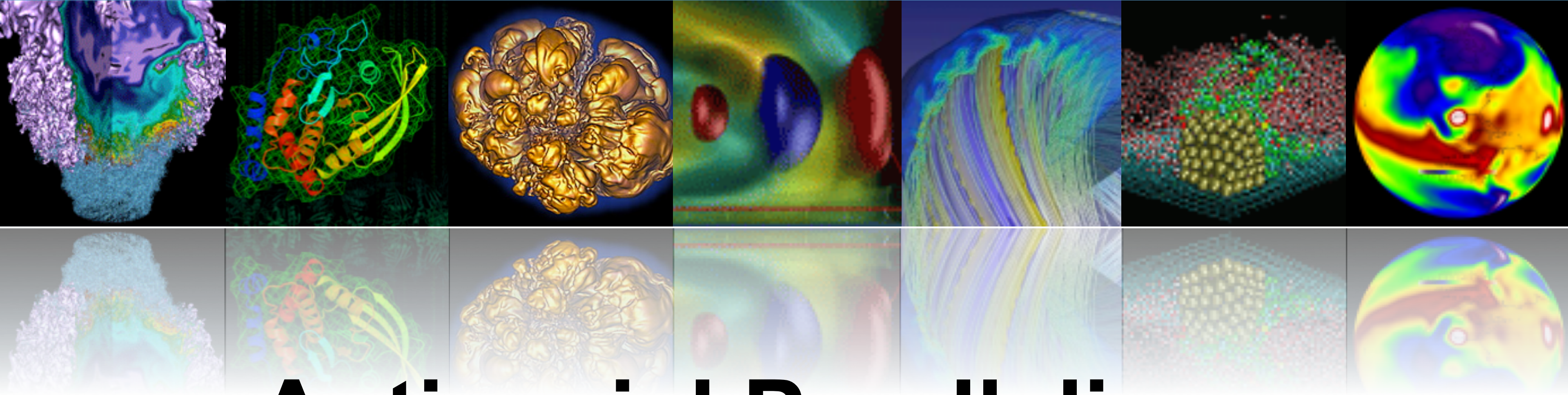


Antisocial Parallelism:
Avoiding, Hiding and Managing Communication

Kathy Yelick

Associate Laboratory Director of Computing Sciences

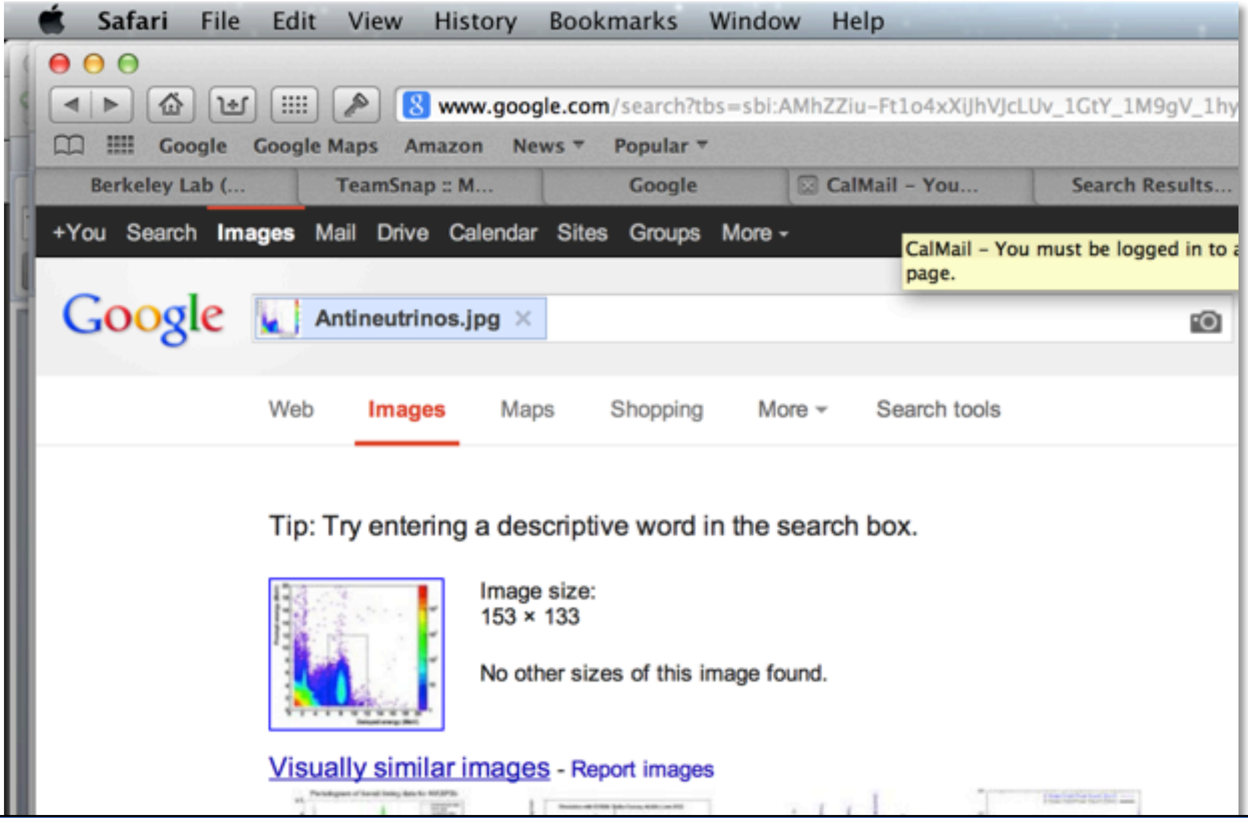
Lawrence Berkeley National Laboratory

EECS Professor, UC Berkeley

"Big Data" Changes Everything…What about Science?

BERKELEY LAB

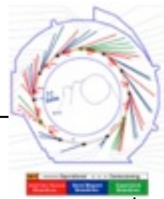
Transforming Science: Finding Data

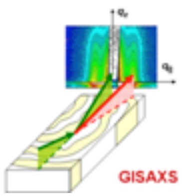


Computing Challenges:

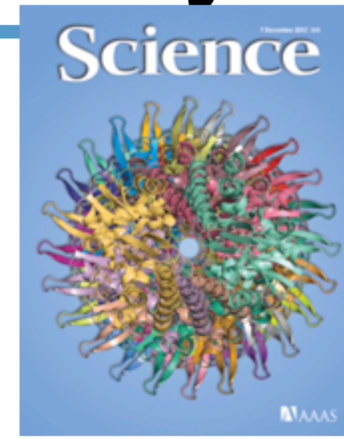
- Search for scientific data on the web
- Automated metadata annotation / feature identification
- Data: images, genomes, simulations, MRI, MassSpec,…

Scientific Workflow Today



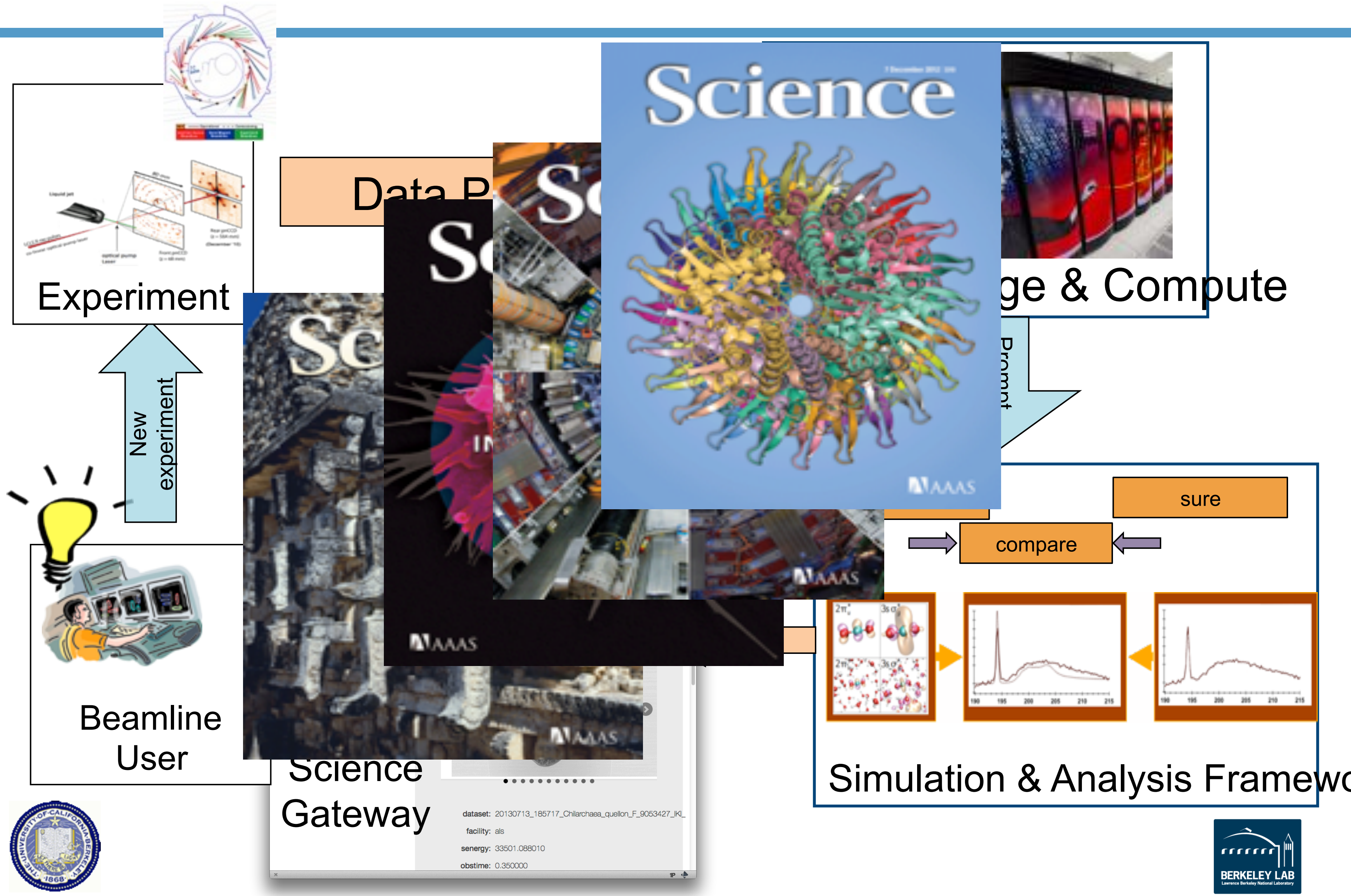


Experiment



Beamline User

The Future of Experimental Science

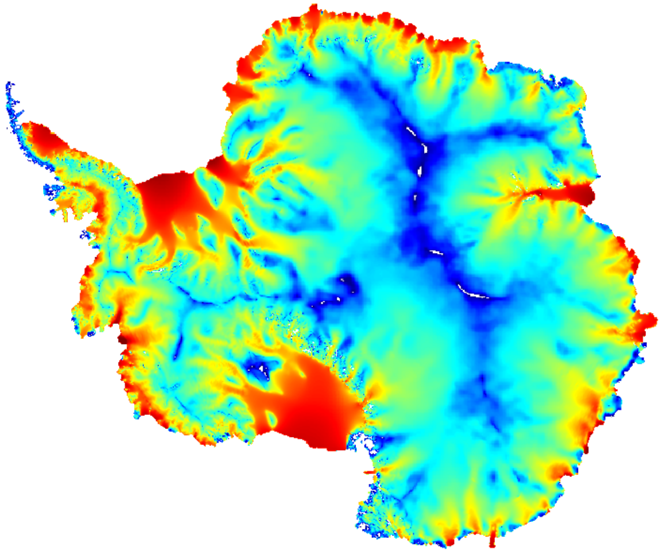


Transforming experimental science: "Superfacility" for Science

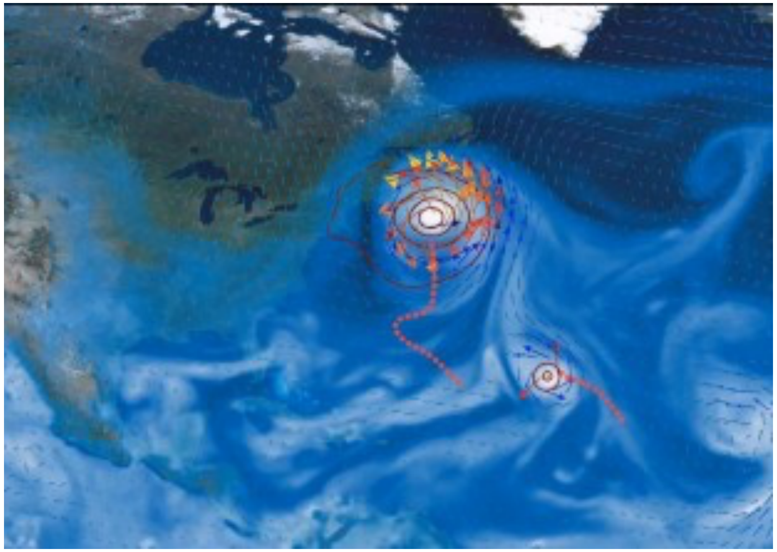
Computing Challenges:

- **Robotics, Special purpose processors at experiments**
- **Mathematics / algorithm for real-time and offline analysis**
- **Massive numbers of simulations for inverse problems**
- **Networks and software for data movement, management**

Science at the Boundary of Simulation and Observation



Adaptive Mesh Refinement simulates sea level impacts from melting of West Antarctic Ice Sheet



Deep learning algorithms identify and help quantify extreme events

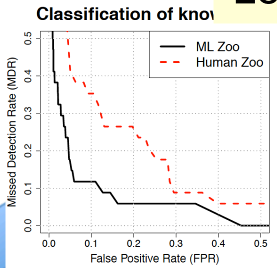
Computing Challenges:

- **Multimodal analysis from sensors, genomes, images…**
- **High performance methods and implementations**
- **Data-driven simulations to predict regional effects on environment and weather events**

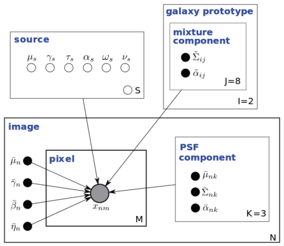
Finding smaller signals in noisy, biased data: Removing Systematic Bias in Cosmology

