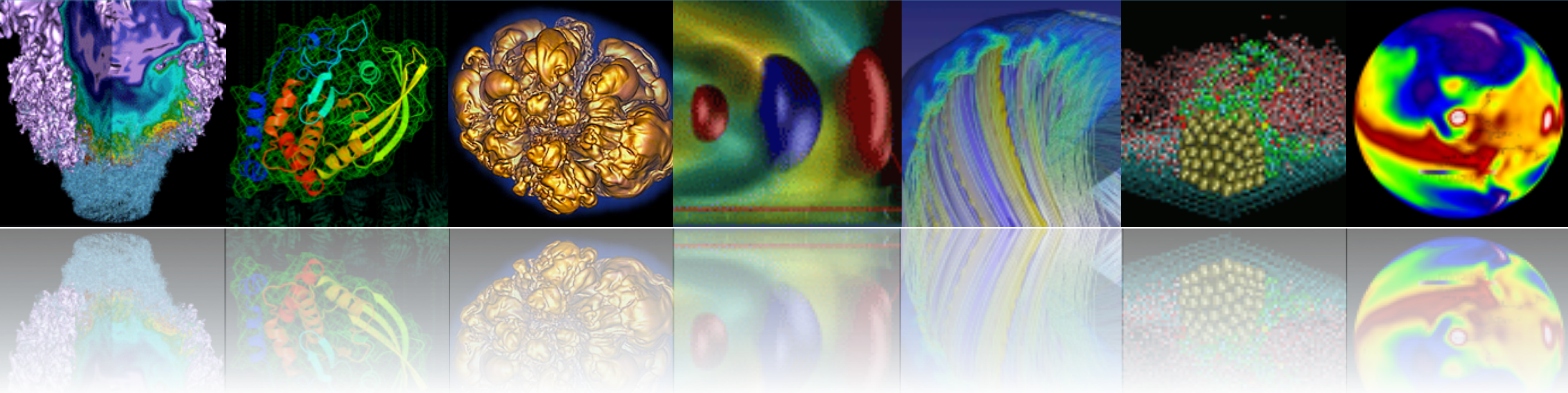


Server Side Applications
(i.e., public/private Clouds and HPC)

Kathy Yelick

UC Berkeley

Lawrence Berkeley National Laboratory

Proposed DOE Exascale Science Problems

Much of science will happen at the boundary between simulation and observation

Superfacility for Science

Global optimization: compute in detectors and on site (lower latency) vs. (cheaper) centralized facilities; network bandwidth

Using HPC in Bioinformatics

HipMer = High Performance Meraculous assembler

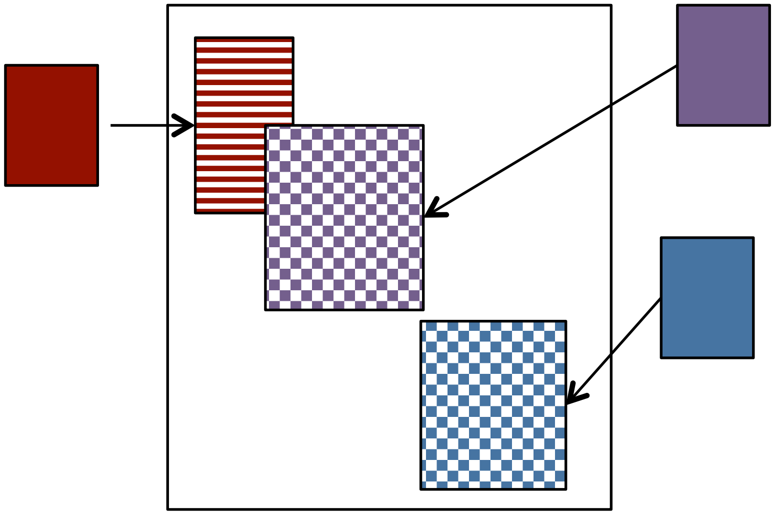
- **Human genome (3Gbp):**
  - SGA assembler: 140 hours
  - Original Meraculous: 48 hours (Perl, some serial bottlnecks)
  - HipMer: 4 minutes (700x speedup)

- **Wheat genome (17 Gbp):**
  - Meraculous (did not run, 170 hours projected):
  - HipMer: 39 minutes; 15K cores (first all-in-one assembly)

