Scientific Computing at Jefferson Lab
Petabytes, Petaflops and GPUs

Chip Watson
Scientific Computing Group
Jefferson Lab

Presented at CLAS12 Workshop, May 25, 2010

Purpose of this Talk

1. Inform

   you about Jefferson Lab's resources and plans

2. Entertain

   with useless (?) trivia about computing trends

3. Provoke

   conversation about computing plans & strategies

Scientific Computing at JLab

Significant computing capacity

- 1400 compute servers
- leading edge GPU cluster
  (single most powerful dedicated LQCD resource)
- growing repository of 6 GeV data, more than 4 PB today
- rapidly growing disk capacity and bandwidth,
  ~ 600 TB disk, half on a 20 Gb/s Infiniband fabric
- exploiting & developing leading edge open source tools
  Lustre file system, Auger / PBS / Maui batch system, Jasmine storage

Software developments

- multi-threading and multi-GPU libraries for high performance computing
- LQCD algorithm development, performance optimizations
- highly scalable analysis database for LQCD
- system management tools
  batch, disk management
  power & system monitoring

Simulation & Data Analysis

Hardware

- farm of ~200 nodes, of which ~50 are recent generation
- cache disk (70 TB) and work disk (110 TB)
- 8 LTO-4, 4 LTO-5 tape drives, aggregate bandwidth of 1 GB/s

Near term improvements

- Cache disk capacity & bandwidth: upgrade this summer
- Compute capacity: 6n LQCD cluster (>200 nodes) to be loaned to the farm for the anticipated summer rush (50% capacity bump)
- Additional nodes to be purchased in the Fall

Annual upgrades (next 3 years)

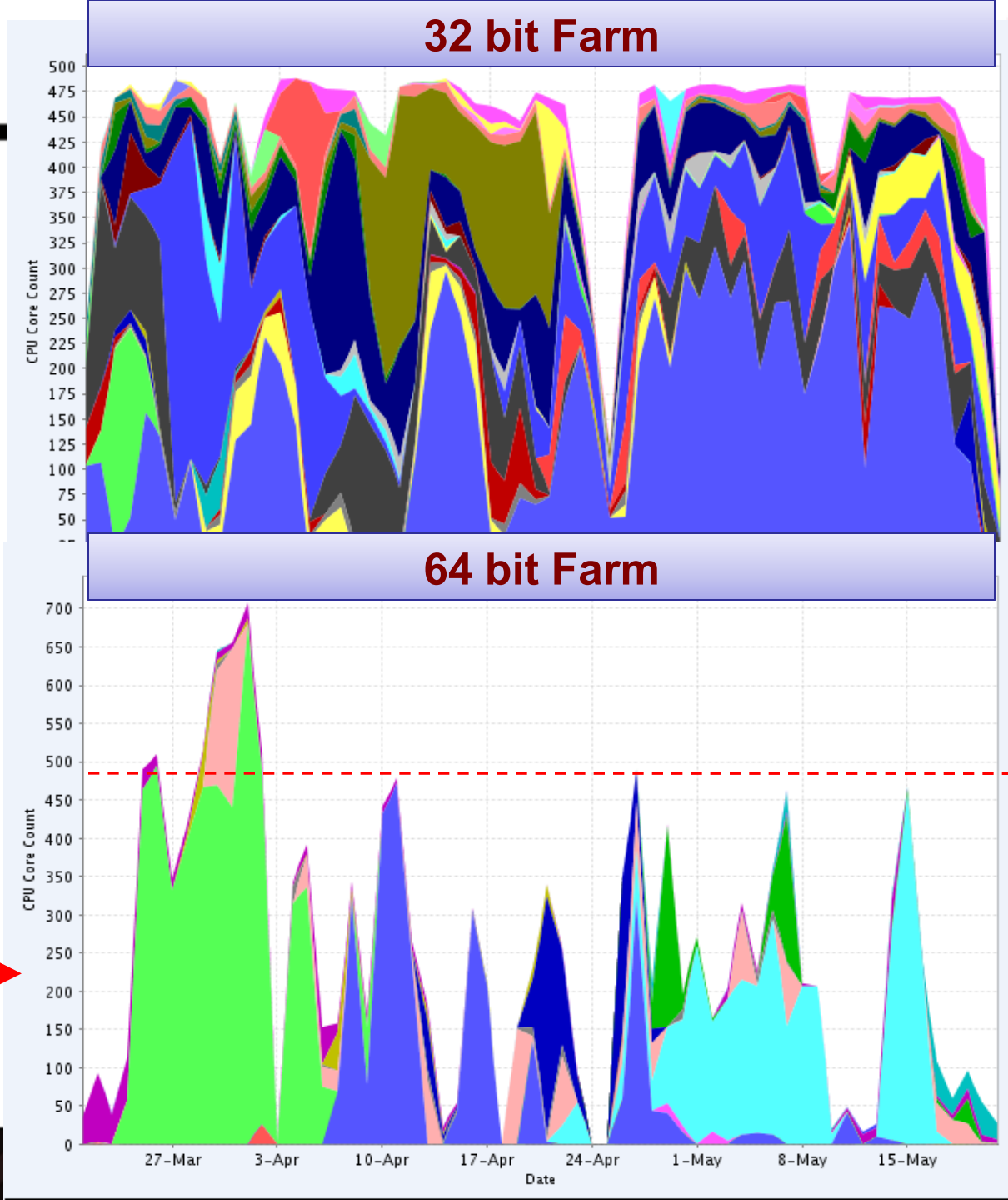
- Requirements driven *(have you submitted yours???)*
- Budget in round numbers: $50K for compute + $50K disk + $50K for tape capacity and bandwidth
- Budget is balanced against needs for 6 GeV staging & running of high priority experiment (i.e. more computing = less running)

