"Housekeeping"

Twitter: #ACMWebinarScaling

- **Welcome** to today's ACM Learning Webinar. The presentation starts at the top of the hour and lasts 60 minutes. Slides will advance automatically throughout the event. You can resize the slide area as well as other windows by dragging the bottom right corner of the slide window, as well as move them around the screen. On the bottom panel you'll find a number of widgets, including Facebook, Twitter, and Wikipedia.

- If you are experiencing any **problems/issues**, refresh your console by pressing the **F5** key on your keyboard in Windows, **Command + R** if on a Mac, or **refresh** your browser if you're on a mobile device; or close and re-launch the presentation. You can also view the Webcast Help Guide, by clicking on the "**Help**" widget in the bottom dock.

- To control **volume**, adjust the master volume on your computer. If the volume is still too low, use headphones.

- If you think of a question during the presentation, please type it into the **Q&A box** and click on the submit button. You do not need to wait until the end of the presentation to begin submitting questions.

- At the end of the presentation, you'll see a survey open in your browser. Please take a minute to fill it out to help us improve your next webinar experience.

- You can download a copy of these slides by clicking on the **Resources** widget in the bottom dock.

- This session is being recorded and will be archived for on-demand viewing in the next 1-2 days. You will receive an automatic email notification when it is available, and check **http://learning.acm.org/** in a few days for updates. And check out **http://learning.acm.org/webinar** for archived recordings of past webcasts.

Extreme Scaling and Performance Across Diverse Architectures

HACC (Hardware/Hybrid Accelerated Cosmology Code) Framework

Salman Habib
HEP and MCS Divisions
Argonne National Laboratory

Vitali Morozov
Nicholas Frontiere
Hal Finkel
Adrian Pope
Katrin Heitmann
Kalyan Kumaran
Venkatram Vishwanath
Tom Peterka
Joe Insley
Argonne National Laboratory

David Daniel
Patricia Fasel
Los Alamos National Laboratory

Zarija Lukic
Lawrence Berkeley National Laboratory

Justin Luitjens
NVIDIA

George Zagaris
Kitware

ASCR
HEP

ACM Highlights

- **Learning Center** tools for professional development: http://learning.acm.org
  - 1,400+ trusted technical books and videos by **O'Reilly, Morgan Kaufmann**, etc.
  - Online training toward top vendor certifications (CEH, Cisco, CISSP, CompTIA, PMI, etc)
  - Learning Webinars from thought leaders and top practitioner
  - ACM Tech Packs (annotated bibliographies compiled by subject experts
  - Podcast interviews with innovators and award winners

- Popular publications:
  - Flagship *Communications of the ACM* ***(CACM)*** magazine: http://cacm.acm.org/
  - *ACM Queue* magazine for practitioners: http://queue.acm.org/

- **ACM Digital Library**, the world's most comprehensive database of computing literature: http://dl.acm.org.

- International conferences that draw leading experts on a broad spectrum of computing topics: http://www.acm.org/conferences.

- Prestigious awards, including the **ACM A.M. Turing** and Infosys: http://awards.acm.org/

- And much more…http://www.acm.org.

“Housekeeping”

Twitter: #ACMWebinarScaling

- **Welcome** to today’s ACM Learning Webinar. The presentation starts at the top of the hour and lasts 60 minutes. Slides will advance automatically throughout the event. You can resize the slide area as well as other windows by dragging the bottom right corner of the slide window, as well as move them around the screen. On the bottom panel you’ll find a number of widgets, including Facebook, Twitter, and Wikipedia.

- If you are experiencing any **problems/issues**, refresh your console by pressing the **F5** key on your keyboard in Windows, **Command + R** if on a Mac, or **refresh** your browser if you’re on a mobile device; or close and re-launch the presentation. You can also view the Webcast Help Guide, by clicking on the “**Help**” widget in the bottom dock.

- To control volume, adjust the master volume on your computer. If the volume is still too low, use headphones.

- If you think of a question during the presentation, please type it into the **Q&A box** and click on the submit button. You do not need to wait until the end of the presentation to begin submitting questions.

- At the end of the presentation, you’ll see a survey open in your browser. Please take a minute to fill it out to help us improve your next webinar experience.

- You can download a copy of these slides by clicking on the **Resources** widget in the bottom dock.

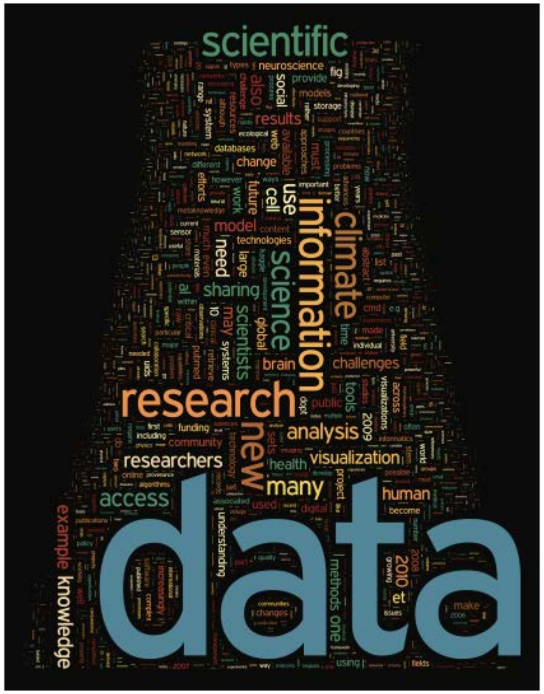
- This session is being recorded and will be archived for on-demand viewing in the next 1-2 days. You will receive an automatic email notification when it is available, and check **http://learning.acm.org/** in a few days for updates. And check out **http://learning.acm.org/webinar** for archived recordings of past webcasts.

