Languages and Compilers for Exascale Science

Kathy Yelick
Professor of Electrical Engineering and Computer Sciences
U.C. Berkeley

Associate Laboratory Director
Computing Sciences
Lawrence Berkeley National Laboratory

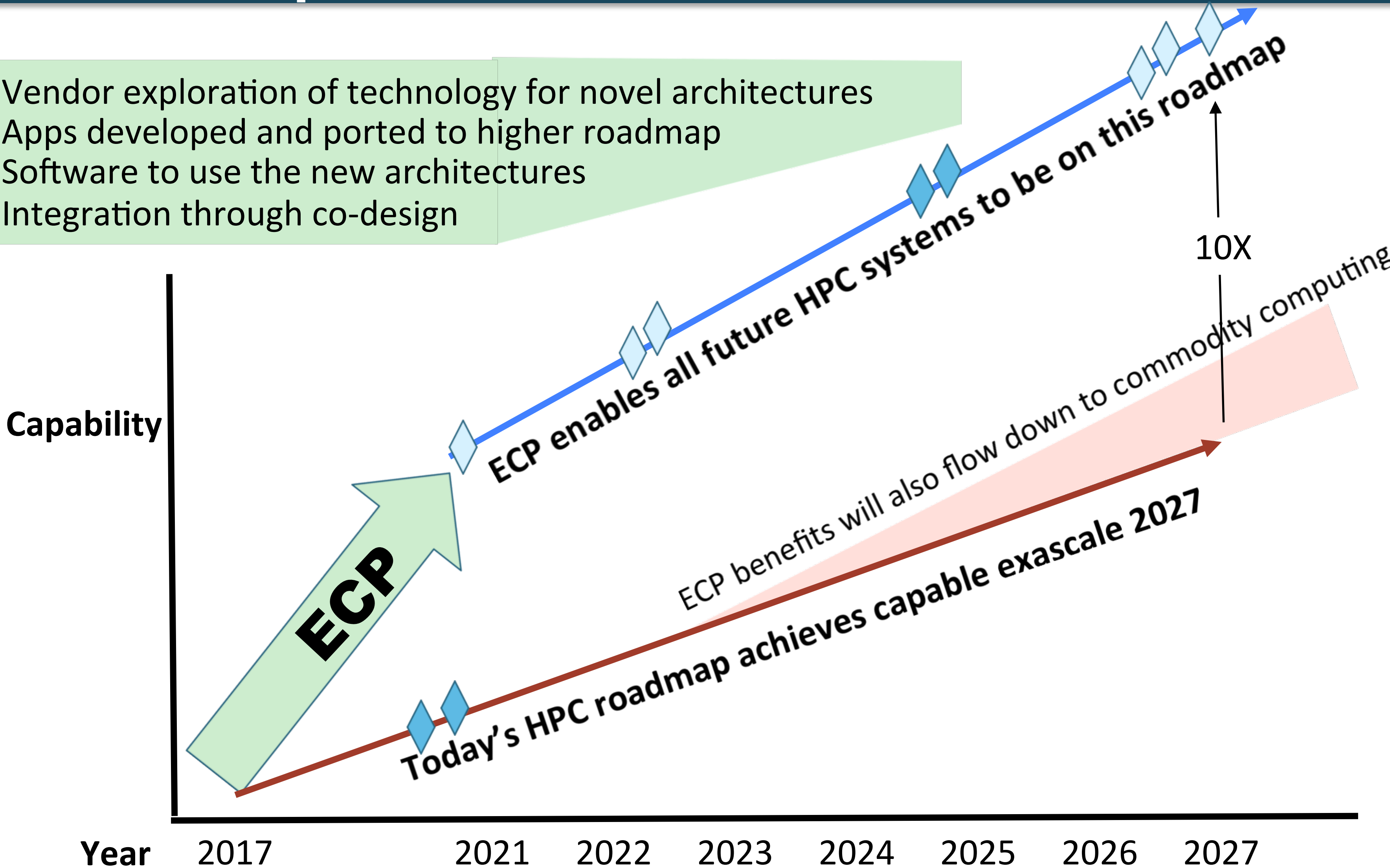
NERSC Supercomputing for Science and Energy

State-of-the art computing for the broad science community – over 7000 users, 700 applications

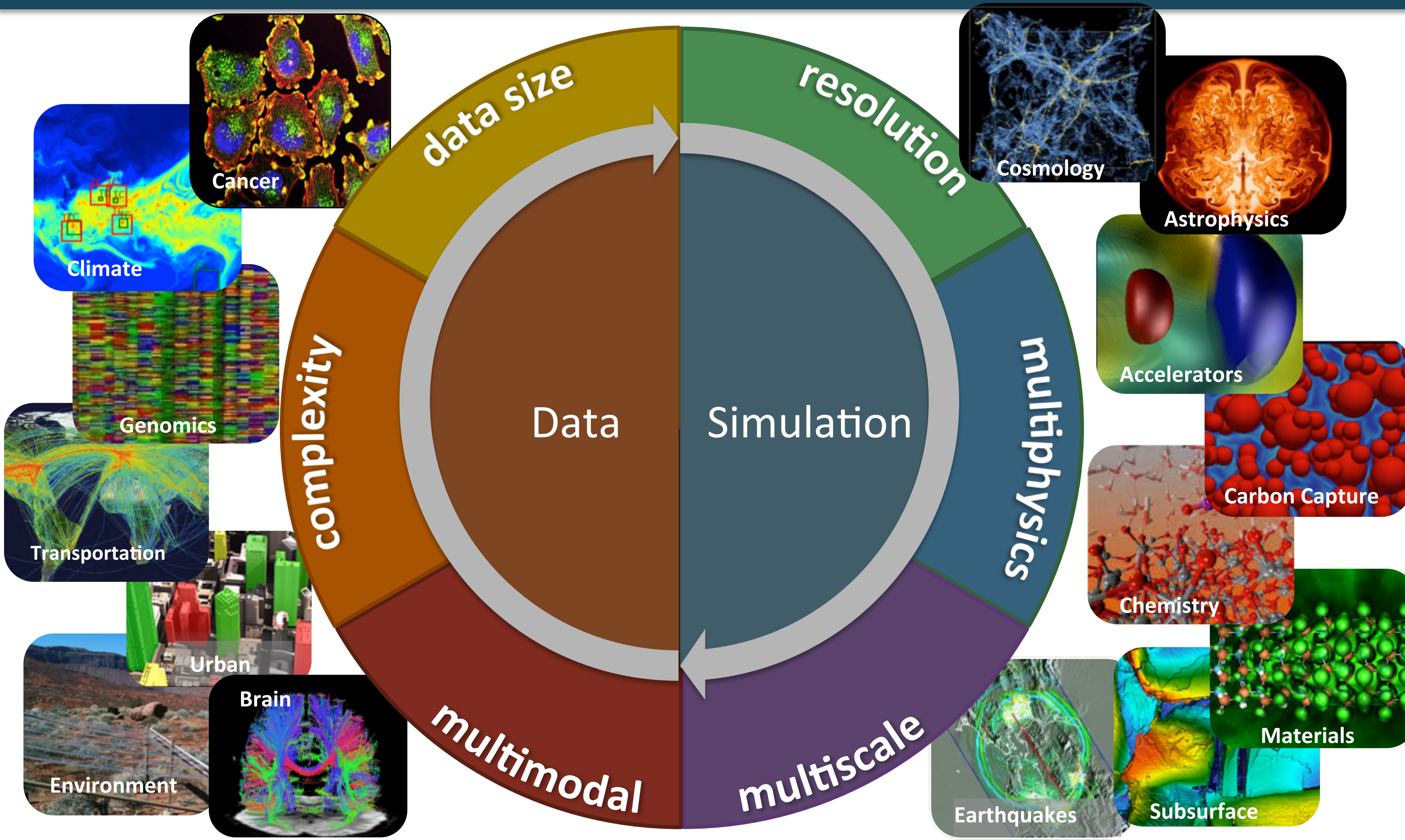
Exascale computing, combined with state-of-the-art mathematical models, algorithms, software techniques and data will enable breakthrough science

Exascale Computing Project (US DOE ECP) to Impact Broad HPC landscape

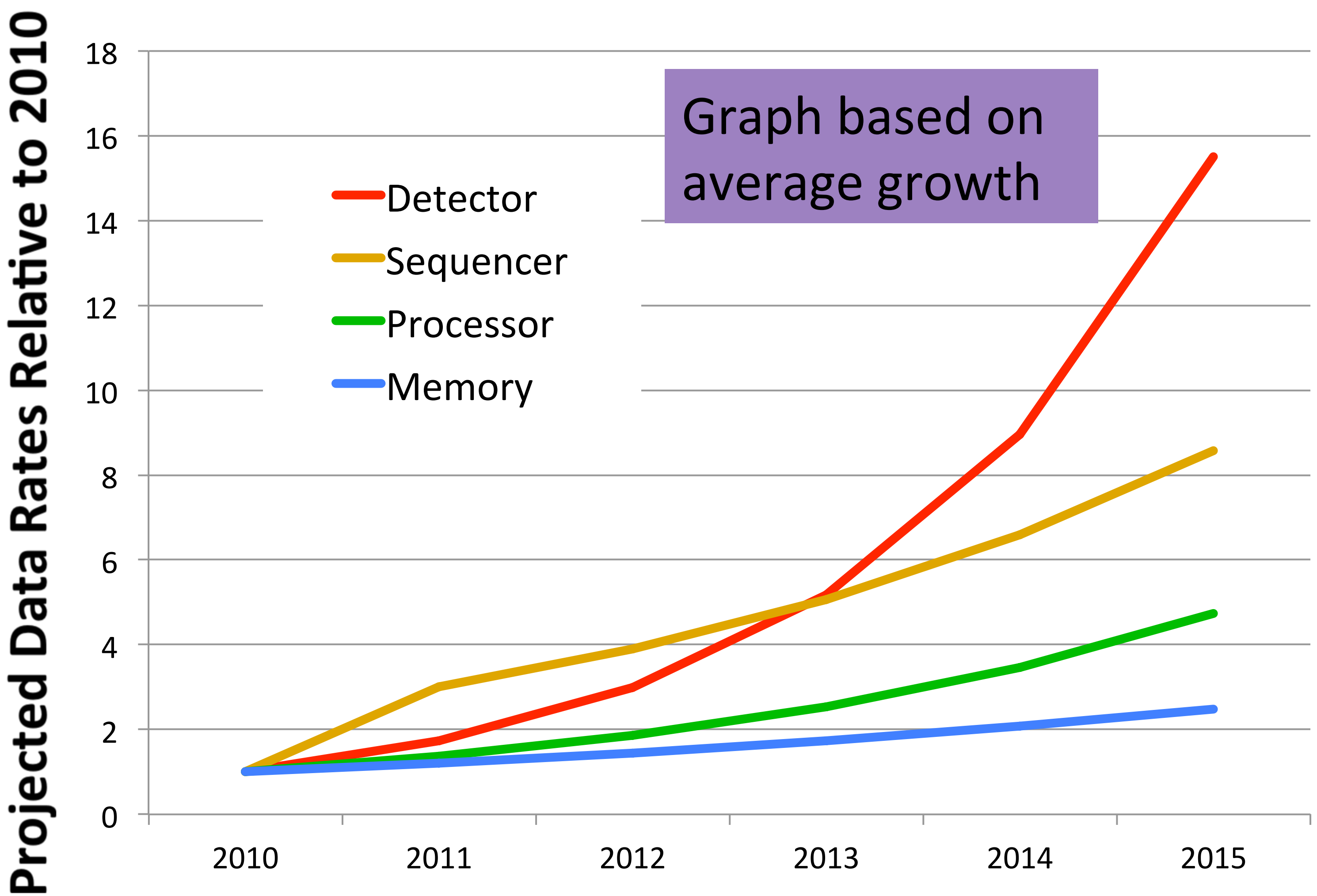
- Vendor exploration of technology for novel architectures
- Apps developed and ported to higher roadmap
- Software to use the new architectures
- Integration through co-design



The Science Challenges at Exascale

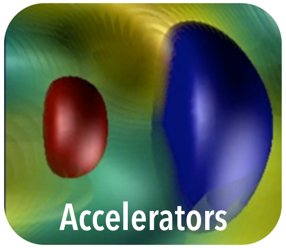


Data Growth is Outpacing Computing Growth

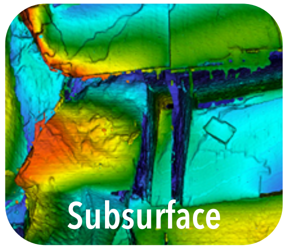


Mathematics will provide Exasacle Science Capabilities

Adaptive Mesh Refinement

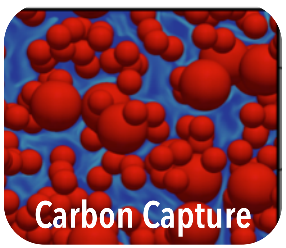


A 1 TeV electron-positron collider

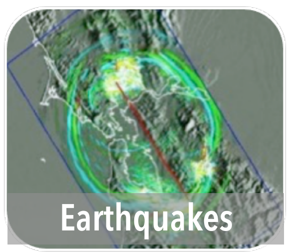


Geo-mechanical chemical evolution of fracking

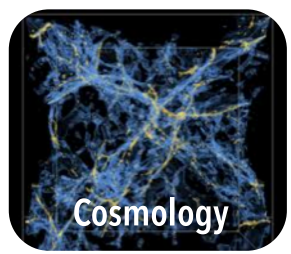
Source of heaviest elements



1MWe chemical looping reactor

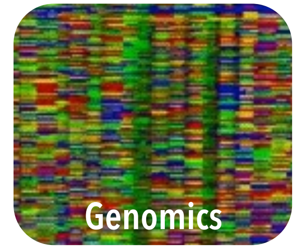


Regional-scale model to simulate structures

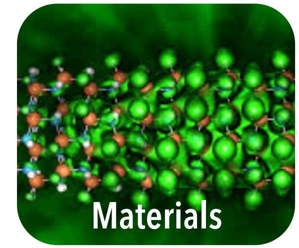


Dark energy equation of state

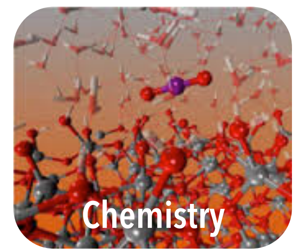
Scalable (Sparse) Solvers



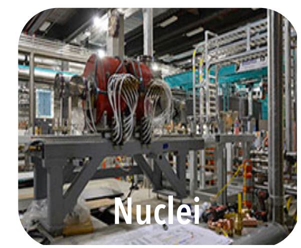
Gene clusters for biomanufacturing



Defects, interfaces and disorder in functional materials



Catalytic conversion of biomass-derived intermediates



Large neutron-rich nuclei and nuclear binding

Berkeley Lab has demonstrated unsurpassed ability to harness the power of advanced mathematics and computer science for high-impact science.

Former cosmology breakthrough (Nobel prize)

More accurate way to measure

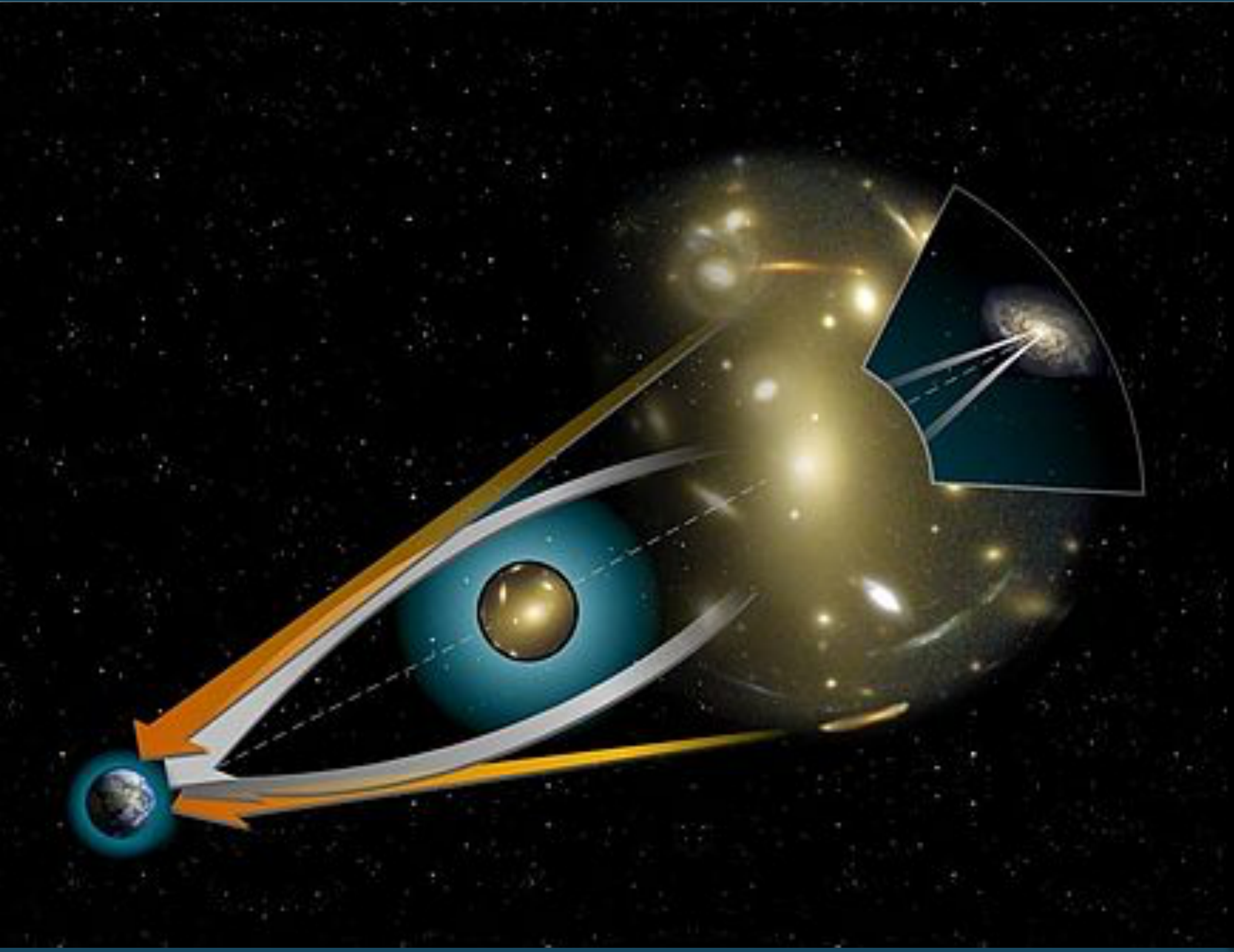
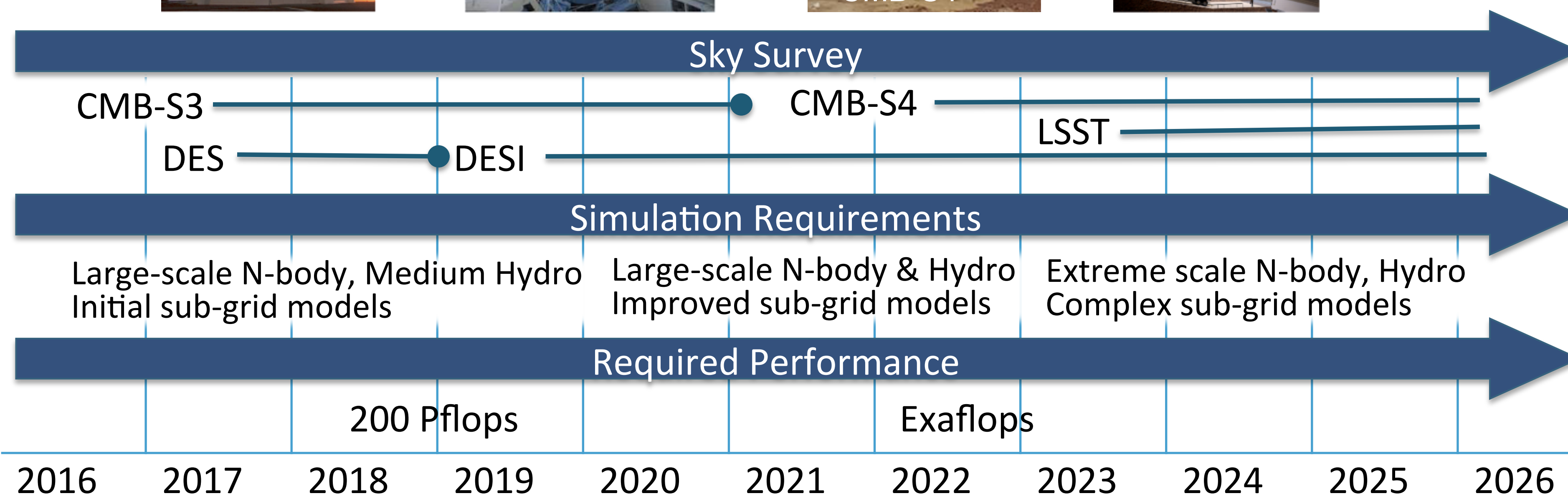


Image: wikipedia

Recent cosmology breakthrough in observation

Cosmology observations drive simulations

- **Science:** *Dark Energy, Dark Matter, Gravitational Waves, Neutrino Mass*
- **Computation:** factor of **X100** increase in science reach, order of magnitude improvement in modeling accuracy and predictability