Migrating to 64 bit

It is past time!

Just do it!

Over 50% of the JLab farm is idle!



The Computing Challenge

Science has an ever increasing appetite for computing power, sustained (mostly) by Moore's Law trends in transistor density.

In the 1970's and 1980's, clock speeds increased 10x/decade. In the 1990's, 100x in one decade. Since then they have gone flat – chips got too hot (power hungry).

Also during this time clock cycles per instruction were reduced, so that now we do several instructions per clock cycle, but that has also hit a wall for serial code (hard for the complier and silicon to find enough concurrency)

Instead, CPU performance is now growing by increasing in the number of cores per chip, from 2 in 2005, to 12 in 2010.

Science requirements have grown much faster, requiring ever larger budgets and machines.

Current State-of-the-art

8 cores per commodity server (dual socket) is now common

AMD is soon to release the Magny Cours CPU:

- 2 quad hex core CPUs in one socket (MCC)
- no price penalty for quad socket vs. dual socket

$\Rightarrow$ 48 cores per commodity server (this summer)

Memory bandwidth per CPU continues to climb:

- Nehalem (Intel, 2009) 3 DDR3-1333 controllers per CPU
- Magny Cours (AMD, 2010) 4 DDR3-1333 controllers
- (future, 2011) 4 DDR3-1600 or even -1866

Good news: memory capacity therefore also grows!

- quad socket MC has 16 memory buses * 4GB = 64GB for 48 cores
- So, for a while longer, 1 GB / core remains affordable
- BUT: memory now costs more than CPUs!!!
  ($800 / 24 GB server vs $500 / dual socket Nehalem server)

Disruptive Technology: GPUs

A Graphics Processing Unit is a device that does simple arithmetic (+-*/) very fast.

High performance is driven by the video gaming market (both PC/workstation, and game consoles): matrix transformations (movement, rotation, perspective), lighting effects, and other visual depth queues.

In order to yield high frame rates at high resolution with a large number of objects in a scene, each with many facets to render (to appear realistically smooth), a very high floating point rate is required. But the software is relatively simple: lots of linear algebra.

GPUs are now so fast, that they vastly outperform CPUs in doing algebra, and have become a target for computational science.