Talk Back

- Use Twitter widget to Tweet your favorite quotes from today's presentation with hashtag #ACMWebinarScaling
- Submit questions and comments via Twitter to @acmeducation – we're reading them!
- Use the Facebook and other sharing tools in the bottom panel to share this presentation with friends and colleagues

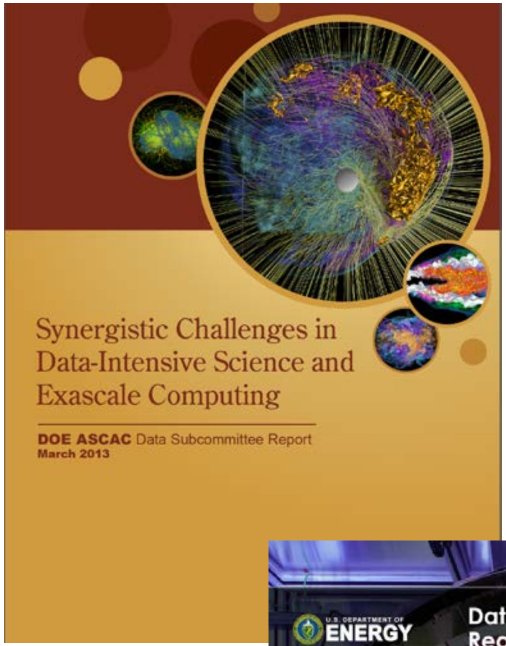
Computing Needs for Science

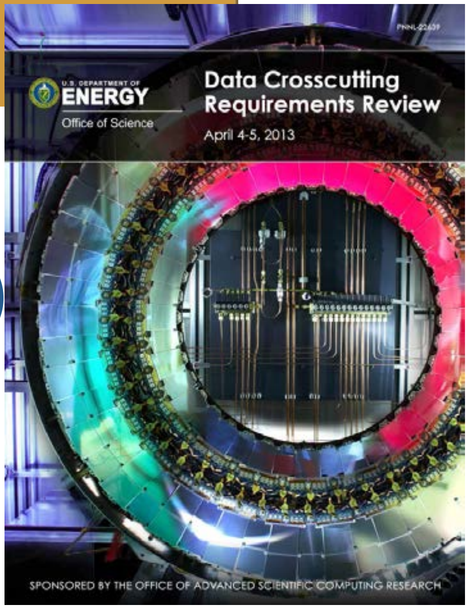
- **Many Communities use Large-Scale Computational Resources**
  - Biology
  - Synchrotron Light Sources
  - Climate/Earth Sciences
  - High Energy Physics
  - Materials Modeling
- **Message: Overall scientific computing use case is driven by traditional supercomputing as well as by data-intensive applications**
- **Optimization of overall balance of compute + I/O + storage + networking**
- **Should think of performance within this global context**



Different Flavors of Computing

- **High Performance Computing ('PDEs')**
  - Parallel systems with a fast network
  - Designed to run tightly coupled jobs
  - High performance parallel file system
  - Batch processing
- **Data-Intensive Computing ('Analytics')**
  - Parallel systems with balanced I/O
  - Designed for data analytics
  - System level storage model
  - Interactive processing
- **High Throughput Computing ('Events'/'Workflows')**
  - Distributed systems with 'slow' networks
  - Designed to run loosely coupled jobs
  - System level/Distributed data model
  - Batch processing



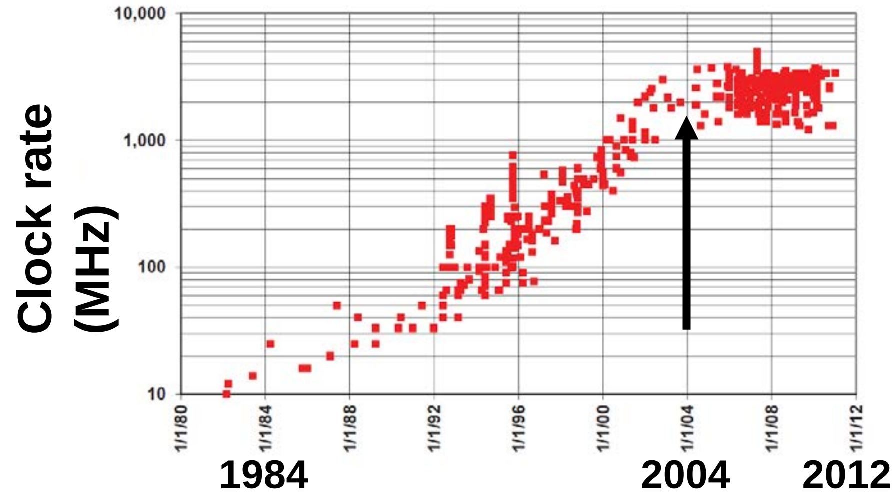


Motivating HPC: The Computational Ecosystem

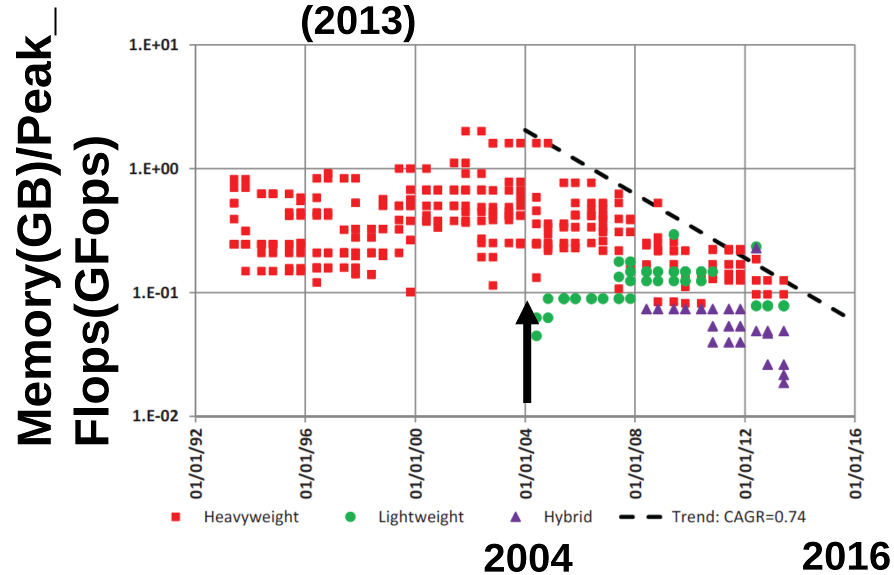
- Motivations for large HPC campaigns:
    1) Quantitative predictions for complex, nonlinear systems
    2) Discover/Expose physical mechanisms
    3) System-scale simulations (‘impossible experiments’)
    4) Large-Scale inverse problems and optimization
- Driven by a wide variety of data sources, computational cosmology must address ALL of the above
- Role of scalability/performance:
    1) Very large simulations necessary, but not just a matter of running a few large simulations
    2) High throughput essential (short wall clock times)
    3) Optimal design of simulation campaigns (parameter scans)
    4) Large-scale data-intensive applications