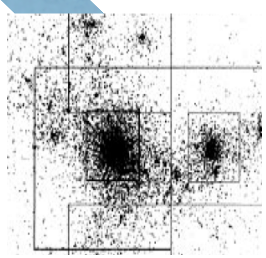


Precision Cosmology: Simulation Frontiers

Fidelity/Complexity

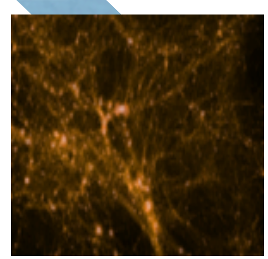
Terascale

- 1st-generation surveys
- Singzle-probe simulations



Petascale

- 2nd-generation surveys
- Multi-probe simulations
- Few precision probes
- Intermediate accuracy parameter estimation



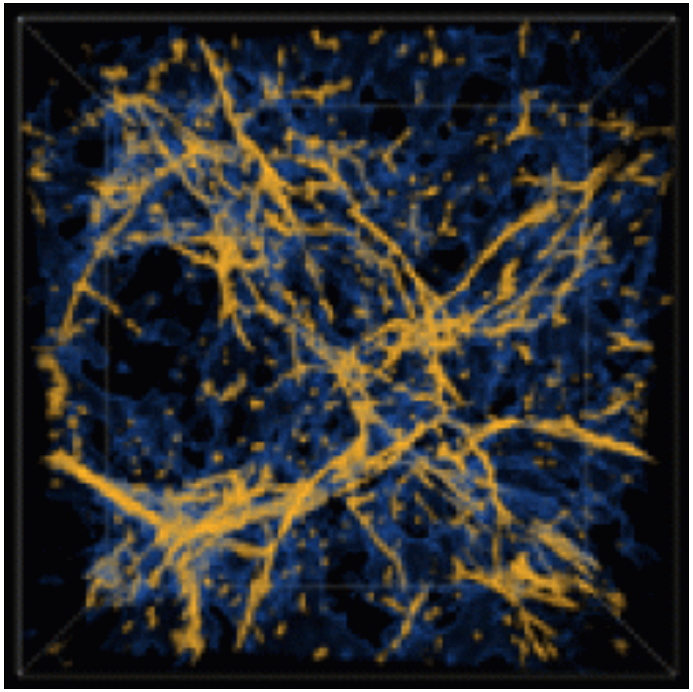
Exascale

- Next-generation surveys
- End-to-end, multi-probe survey-scale simulations
- Multiple cross-calibrated probes
- UQ-enabled cosmic calibration frameworks

Simulation Volume

Cosmology at the Exascale

Synthetic galaxy catalog for LSST generated with HACC and Galacticus codes

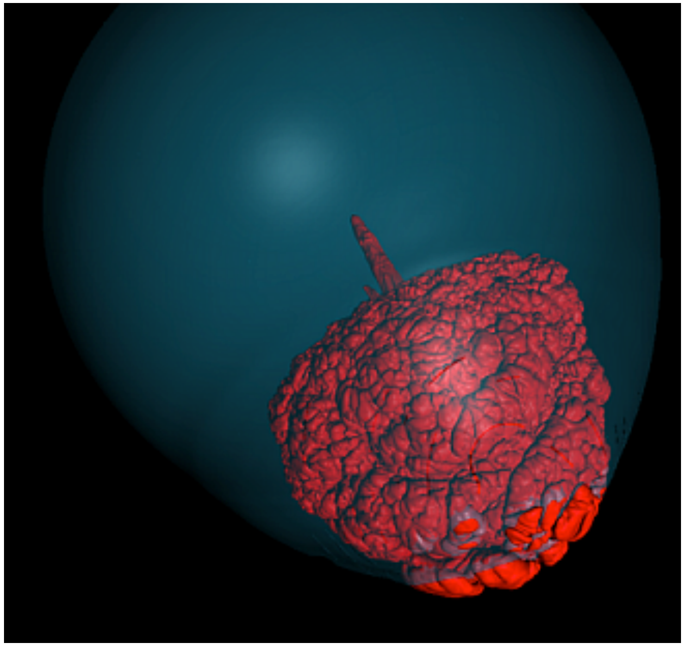


Simulation of Lyman-Alpha Forest with Nyx, used to estimate neutrino mass and as a standard ruler.

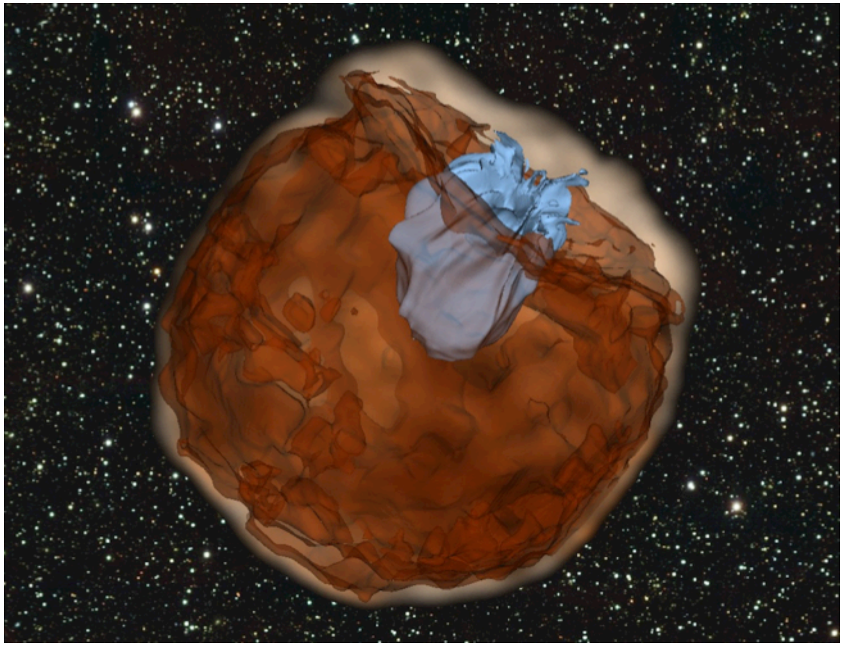
Exascale is needed to model and interpret the latest observations

Improve understanding of *Dark Energy, Dark Matter, Primordial Gravitational Waves, Neutrino Mass, and parametrics such as the Hubble Constant*

Astrophysics at the Exascale



Less than a second after ignition, the flame breaks through the surface of an expanded white dwarf (using AMR)

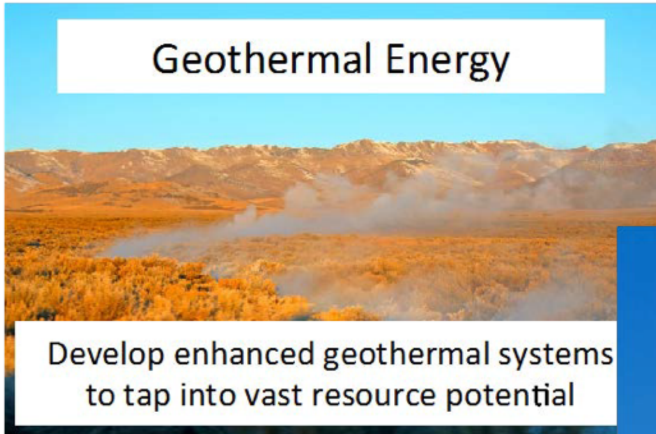


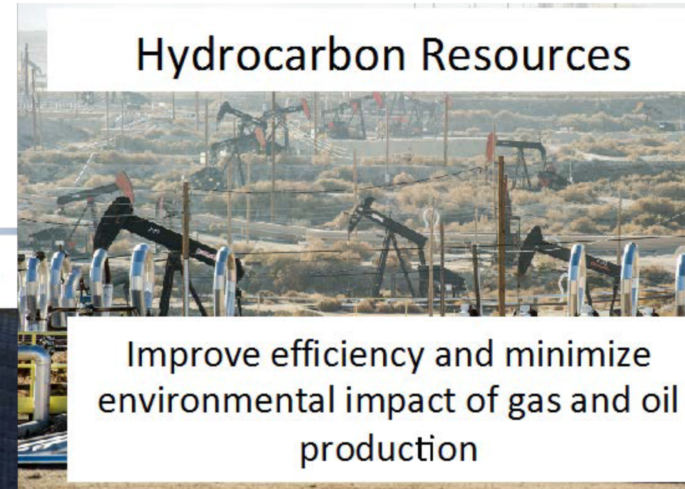
Expanding debris from a supernova explosion (red) running over and shredding a nearby star (blue)

Exascale is needed to identify the source of the heaviest elements

Understand rapid neutron capture process (r-process) by simulating scenarios: core-collapse supernovae, neutron star mergers, and accreting black holes

Subsurface Science at the Exascale

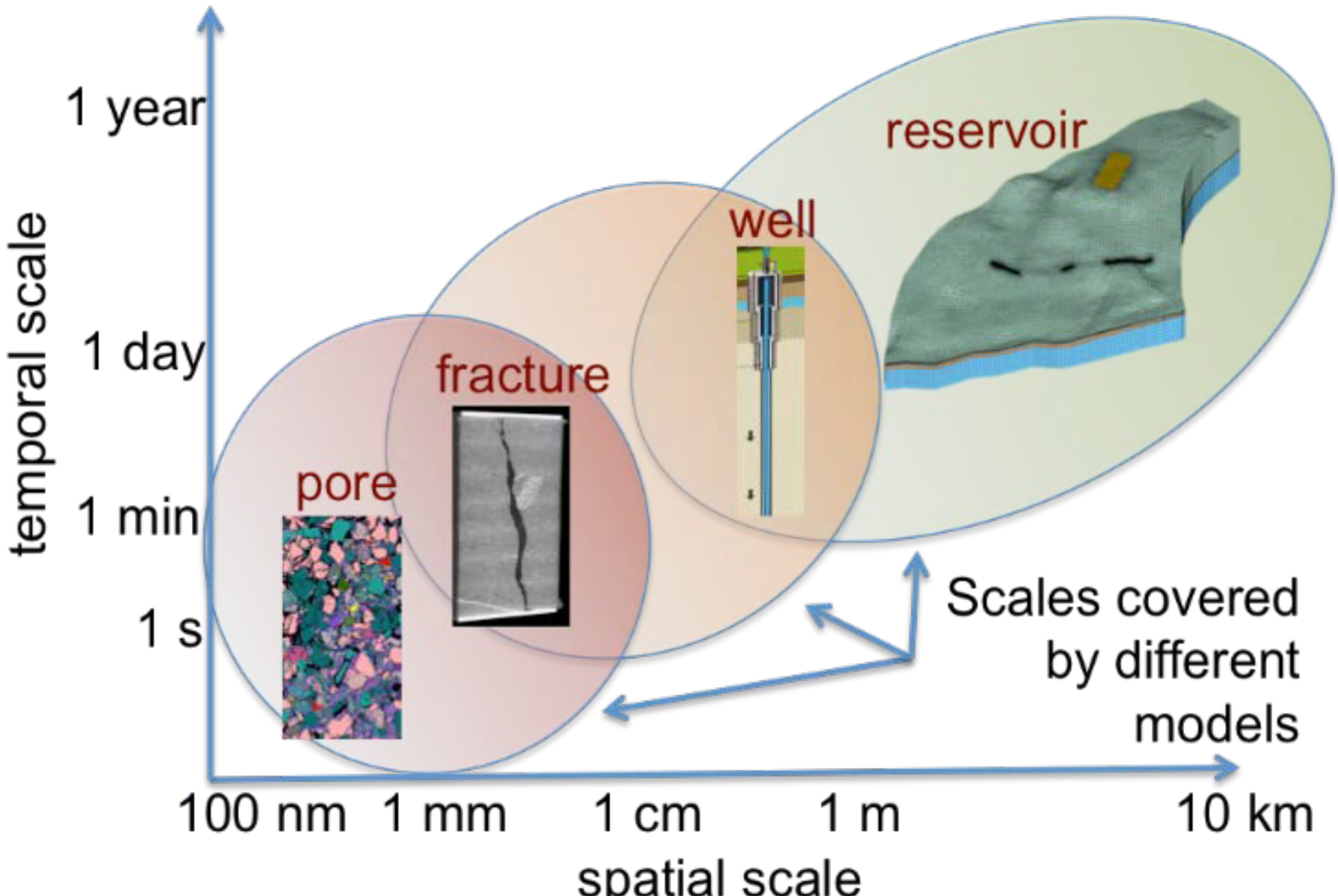




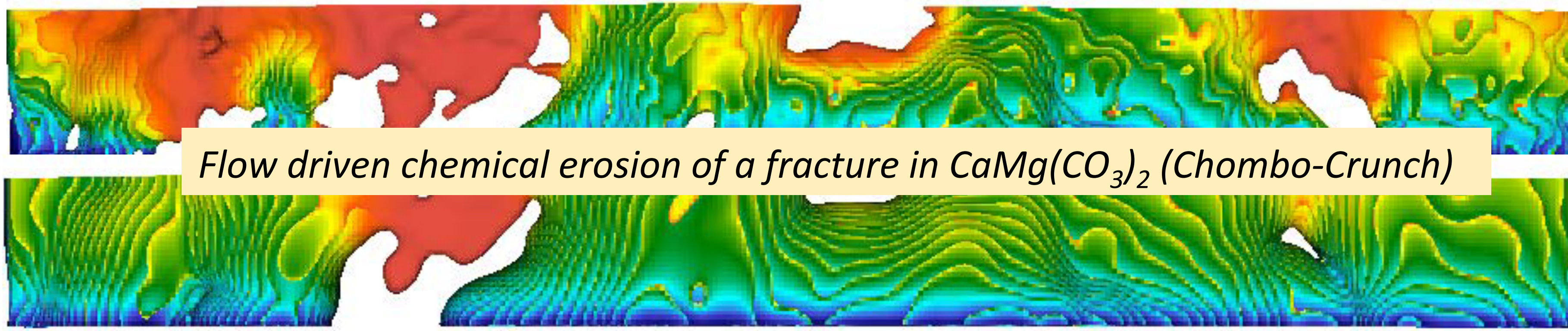
Exascale is needed for impacts of energy extraction and waste storage on subsurface integrity

Simulate an entire field of well bores and their interaction through the reservoir over 100 year timescales. Simulate the evolution of a fracture system in caprock subject to geomechanical and geochemical stresses over scales from pore (micron) to 100 meters

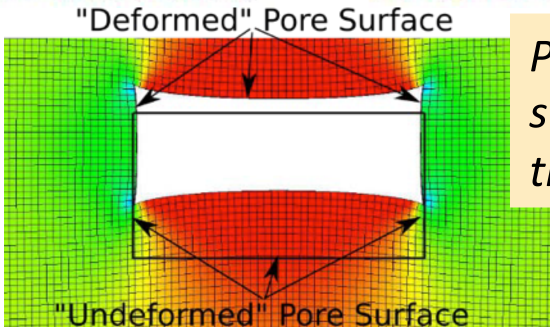
Subsurface science requires modeling across scales



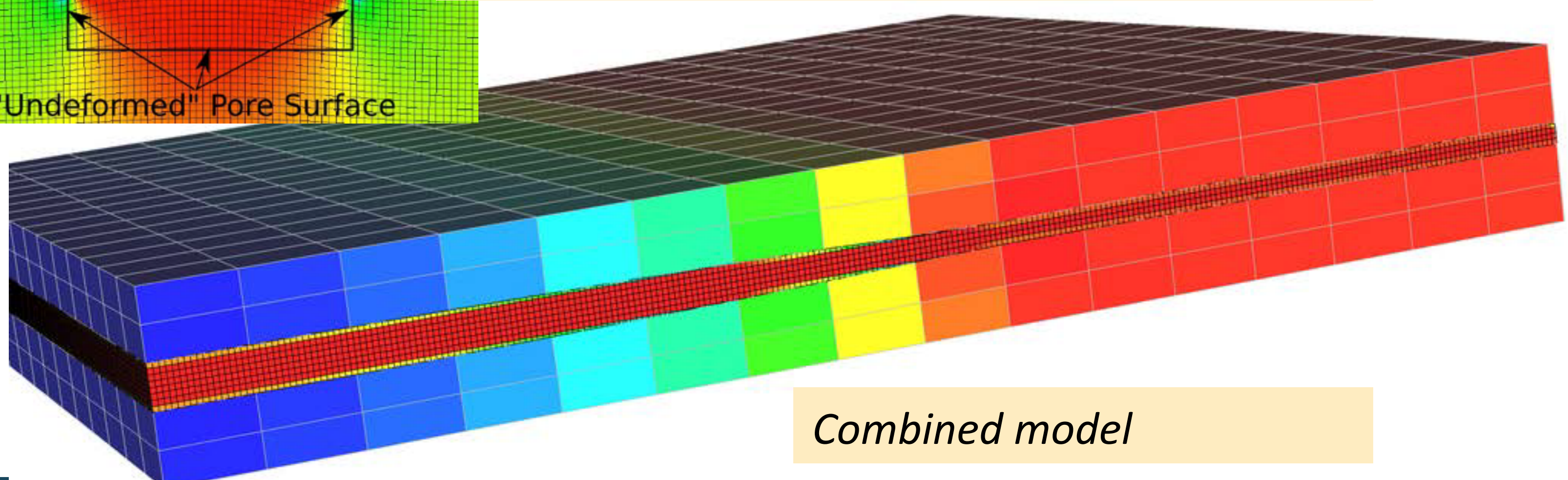
Combining codes to deliver new science capability



Flow driven chemical erosion of a fracture in $CaMg(CO_3)_2$ (Chombo-Crunch)

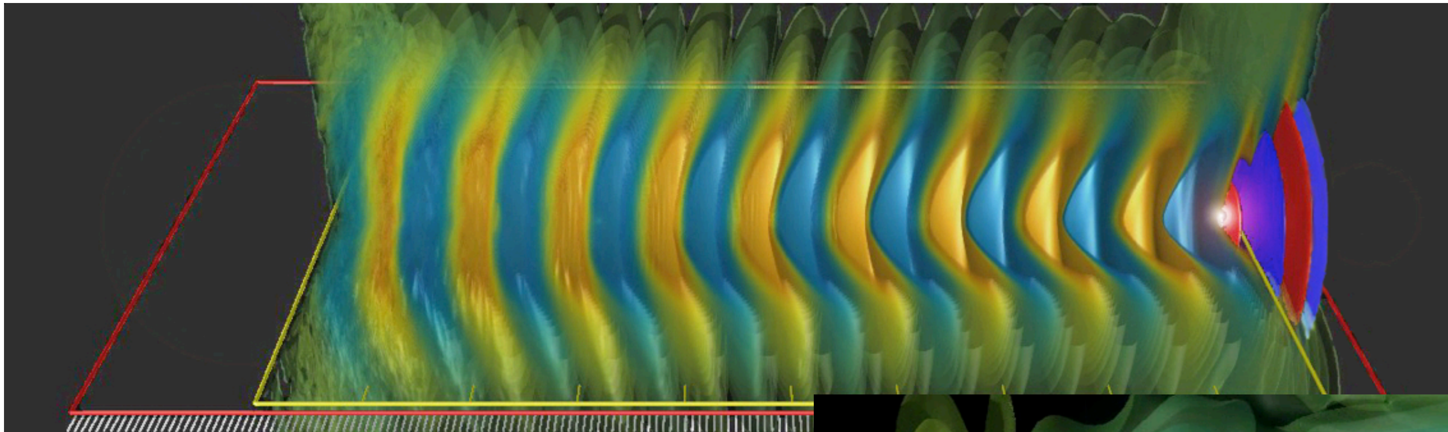


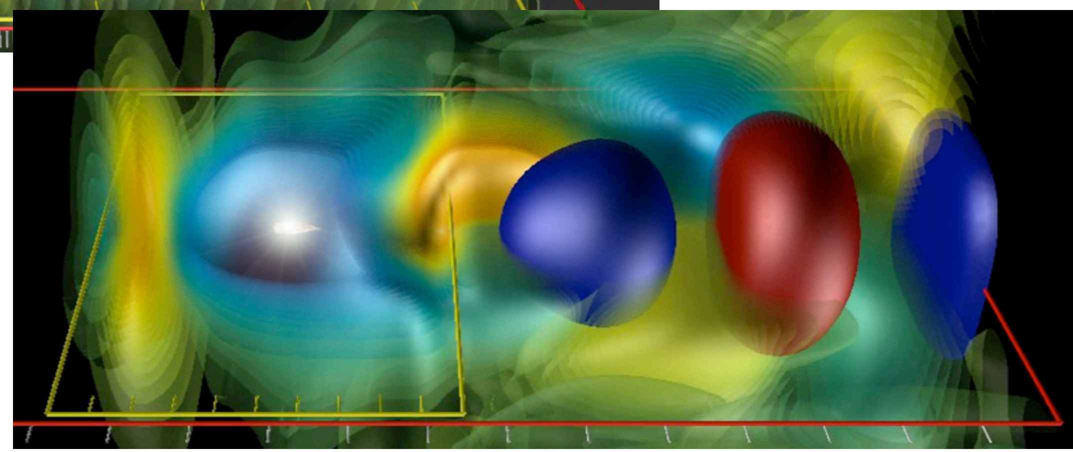
Pore deformation resulting from change in stress loading in a Lagrangian mechanics treatment (GEOS)



Combined model

Accelerator Science at the Exascale



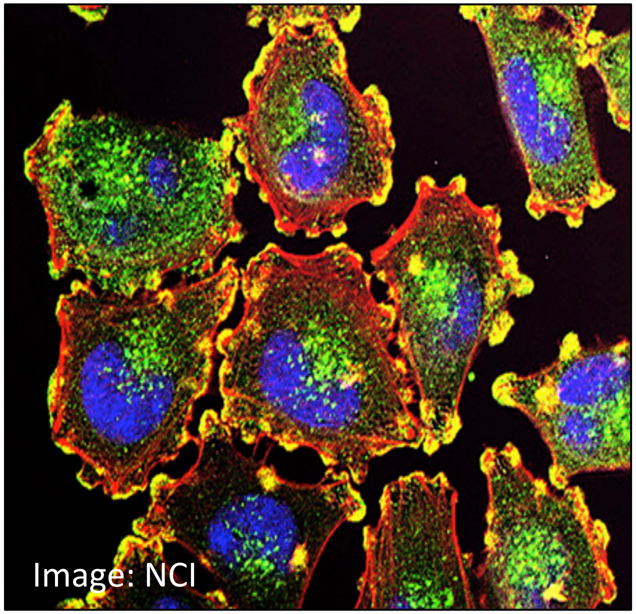


Simulation of laser-plasma acceleration with wavefronts of laser light (red and blue); the wake fields are accelerating (pale blue) or decelerating (orange). Right shows wake in "boosted" frame of reference.