Filtered

Crowd sourced

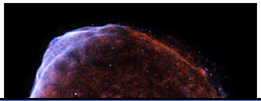


Machine Learning

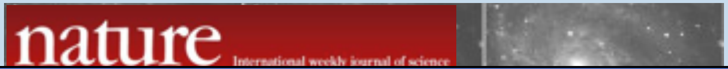


Graphical models

New simulation models and AMR code (Nyx)



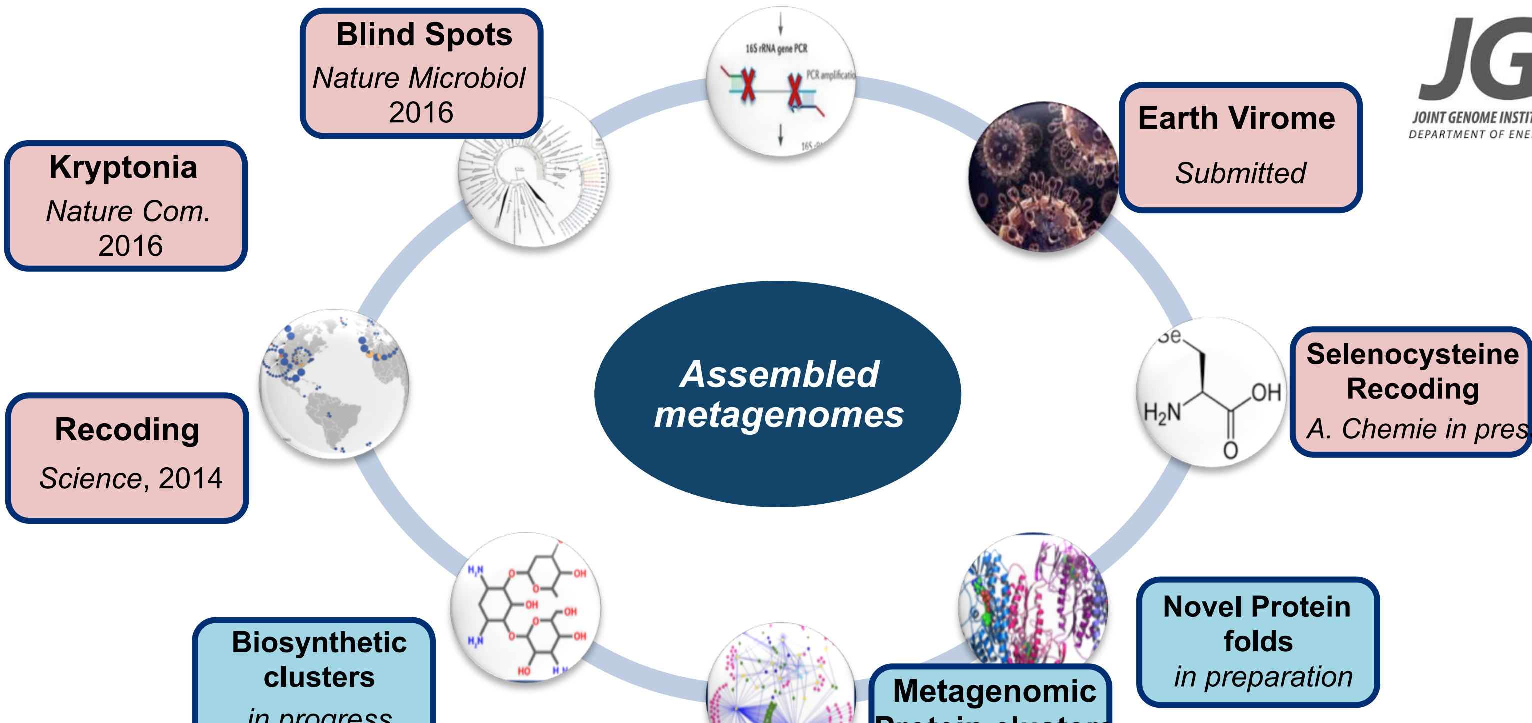
Example: Astrophysicists discover early nearby



Computing Challenges:

- **Better machine learning for event detection**
- **Removing systematic bias in experimental data**
- **Simulations to interpret data; data constrain simulations**

Finding structure and function in noisy data: Metagenomics data mining



Computing Challenges:

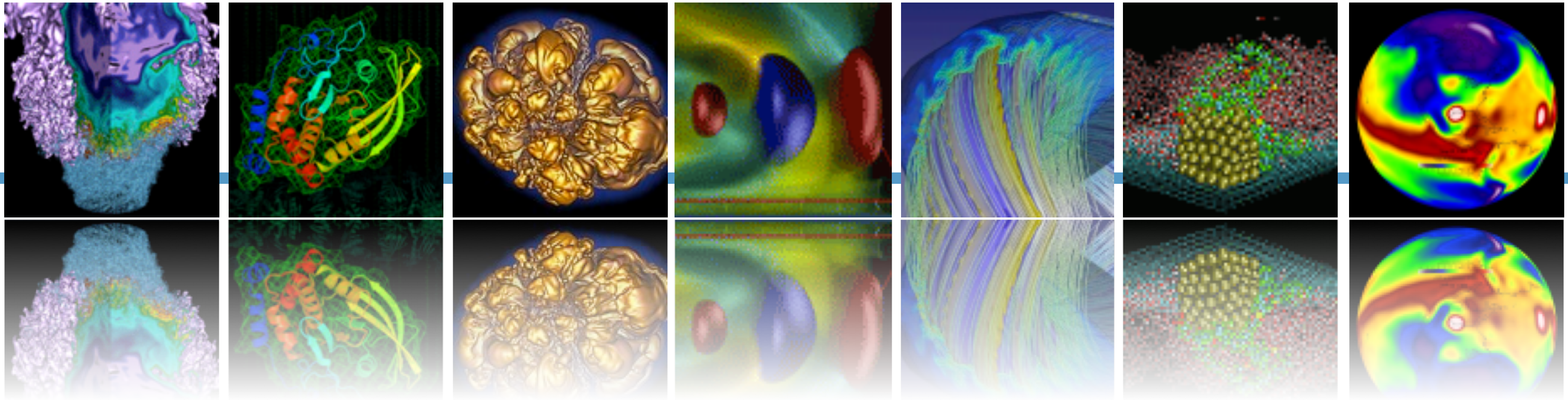
- Distributed memory graph algorithms / hash tables
- Low latency interconnects; low overhead communication
- Algorithms to separate and assembly genomes
- Many-to-Many comparisons against databases

Science Trends

- Science needs (and will always need) more computing
- New science questions at the boundary of simulation and observation
- Changes to computing infrastructure needed for open, reproducible science

Roadmap

✔ Science Trends
- Political Trends
- Technology Trends
- Algorithmic Challenges



The Politics of High Performance Computing

White House Announces the National Strategic Computing Initiative (NSCI)

THE WHITE HOUSE

Office of the Press Secretary

For Immediate Release July 29, 2015

EXECUTIVE ORDER

\- - - - - - -

CREATING A NATIONAL STRATEGIC COMPUTING INITIATIVE

By the authority vested in me as President by the Constitution and the laws of the United States of America, and to maximize benefits of high-performance computing (HPC) research, development, and deployment, it is hereby ordered as follows:

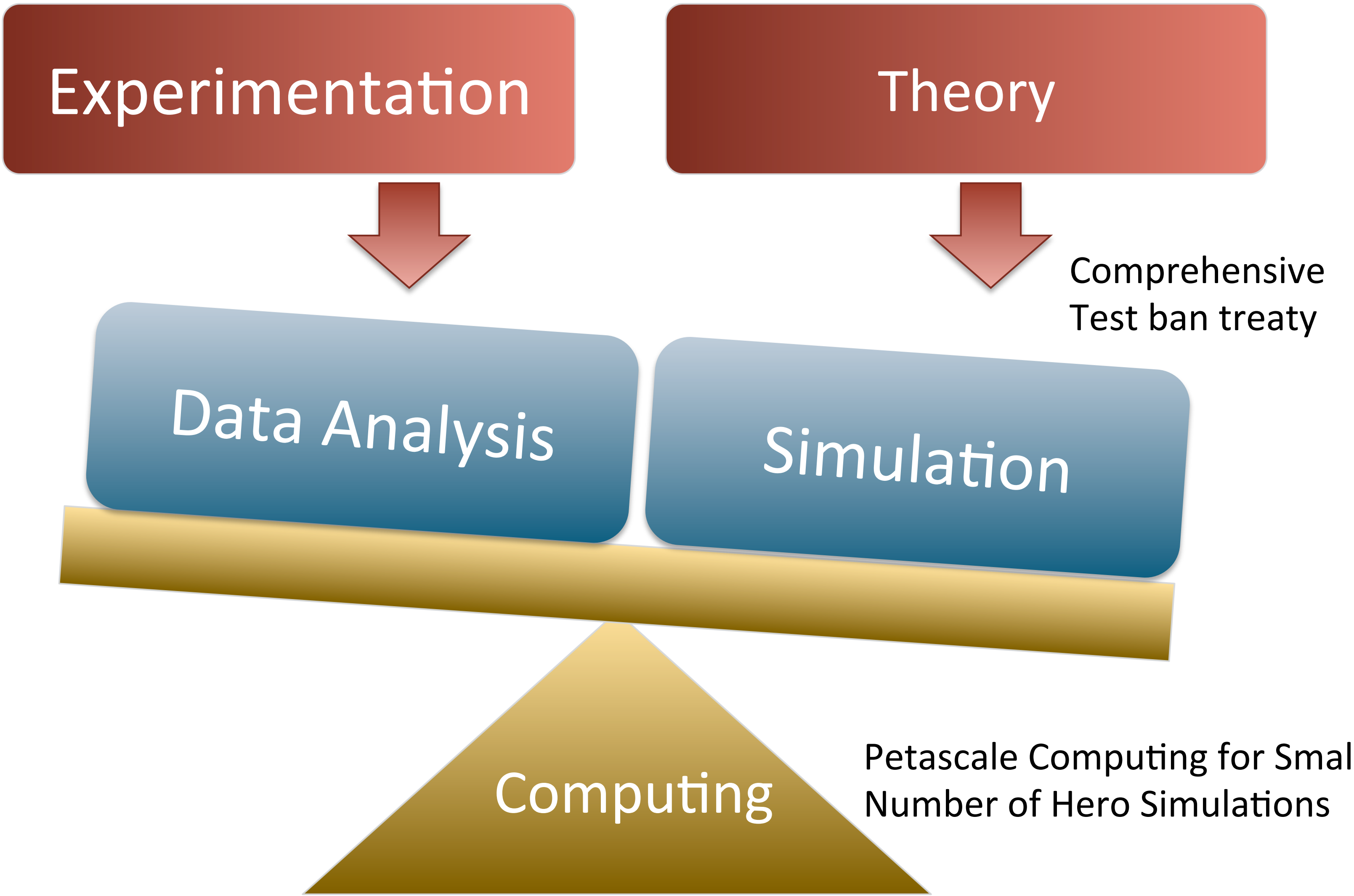
Section 1. Policy. In order to maximize the benefits of HPC for economic competitiveness and scientific discovery, the United States Government must create a coordinated Federal strategy in HPC research, development, and deployment. Investment in HPC has contributed substantially to national economic prosperity and rapidly accelerated scientific discovery. Creating and deploying technology at the leading edge is vital to advancing my Administration's priorities and spurring innovation. Accordingly, this order establishes the National Strategic Computing Initiative (NSCI). The NSCI is a

Five goals:

1. **Create systems that can apply exaflops of computing power to exabytes of data.**
2. **Keep the United States at the forefront of HPC capabilities.**
3. **Improve HPC application developer productivity.**
4. **Make HPC readily available.**
5. **Establish hardware technology for future HPC systems.**

[DOE SC and NNSA] will execute a joint program focused on advanced simulation through a capable exascale computing …

Advanced Computing: Not just for Simulation



Science Needs Computing for Both Experiments (Data) and Theory (Modeling and Simulation)

Experimentation

Theory

Commercial “Big Data”
Growth in Sequencers,
CCDs, etc.

