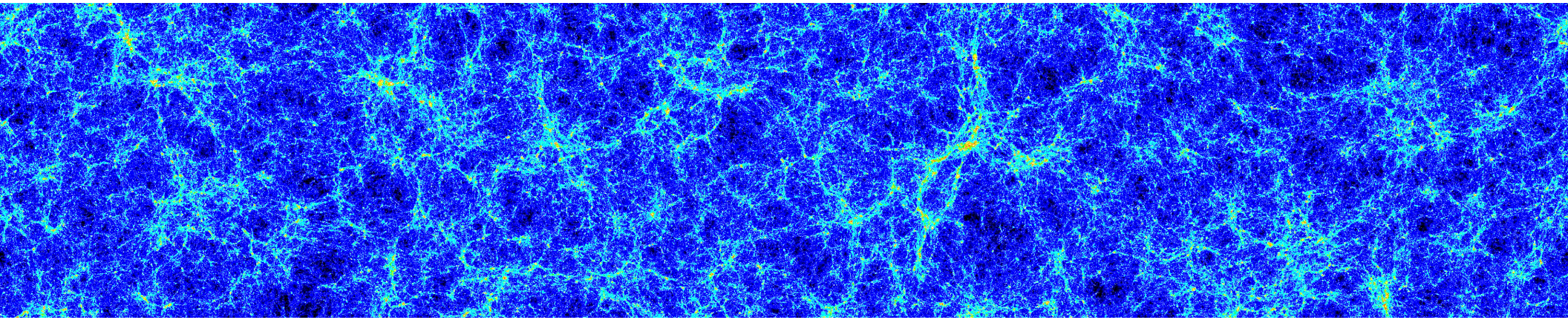
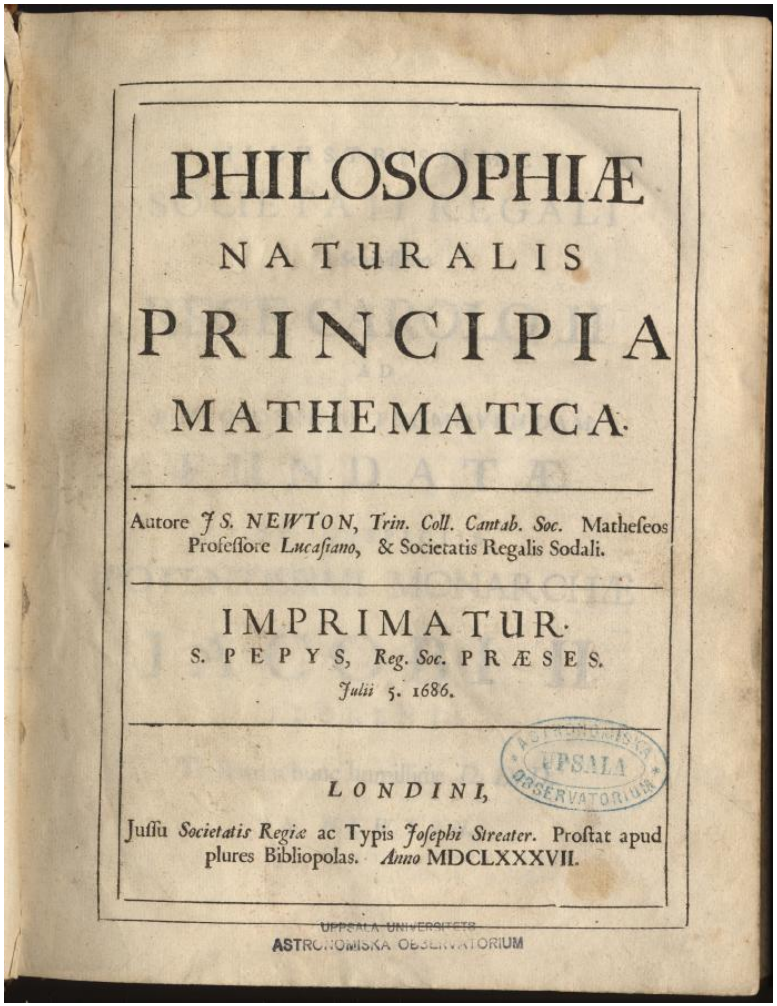
Cosmological Simulations: Approaching Exascale



Michael S. Warren
Theoretical Division
Los Alamos National Laboratory
msw@lanl.gov

Los Alamos
NATIONAL LABORATORY
EST.1943

The System of the World



PHILOSOPHIÆ
NATURALIS
PRINCIPIA
MATHEMATICA.

Autore *J S. NEWTON, Trin. Coll. Cantab. Soc.* Matheſeos Profeſſore *Lucaſiano*, & Societatis Regalis Sodali.

IMPRIMATUR.
S. PEPYS, *Reg. Soc.* PRÆSES.
Julii 5. 1686.

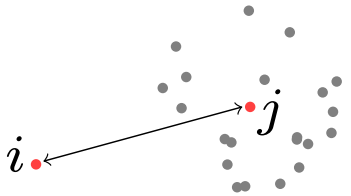
LONDINI,
Juſſu *Societatis Regiæ* ac Typis *Joſephi Streater.* Proſtat apud plures Bibliopolas. *Anno* MDCLXXXVII.