Supercomputing: Hardware Evolution

- **Power is the main constraint**
  - 30X performance gain by 2020
  - ~10-20MW per large system
  - power/socket roughly const.
- **Only way out: more cores**
  - Several design choices
  - None good from scientist's perspective
- **Micro-architecture gains sacrificed**
  - Accelerate specific tasks
  - Restrict memory access structure (SIMD/SIMT)
- **Machine balance sacrifice**
  - Memory/Flops; comm BW/Flops — all go in the wrong direction
  - (Low-level) code must be refactored



Kogge and Resnick (2013)



Supercomputing: Systems View

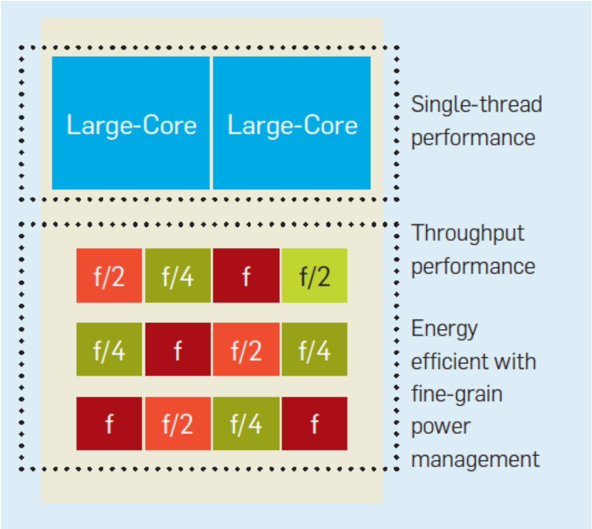
- **HPC is not what it used to be!**
  - HPC systems were meant to be balanced under certain metrics — nominal scores of unity (1990's desiderata)
  - These metrics now range from ~0.1 to ~0.001 on the same system currently and will get worse (out of balance systems)
  - RAM is expensive: memory bytes will not scale like compute flops, era of weak scaling (fixed relative problem size) has ended
- **Challenges**
  - Strong scaling regime (fixed absolute problem size) is much harder than weak scaling (since metric really is 'performance' and not 'scaling')
  - Machine models are complicated (multiple hierarchies of compute/memory/network)
  - Codes must add more physics to use the available compute, adding more complexity
  - Portability across architecture choices must be addressed (programming models, algorithmic choices, trade-offs, etc.)

Supercomputing Challenges: Sociological View

- **Codes and Teams**
  - Most codes are written and maintained by small teams working near the limits of their capability (no free cycles)
  - Community codes, by definition, are associated with large inertia (not easy to change standards, untangle lower-level pieces of code from higher-level organization, find the people required that have the expertise, etc.)
  - Lack of consistent programming model for “scale-up”
  - In some fields at least, something like a “crisis” is approaching (or so people say)
- **What to do?**
  - We will get beyond this (the vector to MPP transition was worse)
  - Transition needs to be staged (not enough manpower to entirely rewrite code base)
  - Prediction: There will be no ready made solutions
  - Realization — “You have got to do it for yourself”

Co-Design vs. Code Design

- **HPC Myths**
  - The magic compiler
  - The magic programming model/language
  - Special-purpose hardware
  - Co-Design (not now anyway, but maybe in the future —)
- **Dealing with Today's Reality**
  - Code teams must understand all levels of the system architecture, but do not be enslaved by it (software cycles are long)!
  - Must have a good idea of the 'boundary conditions' (what may be available, what is doable, etc.)
  - 'Code Ports' is ultimately a false notion
  - Start thinking out of the box — domain scientists and computer scientists and engineers must work together



Future heterogeneous manycore system, Borkar and Chien (2011)

Large Scale Structure: Vlasov-Poisson Equation

$$\frac{\partial f_i}{\partial t} + \dot{\mathbf{x}}\frac{\partial f_i}{\partial \mathbf{x}} - \nabla\phi\frac{\partial f_i}{\partial \mathbf{p}} = 0, \qquad \mathbf{p} = a^2\dot{\mathbf{x}},$$

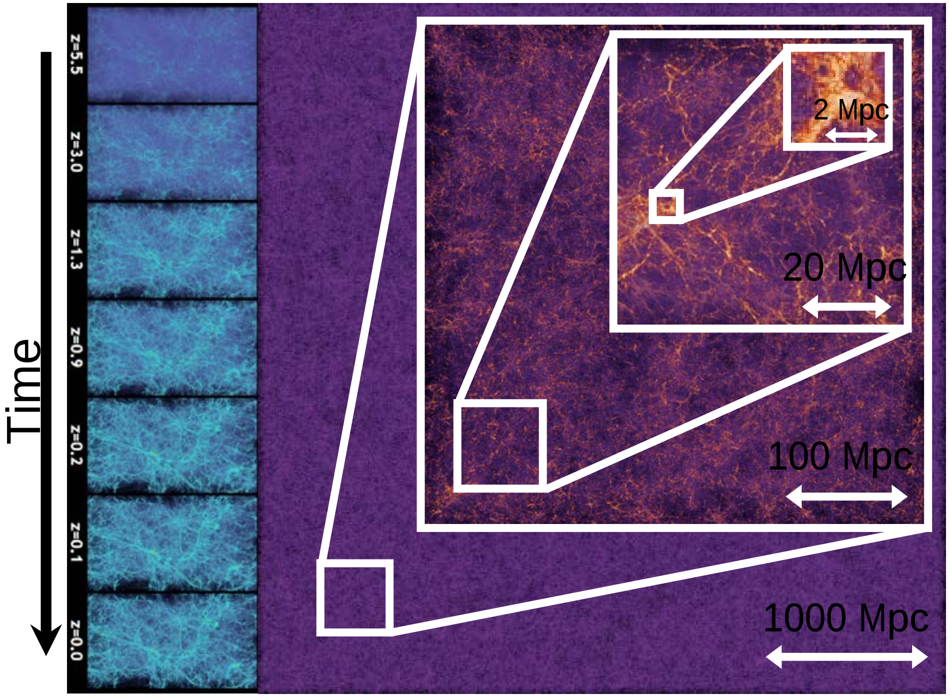
$$\begin{aligned}
\nabla^2\phi &= 4\pi G a^2(\rho(\mathbf{x},t) - \langle\rho_{\rm dm}(t)\rangle) = 4\pi G a^2\Omega_{\rm dm}\delta_{\rm dm}\rho_{\rm cr},\\
\delta_{\rm dm}(\mathbf{x},t) &= (\rho_{\rm dm} - \langle\rho_{\rm dm}\rangle)/\langle\rho_{\rm dm}\rangle),\\
\rho_{\rm dm}(\mathbf{x},t) &= a^{-3}\sum_i m_i \int d^3\mathbf{p}\, f_i(\mathbf{x},\dot{\mathbf{x}},t).
\end{aligned}$$