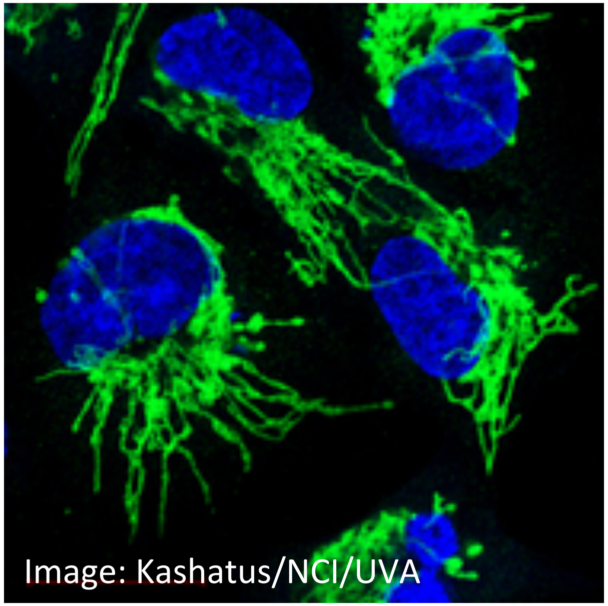
Exascale is needed to simulate future accelerators

Goal: Model a chain of up to a hundred plasma acceleration stages in a few days, for the design of a 1 TeV electron-positron high-energy collider

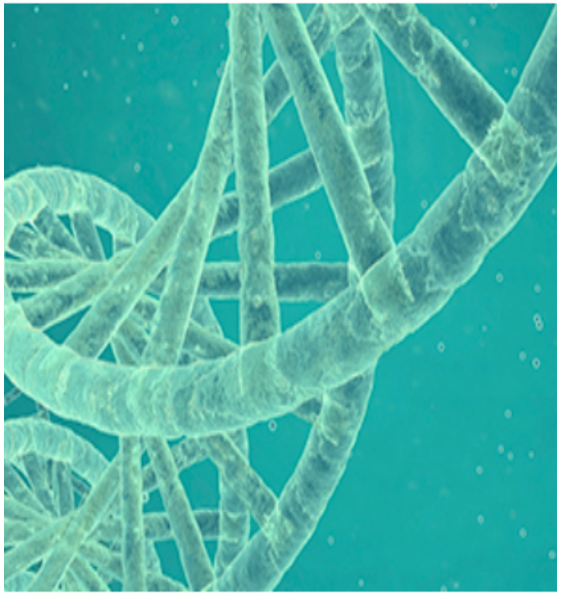
Cancer Analytics at the Exascale



Metastatic cancer classification and genetics improve treatment [Cell 2015]



One third of all cancers caused by mutations in RAS genes

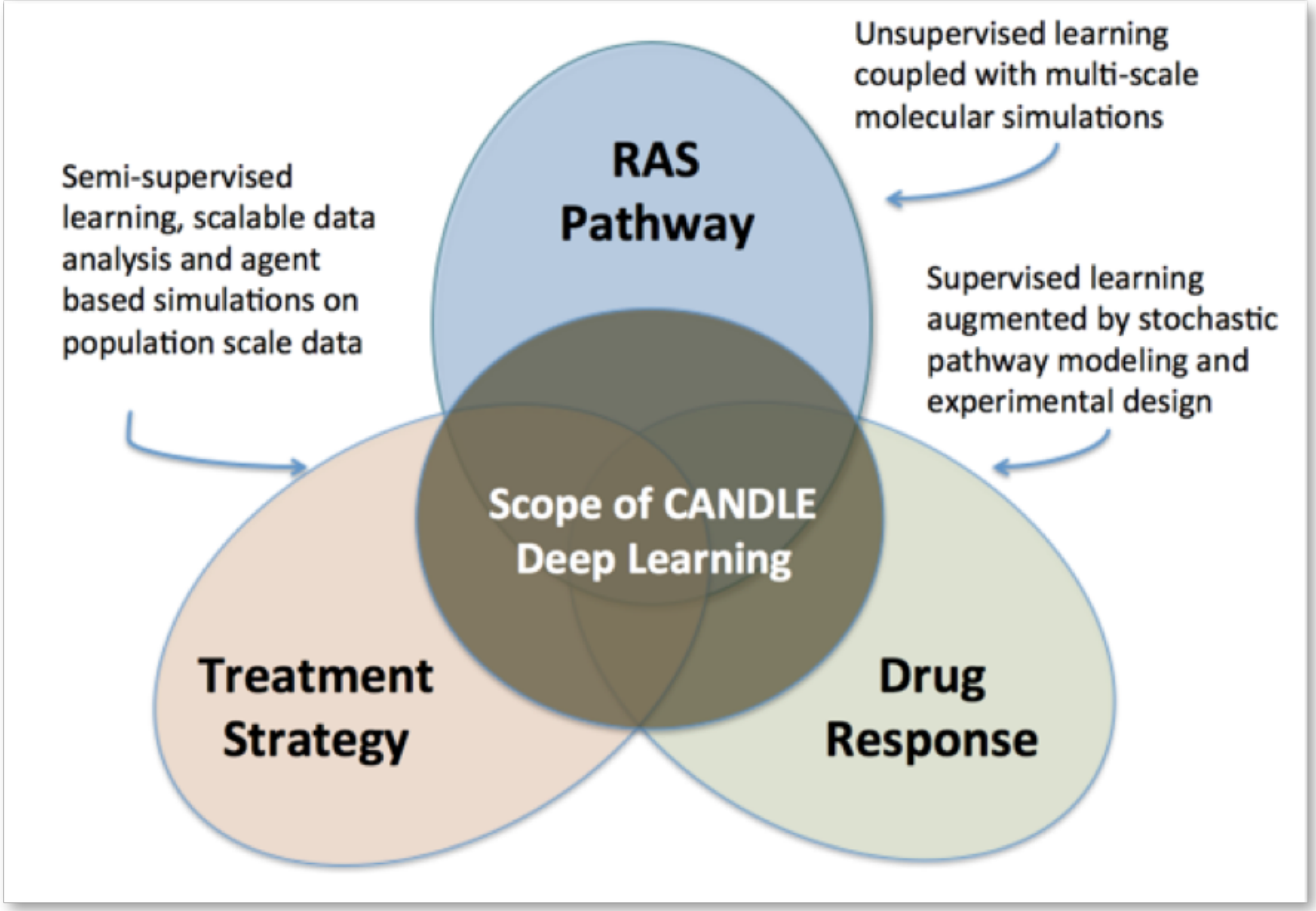


Combinatorial explosion with number of genomic features considered

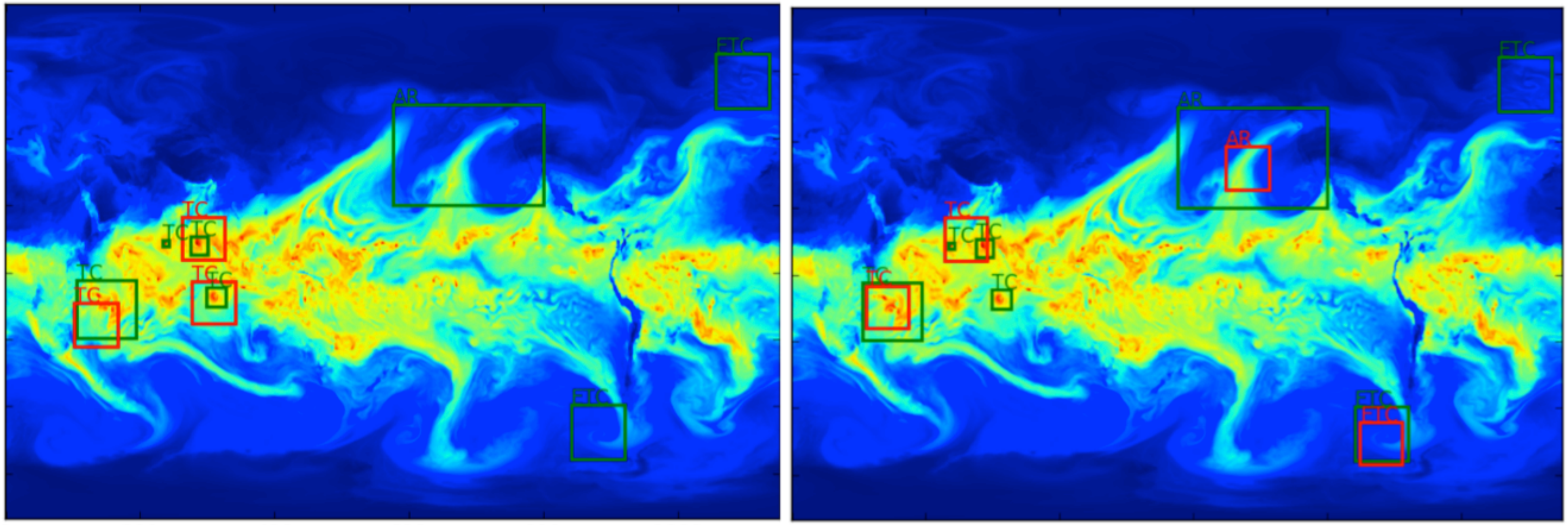
Exascale is needed to develop cell-specific interventions

Mapping genetic susceptibility to cancer and its outcomes; intracellular molecular signaling in complex mutational backgrounds; combine genetic, genomic, and clinical data

Cancer Analytics at the Exascale



Deep-Learning at Scale on HPC systems



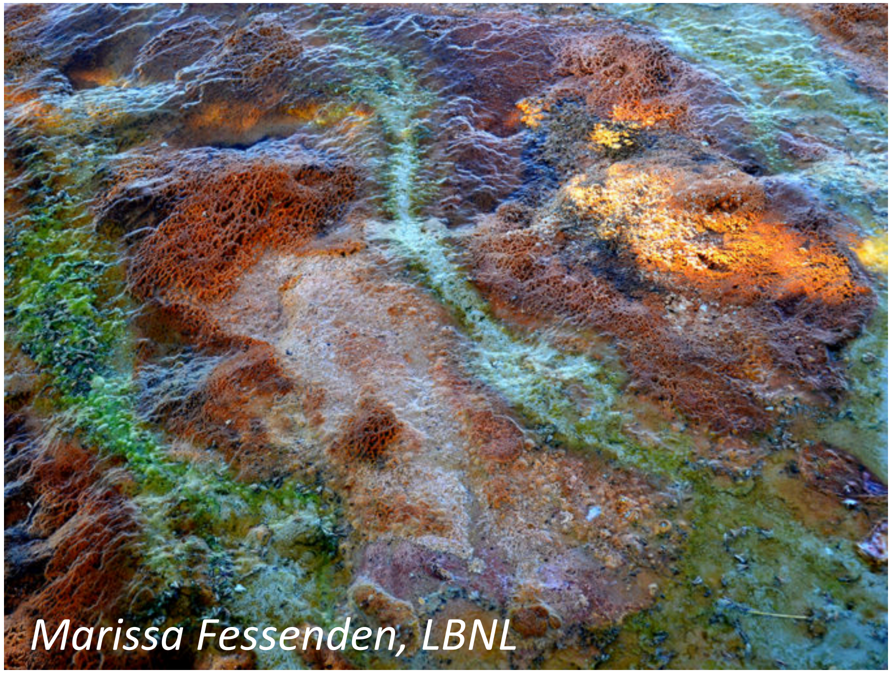
Identified extreme climate events using supervise (left) and semisupervised (right) deep learning. Green = ground truth, Red = predictions (confidence > 0.8). [NIPS 2017]

Deep Learning at 15 PF on NERSC Cori (Cray + Intel KNL)

- Trained in 10s of minutes on 10 terabyte datasets, millions of Images
- 9600 nodes, optimized on KNL with IntelCaffe and MKL (NERSC / Intel collaboration)
- Synch + Asynch parameter update strategy for multi-node scaling (NERSC / Stanford)

Evan Racah, Christopher Beckham, Tegan Maharaj, Samira Ebrahimi Kahou, Prabhat, Christopher Pal, Evan Racah (LBNL, Ecole Poly. Montreal, Microsoft)

Genome Science at the Exascale



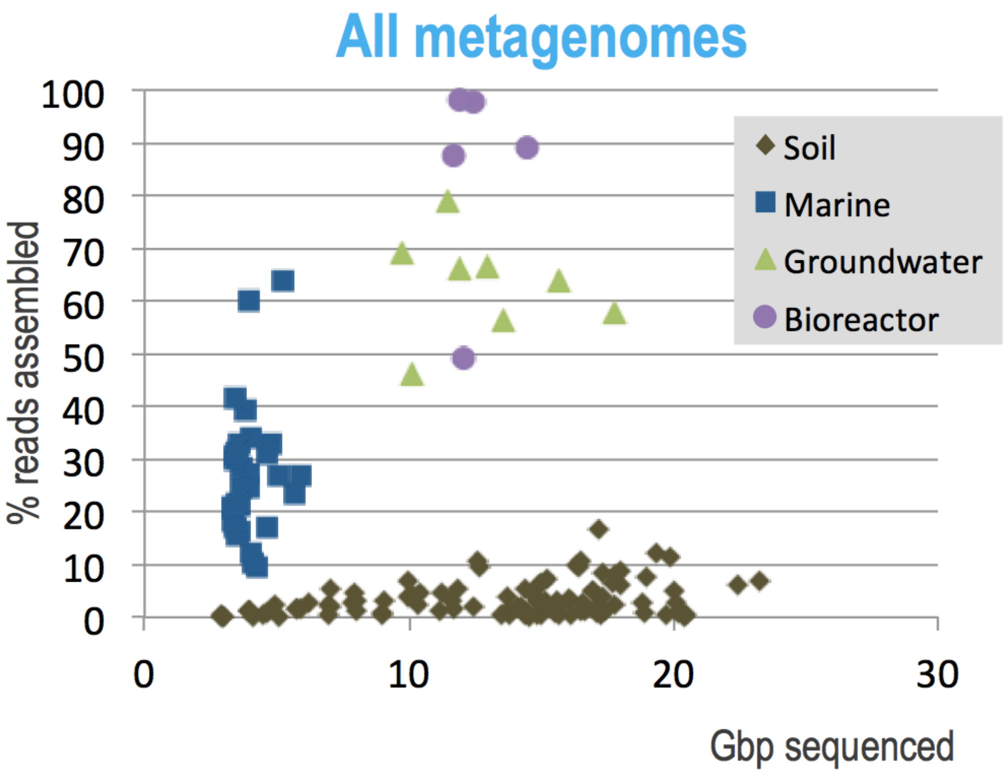
Thermophilic microbial mat in West Thumb Geyser Basin, Yellowstone National Park

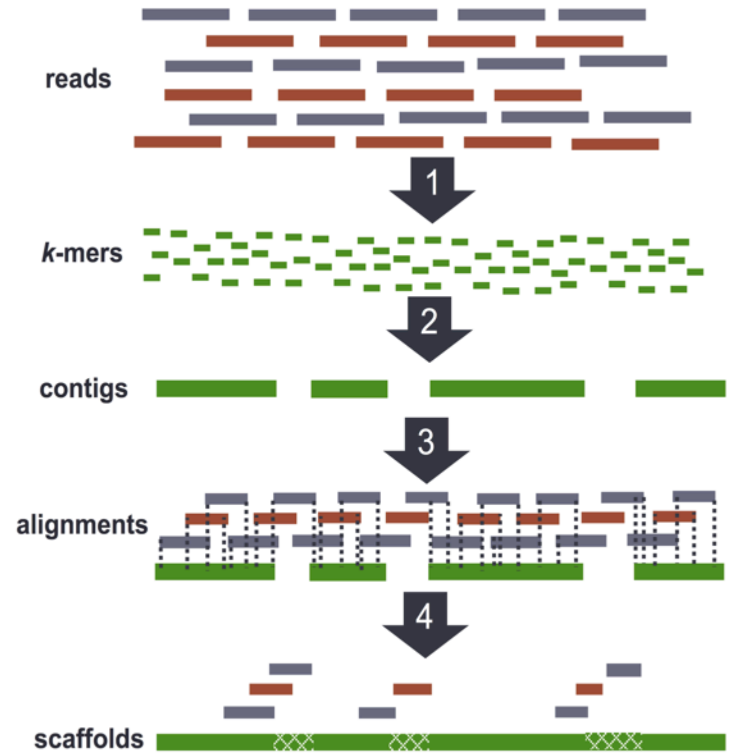
Compact CRISPR systems found in deep underground Crystal Geyser bacteria (Banfield)

Exascale is needed to characterize microbial communities

Metagenome analysis with high performance assembly and machine learning; identify gene clusters for energy, environment, biomanufacturing and health

Environment: orders of magnitude harder than humans





De novo genome assembly

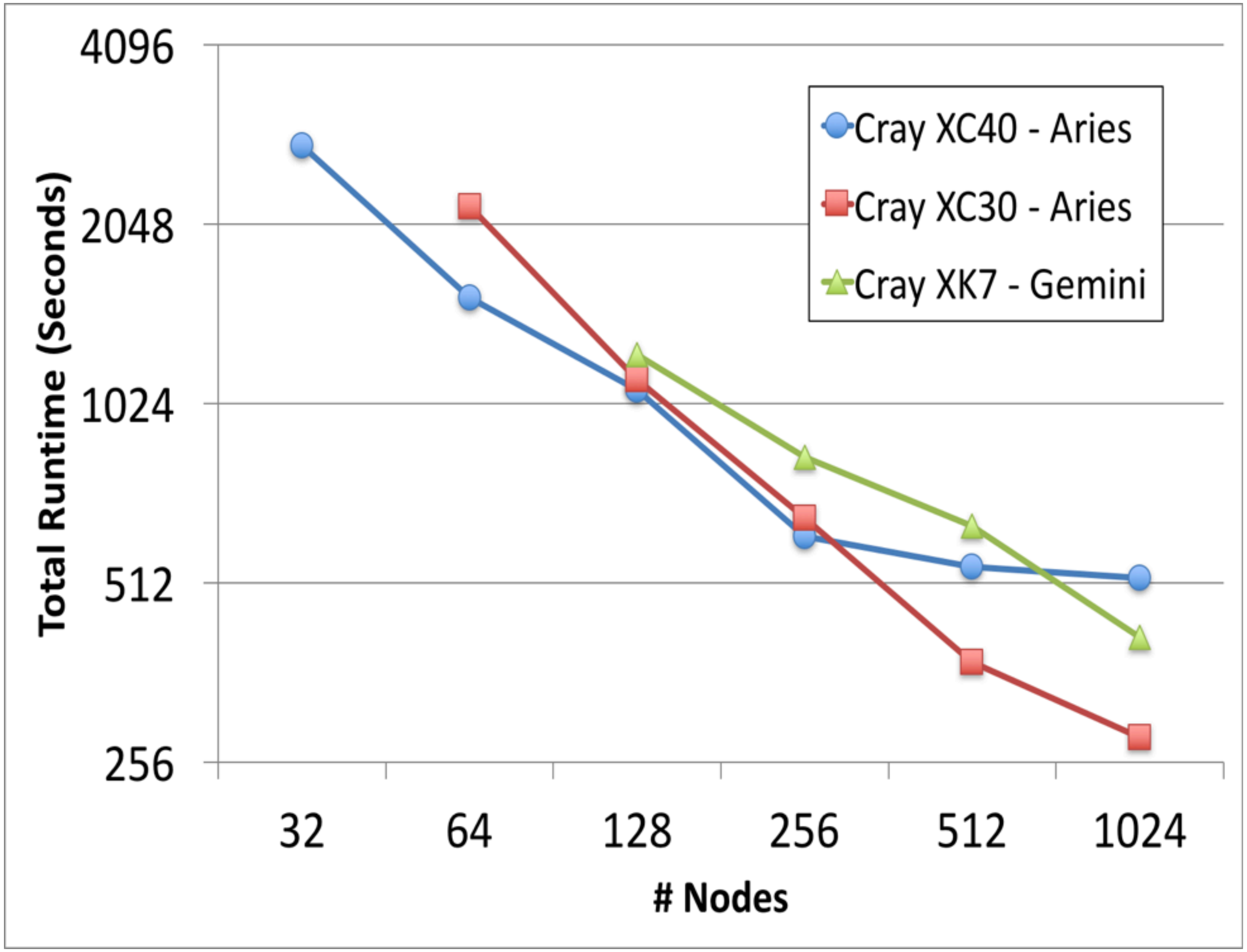
Read multiple times. Chop reads into k-mers	Histogram k-mers (eliminate errors)	DFS walk k-mer graph (stored as hash table)	Various graph operations (more hash tables)