- **Wetland metagenome (1.25 Tbp):**
  - Meraculous (projected): 15 TB
  - HipMER: 11 minutes; 20K cores (contig generation)

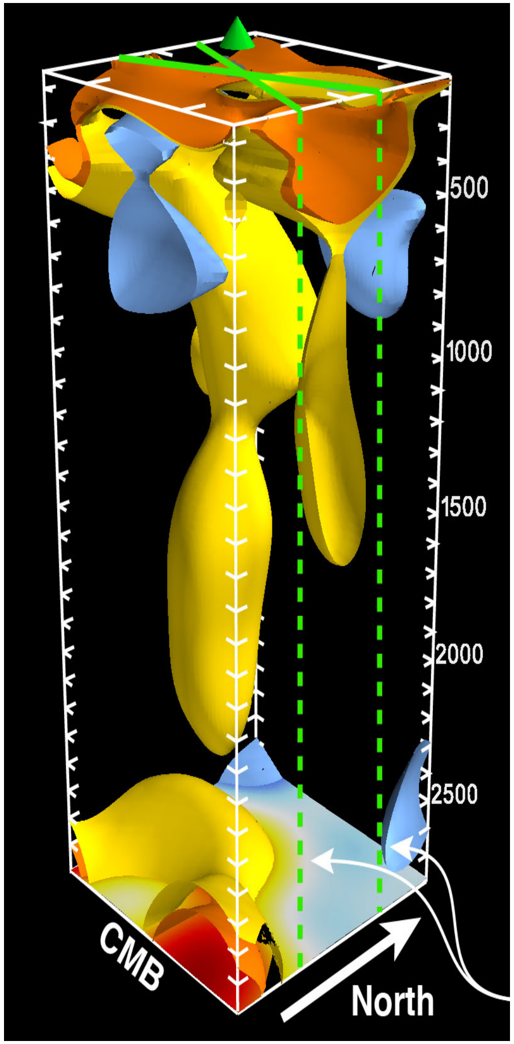
Georganas, Buluc, Chapman, Oliker, Rokhsar, Yelick, Aluru, Egan,Hofmeyr

Data Fusion for Observation with Simulation



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

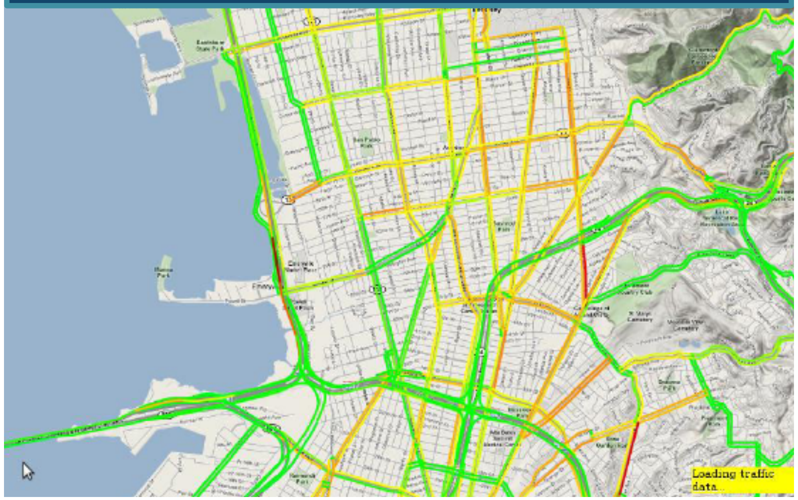
Scott French, Y. Zheng, B. Romanowicz, K. Yelick