Cosmological Vlasov-Poisson Equation

- Properties of the Cosmological Vlasov-Poisson Equation:
  - 6-D PDE with long-range interactions, no shielding, all scales matter; models gravity-only, collisionless evolution
  - Jeans instability drives structure formation at all scales from smooth Gaussian random field initial conditions
  - Extreme dynamic range in space and mass (in many applications, million to one in both space and density, ‘everywhere’)

Large Scale Structure Simulation Requirements

- Force and Mass Resolution:
  - Galaxy halos ~100kpc, hence force resolution has to be ~kpc; with Gpc box-sizes, a dynamic range of a million to one
  - Ratio of largest object mass to lightest is ~10000:1
- Physics:
  - Gravity dominates at scales greater than ~Mpc
  - Small scales: galaxy modeling, semi-analytic methods to incorporate gas physics/feedback/star formation
- Computing 'Boundary Conditions':
  - Total memory in the PB+ class
  - Performance in the 10 PFlops+ class
  - Wall-clock of ~days/week, in situ analysis



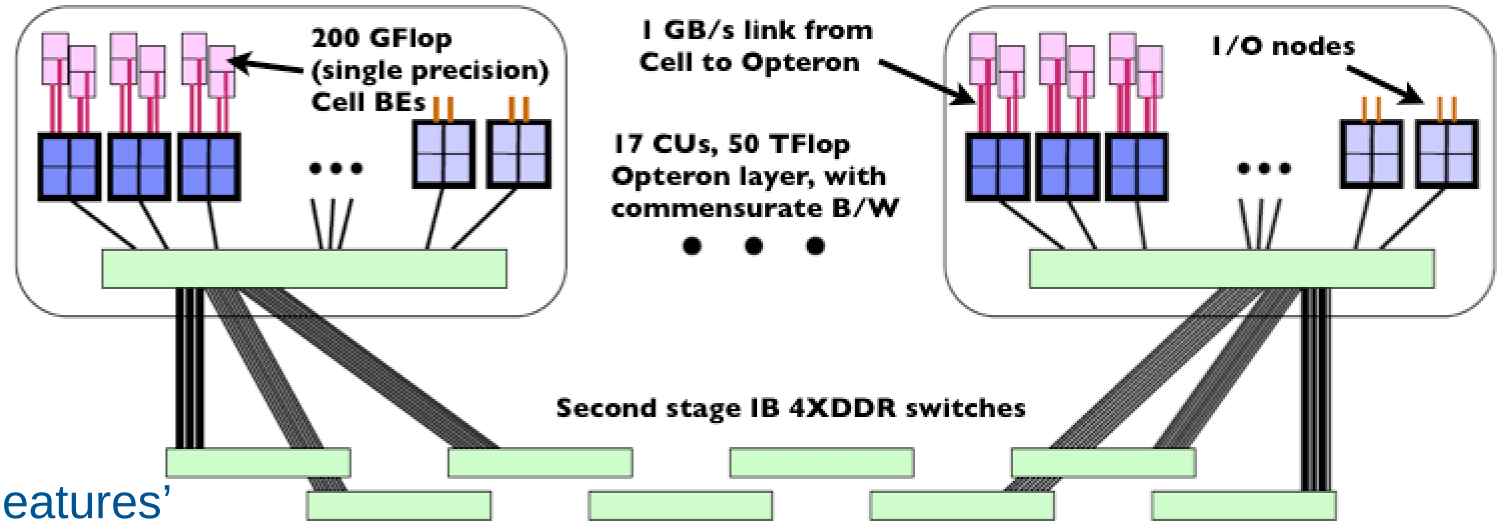
Gravitational Jeans Instability

Can the Universe be run as a short computational 'experiment'?

Architectural Challenges: The HACC Story

Andrew White Dec 7, 2007 + What if you had a petaflop/s

Roadrunner: Prototype for modern accelerated architectures, first to break the PFlops barrier



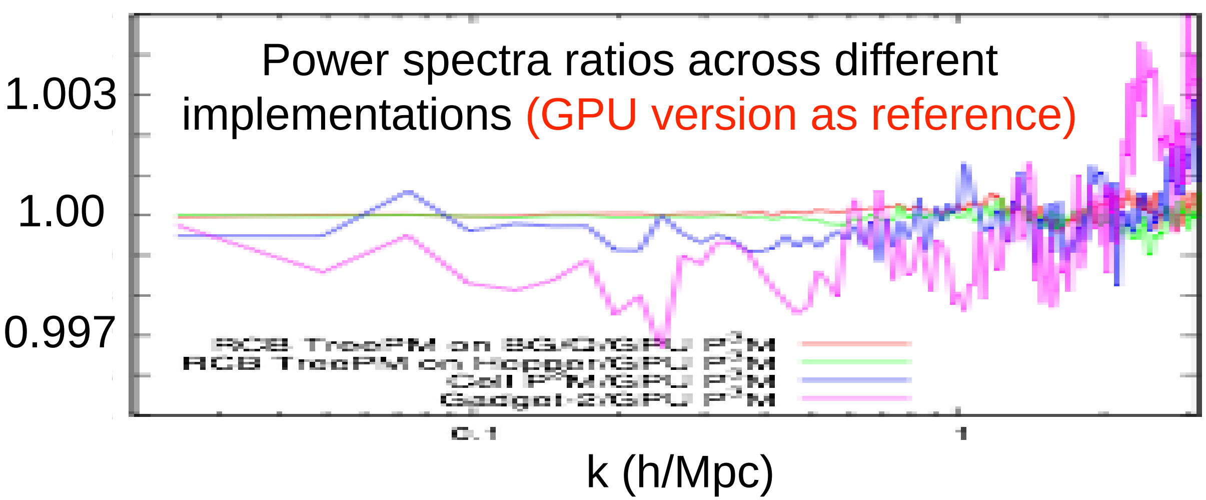
Architectural 'Features'

