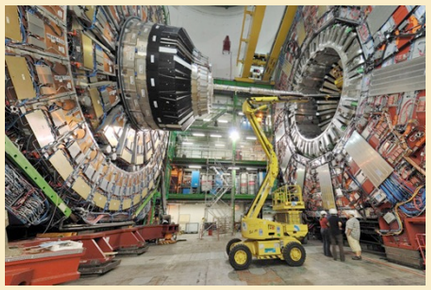
Scientific Computing

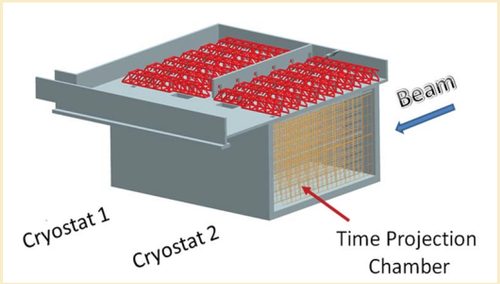
Adam Lyon / Fermilab Scientific Computing Division

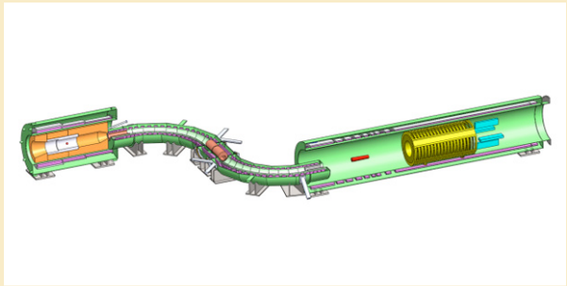
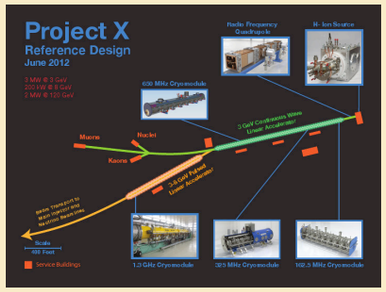
Fermilab Users Meeting, June 2013

The Theme of This Talk

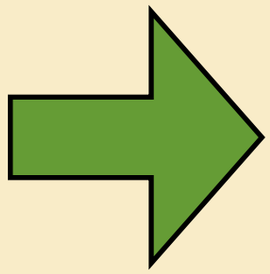
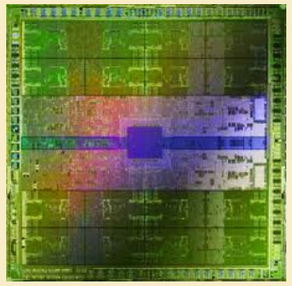
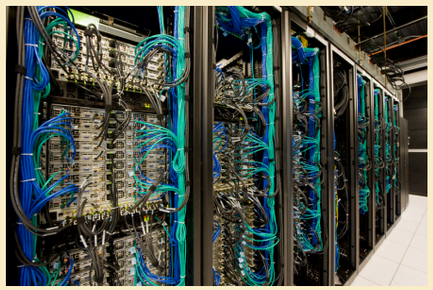
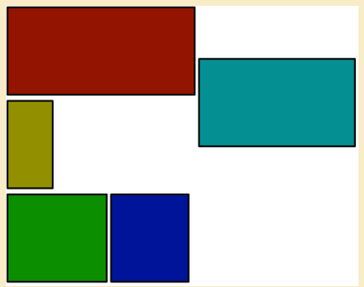
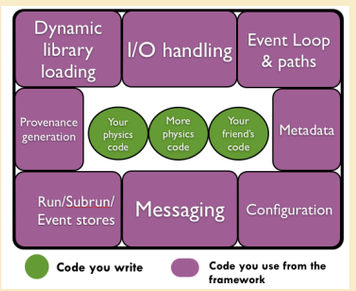
The next level of discoveries





requires the next level of computing

The next level of: Power Scale Efficiency Usability Collaboration

What is taking us to the next level?

- **BIG Data** – process, store, move, plot
- Next level technology – provides new opportunities but also new difficulties
- Computing is more specialized – experts needed
- Complex computing needs to be usable
- Science demands reproducibility; Collaborate through code
- Discoveries demand more complicated analyses

The next level of: Power, Scale, Efficiency, Usability, and Collaboration

The Fermilab Scientific Computing Division is involved in all aspects of the "next level of computing" evolution

About the Fermilab SCD

Mission: **Provide computing, software tools, and expertise to all parts of the Fermilab scientific program including theory simulations (e.g. LQCD) and accelerator modeling.**

Work closely with each scientific program as our valued customers. We also work as collaborators when SCD scientists/staff are directly involved with a program (liaisons).

Create a coherent Scientific Computing program from many parts and many funding sources

Encourage sharing of facilities, common approaches and tools, and re-use of software where ever possible

Work closely with the Core Computing Division as part of an overall coherent program

The Fermilab SCD in a nutshell

160 people in the division, nearly all technically trained

26 Scientists, representation in nearly every experiment/program

Heavily matrixed

Future program and Experiments; Scientific Programs (CMS, Astro, REX); Scientific Computing Facilities

Liaisons: two-way conduit representing an experiment/program to the SCD and the SCD to the experiment/program; an insider on both sides

Computing staff is shared amongst experiments/programs, especially for IF

Agility is important – as the lab changes mission and the computing landscape changes, we adapt – and our structure allows us to do so

Especially important for computing at Intensity Frontier Experiments

Computing Requirements to do Physics

- Science demands reproducibility.
  Need control over our software
  Version control systems; software repositories

- We want to work together.
  Share ideas through code & algorithms
  Expert written common modular frameworks

- Do physics, not computing.
  Processing data should not be taxing on people
  Expert written common infrastructure and services

Next level of efficiency, usability, collaboration

The FIFE Project for IF and others

A collection of projects that provide common computing services and interfaces needed to turn a physics task into results

Happy Physicist (you?)