Data Analysis

Simulation

Computing foundation includes
research (math/stat and CS) and
facilities (data and compute)

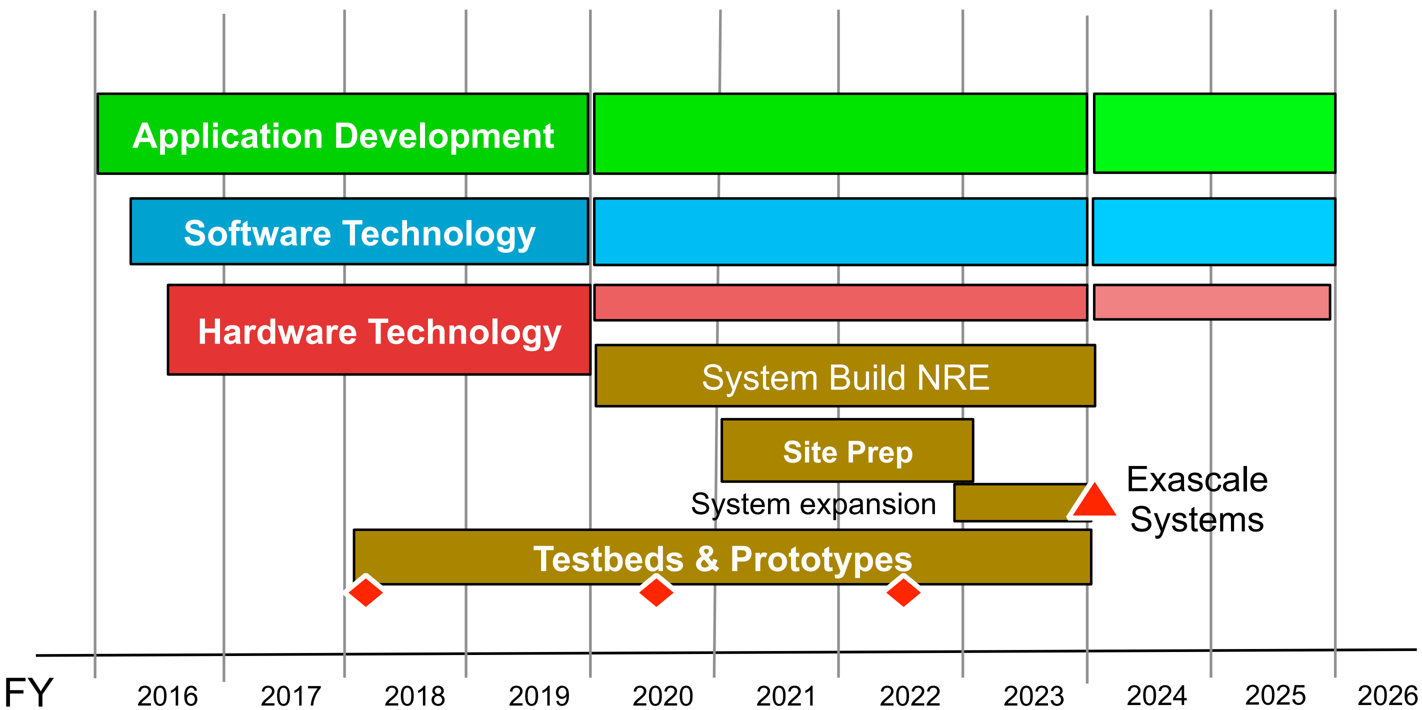
Computing

Future Performance from
Exascale Technology

US DOE Exascale Computing Project (ECP)

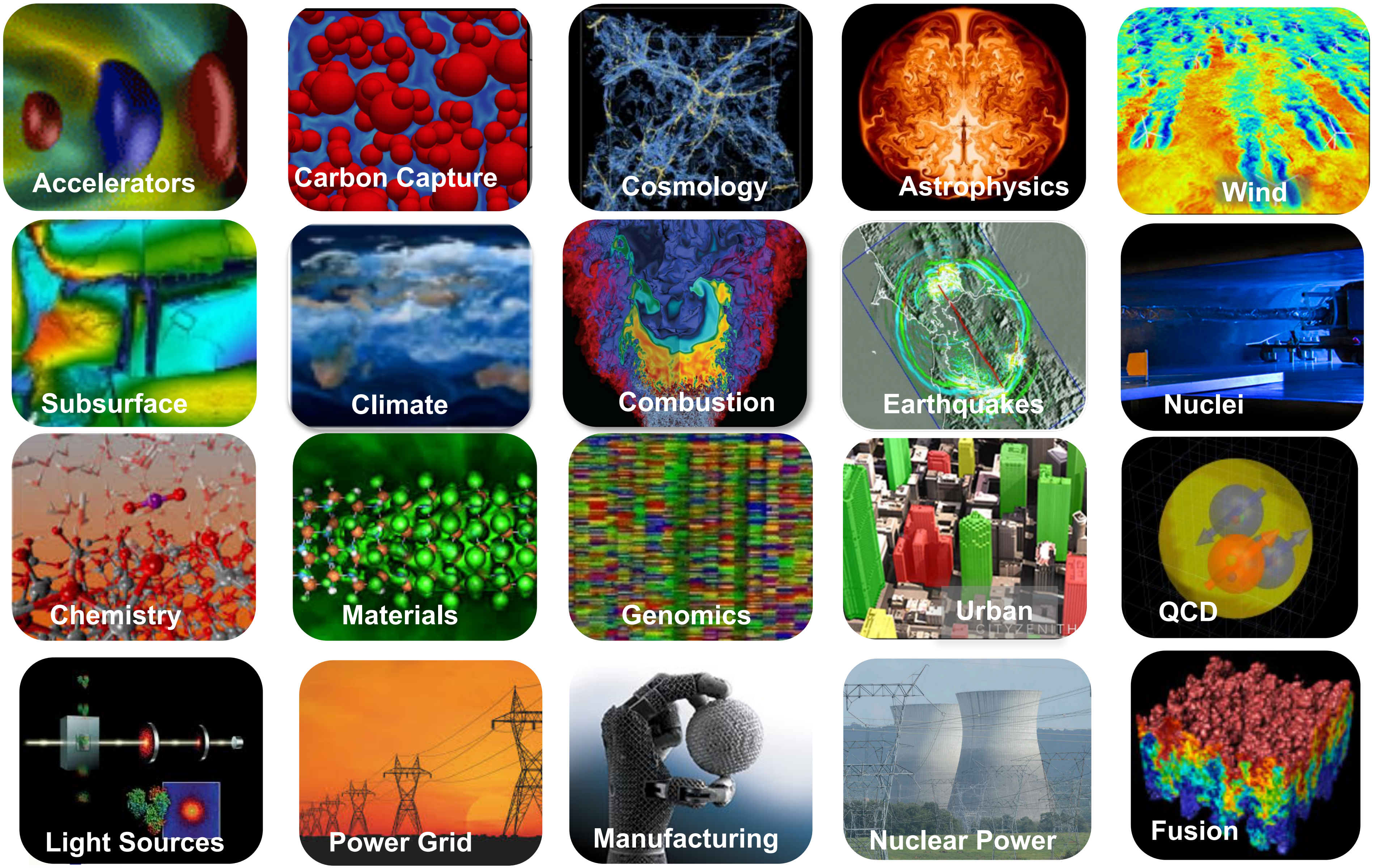
The Project has three phases:

- Phase 1 – R&D before DOE facilities exascale systems RFP in 2019
- Phase 2 – Exascale architectures and NRE are known. Targeted development
- Phase 3 – Exascale systems delivered. Meet Mission Challenges



EXASCALE COMPUTING PROJECT

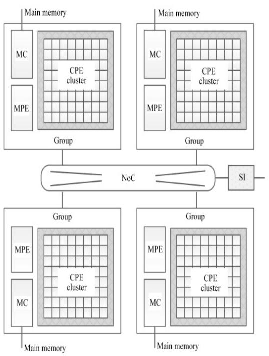
Proposed DOE Exascale Science Problems

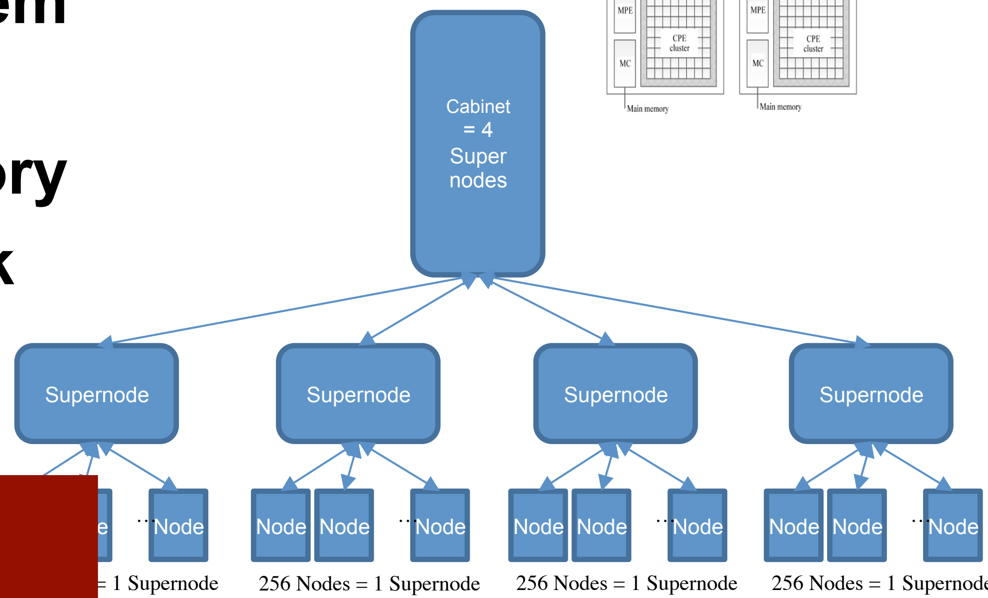


#	Site	Manufacturer	Computer	Country	Cores	Rmax [Pflops]	Power [MW]
1	National Supercomputing Center in Wuxi	NRCPC	**Sunway TaihuLight** NRCPC Sunway SW26010, 260C 1.45GHz	China	10,649,600	93.0	15.4
2	National University of Defense Technology	NUDT	**Tianhe-2** NUDT TH-IVB-FEP, Xeon 12C 2.2GHz, IntelXeon Phi	China	3,120,000	33.9	17.8
3	Oak Ridge National Laboratory	Cray	**Titan** Cray XK7, Opteron 16C 2.2GHz, Gemini, NVIDIA K20x	USA	560,640	17.6	8.21
4	Lawrence Livermore National Laboratory	IBM	**Sequoia** BlueGene/Q, Power BQC 16C 1.6GHz, Custom	USA	1,572,864	17.2	7.89
5	RIKEN Advanced Institute for Computational Science	Fujitsu	**K Computer** SPARC64 VIIIfx 2.0GHz, Tofu Interconnect	Japan	795,024	10.5	12.7
6	Argonne National Laboratory	IBM	**Mira** BlueGene/Q, Power BQC 16C 1.6GHz, Custom	USA	786,432	8.59	3.95
7	Los Alamos NL / Sandia NL	Cray	**Trinity** Cray XC40, Xeon E5 16C 2.3GHz, Aries	USA	301,0564	8.10	4.23
8	Swiss National Supercomputing Centre (CSCS)	Cray	**Piz Daint** Cray XC30, Xeon E5 8C 2.6GHz, Aries, NVIDIA K20x	Switzer-land	115,984	6.27	2.33
9	HLRS – Stuttgart	Cray	**Hazel Hen** Cray XC40, Xeon E5 12C 2.5GHz, Aries	Germany	185,088	5.64	3.62
10	King Abdullah University of Science and Technology	Cray	**Shaheen II** Cray XC40, Xeon E5 16C 2.3GHz, Aries	Saudi Arabia	196,608	5.54	2.83

Sunway TaihuLight

- 125.4 Pflop/s theoretical peak
- SW26010 processor, 1.45 GHz
- Node = 260 Cores (1 socket)
  - 4 – core groups; 32 GB memory (DDR3)
- 40,960 nodes in the system
  - 10,649,600 cores total
- 1.31 PB of primary memory
- 93 Pflop/s HPL, 74% peak
- 15.3 Mwatts (6 MF/Watt)

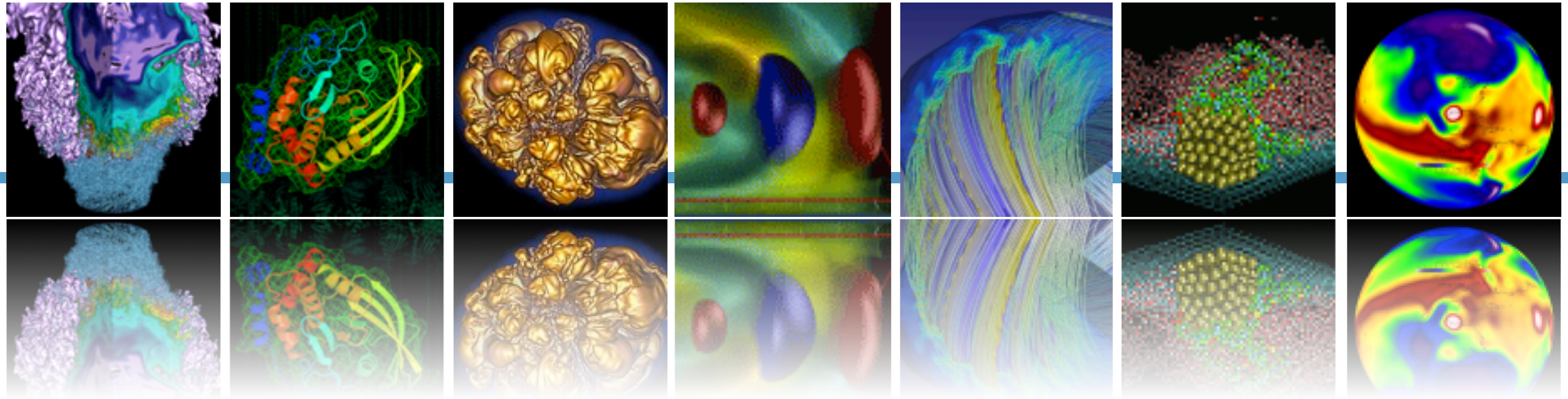




One piece of entire computing strategy on applications, fabs, architecture, software

Roadmap

✔ Science Trends
✔ Political Trends
- Technology Trends
- Algorithmic Challenges



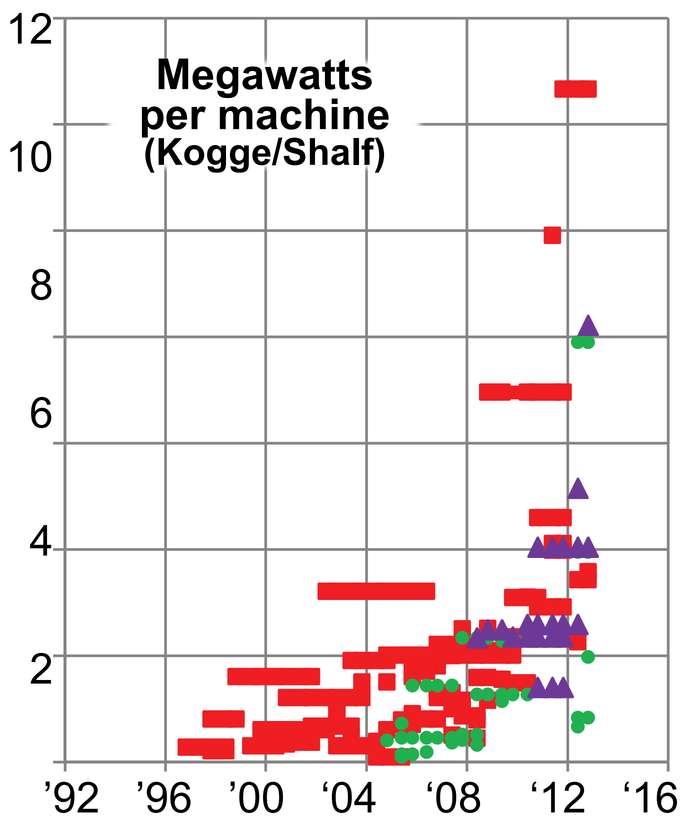
Technology Trends

Computing is energy-constrained

At ~$1M per MW, energy costs are substantial

- 1 petaflop in 2008 used 3 MW
- 1 exaflop in 2018 at 200 MW "usual **chip** scaling"