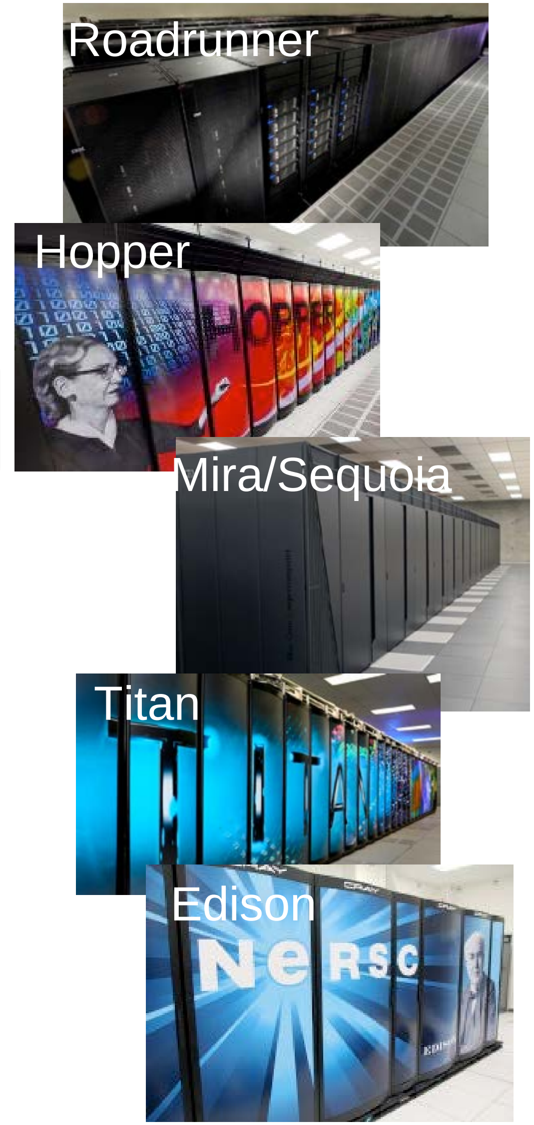
- Complex heterogeneous nodes
- Simpler cores, lower memory/core, no real cache
- Skewed compute/communication balance
- Programming models?
- I/O? File systems?
- Effect on code longevity

HACC team meets Roadrunner

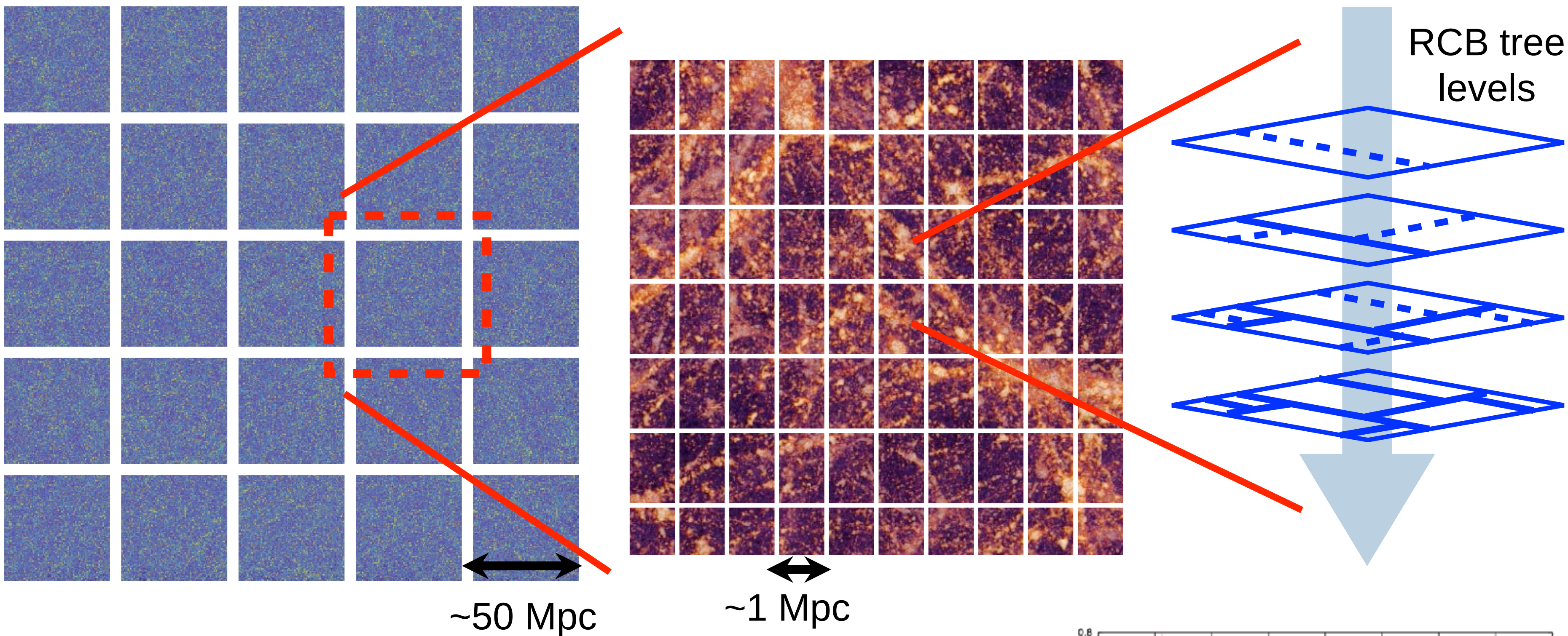
Combating Architectural Diversity with HACC

- Architecture-independent performance/scalability: 'Universal' top layer + 'plug in' node-level components; minimize data structure complexity and data motion
- Programming model: 'C++/MPI + X' where X = OpenMP, Cell SDK, OpenCL, CUDA, --
- Algorithm Co-Design: Multiple algorithm options, stresses accuracy, low memory overhead, no external libraries in simulation path
- Analysis tools: Major analysis framework, tools deployed in stand-alone and in situ modes