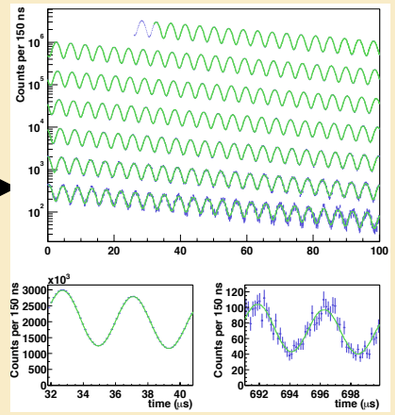
Physics task

Job submission

Open Science Grid

Grid Jobs on Fermigrid/ OSG

Output Files



Result

Data Handling

Files

DB Applications

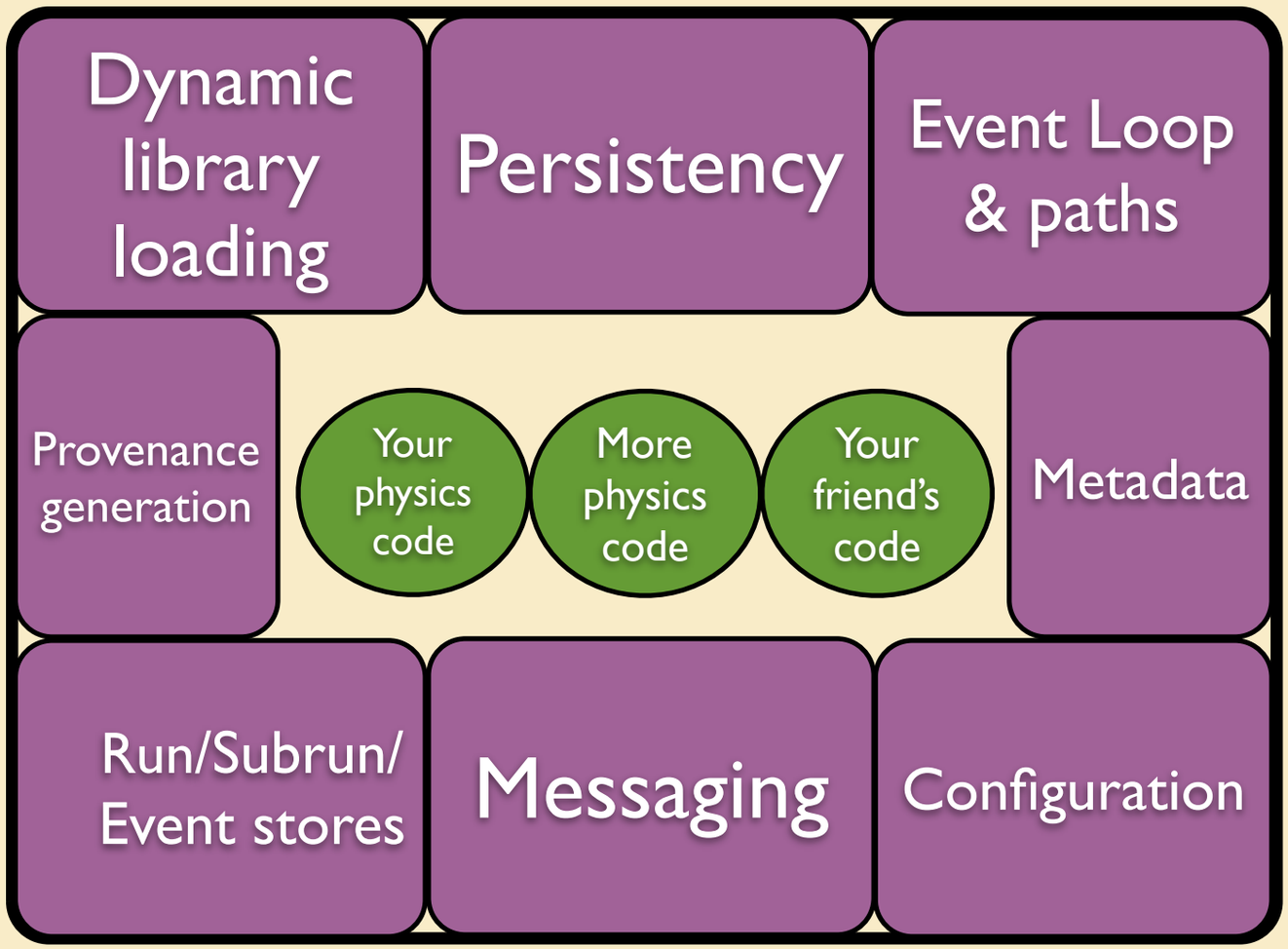
Databases (calib/ conditions, ...)