Multi-Node Strong Scaling

- HipMer scales efficiently to 100’s and 1000’s of nodes

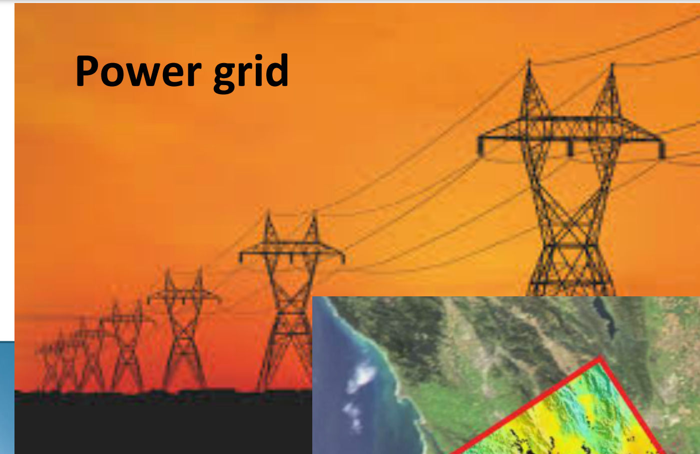
Human Genome Results (small problem)

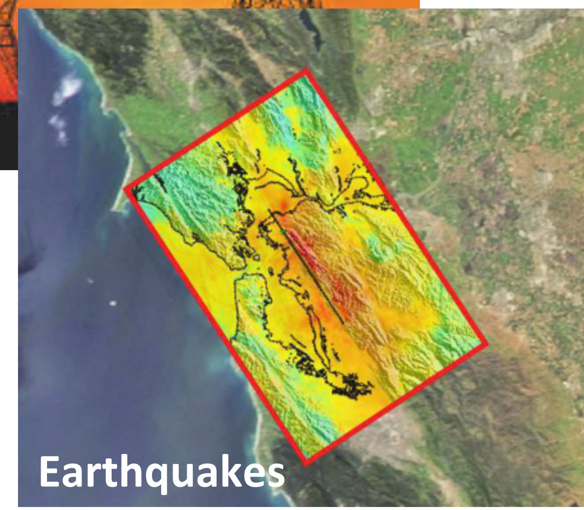
- Minimum aggregate memory required
- Scales linearly on node, KNL (68 cores)
- Requires high injection rate, low latency
- Would benefit from remote hardware atomics



Ellis, Georganas, Egan, Hofmeyr, Buluc, Cook, Oliker, Yelick [Europar ‘17]

Exascale Science in analytics from embedded sensors





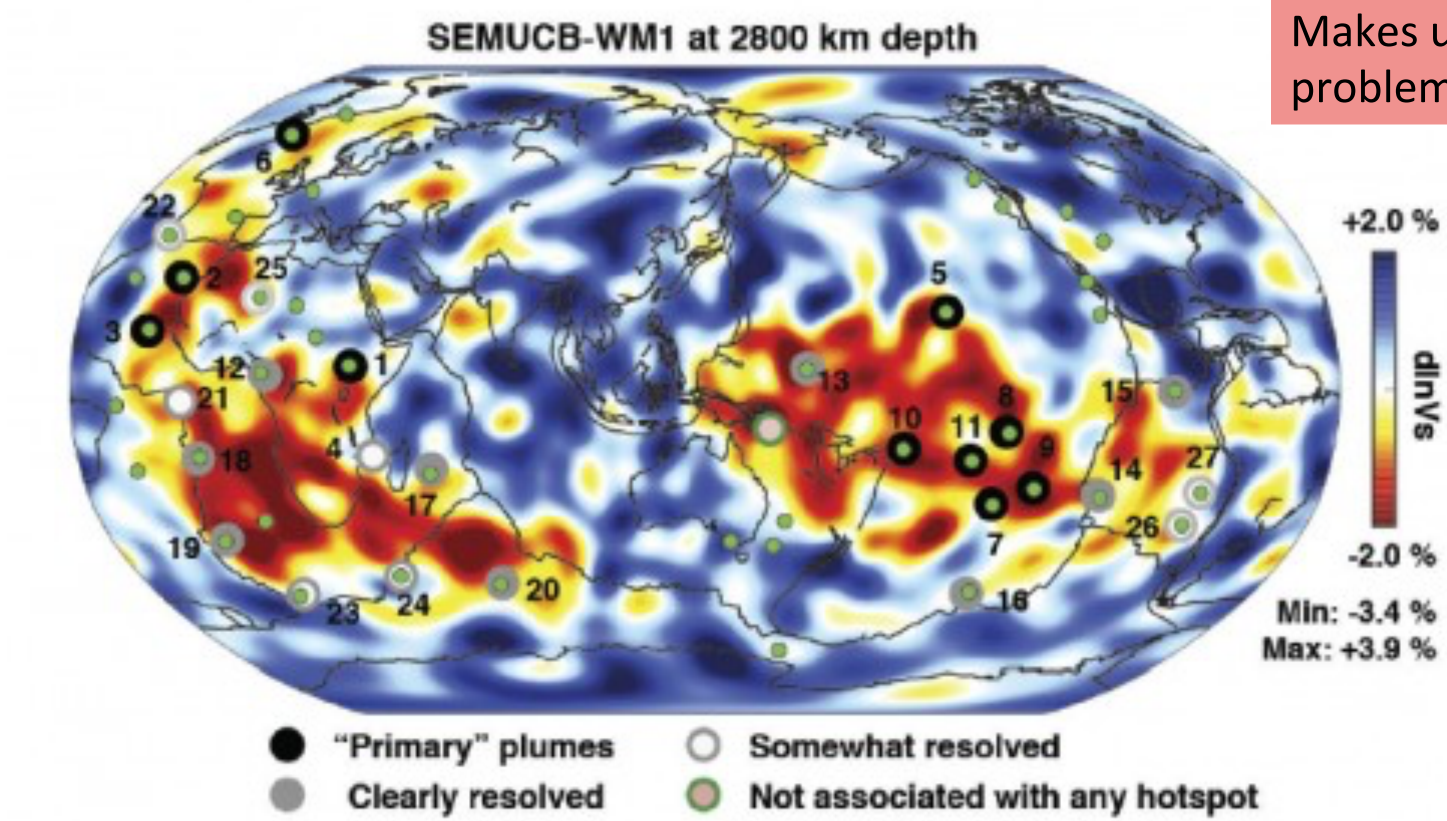
Exascale simulation and combined analytics

Infrastructure planning	Scenario analysis, e.g., emergency response	Behavioral analysis, human in the loop	Policy and ecnomics

Whole-Mantle Seismic Model Using

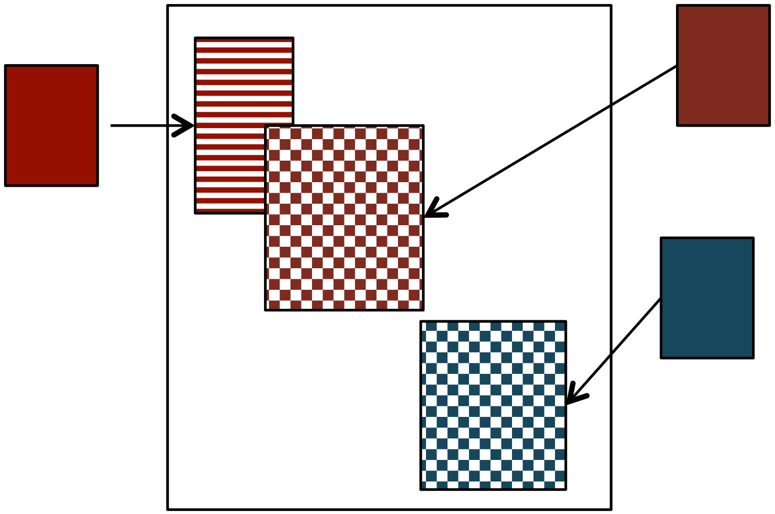
- *First-ever whole-mantle seismic model from numerical waveform tomography*
- *Finding: Most* volcanic hotspots are linked to two spots on the boundary between the metal core and rocky mantle 1,800 miles below Earth's surface.

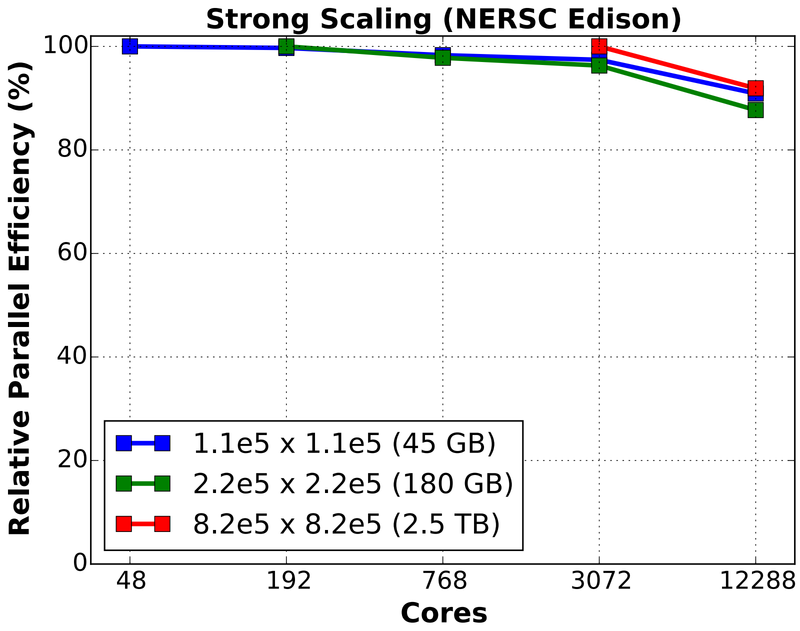
Makes unsolvable problems solvable!



Scott French, Barbara Romanowicz, **Nature, 2015**

Data Fusion for Observation with Simulation



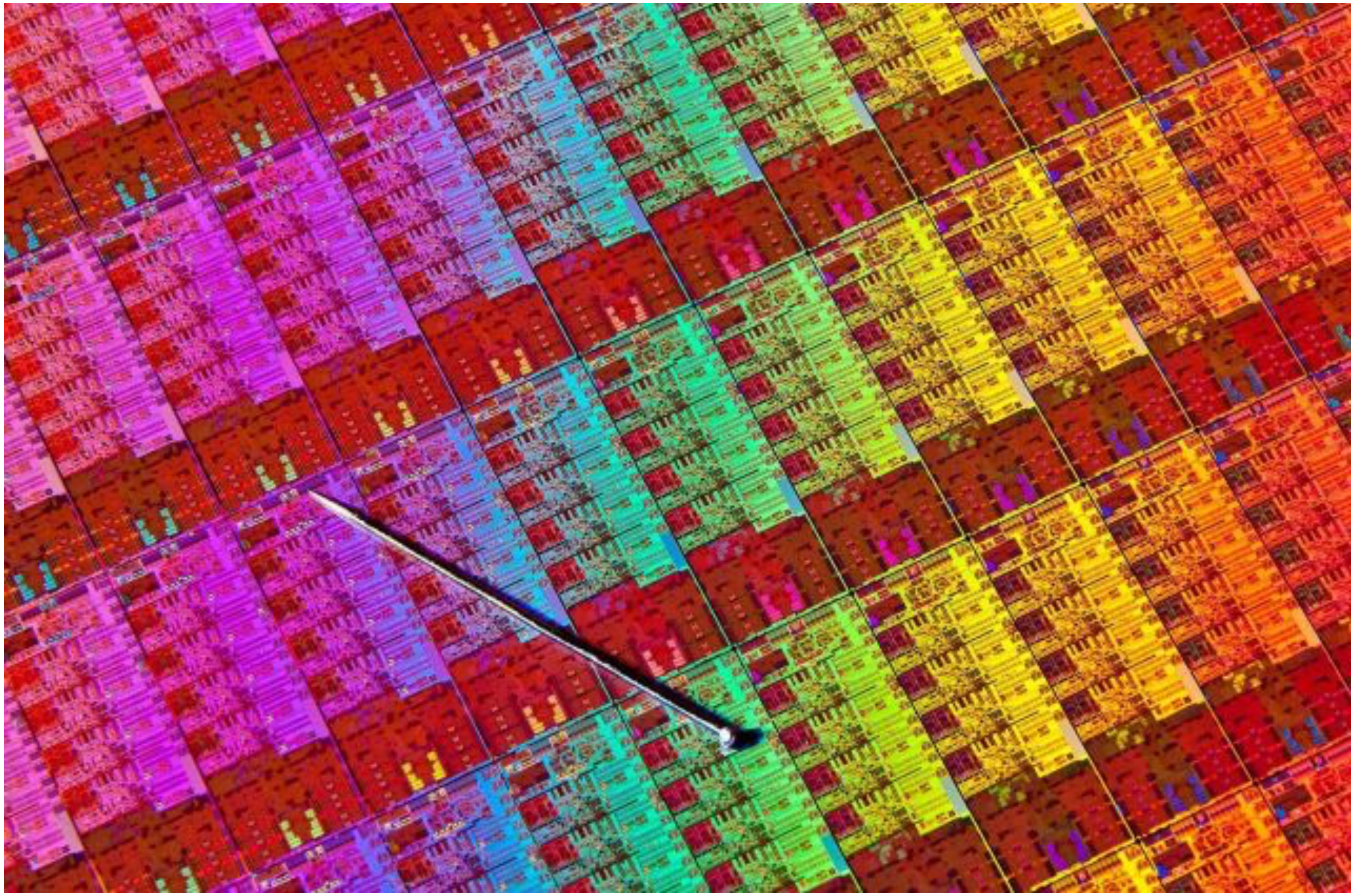


- **Unaligned data from observation**
- **One-sided strided updates**

Scott French, Y. Zheng, B. Romanowicz, K. Yelick challenges

Computer Science ~~breakthroughs~~ at the Exascale

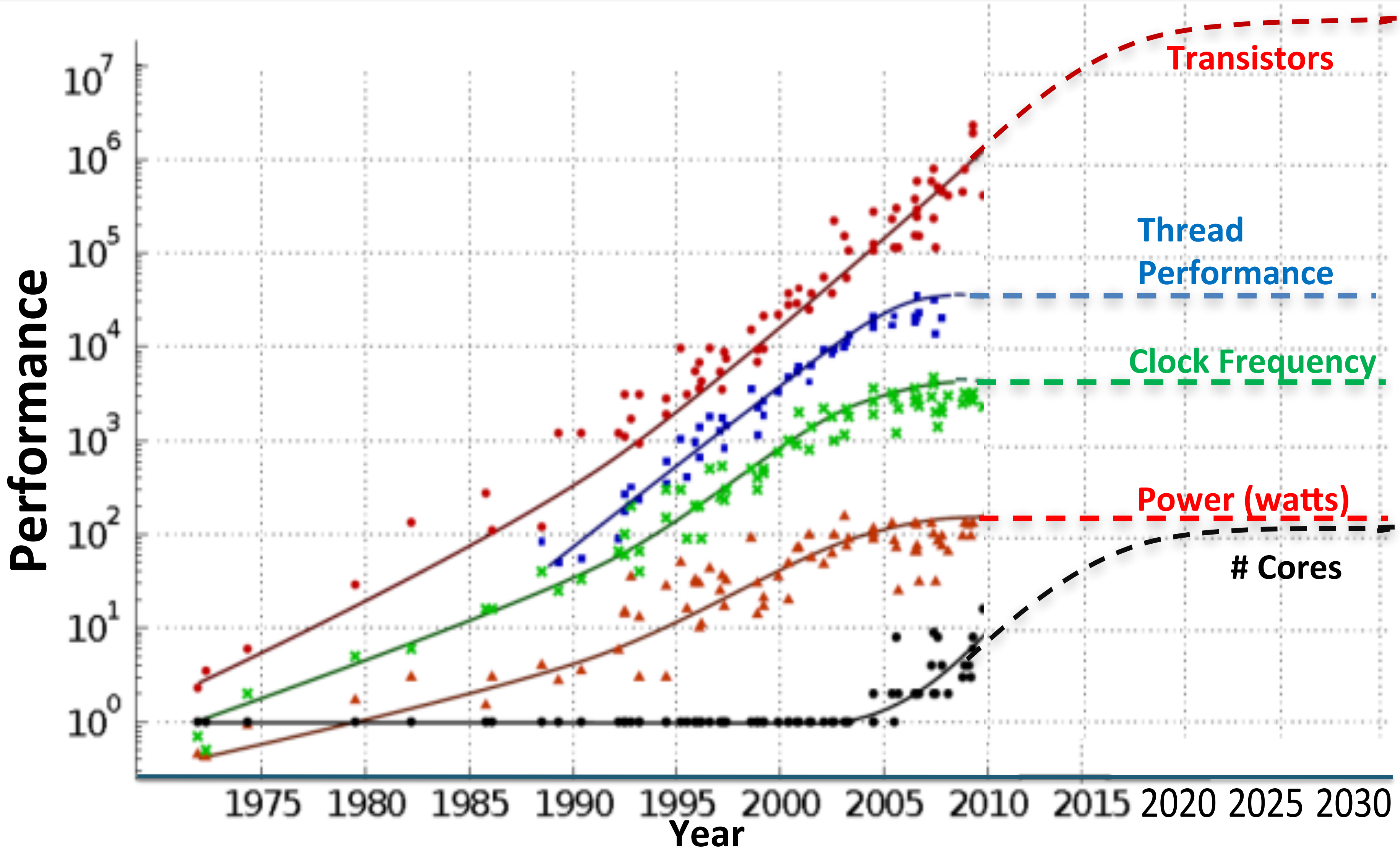
End of Transistor Density Scaling



ITRS now sets the end of transistor shrinking to the year 2021

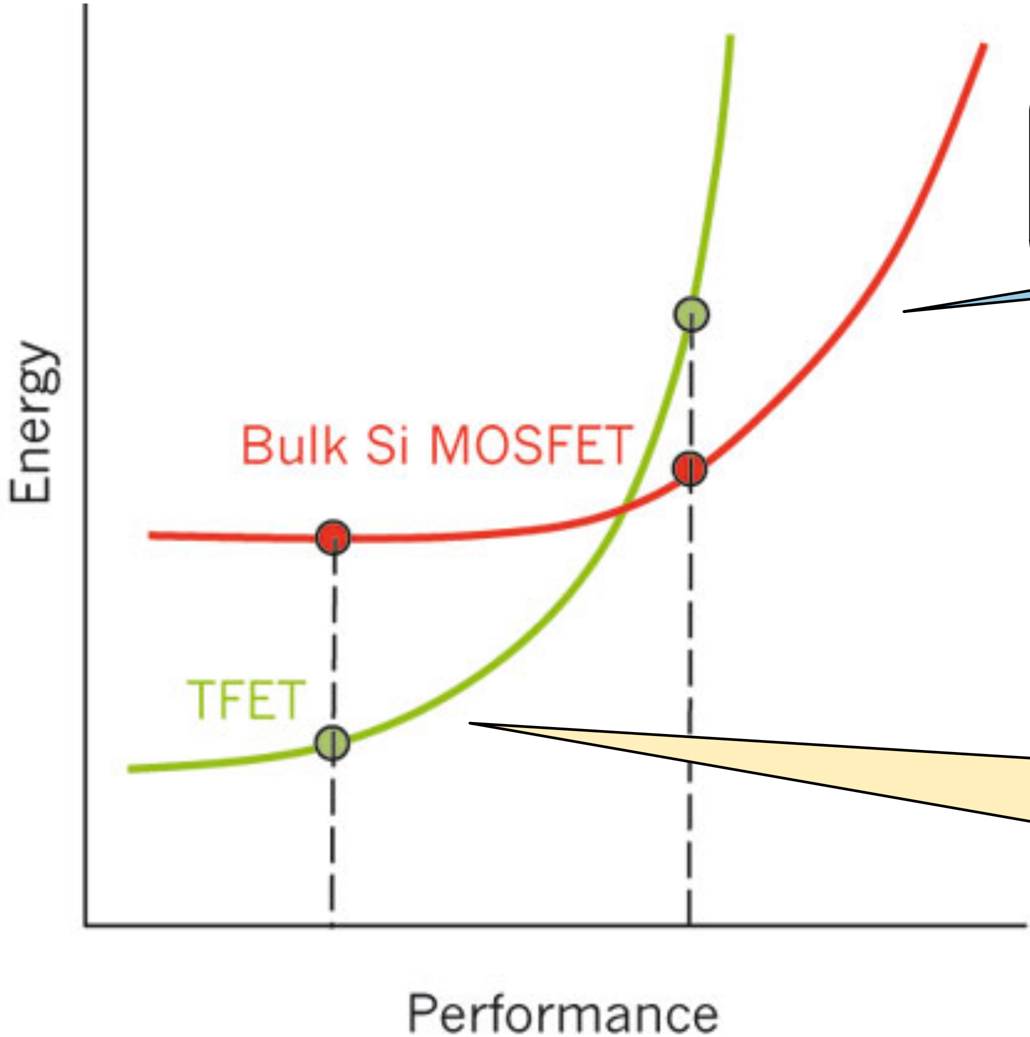
Technology Scaling Trends

The many ends of "Moore's" Law



Source: John Shalf and Kunle Olukotun, Lance Hammond, Herb Sutter, and Burton Smith

Device alternatives require lower clock → more parallelism

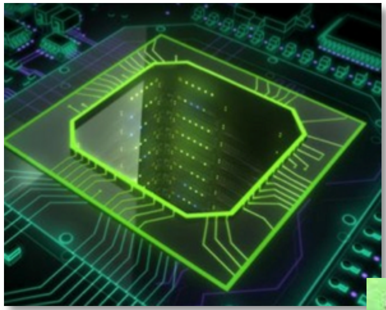


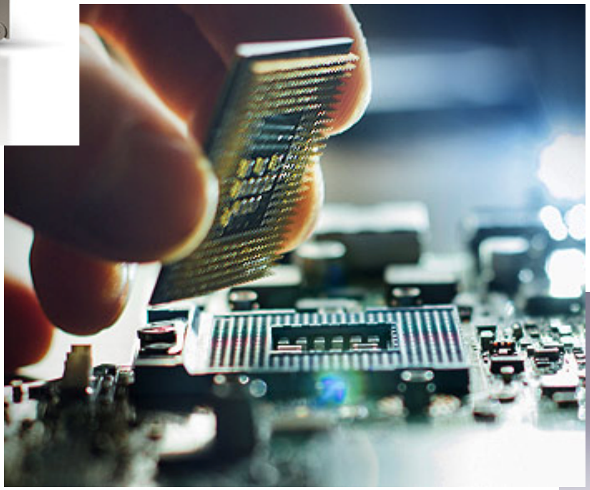
Today's CMOS Technology

Tunneling FET advantage ***only at low clock rates***

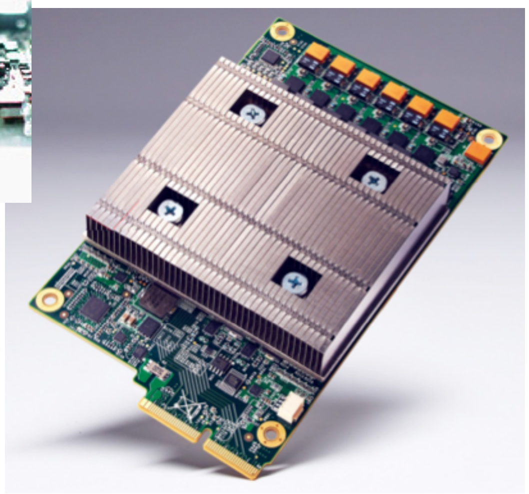
Specialization: End Game for Moore's Law

NVIDIA builds deep learning appliance with P100 Tesla's





Intel buys deep learning startup, Nervana



Google designs its own Tensor Processing Unit (TPU)

The biggest concern for Exascale application developers is the need to write and maintain multiple versions of their software and the uncertainty over what the architectures will be.

