Big Data, Big Iron and the Future of HPC

Kathy Yelick
Associate Laboratory Director of Computing Sciences
Lawrence Berkeley National Laboratory

EECS Professor, UC Berkeley

“Big Data” Changes Everything…What about Science?

Transforming Science: Finding Data

Scientific Workflow Today

Experiment

Beamline User

NERSC Policy Board
Cloud Computing and Midrange

March 2, 2010

Scientific Workflow envisioned

Life of a Scientist in 2031

- No personal/departmental computers
- Users don’t login to HPC Facilities
- Travel replaced by telepresence
- Lecturers teach millions of students
- Theorems proven by online communities
- Laboratory work is outsourced
- Experimental facilities are used remotely
- All scientific data is (eventually) open
- Big science and team science democratized

Extreme Data Science

The scientific process is poised to undergo a radical transformation based on the ability to access, analyze, simulate and combine large and complex data sets.
Cloud Computing and Midrange

Goal: To enable new modes of scientific discovery

- Growth in Data
- New Analysis Methods
- New Science Processes

DOE/SC has a particular challenge due to their user facilities and technology trends

New math, stat, CS algorithms are both necessary and enabling

Multi-modal analysis, re-analysis, pose and validate models

Scientific Discovery

Data in Astrophysics: The Challenge is Systematics

Example: Astrophysicists discover early nearby supernova

Filter and Pattern Match with Machine Learning

TECA Toolkit
- Automatic detection of cyclones, atmospheric rivers, and more
- Single data set is 100 TB
- Scalable analysis (80K cores): 9 years → 1 hour

Ongoing work
- Pattern detection using machine learning

Energy Genomics Grand Challenge

- Plant genome: former grand challenge to assemble wheat
- Next: metagenome assembly, currently limited by memory (and time)

- SMP algorithms fail at ~100GB
- Throwing data away to process at all!
- Distributed memory assembly needed
- Essential to understanding microbial dark-mater and their impacts

Filter and Pattern Match with Machine Learning

Mantissa Project, Prabhat

- FPGA refactored code is better
- Still fails at ~850GB

NERSC Policy Board

March 2, 2010
Filtering, De-Noise and Curating Data

- AmeriFlux & FLUXNET: 750 users access carbon sensor data from 960 carbon flux data years
- Arno Penzias and Robert Wilson discover Cosmic Microwave Background in 1965

Re-Use and Re-Analyze Previously Collected Data

- Materials Genome Initiative
  - Materials Project: 4500 users 18 months!
  - “World-Changing Idea of 2013”

Brain Imaging: Multi-Modal Analysis and Data Fusion

- Analyze brain using multiple modalities and scales
- Detection of regions across community
  - 100 individuals takes 18 days right now
- Graph to classify disease
  - Features: biomarkers, image modalities
  - Use hierarchy of regions from Pearson distance

Science Data is Big (and Growing)
"Big Data" Challenges in Science

**Volume, velocity, variety, and veracity**

**Biology**
- Volume: Petabytes now; computation-limited
- Variety: multi-modal analysis on biomages

**Cosmology / Astronomy**
- Volume: 100x increase every 15 years
- Variety: combine data sources for accuracy

**High Energy Physics**
- Volume: 3-5x in 5 years
- Velocity: real-time filtering adapts to intended observation

**Light Sources**
- Velocity: CCDs outpacing Moore’s Law
- Veracity: noisy data for 3D reconstruction

**Materials**
- Variety: multiple models and experimental data
- Veracity: quality and resolution of simulations

**Climate**
- Variety: Combine data sources for accuracy
- Veracity: Reanalysis of 100-year-old sparse data

Data Growth is Outpacing Computing Growth

Graph based on average growth

Projected Data Rates Relative to 2010

Superfacility Concept

Extreme Data Science Facility (XDSF)

Make science easier, more reproducible, and democratic

Transform Experimental Science

Create a “superfacility” that integrates DOE Experimental facilities with computing centers and networking

Data Demos 2014
Cloud Computing and Midrange

A SuperFacility Demo for Light Sources / Photovoltaic Printing

Data collection

- On-the-fly calibration, processing
- Real-time access via web portal

Analysis and modeling on NERSC supercomputers:
- HipGISAXS simulation
- HipRMC fitting