Next level of usability and scale

Tailored for opportunistic running;

CMS has similar system (workflow, glide-ins); adopting oppo running too

Framework for the Intensity Frontier and others



ART – A lite, forked version of CMSSW tailored for IF

Modularity makes it easy to collaborate

Physicists write physics code and algorithms, not infrastructure

Utilizes modern C++2011

Adopted by NOvA, Mu2e, MicroBoone, LBNE (LarSoft) Muon g-2, DarkSide50

Adapting these ideas to Cosmology and Astronomy

Art-daq and multicore/multiprocessor ART

Next level of usability and collaboration

The Free Lunch is Over

Historically, CPU speed doubled every 18 months (Moore's law)

My awesome computer when I was an undergrad

An Osborne Executive portable computer, from 1982 with a Zilog Z80 4MHz CPU, and a 2007 Apple iPhone with a 412MHz ARM11 CPU. The Executive weighs 100 times as much, is nearly 500 times as large by volume, costs approximately 10 times as much (adjusting for inflation), and has 1/100th the clock frequency of the phone.

iPhone 5 is 1.3 GHz dual core (300x clock spd, 125 times lighter, 1/10th the cost)

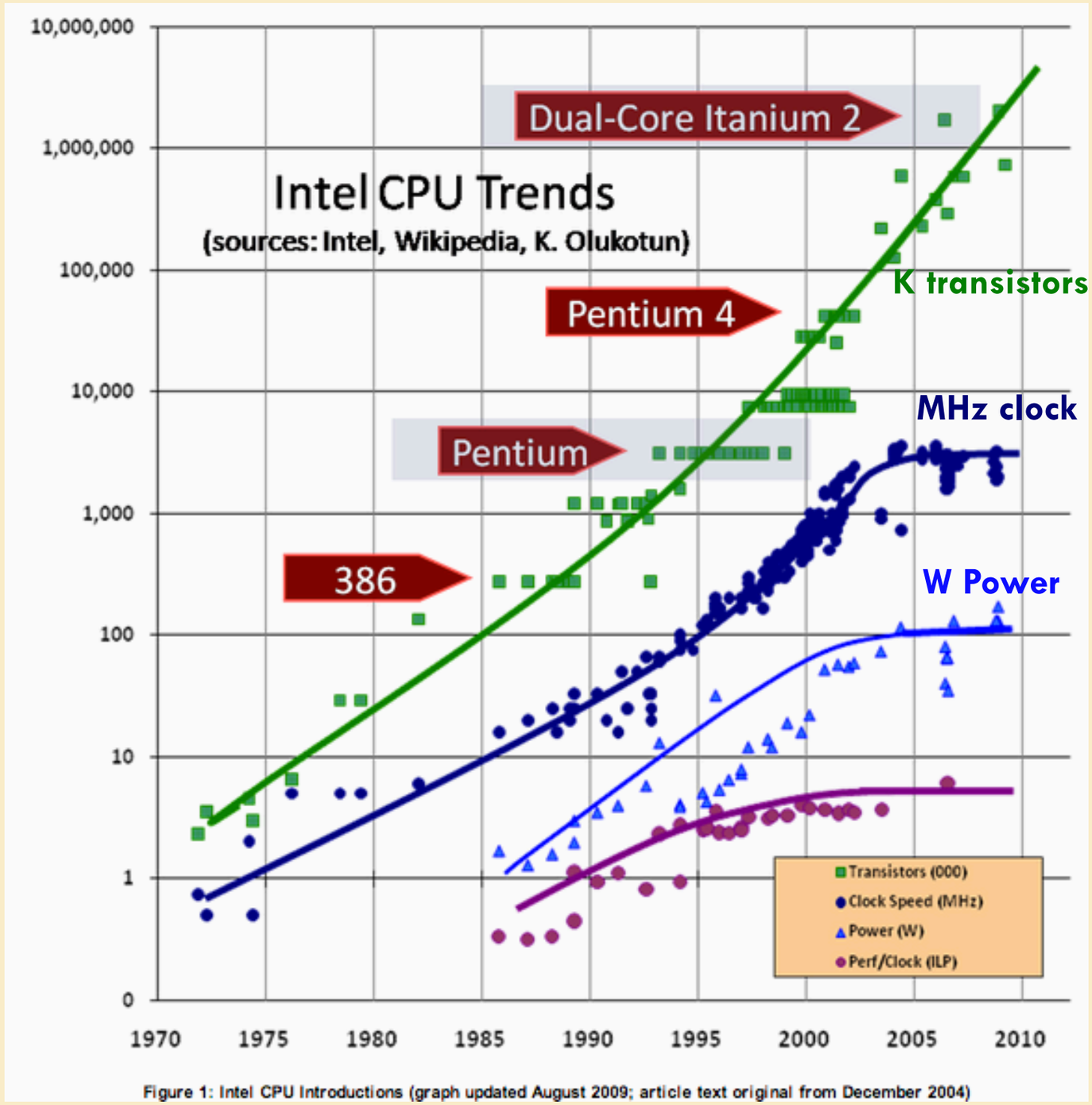


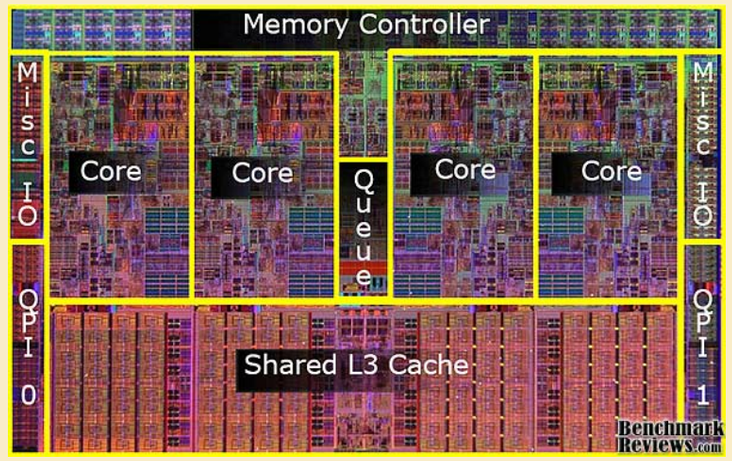
Figure 1: Intel CPU Introductions (graph updated August 2009; article text original from December 2004)

But not anymore (since 2004)

Why no 10 GHz CPUs? Heat dissipation, power consumption, current leakage (see http://www.gotw.ca/publications/concurrency-ddj.htm)

The increase in speed had been useful, sometimes crucial, for experiments

More cores == more throughput, maybe