Modern GPUs

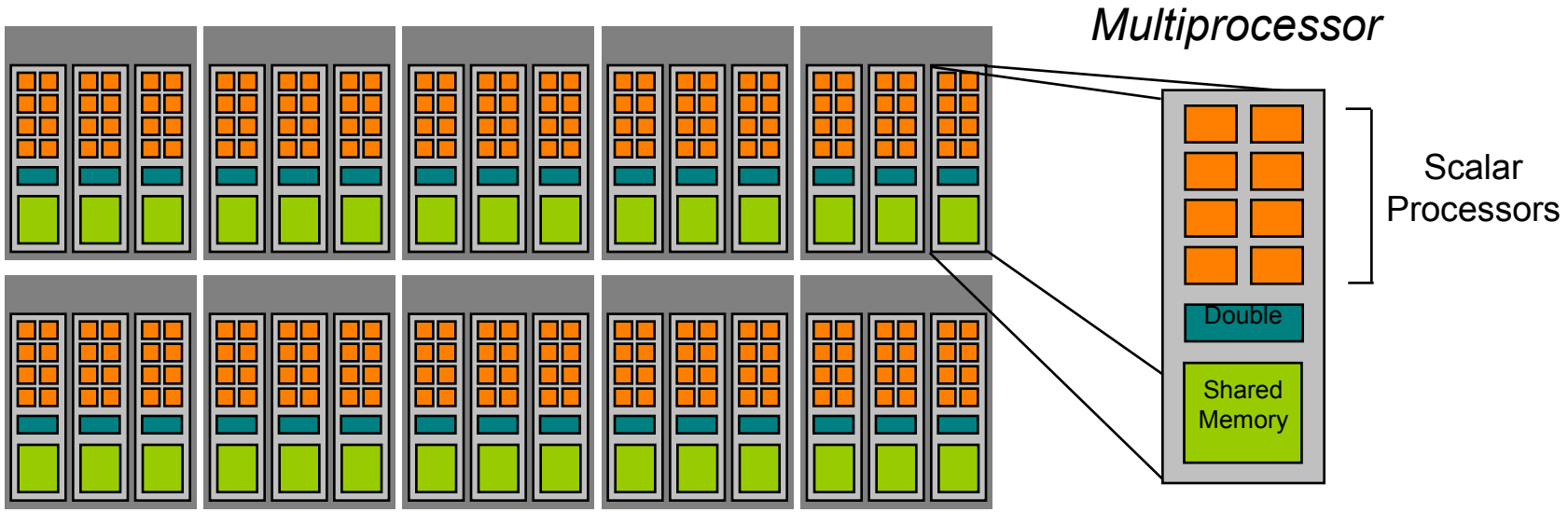
Characteristics of GPUs:

- Lots of simple cores with fast context switching to hide latency
- SIMD architecture (single instruction, multiple data)
- High memory bandwidth
- Complex memory hierarchy, to achieve high bandwidth at low cost
- Recent evolution: IEEE floating point to intentionally support science: GPGPU = general purpose GPU

Commodity Processors	CPU	NVIDIA GPU
#cores	4 (12)	240 (480)
Clock speed	3.2 (2.x) GHz	1.4 GHz
Main memory bandwidth	18 (20) GB/s	159 (177) GB/s (gaming card)
I/O bandwidth	7 GB/s (dual QDR IB)	5 GB/s
Power	80 watts	185 watts

10-Series Architecture (GT200)

- 240 *Scalar Processor (SP) cores* execute kernel threads
- 30 Streaming *Multiprocessors (SMs)* each contain
  - 8 scalar processors
  - 2 Special Function Units (SFUs)
  - 1 double precision unit
  - *Shared memory* enables thread cooperation



New: “Fermi” GPU

Major Changes from last version:

- up to 512 cores (32 sets of 16)
- cache memory: L1 (single SM) and L2 (all cores)
- ECC memory, 2.6 GB or 5.2 GB (or 3 - 6 with ECC off)
- 8x double precision peak rate of prior GPU
  (1 for every 2 cores; aggregate now >600 Gflops)
- wider memory bus, GDDR-5 instead of GDDR-3
  (allows future versions to increase bandwidth)

Software model remains similar: lots of cores, context switching every clock cycle

GPU Best Fit

1. High data parallelism
   - Perform the same set of operations on many data items

2. High flop count on a small kernel
   - Problem needs to fit into the GPU (1 - 6 Gbytes)
   - Little data dependent branching, no recursion
   - Need to do enough work to amortize cost of pushing the problem into the GPU and getting the results back

     (it is possible to pipeline the problem and data)

