

# Data processing in the wake of massive multicore processors

How our software infrastructure is evolving to meet  
changes in the large-scale computing environment

Jim Kowalkowski  
Fermi National Accelerator Laboratory

# Moving away from the simple commodity cluster

The computing environments that run our major event-processing framework software are changing. These frameworks have been evolving to accommodate this change. The changes need to go quite a bit further, to better handle coprocessors with alternative architectures.

Are we going far enough with our software framework and related infrastructure?

By the way: Heterogeneous computing is back

# Overview

- Reminders
  - A few reminders concerning the changes
  - Changes in computing architecture
- Software
  - Software development in this future context
  - Software frameworks on modern architectures
- Programming
  - Aspects of programming many-core
  - Already visible software issues
- Putting it all together

# A few reminders

# The changes are real

- What specially am I talking about?
  - The many-core processor, not the multicore processor.
  - The accelerator or **coprocessor**
  - The Nvidia GPGPU and the Intel Xeon Phi are the best examples out there

The computing challenge has been recognized nearly everywhere, as evident from the R&D groups that have popped up over the past couple of years.

From the frameworks perspective, the focus has been on multicore, which of course is an excellent start - but does it go far enough?

Concurrency Forum at <http://concurrency.web.cern.ch/>

Software Engineering, Parallelism & Multi-Core - Track 5  
Event Processing, Simulation and Analysis - Track 2

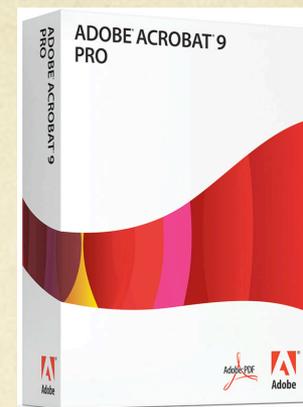
# The changes are fundamental

- The many-core computing change
  - Large core counts is primary focus
  - Slower cores with reduced memories, reduced operations, and much greater vector processing.
  - The larger leadership class machines have been headed in this direction, but also include multiple high speed interconnects.
- Even the simpler commodity-computing environment is changing.
  - ARM based server announcements.
  - Tablets replacing laptops and laptops have been replacing desktops.
  - Integrated CPU-GPU systems.

# Changes in architecture

# Mainstream Computing in HEP

- Heavyweight cores made life easy
  - Endless memory
  - High-level languages
  - Single architecture
  - Single OS (mostly)
- Are we spoiled?
- Must all good things come to an end?



# On the horizon



- Top 10 supercomputers –
  - Where is money being spent?
  - Who is driving change and innovation?
- Looks like they are moving towards a more useful measures of effectiveness: cycles of computing accomplished for the energy used.
- Note that the core counts are not all that exciting, as perhaps we expected a few years back.

Rank	Name	Total Cores	Processor	Mflops/Watt	Coprocessor
1	Tianhe-2 (MilkyWay-2)	3120000	Intel Xeon E5-2692 12C 2.2GHz	1901.54	Intel Xeon Phi 31S1P
2	Titan	560640	Opteron 6274 16C 2.200GHz	2142.77	NVIDIA K20x
3	Sequoia	1572864	Power BQC 16C 1.600GHz	2176.58	None
4	K computer	705024	SPARC64 VIIIfx 8C 2.000GHz	830.18	None
5	Mira	786432	Power BQC 16C 1.600GHz	2176.58	None
6	Stampede	462462	Xeon E5-2680 8C 2.700GHz	1145.92	Intel Xeon Phi SE10P
7	JUQUEEN	458752	Power BQC 16C 1.600GHz	2176.82	None
8	Vulcan	393216	Power BQC 16C 1.600GHz	2177.13	None
9	SuperMUC	147456	Xeon E5-2680 8C 2.700GHz	846.42	None
10	Tianhe-1A	186368	Xeon X5670 6C 2.930GHz	635.15	NVIDIA 2050

# An quick aside ...

- Note that our old friend the heavyweight multicore cluster node has not changed much regarding core counts.
  - Nothing greater than 16 cores per nodes on the top ten!
- Note that high-speed networking is important on the top ten. This is now affordable, even at speeds of 40Gb/s.
  - Does this mean anything for computing.
  - At this speed, is it cheaper to move things around the network than to store and move files?
  - Are there advantages to protocols outside TCP/IP over ethernet?

# Many-core: Available now

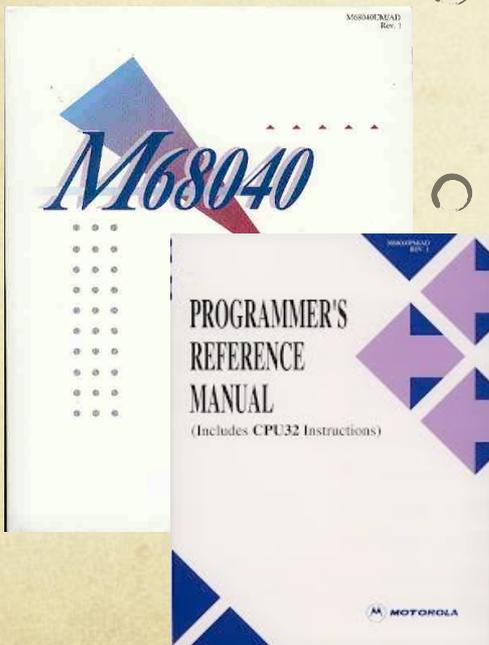
- Key features that impact software frameworks
  - Lots of cores and support for many threads
  - Low power given the computing resources
  - Low memory per thread
- Other key aspects
  - NVIDIA K20: unusual synchronized thread operation
  - Xeon Phi: Big 512 bit VPU registers

	Phi	K20
Power	~ 300W	~ 235W
Cores	60	~ 2500
Threads	$60 * 4 = 240$	$2048 * 13 = 26624$
Memory / thread	$8\text{GB} / 240 = 33\text{MB}$	32 regs + 24KB shared

Most interesting is how many-core affects event processing applications

# Past: Remembering days gone by

- Does anyone remember the era on the left?
- What was unique about it?
- Are the good times over or have they returned?
- Does anyone remember running these things without an OS?