HACC Structure: Universal vs. Local Layers

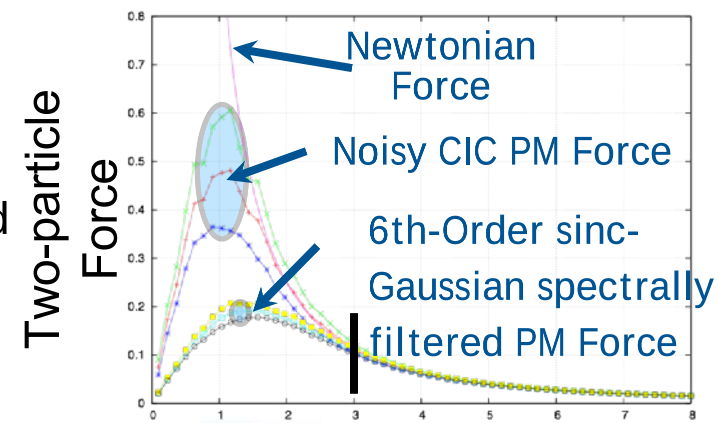


HACC Top Layer:
3-D domain decomposition with particle replication at boundaries ('overloading') for Spectral PM algorithm (long-range force)

Host-side: Scaling controlled by FFT

HACC 'Nodal' Layer:
Short-range solvers employing combination of flexible chaining mesh and RCB tree-based force evaluations

Performance controlled by short-range solver



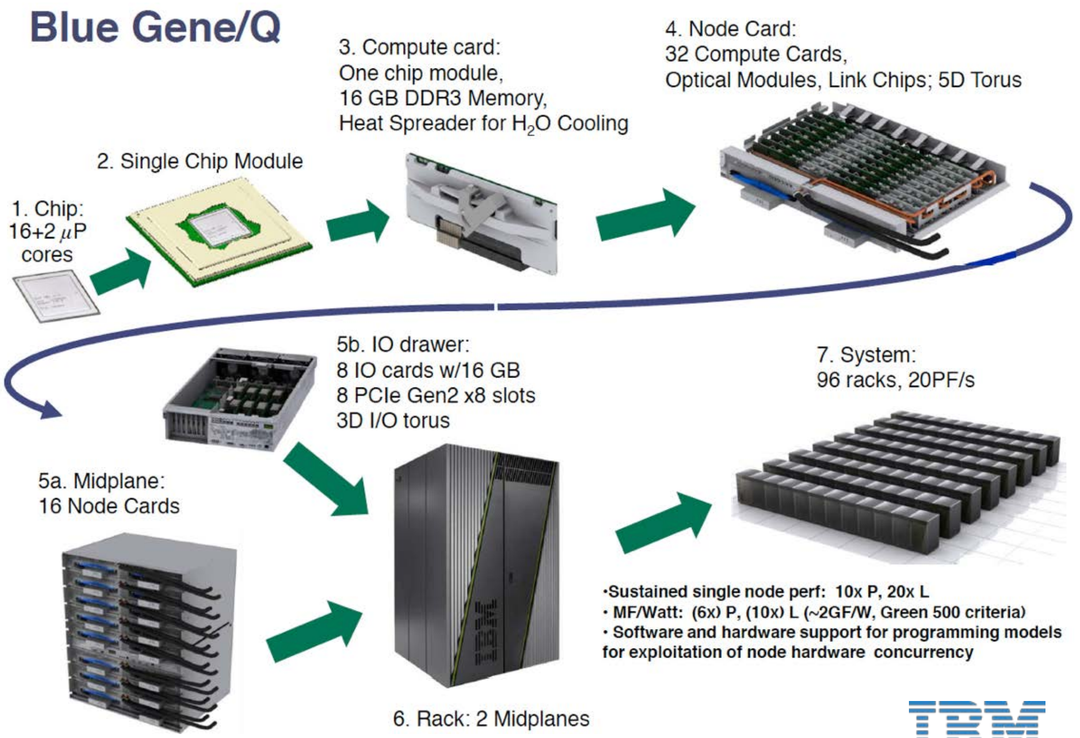
HACC: Algorithmic Features and Options

- Fully Spectral Particle-Mesh Solver: 6th-order Green function, 4th-order Super-Lanczos derivatives, high-order spectral filtering, high-accuracy polynomial for short-range forces
- Custom Parallel FFT: Pencil-decomposed, high-performance FFT (up to 15K^3)
- Particle Overloading: Particle replication at 'node' boundaries to reduce/delay communication (intermittent refreshes), important for accelerated systems
- Flexible Chaining Mesh: Used to optimize tree and P3M methods
- Optimal Splitting of Gravitational Forces: Spectral Particle-Mesh melded with direct and RCB ('fat leaf') tree force solvers (PPTPM), short hand-over scale (dynamic range splitting ~ 10,000 X 100); pseudo-particle method for multipole expansions
- Mixed Precision: Optimize memory and performance (GPU-friendly!)
- Optimized Force Kernels: High performance without assembly
- Adaptive Symplectic Time-Stepping: Symplectic sub-cycling of short-range force timesteps; adaptivity from automatic density estimate via RCB tree
- Custom Parallel I/O: Topology aware parallel I/O with lossless compression (factor of 2); 1.5 trillion particle checkpoint in 4 minutes at ~160GB/sec on Mira

HACC on the IBM Blue Gene/Q

HACC BG/Q Experience

- System: BQC chip — 16 cores, 205GFlops, 16GB RAM, 32MB L2, 400GB/s crossbar; 5-D torus network at 40GB/s
- Programming Models: Two-tiered programming model (MPI+OpenMP) very successful, use of vector intrinsics (QPX) essential
- I/O: Custom I/O implementation (one file per I/O node, disjoint data region/process) gives ~2/3 of peak performance under production conditions
- Job Mix: Range of job sizes running on Mira, from 2 to 32 racks