Hawaii hotspot geology

Science in embedded sensors: Internet of Things

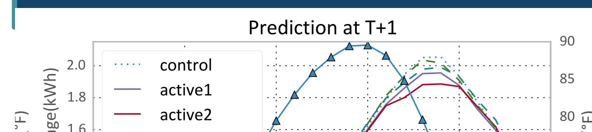
Transportation Modeling



Power Grid Modeling

Scenario Prediction, Planning

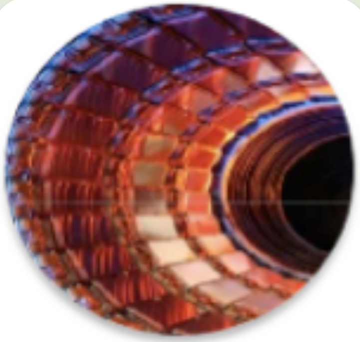
Decision Science



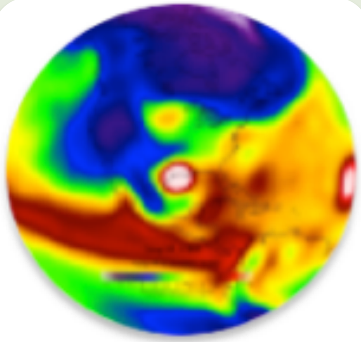
Computing Challenges:

- **Real-time processing, filtering, denoising with model fitting**
- **Machine learning algorithms and scalable implementations**
- **Decision support models; understanding and influencing humans**