3. High memory bandwidth requirements
   - And problem doesn't fit into CPU cache

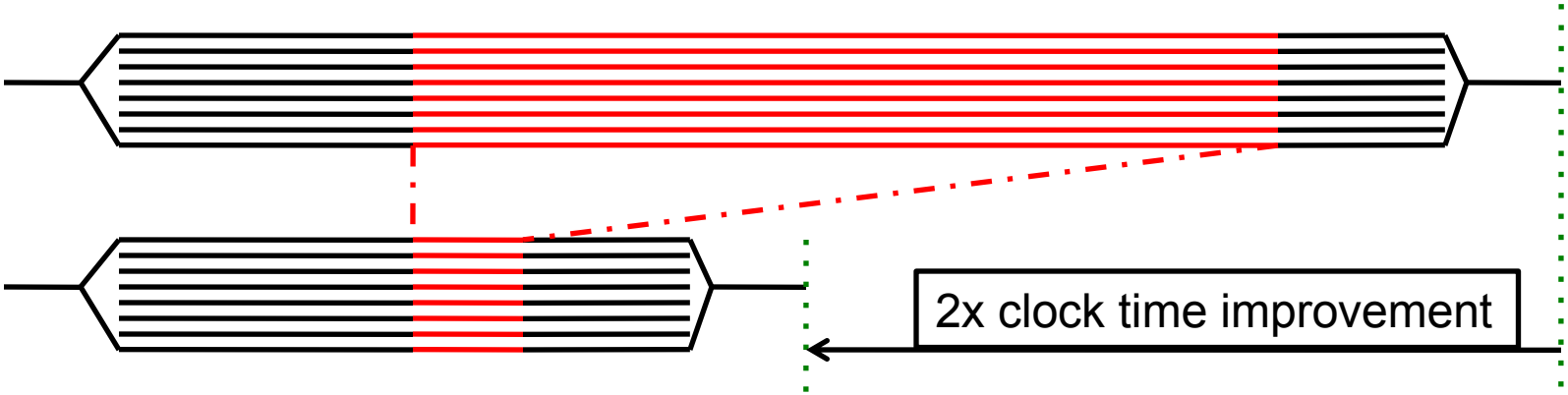
Parallel Software Models

- on CPU: thread libraries to use all the cores
  - OpenMP is the easiest
  - also: POSIX threads, MPI
- on GPU: parallel languages
  - CUDA: mature, NVIDIA specific, widely used
    (Compute Unified Device Architecture, refers to hardware & language+library)
  - OpenCL: new standard
    - Can also run on CPU cores (common abstraction)
    - Task parallel as well as data parallel models
  - Much less mature compilers exist for GPUs than CPUs
    - Need to do manual optimizations, like loop unrolling, data movement to faster scratch pad (similar to a cache memory)
    - Simple code: 30 Gflops, hand optimized: 260 Gflops !

Amdahl's Law (Problem)

A major challenge in exploiting GPUs is Amdahl's Law:

If 60% of the application running time is GPU accelerated by 6x, the net gain is only 2x:



Also disappointing: the GPU is idle 80% of the time!

Conclusion: need to move more code to the GPU, and/or need task level parallelism (overlap CPU and GPU)

Quick Look: LQCD

- Lattice QCD is very memory bandwidth intensive
  - 1 instruction per byte of memory touched; repetitive sweeps through memory
  - Size of lattice problem ~1000 x CPU cache memory size
  - Flops (sustained) is about ½ memory bandwidth
  - Dual Nehalem:

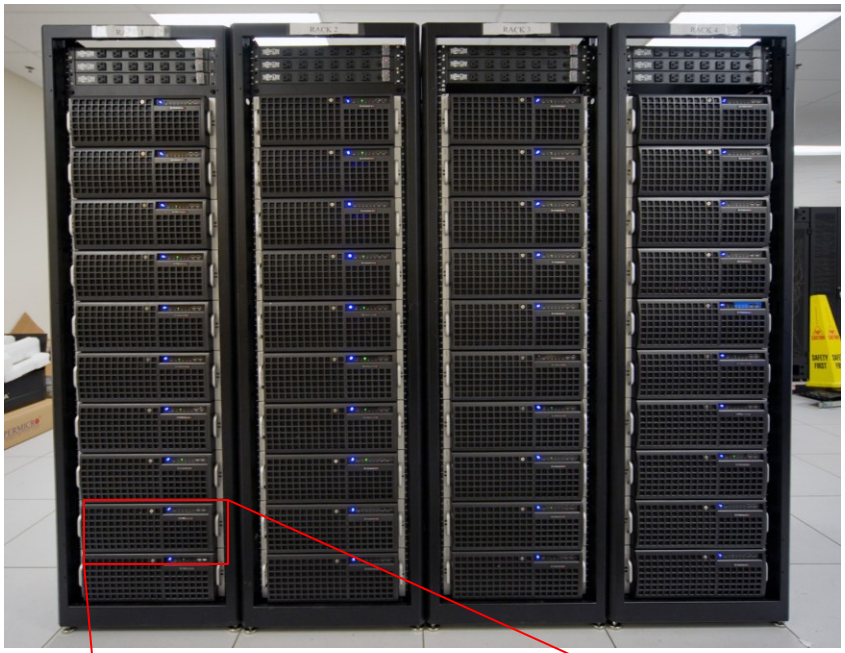
    Streams 37 GB/s, LQCD: ~20 Gflops
  - GPU (GTX-285 gaming card):

