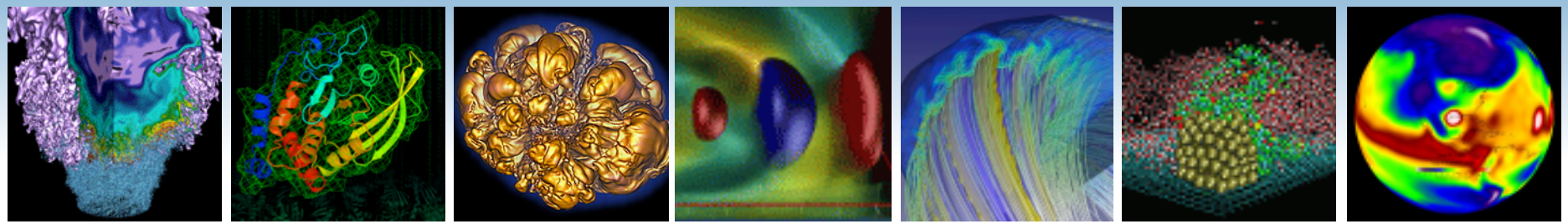


Berkeley Lab Welcomes NUG

Kathy Yelick

Associate Laboratory Director for Computing Sciences
Lawrence Berkeley National Laboratory

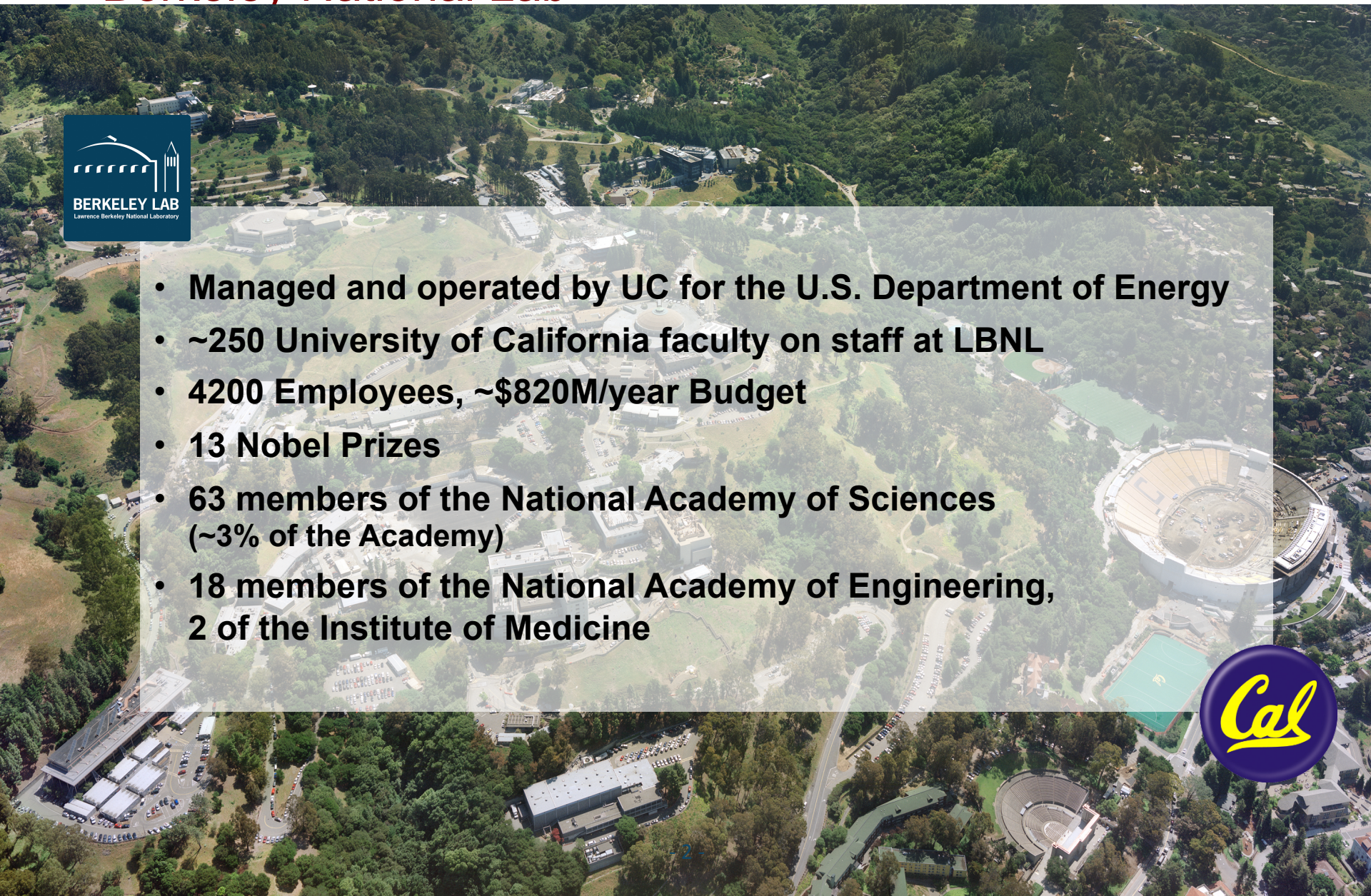
EECS Professor, UC Berkeley

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

BERKELEY LAB
Lawrence Berkeley National Laboratory

80 Years of World Leading Team Science at Lawrence Berkeley National Lab



- **Managed and operated by UC for the U.S. Department of Energy**
- **~250 University of California faculty on staff at LBNL**
- **4200 Employees, ~$820M/year Budget**
- **13 Nobel Prizes**
- **63 members of the National Academy of Sciences (~3% of the Academy)**
- **18 members of the National Academy of Engineering, 2 of the Institute of Medicine**

World-Class User Facilities Underpin Today's Berkeley Lab

Over 7,000 visiting scientists (~2/3 from universities) use Berkeley Lab research facilities each year

Berkeley Lab Science Focus Areas

BIOSCIENCES

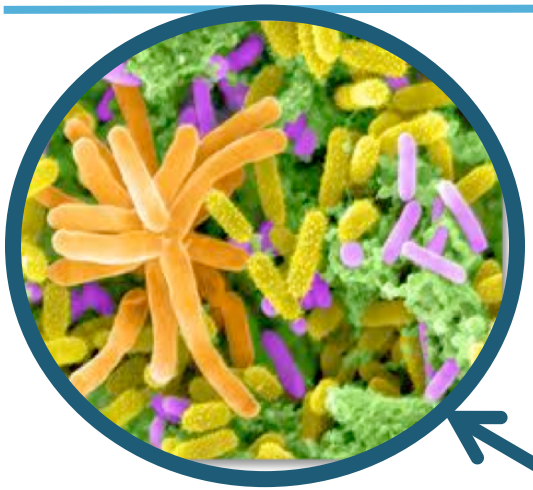
ENERGY & ENVIRONMENTAL SCIENCES

PHOTON SCIENCES

COSMOLOGY & PHYSICS

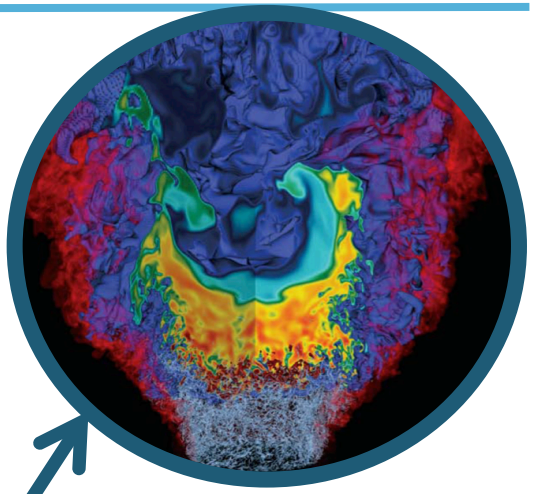
COMPUTING SCIENCES

Computing is Essential for Science Programs in All Areas of the Lab

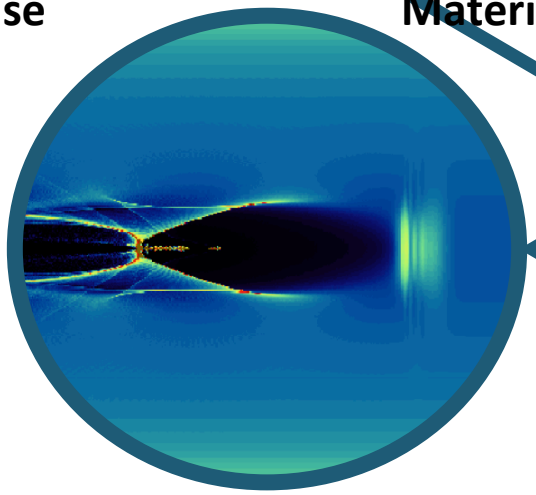


BIOSCIENCES: JGI & KBase

ENERGY & ENVIRONMENTAL: Materials Genome



PHOTON SCIENCES: NGLS



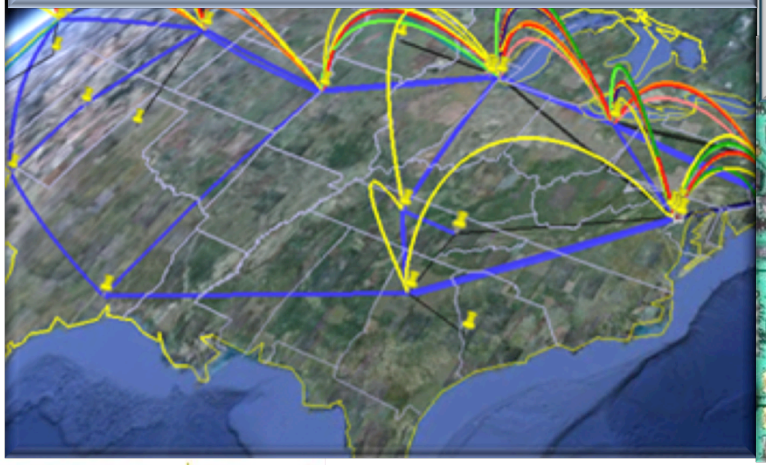
COSMOLOGY & PHYSICS: BELLA

COMPUTING SCIENCES

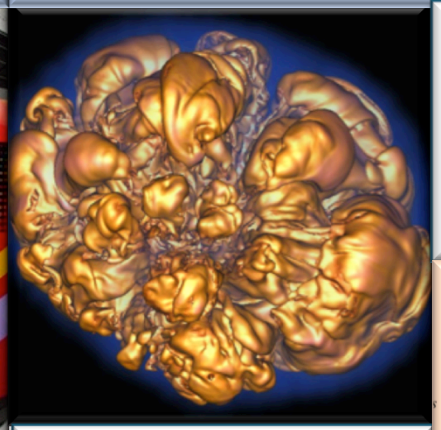
Computing Sciences at Berkeley Lab

NERSC Facility

ESnet Facility



Computational Research



Computational Science

Applied Mathematics

$$\frac{dVar(t)}{dt} =$$

$$\int_{s_2(t)}^{s_1(t)+S} - K^2Fg + gK^2F)ds - (-(g^{-1}F_s)_s - K$$

$$= -\int_{s_1(t)}^{s_2(t)} (g^{-1}F_s)_s \, ds + \int_{s_2(t)}^{s_1(t)+S} (g^{-1}F_s)_s \, ds$$

$$^{-1}F_s \mid_{s_2(t)} - g^{-1}F_s \mid_{s_1(t)} \Big] + \Big[g^{-1}F_s \mid_{s_1(t)+S} - g$$

$$= -2\,(g^{-1}F_K K_s) \mid_{s_2(t)} + 2\,(g^{-1}F_K K_s) \mid_{s_1(t}$$

Computer Science

ESnet is a Unique Instrument for Data-Intensive Science

ESnet designed for large data

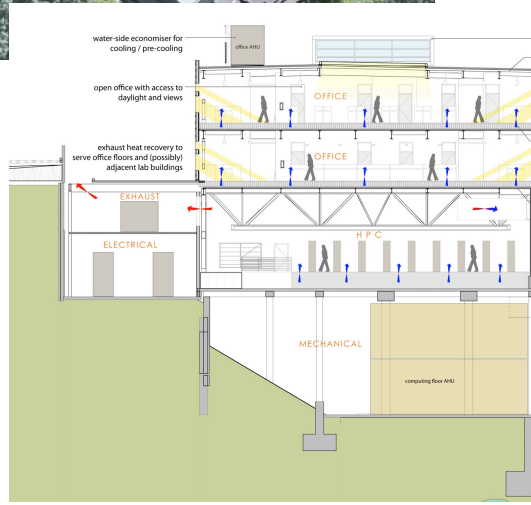
- Connects 40 DOE sites to 140 networks
- Growing 2x commercial networks
- 50% of traffic is from “big data”