Missing Tihanhe-2 at 18MW

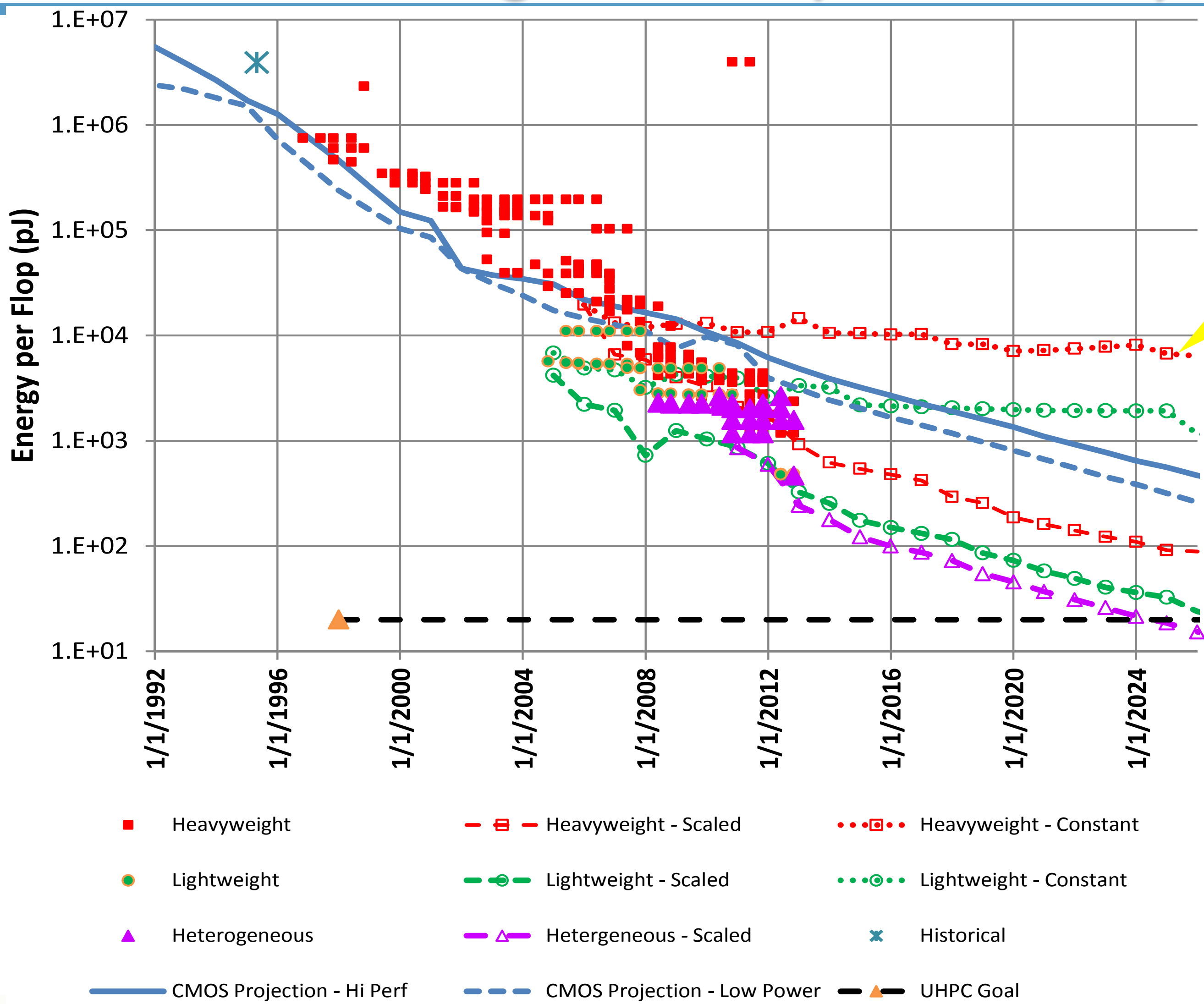


Goal: 1 Exaflop in 20 MW
= 20 pJ / operation

Note: The 20 pJ / operation is

- **Independent of machine size**
- **Independent of # cores used per application**
- **But "operations" need to be useful ones**

Multi-Core is NOT good enough! *(need to go to simpler cores)*

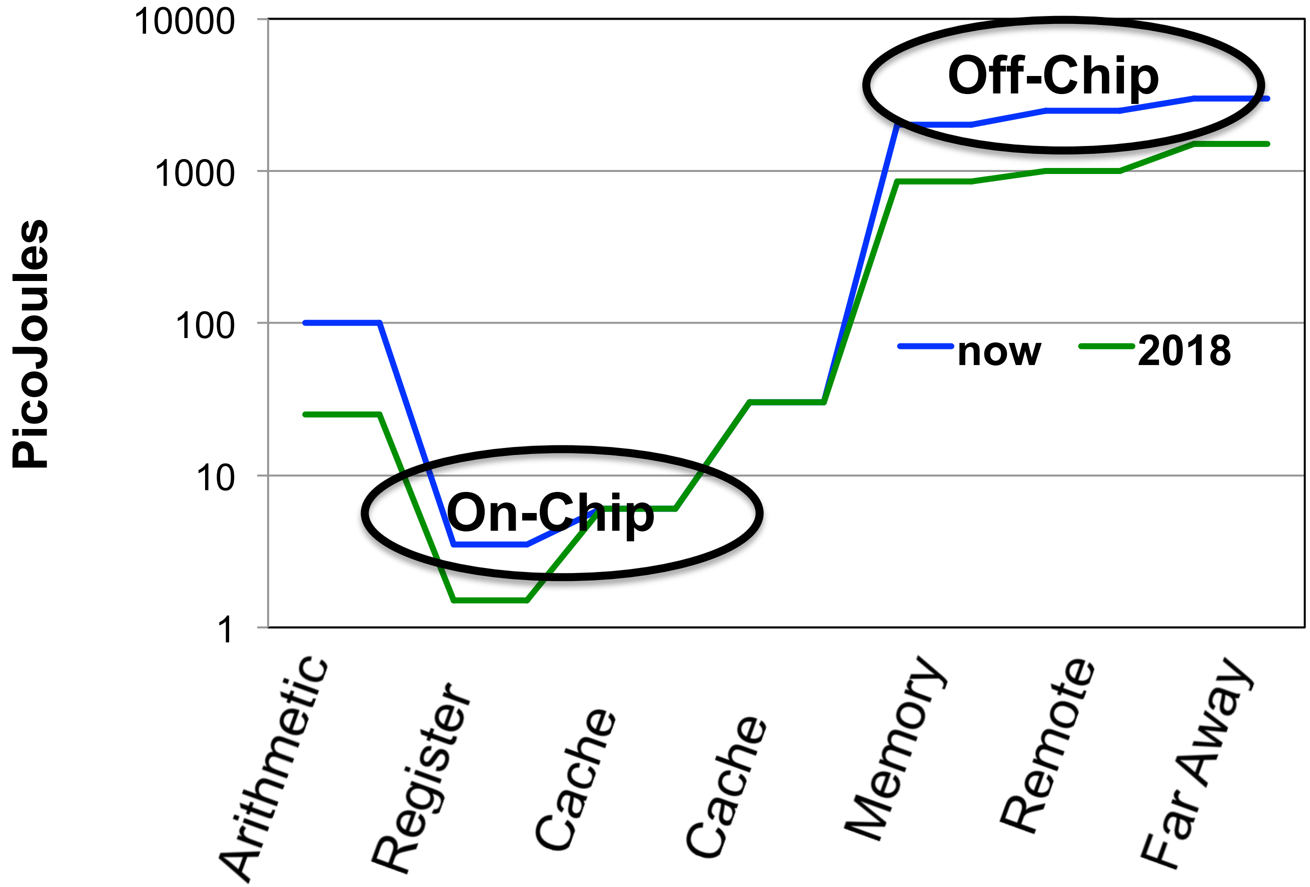


Can continue with conventional x86 architectures if you want.

Lightweight cores **OR** Hybrid is the only approach that crosses the exascale finish line

1/2

Communication Consumes Energy

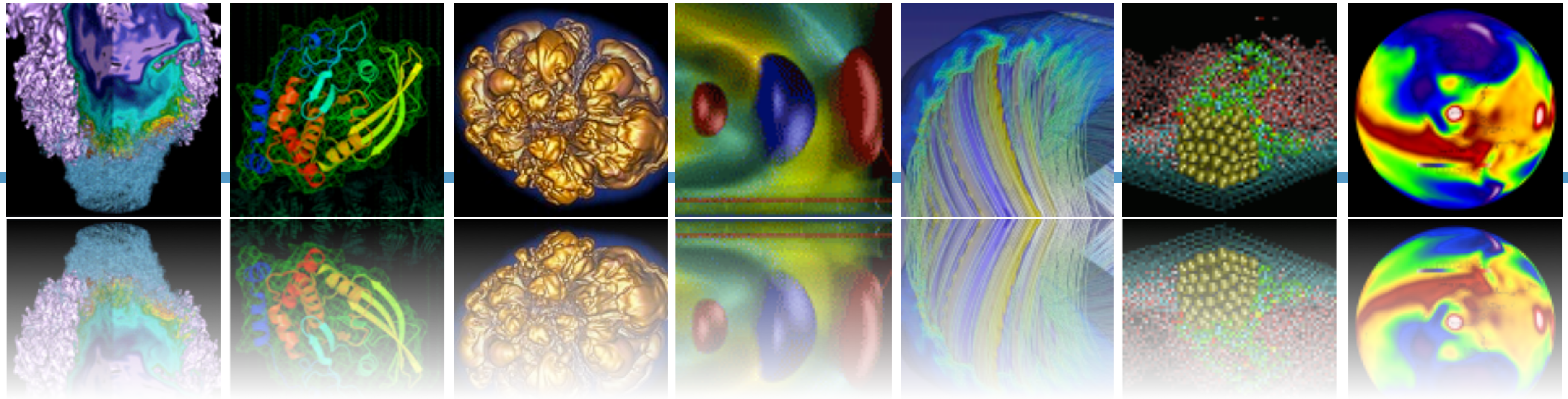


Latency is physics; bandwidth is money, …

but overhead we can fix

Roadmap

- ✔ Science Trends
- ✔ Political Trends
- ✔ Technology Trends
- **Algorithmic Challenges**

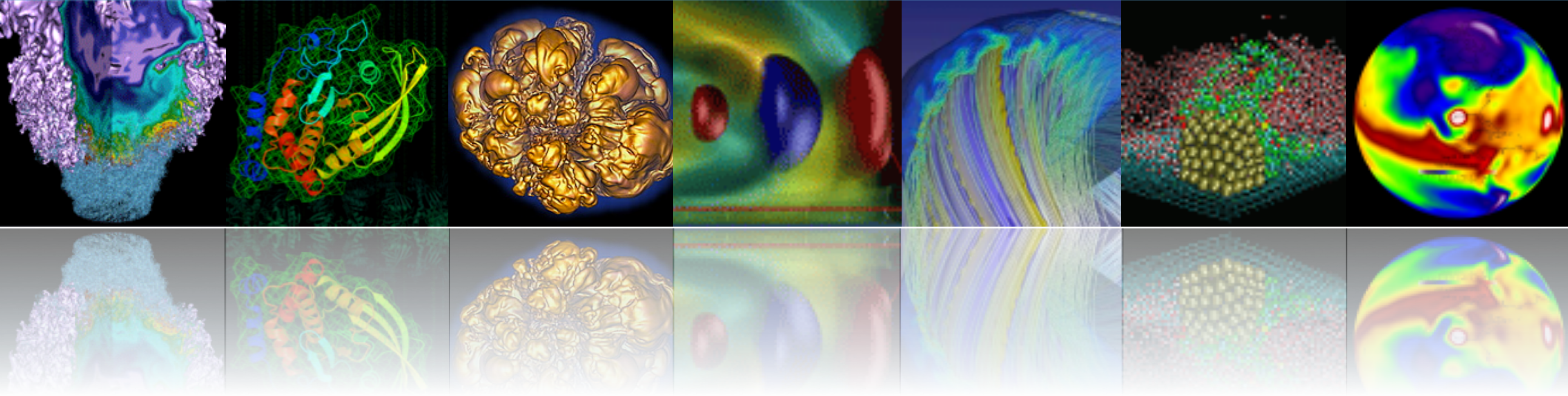


Algorithm Challenge: Communication

Analytics vs. Simulation Kernels:

7 Giants of Data	7 Dwarfs of Simulation
Basic statistics	Monte Carlo methods
Generalized N-Body	Particle methods
Graph-theory	Unstructured meshes
Linear algebra	Dense Linear Algebra
Optimizations	Sparse Linear Algebra
Integrations	Spectral methods
Alignment	Structured Meshes

There are some differences between data and simulation algorithms, but more similarities than differences. Some of the data algorithms use no arithmetic (genomics) or lower precision (deep learning)



Never Waste Fast Memory

Don’t get hung up on the “owner computes” rule.