    Streams 159 GB/s, LQCD: 120 Gflops

    (with “tricks” up to 200 Gflops!)

- GPU programming is difficult, but the end result is compelling for many science problems

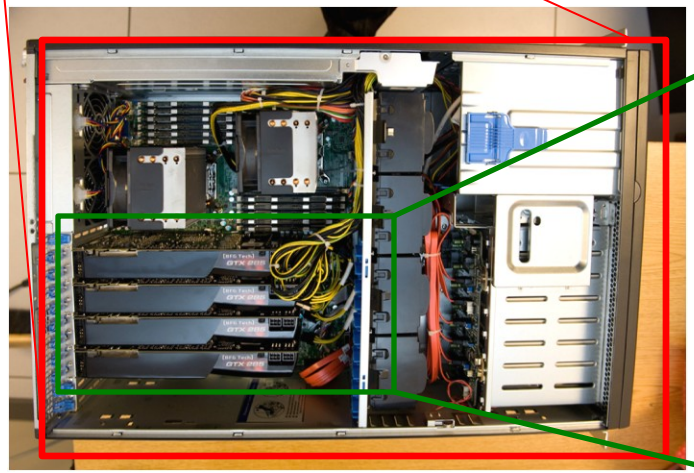
Hardware: ARRA GPU Cluster



Host:
- 2.4 GHz Nehalem
- 48 GB memory / node
- 65 nodes, 200 GPUs

Original configuration:
- 40 nodes w/ 4 GTX-285 GPUs
- 16 nodes w/ 2 GTX-285 + QDR IB
- 2 nodes w/ 4 Tesla C1050 or S1070