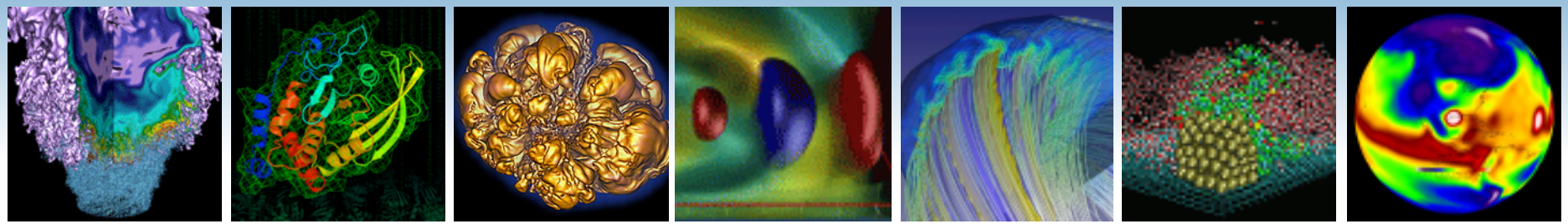
Unique capabilities:

- First 100G continental scale network
- Dark fiber could support 1 terabit
- Services based on science:
  - Bandwidth reservations (OSCARS)
  - Performance monitoring (perfSONAR)
  - Endpoint features (ScienceDMZ and Data Transfer nodes)
  - Research testbeds (dark fiber)

UC's Computational Research and Theory (CRT) Facility

- **Unique energy efficient design from weather / hillside**
  - Collaborative space for 300
  - $124M UC Project (up $12M)
  - $20M DOE Project
  - 100 MW at Berkeley Lab and space for 2 exascale systems
- **But "efficiency" is complicated**
  - Biggest factor in efficiency is utilization (90%)
  - Race-to-halt is often best
- **If we measure productivity by publications…**
  - *NERSC in 2010 has 450 publications per MW-year*
- **Next best: application performance per Watt**





The Future of High Performance Scientific Computing

Kathy Yelick

Associate Laboratory Director for Computing Sciences
Lawrence Berkeley National Laboratory

EECS Professor, UC Berkeley

U.S. DEPARTMENT OF ENERGY | Office of Science

NERSC

BERKELEY LAB
Lawrence Berkeley National Laboratory

2013 Computing with 1993 Technology

Google™

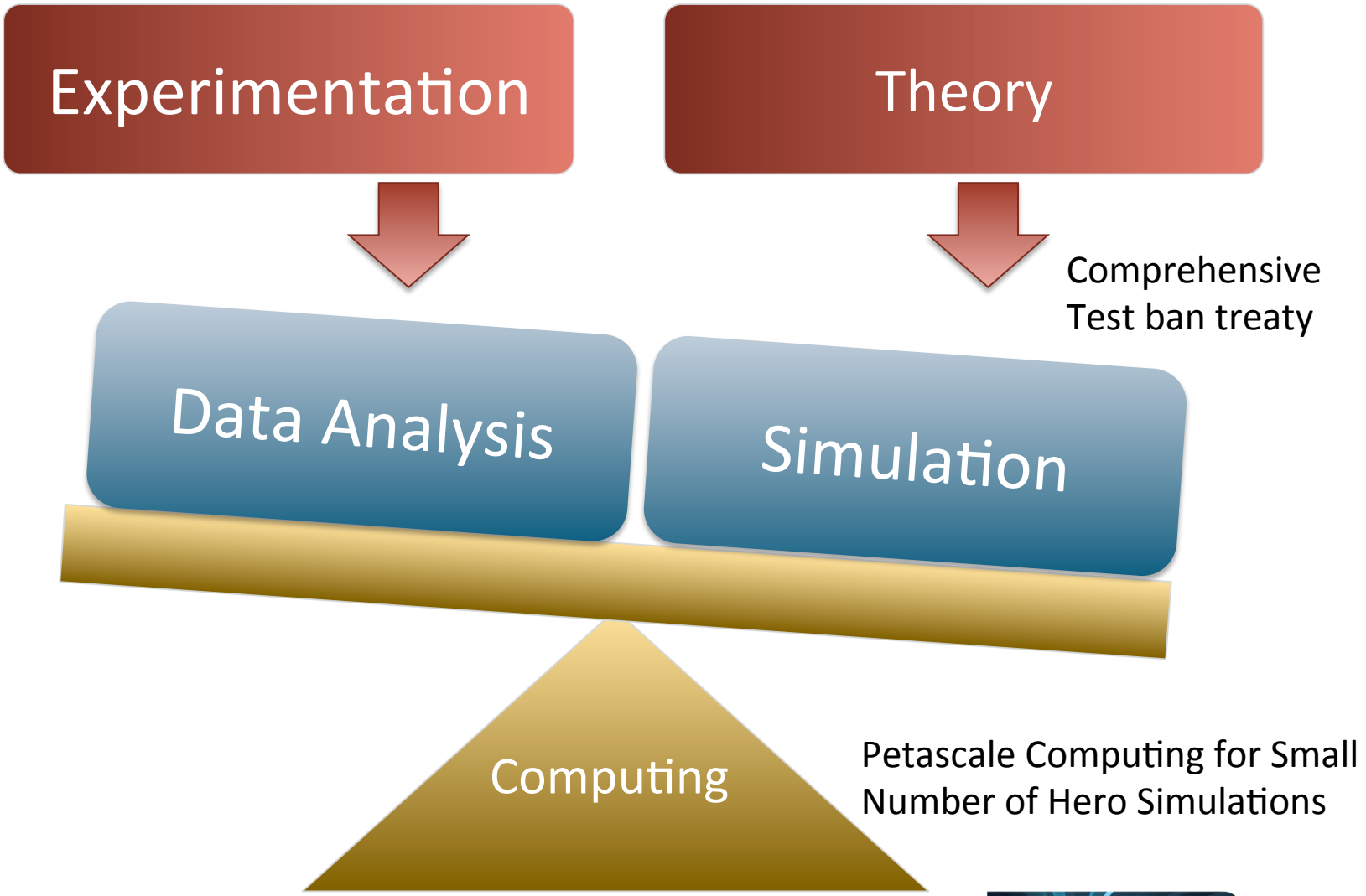
NERSC

BERKELEY LAB
Lawrence Berkeley National Laboratory

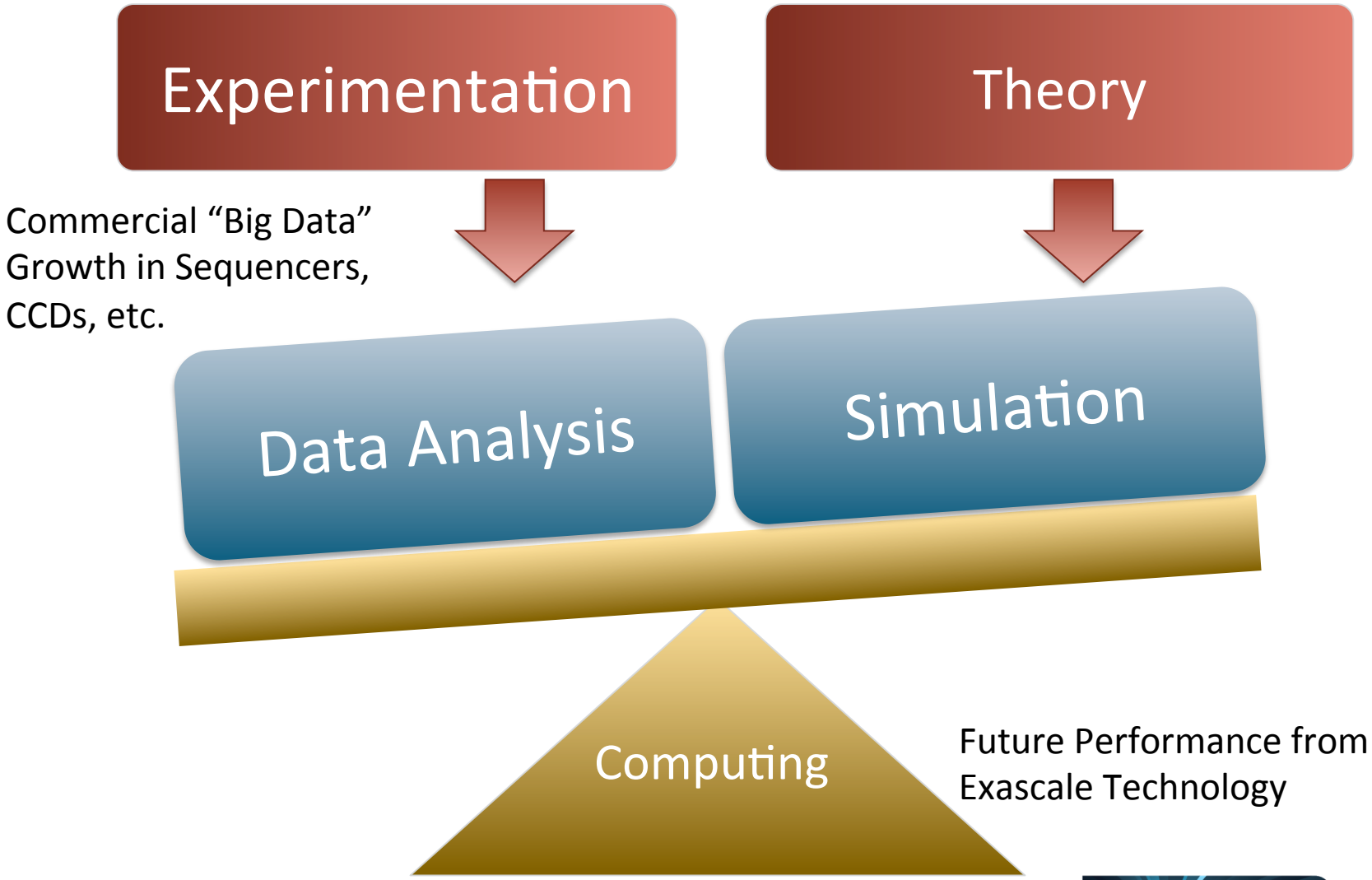
Life of Scientist, 2033

- No personal/departmental computers
- Travel replaced by telepresence
- Lecturers teach millions of students
- Theorems proven online (Polymath)
- Users never login to NERSC systems
- Computers “intuit” what jobs should be run
- No users visit other user facilities
- What does this mean for big, team science?