# Software development in this future context

# Modern software on current cores

- What has the large-core processors brought us?
- Languages
  - Java, Javascript, PHP very popular
  - C++
    - still an interesting choice
    - what do many people outside of HEP say about it?
    - How well do people know it?
- Complexity
  - Performance is difficult to quantify, even with the extremely advanced tools of today.
  - Still difficult to estimate performance changes from measurements
  - Many interacting parts contribute to this

# Living within the future context

- What has our software infrastructure team been doing to accommodate these hardware changes?
- A clash of thinking
  - Have you worked with people that have been using complex parallel-capable machine for years?
  - What about optimizing the computational kernel after locating it?
- Still a mismatch for many-core
  - Irregular problems and workloads
  - Reduced flexibility changes programming patterns
  - Slow progress in showing reasonably good value

Is more code design actually going to be necessary?

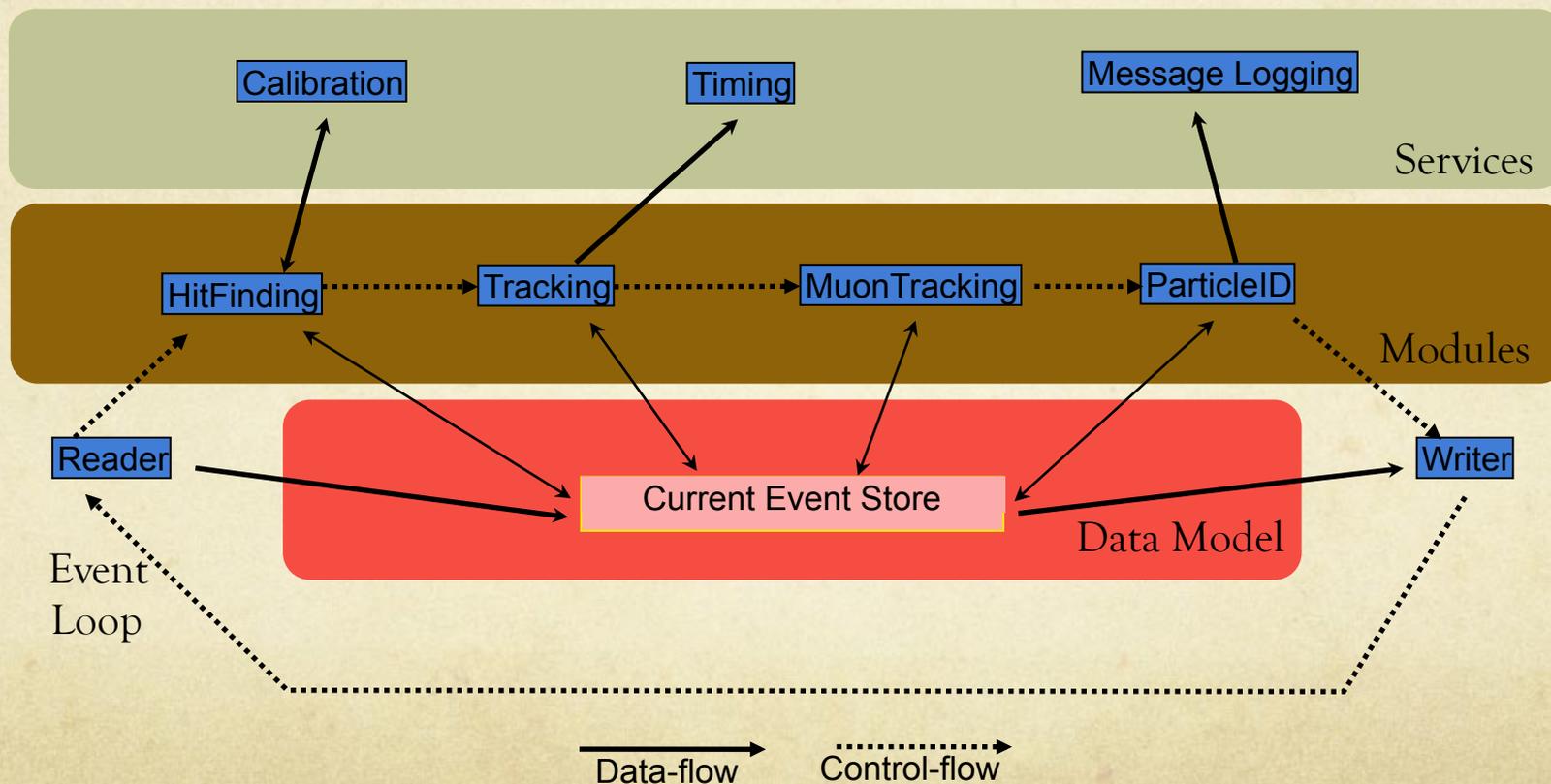
# Why all the excitement?

- Are massively parallel platforms really all that interesting?
- Software frameworks are used to define our applications
  - Organization and programming of these will be the my focus
- Key changes to software requirements to watch for
  - The Goldilocks effect: choosing just the right unit of work
  - Extra steps likely needed: ordering and summarizing partial results

# Software frameworks on modern architectures

# An essential organ: The standard framework

**Event Processing Framework:** Software that coordinates the processing of **collision events** by pluggable reconstruction, filtering, and analysis modules. Events and modules are independent. Modules add data to and retrieve data from one event at a time.