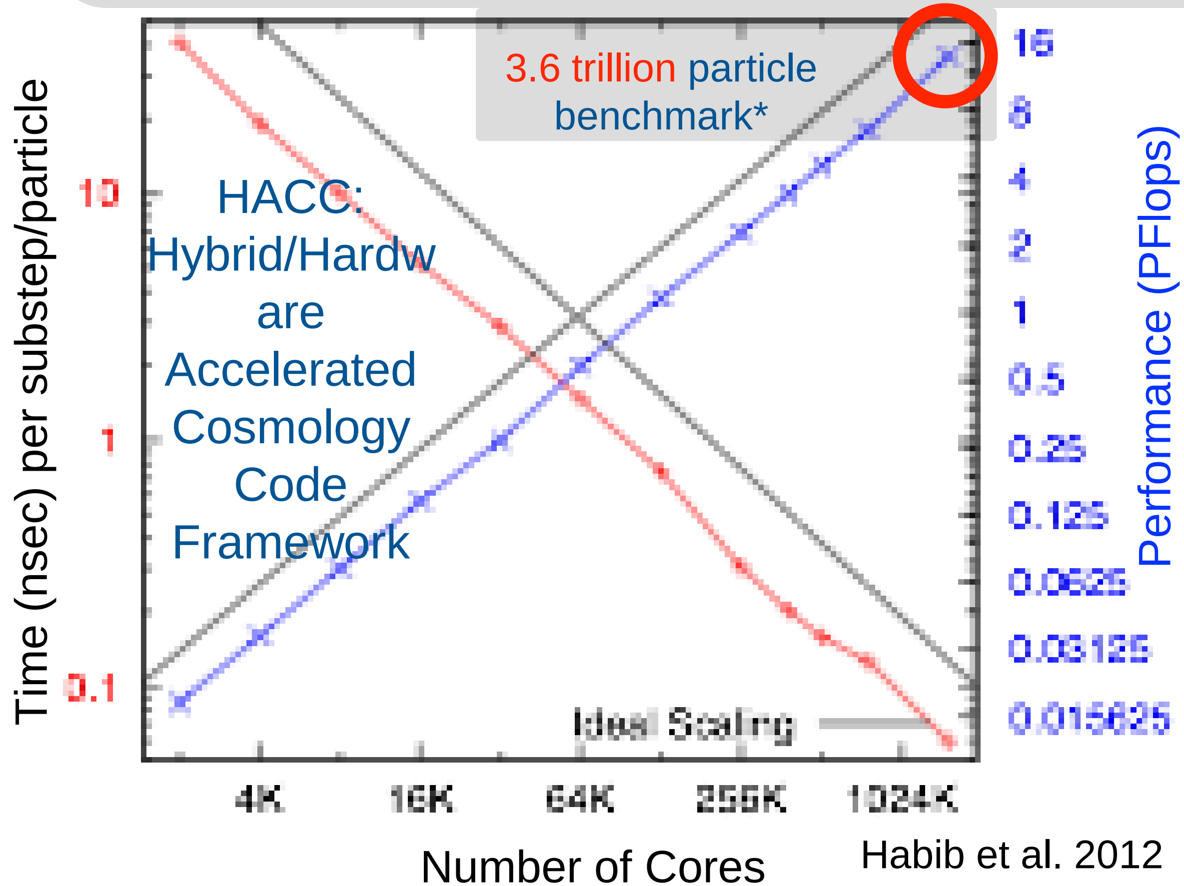


HACC on the BG/Q

HACC BG/Q Version

- Algorithms: FFT-based SPM; PP+RCB Tree
- Data Locality: Rank level via ‘overloading’, at tree-level use the RCB grouping to organize particle memory buffers
- Build/Walk Minimization: Reduce tree depth using rank-local trees, shortest hand-over scale, bigger p-p component
- Force Kernel: Use polynomial representation (no look-ups); vectorize kernel evaluation; hide instruction latency

13.94 PFlops, 69.2% peak, 90% parallel efficiency on 1,572,864 cores/MPI ranks, 6.3M-way concurrency



Habib et al. 2012

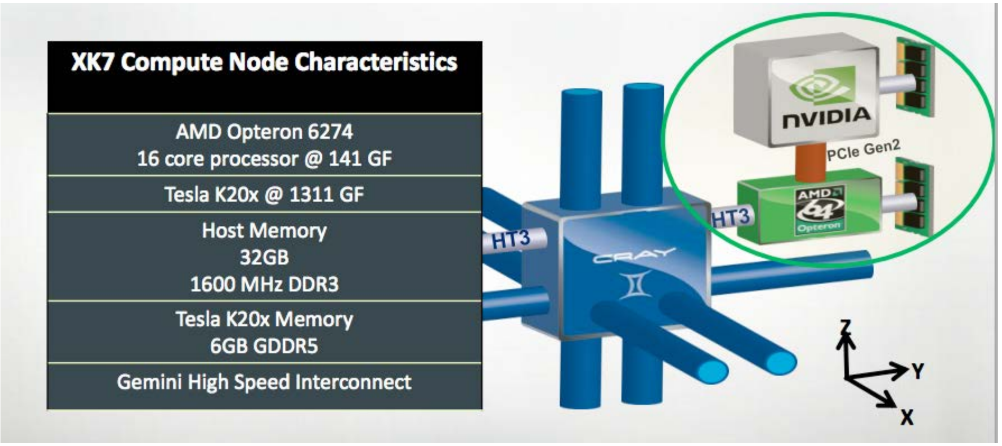
*largest ever run

HACC weak scaling on the IBM BG/Q (MPI/OpenMP)

Accelerated Systems: HACC on Titan (Cray XK7)

Imbalances and Bottlenecks

- Memory is primarily host-side (32 GB vs. 6 GB) (against Roadrunner's 16 GB vs. 16 GB), important thing to think about (in case of HACC, the 'grid/particle' balance)
- PCIe is a key bottleneck; overall interconnect B/W does not match Flops (not even close)
- There's no point in 'sharing' work between the CPU and the GPU, performance gains will be minimal — GPU must dominate
- The only reason to write a code for such a system is if you can truly exploit its power (2 X CPU is a waste of effort!)