Problem 1: Languages

Why develop new languages?

- **Productivity: higher level syntax**
  - We need a language
- **Correctness: static analysis can eliminate errors**
  - We need a compiler (front-end)
- **Performance: optimizations**
  - We need a compiler (back-end)

Language design enforces clarity in concepts

- **But you need to "know your audience"**
  - Need to rewrite installed base of code (anti-productivity)
  - Risk of compiler disappearing (maintainability)
  - Syntax matters (familiarity)
- **Language adoption is often about its libraries**

Real-Time MRI Challenge

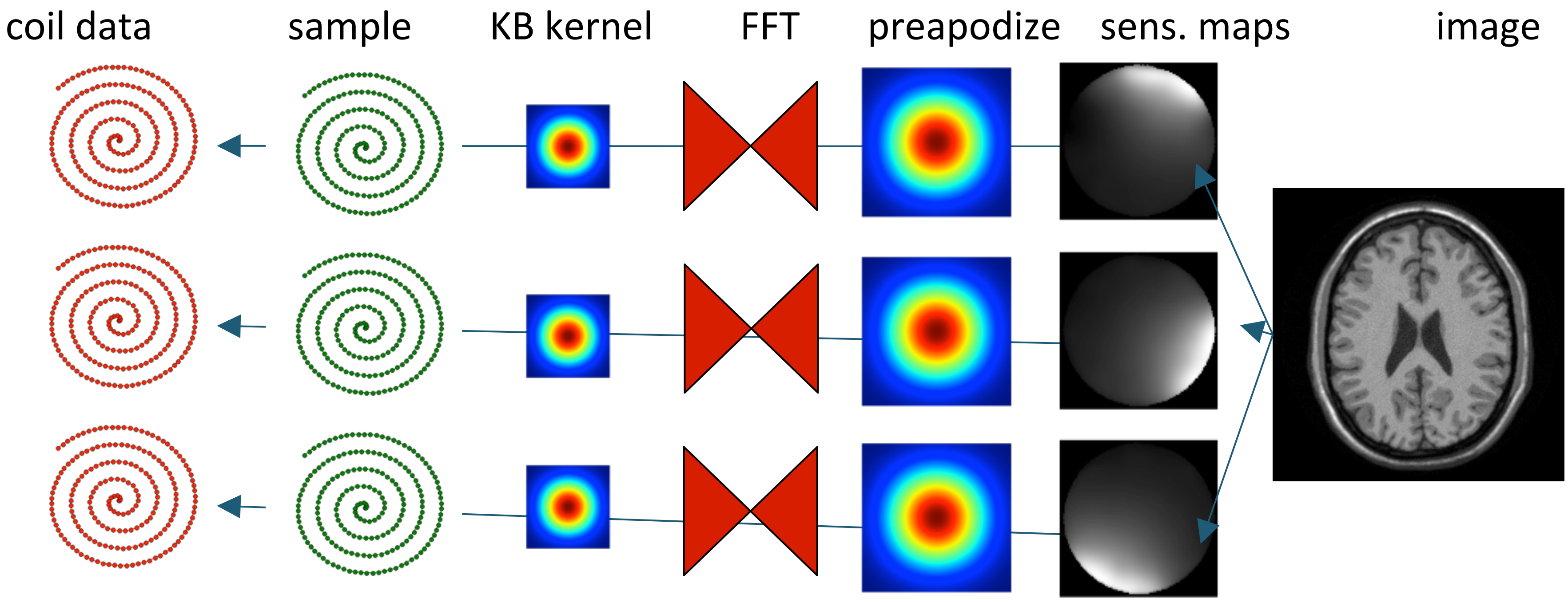
3 min goal

Time (min)	Architecture
6.15	KNL
5.42	Ivy Bridge
4.47	Broadwell
4.31	Kepler
4.12	Haswell
3.71	Broadwell
3.16	Kepler
0.94	Pascal

Michael Driscoll HPC optimization

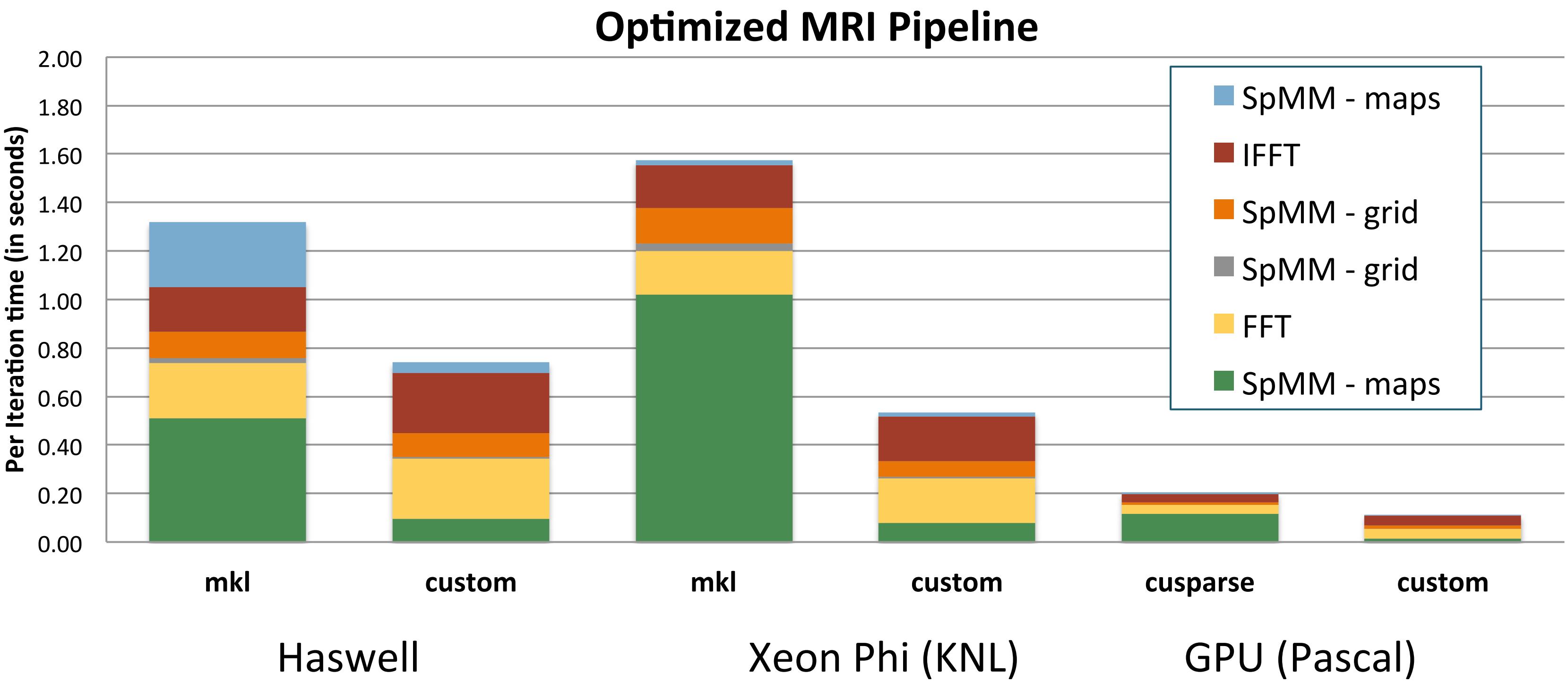
Compressed Sensing Approach by Mike Lustig et al
MRI results Wenwen Jiang

Matrix-free (loop optimization) vs. Matrix-full



Loops	Structured matrices	Matrices
Operators as loop nests	Operators as matrices with structure that compiler can optimize	Operators as arbitrary sparse matrices

Python-Based Domain-Specific Language (EDSL)



- **Original Numpy code on Haswell: 87 sec/iteration**
- **Runtime optimization reorganize tree of operators (matrices + FFTs) cognizant of matrix structure**
- **Library or custom matrix kernels**

Problem 2: Compilers

Reports of our death have been greatly exaggerated

Auto-vectorization for vector processors

Auto-SIMDization

Auto-parallelization for SMPs

Auto-parallelism for attached accelerators

Auto-parallelization for HPC

Architecture difficulty (related to granularity of parallelism)

Autotuner code generation

Domain specific (in degrees of specificity)

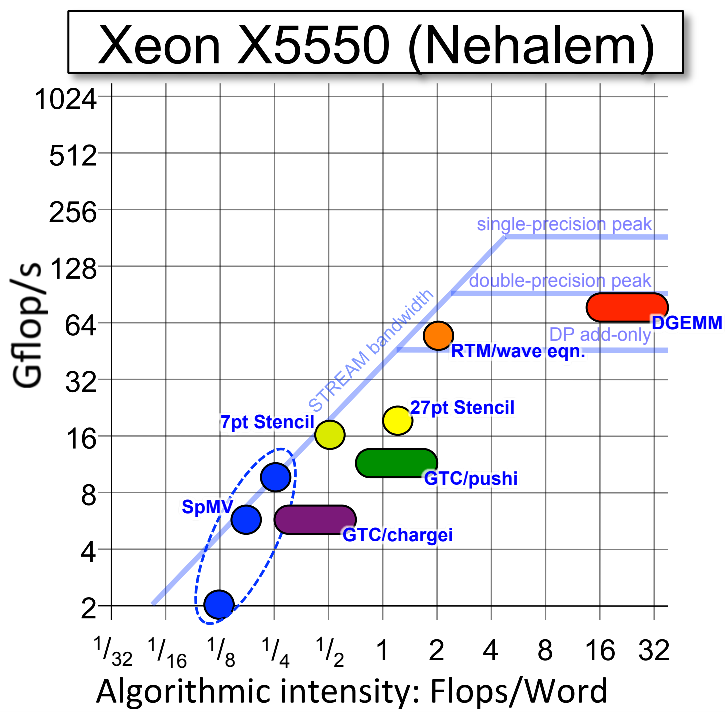
General purpose strongly typed

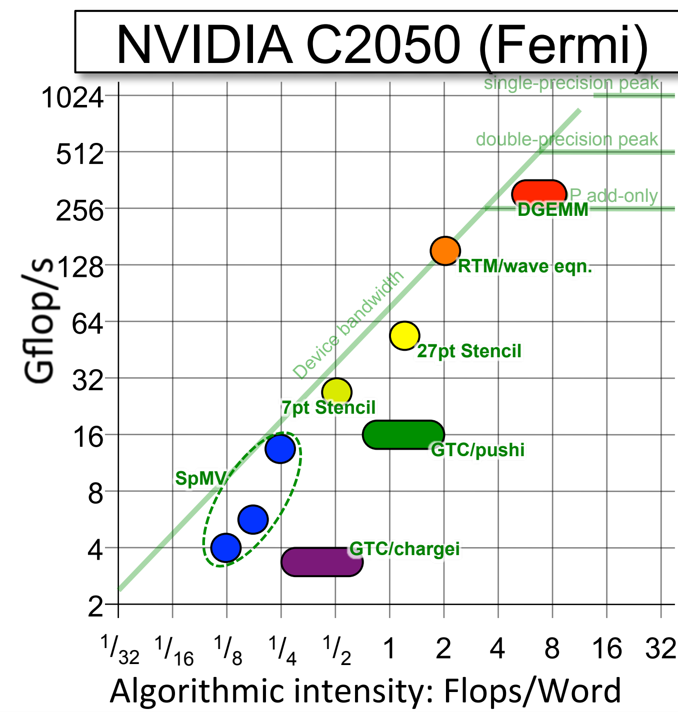
General purpose loosely typed

Language difficulty

Programming for diverse (specialized) architectures

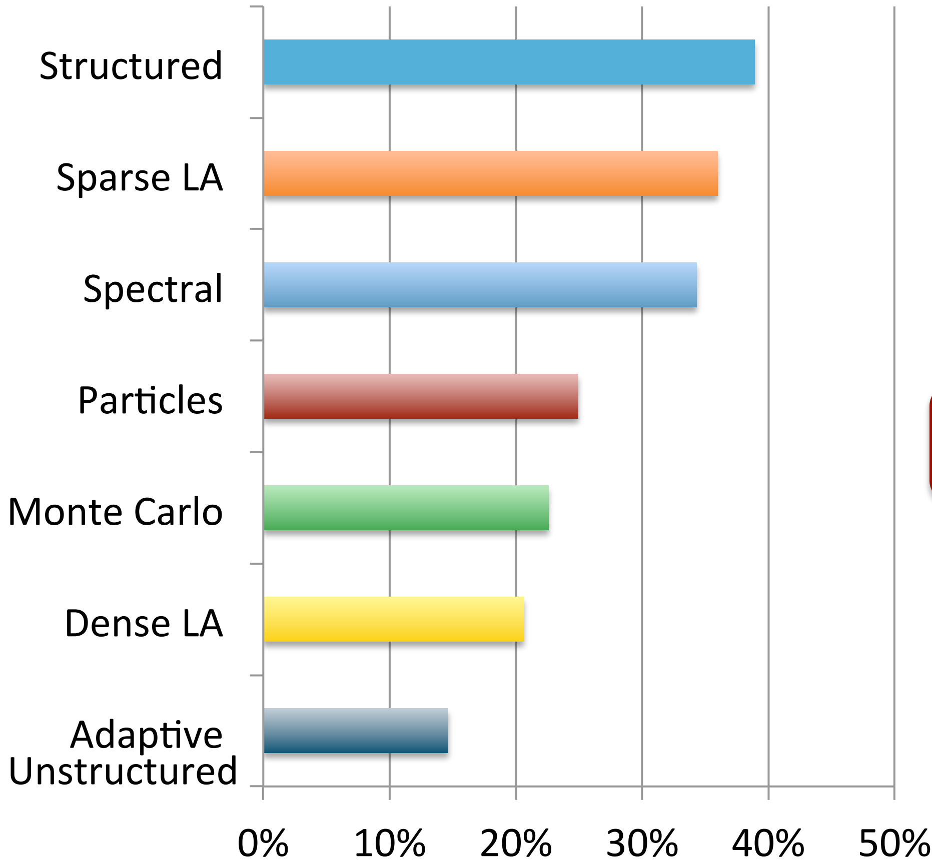
- **Two "unsolved" compiler problems:**
  - dependence analysis and — Domain-Specific Languages help with this
  - ✔ accurate performance models — Autotuning avoids this problem
- **Autotuners are code generators plus search**





Libraries vs. DSLs (domain-specific languages)

NERSC survey: what motifs do they use?



What code generators do we have?

Dense Linear Algebra	Atlas
Spectral Algorithms	FFTW, Spiral
Sparse Linear Algebra	OSKI
Structured Grids	TBD
Unstructured Grids	
Particle Methods	
Monte Carlo	

Stencils are both the most important motifs and a gap in our tools

DSLs popular outside scientific computing

Developed for Image Processing

- 10+ FTEs developing Halide
- 50+ FTEs use it; > 20 kLOC

HPGMG (Multigrid on Halide)

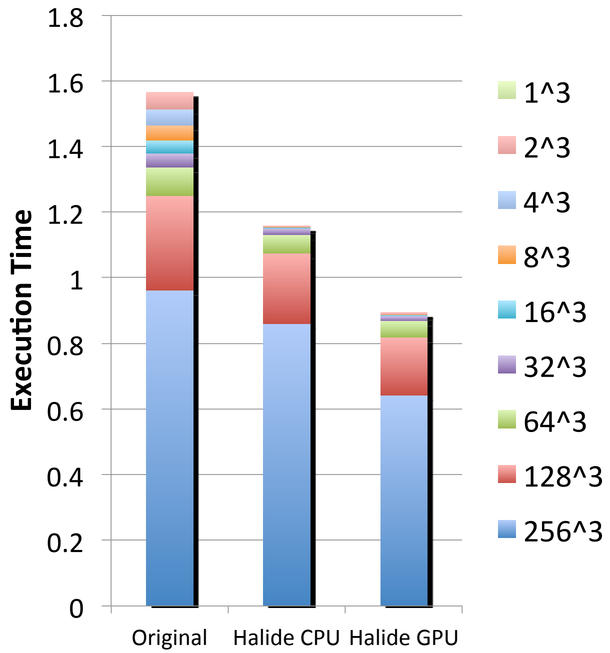
- **Halide Algorithm by domain expert**

```
Func Ax_n("Ax_n"), lambda("lambda"), chebyshev("chebyshev");
Var i("i"),j("j"),k("k");
Ax_n(i,j,k) =  a*alpha(i,j,k)*x_n(i,j,k) - b*h2inv*(
    beta_i(i,j,k)  *(valid(i-1,j,k)*(x_n(i,j,k) + x_n(i-1,j,k)) - 2.0f*x_n(i,j,k))
  + beta_j(i,j,k)  *(valid(i,j-1,k)*(x_n(i,j,k) + x_n(i,j-1,k)) - 2.0f*x_n(i,j,k))
  + beta_k(i,j,k)  *(valid(i,j,k-1)*(x_n(i,j,k) + x_n(i,j,k-1)) - 2.0f*x_n(i,j,k))
  + beta_i(i+1,j,k)*(valid(i+1,j,k)*(x_n(i,j,k) + x_n(i+1,j,k)) - 2.0f*x_n(i,j,k))
  + beta_j(i,j+1,k)*(valid(i,j+1,k)*(x_n(i,j,k) + x_n(i,j+1,k)) - 2.0f*x_n(i,j,k))
  + beta_k(i,j,k+1)*(valid(i,j,k+1)*(x_n(i,j,k) + x_n(i,j,k+1)) - 2.0f*x_n(i,j,k)));
lambda(i,j,k) = 1.0f / (a*alpha(i,j,k) - b*h2inv*(
    beta_i(i,j,k)  *(valid(i-1,j,k) - 2.0f)
  + beta_j(i,j,k)  *(valid(i,j-1,k) - 2.0f)
  + beta_k(i,j,k)  *(valid(i,j,k-1) - 2.0f)
  + beta_i(i+1,j,k)*(valid(i+1,j,k) - 2.0f)
  + beta_j(i,j+1,k)*(valid(i,j+1,k) - 2.0f)
  + beta_k(i,j,k+1)*(valid(i,j,k+1) - 2.0f)));
chebyshev(i,j,k) = x_n(i,j,k) + c1*(x_n(i,j,k)-x_nm1(i,j,k))+
                   c2*lambda(i,j,k)*(rhs(i,j,k)-Ax_n(i,j,k));
```