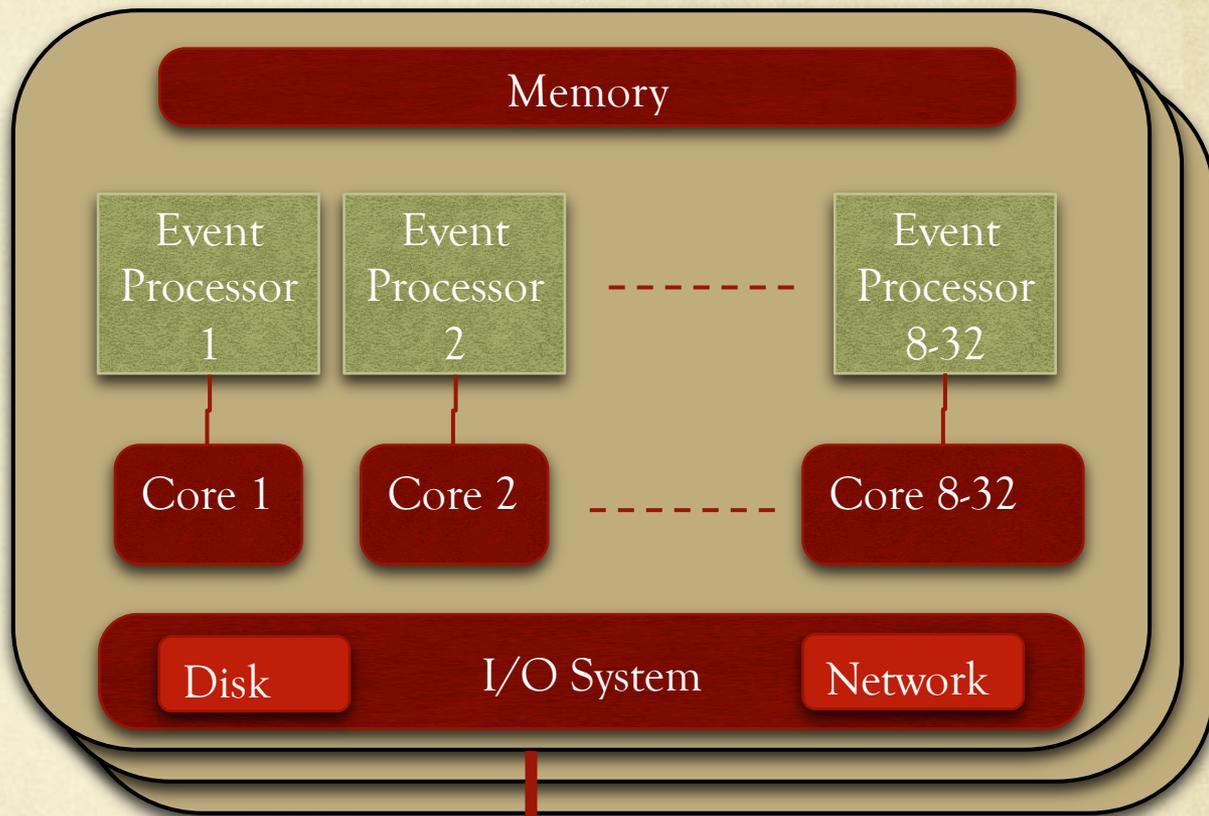


# Assumptions for this discussion

- Event time and size estimates from CMS
  - $\sim$  10-30 seconds for reconstruction
  - 1-20Hz processing rate for skim jobs
  - 1MB event size during reconstruction
  - 150KB size for AOD
- Assuming that the processing times will be 10x bigger in next run
- Bandwidth estimates based on these numbers and only on input data rates.

# Let's start from where we are now

- Same old organization
  - Serial processing of events
  - One processor per core
- New
  - Rules for making thread-safe module code
  - Removal of global variables
  - Addressing global service issues



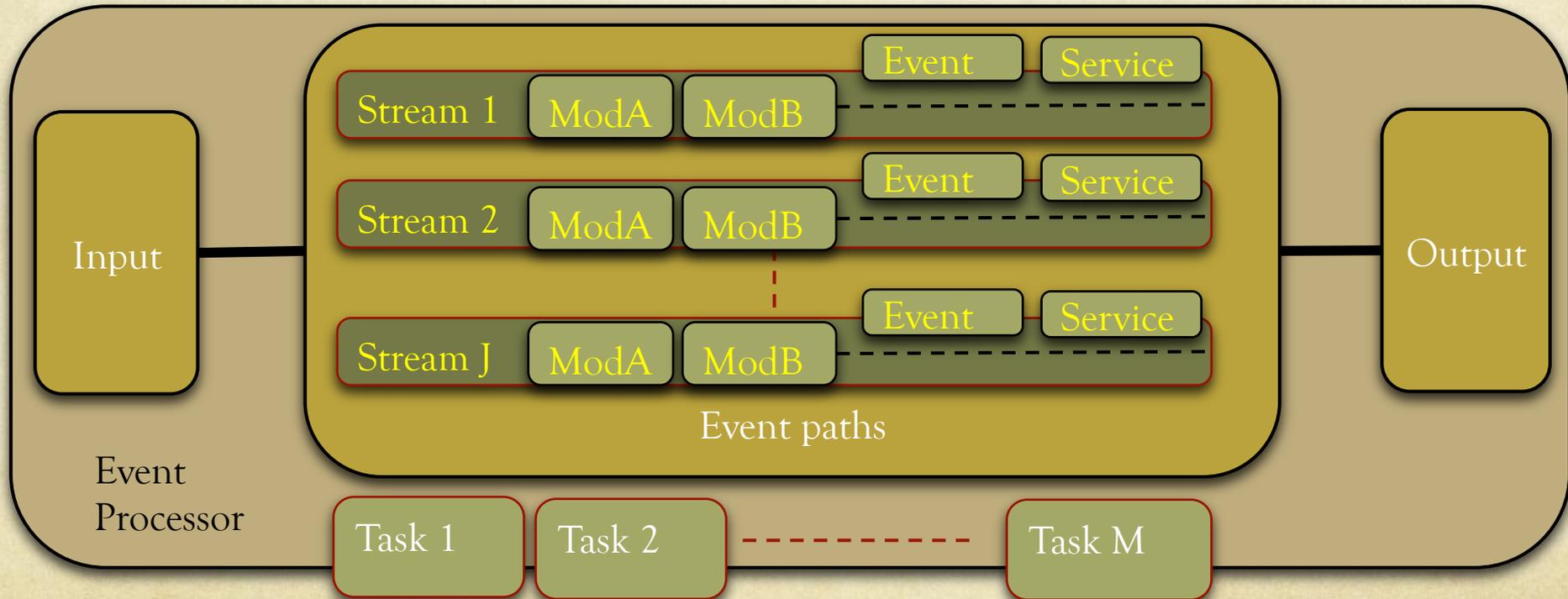
$$3\text{MB/s} * 32 = 96\text{MB/s (AOD)}$$