Combining:
- GIXSGUI, dpdaik + …

Printing demo experiments created 36,000 frames in 3 days (1/2 year on TITAN)

Advanced Computing: Not just for Simulation

Experimentation

- Data Analysis
- Simulation

Theory

- Comprehensive Test ban treaty

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

- Experimentation
  - Data Analysis
  - Simulation

- Theory

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

Future Performance from Exascale Technology

Myth: Supercomputers are Expensive, Clouds are Cheap

<table>
<thead>
<tr>
<th>Component</th>
<th>Annual Cost (rough estimate)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud cost on apps (ave 5x slowdown)</td>
<td>~$960M</td>
</tr>
<tr>
<td>Cloud cost (1.388 core hours)</td>
<td>$161M</td>
</tr>
<tr>
<td>NERSC Budget</td>
<td>$57M</td>
</tr>
<tr>
<td>NERSC HPC HW</td>
<td>~$20M</td>
</tr>
</tbody>
</table>

To buy raw NERSC core hours costs more than NERSC budget
- Even ignoring the measured performance slowdown
- Doesn’t include consulting staff, account management, licenses, bandwidth, software support: ~2/3 of NERSC’s Budget

Why?
- NERSC runs at higher utilization ( > 90%) and no profit.
- NERSC cost/core hours dropped 10x (1000%) from 2007 to 2011, while Amazon pricing dropped 15% in the same period
Data Analytics: Case for PGAS

More Regular
- Message Passing Programming
  - Divide up domain in pieces
  - Compute one piece
  - Send/Receive data from others
- MPI, and many libraries

More Irregular
- Global Address Space Programming
  - Each start computing
  - Grab whatever / whenever

UPC, CAF, X10, Chapel, GlobalArrays

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
... often they fit in multiple categories

What about Exascale?

Computational Science has Moved through Difficult Technology Transitions

Application Performance Growth (Gordon Bell Prizes)

Attack of the “killer micros”?
The rest of the computing world gets parallelism

NERSC Policy Board
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 Exaflop machine. It affects every university department cluster and cloud data center.

“Exascale” Challenges Affect Performance Growth at all Scales

1) Power is the primary constraint
2) Parallelism (1000x today)
3) Processor architecture will change
4) Data movement dominates
5) Memory growth will not keep up
6) Programming models will change
7) Algorithms must adapt
8) I/O performance will not keep up
9) Resilience will be critical at this scale
10) Interconnect bisection must scale

Challenge: New Processor Designs are Needed to Save Energy

• Server processors have been designed for performance, not energy
  – Graphics processors are 10-100x more efficient
  – Embedded processors are 100-1000x
  – Need manycore chips with thousands of cores

Node Programming, Heterogeneity

• Case for heterogeneity
  – Many small cores and SIMD for energy efficiency; few CPUs for OS / speed
  – Dark silicon too many transistors to power

• Local store, explicitly managed memory
  – More efficient (get only what you need) and simpler hardware

• Split memory between CPU and “Accelerators”
  – Driven by market history and simplicity, but may not last
  – Communication: The bus is a significant bottleneck.

• Co-Processor interface between CPU and Accelerator
  – Default is on CPU, only run “parallel” code in limited regions
  – Why are the minority CPUs in charge?

Is there a programming model that works for everyone?
Challenge: Memory is Not Keeping Pace

Technology trends against a constant or increasing memory per core

- Memory density is doubling every three years; processor logic is every two
- Storage costs (dollars/Mbyte) are dropping gradually compared to logic costs

![Evolution of memory density](image)

Question: Can you double concurrency without doubling memory?

The Memory Wall Swamp

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

Emerging Exascale Node Architecture

- Fat Core
  - Latency Optimized
  - DRAM/DIMMs
  - High Capacity Low Bandwidth

- Thin Cores
  - (tiny, simple, massively parallel)
  - Throughput Optimized

- Memory Stacks on Package
  - Low Capacity, High Bandwidth

- NVRAM: Burst Buffers / rack-local storage

Node Architecture Problems

- Problems
  - Many slow cores with less memory per core
  - Wide SIMD (wide enough you can’t ignore it)
  - Locality issues (NUMA)

- Possible problems
  - Limited cache coherence?
  - Fat cores (heterogeneity)?
  - Fat cores in charge (co-processor / accelerator)
  - Scratchpad (local store) memory?