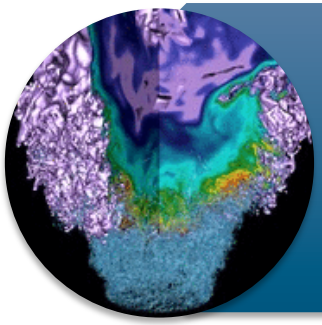
The Influence of World Politics on HPC



Data-Intensive Computing is Growing in Science

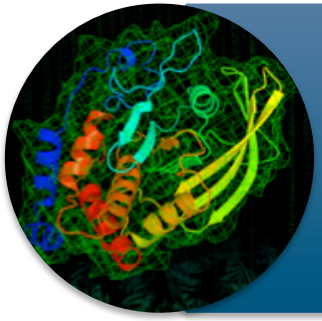


More Science Requires More Computing



Science at Scale

Petascale to Exascale Simulations



Science through Volume

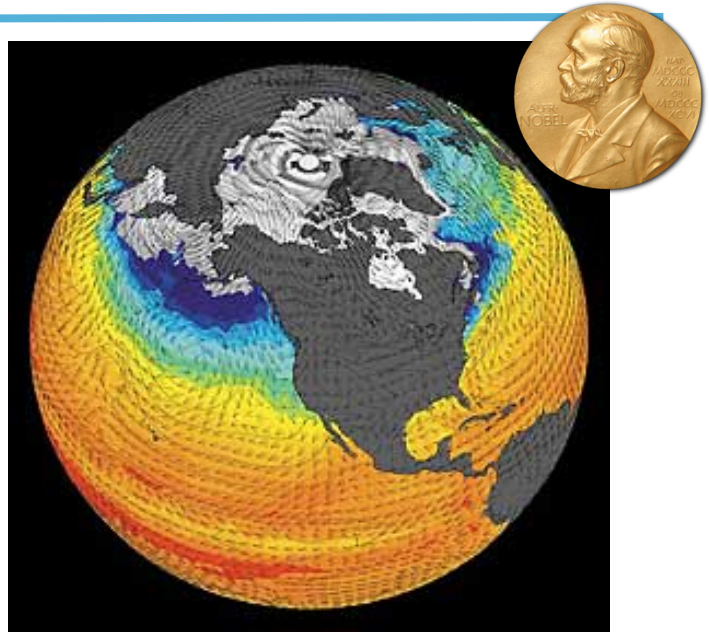
Thousands to Millions of Simulations

Science in Data

Petabytes to Exabytes of Data

Science at Scale

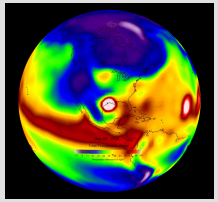
- **Climate at NERSC dates back to 1990s**
  - Many IPCC 4th Assessment Report (AR4) simulations run at NERSC
  - PCM database first truly public climate database
  - AR5* runs completed, AR6** runs being planned
  - NERSC hosts climate collaborations from NCAR, Scripps, LBNL, LLNL, NOAA, PNL, NASA, IGES/COLA, ORNL, State of CA, and many universities.
- **Cloud resolution, quantifying uncertainty, etc. will drive climate to exascale platforms**
  - New math, science, systems support, data analytics necessary along the way



*2013 release **2017/18 release

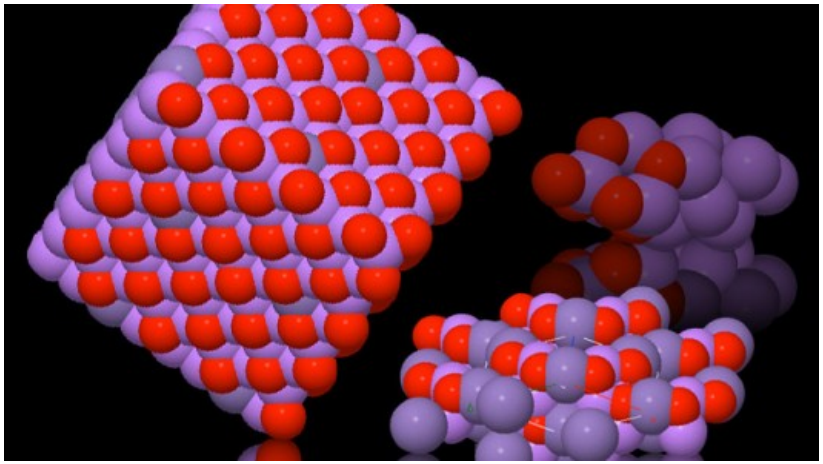
2011 *Scientific Computing* HPC Innovation Award

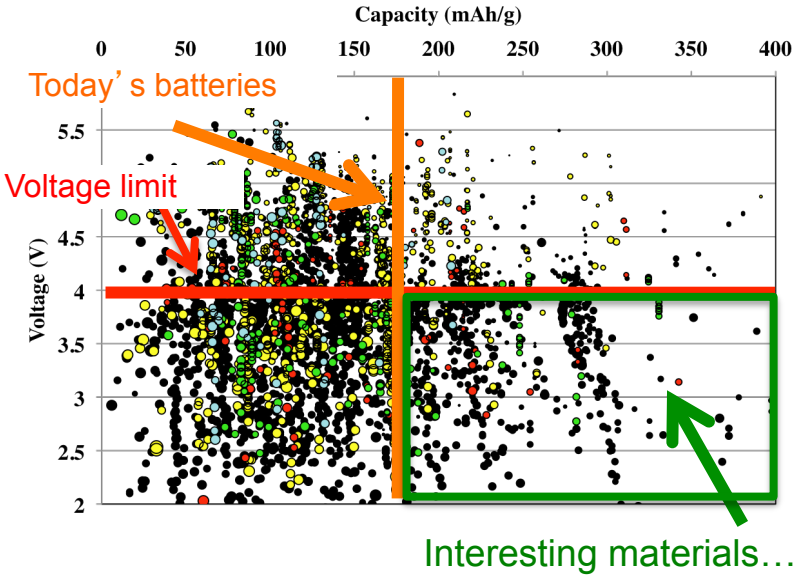
NERSC and U. Colorado's Gil Compo honored for an "*International study that has enabled much more detailed and longer (100 years) record of past weather to improve climate studies.*"



Science through Volume: Large Numbers of Simulations for Materials

- Tens of thousands of simulations are used to screen related materials for use in battery design and other domains
- Goal: cut in half the 18 year from design to manufacturing





Materials Project, Gerd Ceder PI (MIT): *website has a database materials from simulations, e.g., over 20,000 potential battery materials.*

PIs: Gerd Ceder, MIT and Kristin Persson, LBNL

Joint Genome Institute (JGI) database has billions of genes

- **Metagenomics is key to DOE**
- **Integrated Microbial Genomes (IMG)**
  - Community resource for comparative genome analysis
  - From 1 to 3 billion genes in 2012
    - Weeks to days for 100M gene sets
    - Data streams over ESnet and is stored in NERSC's data archive (tapes)
    - Use dedicated JGI clusters and 20,000 – 75,000 cores on largest HPC system (Hopper)
- **Future growth from KBASE**
  - Distributed ***Knowledge***base enabling ***predictive*** systems biology, e.g., microbes, plants, communities
  - Extensible **open-source** software

Radial Phylogenetic Tree

JGI DOE JOINT GENOME INSTITUTE

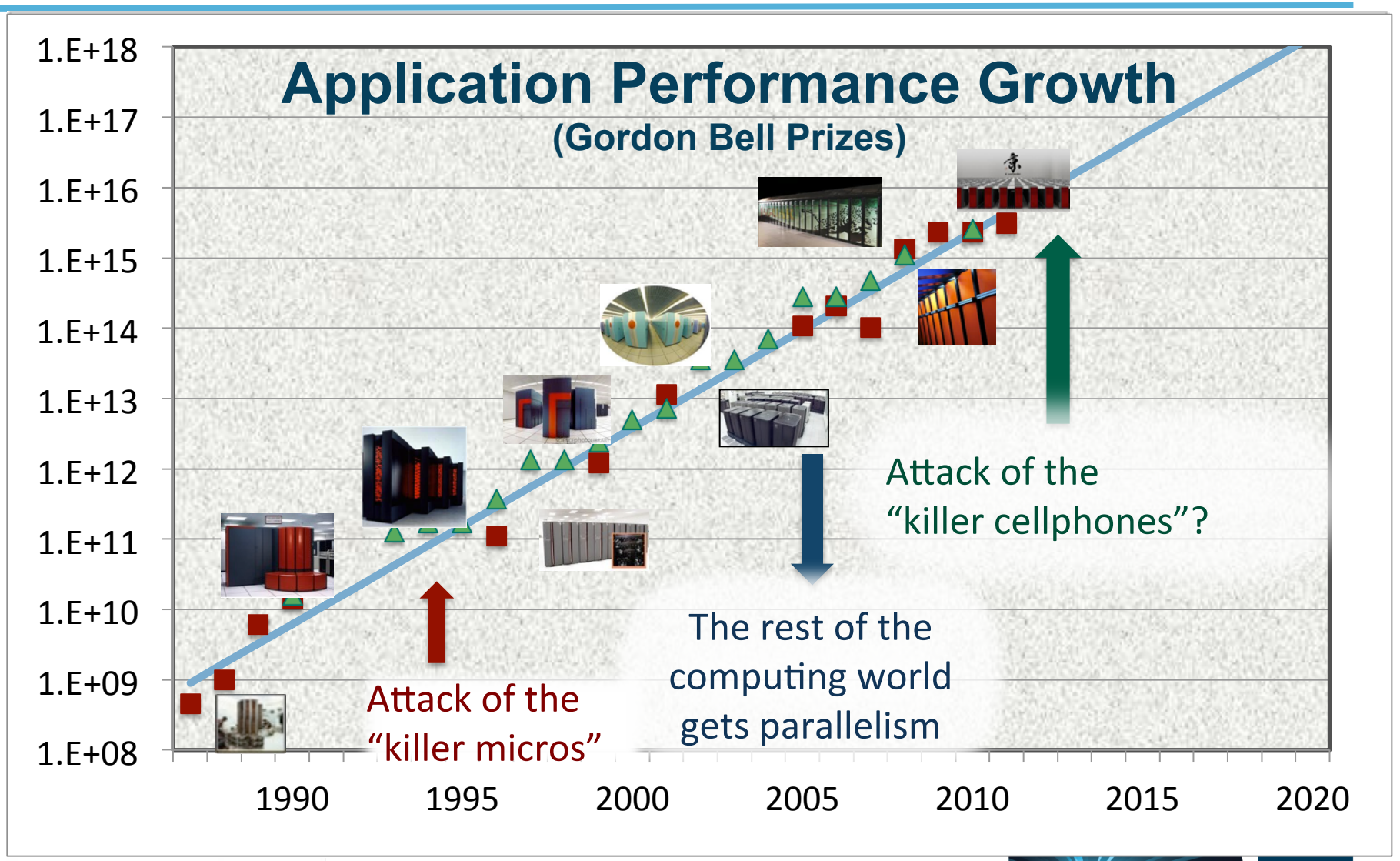
img

DOE Systems Biology Knowledgebase

KBASE

Data and modeling for predictive biology

Computational Science has Moved through Difficult Technology Transitions



NERSC is Cost Effective Relative to Clouds

Component	Annual Cost
Compute Systems (1.38B hours)	$181M
HPSS (17 PB)	$12M
File Systems (2 PB)	$3M
Total (Annual Cost)	**~$200M**

NERSC FY11-12 Budget $57M

These "list" prices overestimate cloud costs, but other underestimate:

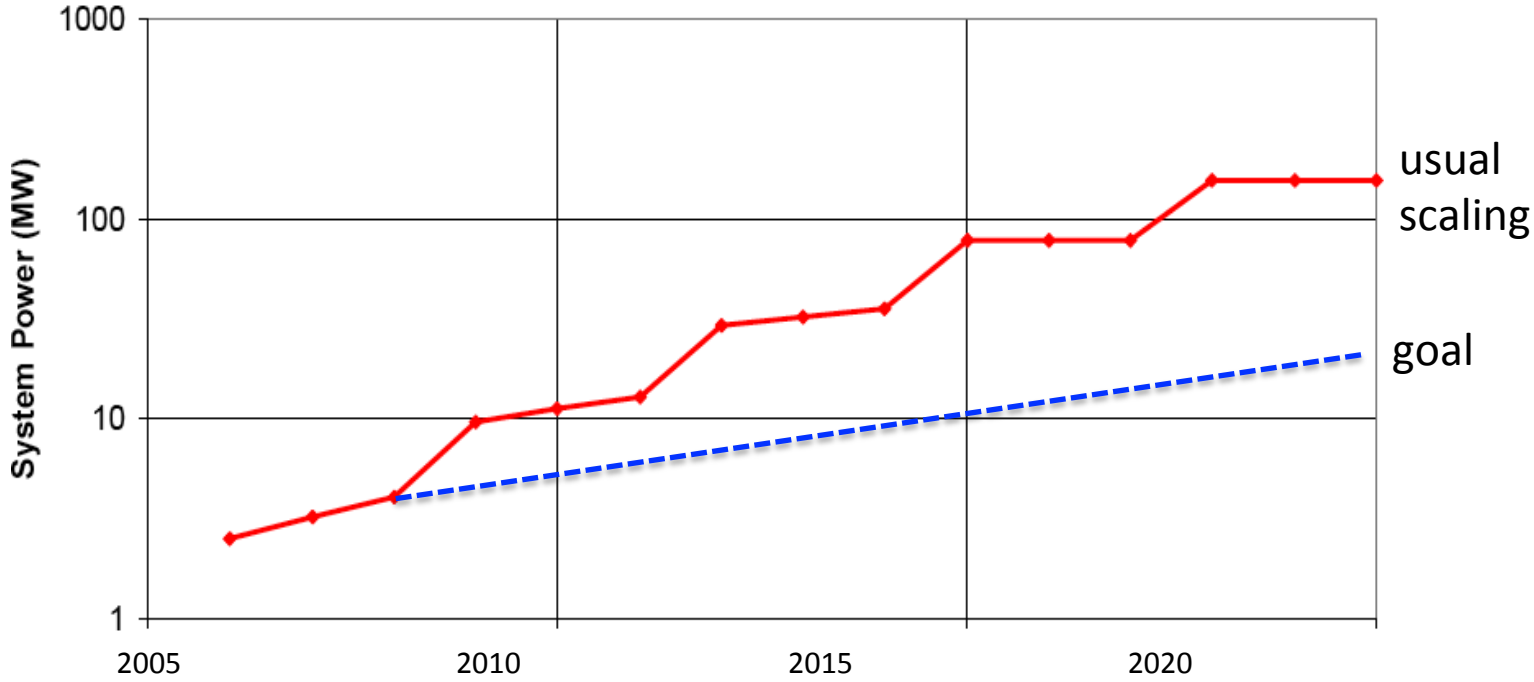
- Doesn't include the measured performance slowdown 2x-50x.
- **No consulting staff, no account management, no software support which is ~1/3 of NERSC's Budget**

Why? NERSC enjoys many of the cloud benefits at scale, but higher utilization (> 90%) and no profit.
NERSC cost/core hours dropped 10x (1000%) from 2007 to 2011
Amazon pricing dropped 15% in the same period

Energy Efficient Computing is Key to Performance Growth

At $1M per MW, energy costs are substantial

- **1 petaflop in 2010 used 3 MW**
- **1 exaflop in 2018 would use 100+ MW with "Moore's Law" scaling**



This problem doesn't change if we were to build 1000 1-Petaflop machines instead of 1 Exasflop machine. It affects every university department cluster and cloud data center.