$$100\text{KB/s} * 32 = 3.2\text{MB/s (Reco)}$$



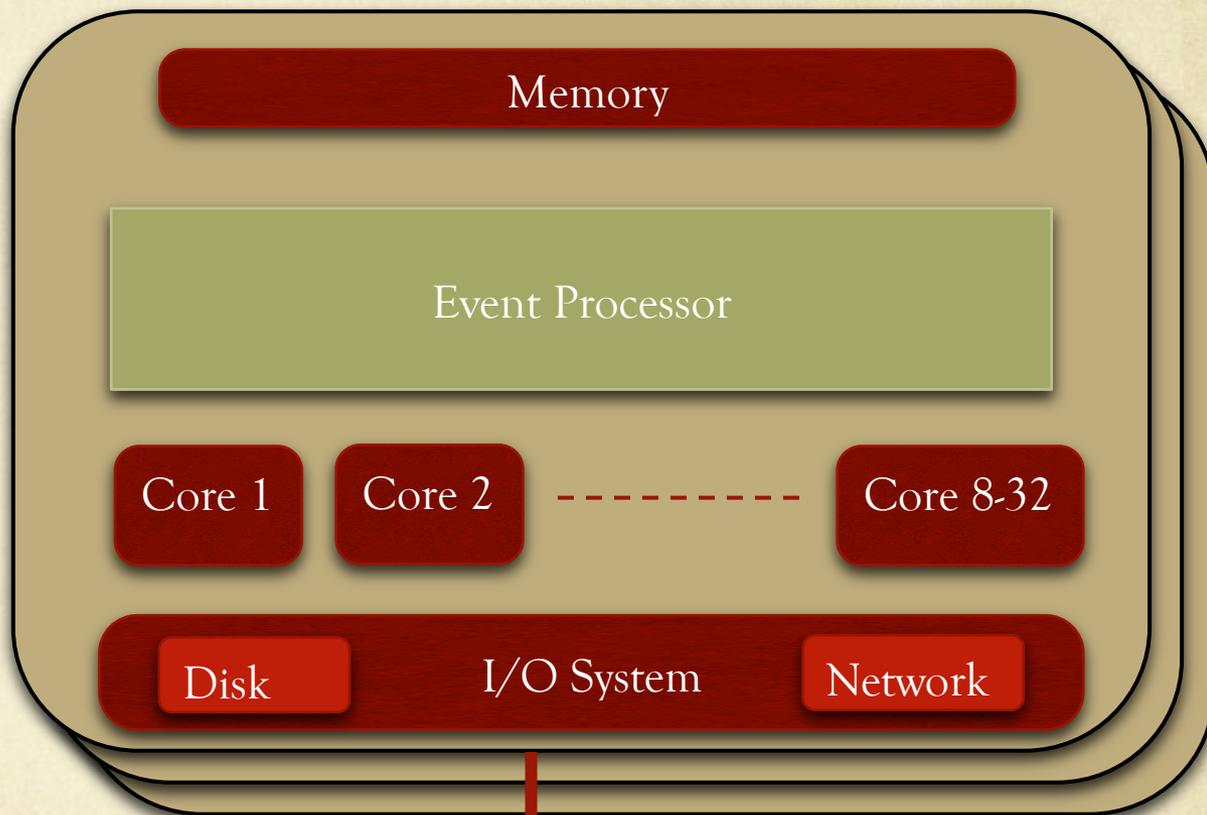
# Framework reorganization

- Key principle: Work is assigned to tasks, and tasks are scheduled to run on cores.
- Intel Threading Building Blocks (TBB) operates like this and is commonly used
- Parallelism at the event, module, and algorithm level all available
- Framework schedules streams and modules, algorithms can generate more work



# Near future framework

- Focus on multi-core heavy-weight “serial cores”
- Some goals
  - Reduce memory use
  - Maintain processing efficiency
  - Reduce run latencies
  - Allow multi-threaded algorithms
  - Multiple active events



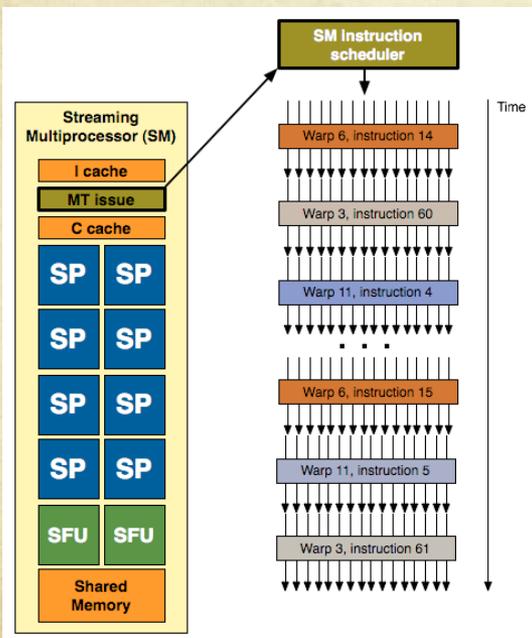
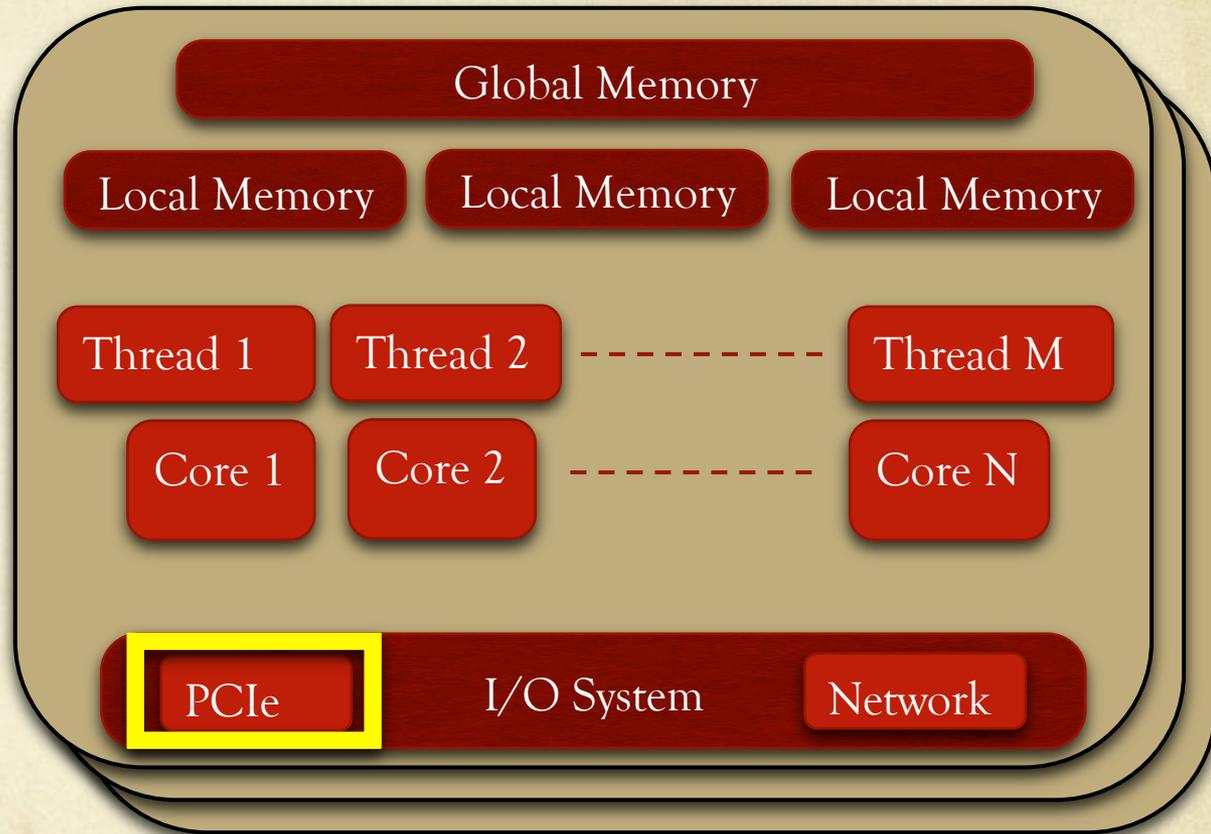
$$3\text{MB/s} * 32 = 96\text{MB/s (AOD)}$$