$$\sum \mathbf{F} = 0 \implies \dot{\mathbf{r}} = 0$$

$$\mathbf{F} = m\ddot{\mathbf{r}}$$

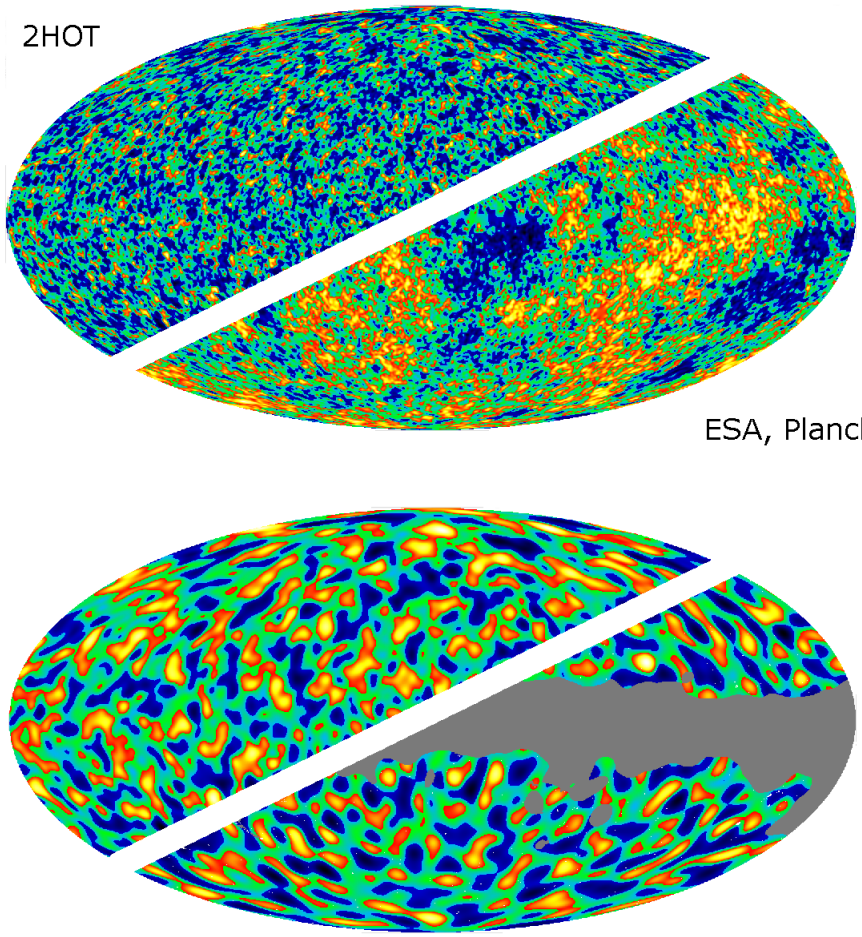
$$\mathbf{F}_1 = -\mathbf{F}_2$$

$$F = \frac{GM_1M_2}{|r|^2}$$



$$\mathbf{F}_{ij} = GM_i \sum_{j \neq i}^{N} M_j \frac{\hat{\mathbf{r}}}{|\mathbf{r}|^2}$$

Theory, Observation, Simulation



$$\Omega_m = 0.268,\ \Omega_b = 0.049,$$
$$\Lambda = 0.682,\ H_0 = 67.0,$$
$$m_\nu = 0.6,\ t_0 = 13.82$$

- Introduction
- What is Exascale?
- Algorithms
- Implementation
- Scientific Results
- Conclusion

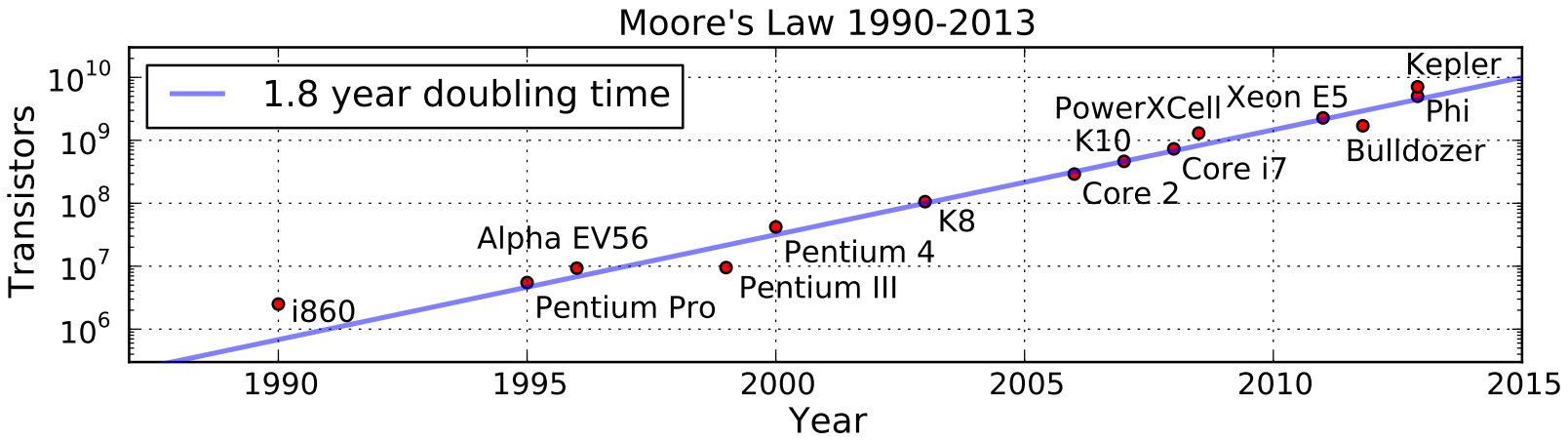
Los Alamos NATIONAL LABORATORY EST.1943

Single Processor Benchmarks

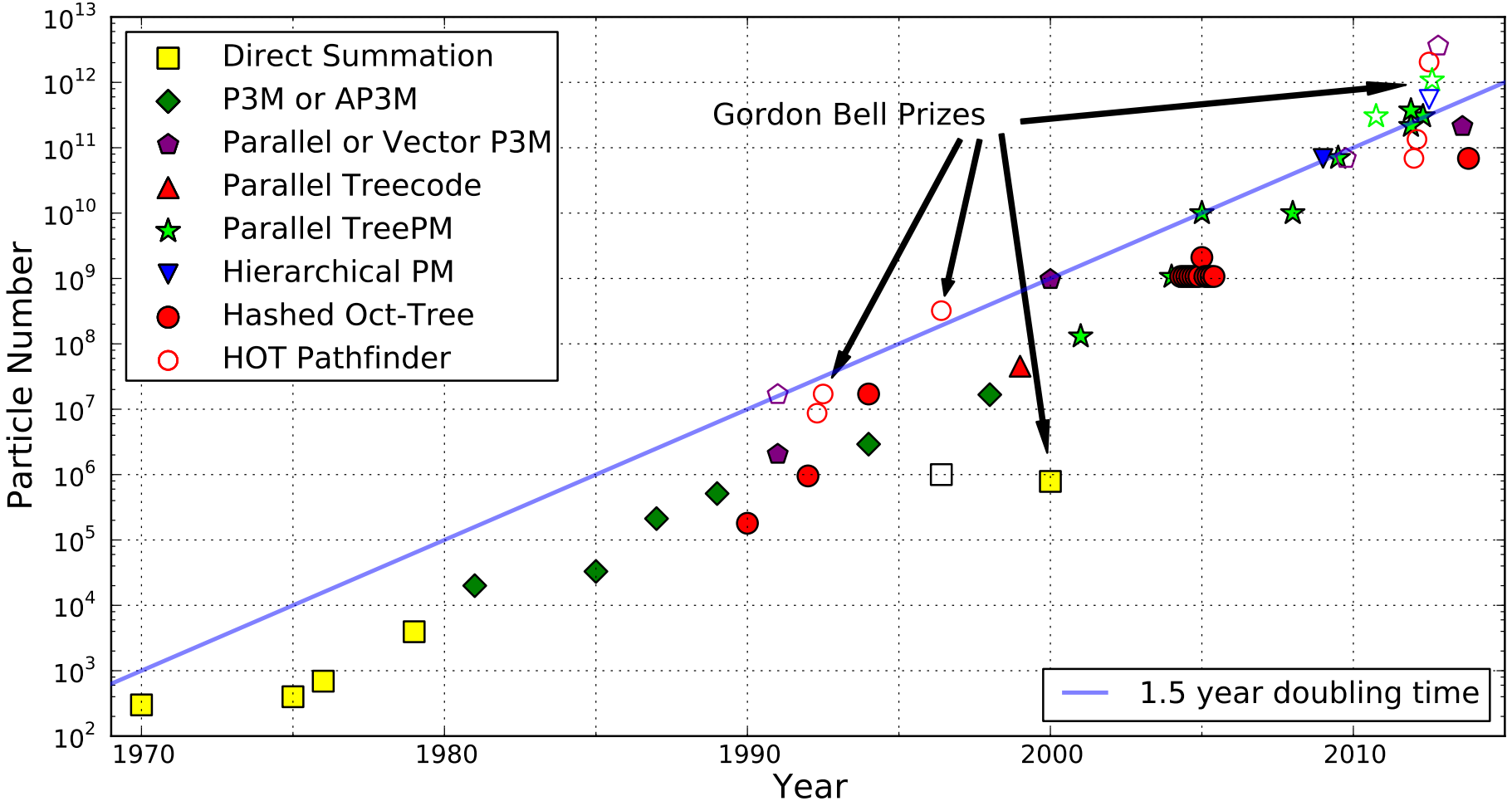
Processor	MHz	Gflop/s
Intel i860	40	0.035
Cray Y-MP	167	0.047
CM-5	32	0.050
Intel P3	500	0.186
Alpha EV56	533	0.242
Intel P4	2530	1.170
Intel P4 (SSE)	2530	6.510