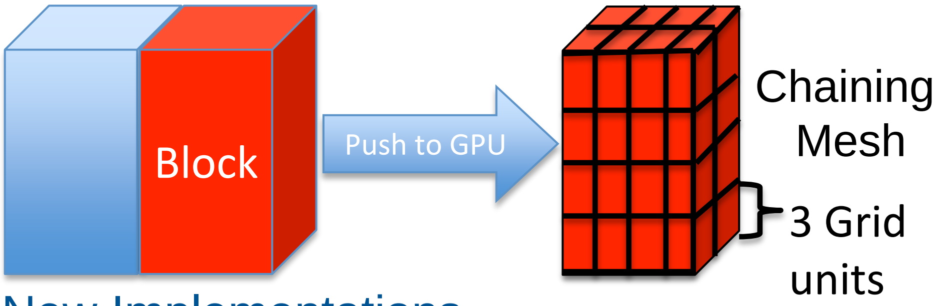
Strategies for Success

- It's (still) all about understanding and controlling data motion
- Rethink your code and even approach to the problem
- Isolate hotspots, and design for portability around them (modular programming)
- Pragmas will never be the full answer (with maybe an exception or two)

HACC on Titan: GPU Implementation (Schematic)

P3M Implementation (OpenCL):

- Spatial data pushed to GPU in large blocks, data is sub-partitioned into chaining mesh cubes
- Compute forces between particles in a cube and neighboring cubes
- Natural parallelism and simplicity leads to high performance
- Typical push size ~2GB; large push size ensures computation time exceeds memory transfer latency by a large factor
- More MPI tasks/node preferred over threaded single MPI tasks (better host code performance)

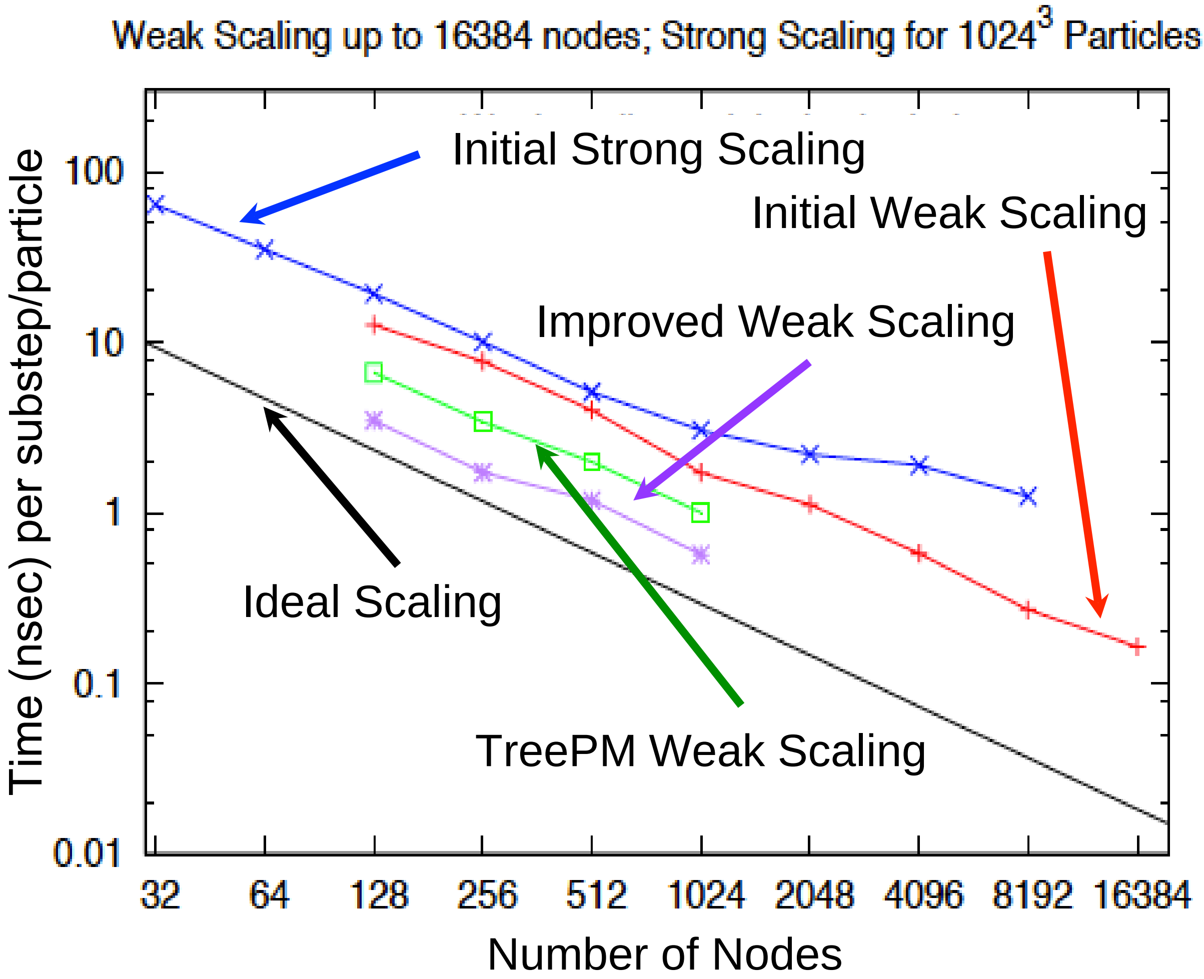


New Implementations (OpenCL and CUDA):

- P3M with data pushed only once per long time-step, completely eliminating memory transfer latencies (orders of magnitude less); uses ‘soft boundary’ chaining mesh, rather than rebuilding every sub-cycle
- TreePM analog of BG/Q code written in CUDA, also produces high performance

HACC on Titan: GPU Implementation Performance

- P3M kernel runs at 1.6TFlops/node at 40.3% of peak (73% of algorithmic peak)
- TreePM kernel was run on 77% of Titan at 20.54 PFlops at almost identical performance on the card
- Because of less overhead, P3M code is (currently) faster by factor of two in time to solution



99.2% Parallel Efficiency

Summary

Basic Ideas:

- Thoughtful design of flexible code infrastructure; minimize number of computational ‘hot spots’, explore multiple algorithmic ideas — exploit domain science expertise
- Because machines are so out of balance, focusing only on the lowest-level compute-intensive kernels can be a mistake (‘code ports’)
- One possible solution is an overarching universal layer with architecture-dependent, plug-in modules (with implications for productivity)
- Understand data motion issues in depth — minimize data motion, always look to hide communication latency with computation
- Be able to change on fast timescales (HACC needs no external libraries in the main simulation code — helps to get on new machines early)
- As science outputs become more complex, data analysis becomes a very significant fraction of available computational time — optimize performance with this in mind

ACM: The Learning Continues…

- Questions about this webcast? learning@acm.org
- ACM Learning Webinars (on-demand archive): http://learning.acm.org/webinar

  ACM Learning Center: http://learning.acm.org
- ACM SIGHPC: http://www.sighpc.org/
- ACM Queue: http://queue.acm.org/