Communication is expensive

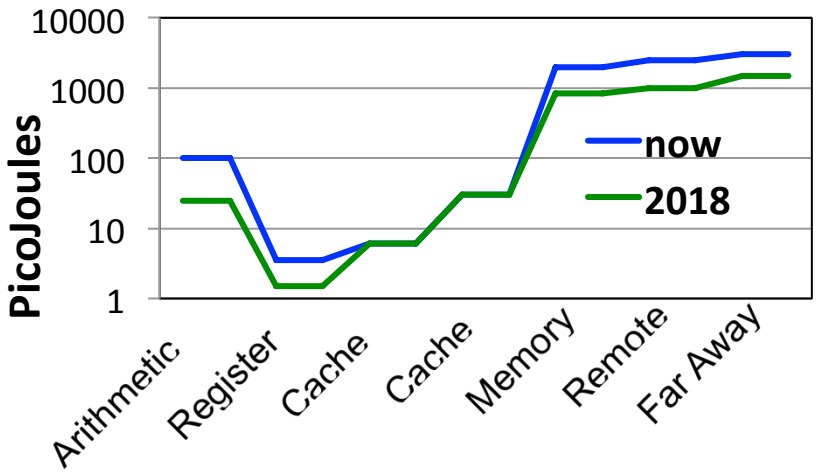
Communication is expensive…

… time and energy

Annual improvements			
Flops		BW	Latency
59%	Network	26%	15%
	DRAM	23%	5%

Cost components:

- **Bandwidth:** # of words
- **Latency:** # messages



Strategies

- Overlap: hide latency
- Avoid: new algorithms to reduce bandwidth (at least)

Hard to change: Latency is physics; bandwidth is money!

The Memory ~~Wall~~ Swamp

Multicore didn't cause this, but kept the bandwidth gap growing.

Obama for Communication-Avoiding Algorithms

"New Algorithm Improves Performance and Accuracy on Extreme-Scale Computing Systems. **On modern computer architectures, communication between processors takes longer than the performance of a floating point arithmetic operation by a given processor.** ASCR researchers have developed a new method, derived from commonly used linear algebra methods, to **minimize communications between processors and the memory hierarchy, by reformulating the communication patterns specified within the algorithm**.."

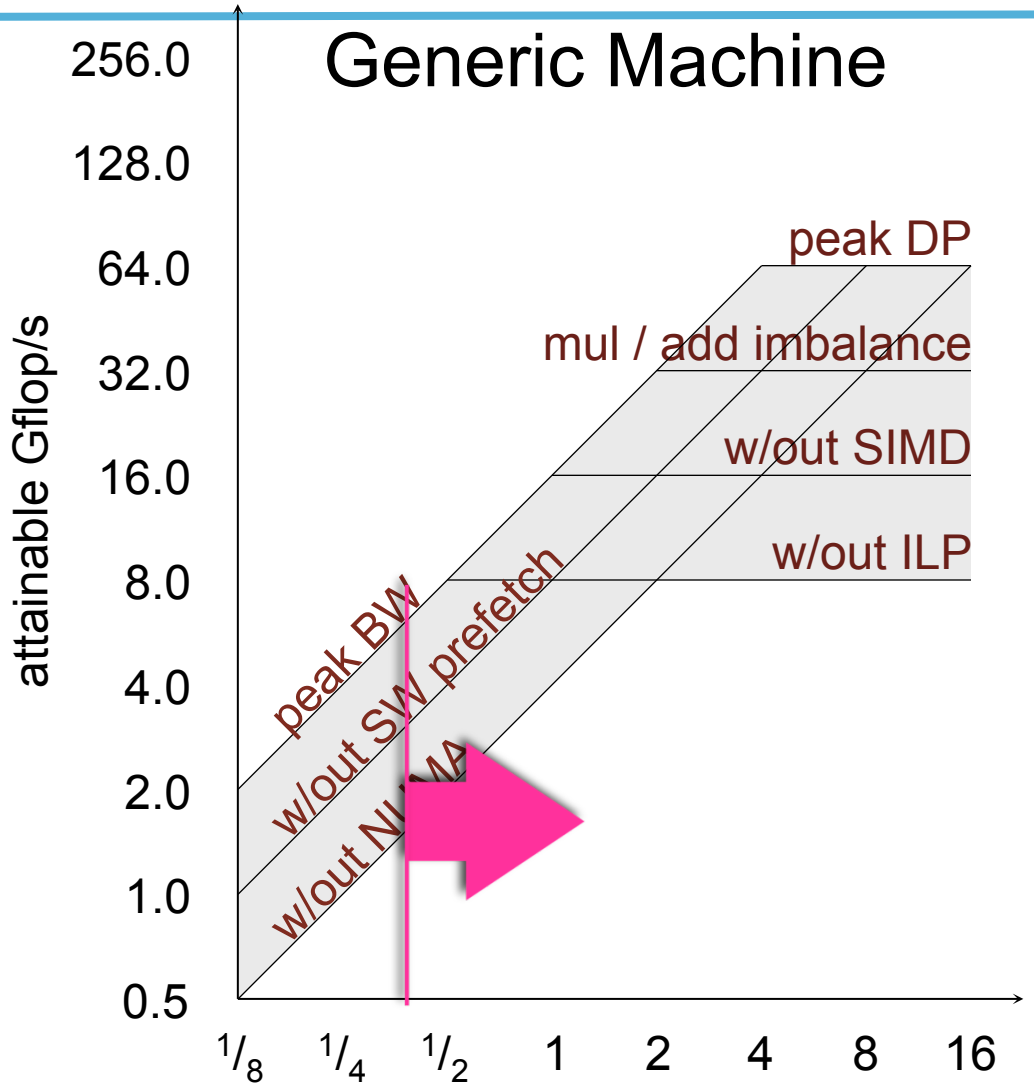
FY 2012 Congressional Budget Request, Volume 4, FY2010 Accomplishments, Advanced Scientific Computing Research (ASCR), pages 65-67.

But perhaps Obama took this too seriously in the first debate!

Lessons #1: Understand communication limits

Latency is physics; bandwidth is money!

William's Roofline Performance Model



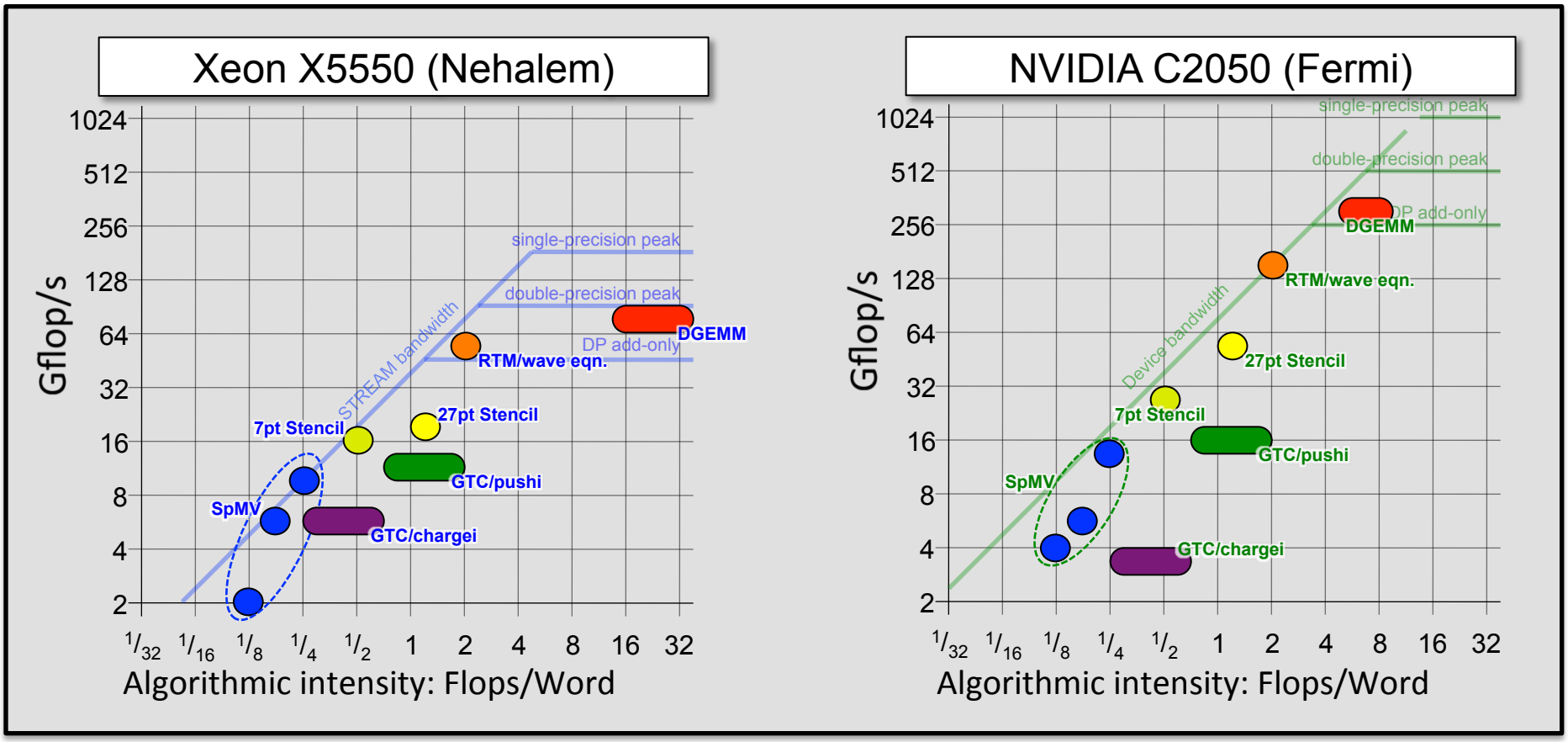
- Flat top is flop limit
- Slanted is bandwidth limit
- X-Axis is arithmetic intensity of algorithm

- Bandwidth-reducing optimizations, such as better cache re-use (tiling), compression improve intensity

Model by Sam Williams

Autotuning Gets Kernel Performance Near Optimal

- Roofline model captures bandwidth and computation limits
- Autotuning gets kernels near the roof



Work by Williams, Oliker, Shalf, Madduri, Kamil, Im, Ethier,…

Lessons #2: Target Higher Level Optimizations

Harder than inner loops....

