A (Hypothetical) Data to Discovery Engine

Mark Stalzer
Center for Advanced Computing Research
California Institute of Technology
stalzer@caltech.edu
www.cacr.caltech.edu

AstroInformatics, June 16, 2010
Extended IST Version, November 2, 2010
This is an informal talk about making data to discovery happen faster using a few truths about computing...

...and things are happening much faster than expected and much slower than we think...
The mission of the Center for Advanced Computing Research (CACR) is to ensure that Caltech is at the forefront of computational science and engineering. Leading in computer-based modeling and big data analysis for scientific discovery and exploration of engineering designs. CACR provides an environment that cultivates multidisciplinary collaborations.

Center staff have expertise in data-intensive scientific discovery, physics-based simulation, scientific software engineering, visualization techniques, novel computer architectures, and the design and operation of large-scale computing facilities.

Contact Us — Let’s Explore How We Can Work Together!
director@ac.celtech.edu — Workkeley CACR Office

John P. Doe
MC 148-92
Exponential Growth in Data Volumes & Complexity

- Moore’s law works for detectors too: PBs of data to process
- Data is more complex: multi-messenger & discrete data fusion
- Simulations also produce many TB of data
- Data + Theory = Understanding
- What might a Data to Discovery Engine look like?

(L) Data growth trends for optical telescopes

(R) Crab Nebula at visible and X-ray bands
LHC/CMS Data Analysis

Virtual Detector
- Design
- Calibration
- Transport
- Classification

New physics?
Some Truths about Computing

• It must be COTS (cheaply manufacturable IP)
  ‣ Cray-1 was COTS: 6 dual in-out ECL gates per chip
  ‣ Drivers: missile guidance and virtual missile guidance

• Advanced computing systems are all about **power** and packaging
  ‣ Batteries and 100 MW power plants are expensive
  ‣ Apple iPad and Cray-1

• Advanced computers are hard to program
  ‣ But they can be easy to use
  ‣ It only takes a few good abstractions
CACR & The Intel Touchstone Delta: World’s Fastest Computer in 1991 (30 Gflops)

Nehalem 2009
# Top 10 Supercomputers

Rank | Site | Computer/Year Vendor | Cores | R<sub>max</sub> | R<sub>peak</sub> | Power
--- | --- | --- | --- | --- | --- | ---
1. | Oak Ridge National Laboratory, United States | Jaguar - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc. | 224162 | 1759.00 | 2331.00 | 6950.60
2. | National Supercomputing Centre in Shenzhen (NSCS), China | Nebulae - Dawning TC3600 Blade, Intel X5660, Nvidia Tesla C2050 GPU / 2010 Dawning | 120640 | 1271.00 | 2984.30 |
3. | DOE/NCSA/LANL, United States | Roadrunner - BladeCenter QS22/LS21 Cluster, PowerXCell 8i 3.2 GHz / Opteron DC 1.8 GHz, Voltaire Infiniband / 2009 IBM | 122400 | 1042.00 | 1375.78 | 2345.60 |
4. | National Institute for Computational Sciences/University of Tennessee, United States | Kraken XT5 - Cray XT5-HE Opteron Six Core 2.6 GHz / 2009 Cray Inc. | 98928 | 831.70 | 1028.85 |
5. | Forschungszentrum Juelich (FZJ), Germany | JUQUEEN - Blue Gene/P Solution / 2009 IBM | 284912 | 825.50 | 1002.70 | 2266.00 |
6. | NASA/Ames Research Center/NAS, United States | Piz Daint - SGI Altix ICE 8200EX/8400EX, Xeon HT QC 3.0/Xeon Westmere 2.93 GHz, Infiniband / 2010 SGI | 81920 | 772.70 | 973.29 | 3096.00 |
8. | DOE/NCSA/LNLN, United States | BlueGene/L - eServer Blue Gene Solution / 2007 IBM | 212992 | 478.20 | 596.38 | 2329.60 |
9. | Argonne National Laboratory, United States | Intrepid - Blue Gene/P Solution / 2007 IBM | 163840 | 458.61 | 557.06 | 1260.00 |
10. | Sandia National Laboratories / National Renewable Energy Laboratory, United States | Red Sky - Sun Blade x6275, Xeon X55xx 2.93 GHz, Infiniband / 2010 Sun Microsystems | 42440 | 433.50 | 497.40 |