Processor	MHz	Gflop/s
AMD Opteron 8435	2600	13.88
Intel Xeon E5430 (SSE2)	2660	16.34
PowerXCell 8i (1 SPE)	3200	16.36
AMD Opteron 6274	2200	16.97
Intel Xeon E5-2670 (AVX)	2600	28.41
NVIDIA M2090 (1 SM)	1300	68.56
NVIDIA K20X (1 SM)	732	149.53

Single core/SM performance in Gflop/s obtained with our inner loop benchmark. (Monopole interaction, single-precision, 28 flops per interaction.



Particle Number vs Time



The growth in scale of N-body simulations from 1970 to the present. Figure originally from Springel (2005) for a subset of the data up to 2005.

Technology and Costs

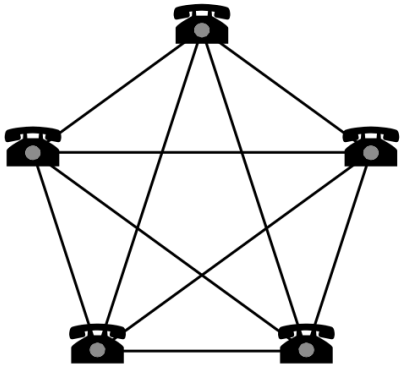
2014 Cost ($)	Description
$8.0 \cdot 10^{-18}$	GPU Peak flop
$9.4 \cdot 10^{-18}$	CPU Peak flop
$5.2 \cdot 10^{-18}$	CPU Peak flop energy
$4.2 \cdot 10^{-16}$	Memory store 1 second
$1.7 \cdot 10^{-14}$	Disk read
$1.7 \cdot 10^{-14}$	Local Network read
$4.0 \cdot 10^{-12}$	FedEx read
$1.6 \cdot 10^{-10}$	Disk store permanently
$1.3 \cdot 10^{-10}$	Wide Area Network read

Cost ($)	Description
$3.9 \cdot 10^{-9}$	SSD store permanently
$3.0 \cdot 10^{-9}$	Disk dependent read
$1.2 \cdot 10^{-9}$	Google Drive store permanently
$1.4 \cdot 10^{-8}$	Amazon EBS store
$1.2 \cdot 10^{-7}$	PTF pixel
$1.5 \cdot 10^{-6}$	WAN dependent read
$4.3 \cdot 10^{-5}$	Photon ($m_R = 20$ object)
$2.0 \cdot 10^{-3}$	Human labor for 1 second
$1.0 \cdot 10^{1}$	Cost of 1 line of software

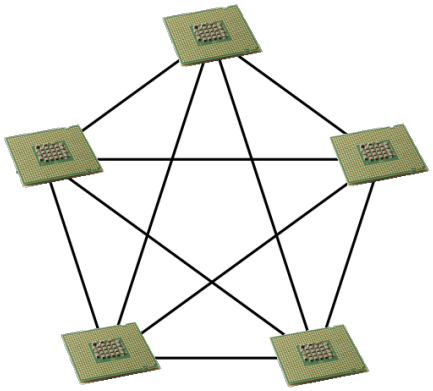
Table 1: Fundamental costs, data unit of 4-bytes

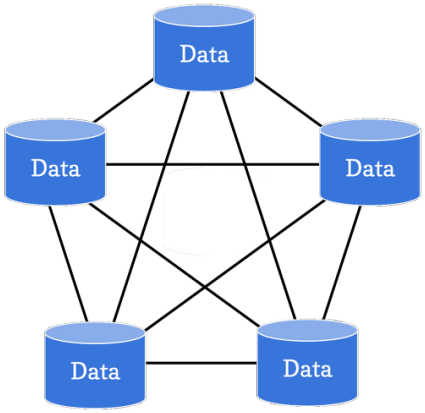
Scalable systems allow system optimization over a vast range (constrained only by cost). Scalability requires careful management of concurrency, since a serialized process can cost a factor of $10^2 - 10^6$ (e.g. reading 4-bytes from disk takes 4×10^{-8} seconds but seek latency is 10^{-1} seconds.