One quad GPU node =
one rack of conventional nodes

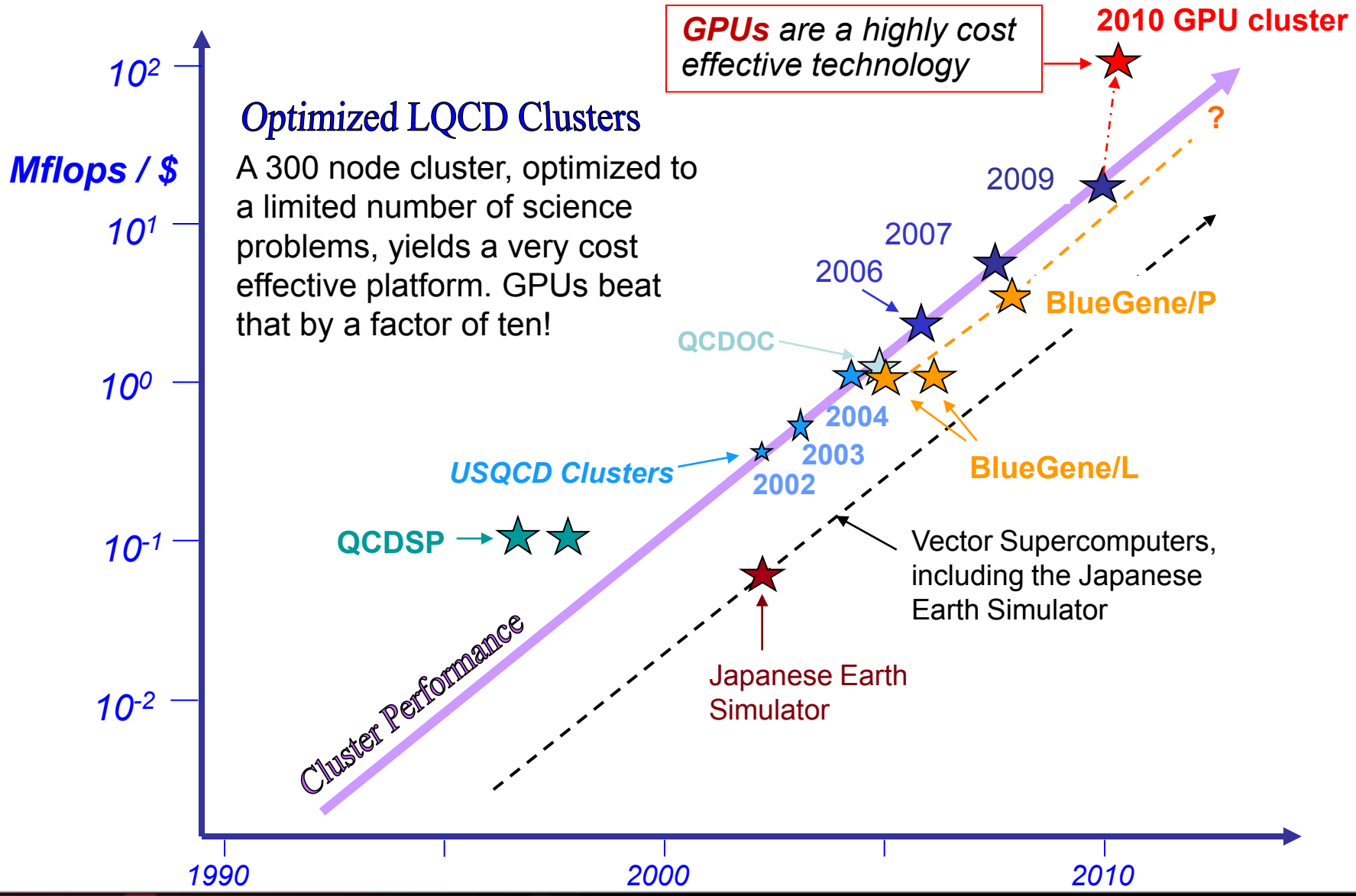


10x Reduction in $/MFlops

- Usage model
  - Single job / GPU; multiple jobs / host
  - Parallel GPU (one host, multi-host)
  - Scaling to teraflops, but constrained by Amdahl's law

- High Capacity Computing Example
  - Dual Nehalem 2.4 GHz host, ~$3,200
    $0.16 / LQCD Mflops
  - 4 2GB gaming cards, each $400
  - 4 jobs, 1 per GPU (low hanging fruit), 1 core on host, 99% of code in the GPU
  - Quad GPU performance: average of 400 Gflops, $4,800
    on kernel, we achieve $0.008 / LQCD Mflops
    w/ Amdahl's Law we achieve < $0.02 / MFlops

Science per Dollar for (some) LQCD Capacity Applications



A Large Capacity Resource

530 GPUs at Jefferson Lab (July)

- ★ 200,000 cores (1,600 million core hours per year)
- ★ 600 Tflops peak single precision
- ★ 100 Tflops aggregate sustained in the LQCD matrix inverter, (mixed half / single precision)
- ★ Significant increase in dedicated USQCD resources

All this for only $1M with hosts, networking, etc.

Disclaimer: to exploit this performance, code has to be run on the GPUs, not the CPU (Amdahl's Law problem). This is a software development problem.

Another Example: ***Folding @ home***

The second generation GPU core, aka GPU2 (April 2008)

"…One of the really exciting aspects about GPUs is that not only can they accelerate existing algorithms significantly, they get really interesting in that they can open doors to new algorithms that we would never think to do on CPUs at all (due to their very slow speed on CPUs, but not GPUs)... The preliminary results so far from it look very exciting, and we're excited to now open up the client for FAH donors to run."

Looking forward

Jefferson Lab can make available GPUs for software development & testing for 12 GeV work

If GPUs prove advantageous, we can deploy GPUs as part of the annual farm upgrades

Needed:

clever programmers who can exploit this powerful tool!