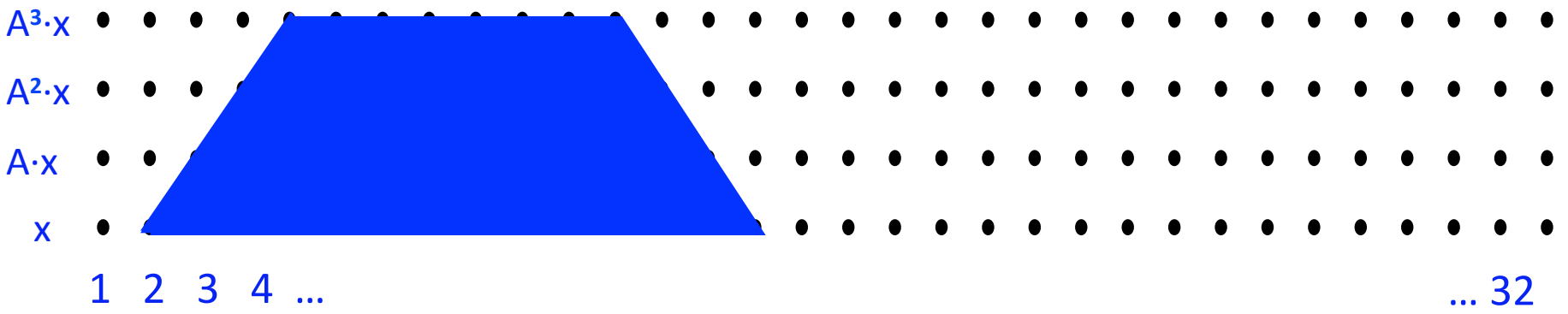
Avoiding Communication in Iterative Solvers

- **Consider Sparse Iterative Methods for *Ax=b***
  - Krylov Subspace Methods: GMRES, CG,…
  - Can we lower the communication costs?
    - Latency of communication, i.e., reduce # messages by computing multiple reductions at once
    - Bandwidth to memory hierarchy, i.e., compute Ax, A^2x, … A^kx with one read of A
- **Solve time dominated by:**
  - Sparse matrix-vector multiple (SPMV)
    - Which even on one processor is dominated by "communication" time to read the matrix
  - Global collectives (reductions)
    - Global latency-limited

Joint work with Jim Demmel, Mark Hoemman, Marghoob Mohiyuddin

Communication Avoiding Kernels:
The Matrix Powers Kernel : [Ax, A^2x, …, A^kx]

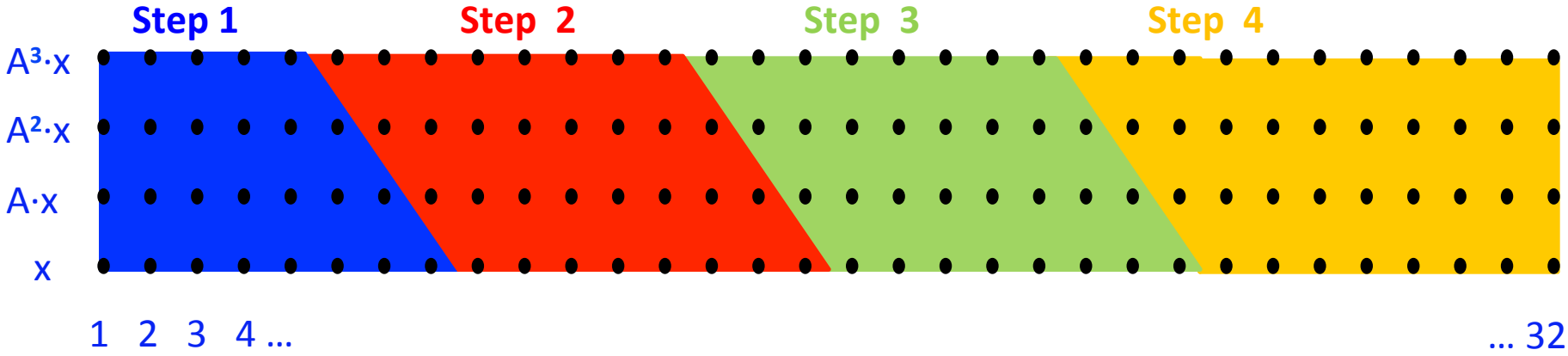
- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]



- Idea: pick up part of A and x that fit in fast memory, compute each of k products
- Example: A tridiagonal, n=32, k=3
- Works for any "well-partitioned" A

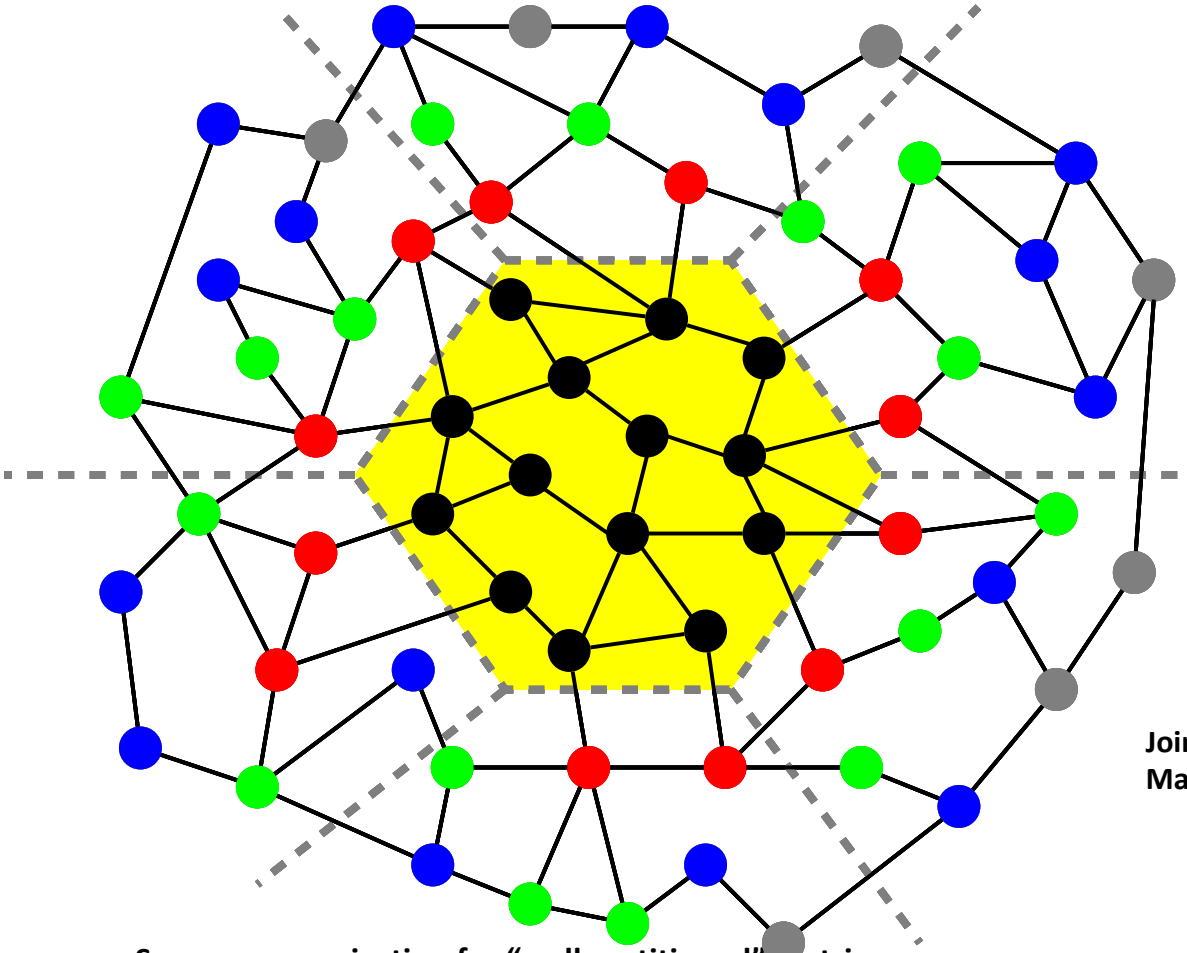
Communication Avoiding Kernels:
The Matrix Powers Kernel : [Ax, A^2x, …, A^kx]

- Replace k iterations of y = A·x with [Ax, A^2x, …, A^kx]
- **Sequential Algorithm**



- Example: A tridiagonal, n=32, k=3
- Saves bandwidth (one read of A&x for k steps)
- Saves latency (number of independent read events)

Matrix Powers Kernel on a General Matrix

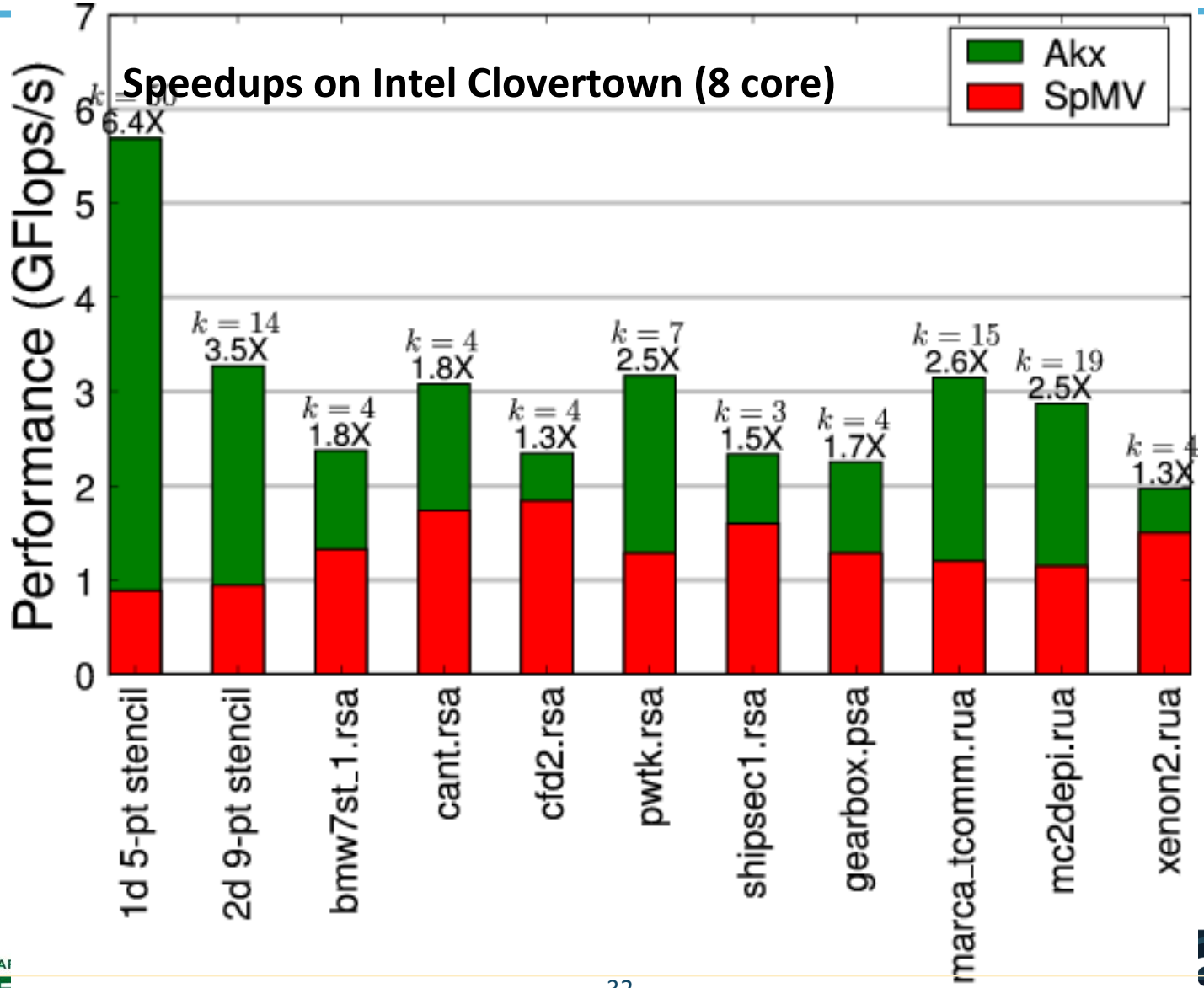


For implicit memory management (caches) uses a TSP algorithm for layout

Joint work with Jim Demmel, Mark Hoemman, Marghoob Mohiyuddin

- **Saves communication for "well partitioned" matrices**
  - Serial: O(1) moves of data moves vs. O(k)
  - Parallel: O(log p) messages vs. O(k log p)

Bigger Kernel ($A^k x$) Runs at Faster Speed than Simpler (Ax)



Lessons #3: Understand Numerics (Or work with someone who does)