$$100\text{KB/s} * 32 = 3.2\text{MB/s (Reco)}$$



# The many-core coprocessor

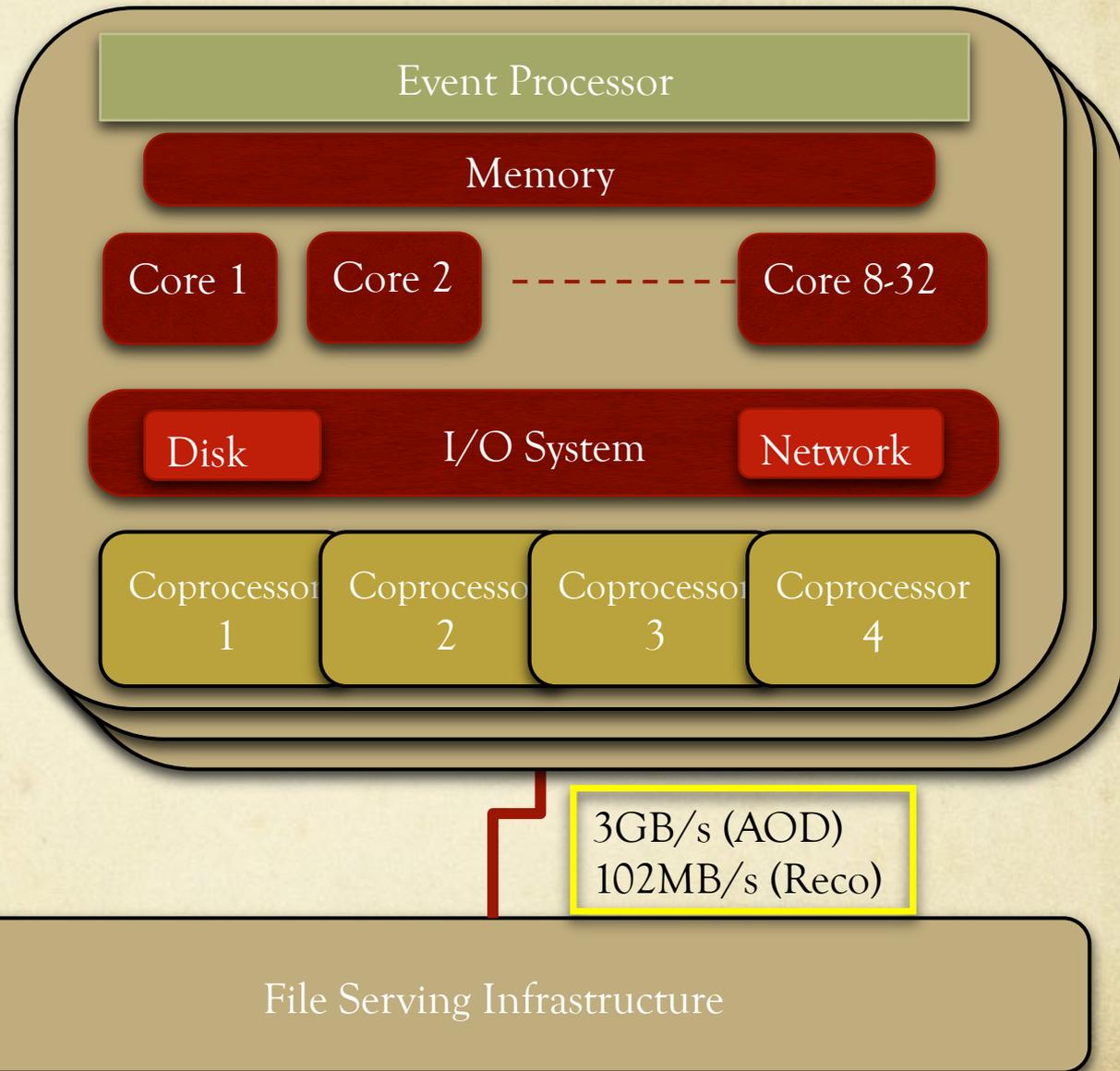
- Small memories
- Simple cores
- Data transfer costs
- Lock step operations
- Specialized languages



	Phi	K20
Cores	60	~2500
Threads	240	~26624
Memory / thread	~33MB	32 regs+24KB shared