Beyond Domain Decomposition: 2.5D Matrix Multiply

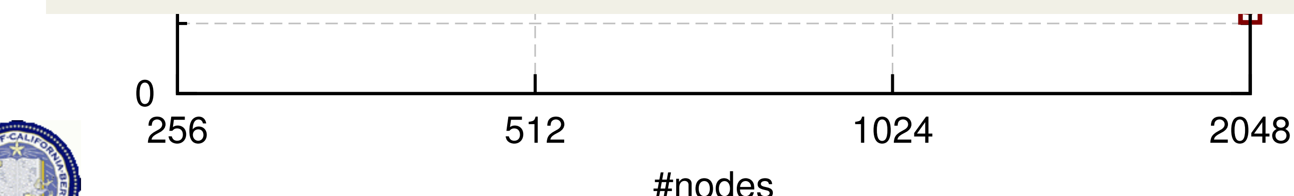
- Conventional "2D algorithms" use $P^{1/2}$ x $P^{1/2}$ mesh and minimal memory
- New "2.5D algorithms" use $(P/c)^{1/2}$ x $(P/c)^{1/2}$ x $c^{1/2}$ mesh and c-fold memory

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

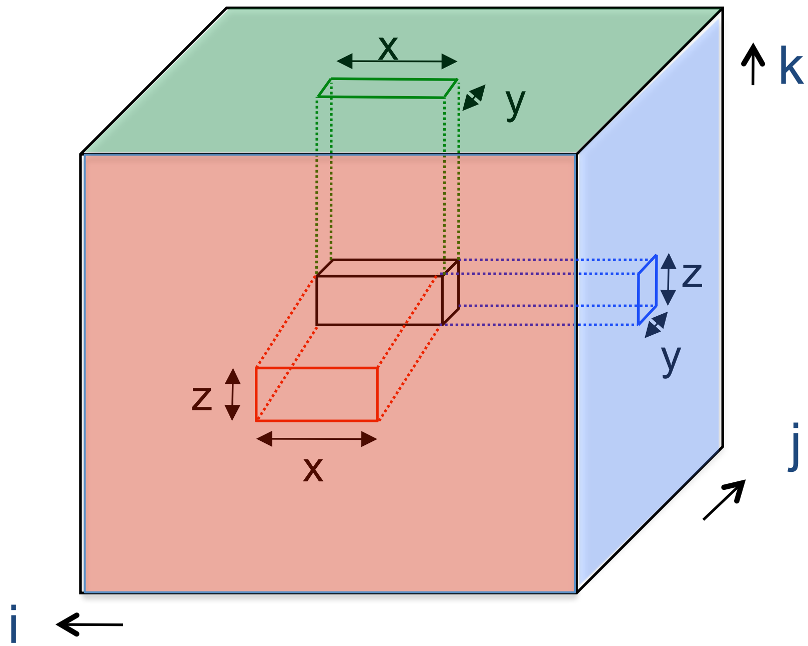
Lesson: Never waste fast memory

Can we generalize for compiler writers?



Deconstructing 2.5D Matrix Multiply

Solomonick & Demmel



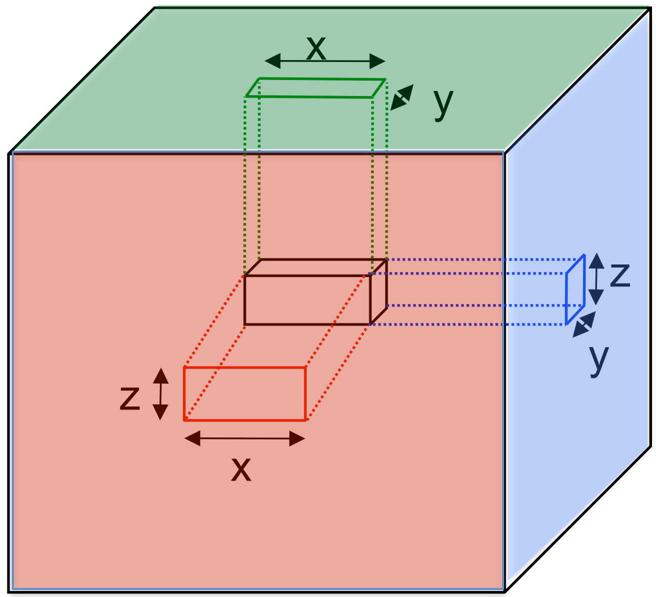
- Tiling the iteration space
- 2D algorithm: never chop k dim
- 2.5 or 3D: Assume + is associative; chop k, which is → replication of C matrix

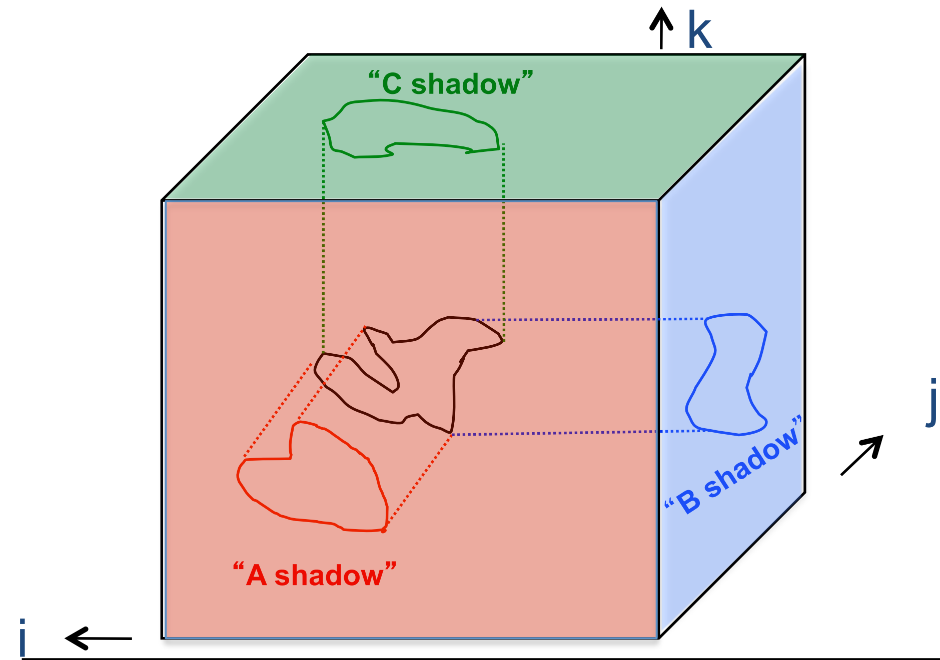
Matrix Multiplication code has a 3D iteration space
Each point in the space is a constant computation (*/+)

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

Lower Bound Idea on C = A*B

Iromy, Toledo, Tiskin





Cubes in black box with side lengths x, y and z
= Volume of black box
= x*y*z
= (#A□s * #B□s * #C□s)$^{1/2}$
= (xz * zy * yx)$^{1/2}$

(i,k) is in "A shadow" if (i,j,k) in 3D set
(j,k) is in "B shadow" if (i,j,k) in 3D set
(i,j) is in "C shadow" if (i,j,k) in 3D set

Thm (Loomis & Whitney, 1949)
cubes in 3D set = Volume of 3D set
≤ (area(A shadow) * area(B shadow) * area(C shadow)) $^{1/2}$

Generalizing Communication Lower Bounds and Optimal Algorithms

- **For serial matmul, we know #words_moved = Ω ($n^3/M^{1/2}$), attained by tile sizes $M^{1/2}$ x $M^{1/2}$**

- **Thm (Christ,Demmel,Knight,Scanlon,Yelick):**
  For any program that "smells like" nested loops, accessing arrays with subscripts that are linear functions of the loop indices

  #words_moved = Ω (#iterations/M^e)

  for some e we can determine

- **Thm (C/D/K/S/Y): Under some assumptions, we can determine the optimal tiles sizes**
  - **E.g., index expressions are just subsets of indices**

- **Long term goal: All compilers should generate communication optimal code from nested loops**

Implications for Arithmetic

x += …

x += …

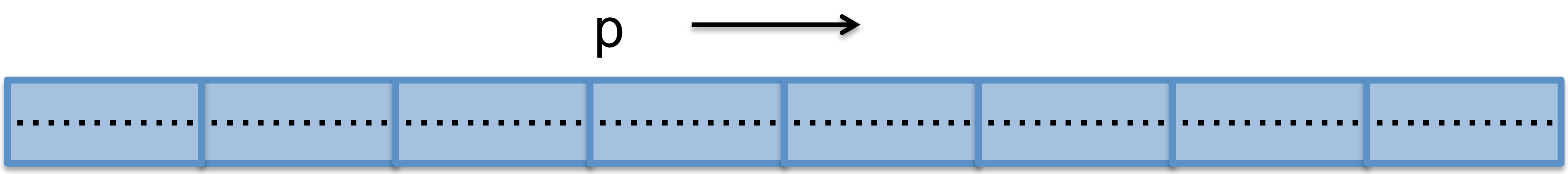
x += …

x += …

Using x for C[i,j] here

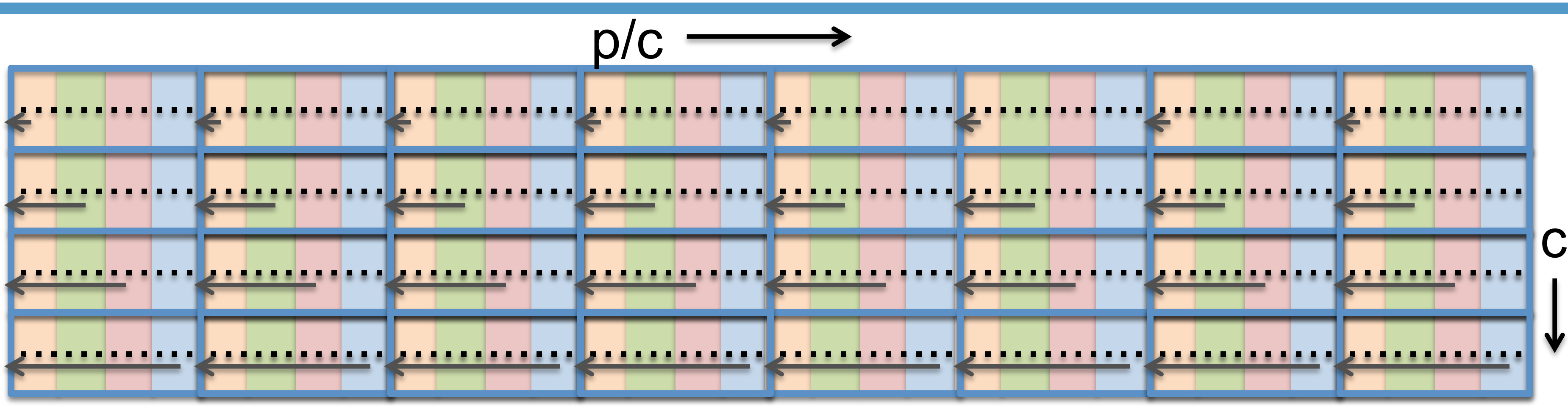
- **Much of the work on compilers is based on owner-computes**
  - **For MM: Divide C into chunks, schedule movement of A/B**
  - **Data-driven domain decomposition partitions data; but we can partition work instead**
- **Ways to compute C "pencil"**
  1. **Serially**
  2. **Parallel reduction** *Standard vectorization trick*
  3. **Parallel asynchronous (atomic) updates**
  4. **Or any hybrid of these**
- **For what types / operators does this work?**
  - **"+" is associative for 1,2 rest of RHS is "simple"**
  - **and commutative for 3**

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

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

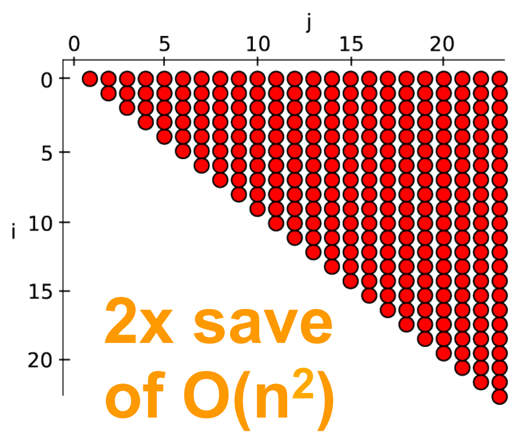


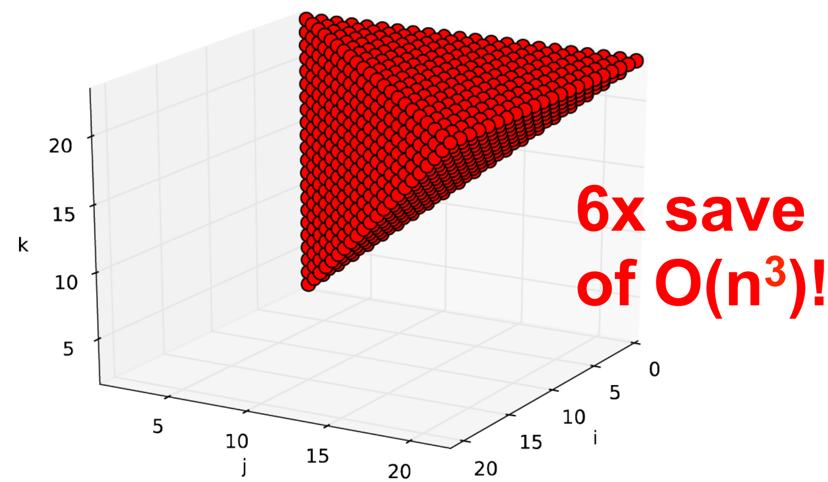
- **Divide p into c groups. Replicate particles within group.**
  - First row responsible for updating all by orange, second all by green,…
- **Algorithm: shift copy of n/(p*c) particles to the left**
  - Combine with previous data before passing further level (log steps)
- **Reduce across c to produce 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: $c \leq p^{1/2}$

Challenge: Symmetry & Load Balance

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



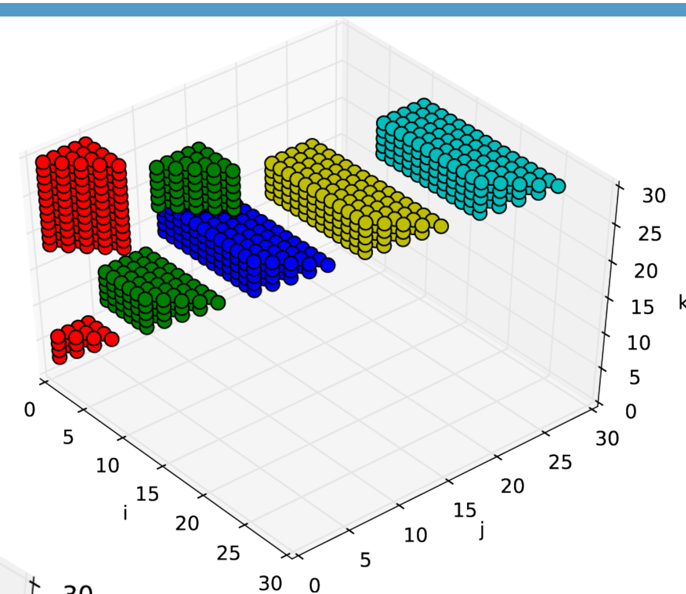


- How to divide work equally?