Don’t be afraid to change to a “different” right answer

Minimizing Communication of GMRES to solve Ax=b

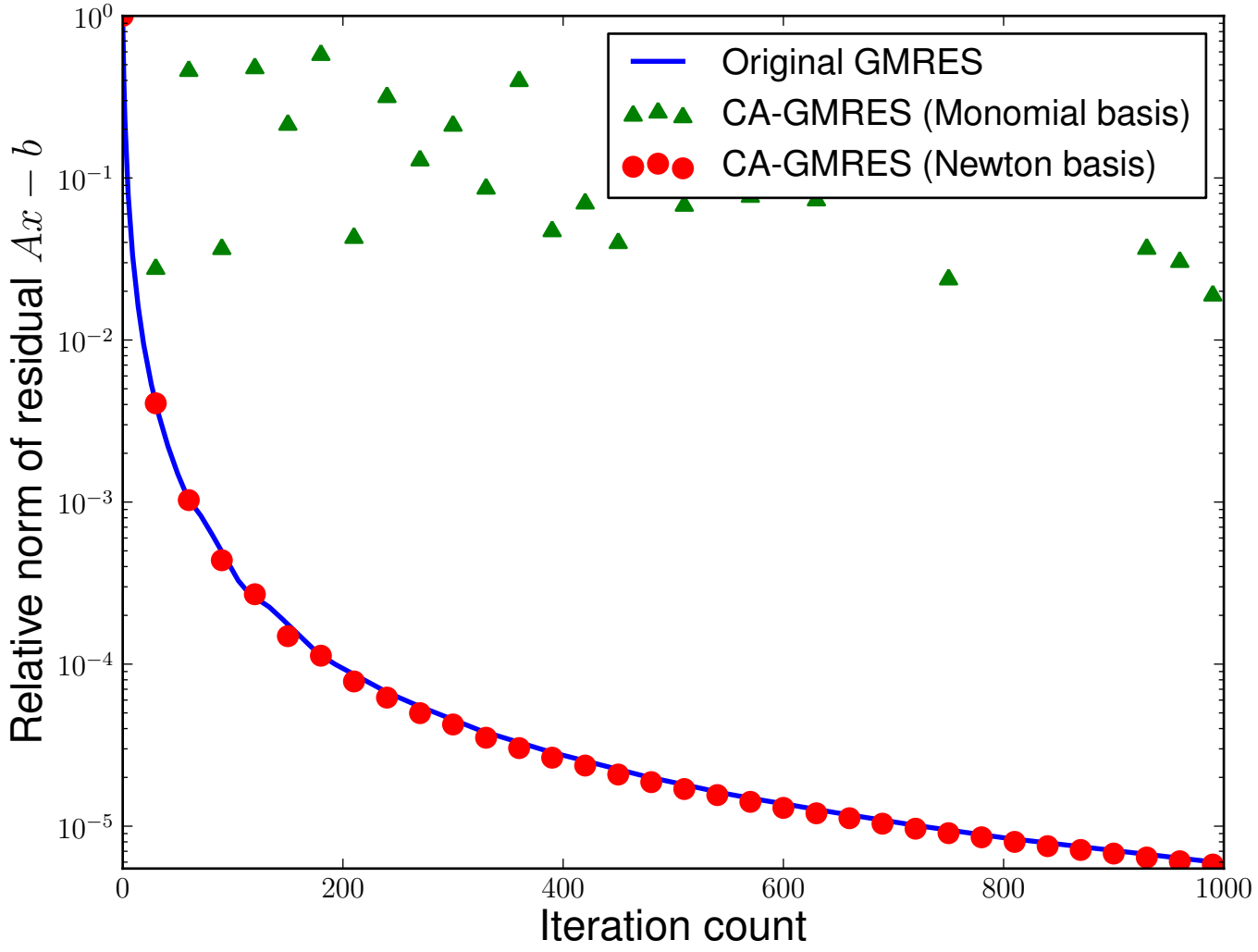
- **GMRES: find x in span{b,Ab,…,A^kb} minimizing || Ax-b ||$_2$**

Standard GMRES
for i=1 to k
w = A · v(i-1) … *SpMV*
MGS(w, v(0),…,v(i-1))
update v(i), H
endfor
solve LSQ problem with H

Communication-avoiding GMRES
W = [v, Av, A^2v, … , A^kv]
[Q,R] = TSQR(W)
… *"Tall Skinny QR"*
build H from R
solve LSQ problem with H

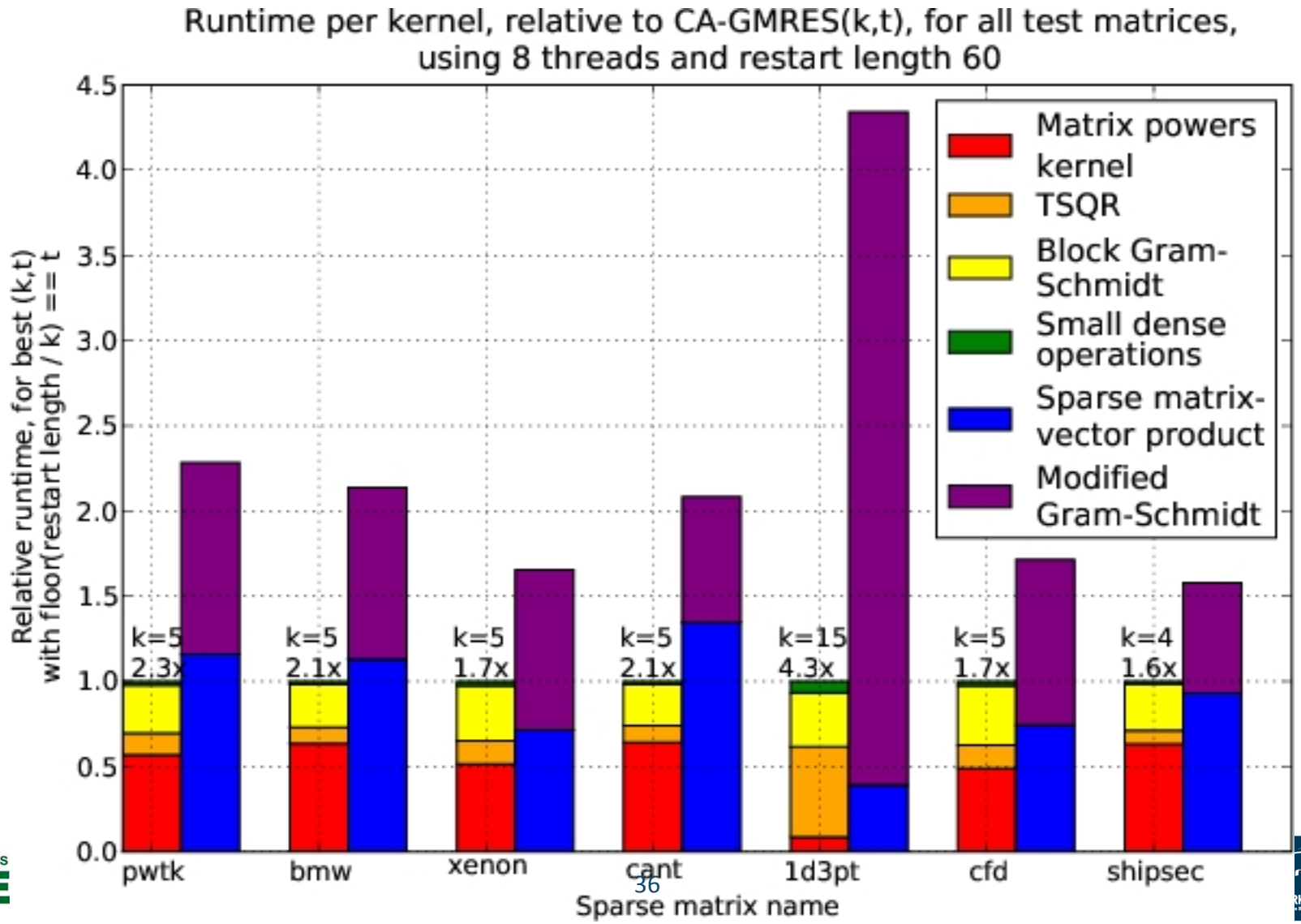
Sequential case: #words moved decreases by a factor of k
Parallel case: #messages decreases by a factor of k

Matrix Powers Kernel (and TSQR) in GMRES



Communication-Avoiding Krylov Method (GMRES)

Performance on 8 core Clovertown

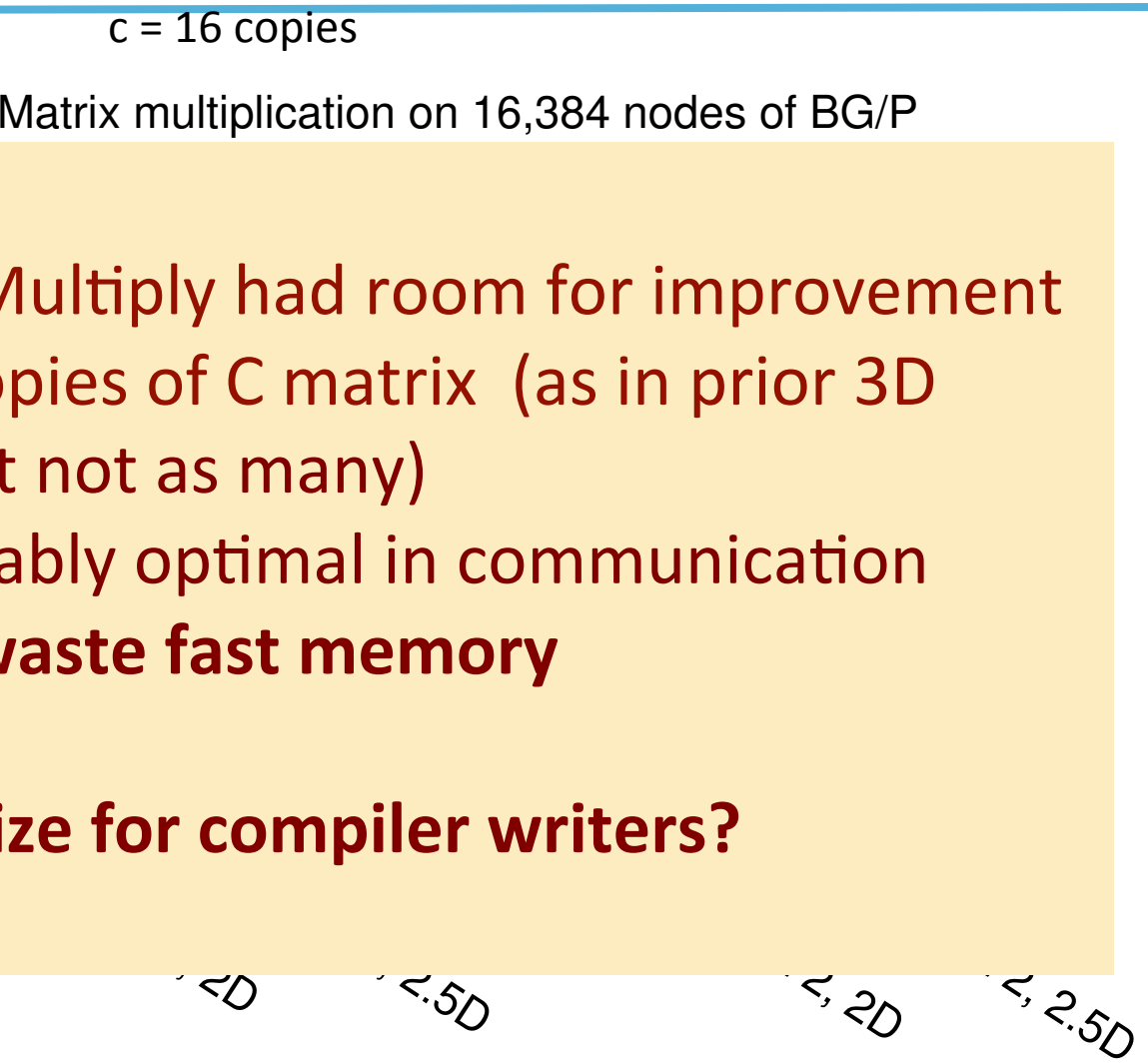


Lessons #3: Never Waste Fast Memory

Don't get hung up on the "owner computes" rule.