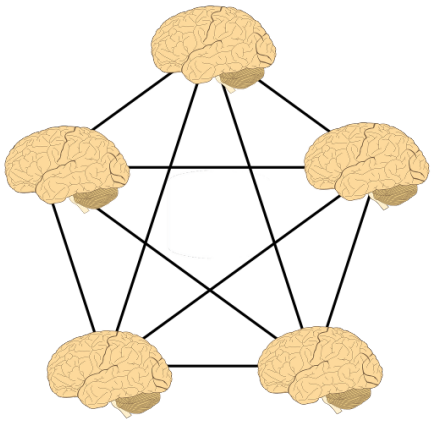
Network Effects, Interfaces and Scaling



Wikipedia: Metcalfe's Law

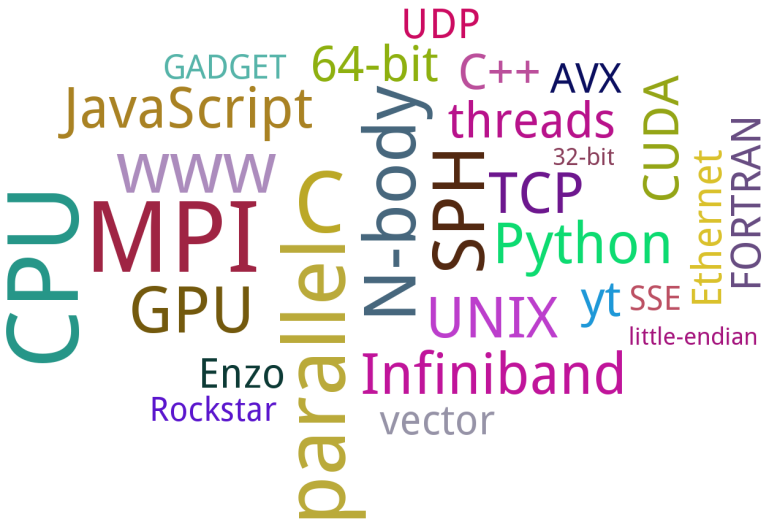






Timescale for change:

People	10,000 years
Language	100 years
SW Interfaces	40 years
Software	20 years
Hardware	5 years



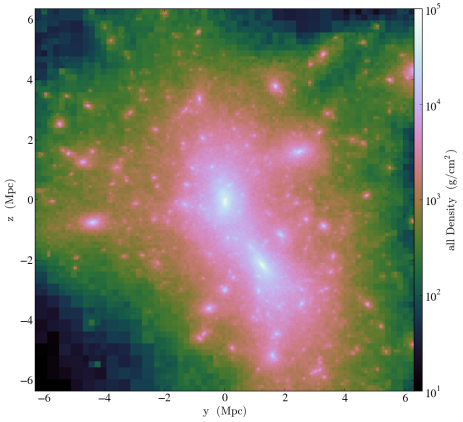
Supercomputing is Inherently Interdisciplinary

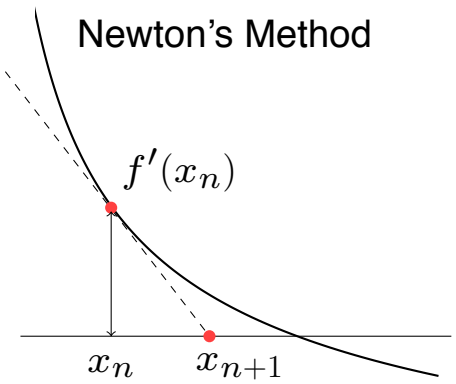
Scientist Understand the System of the World (Is this hypothesis correct?)

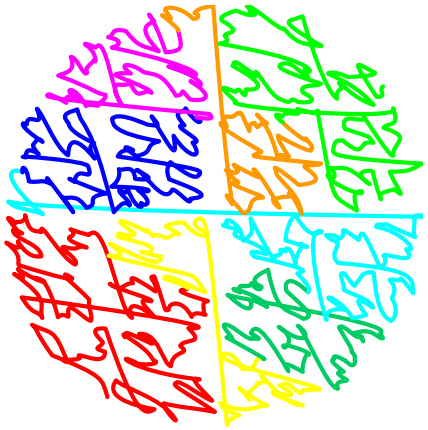
Mathematician Understand patterns (Can I prove this conjecture?)

Computer Scientist Understand the theoretical foundations of information and computation — Create complexity (Can a computer do this?)

Software Engineer Apply established knowledge to create computer programs — Manage complexity (How do I implement this algorithm?)



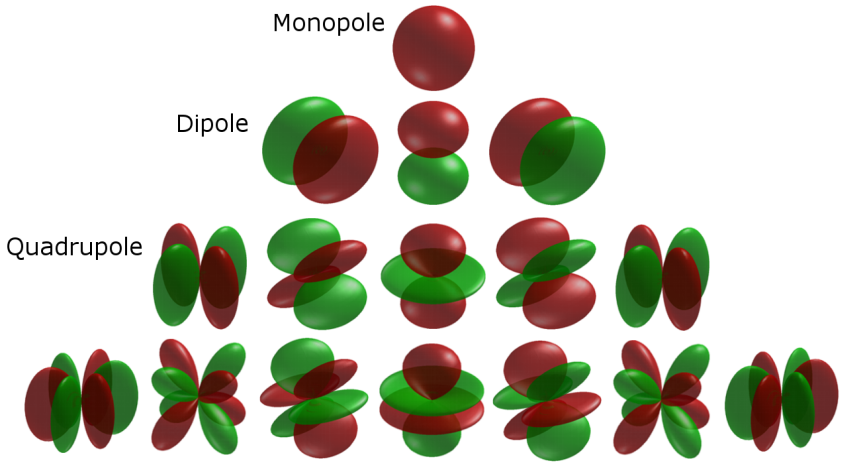




```
typedef float v8sf __attribute__ \
   ((vector_size (32)));
v8sf x = ppos0 - xp;
v8sf y = ppos1 - yp;
v8sf z = ppos2 - zp;
r2 = x*x + y*y + z*z;
rinv = t = _ia32_rsqrtps256(r2);
r2 *= rinv;  /* Newton-Raphson */
rinv *= r2;
rinv -= three;
rinv *= t;
rinv *= half;
```

Los Alamos NATIONAL LABORATORY EST.1943

"Fast" Methods: Treecodes, FMM, Fast N-body



Wikipedia: Spherical_Harmonic

- Appel (1981, 1985)
- Barnes & Hut (1986)
- Greengard & Rokhlin (1987)
- Salmon & Warren (1994)
- *Recent review,* Yokota (SC '12)