Non-problem

- No caching == no problem (trivially coherent)
- PGAS hardware lesson: don’t cache remote values
- MPI or/accelerator PGAS between domains will be fine

Based on slide from J. Shalf

Source: IBM
Node Programming for Homogenous Cores

Requirements:
- Hardware exposes fast local accesses (minimize coherence)
- Low level software to control data layout and work assignment (pin to core)
- Algorithms that minimize data movement and overlap

<table>
<thead>
<tr>
<th>Approach</th>
<th>Argument against</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flat MPI</td>
<td>Need different within/between node algorithms</td>
</tr>
<tr>
<td>MPI + OpenMP</td>
<td>Not enough memory per core</td>
</tr>
<tr>
<td>MPI + PGAS (SPMD, C++)</td>
<td>NUMA effects too strong, compilation too hard</td>
</tr>
<tr>
<td>MPI + TIDA (SPMD, F)</td>
<td>Not tuned and not yet standard</td>
</tr>
<tr>
<td>MPI + Dynamic Tasking</td>
<td>Runtime overheads and poor locality control</td>
</tr>
</tbody>
</table>

Titanium Arrays in UPC++

Amir Kamal (previously Phil Colella, Paul Hilfinger, Alex Aiken, Susan Graham, Kathy Yelick and many others)

- Key features of Titanium arrays
  - Generality: indices may start/end and any point
  - Domain calculus allow for slicing, subarray, transpose and other operations without data copies
- Use domain calculus to identify ghosts and iterate:
  ```
  foreach (p in gridA.shrink(1).domain()) ... 
  ```
- Array copies automatically work on intersection
  ```
  gridB.copy(gridA.shrink(1));
  ```

TiDA: Tiling as a Durable Abstraction

Derek Unut, Cy Chan, Weiquan Zhang, John Bell, John Shalf

- Tiling: Add loop nests so inner ones fit in cache, e.g., 3-loop matmul  6-loop
- TiDA: Add tile shape/size information to each array
- Optionally change the data layout to match
- Can also add ghost regions as needed

```c
int tlen();
int tlo, thi, tlo(2), thi(2), i, j;

darray<double, 3, global> barray = 
  barray[PT(level, id, dir, i, j, k)];
    barray.async_copy(aArrays[PT(level, id, dir, i, j, k)]);
```
Memory Technology (Sandia, Micron, Columbia LBNL)
Understand the Potential of Intelligent, Stacked DRAM Technology

• Data movement are projected to account for over 75% of power on an exascale platform
• Work to reduce that via
  – Optical interconnect(s)
  – 3D stacking (logic + memory + optics)
  – New memory protocols

• Research Questions
  – What is the performance of stacked memory (power & speed)
  – How much intelligence to put into logic layer
    • Atomics, gather/scatter, checksums, full-processor-in-memory
  – What is the memory consistency model
  – How to program it?

Co-Design architectures for Science

Keeping in mind market pressures

Co-Design in the Green Flash Project

• Demonstrated during SC ’09
• CSU atmospheric model ported to low-power core design
  – Dual Core Tensilica processors running atmospheric model at 25MHz
  – MPI Routines ported to custom Tensilica Interconnect
• Memory and processor Stats available for performance analysis
• Emulation performance advantage
  – 250x Speedup over merely function software simulator
• Actual code running - not representative benchmark

John Shalf, Dave Donofrio, Lenny Oliker, Michael Wohner, Marghoob Mohiyuddin, Shaiba Kamil

Enabling Manycore Architecture Research

• ISIS: rapid, accurate FPGA emulation of manycore chips
• Spans VLSI design and simulation and includes chip fab
  – Trains students in real design trade-offs, power and area costs
• Mapping RTL to FPGAs for algorithm/software co-design
  – 100x faster than software simulators and more accurate

Pls: John Wawrzynek and Krste Asanovic, UC Berkeley

NERSC Policy Board

Cloud Computing and Midrange
March 2, 2010
Cloud Computing and Midrange

Co-Design Analysis for Combustion

Estimated Performance Improvements

- Fast NIC (400 GB/s)
- Fast-exp
- Fast-div
- Fast memory (4 TB/s)
- Loop fusion
- Cache blocking
- Baseline

Number of Species

Hardware and software need to change together

Autotuning: Write Code Generators

- Autotuners are code generators plus search
- Avoids two unsolved compiler problems: dependence analysis and accurate performance models
- Popular in libraries: Atlas, FFTW, OSKI...