- **Halide Schedule either**
  - Auto-generated by autotuning with opentuner
  - Or hand created by an optimization expert

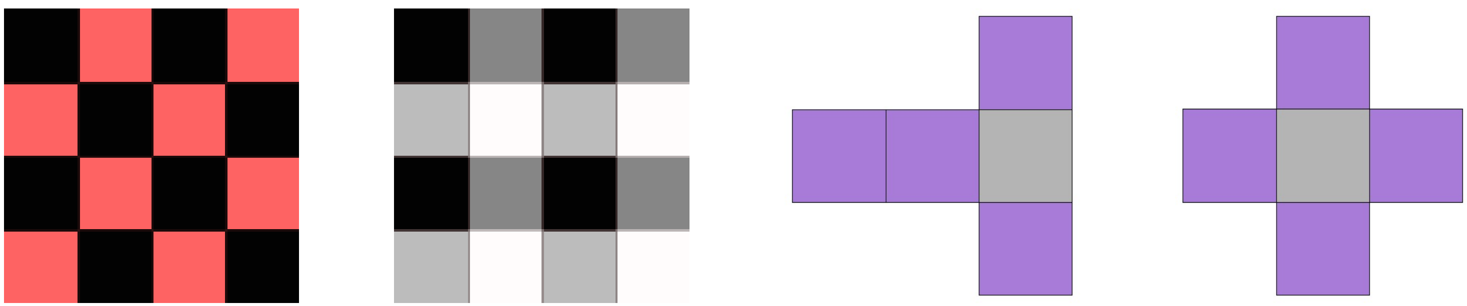
Halide performance

- Autogenerated schedule for CPU
- Hand created schedule for GPU
- No change to the algorithm



Approach: Small Compiler for Small Language

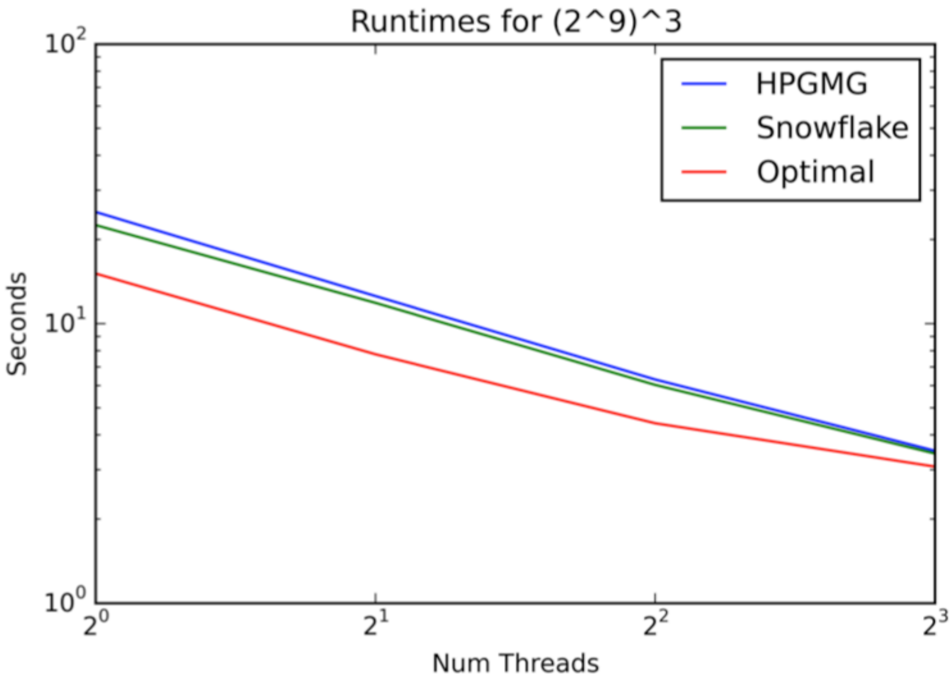
- **Snowflake: A DSL for Science Stencils**
  - Domain calculus inspired by Titanium, UPC++, and AMR in general

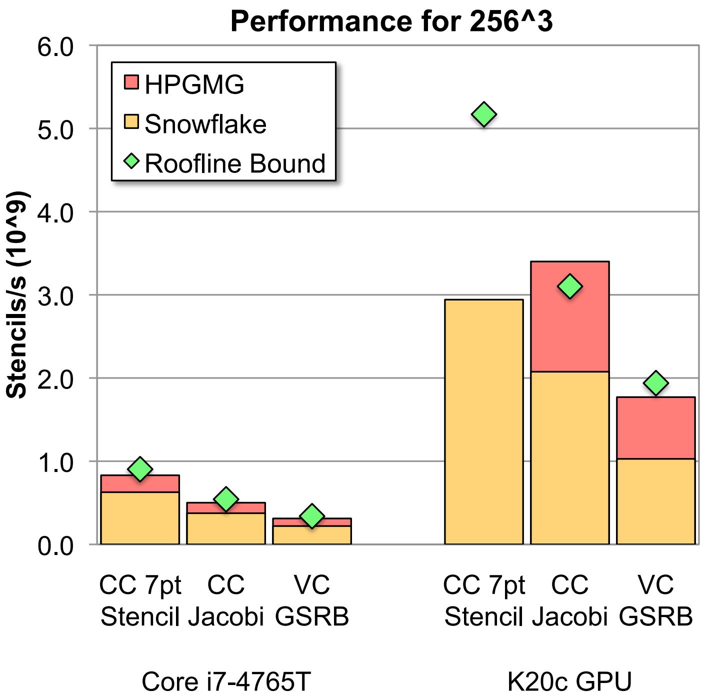


(a) Red-Black tiling (b) 4-color tiling (c) Asymmetric stencil used near mesh boundary (d) 5-point Jacobi stencil

- **Complex stencils: red/black, asymmetric**
- **Update-in-place while preserving provable parallelism**
- **Complex boundary conditions: key to Adaptive Meshes**

Snowflake performance





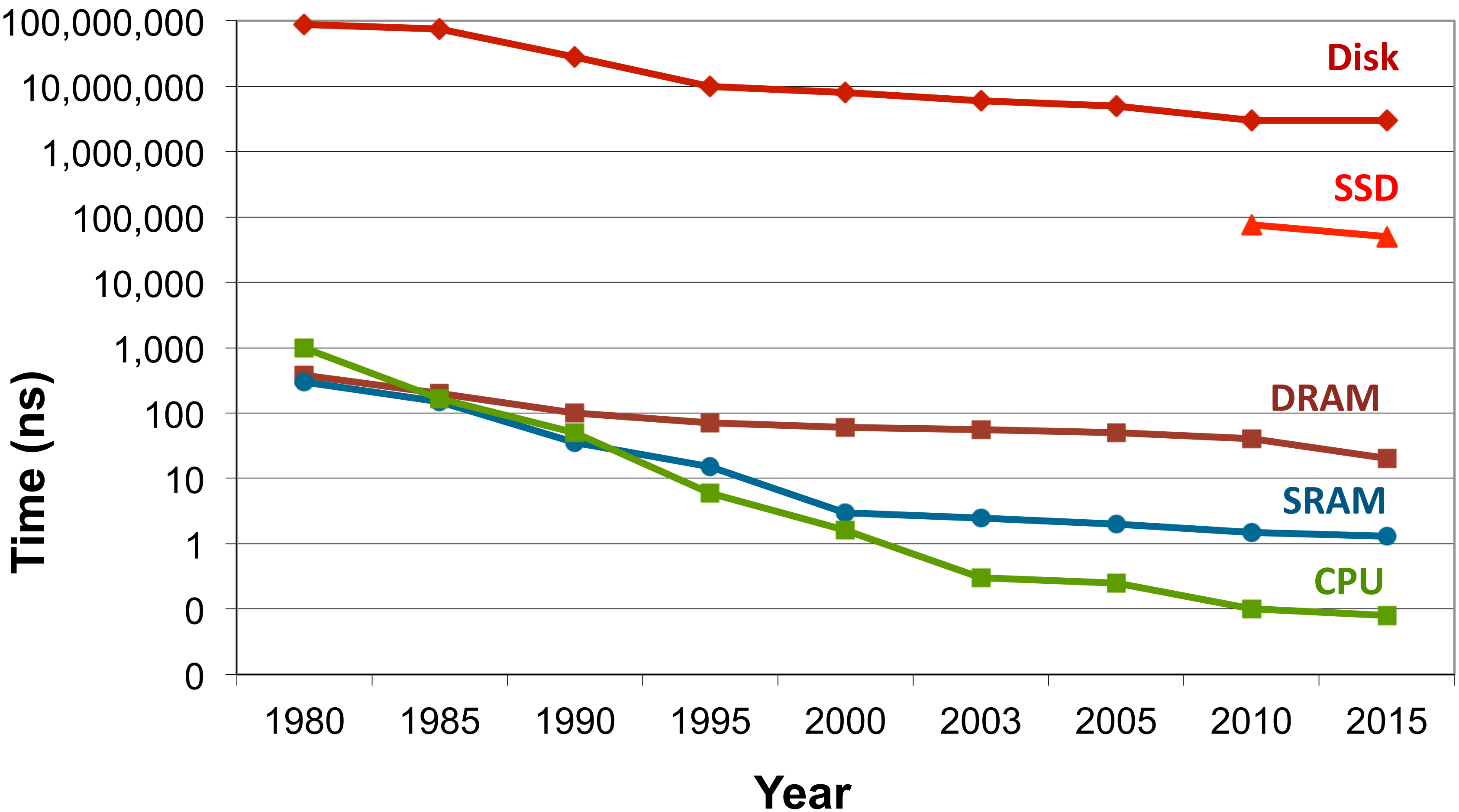
- **Performance on the HPGMG application benchmark using all the features of Snowflake**
- **Competitive with hand-optimized performance**
- **Within 2x of optimal roofline**

Open Problem: Compiling for communication optimality…

... with irregular loop nests and sparsity

Data Movement is Expensive

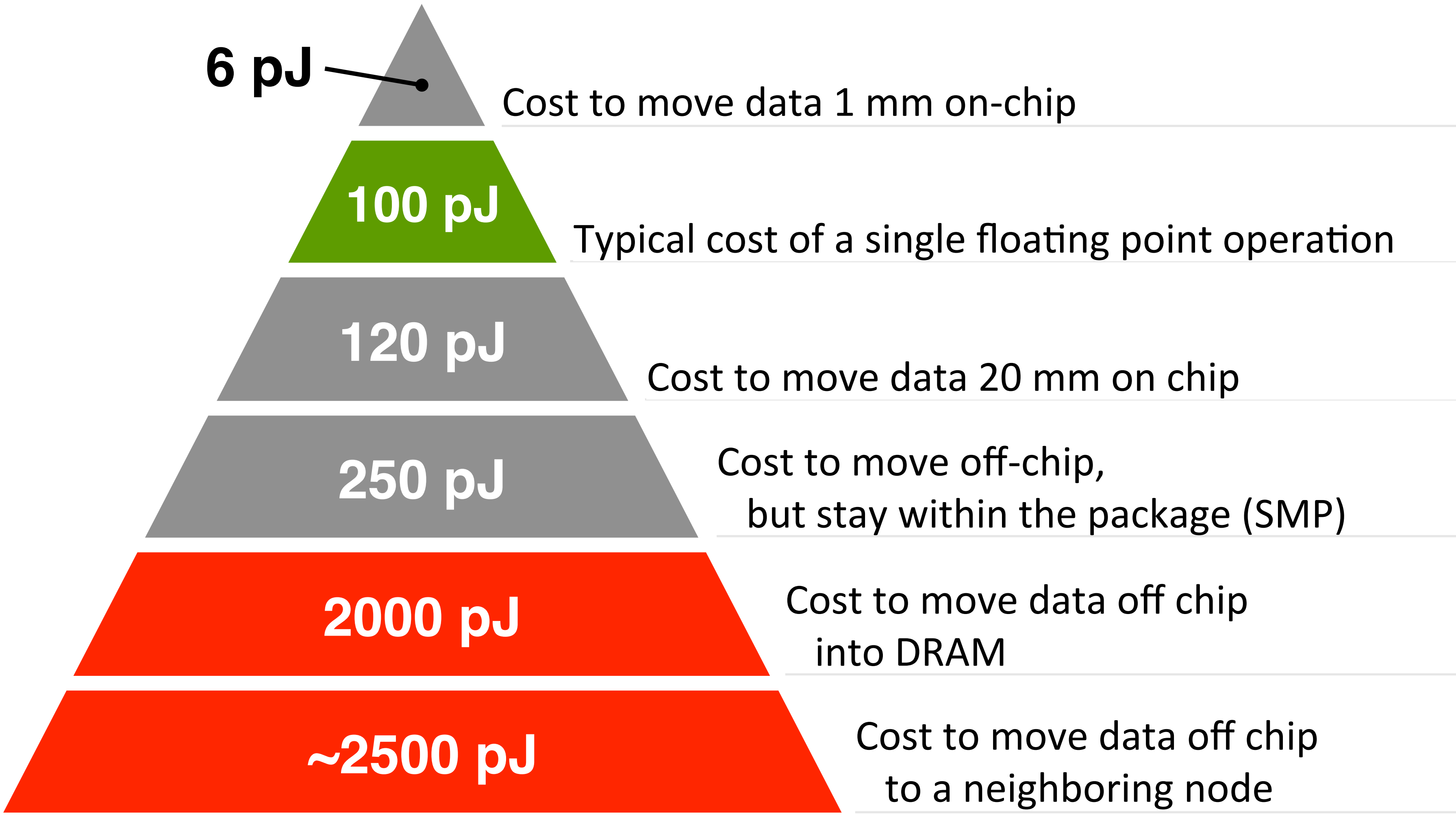
CPU cycle time vs memory access time



Source: http://csapp.cs.cmu.edu/2e/figures.html, http://csapp.cs.cmu.edu/3e/figures.html

Data Movement is Expensive

Hierarchical energy costs.



Beyond Domain Decomposition

2.5D Matrix Multiply on BG/P, 16K nodes / 64K cores

Surprises:

- Even Matrix Multiply had room for improvement
- Idea: make copies of C matrix (as in prior 3D algorithm, but not as many)
- Result is provably optimal in communication

Lesson: Never waste fast memory

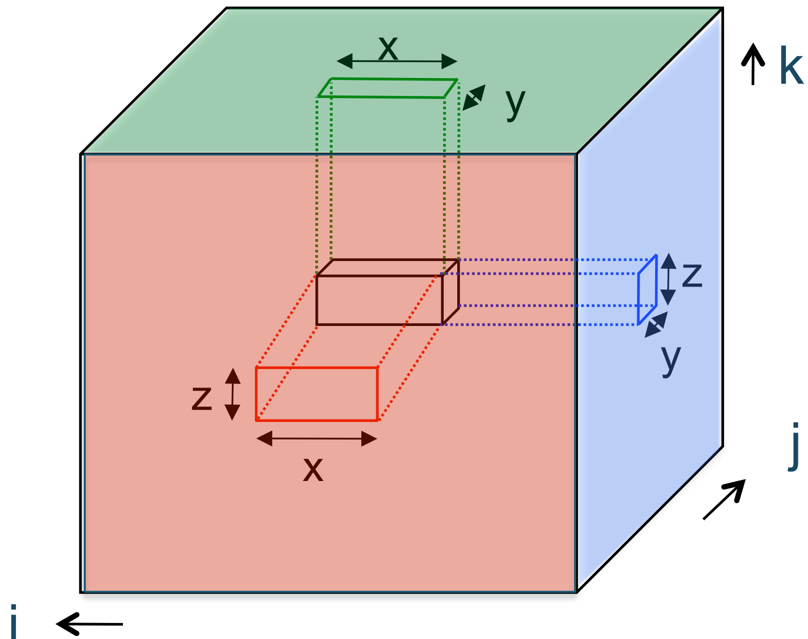
And don't get hung up on the owner computes rule

Can we generalize for compiler writers?

EuroPar'11 (Solomonik, Demmel)
SC'11 paper (Solomonik, Bhatele, Demmel)

Deconstructing 2.5D Matrix Multiply

Solomonick & Demmel



- Tiling the iteration space
- 2D algorithm: never chop k dim
- 2.5 or 3D: Assume + is associative; chop k, which is → replication of C matrix

Matrix Multiplication code has a 3D iteration space
Each point in the space is a constant computation (*/+)

```
for i
  for j
    for k
        C[i,j] …  A[i,k] …   B[k,j]  …
```

Using .5D ideas on N-body

- **n particles, k-way interaction.**
  – Molecules, stars in galaxies, etc.
- **Most common: 2-way N-body**

for t timesteps

  forall $i_1, \ldots, i_k$

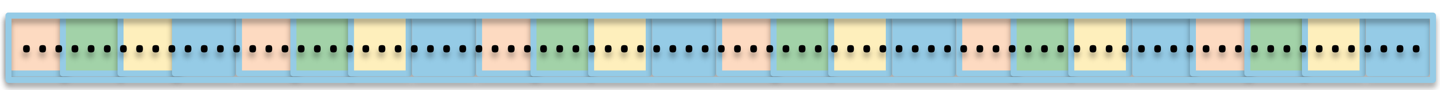
    $\boxed{\text{force}[i_1] \mathrel{+}= \text{interact}(\text{particle}[i_1], \ldots, \text{particle}[i_k])}$

  forall i

    move(particle[i], force[i])

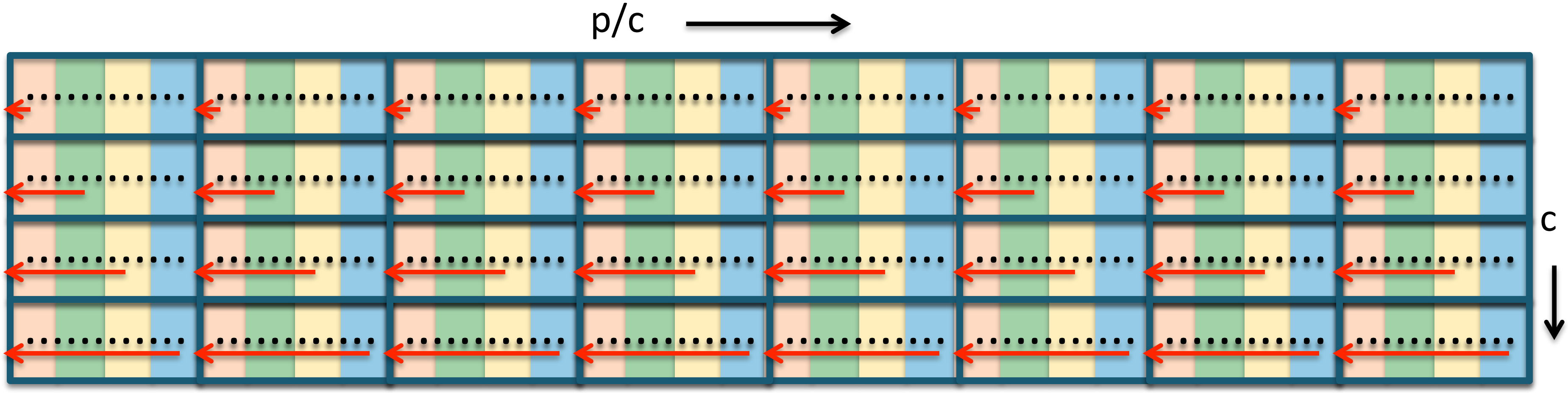
$O(n^k)$.

- **Best algorithm is to divide n particles into p groups??**



No!