Source: top500.org 6/2010
Roadrunner

Three programs work together

12,240 Cells and Opteron cores

Good power efficiency (acceleration): 444 Mflops/W

Low cross section (reliability)
Portable Electronic Devices: Apple A4 to CI

- **A4 guess?**
  - SoC/PoP
  - ARM/GPU/USB 2.0/Flash cntrl.
  - 256 MB
  - 4 Gflops? at 1 W
  - 64 GB NAND Flash

- **CI**
  - Proc./Accel./Link/Flash engine
  - 1 GB LDDR2 (64 bit wide) PoP
  - 40 Gflops at 6 W + 2 W (1 Ghz?)
  - 128 GB NAND Flash RAID
  - All existing IP; need <1 year

Sources: Apple, Wikipedia, Micron, Toshiba
What do you get when you cross Roadrunner with iPads?
The Engine’s Cylinders: FlashBlades

- “X1” is an FPGA switch for C1 array & QPI to CPU & PCIe to IB
- The CPU orchestrates *abstractions*; to the CPU the array looks like:
  - A 6.14 TB, 25 GB/s (burst), 50 us, || disk (file system, triple stores)
  - A 2.56 TFlops accelerator (OpenCL with embedded triple stores)
- This all fits on a standard blade (2 sides) and uses commodity IP
  - Draws about 600 W (~4,300 Mflops/W + CPU perf.)
FlashBlade Detailed
Blade & Server Specifications

- **Flashblade**
  - CI array (64x 40 Gflops/1 GB): 2.56 Tflops/64 GB
  - Flash array (64x 128 GB 75% RAID): 6.14 TB useable
  - X I/O bandwidth: 32 GB/s in CI array & 25 GB/s to CPU
  - Power (64x @ 8 W + 100 W): ~600 W
  - Estimated cost per blade: $25K (not including CI&blade NRE)

- **Blade server** (e.g. IBM BladeCenter/E 14x at 7U)
  - CI arrays: ~36 Tflops & ~900 GB
  - Flash arrays: ~86 TB
  - Network (28x IB @ 40 Gb/s): 112 GB/s
  - Power: 8400 W for blades (BC/E max is 9300 W)
  - Estimated cost per server: $400K = $350K+$50K system
Implications for Data to Discovery

• **HUGE** data processing capability: a single server can read (and “process”) its entire contents in about 10 minutes (200 MB/s)
  ‣ A same size disk array would take 100x to read (2 TB disks)
  ‣ 100x faster at random access (50 us vs. 5 ms)
  ‣ Balanced I/O and computation

• *This is qualitatively new, what could we do with it?*

• Want applications with #reads >> #writes
  ‣ One rack (0.5 PB) could handle LSST processing for a year?
  ‣ Good for LHC/CMS data analysis (comparing data to Monte-Carlo simulations of the standard model)
  ‣ Analytics, search, intelligence, video processing, tomography, ...
  ‣ Metadata server for massive archival disk arrays
Basic Web Search

• Basic search data layout on blade
  ‣ Page text on flash with cross link info
  ‣ Word to page hash table in C1 array DRAMS

• Data capacity in server
  ‣ \(~1\) B pages and \(~10\) M words (@ \(~100\) KB data/word) max
  ‣ One of the CPUs assembles final result across blades
  ‣ Analysis shows “bottleneck” is flash page read latency
  ‣ Assume 100 fetches to get top 20 results & random dist.