Instead of speed doubling in 18 months, number of cores doubles

(Can't make them faster, so give you more of them) The next level of computing power

But more cores means you need more memory

Fortunately, memory prices halve every ~18 months

Our standard is 2 GB memory/core; we purchase machines with 64 cores (4x16) and 128 GB of memory

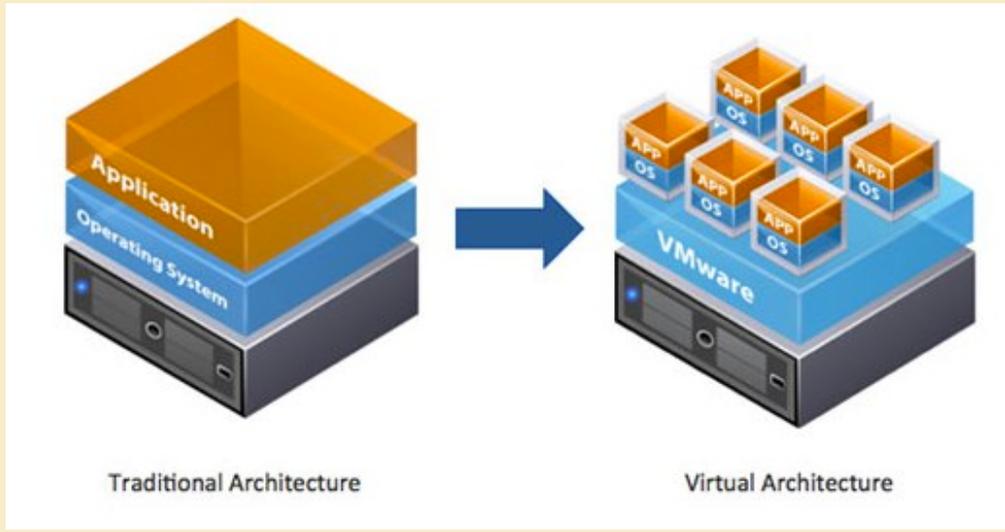
Typical use is to give each job a core -- one core per "slot" (Event level parallelism)

Implications of Multicore Machines (1)

All this power is wasted on single-service, single-use machines (servers, interactive nodes, development boxes). To the Next level of efficiency

Solution: Virtualization and Clouds

Virtualization: Run many "virtual" machines on a real machine (bare metal)
E.g. General Physics Computing Facility for interactive nodes and Run II Data Preservation



Clouds: Dynamically provision virtual machines from a pool
FermiCloud – Used for development & testing, HA services for Fermigrid, Servers

Next level of scale (e.g. studying Cloud Bursting to address usage spikes; using CMS High Level Trigger Farm as a cloud)

But virtualization is not optimized for everything (e.g. building code)

Implications of Multicore Machines (2)

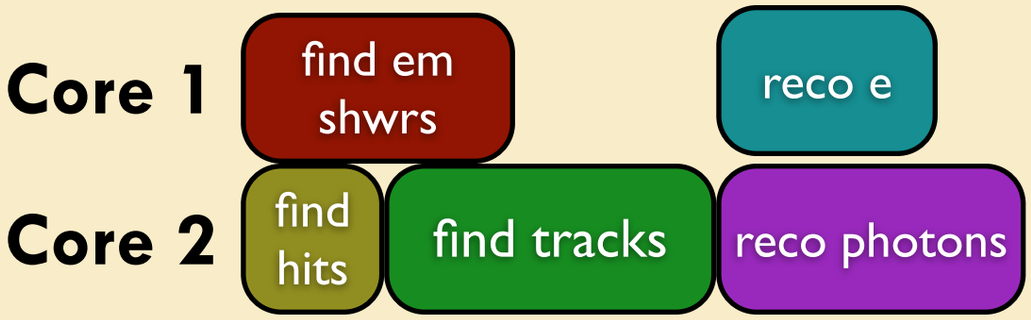
The 2 GB/core memory limitation can be problematic

2 GB may not be enough for reconstruction algorithms
E.g. CMS reconstruction with large pileup, LBNE hit finding

Perhaps idle some cores to take the memory, but problematic and wasteful
Sites are reluctant to devote resources to whole-node job queues