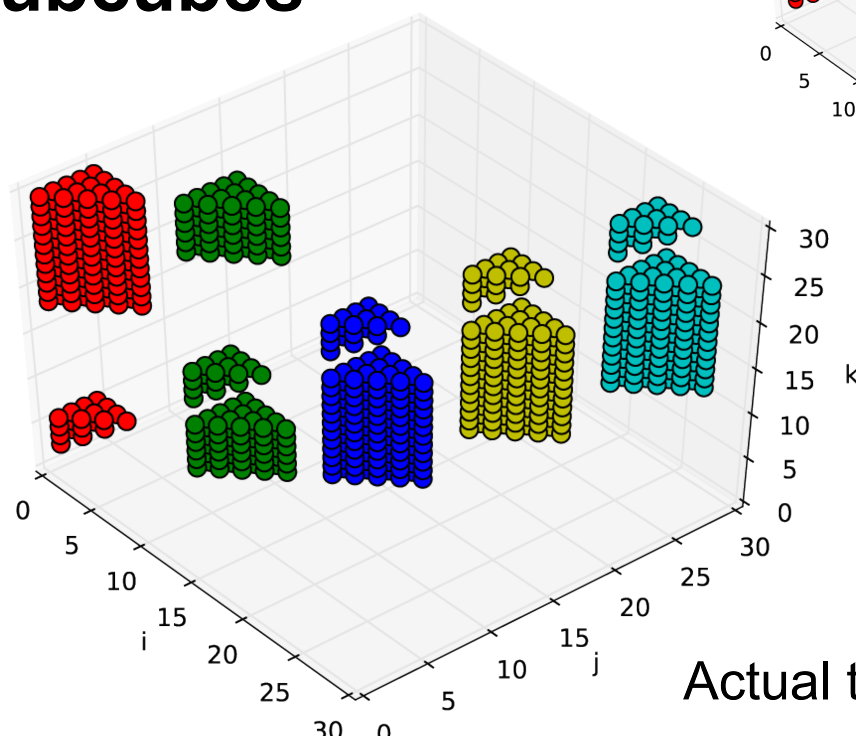
Koanantakool & Yelick

3-Way N-Body Animation

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



Equivalent triplets in the big tetrahedron

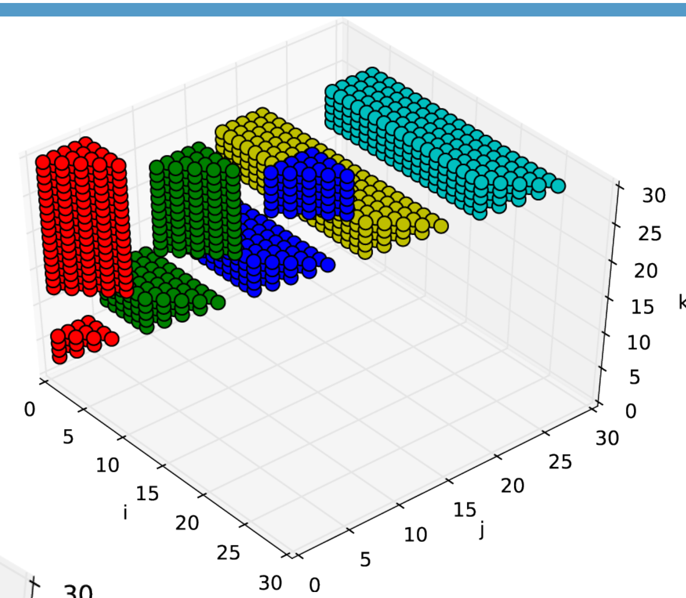


Actual triplets

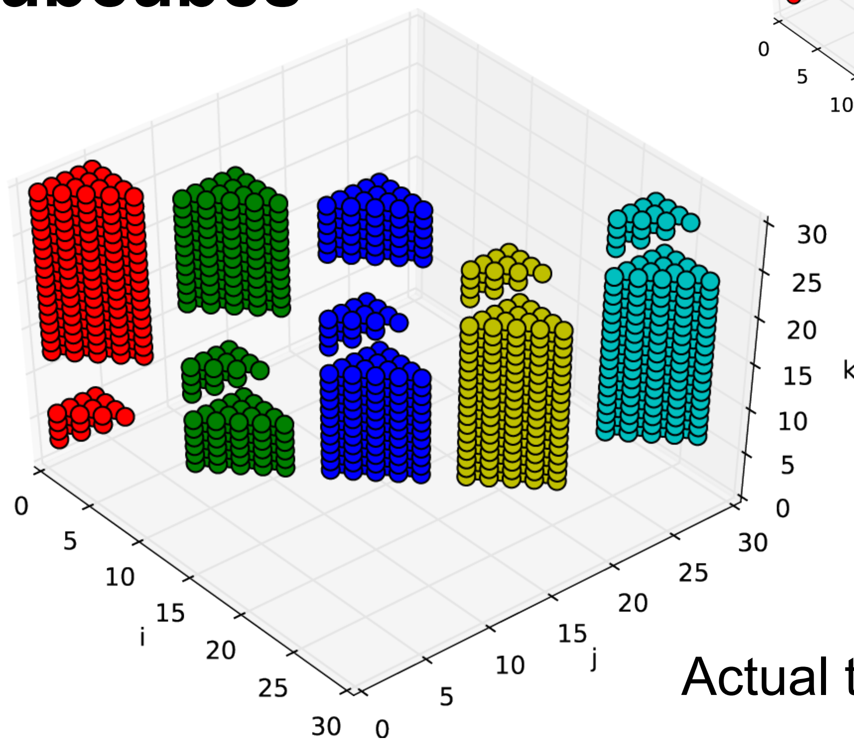
Koanantakool & Yelick

3-Way N-Body Animation

- p=5, n=30
- 6 particles per processor
- 5x5 subcubes



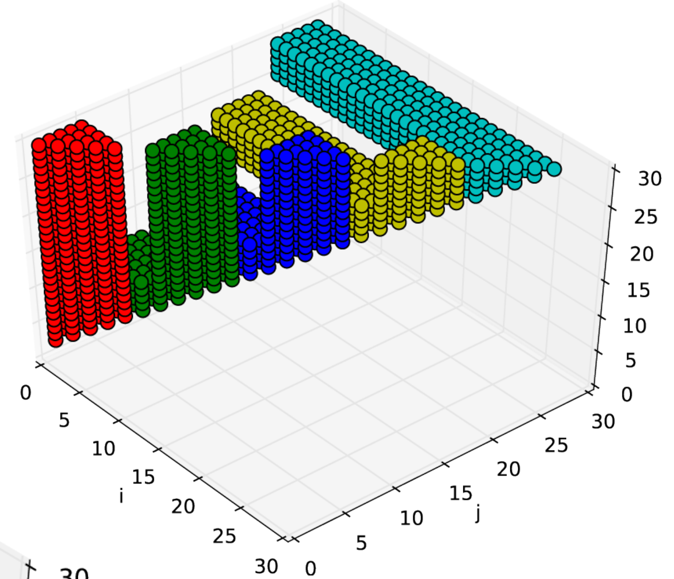
Equivalent triplets in the big tetrahedron