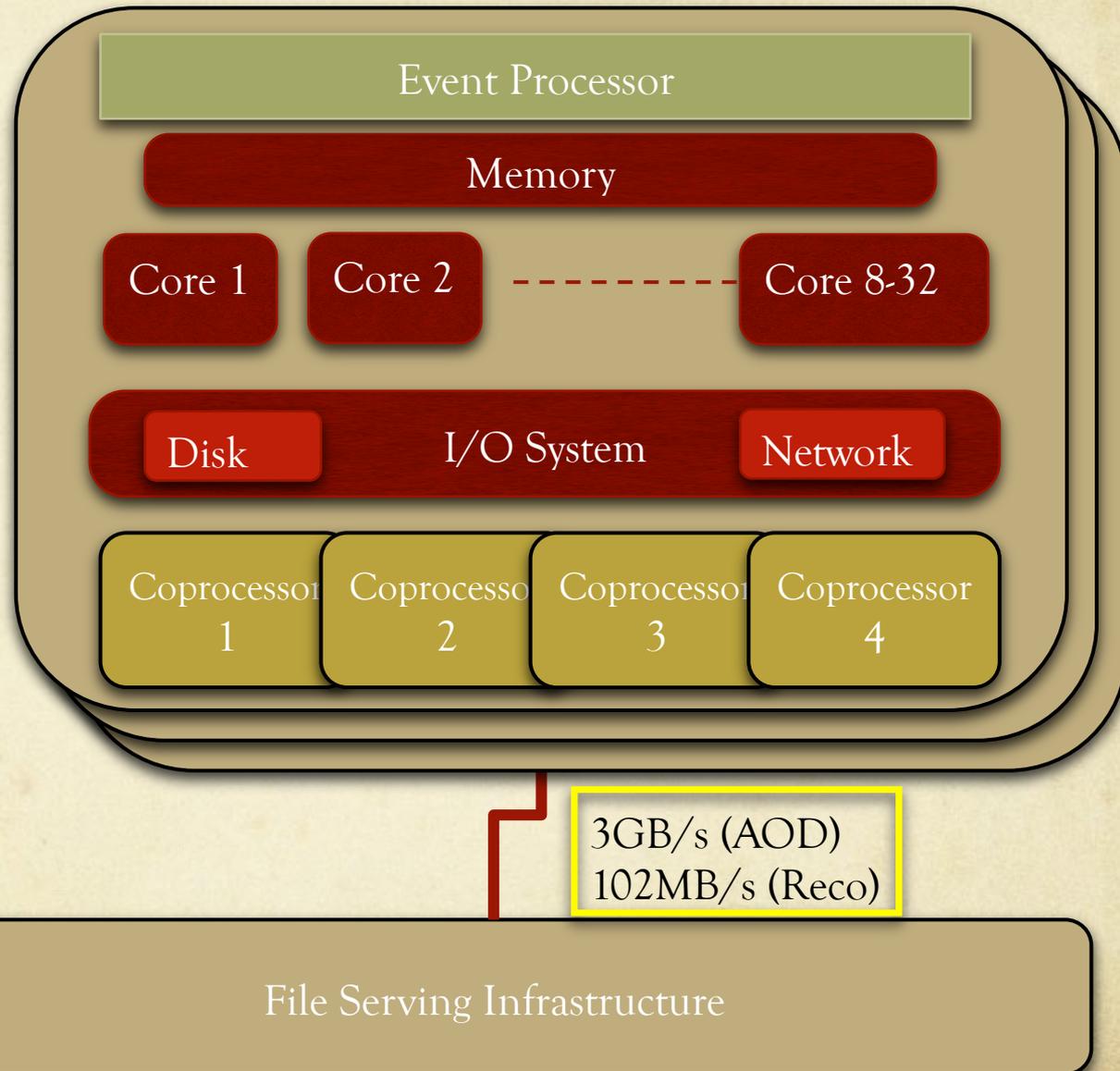
# Where are we heading?

- 10x compute capacity?
- Low overhead scheduling and coordination essential
- Is the bandwidth increase an issue?



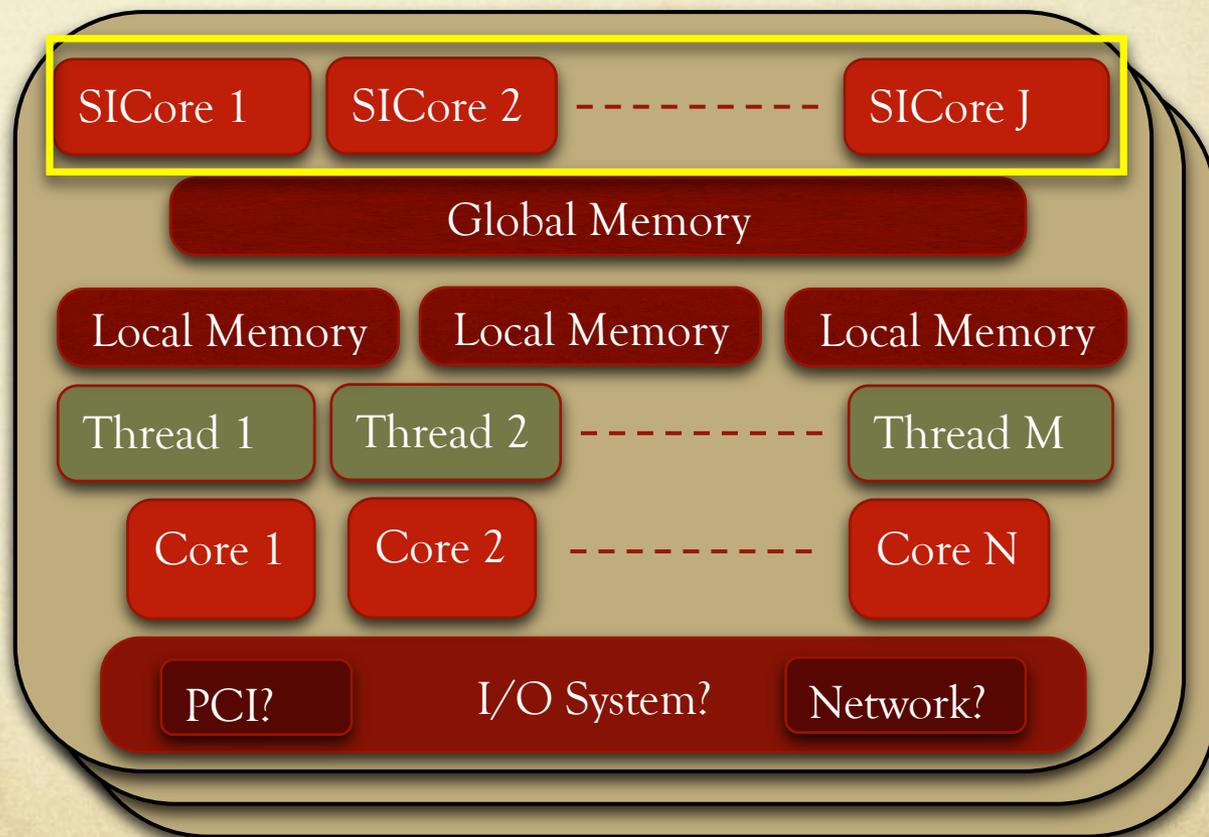
# Where are we heading?