Next level of efficiency: Split up the event processing tasks among multiple cores (parallelization). But now you have to handle task dependencies

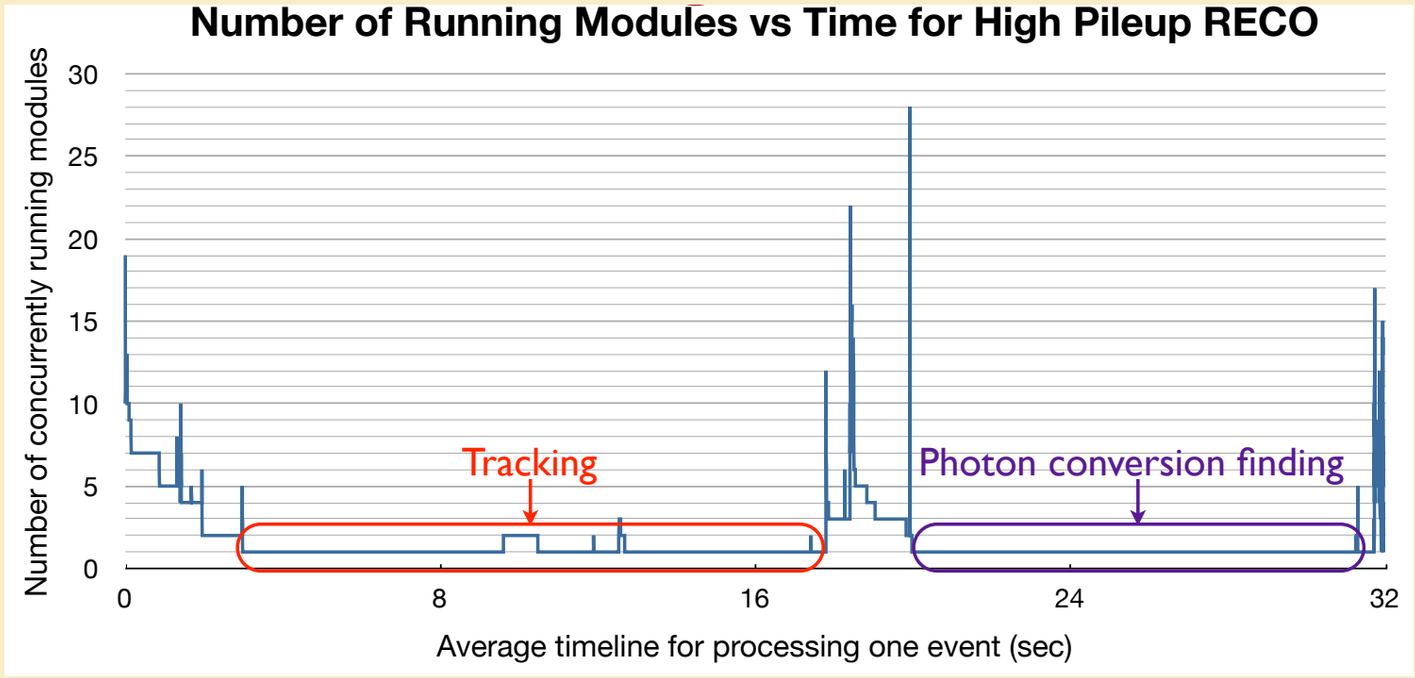
Find EM cal showers, Find tracker hits,
Find tracks, Reconstruct electrons,
Reconstruct photons



Sites need to give >1 core/job, but won't have completely idle cores

Parallelization of processing tasks

Threading frameworks help with coding the dependencies (e.g. Intel Thread Building Blocks – TBB)



Both CMS and ART Framework groups in the SCD are adapting

E.g. CMS reconstruction, Opportunities for deep parallelization?

Need to be conscious of i/o and network bandwidth

Don't want cores repeating operations

These techniques begin to blur the line between commodity Grid nodes and HPC (e.g. massively parallel BlueGene). CMS is studying HPC use (1 yr initiative) and we have other HPC efforts (ComPASS, LQCD)