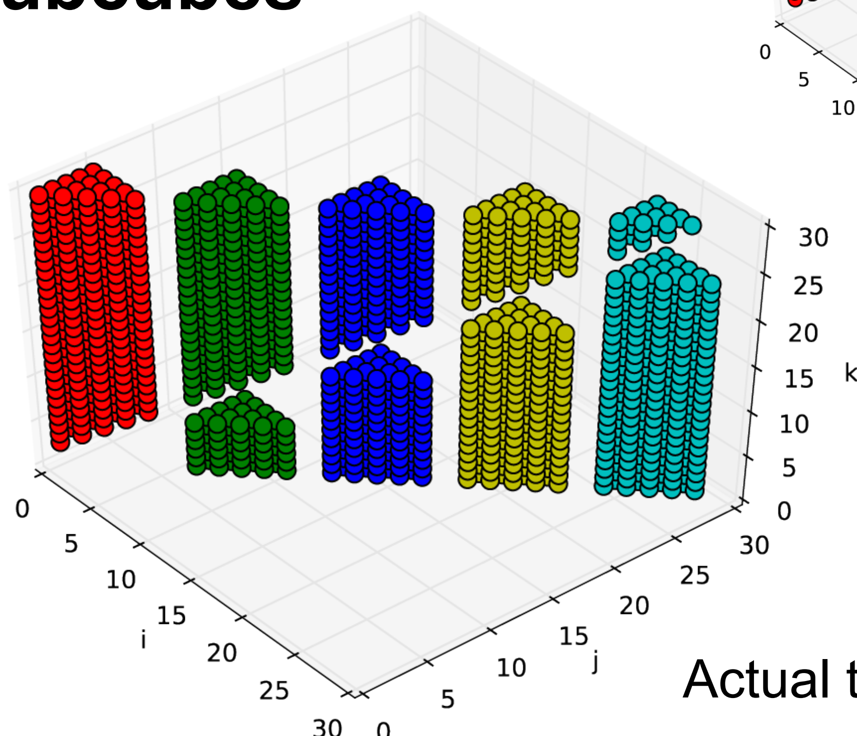
Actual triplets

3-Way N-Body Animation

- **p=5, n=30**
- **6 particles per processor**
- **5x5 subcubes**



Equivalent triplets in the big tetrahedron

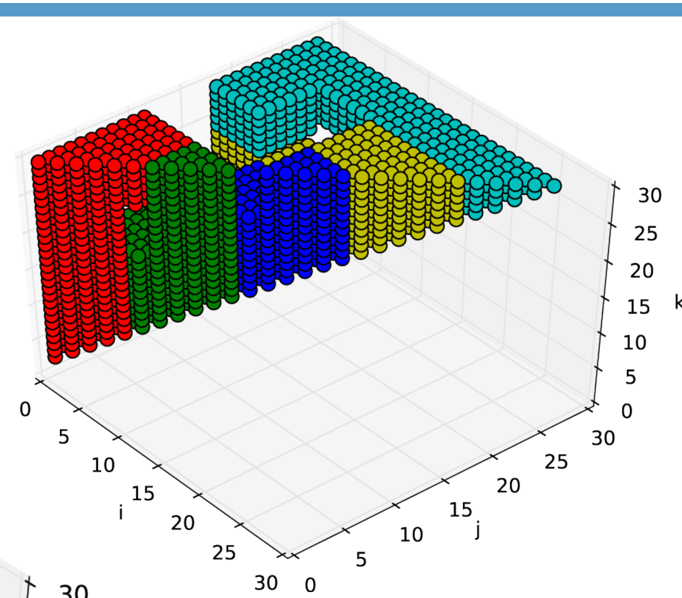


Actual triplets

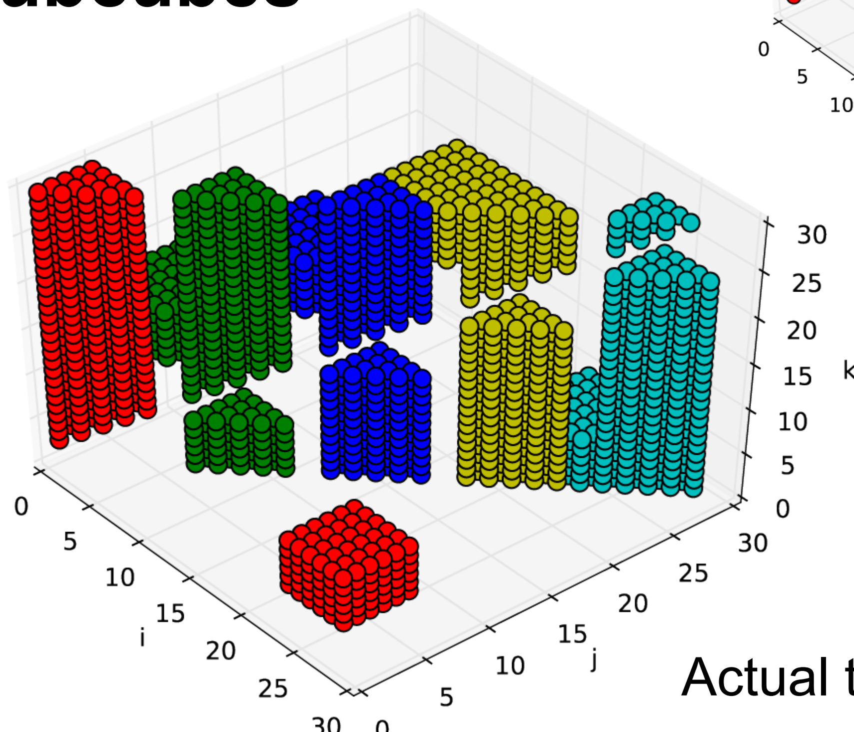
Koanantakool & Yelick

3-Way N-Body Animation

- **p=5, n=30**
- **6 particles per processor**
- **5x5 subcubes**



Equivalent triplets in the big tetrahedron

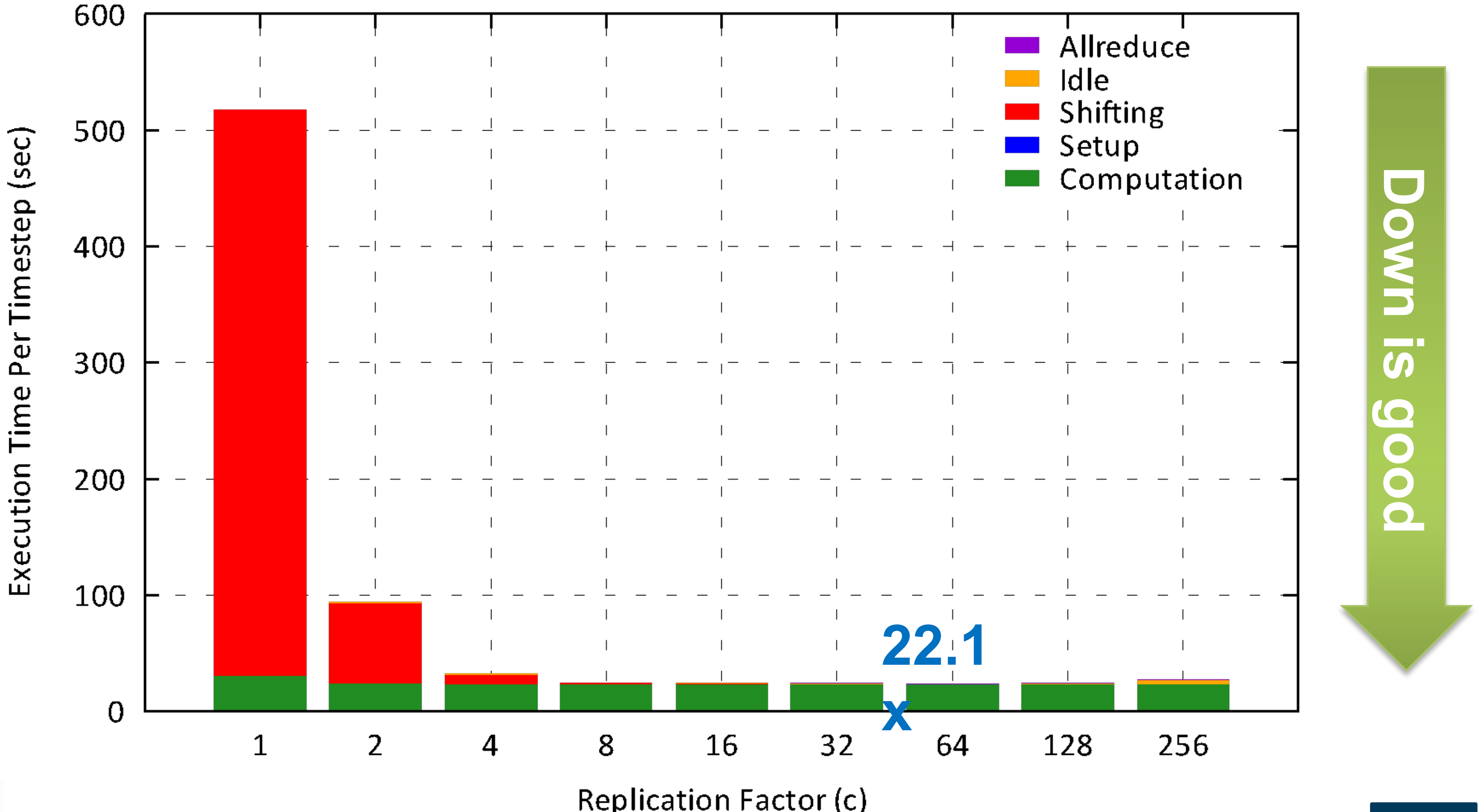


Actual triplets

Koanantakool & Yelick

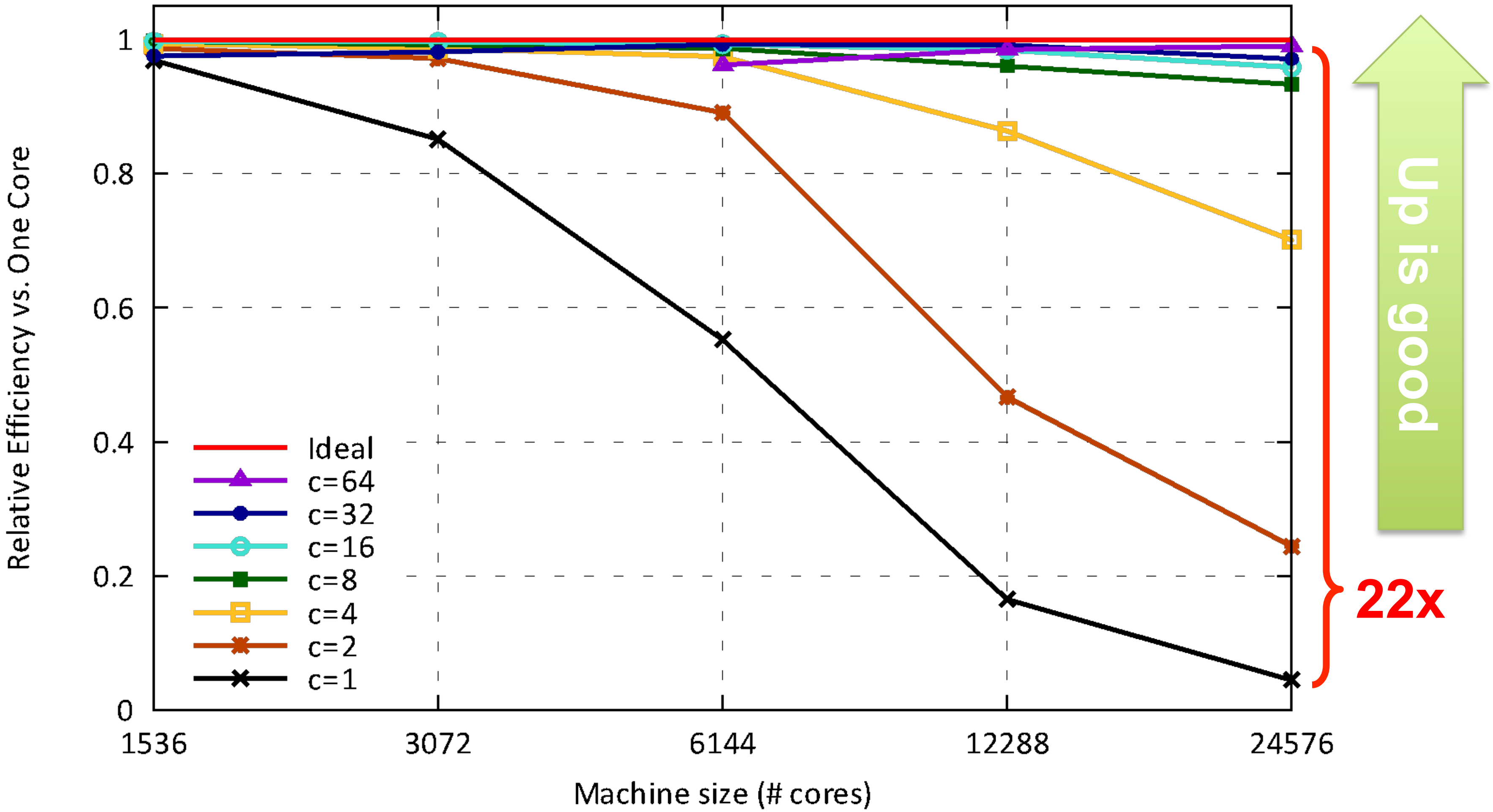
3-Way N-Body Speedup

- Cray XC30, 24k cores, 24k particles



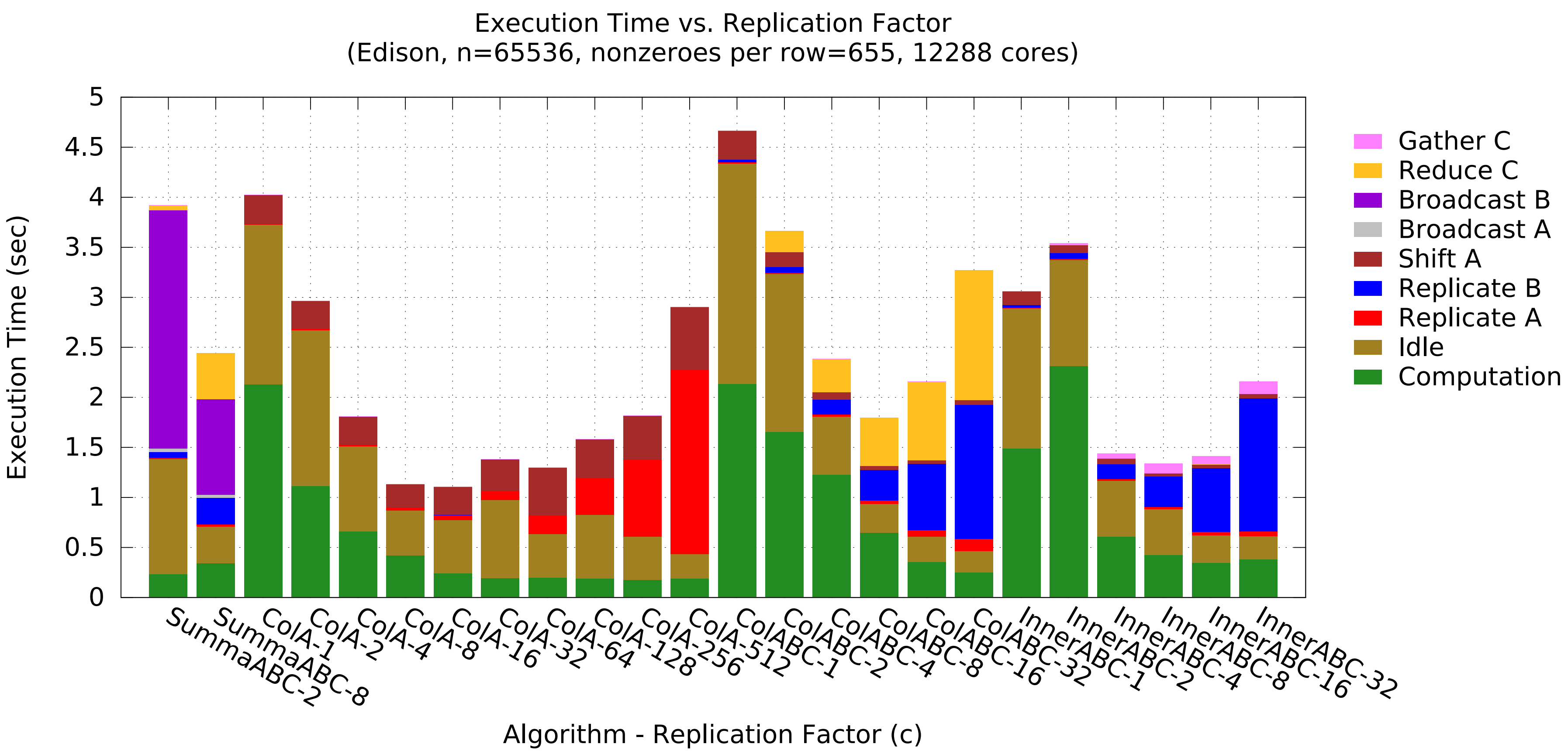
Koanantakool & Yelick

Strong Scaling of .5D Algorithms



Koanantakool & Yelick

Sparse-Dense Matrix Multiply Too!



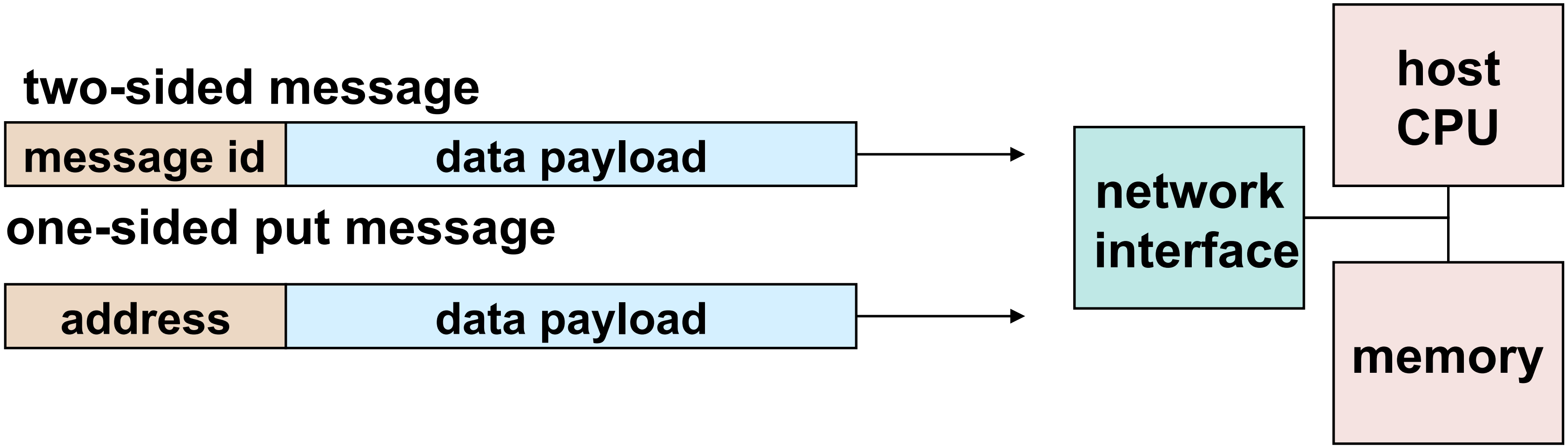
- **Variety of algorithms that divide in or 2 dimensions**

Koanantakool & 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

Avoid Latency and Implicit Synchronization in Communication

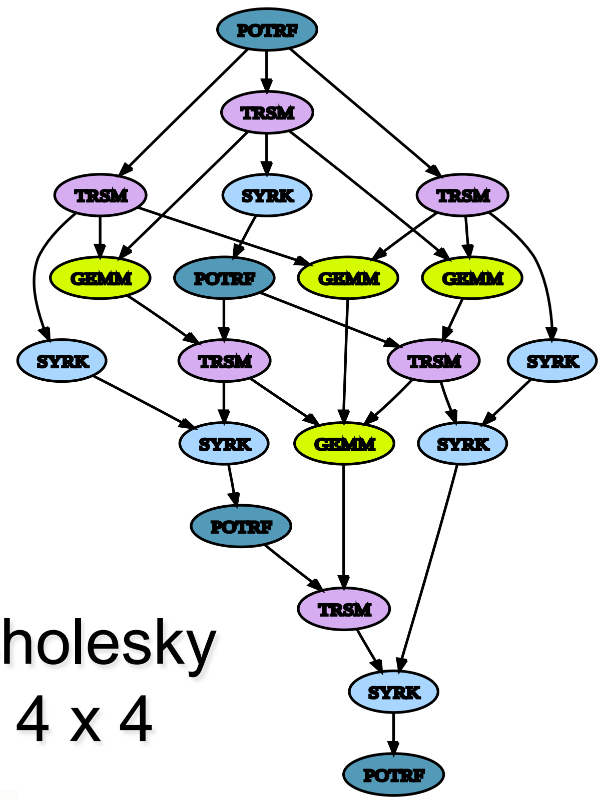


- Two-sided message passing (e.g., send/receive in MPI) requires matching a send with a receive to identify memory address to put data
  - Couples data transfer with synchronization, which is sometimes what you want
- Using global address space decouples synchronization
  - Pay for what you need!