Approximate,

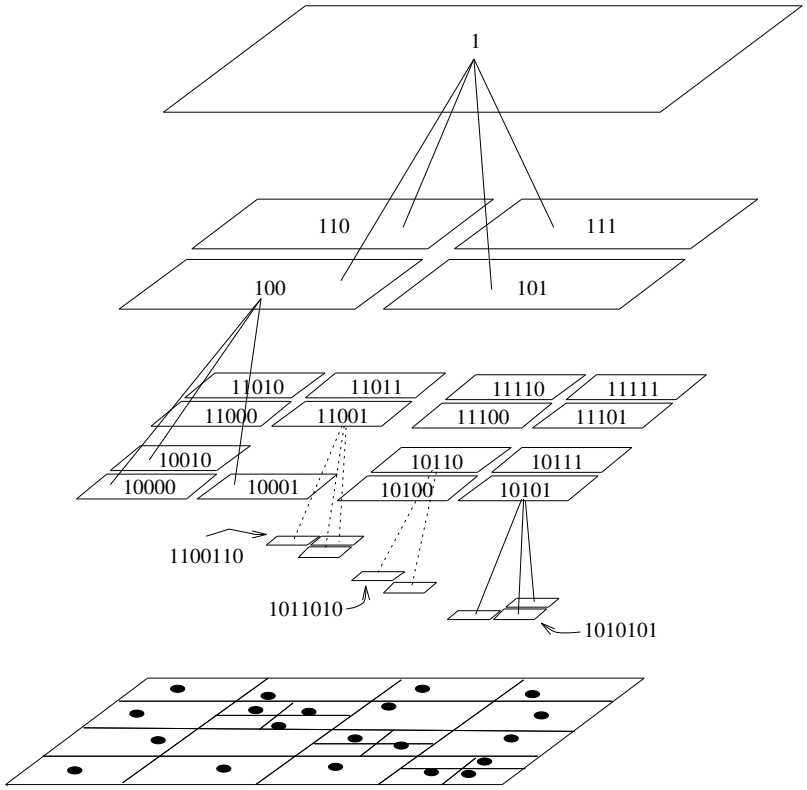
$$\mathbf{F} = \sum_i^N \sum_j^N f_{ij}(\mathbf{r})$$

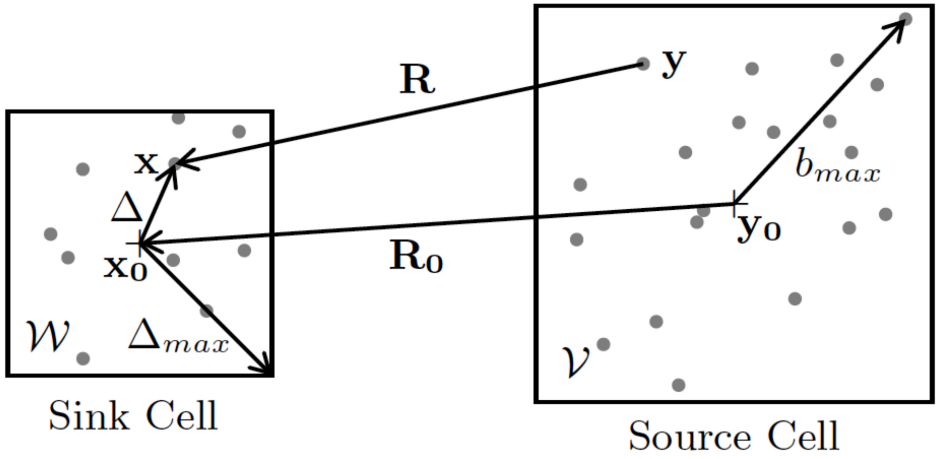
with a smaller number of terms $n\,N \ll N^2$, with controlled error.

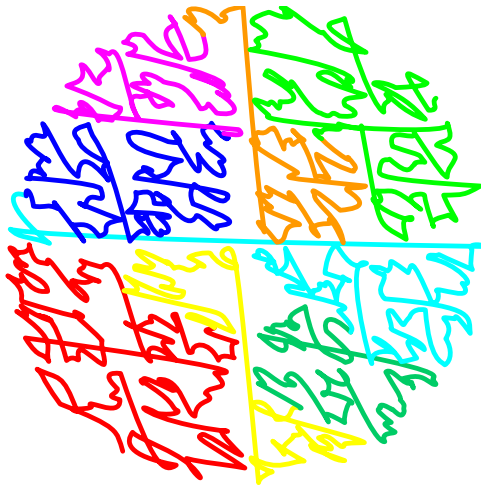
$$\mathbf{F}_i = \sum_j^n f_j(\mathbf{r}) + \sum_j^n \Delta f_j(\mathbf{r})$$

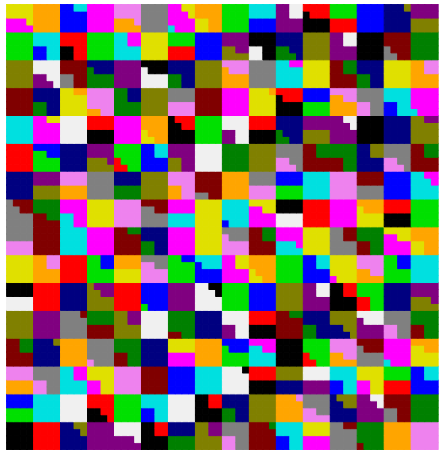
For error on $\mathbf{F}_i$ to scale as $\epsilon\sqrt{n}$, $\Delta f_j(\mathbf{r})$ must be independent and identically distributed (iid).

Hashed Oct-tree (HOT)