Approaches to Autotuning

Approaches to Autotuning

Let computers, not humans, tune for modern architectures code

But beware of trusting compilers

Approaches to Autotuning

How do we produce all of these (correct) versions?
- Using scripts (Python, perl, C,..)
- Transform high level representation (FFTW, Spiral)
- Compiling a domain-specific language (D-TEC)
- Compiling a general-purpose language (X-Tune)
- Dynamic compilation of a domain-specific (SEJITS)
Target Higher Level Loops

Harder than inner loops....

Iterative Solves are Dominated by Sparse Matrix-Vector Multiply (nearest neighbor on graph)

- Can do better: 1 matrix read, multiple multiplies
- 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)

Avoid Synchronization

The end of bulk-synchronous programming?
Cloud Computing and Midrange

Reasons to avoid synchronization

- Processors do not run at the same speed
  - Never did, due to caches
  - Power/temperature management makes this worse

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

One-sided communication is a mechanism that works everywhere

- Fast one-sided network communication (RDMA, Remote DMA)
- Move data to/from accelerators
- Move data to/from I/O system (Flash, disks,..)
- Movement of data in/out of local-store (scratchpad) memory

PGAS is a programming model

```c
*p1 = *p2 + 1;
A[i] = B[i];
upc_memput(A, B, 64);
```

Uses 1-sided communication: put/get

UPC LU factorization code adds cooperative (non-preemptive) threads for latency hiding

- New problem in partitioned memory: allocator deadlock
- Can run on of memory locally due to unlucky execution order

DAG Scheduling Outperforms Bulk-Synchronous Style

PLASMA on shared memory

UPC on partitioned memory

UPC vs. ScaLAPACK

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
Cloud Computing and Midrange

Resilience

Is the sky really falling?

Resilience Approaches

- Containment Domains (CDs) for trees
  - Flexible resilience techniques (mechanism not policy)
  - Each CD provides own recovery mechanism
  - Analytical model: 90%+ efficiency at 2 EF vs. 0% for conventional checkpointing
- Berkeley Lab Checkpoint Restart
  - BLCR is a system-level Checkpoint/Restart
    - Job state written to filesystem or memory; works on most HPC apps
    - Checkpoint/Restart can be used for rollback recovery
    - a course-grained approach to resilience
    - BLCR also enables use for job migration among compute nodes
  - Requires support from the MPI implementation

CD PIs: Mattan Erez (+Eric Roman for PGAS); GVR PI: Andrew Chien

What is Wrong with Current Operating Systems?

Tessellation: Joint UCB/LBNL to rethink Manycore OSs
Assumes limited number of CPUs that must be shared
- Old CW: time-multiplexing
- Tessellation: spatial partitioning

Greedy allocation of finite I/O device interfaces
- Old CW: First process to acquire lock gets device
- Tessellation: QoS management for symmetric device access

Fault Isolation
- Old CW: CPU failure → Kernel Panic (increasingly frequent)
- Tessellation: CPU failure → Partition Restart (w/ drivers)

Inter-Processor Communication
- Old CW: invoked for ANY interprocessor communication
- Tessellation: direct HW access mediated by hypervisor

Impact:
- Convex optimization major thrust for Microsoft Research
  - Launching into new OS/R CFP with Sandia National Lab

NERSC Policy Board

March 2, 2010
Cloud Computing and Midrange

What does this mean for NERSC?

Essentially, all models are wrong, but some are useful.