Communication Avoiding 2-way N-body
(using a “1.5D” decomposition)



- **Divide p into c groups**
- **Replicate particles across groups**
- **Repeat: shift copy of n/(p*c) particles to the left within a group**
- **Reduce across c to produce final value for each particle**

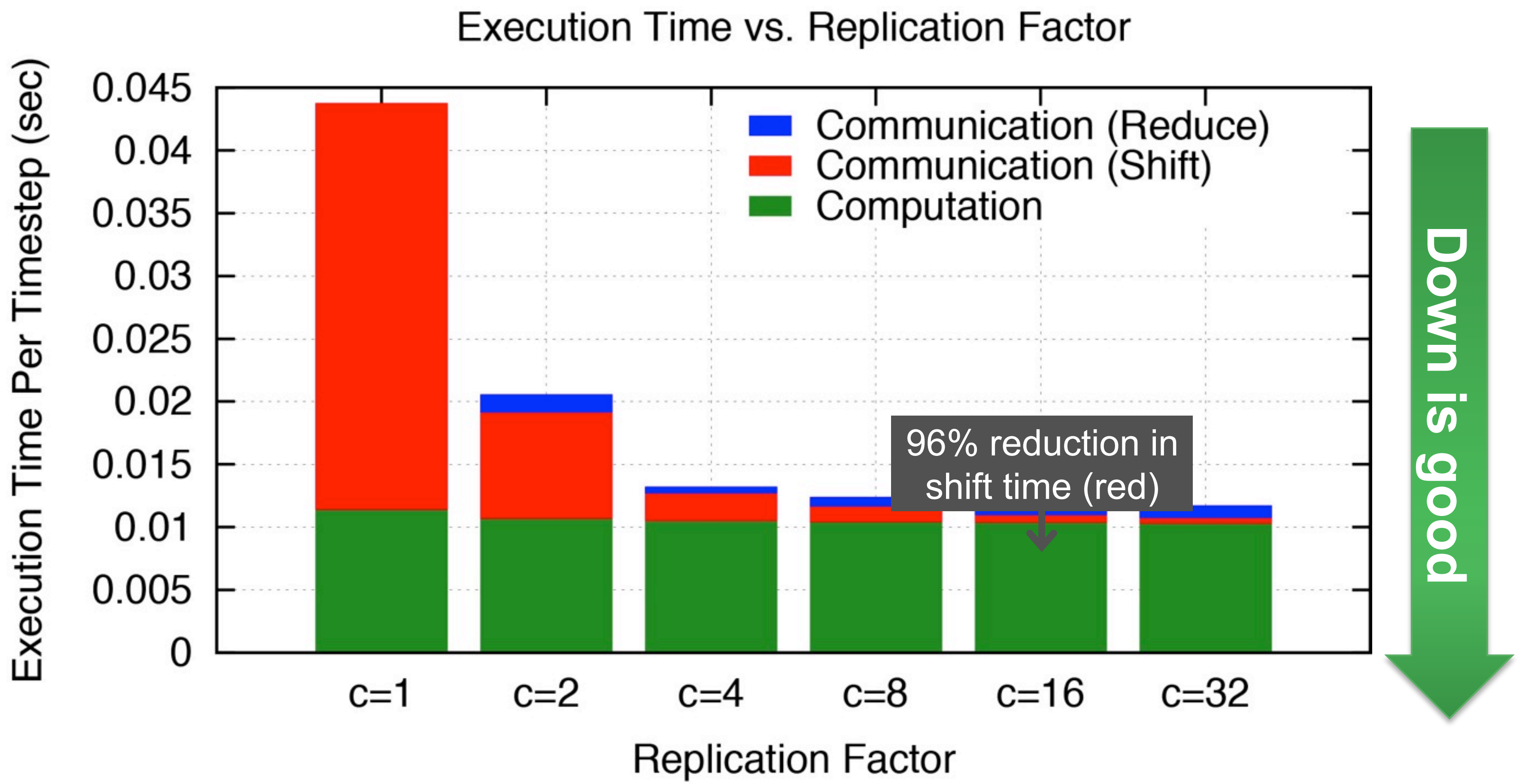
Total Communication: O(log(p/c) + log c) messages,

O(n*(c/p+1/c)) words

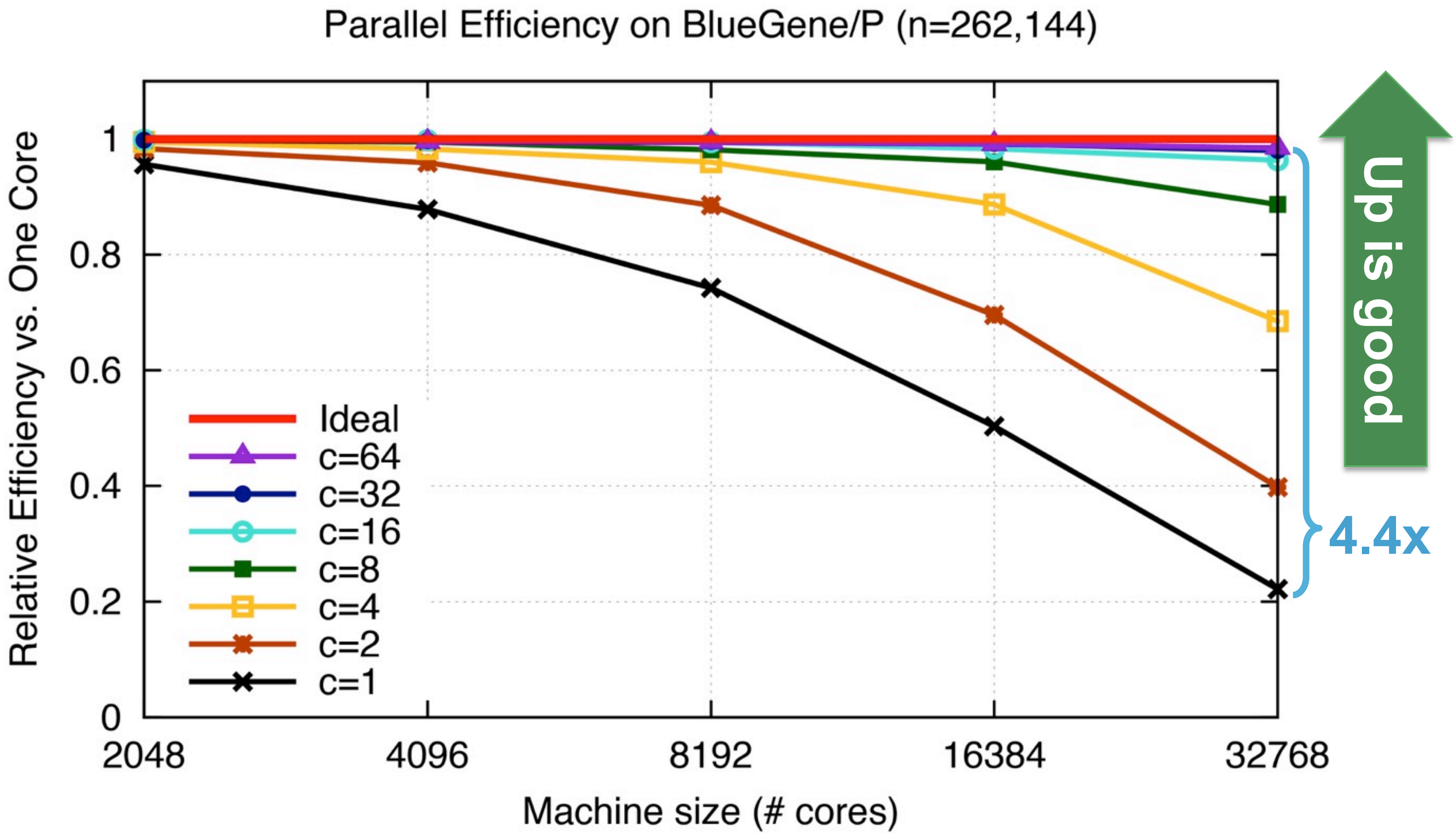
Driscoll, Georganas, Koanantakool, Solomonik, Yelick

Less Communication..

Cray XE6; n=24K particles, p=6K cores

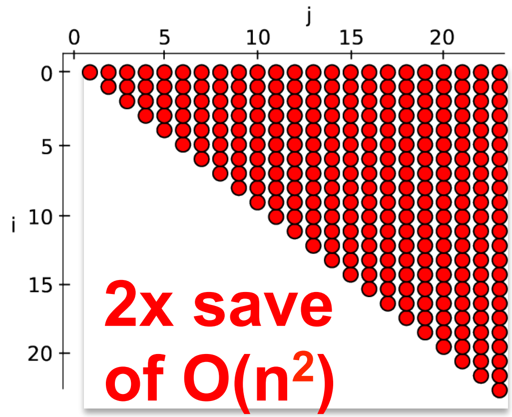


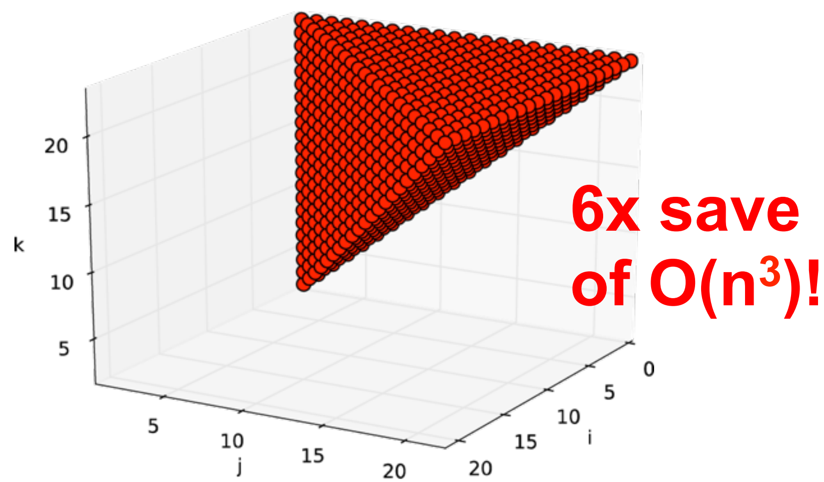
Strong Scaling



Challenge: Symmetry & Load Balance

- Force symmetry ($f_{ij} = -f_{ji}$) saves computation
- 2-body force matrix vs 3-body force cube

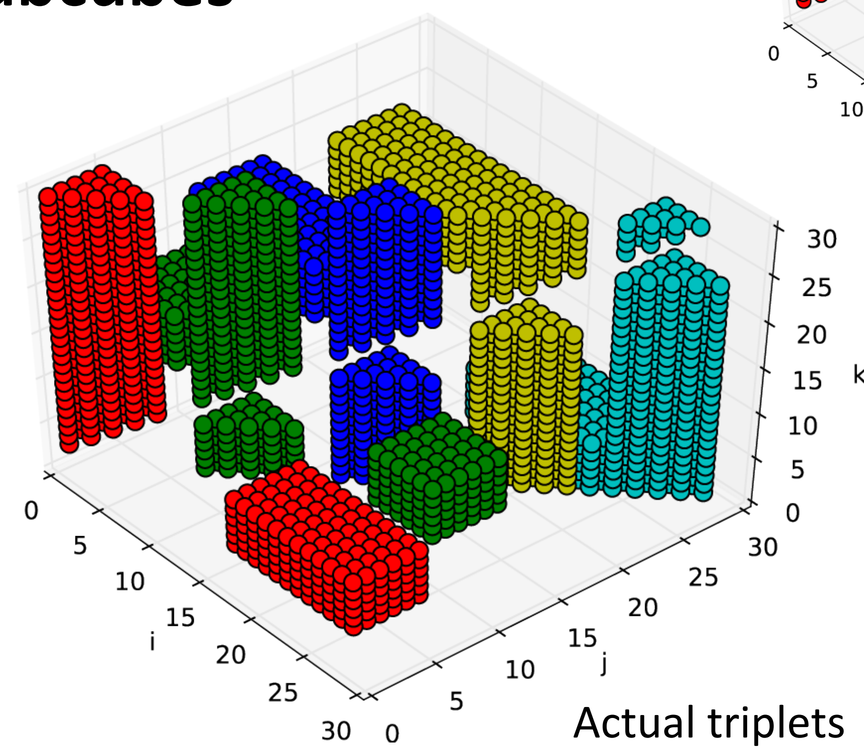




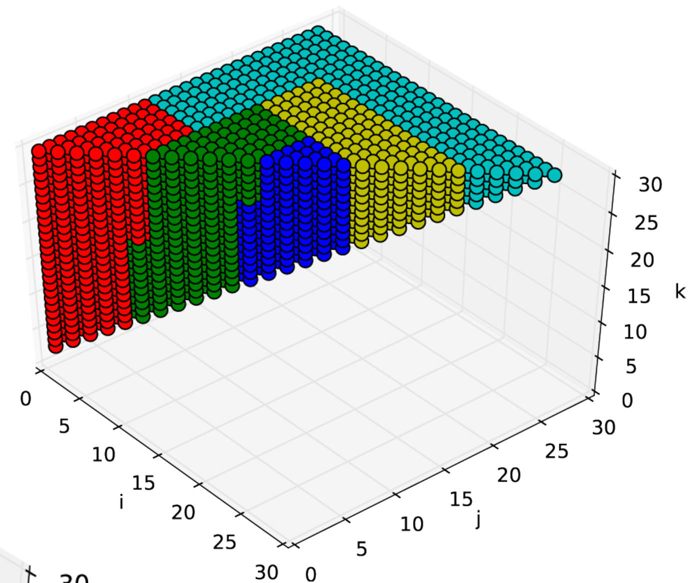
- How to divide work equally?

Communication-Avoiding 3-body

- p=5 (in colors)
- 6 particles per processor
- 5x5 subcubes



Equivalent triplets in the big tetrahedron

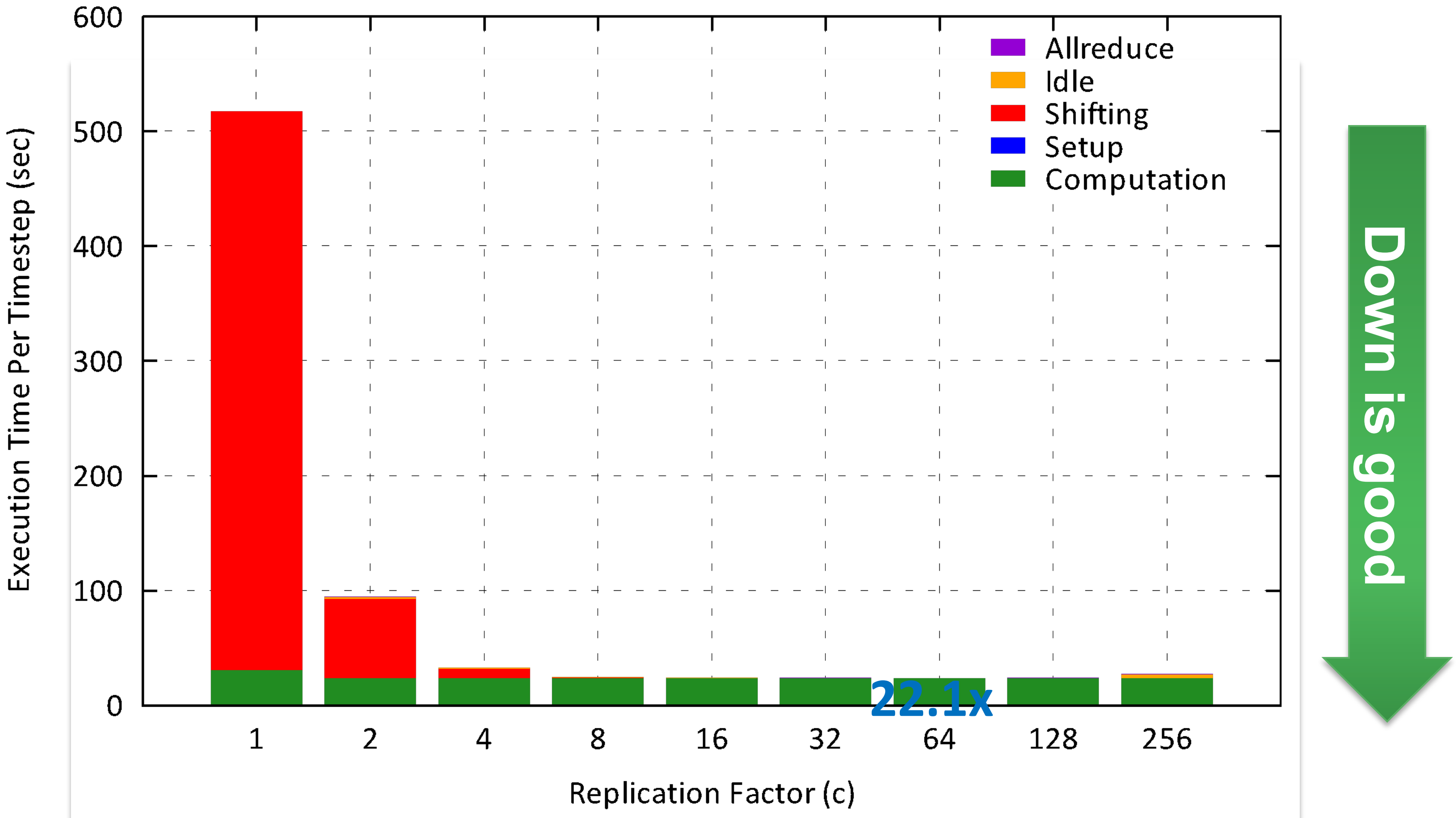


Actual triplets

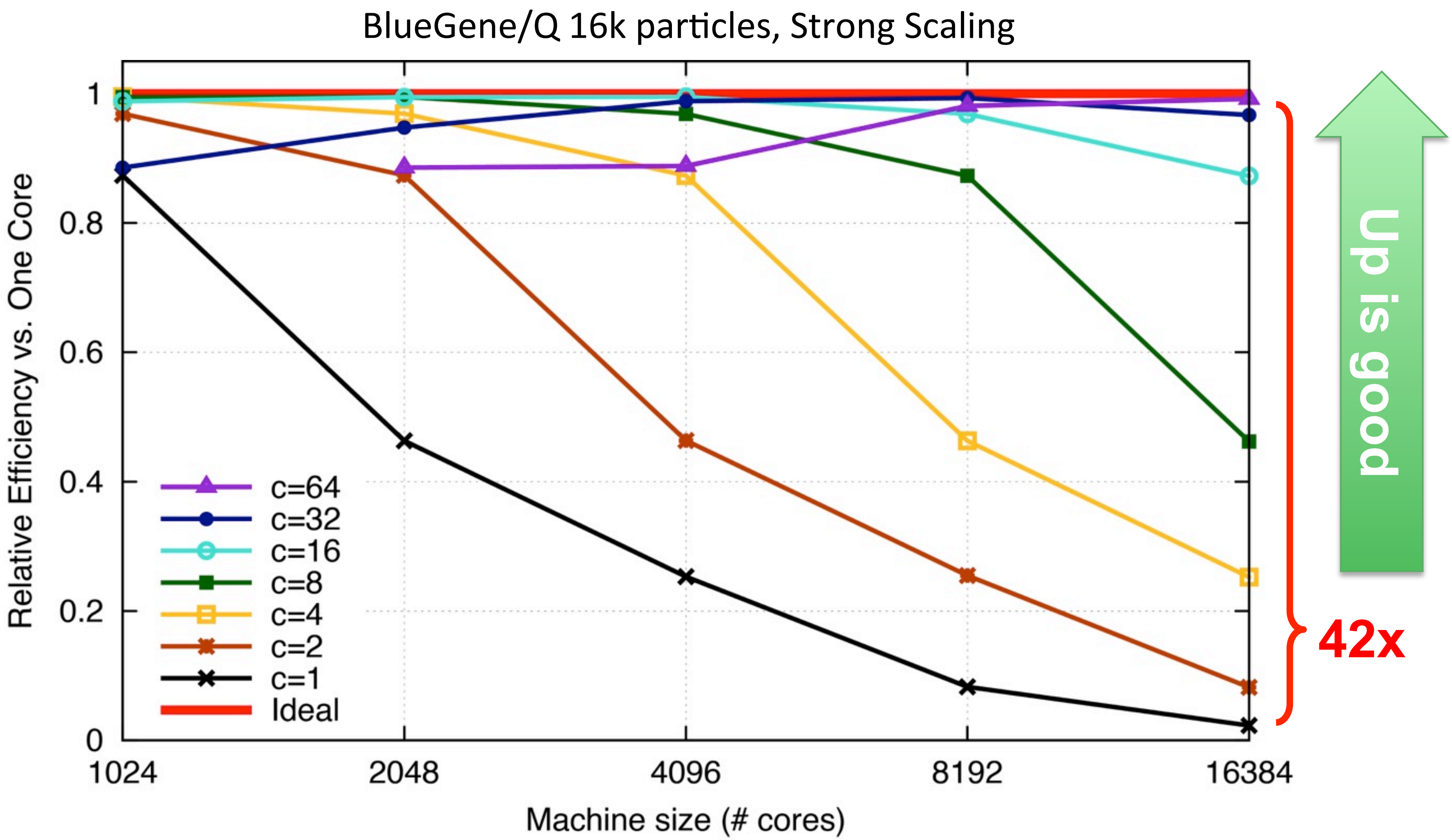
Communication optimal. Replication by c decreases #messages by c^3 and #words by c^2

3-Way N-Body Speedup

- **Cray XC30, 24k cores, 24k particles**

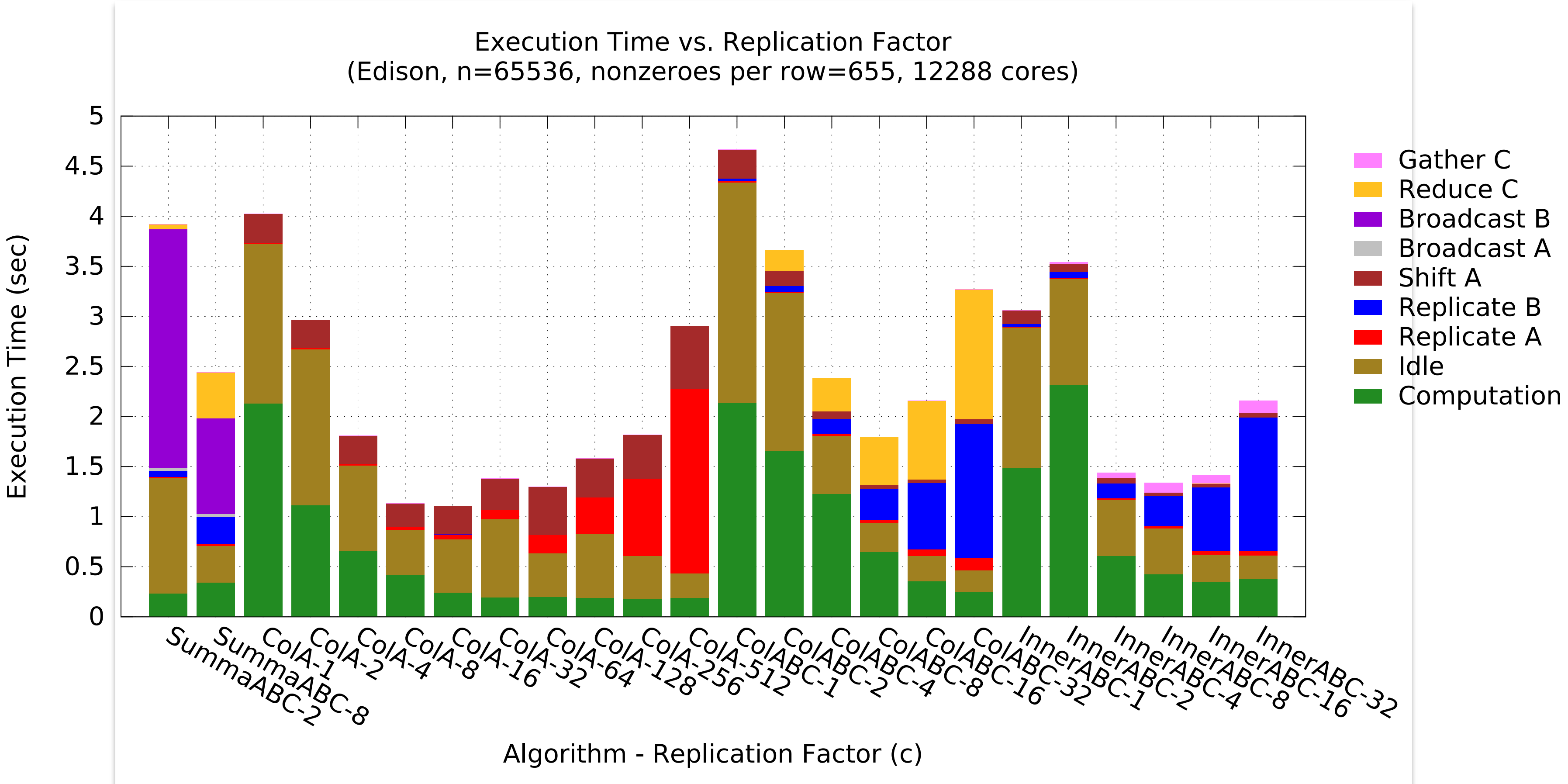


Perfect Strong Scaling



BERKELEY LAB
U.S. DEPARTMENT OF ENERGY
Office of Science
Koanantakool & Yelick

Sparse-Dense Matrix Multiply Too!

