



Future Computing Challenges

Graeme Stewart

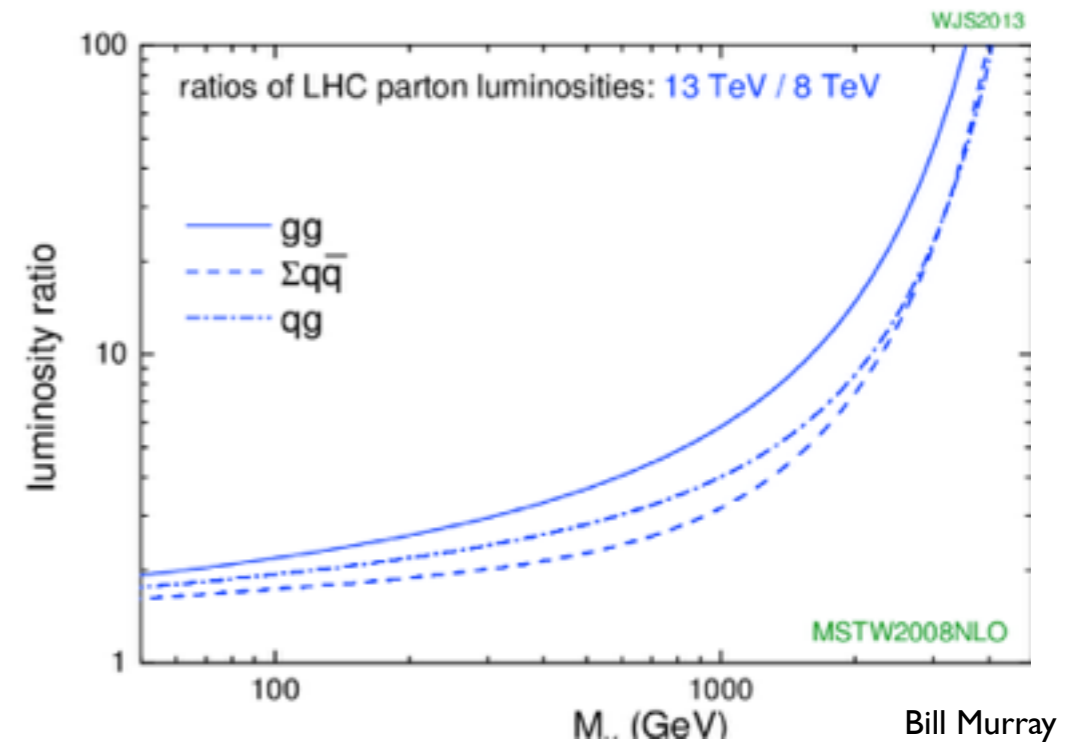
Overview

- Challenges from Run 2
an towards HL-LHC
- Evolving Processor
landscape
- I/O, Storage and the
Grid
- Scaling up performance
- Rethinking Algorithms



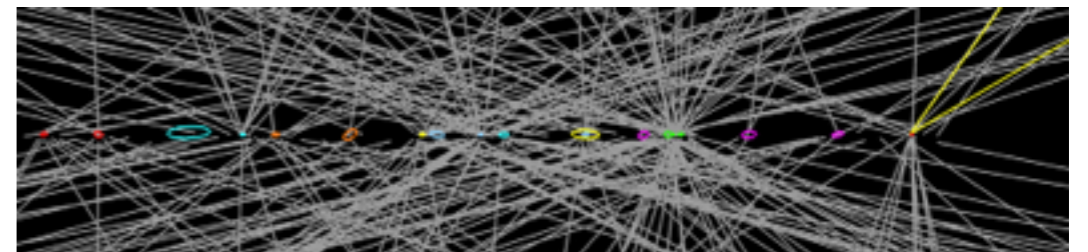
Run 2

- LHC Run 2 will bring
 - Increased centre of mass energy (13TeV)
 - More interesting events with enhanced physics reach
 - Higher luminosity
 - More interactions per bunch crossing (μ)
 - Higher detector occupancies and track multiplicities translate directly to longer time to reconstruct events
 - 25ns running will allow good integrated luminosity with pile-up more under control, but the scaling is still worse than linear
 - Higher trigger rates into the bargain



Enhancements from 13TeV/8TeV:

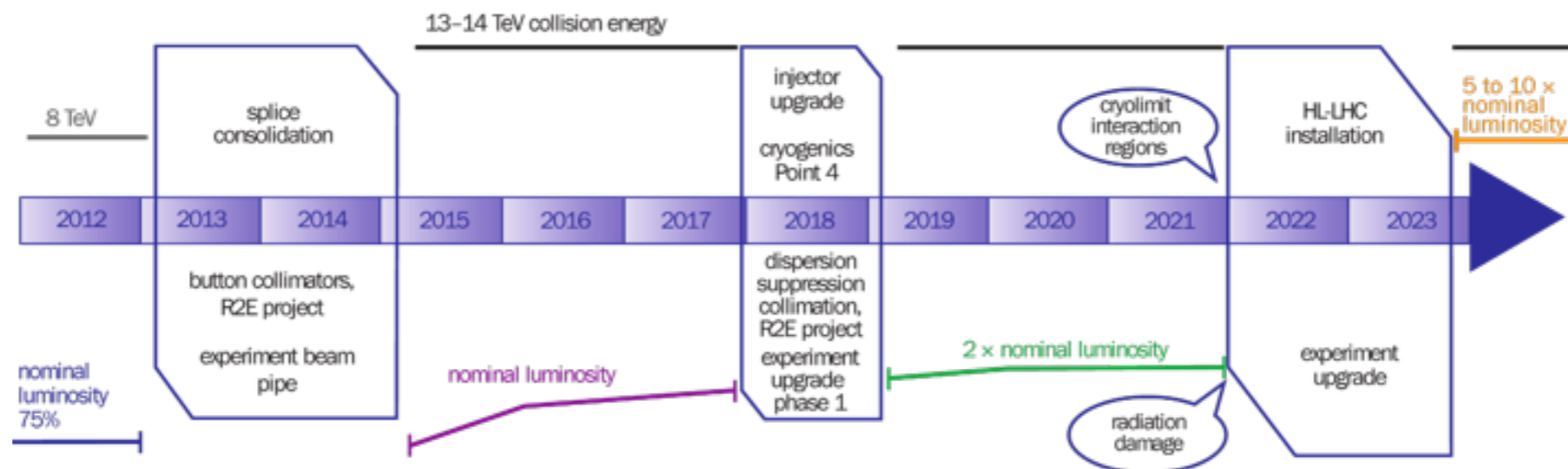
- Factor 100 for objects of mass 3.5-4TeV
- Factor 10 for mass around 2TeV;
- Factor 2 for 100GeV objects