- Now a **distributed framework**?
  - Coordination of separate memories
  - EDM and data sharing affected
  - Load balancing difficulties
  - **Must not overload serial cores**
  
- Is this an intermediate step?



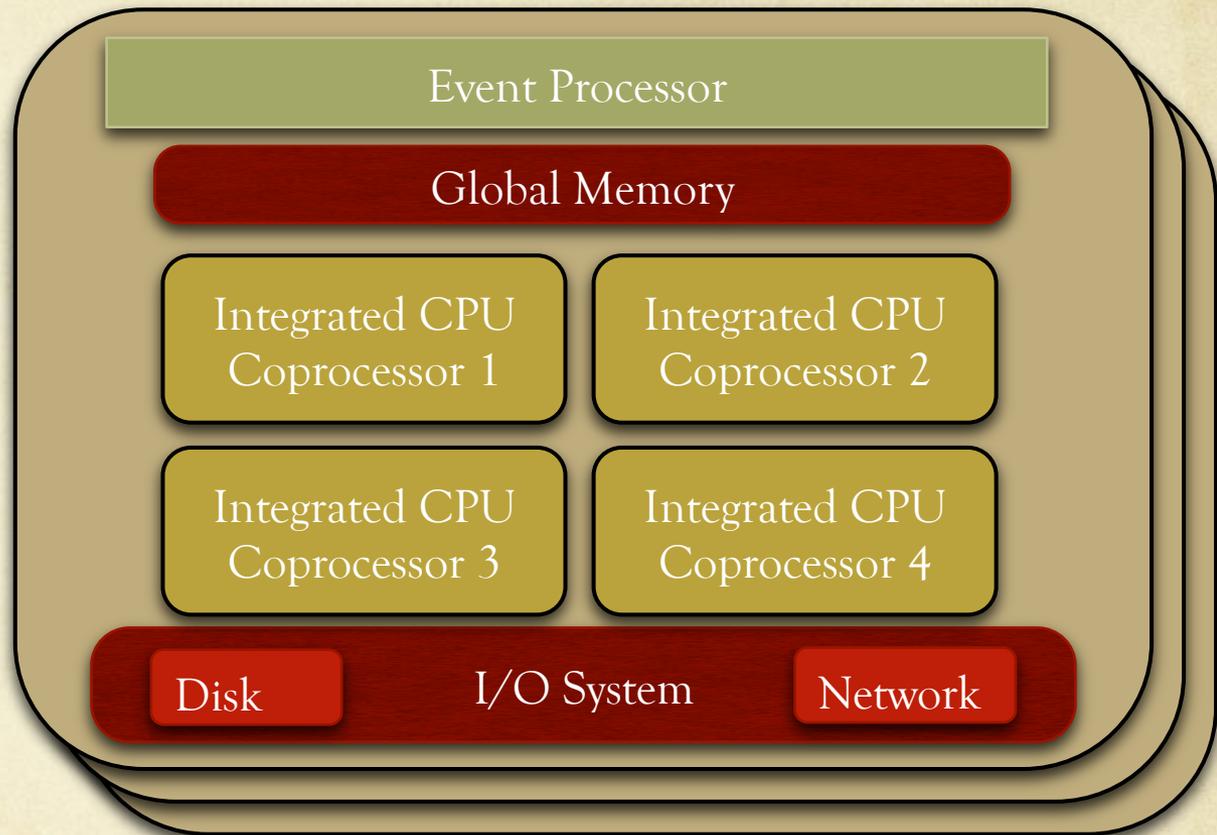
# Is this still a coprocessor?

- SICore = serial, integrated core
  - ARM?
  - PowerPC?
- Shared memory
  - Low overhead communications
  - Single address space possible
  - Usual coordinated access needed
- **A Hybrid Processor**



# Not that far away

- What can we do this something like this?
- Are files still useful?
- Many options for event processor application
- Is 20x - 40x increase plausible?



$$3\text{GB/s} * 4 = 12\text{GB/s (AOD)}$$

$$102\text{MB/s} * 4 = 408\text{MB/s (Reco)}$$



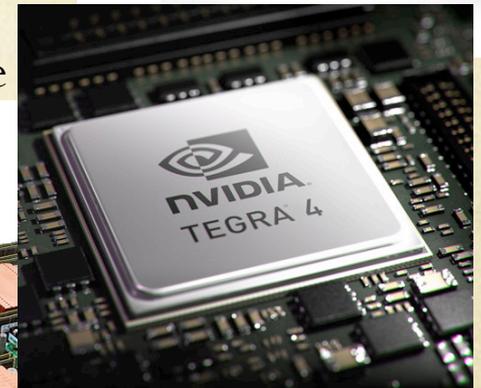
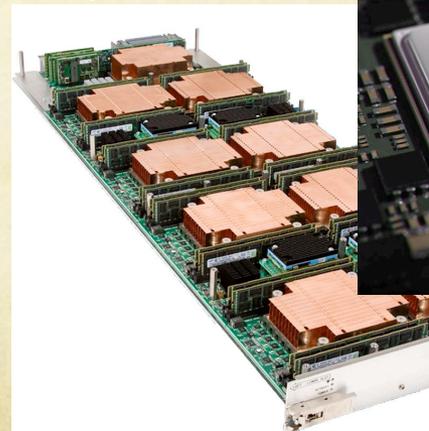
# These things already exist

- NVIDIA Tegra 4 (hand-helds)
  - 72 GPU Cores
  - Quad-core ARM
- Dell Copper
  - 4 quad-core ARMs per sled
  - 12 sleds per chassis
- Cray XC30 blade
  - High density
  - multicore, GPGPU, MIC
  - Pick and choose
  - High-speed interconnect

Dell Copper



Cray XC30 Blade



NVIDIA Tegra

# Programming many-core

# Programming aspects

- Language extensions and special directives are needed
  - Chose CUDA
  - Will use the Intel tools: Cilk Plus, intrinsics, pragmas
- Our development tools have evolved to incorporate many-core
  - CUDA compilation is in our build suite.
  - Integrated TBB into our tool chain
  - Xeon Phi not far behind
- Multiple architecture builds (again)
  - Cross compiling necessary
  - More than one algorithm implementation likely
  - Complicates testing and validation
  - Two or more kinds of compiled code available at run-time
  - Moving to new externals requires more synchronization