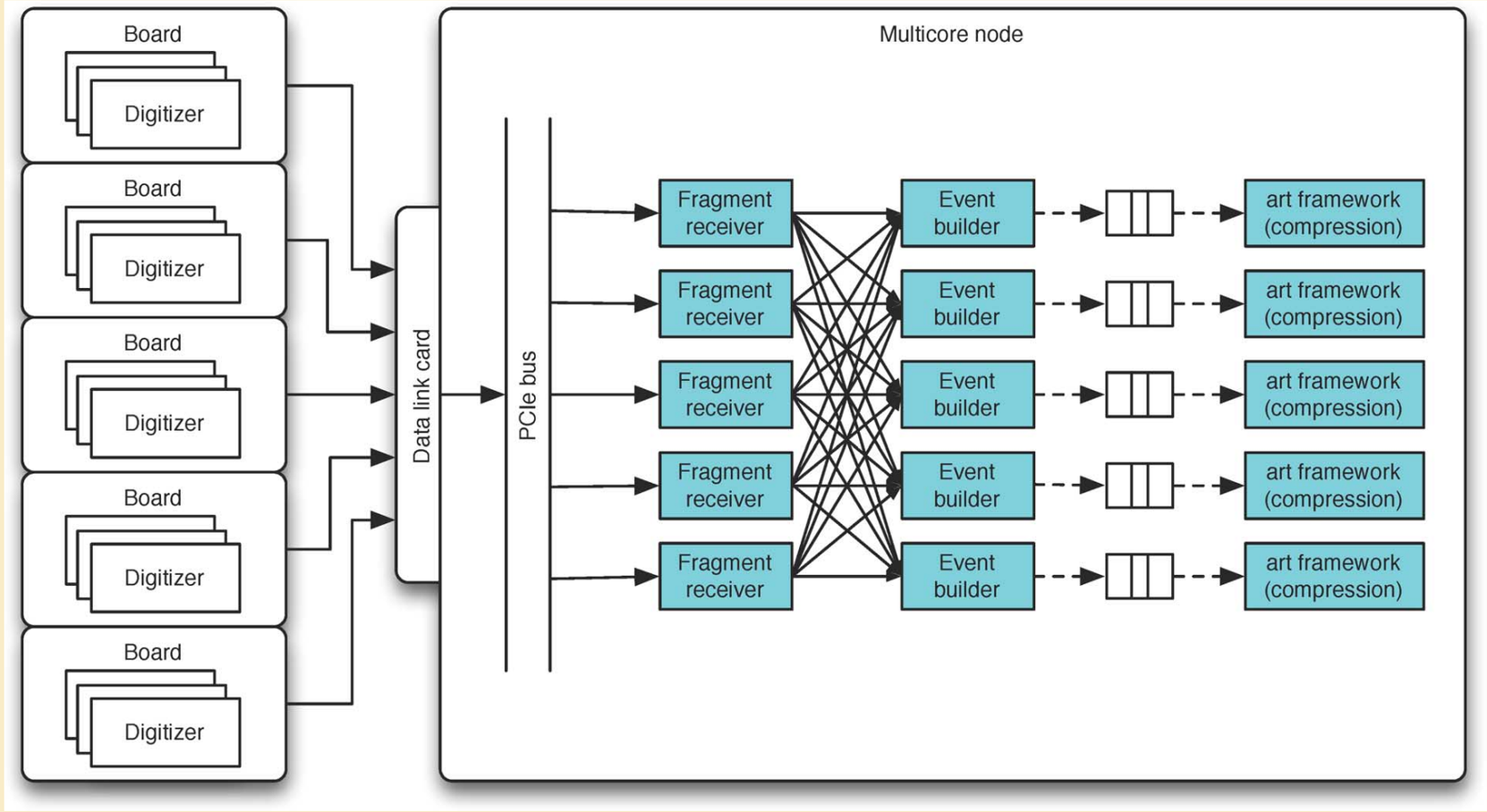
Parallelization useful for DAQs too

Art-daq

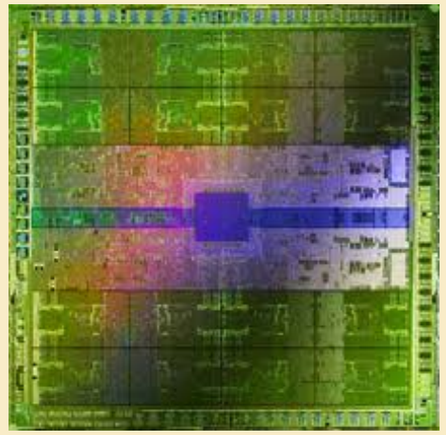
Next level of power, scale and efficiency

Similar online/offline systems have advantages

Adopted by DarkSide50 with HPC networking (Infiniband)



General Purpose GPUs



Massively parallel systems on a chip

Thousands of cores

GPGPUs are best suited for "Data Parallelism" - running the same specialized task on different pieces of distributed data

Contrast with multicore CPUs - best suited for "Task Parallelism"

Next level of power, scale and efficiency

Can see orders of magnitude speed improvements over CPUs for appropriate use cases (e.g. 200x speed up)

Need C/C++ extensions (CUDA, OpenCL) - not trivial to program

GPGPU Farms exist – challenge is to integrate with CPU based workflows

Fermilab recently installed a 152 GPU farm for Lattice QCD (part of wider 5 yr project)

Next generation GPGPUs and Beyond

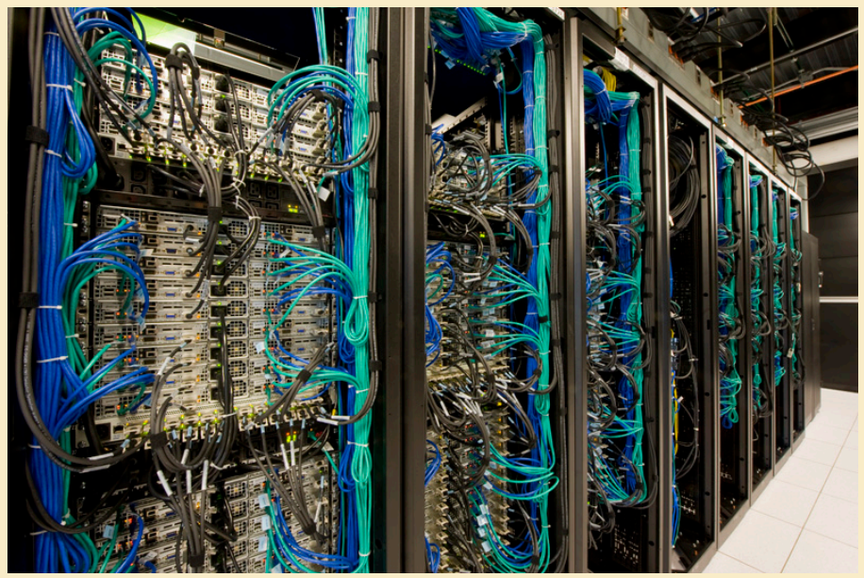
Next level of power, scale and efficiency - and usability?