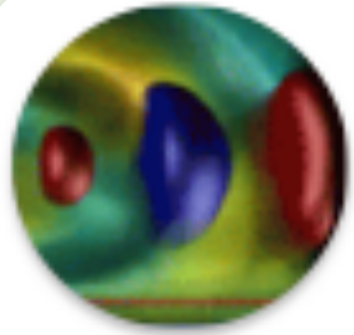
Programming Challenge? Science Problems Fit Across the "Irregularity" Spectrum



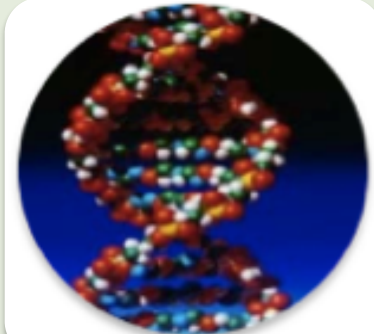
Massive Independent Jobs for Analysis and Simulations



Nearest Neighbor Simulations



All-to-All Simulations



Random access, large data Analysis

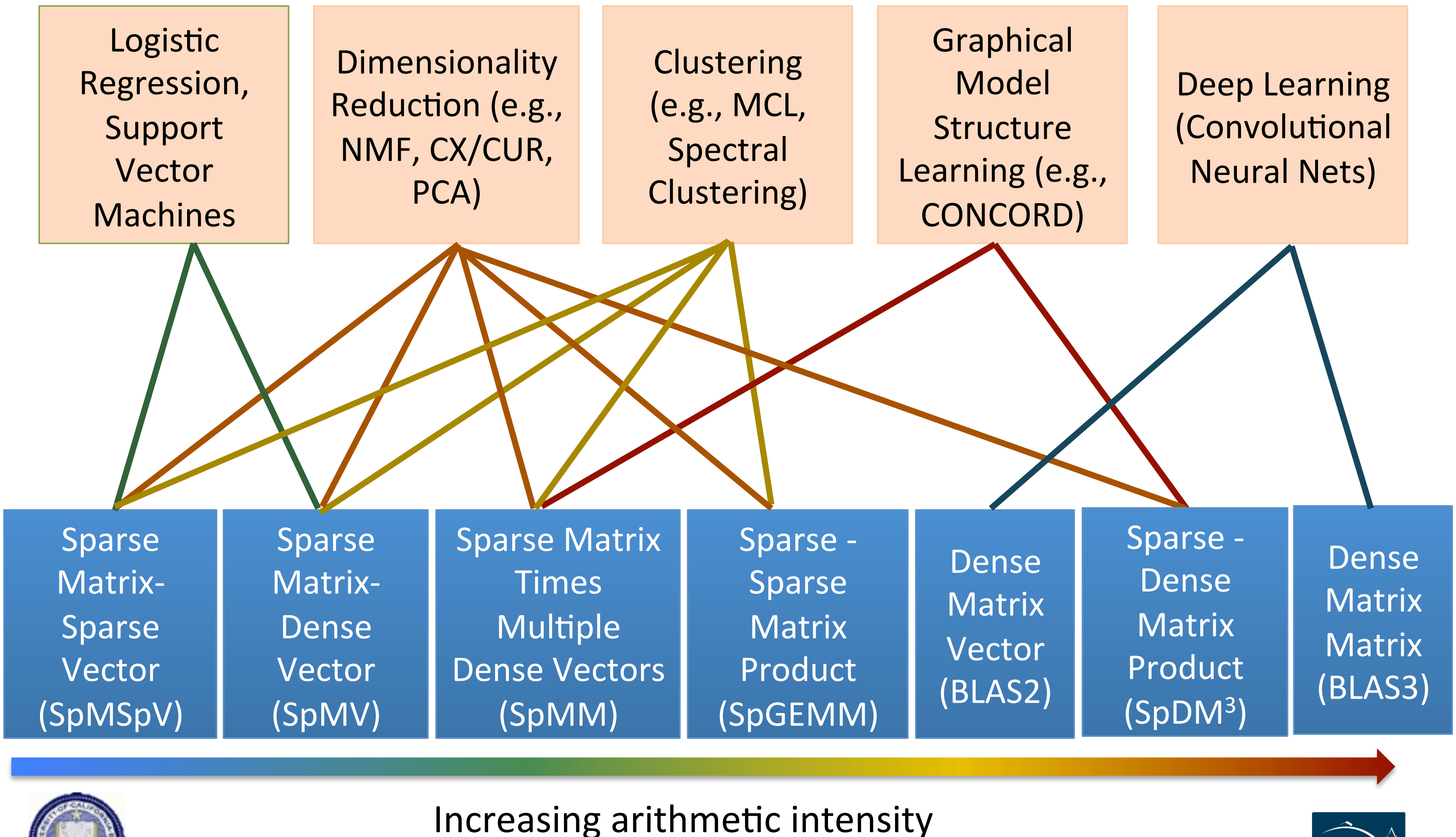
Increasing irregularity (lower computational intensity, and lower spatial and temporal locality)