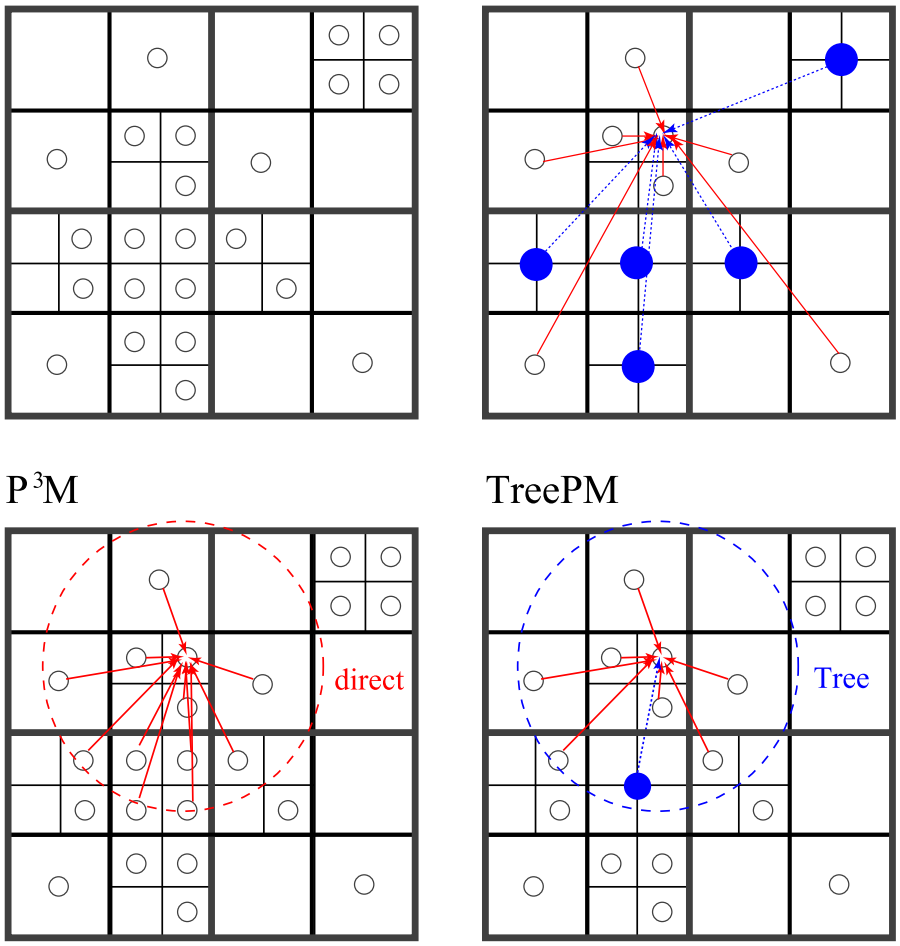




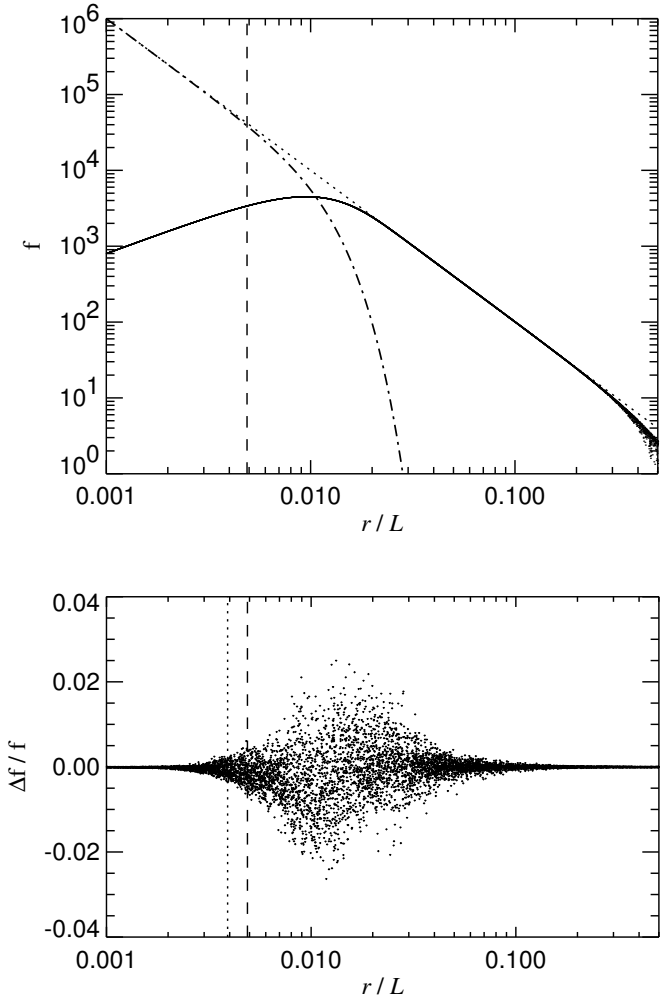




TreePM Approach

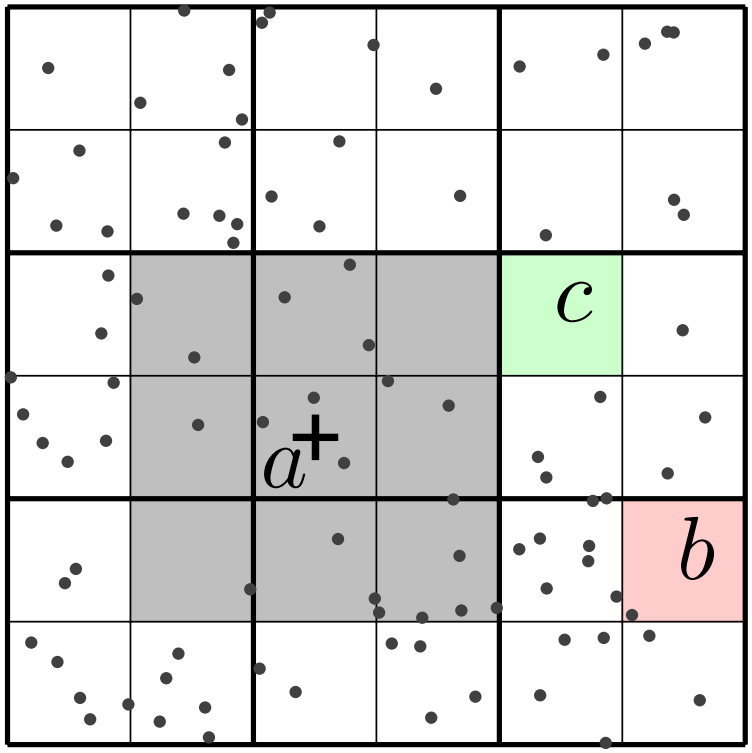


Ishiyama, Nitadori & Makino (SC '12)



Springel (2005)

Background Subtraction



$$V(x_0, y_0, z_0) = \sum_{i=0,\, j=0,\, k=0}^{1} \left(x_i y_j \operatorname{Artanh}\left(\frac{z_k}{r_{ijk}}\right) + y_j z_k \operatorname{Artanh}\left(\frac{x_i}{r_{ijk}}\right) + z_k x_i \operatorname{Artanh}\left(\frac{y_j}{r_{ijk}}\right) - \frac{x_i^2}{2} \arctan\left(\frac{y_j z_k}{x_i r_{ijk}}\right) - \frac{y_j^2}{2} \arctan\left(\frac{z_k x_i}{y_j r_{ijk}}\right) - z_k^2 \arctan\left(\frac{x_i y_j}{z_k r_{ijk}}\right)\right)$$

$$x_1 = a - x_0, \qquad y_1 = b - y_0, \qquad z_1 = c - z_0,$$

$$r_{ijk} = \sqrt{x_i^2 + y_j^2 + z_k^2}, \quad i, j, k = 0, 1.$$

Waldvogel (1976)

An illustration of background subtraction, which greatly improves the performance of the treecode algorithm for nearly uniform mass distributions (such as large-volume cosmological simulations, especially at early times).