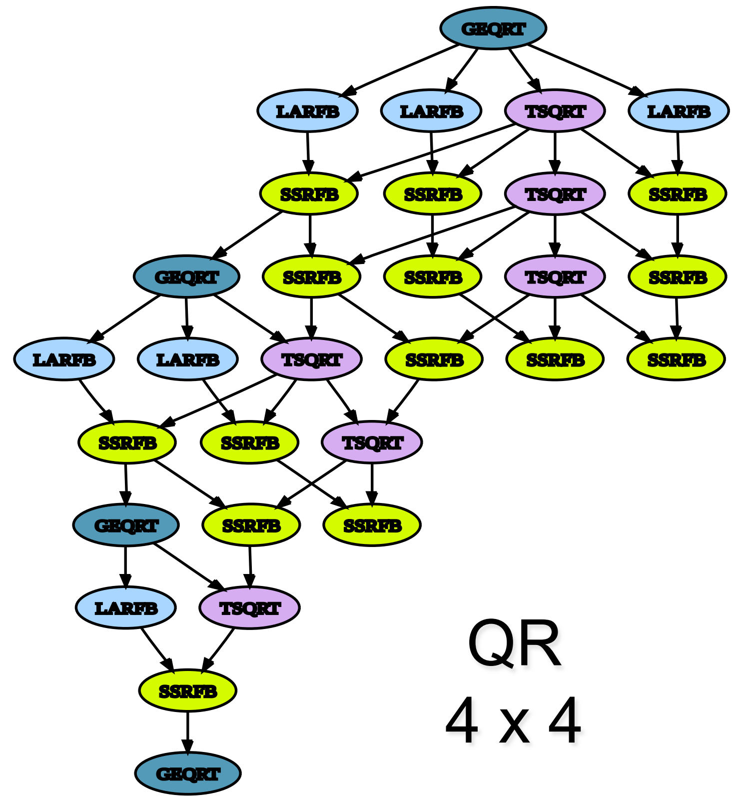
Avoid Synchronization from Applications

Computations as DAGs

View parallel executions as the directed acyclic graph of the computation



Cholesky
4 x 4

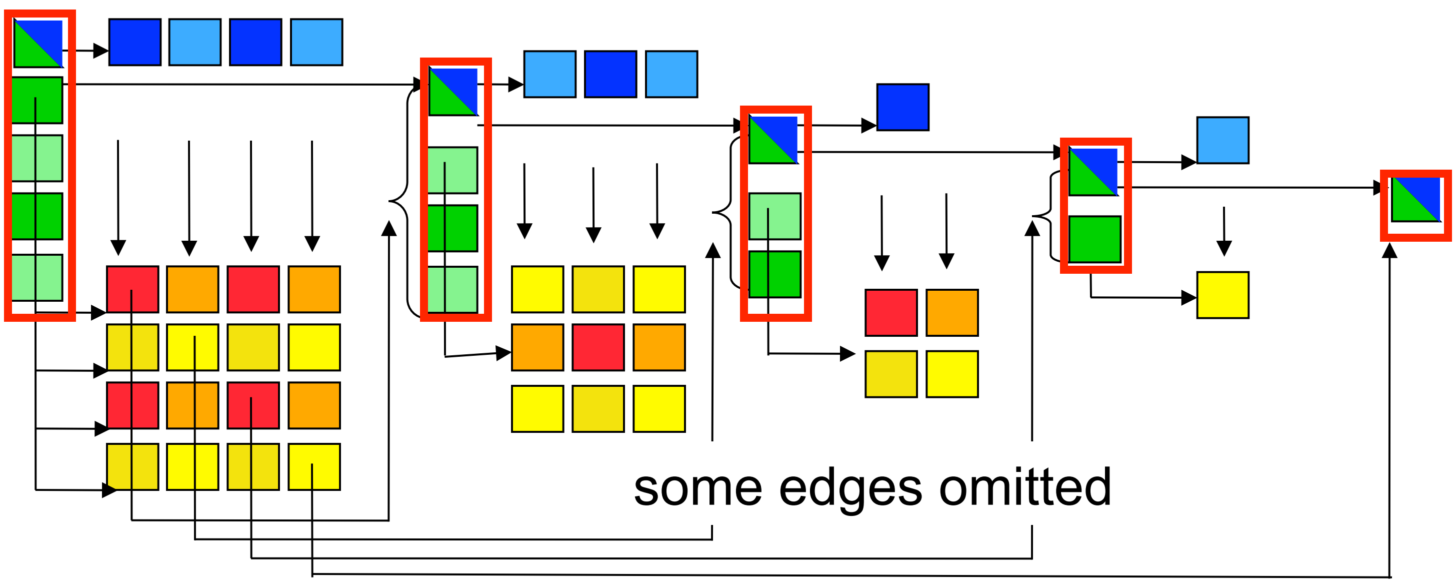


QR
4 x 4

Slide source: Jack Dongarra

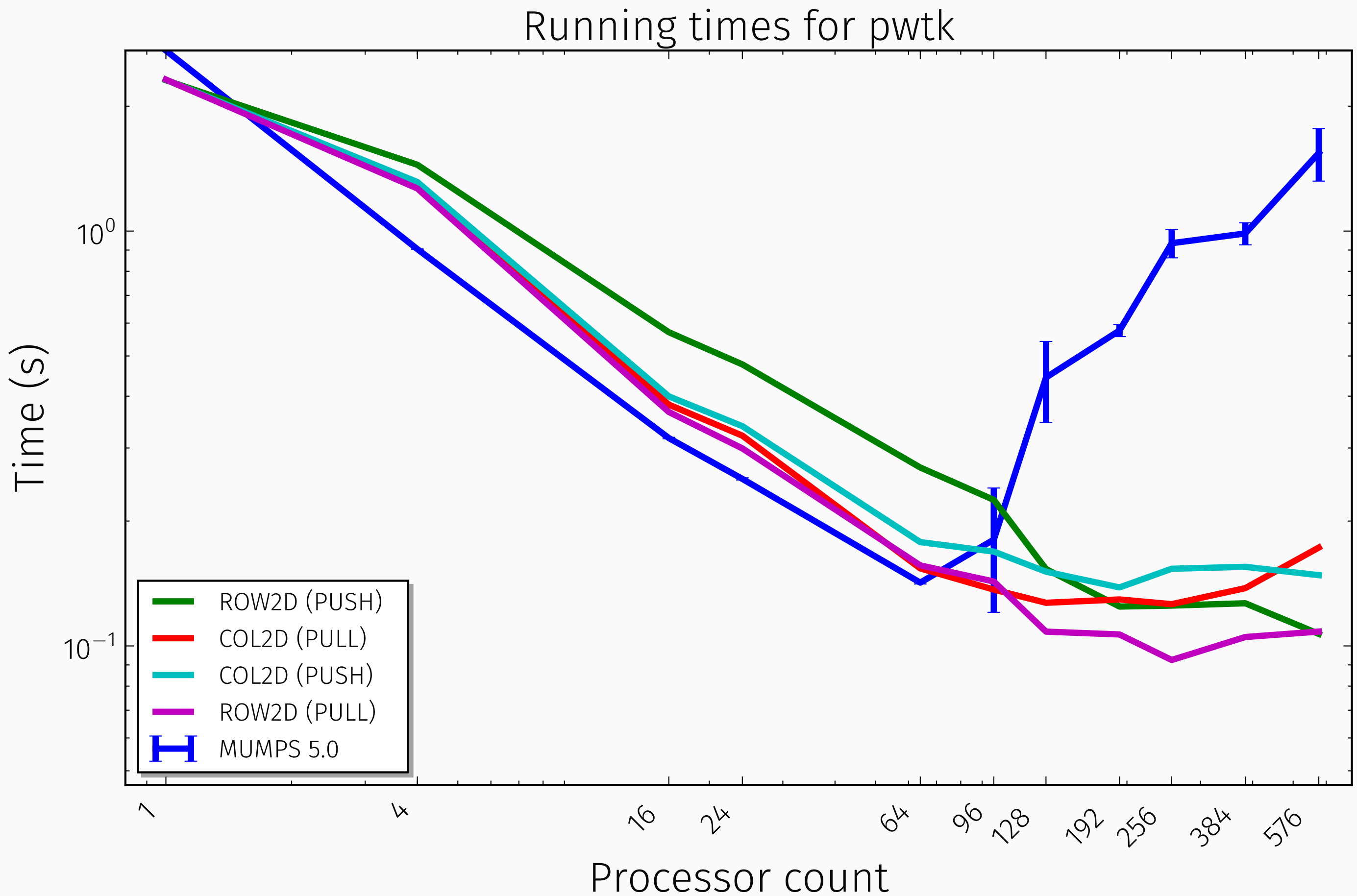
Event Driven LU in UPC

- **Assignment of work is static; schedule is dynamic**
- **Ordering needs to be imposed on the schedule**
  - **Critical path operation: Panel Factorization**
- **General issue: dynamic scheduling in partitioned memory**
  - **Can deadlock in memory allocation**
  - **"memory constrained" lookahead**



Sparse Cholesky

- Timings on NERSC Edison, 24 Intel Ivy-bridge cores per node



- **Fan-both algorithm by Jacquelin & Ng, in UPC++**

OpenMP Loop Parallelism is the Wrong Level

- OpenMP is popular for its convenient loop parallelism
- Loop level parallelism is too coarse and too fine:
  - Too coarse: Implicit synchronization between loops limits parallelism and adds overhead
  - Too fine: Need to create larger chunks of serial work by combining across loops (fusion) to minimize data movement

```
!$OMP PARALLEL DO
   DO I=2,N
     B(I) = (A(I) + A(I-1)) / 2.0
   ENDDO
!$OMP END PARALLEL DO
```

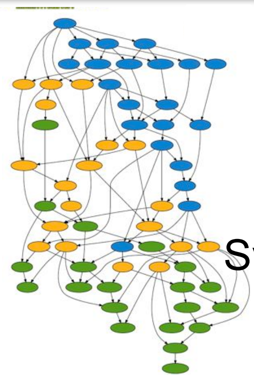
Sources of Unnecessary Synchronization

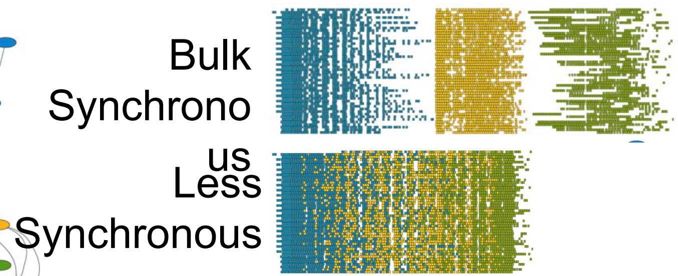
Loop Parallelism

```
!$OMP PARALLEL DO
   DO I=2,N
     B(I) = (A(I) + A(I-1)) / 2.0
   ENDDO
!$OMP END PARALLEL DO
```

“Simple” OpenMP parallelism implicitly synchronized between loops

Abstraction





LAPACK: removing barriers ~2x faster (PLASMA)

Libraries

Analysis	% barriers	Speedup
Auto	42%	**13%**
Guided	63%	**14%**

NWChem: most of barriers are unnecessary (Corvette)

Accelerator Offload

```
!$acc data copyin(cix,ci1,ci2,ci3,ci4,ci5,ci6,ci7,ci8,ci9,ci10,ci11,&
!$acc& ci12,ci13,ci14,r,b,uxyz,cell,rho,grad,index_max,index,&
!$acc& ciy,ciz,wet,np,streaming_sbuf1, &
!$acc&    streaming_sbuf1,streaming_sbuf2,streaming_sbuf4,streaming_sbuf5,&
!$acc&    streaming_sbuf7s,streaming_sbuf8s,streaming_sbuf9n,streaming_sbuf10s,&
!$acc&    streaming_sbuf11n,streaming_sbuf12n,streaming_sbuf13s,streaming_sbuf14n,&
!$acc&    streaming_sbuf7e,streaming_sbuf8w,streaming_sbuf9e,streaming_sbuf10e,&
!$acc&    streaming_sbuf11w,streaming_sbuf12e,streaming_sbuf13w,streaming_sbuf14w, &
!$acc&    streaming_rbuf1,streaming_rbuf2,streaming_rbuf4,streaming_rbuf5,&
!$acc&    streaming_rbuf7n,streaming_rbuf8n,streaming_rbuf9s,streaming_rbuf10n,&
!$acc&    streaming_rbuf11s,streaming_rbuf12s,streaming_rbuf13n,streaming_rbuf14s,&
!$acc&    streaming_rbuf7w,streaming_rbuf8e,streaming_rbuf9w,streaming_rbuf10w,&
!$acc&    streaming_rbuf11e,streaming_rbuf12w,streaming_rbuf13e,streaming_rbuf14e, &
!$acc&    send_e,send_w,send_n,send_s,recv_e,recv_w,recv_n,recv_s)
```

The transfer between host and GPU can be slow and cumbersome, and may (if not careful) get synchronized

Thanks to many Collaborators!

- Jim Demmel
- Dan Bonachea
- Bill Carlson
- Michael Driscoll
- Marquita Ellis
- Evangelos Georganas
- Paul Hargrove
- Costin Iancu
- Khaled Ibrahim
- Shoaib Kamil
- Amir Kamil
- Penporn Kaonantakool
- Nick Knight
- Leonid Oliker
- Eric Roman
- Hongzhang Shan
- Edgar Solomonik
- Erich Strohmaier
- Rich Vuduc
- Sam Williams
- Yili Zheng
- Greg Balls
- Christian Bell
- Soumen Chakrabarti
- Wei Chen
- Kaushik Datta
- Etienne Deprit
- Jason Duell
- Tarek El-Ghazawi
- Ed Givelberg
- Parry Husbands
- Eun-Jin Im
- Jeff Jones
- Brian Kazian
- Arvind Krishnamurthy
- C.J. Lin
- Sabrina Merchant
- Carelton Miyamoto
- Mani Narayanan
- Rajesh Nishtala Steve
- Steinberg Jimmy Su
- Randi Thomas
- Noah Treuhaft
- Chih-Po Wen
- Siu-Man Yau