• Google: 1 B searches/day; 25 B pages?? Many other services.
  • A single server could handle \(~10\) B basic searches/day on 1 B pages
  • Tremendous power savings

• Metadata: Store only cross link info in flash; pages on disk
• Conclusion: viable for basic search (#reads >> #writes)
Triple Stores

- Triple stores
  - (object, attribute, value)
  - Graphs: (a, f, b)
- Query: “select (?, f, b) and (b, ?, c)”
- Implementation
  - Hash tables for partial triples (x, y, ?)
  - Performance order dependent; need working memory
- Use distributed hash tables across C1 array
- Flash byte random access mode latency: ~1 us vs. 5,000 us
  - 1,000x +
- Inferencing performance and mutation rate?
## Flash Parameters & Retention

<table>
<thead>
<tr>
<th>Projection</th>
<th>FB1 2012</th>
<th>FB2 2014</th>
</tr>
</thead>
<tbody>
<tr>
<td>Package capacity (GB)</td>
<td>32</td>
<td>64</td>
</tr>
<tr>
<td>Package size (mm x mm)</td>
<td>14 x 18</td>
<td>14 x 18</td>
</tr>
<tr>
<td>Package area (in²)</td>
<td>0.39</td>
<td>0.39</td>
</tr>
<tr>
<td>Read bandwidth (MB/s)</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>Write bandwidth (MB/s)</td>
<td>200</td>
<td>400</td>
</tr>
<tr>
<td>Read latency (us)</td>
<td>50.6</td>
<td>37.8</td>
</tr>
<tr>
<td>Read power (mW)</td>
<td>175</td>
<td>296</td>
</tr>
<tr>
<td>Write power (mW)</td>
<td>368</td>
<td>368</td>
</tr>
<tr>
<td>Max power density (W/in²)</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>Capacity density (GB/in²)</td>
<td>82</td>
<td>164</td>
</tr>
</tbody>
</table>

---

**NAND Characteristic Retention vs Program/Erase Cycles**

- Constant Bit Error Rate

---

Source: Micron Technology
A Plan for Viability Demonstration

- Experimentally explore write-wear vs. retention time
  - Use production parts and project to $FB_i$
- Applications:
  - HPC Challenge Benchmarks, e.g. DGEMM & 1d FFT
  - Triple store performance (Lehigh Univ. Benchmarks)
- Optimize blade parameters for applications performance
  - CI power allocation to blocks
  - X1 latency and X1-CI link speeds
- Estimate sustainable triple store mutation rate vs. fixed MTTF
- Viable (cost effective?) for some (important?) class of apps.
Evolution is the default optimization process for natural computing...

...what does “it” have to say about power and packaging?
It Only Takes $10^{16}$ Ops (?)

A $10^{16}$ Flops Engine

- Quantity: Need about 300 servers (50 racks)
  - Big IB switch fabric too (168 IB ports/rack)
  - Volume (need lots of air): 10,000 ft$^3$
- Flash memory: $\sim 26$ PB
  - SDSC runs 18 PB of tape storage (1,000’s of scientific data sets)
  - Constraint: no more than 1,000-2,000 writes/chip/day?
  - Data ingestion and reflective analysis by engine
  - Checkpoints in <10 s
- About 10 Pflops at 3 MW (does not include fabric or cooling)
- Cost unknown due to rapidly dropping flash costs, new packaging, and other economies of scale
  - Guess about $\$100M$ acquisition (2012)
Comparison to Another Data to Discovery Engine

- Operations: 10 Pops (1x)
- Memory: 1 PB (0.04x & forgets)
- Bandwidth: 1 PB/s? (0.5x - 12x - 30x)
- Packaging: 0.25 ft³ (40,000x)
- Power: 25 W (120,000x)!