Back to the Future

or, Jefferson Lab's 12 GeV Computing Model

Assumptions

- Moore's Law will hold for another 4 years (in core counts)
- Your software will become multi-threaded to use the cores
- A 10x growth in computing power is free, and matches the expected growth from the 6 GeV era to the 12 GeV era
- While computing is (approximately) free, people are not (currently 80% of farm related budget goes into people)
- The 12 GeV operations budget will allow an additional doubling in compute capacity (hardware) with no discussion (i.e. upon demand); this is only a 20% budget impact
- All raw, reconstructed, and simulation data will be held at Jlab, with redundant copies for all raw data

Tape Trends

Tape Library & LTO Technology

- LTO-5 today: 2 TB / cartridge, 140 MB/s
- 2x in capacity in 3 years: 4 TB per LTO-6 tape
- LTO plan: 1.5x in bandwidth per generation but, LTO-5 missed this target (only 1.2x)
- assume LTO-6 does likewise=> 170 MB/s per drive
- price model: $40 / tape, $30 / slot, $10K / drive
- capacity thus dropping from $70/TB to $50/TB this year
- bandwidth: $70K / GB/s today; ~$60K / GB/s at turn on
- implications: we will need to pay more for bandwidth in the future

Archival Storage for 12 GeV

Tape library today:

- LTO-4 5,500 slots, can go to 10,800:

  ~20 PB LTO-5 capacity @$70/TB

- 2nd library of 20K slots, LTO-6 expansion as needed, FY13

  + 80 PB LTO-6, $30/TB

- 6 GeV era: 1 PB/year, $80K / year
  (FY2009, includes bandwidth increase)

- **12 GeV era, 10 PB/year, $300K / year**
  (largest offline component)

- at 40 PB / year, storage cost becomes *very* significant,
  as does footprint, and we may need to evaluate on-demand
  simulation instead of storage of pre-simulated data

Thought Experiment

"Generate on demand" cost:

1. What does computing cost today to generate 100 MB/s of simulated data (2.8PB / year)?
2. Divide this by 4 for 2013, and by 10 for 2015

"Store and retrieve" cost:

1. Cost of storage divided by average number of times each simulated event will be re-used
2. Cost of bandwidth to/from tape: $12K per 100 MB/s stream, amortized over 3 years

Example: Generate at 100 MB/s, reuse=2, so consume at 200 MB/s; storage cost = 130 + 3*12/3 = $142K / year (plus library maintenance)

If the simulation cost of 100 MB/s is a farm with replacement cost of $112K (i.e. $50K / year), then doubling the size of the simulation farm and doing "generate on demand" saves $30K in the first year, and $100K+ in future years (computing falls faster than storage)

Disk Trends

Capacity Costs

- Disk: $400 / TB (moderate performance)
- Low bandwidth disk: $200 / TB
- Tape: $50 / TB, so 8x below cache disk, 4x below archival disk

Trends

- Disk is falling faster than tape, but won't overtake it in 4 years
- Statements 5 years ago that disk would overtake tape in 5 years remain "true" today
- My model: disk will gain by 2x in 4 years
- If deployed disk is 10% of tape capacity (moderate performance cache & workspace), then disk is 44% of cost today, and 28% in 2014

Grid vs Mostly Central

Observations

- The grid does not address the largest cost item in the budget (storage)
- A fully functional computational & storage grid adds ~1 FTE in labor at Jefferson Lab, between $150K and $200K with overhead, or 3x to 4x times the computing hardware budget
- Based upon current requirements, a mostly central computing model is the most cost effective (highest performing) solution
- There are other advantages to a grid (easier transparent use of a user local resource), but these mostly exist for larger institutions that are LHC collaborators (costs already paid by HEP)
- There may be other advantages in other software tools, but these have not yet been captured by requirements
- There are non-trivial disadvantages to using a grid, even beyond labor costs for the computer center

Current 12 GeV Computing Plan

Centralized computing model

- full computing capacity for reconstruction, simulation, analysis
- storage for all raw, reconstructed and simulation data
- high network bandwidth for fast transport of reduced (cooked) data to home institutions for interactive analysis

Planned improvements

- better process for updating computing requirements, to allow sufficient lead time to meet those requirements
- easier tools for managing fair share (priority) by hall computing coordinators
- easier tools for managing storage (quotas) by hall computing coordinators
- easier tools for access to farm visible disks, and the tape library from offsite (although not necessarily a full blown data grid)

(Questions / Discussion)