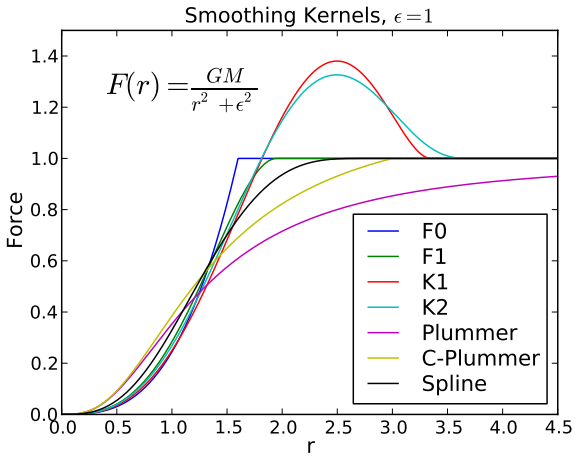
Time Stepping, Softening & Boundary Conditions

Symplectic leapfrog for cosmology.
Quinn, Katz, Stadel & Lake (1997)

$$H = \frac{\mathbf{p}'^2}{2a^2} + \frac{\phi'}{a}$$

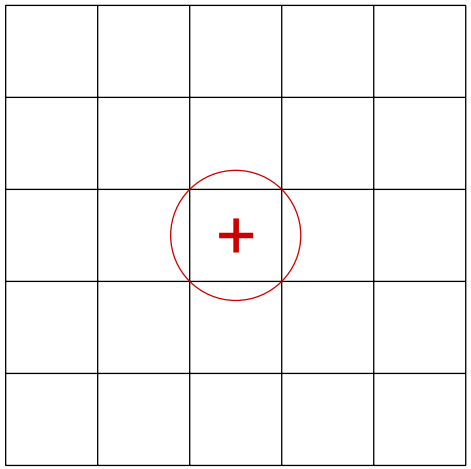
$$D(\tau) \equiv \mathbf{r}'_{t+\tau} = \mathbf{r}'_t + \mathbf{p}' \int_t^{t+\tau} \frac{dt}{a^2}$$

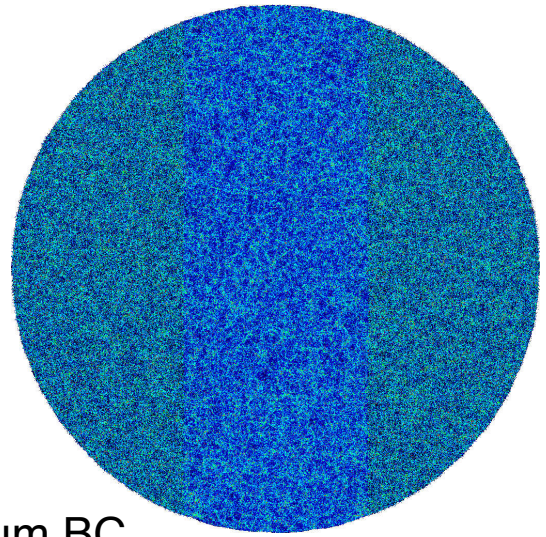
$$K(\tau) \equiv \mathbf{p}'_{t+\tau} = \mathbf{p}'_t - \nabla'\phi' \int_t^{t+\tau} \frac{dt}{a}$$



K_i kernels from Dehnen (2002)

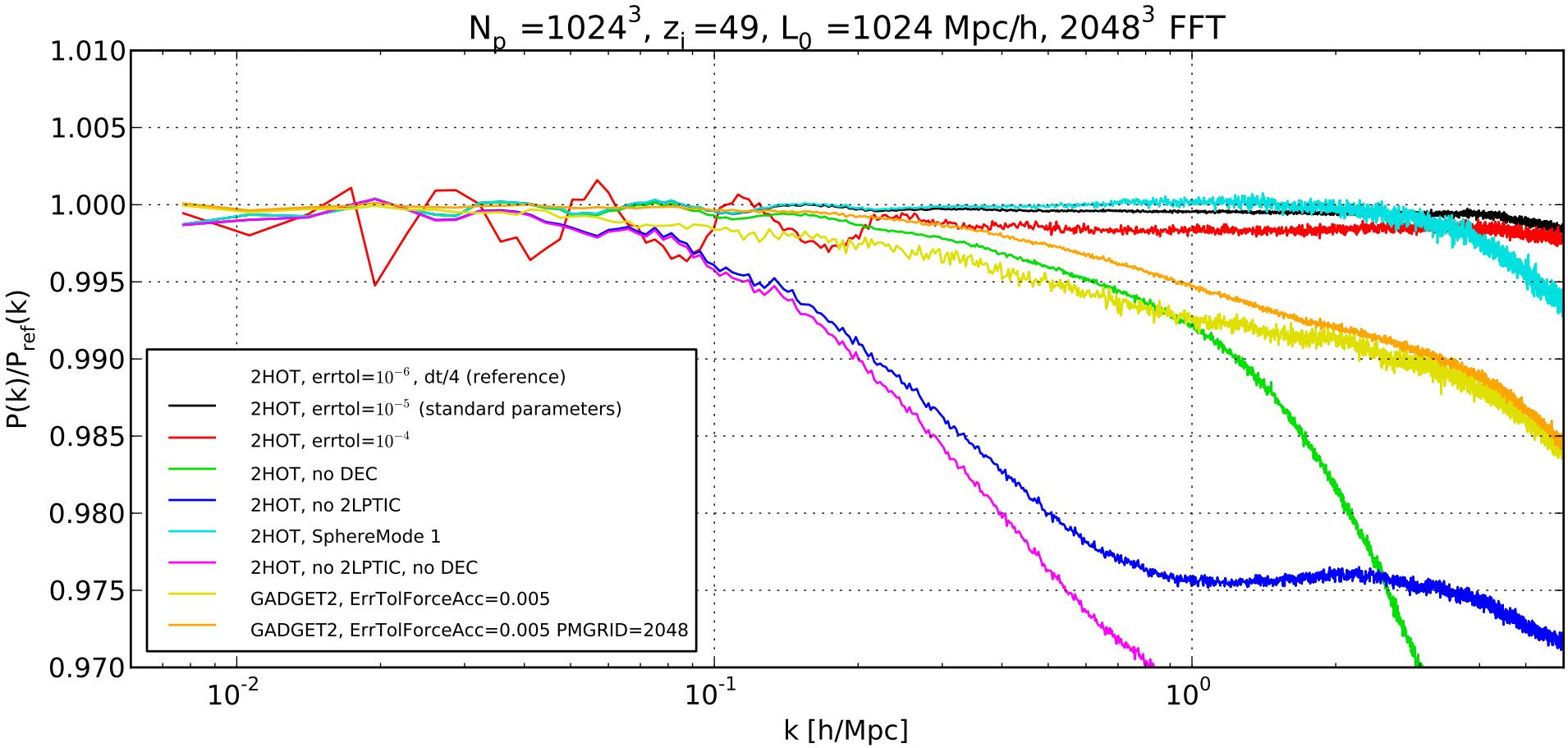
Periodic BC, Challacombe (1997)