- Run 2 pile up likely to peak close to 50

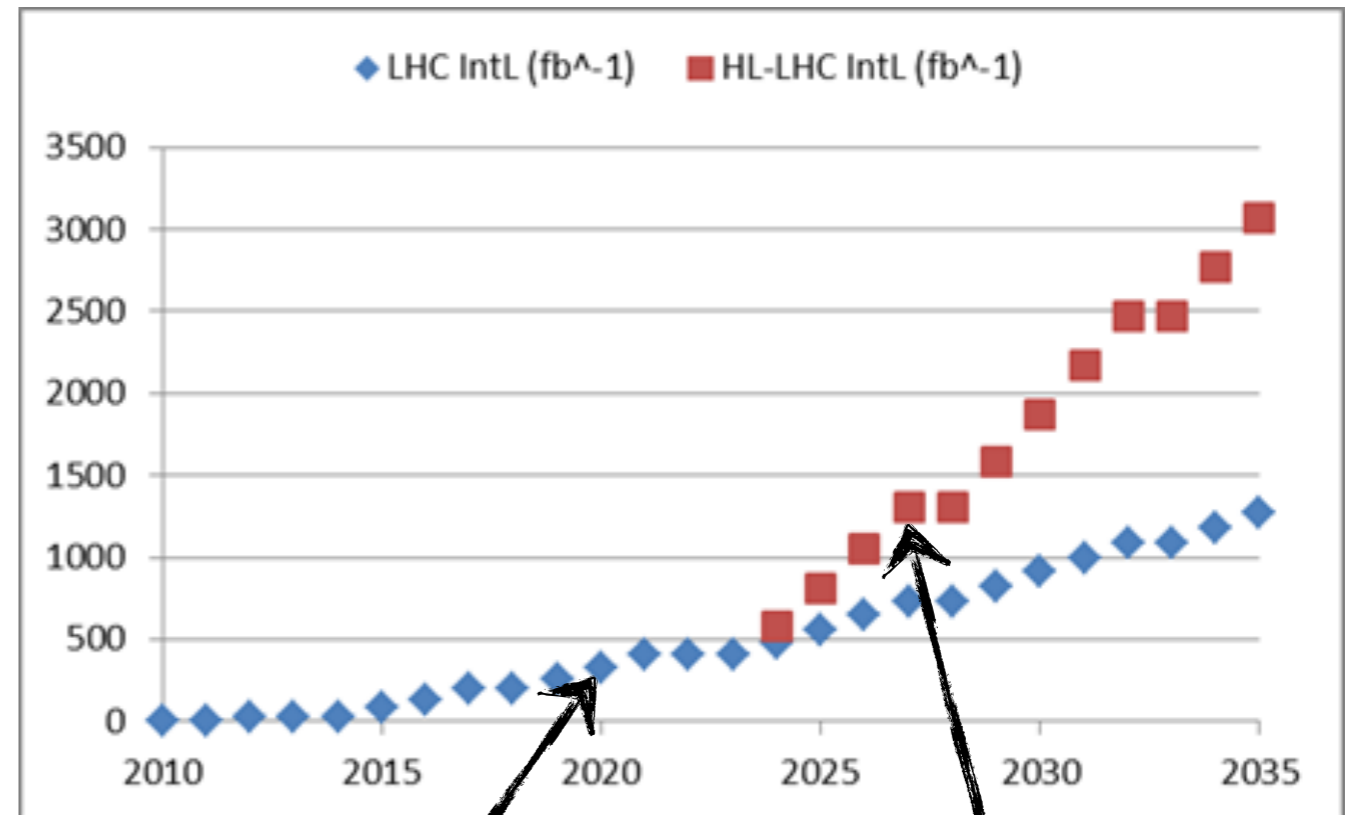
Towards HL-LHC

- Beyond LS1 and Run 2 LHC continues to deliver more luminosity
- Arriving at HL-LHC after LS3
 - New injection, upgrading inner triplet magnets, crab cavities, ...