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) and the sparse matrices are typically less structured.

Machine Learning Mapping to Linear Algebra

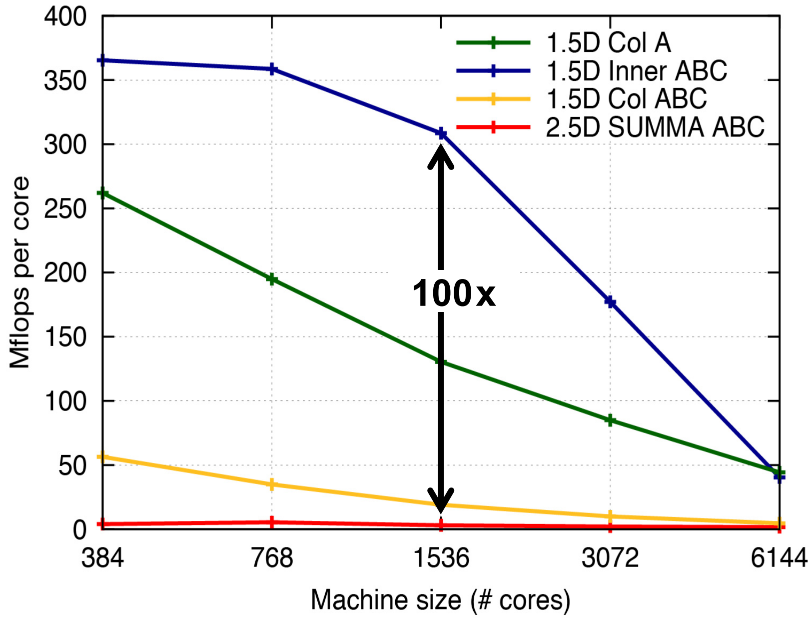


What users want from (nano-inspired) computers

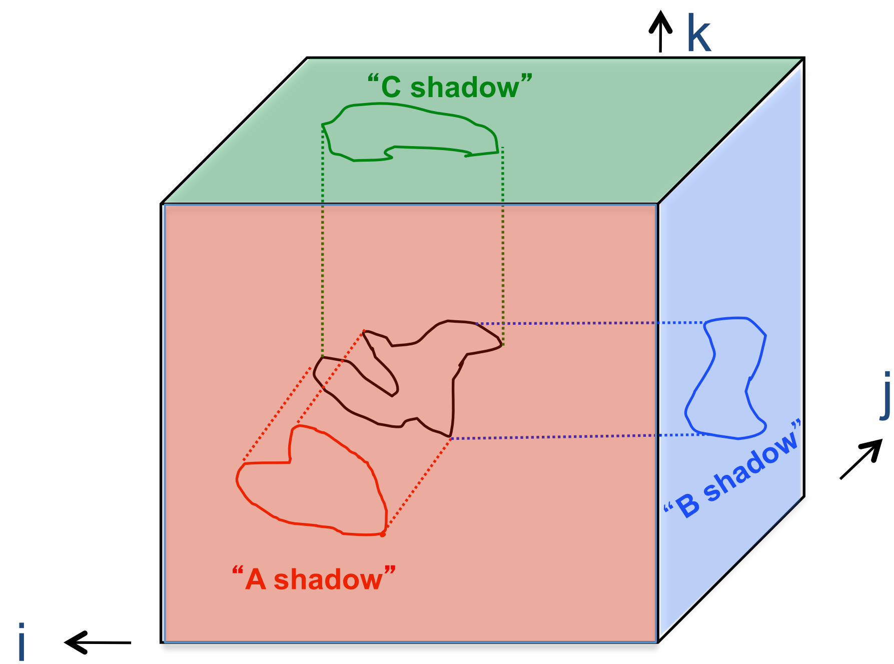
- **An exaop single threaded general processor with 1 exabyte of bandwidth**
- **1000x reduction in cost per "core hour" in the cloud**
  - Make impractical problems practical
  - Cost is best represented by energy (reflect system size, personnel costs, power bill, etc.)
- **Flat memory hierarchy**
  - Increase fast memory capacity 1000x
  - Increase bandwidth to slower memory and lower latency
- **Or a machine that does this at least for key kernels:**
  - Sparse / dense matrix products, FFTs, convolutions, stencils

Communication Avoiding Algorithms

Communication-Avoiding Sparse/Dense Matrix Multiply



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



for i

for j

for k

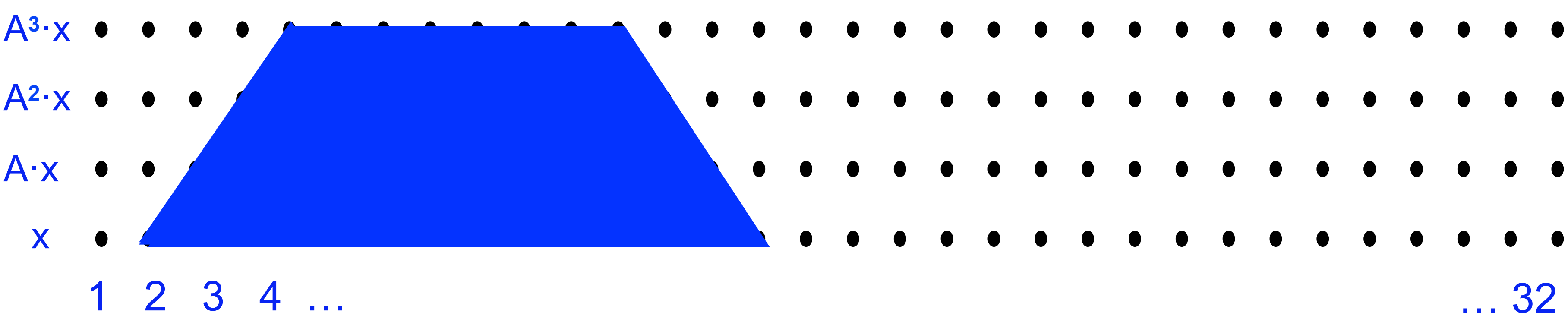
C[i,j] … A[i,k] … B[k,j] …

Penporn Koanantakool et al, 2016

Communication Avoiding Matrix Vector Multiply On a 1D Grid (aka a line)