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

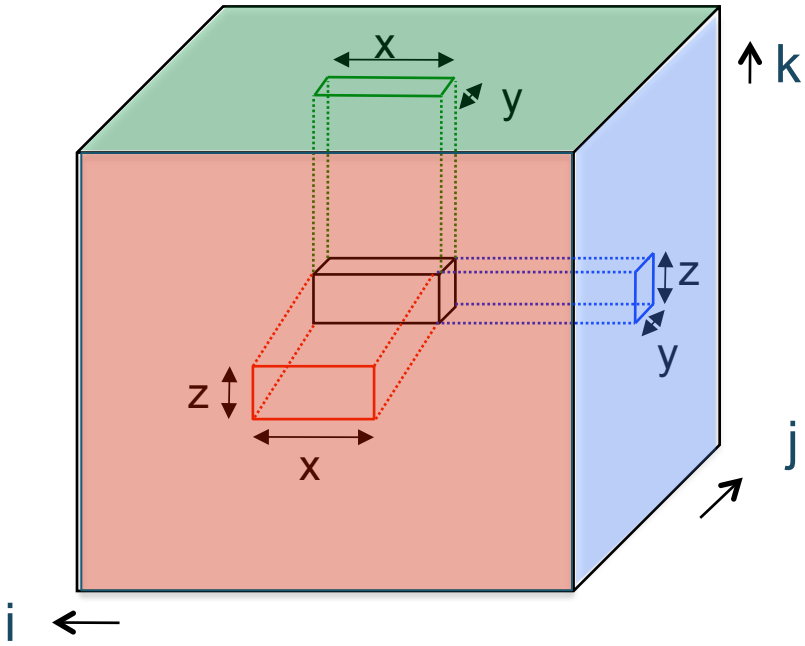
Can we generalize for compiler writers?

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**

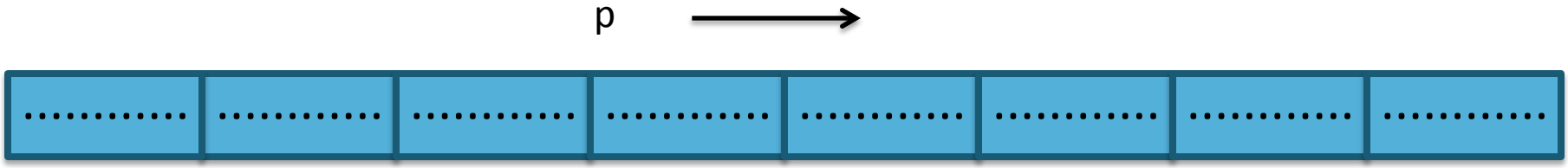
Matrix Multiplication code has a 3D iteration space
Each point in the space is a constant computation (*/+)

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

Lessons #4: Understand Theory

Lower bounds help identify optimizations.

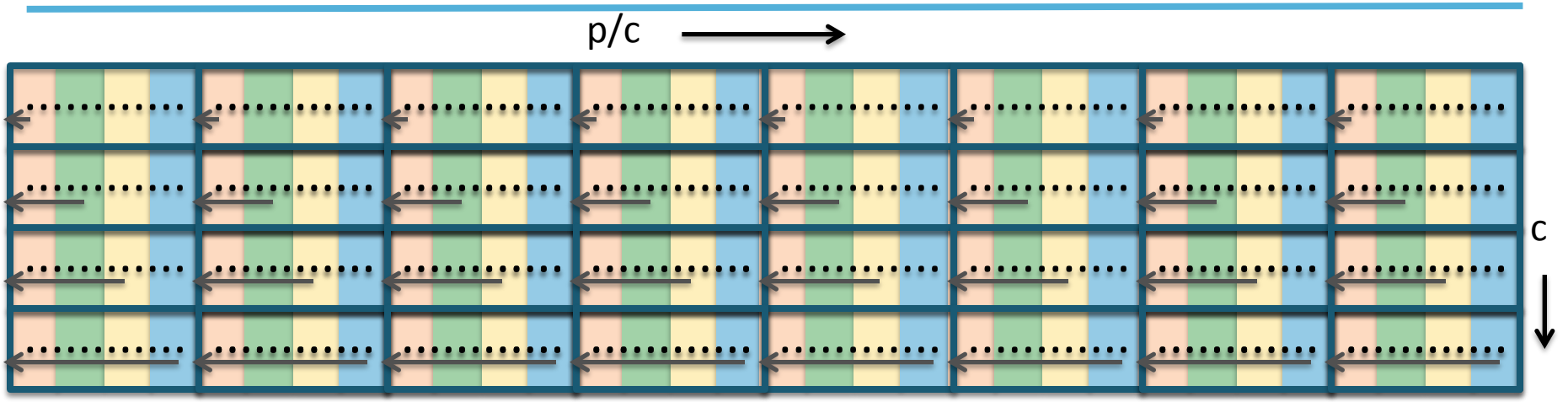
Traditional (Naïve n^2) Nbody Algorithm (using a 1D decomposition)



- Given n particles and p processors, size M memory
- Each processor has n/p particles
- Algorithm: shift copy of particles to the left p times, calculating all pairwise forces
- Computation cost: n^2/p
- Communication cost: O(p) messages, O(n) words

IPDPS'13 paper (Driscoll, Georganas, Koanantakool, Solomonik, Yelick)

Communication Avoiding Version (using a "1.5D" decomposition)



- **Divide p into c groups. Replicate particles within group.**
  - First row responsible for updating all by orange, second all by green,…
- **Algorithm: shift copy of n/(p*c) particles to the left**
  - Combine with previous data before passing further level (log steps)
- **Reduce across c to produce final value for each particle**
- Total Computation: $O(n^2/p)$;
- Total Communication: O(log(p/c) + log c) messages,
  O(n*(c/p+1/c)) words

Limit: $c \leq p^{1/2}$

In theory there is no difference between theory and practice, but in practice there is.

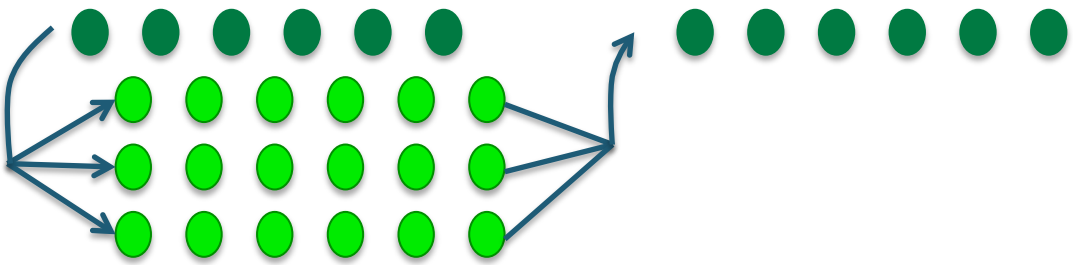
-- Jan L. A. van de Snepscheut, Computer Scientist

or

-- Yogi Berra, Baseball player and manager

Generalizing Communication Optimal Transformations to Arbitrary Loop Nests

1.5D N-Body: Replicate and Reduce



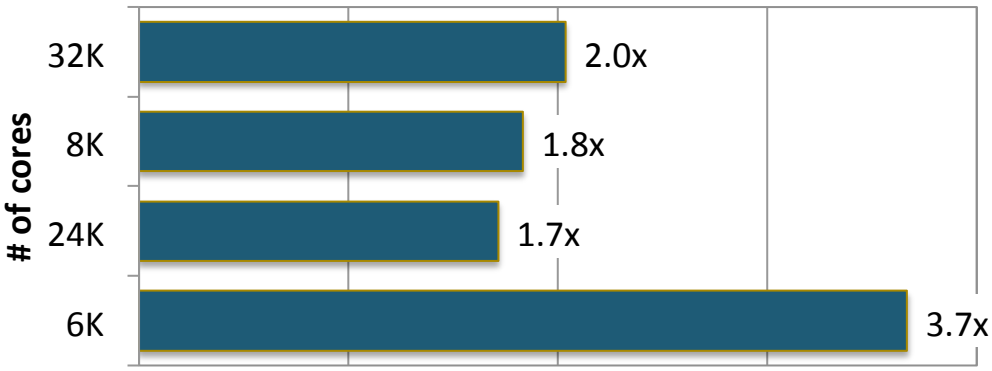
The same idea (replicate and reduce) can be used on (direct) N-Body code:

1D decomposition → "1.5D"

Does this work in general?

- *Yes, for certain loops and array expressions*
- *Relies on basic result in group theory*
- *Compiler work TBD*

Speedup of 1.5D N-Body over 1D



IPDPS'13 paper (Driscoll, Georganas, Koanantakool, Solomonik, Yelick)

Have We Seen this Idea Before?

- **These algorithms also maximize parallelism beyond "domain decomposition"**
  - SIMD machine days
- **Automation depends on associative operator for updates (e.g., M. Wolfe)**
- **Also used for "synchronization avoidance" in Particle-in-Cell code (Madduri, Su, Oliker, Yelick)**
  - Replicate and reduce optimization given p copies
  - Useful on vectors / GPUs

Lessons #5: Aggregate Communication

Pack messages to better amortize per-message communication costs.

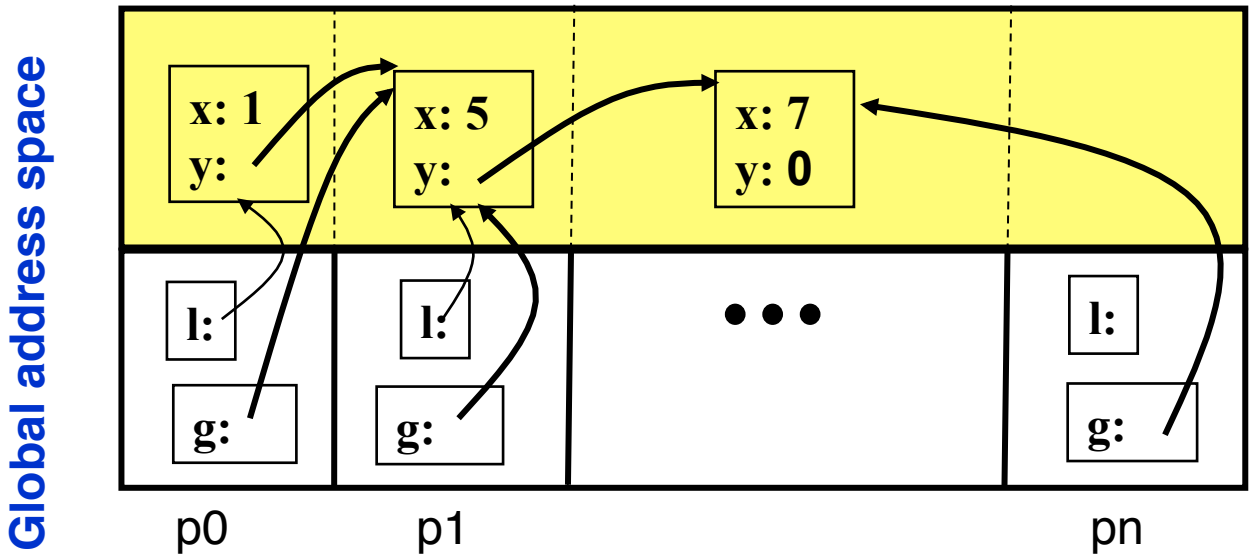
Lessons #6: Overlap and Pipeline Communication

Sometimes runs contrary to lesson #5 on aggregation

What is a PGAS Programming Model? Partitioned Global Address Space

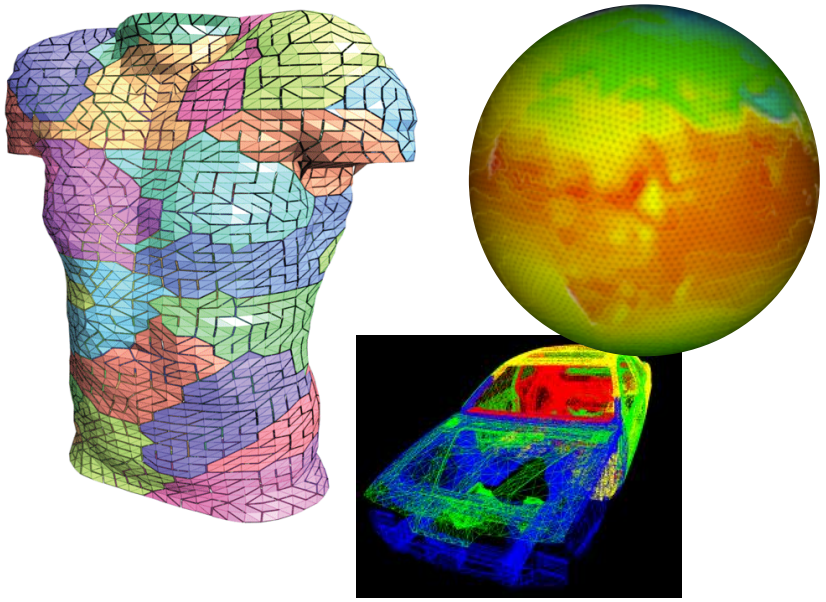
Global Address Space: **directly read/write remote data**