o GPGPUs with conventional ARM processor on board
Offload administrative tasks – e.g. moving data

o INTEL Xeon Phi Co-processor – Many integrated Cores (MIC)
[actually 60 souped up pentium cores]

Runs Linux – Easier to program – Optimize code for MIC but will run on regular CPUs

Designed with science applications in mind



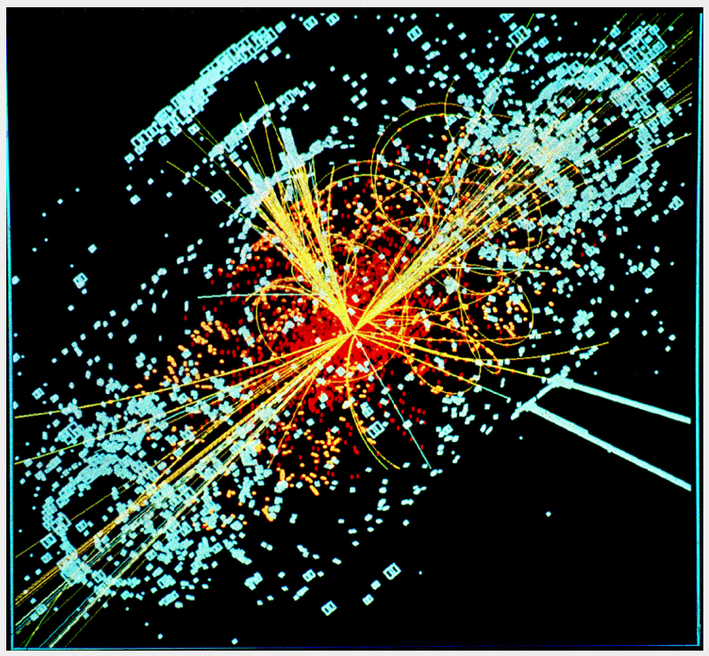
These are under study for LQCD
Can we use them elsewhere in HEP?

More Parallelization efforts in the SCD

Transforming GEANT4 for the Future

Report from the Workshop on Transforming GEANT4 for the Future,
Rockville Maryland, USA
May 8-9, 2012
DRAFT

Robert Lucas and Robert Roser, Editors and Workshop Chairs



Transform Geant4 to run efficiently on modern and future multi-core computers and CPU/GPU hybrids

Joint 2 year initiative between HEP and ASCR (Advanced Scientific Computing Research) with SCD involvement

Efforts to parallelize Root as well

Investments in software to take advantage of the next level

Next level of Simulations

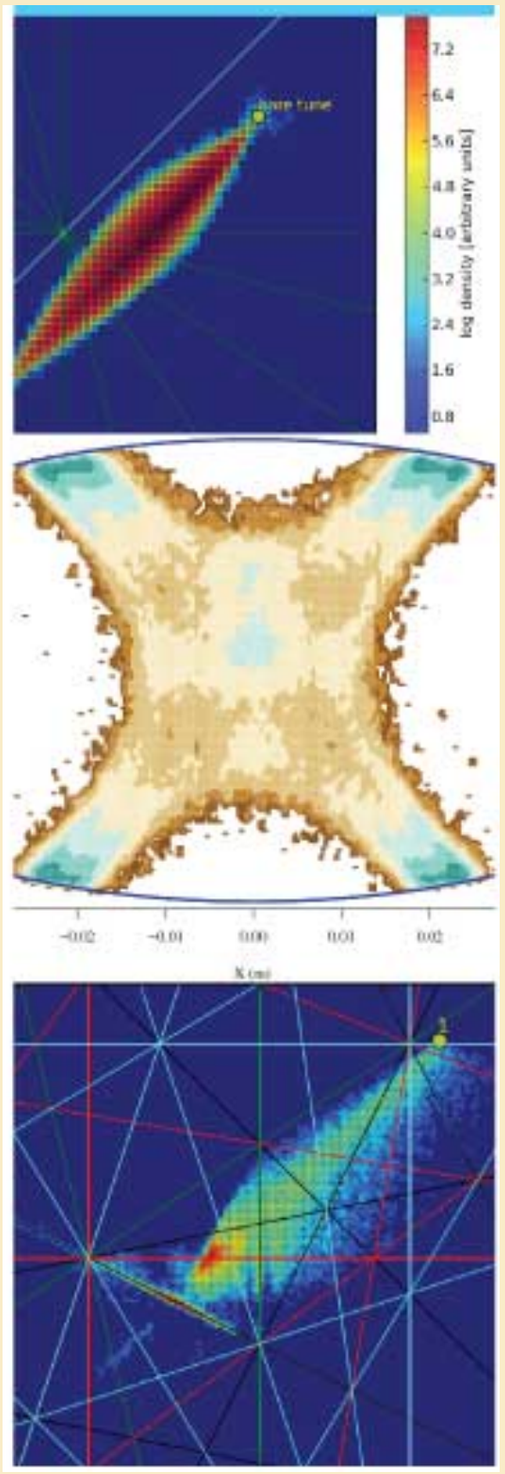
Fermilab is leading the ComPASS Project

Multi-institution collaboration of computational accelerator physicists

Developing HPC accelerator modeling tools

This and many previously mentioned projects part of SciDAC (Scientific Discovery through Advanced Computing)