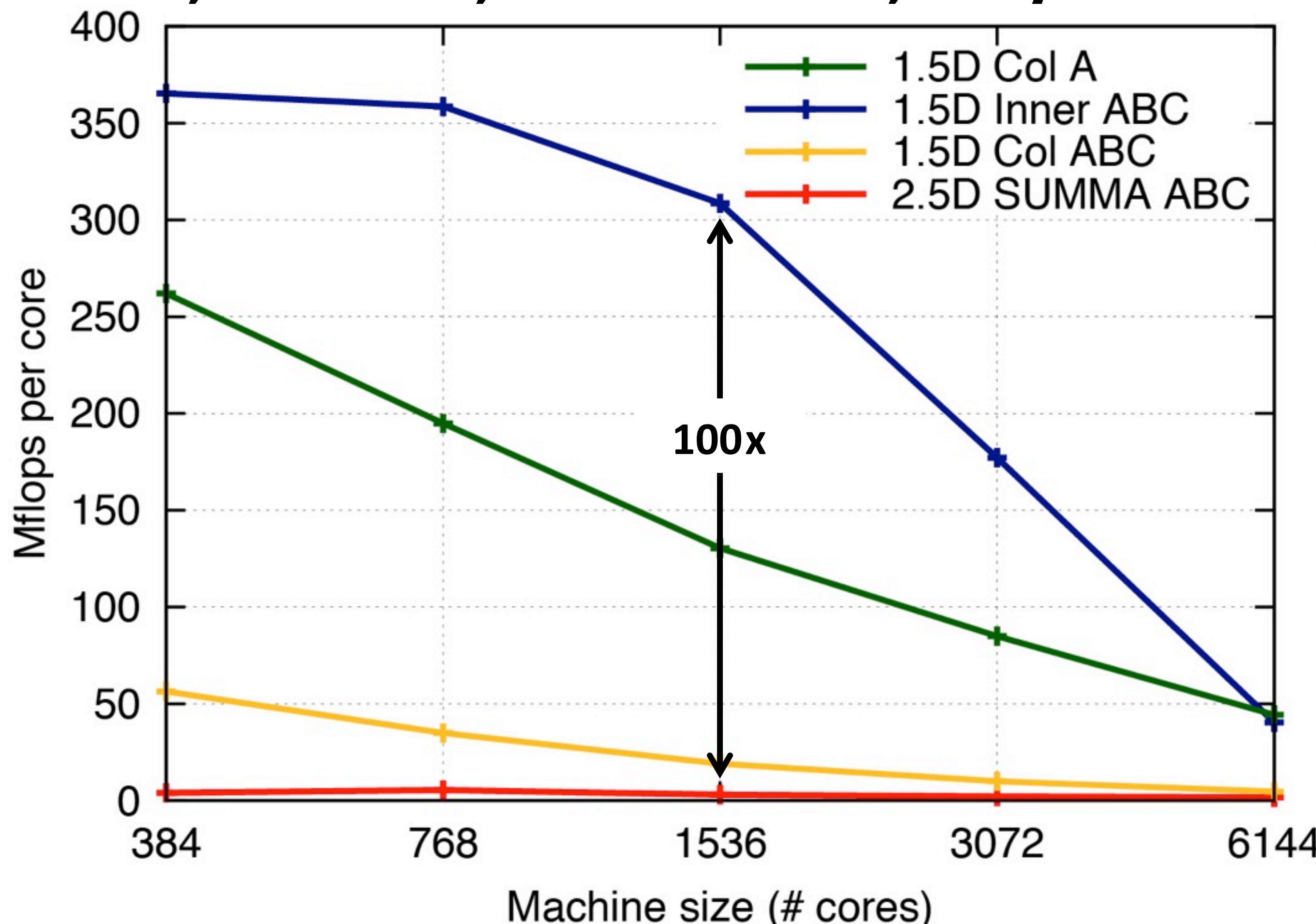


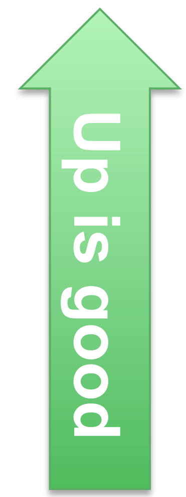
- **Variety of algorithms that divide in or 2 dimensions**

Koanantakool et al

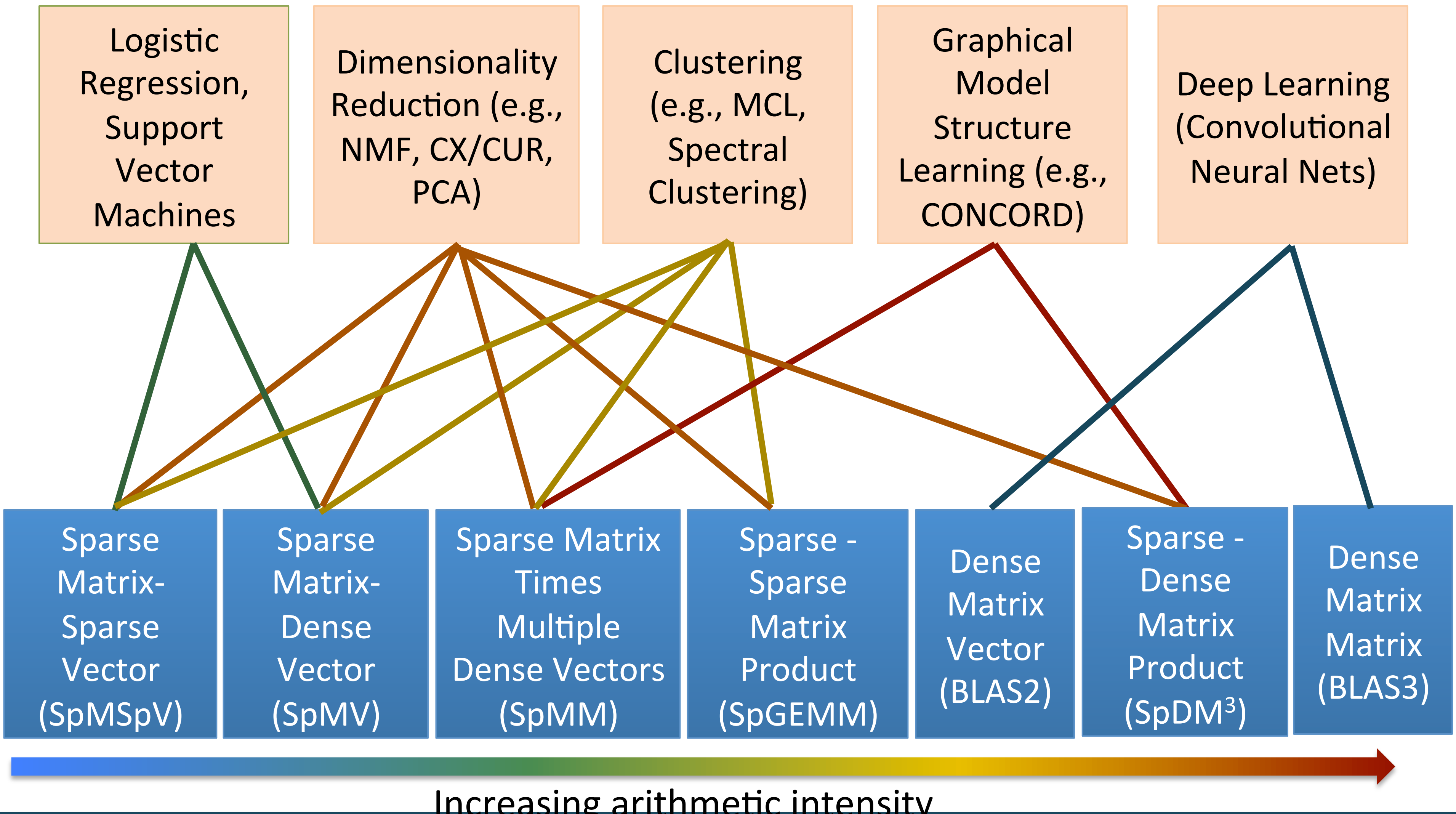
100x Improvement

- $A^{66k \times 172k}$, $B^{172k \times 66k}$, 0.0038% nnz, Cray XC30





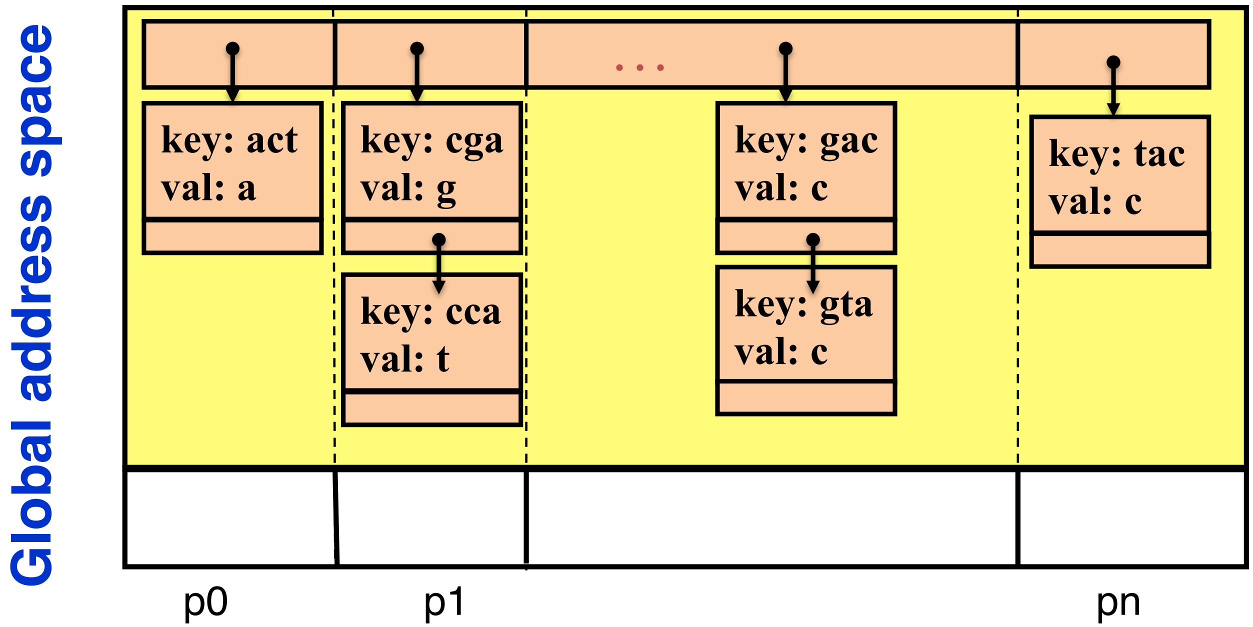
Linear Algebra is important to Machine Learning too!



Aydin Buluc, Sang Oh, John Gilbert, Kathy Yelick

Problem 3: Runtimes

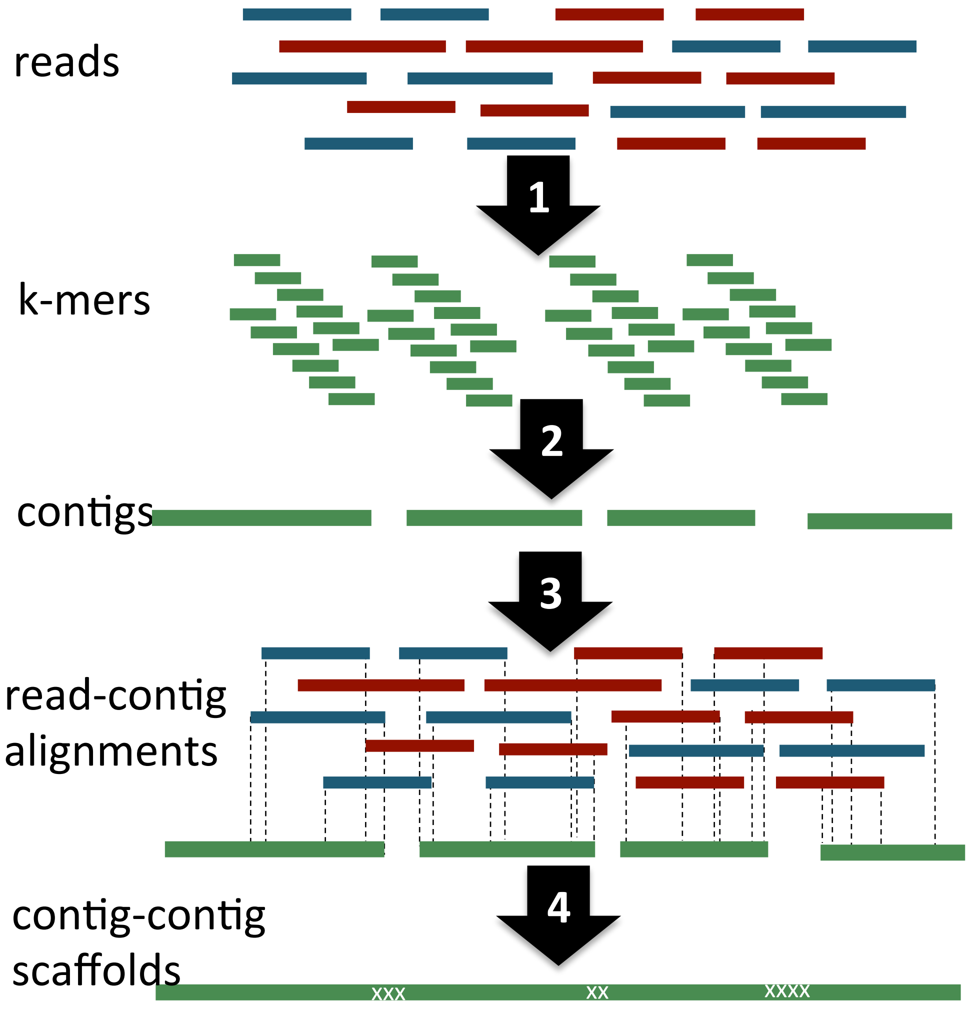
What we love about Partitioned Global Address Space Programming (PGAS)



Key: Never cache remote data (trivially coherent)

- **Convenience**
  - Build large shared structures
  - Read and write data "anywhere" (global), "anytime" (asynchronous) and without the other thread (one-sided)
- **Performance control**
  - Explicit control over data layout, direct use of RDMA hardware

HipMer is all about the runtime and data structures



1) *K-mer Analysis*

(synchronous) irregular all-to-all

2) *Contig Generation*

asynchronous remote insert (aggregate and overlap) and get

3) *Alignment*

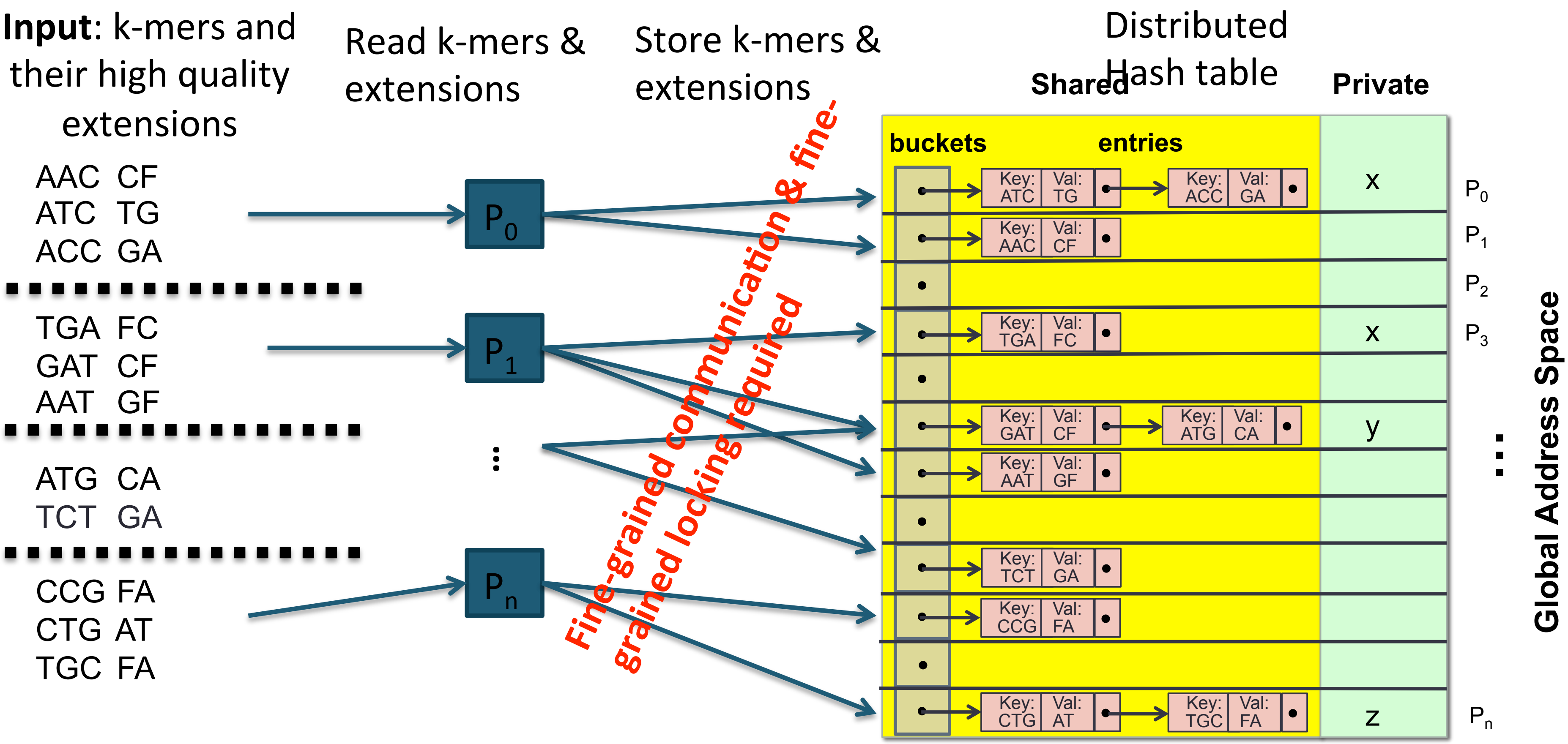
asynchronous remote insert and lookup (software caching)

4) *Scaffolding & Gap Closing*

asynchronous remote insert and lookup (software caching)

Graph algorithms (hash tables) in genome assembly

Graph construction, traversal, and all later stages are written in UPC to take advantage of its global address space



Lessons learned and open problem

- **Asynchronous one-sided communication model changes algorithmic intuition**
- **Machines still require aggregation, even if it's asynchronous**
- **Understanding side-effects of usage is key**
  - Insert-only phase, lookup-only phase, marking elements but not changing table, etc.
  - Caching of hash table depends on the statics (useful in some phases, not others)
- **This model should not be built for each application**

Summary

- **Exascale computing will deliver science breakthroughs**
  - In simulation and data analytics
  - **But requires advances in models, algorithms, languages, compilers and runtime systems**
- **Languages must be matched to user needs**
  - Domain specific languages and application level libraries
- **Compiler techniques are key to performance**
  - Novel delivery mechanisms (runtimes, DLS, libraries,…)
- **Runtime systems**
  - Lightweight communication still matters, but better synchronization models are needed