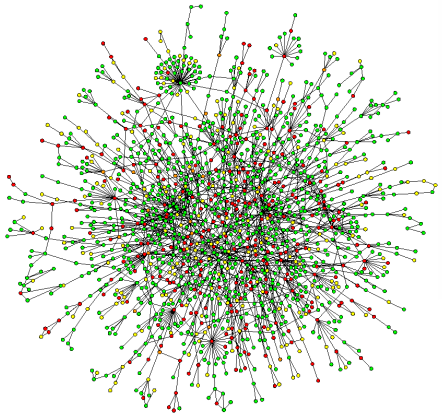
Partitioned: **data is designated as local or global**

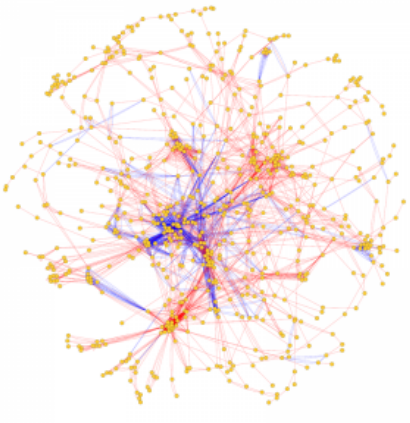


- **PGAS ≠ cache coherent shared memory**
- **PGAS ≠ data parallel (heroic compilers not required)**
- **Affinity control through partitioning ➔ scalability**
- **Multiple PGAS libraries (GA) and languages (UPC, CAF, Titanium, X10, Chapel, Sequoia, PyGAS)**

Latency really matters in irregular problems: Drives Programming Models







Message Passing Programming

Divide up domain in pieces

Compute one piece and exchange

PVM, MPI, and many libraries

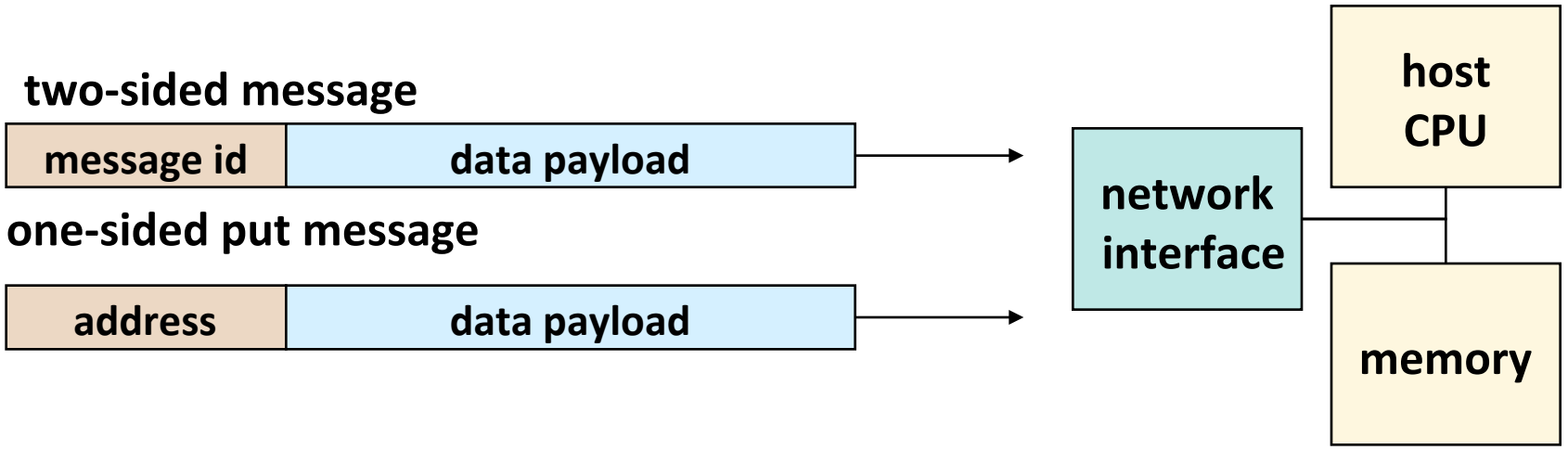
Global Address Space Programming

Each start computing

Grab whatever / whenever

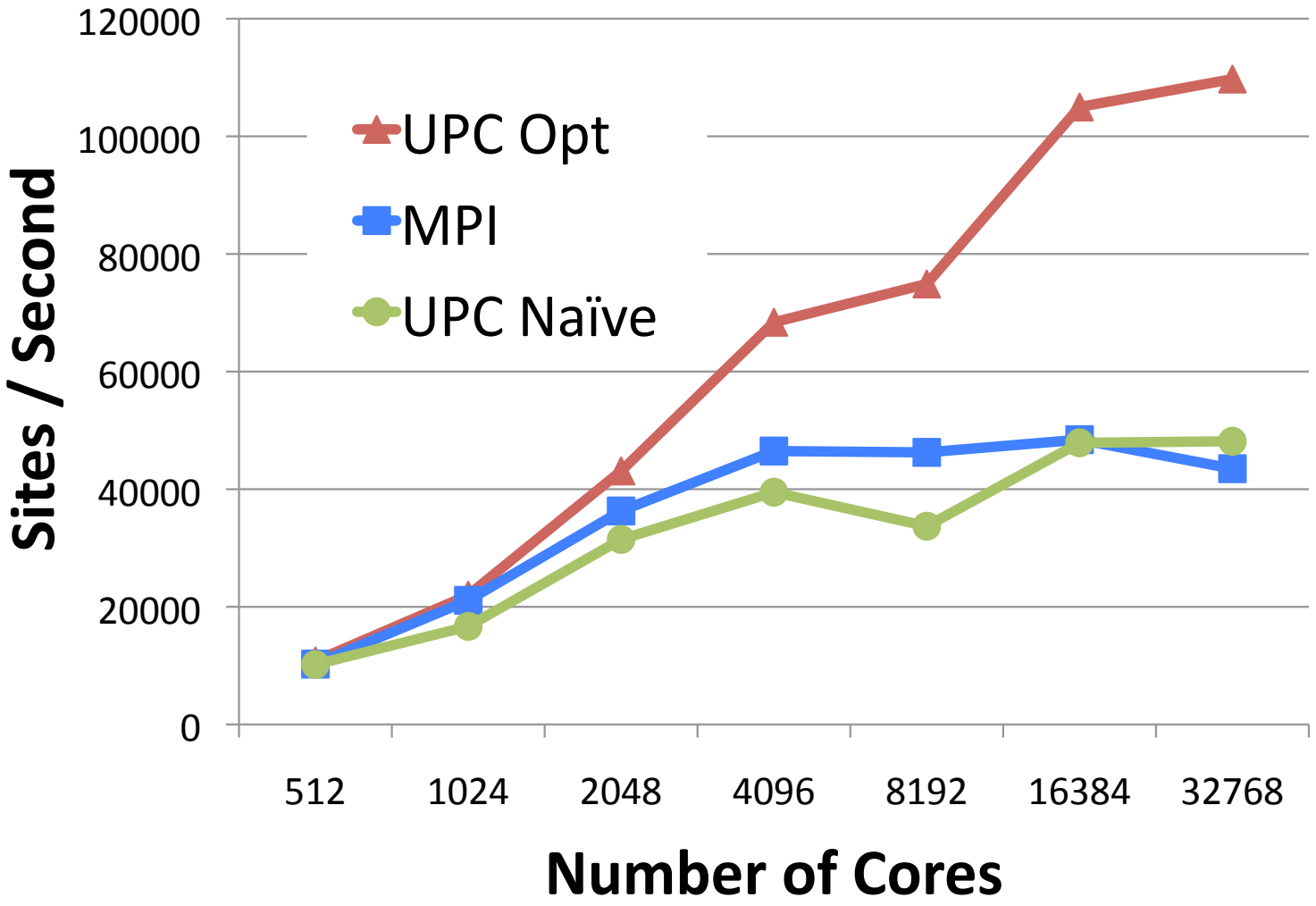
UPC, CAF, X10, Chapel, Fortress, Titanium, GA,

Avoiding Synchronization in Communication



- **Two-sided message passing (e.g., MPI) requires matching a send with a receive to identify memory address to put data**
  - Wildly popular in HPC, but cumbersome in some applications
  - Couples data transfer with synchronization
- **Using global address space decouples synchronization**
  - Pay for what you need!
  - Note: Global Addressing ≠ Cache Coherent Shared memory

UPC at Scale on the MILC (QCD) App

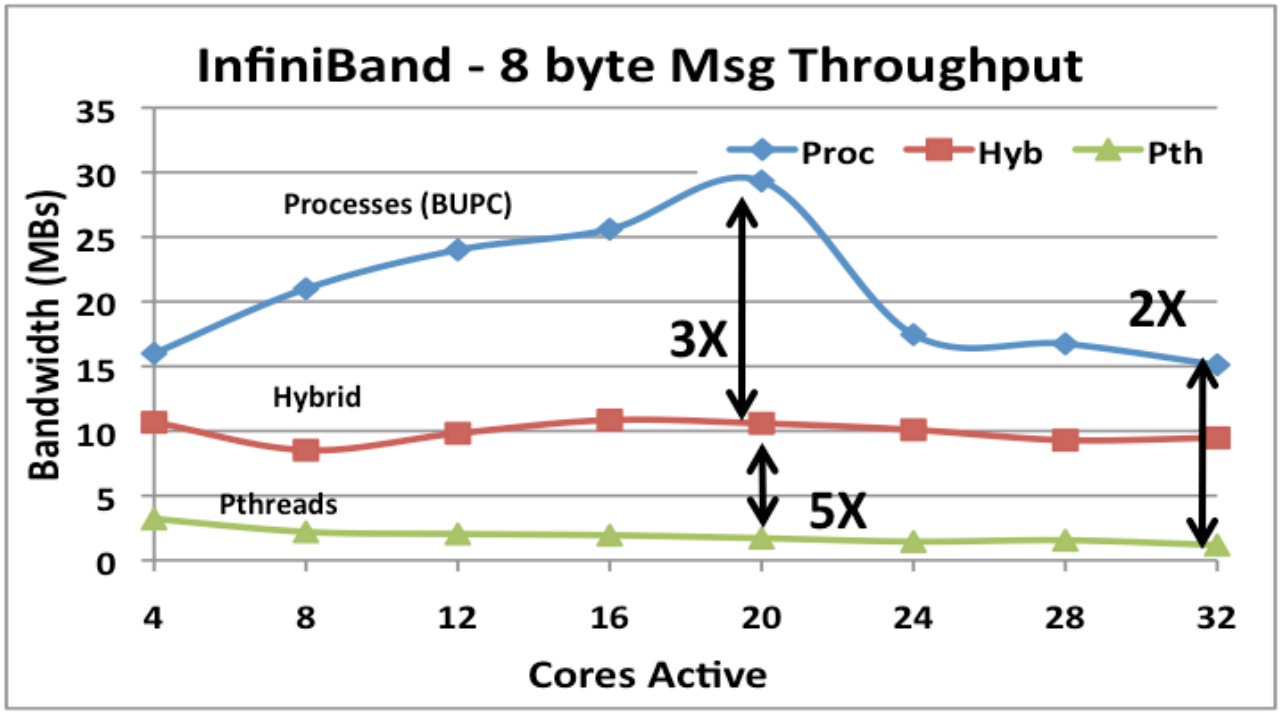


Shan, Austin, Wright, Strohmaier, Shalf, Yelick (unpublished)

Communication costs will also lead to under-provisioned networks

Management of critical resources is increasingly important:

- **Memory and network bandwidth limited** by cost and energy
- **Capacity limited at many levels:** network buffers at interfaces, internal network congestion are real and growing problems



Even at modest core counts, interface to network can be bottleneck

(Iancu, Khaled, Hofmeyr)

Exascale challenges

There are many challenges facing next generation scientific computing

- **Scaling**
- **Synchronization,**
- **Dynamic system behavior**
- **Irregular algorithms**
- **Resilience**

don't forget what's important

Location, Location, Location

Communication Hurts!