Joint HEP-ASCR funding to advance the HEP mission by fully exploiting DOE SC leadership class computing resources



Conclusions

- The next level of discoveries requires the next level of computing
- Power, Scale, Efficiency, Usability, and Collaboration

The Fermilab Scientific Computing Division is involved in all aspects of the "next level of computing" activities

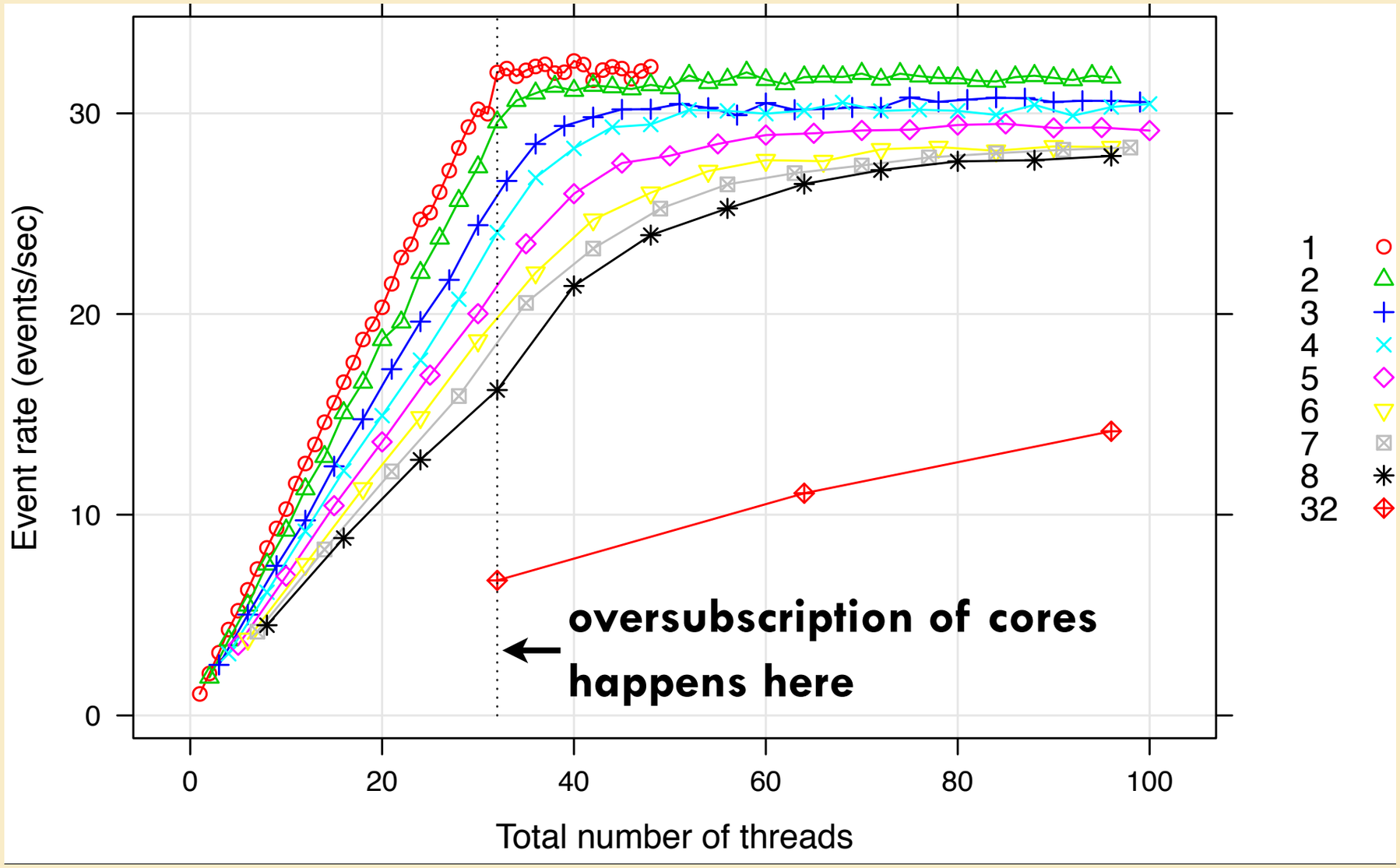
Preparing for the challenges of

- 14 TeV LHC running at CMS
- Present and future Intensity and Cosmic Frontier experiments
- Exploiting new technology for LQCD, HEP processing, and DAQs
- BIG DATA
- Providing easier and more powerful tools to collaborations of physicists

Large investments in software/techniques will be required to utilize the "next level" technology

Parallelization and sharing memory

Test NOvA reconstruction (in multi-core Art) on a 32 core machine



- One thread/job gives you maximal throughput, but no memory sharing
- Two threads/job has nearly same throughput, but thread pairs can share memory
  - So now you can fit that big 2 GB geometry DB into memory, since have 4 GB/job
- 32 threads/job – bad idea