**Where’s the algorithm?**

Sources: Brain picture: Wikipedia; Memory: R. Kurzweil, pg 127; Bandwidth: author
Scaling Speculation

- At 32 nm and can probably get to ~10 nm w/ Si: 4 generations?
  - Off by 7,500x in power (~200 KW)
- Graphene-based tech can get to ~1 nm?: 6 more generations?
  - Power efficiency unclear, but getting close
  - A high number of (virtual) edges between packages is essential
- If all goes really well, we might be at 25 W by ~2035
  - May need time to climb another “S” curve
- See “ExaScale Computing Study”, P. Kogge (ed), 2008 for what might be achievable by ~2018 with Si
Some Clues from Biological Systems: Power

- Fly photoreceptor energy:information trade-offs
- Energy: ATP molecules hydrolyzed per bit
- Higher performance neurons required more energy per bit
- From Niven et al., PLoS Biology, April 2007:
Some Clues from Biological Systems: Packaging

- VLSI has a few interconnect layers (many more process layers)
- Flashblades are essentially 3 layer instead of 1 layer
- Fractal topology: \( \log(#\text{nodes}):\log(#\text{ext. edges}) \) scaling with box size
- Data from Bassett et al., PLoS Computational Biology, Apr 2010:

<table>
<thead>
<tr>
<th>Network</th>
<th>( D_{\text{Euclidean}} )</th>
<th>( D_{\text{Fractal}} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>VLSI</td>
<td>2</td>
<td>3.81( \pm )0.64</td>
</tr>
<tr>
<td>C. elegans</td>
<td>3</td>
<td>4.42( \pm )1.53</td>
</tr>
<tr>
<td>Brain (MRI)</td>
<td>3</td>
<td>4.12( \pm )1.55</td>
</tr>
</tbody>
</table>
Back to the present...
Programming

• Stuck with assembly language for many years (C+OpenCL+MPI)
  ‣ MPI rank ~4,000 with ~1,000 threads/blade
  ‣ Theoretical limits on what we can do automatically
  ‣ Although introspective tuning is promising
• Scalable parallel programming requires kernel-level hacking skills
  ‣ These skills are rare, perhaps 2% of programmers
• Abstraction is the key:
  ‣ LAPACK (Vector & MPI parallel)
  ‣ OpenGL and OpenCL (Accelerators)
  ‣ File systems, triple stores, SQL, and MapReduce (Disk arrays)
• What is an abstraction for “data to discovery”?
Some Characteristics of Good Abstractions

• Specify *what* not *how*
  ‣ E.g. factor a matrix or render an image
• The *what* should be implicitly parallel in some way
  ‣ Map over matrix elements, polygons, pixels, rows, columns, ...
  ‣ Implementations provide balance & fault tolerance
  ‣ Communicating Accelerated Processes (CSP→CAP)
• *The fundamental operations must lead to global error bounds and convergence rates (algorithms)*
  ‣ This is crucial because the algorithms might do $10^{21}$ ops
• Parameterized (consistent with the above)
  ‣ Error bounds, data types, realism level, ...
Engine Software Architecture

Applications
- Domain Abstractions & “Orchestration”
- Numeric & Storage Kernels
- “MPI” & OpenCL

Distributed Substrate
- Integrated Compute & Storage
- Networked Data Resources

Balance

LAPACK, SQL, PETSc, Root, etc., Data to Discovery?

Algorithms Convergence & Complexity?

Communicating Accelerated Processes Semantics & Tuning?

FlashBlades?
Concluding Remarks

• We are not stuck with “clusters”: COTS is also IP not just Fry’s
• Trans petascale data computing systems can be built now
  ‣ Hardware and software are not necessarily barriers
  ‣ It will require investment
  ‣ Two+ orders of magnitude step-up possible for data intensive systems
• A challenge is finding good abstractions for “data to discovery”
  ‣ Implementing the abstractions might require algorithm research
  ‣ This is the key interplay between Science & Computer Science
  ‣ Efficient implementations of good abstractions will happen
• FlashBlades are probably viable: we should find out
• Think in terms of what can be done with a shrinking $10^{16}$ ops system