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

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

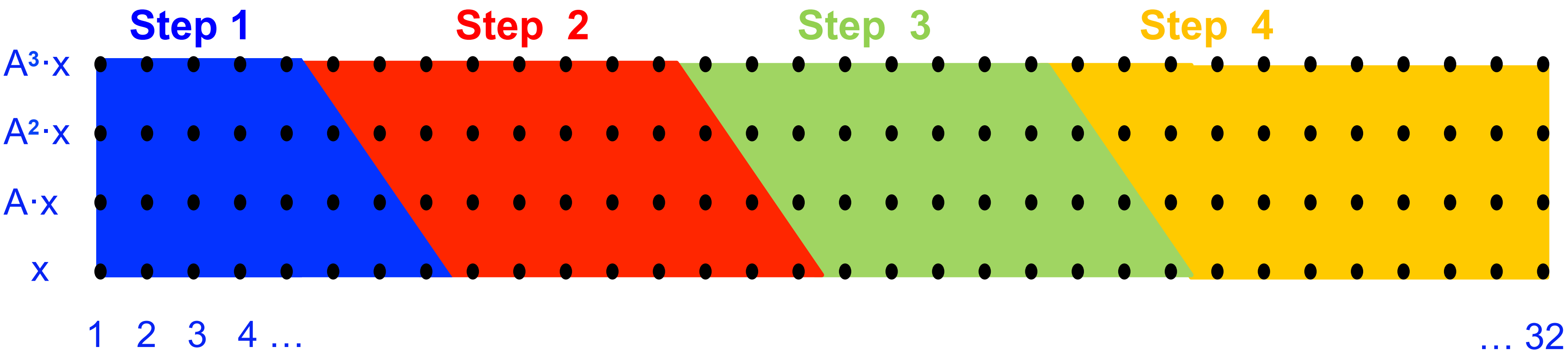


- Idea: pick up part of A and x that fit in fast memory, compute each of k products
- Example: A tridiagonal matrix (a 1D "grid"), n=32, k=3
- General idea works for any "well-partitioned" A

Communication Avoiding Kernels
(Sequential case)

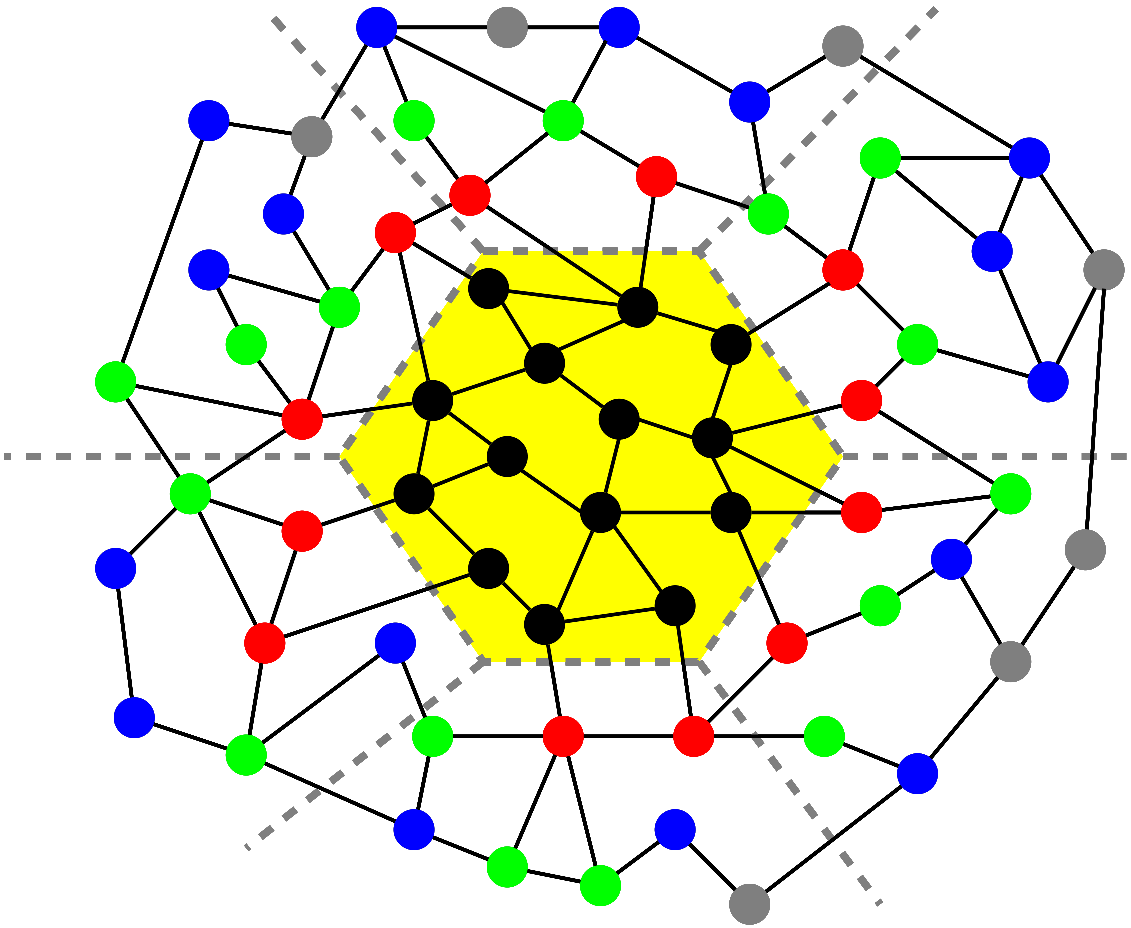
The Matrix Powers Kernel : $[Ax, A^2x, \ldots, A^kx]$