Vacuum BC

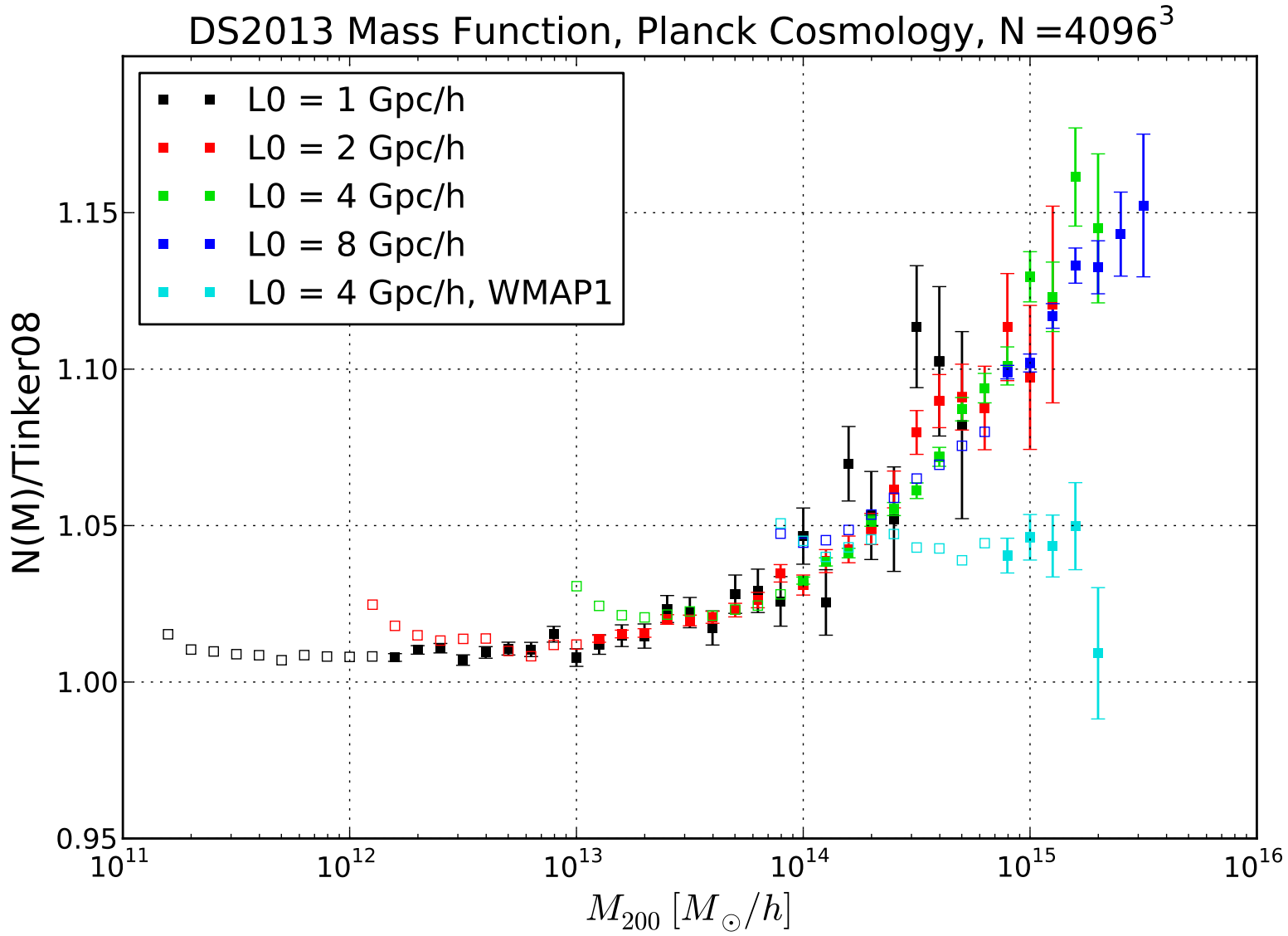
Convergence and Code Comparison



We demonstrate accuracy to 1 part in 1000 for the power spectrum.

Los Alamos NATIONAL LABORATORY EST.1943

The Mass Function of Dark Matter Halos



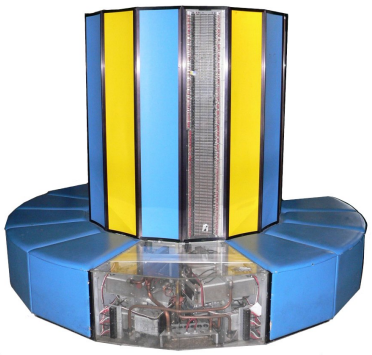
2014 INCITE Project

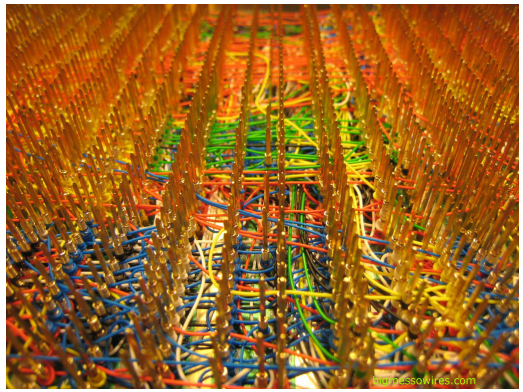
Probing Dark Matter at Extreme Scales

- Michael S. Warren, Alexander Friedland, Ben Bergen (LANL)
- Daniel Holz (University of Chicago)
- Samuel Skillman, Risa Wechsler (KIPAC, Stanford)
- Paul Sutter (Paris Institute of Astrophysics)
- Matthew Turk (Columbia University)

80M Titan processor hours, 1 Petabyte of storage

The Exascale Challenge





From the perspective of a programmer, an exascale computer is a collection of the worst ideas of the past 30 years.

- Long vectors
- Deep pipelines
- Distributed shared memory
- Hybrid architecture

Success at exascale depends much less on hardware than it does on the human interface to that hardware, a.k.a. software.

Debugging is twice as hard as writing the code in the first place. Therefore, if you write the code as cleverly as possible, you are, by definition, not smart enough to debug it.

–Brian Kernighan (1978)

The Meta-Problem: Software

Exponential growth (Moore's Law) meets arithmetic growth (human capability).

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

—Edsger Dijkstra, 1972

- Single processors are ∼1,000 times faster than 20 years ago.
- Parallel computers are ∼1,000,000 times faster than 20 years ago.
- Software has limited progress so long we don't pay attention any more.

Conclusion

- We aim to perform over 3 zettaflops of computation on Titan in 2014.
- Software is the New System of the World.
- We need more and better software, not faster more complex hardware that breaks established software interfaces.
- A machine will run Linpack at an Exaflop around 2022. When a machine runs something useful at Exascale depends on us.
- ***Exascale is People.***