# More programming aspects

- Programming the many-core machines is getting better
- Dynamic parallelism and Hyper-Q
  - **Hyper-Q** = many processing streams active at the same time
  - **Dynamic parallelism** = GPU functions can start more GPU functions without returning control to host
  - Allows for more than one independent activity on the GPU
- Overlap of transfers and computation boosts performance
  - Hides the transfer latencies present in the current series of processors

# Affected framework areas

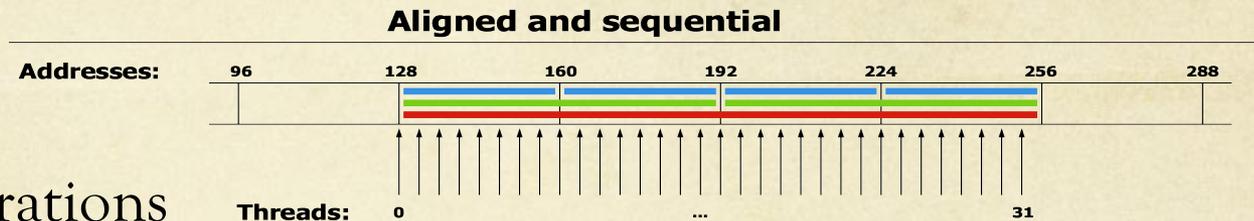
- Current HEP cluster defined as High Throughput (HTC), not High Performance (HPC)
  - Increase in node capabilities will stress the I/O systems
- A more distributed framework will affect the
  - Scheduling of work (TBB will help, data dependencies more visible)
  - Event data model (EDM delayed reading or loading is troublesome)
  - Configuration (boundaries and constraints more important)
  - Services (global will need to be eliminated)
- Larger scale distributed reductions (sums and aggregations) will be necessary
- Handling synchronization points such as runs and other event groupings become more complex

	1 core	32 cores	256 cores	512 cores	>1024 cores
150KB @ 20Hz	3MB/s	96MB/s	768MB/s	1.5GB/s	3GB/s
1MB @ .1Hz	100KB/s	3.2MB/s	25MB/s	51MB/s	102MB/s

Summary of bandwidth guesses

# Already visible software issues

# Issues that are already visible



- Memory considerations
  - Bank access patterns visible in code (must understand)
  - Calculating values better than caching results
  - Can quickly become the bottleneck and source of bad performance
  - Small tweaks in access pattern can yield wildly difference results
- Task model for parallelism has been good
  - Common direction (layers easily cooperate)
  - Can avoids locking
  - Design decisions on data structures and algorithm decomposition become more critical than ever

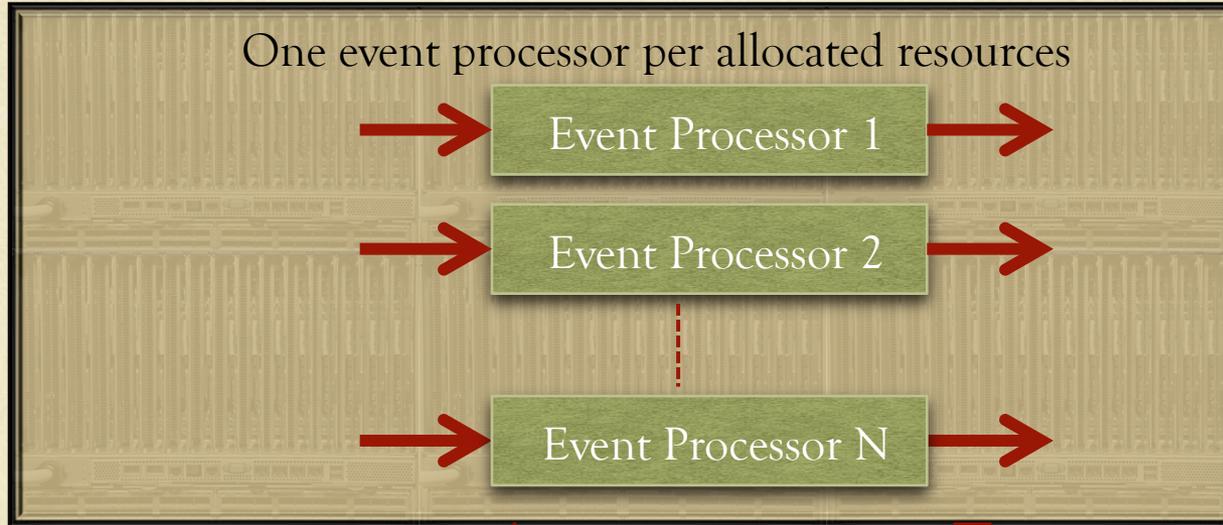
# Issues that are already visible

- Mixed execution locations needed
  - For algorithm phases
  - For modules
  - **Heterogeneity is back** and will be a big issue
  - Unclear how much visibility and control the framework will have here
- Unclear how to keep a balanced load: many run-time parameters
- New methods of collecting summary data and diagnostic information from algorithm phases will likely be needed
  - Conditional logic troubles
  - Deviations in work load problems
  - Temporary storage of extra data issues
  - Making it widely available concerns
- What is good for the lighter weight processors has also been good for the heavier weight processors

# Pulling it all together

# A possible overcome

HPC-HTC  
hybrid  
server  
cluster



Framework  
configuration  
and functions  
for job now  
span specialized  
resources

Allocate,  
Run,  
Monitor,  
Obtain results



Source  
data  
push

Results  
gather

Essential  
infrastructure  
services



Big Data management: Commercial server cluster

# Summary

- A mixed computing environment is inevitable
  - Heterogeneous within and among resource types
  - Job specification / configuration split across resource types
- Changes will affect our I/O subsystem
  - Focus on simpler streaming methods
  - Leave file and catalog handling to special-purpose nodes
  - Newer network protocols and hardware may help
- Changes in algorithm development strategy
  - Began a while back
  - Accommodate all types of many-core and multi-core

# Conclusion

- Movement in industry and research towards
  - Increasing performance through specialization
  - Keeping operational cost constant (or lower)
- All of this means more independent parts within our software infrastructure
- To keep up, we need continued and increased cooperation
  - Between physics laboratories
  - Between HPC communities