- Replace k iterations of $y = A \cdot x$ with $[Ax, A^2x, \ldots, A^kx]$
- **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



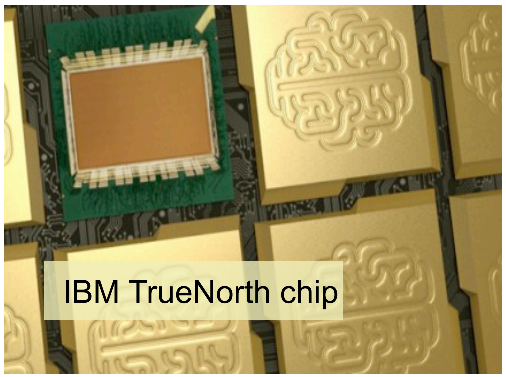
For implicit memory management (caches) uses a TSP algorithm for layout

Joint work with Jim Demmel, Mark Hoemman, Marghoob Mohiyuddin

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

Special Purpose Devices for Science Mission

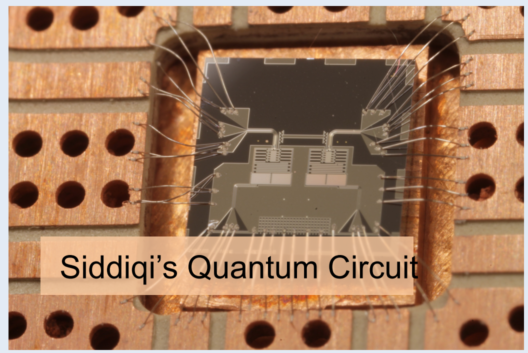
Neuromorphic



IBM TrueNorth chip

Use Convolutional Neural Nets for low-power, real-time data analysis in materials, biology and cosmology

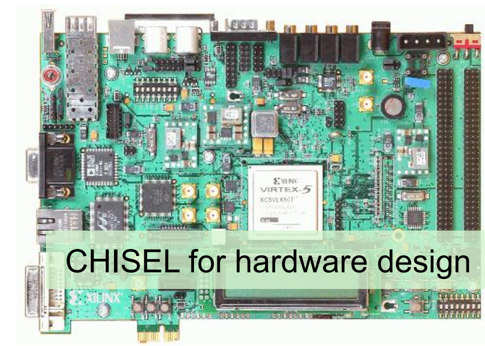
Quantum simulation



Siddiqi's Quantum Circuit

Experimentally implement chemical simulation protocols in existing qubit simulation platform

Custom Processing



CHISEL for hardware design

Even with new devices, gains in performance are from parallelism and specialization

Principles for Algorithms, Systems and Applications (What the users will do)

- **Increase parallelism**
  - Add expression level parallelism in addition to task/data/loop
  - Expose the data to map (statically?) to hardware
- **Avoid communication** (in spite of innovations)
  - Novel algorithms and software
- **Avoid synchronization** (including for new architectures)
  - Overlap, tasking, event-driven execution: for variable clock speeds
  - Reproducibility in spite of asynchrony and approximation
- **Increase specialization and adaptation**
  - Code generation and optimization to use special purpose devices, FPGAs, analog, neural, quantum,…
  - New models of computation and types of algorithms
  - What does computational mean?