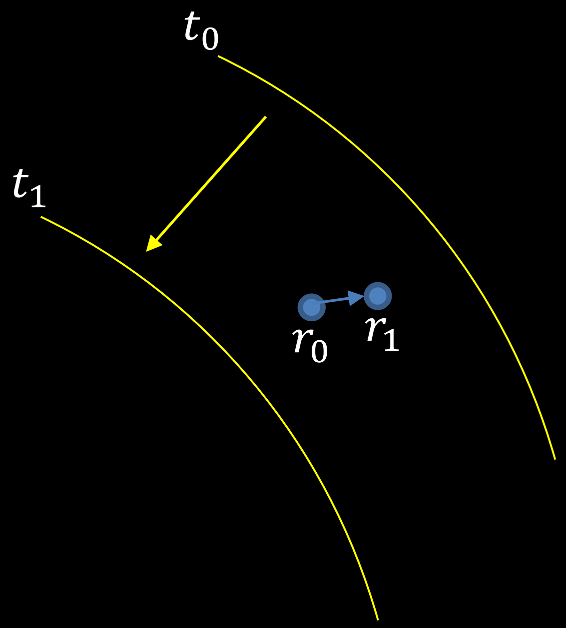


Solve for the precise time at which the light surface intersects a given particle, even when this particle is between 2 of the smallest sub-steps in the simulation.

Speed this test up by considering each of the 8 corners of the light cone using AVX vector operations.

For 2 replicas there are 64 tests and for 3 replicas there are 184 tests.

$$(1-x) * r_0 + x * r_1 = (1-x) * c\, t_0 + x * c\, t_1$$



Solve for x. Where x >= 0 and x < 1!

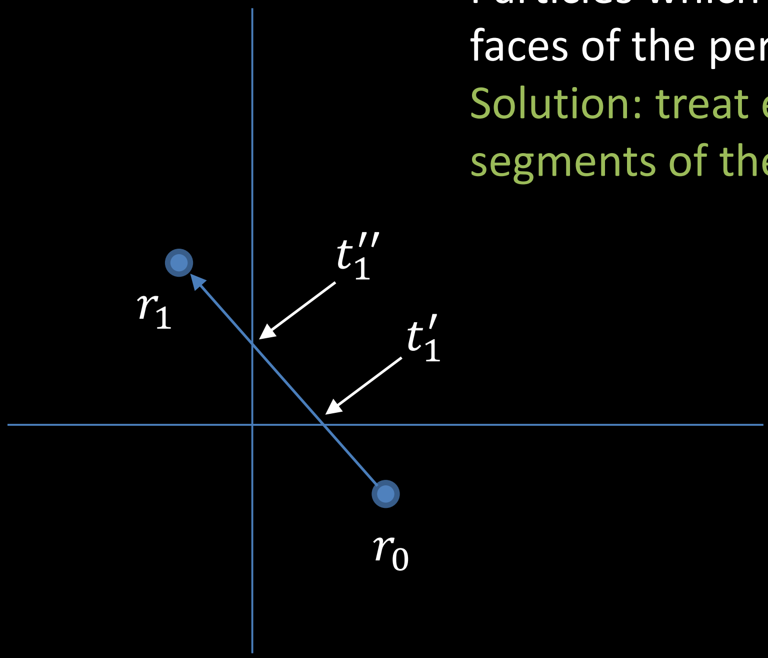
Assuming that the radial position of the particle changes linearly with time. This is a good approximation for the small sub-steps where the particle's position actually changes linearly with time.

Output: $\boldsymbol{r}_{LC} = (1-x) * \boldsymbol{r}_0 + x * \boldsymbol{r}_1$

$$v_{LC} = (1-x) * v_0 + x * v_1$$

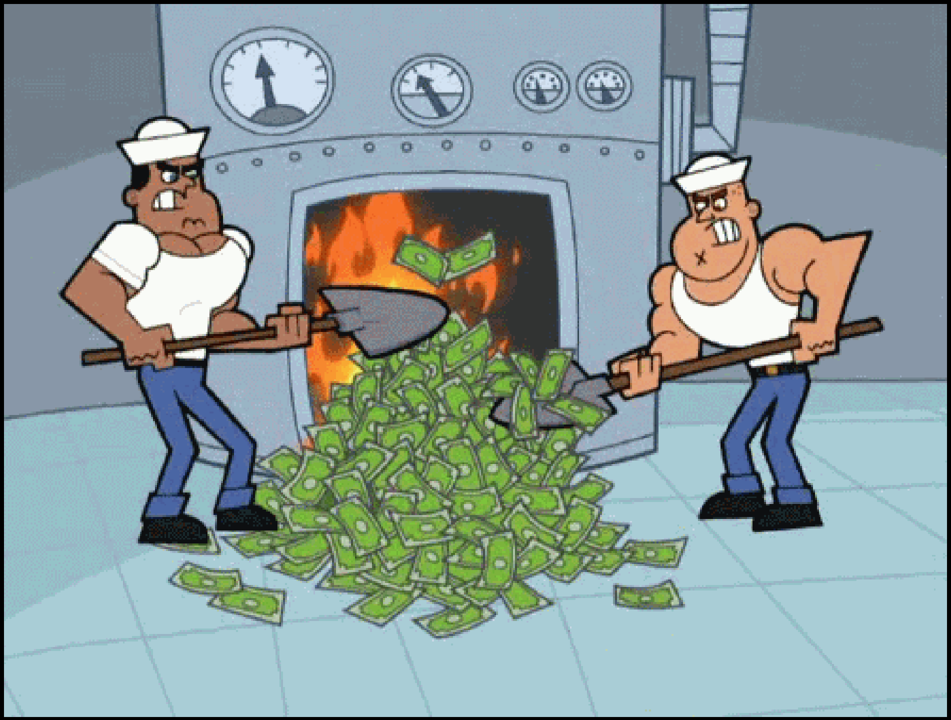
Where the interpolation is on the actual positions and velocities.
Note: this output can be done completely asynchronous to the simulation!

A Slight Complication…



Particles which cross one or more (max 3) faces of the periodic simulation volume.
Solution: treat each of the (max 4) segments of the sub-step in order.

Dollars and $ense



Euclid 2 trillion particle simulation

$\sim$ 80 hours

$\times$ 4000+ nodes

$\approx$ 350,000 node hours

$\times$ CHF 0.50 per node hour

$\approx$ **CHF 175,000**

Conclusion

- Predictability
- The Euclid Flagship Light-cone and Mock (March)
- Pkdgrav3: Fast Multipole Method (> 5x)
- Hybrid Computing (4 x)
- Dual Tree Multistepping (5x & 1.5 x)
- Performance and Scaling (Predictable run-time)
- Light-cone Algorithm (Continuous! Halos?)
- Dollars and $ense (Investment in code! Commodity Hardware?)