-- George E. Box, Statistician

NERSC Workload

- > 5000 users
- > 700 projects
- > 2000 publications per year
- 2 Petascale systems today
  - NERSC-7: Hopper
  - NERSC-8: Edison
- Moving data workload

The workload is diverse and increasingly complex due to science workflows, integration of data, and demand for higher resolution and scale

Keeping up with user needs will be a challenge

NERSC Policy Board

March 2, 2010
**Sky Computing and Midrange**

**NERSC Policy Board**

**March 2, 2010**

---

**Edison, a Cray XC-30 plays a key role in NERSC’s strategy**

- NERSC assessed that our broad workload was not ready for GPUs and procured Edison, with Ivy Bridge Intel CPUs
- Workloads that have difficulty moving to NERSC-8 can still work productively on Edison while the code is adapted
- In 2016 Edison will likely provide ~20% of NERSC’s cycles

---

**The Cori System Scheduled for 2016**

- Cori will support the broad Office of Science research community transition to energy efficient architectures
  - Cray XC system with > 9300 Intel KNL nodes
    - Self-hosted (not an accelerator) manycore processor with over 60 cores per node
    - 32 Flops / cycle (AVX 512 SIMD)
    - On-package high-bandwidth memory (scratchpad)
    - Scheduled for 2016 installation
    - NVRAM Burst Buffer for data intensive applications
    - 28 PB of disk, 432 GB/sec I/O bandwidth
    - Scheduling for complex workflows

---

**NERSC System Roadmap**

- NERSC-7 Edison in production
- NERSC-8 Cori Data Partition (Haswell) installed
- NERSC-9 pre-exascale system installed
- NERSC Exascale 2024

- NERSC Exascale Strategy is designed to balance the needs of current science with future science

---

**NESAP Codes**

**Basic Energy Sciences**
- Kent (ORNL)
- Quantum

**Biological and Environmental Research**
- Smith (ORNL)
- Gromacs

**Nuclear Physics**
- Mads (Iowa St.)

**Fusion Energy Sciences**
- Jardin (PPPL)

**High Energy Physics**
- Vay (BNL)

**Low Energy Physics**
- MDFn

**Fundamental Interactions**
- Chen (UCSD)

**Quantum Matter**
- MLI

---

**Image source: Wikipedia**

System named after Gerty Cori, Biochemist and first American woman to receive the Nobel prize in science.
Numerical Methods at NERSC

- Quantitative (but not so deep) measure of algorithms classes
- Based on hours allocated to a project that the PI claims uses the method

Algorithm Diversity

<table>
<thead>
<tr>
<th>Science areas</th>
<th>Dense linear algebra</th>
<th>Sparse linear algebra</th>
<th>Spectral Methods (FFT)</th>
<th>Particle Methods</th>
<th>Structured Grids</th>
<th>Unstructured or AMR Grids</th>
</tr>
</thead>
<tbody>
<tr>
<td>Accelerator Science</td>
<td>X</td>
<td>X</td>
<td>IMPACT-T</td>
<td>IMPACT-T</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Astrophysics</td>
<td>X</td>
<td>X</td>
<td>MAESTRO</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>Chemistry</td>
<td>X</td>
<td>X</td>
<td>GAMESS</td>
<td>X</td>
<td>MAESTRO</td>
<td>X</td>
</tr>
<tr>
<td>Climate</td>
<td>X</td>
<td>X</td>
<td>IMPACT-T</td>
<td>X</td>
<td>MAESTRO</td>
<td>X</td>
</tr>
<tr>
<td>Fusion</td>
<td>X</td>
<td>X</td>
<td>GTC</td>
<td>X</td>
<td>GTC</td>
<td>X</td>
</tr>
<tr>
<td>Lattice Gauge</td>
<td>X</td>
<td>X</td>
<td>MILC</td>
<td>X</td>
<td>MILC</td>
<td>X</td>
</tr>
<tr>
<td>Material Science</td>
<td>X</td>
<td>X</td>
<td>PARATEC</td>
<td>X</td>
<td>PARATEC</td>
<td>X</td>
</tr>
</tbody>
</table>

Previous Procurement Strategy: Publish Representative Benchmarks

Co-design for Data: Finding Middle Ground

- Mount BB as a disk: /fast – then user has to do all the work/juggling
- Have software that automatically determines best way to use BB - $'s

NERSC Qualitative In-Depth Analysis of Methods by Science Area
Computational Research and Theory (CRT):
A Building Designed for Exascale Systems