Goals of HL-LHC

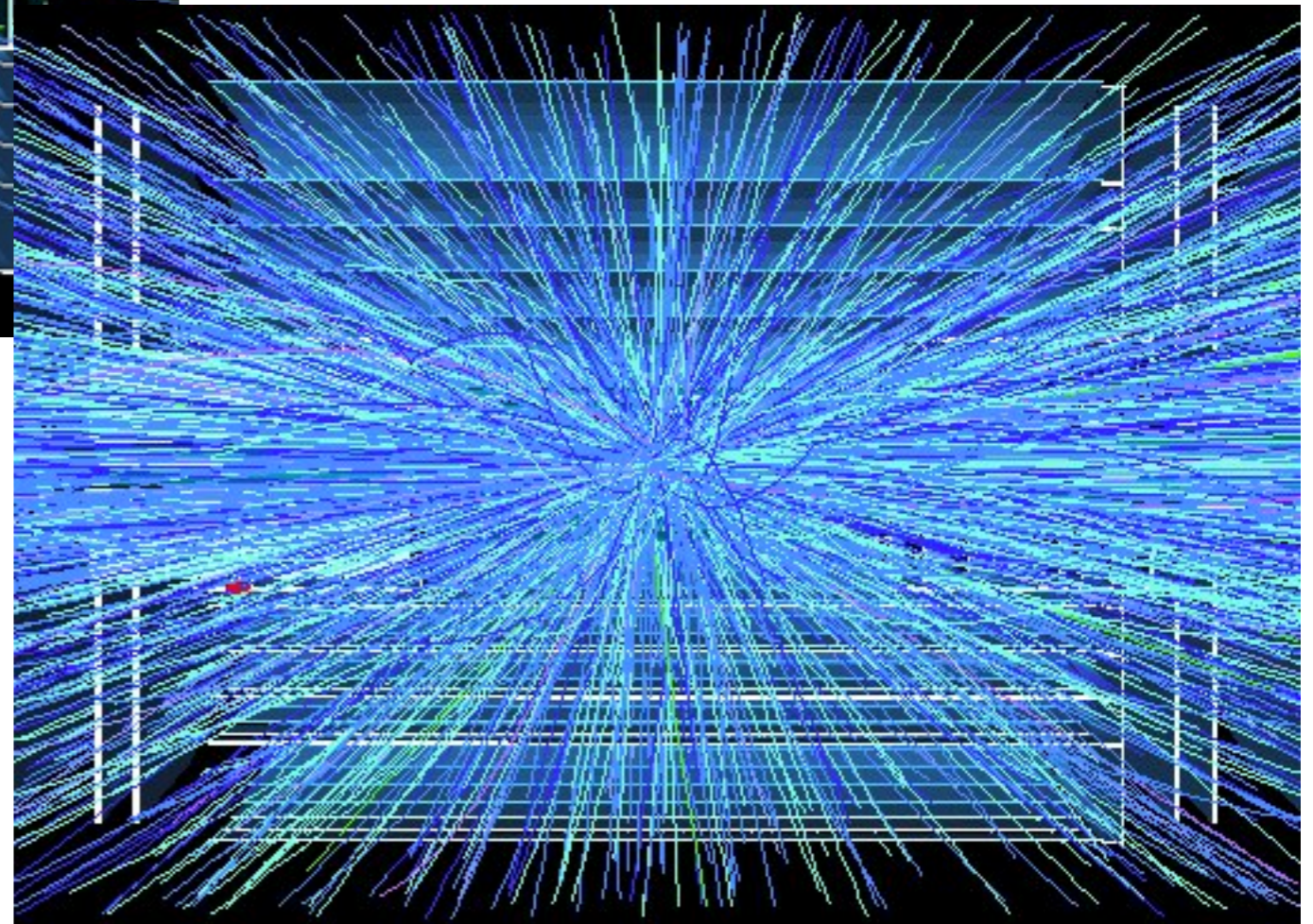
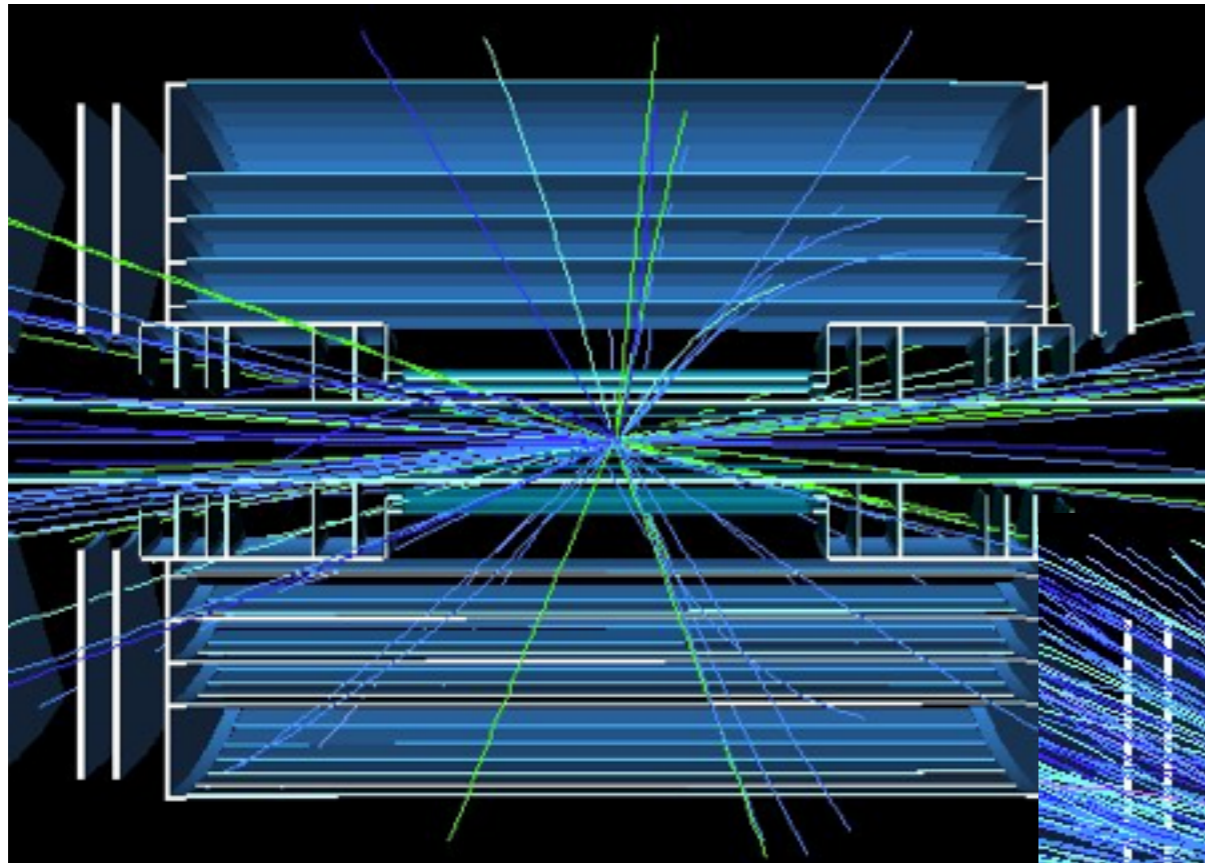
- Precision measurements of Higgs properties requires high statistics of very rare processes
- Goal of HL-LHC is to provide 3000fb^{-1} by 2030(ish) at about 300fb^{-1} per year
- 200 signal events for $t\bar{t} \rightarrow H \rightarrow \gamma\gamma$ will measure top/Higgs coupling to 10%



Vanilla LHC

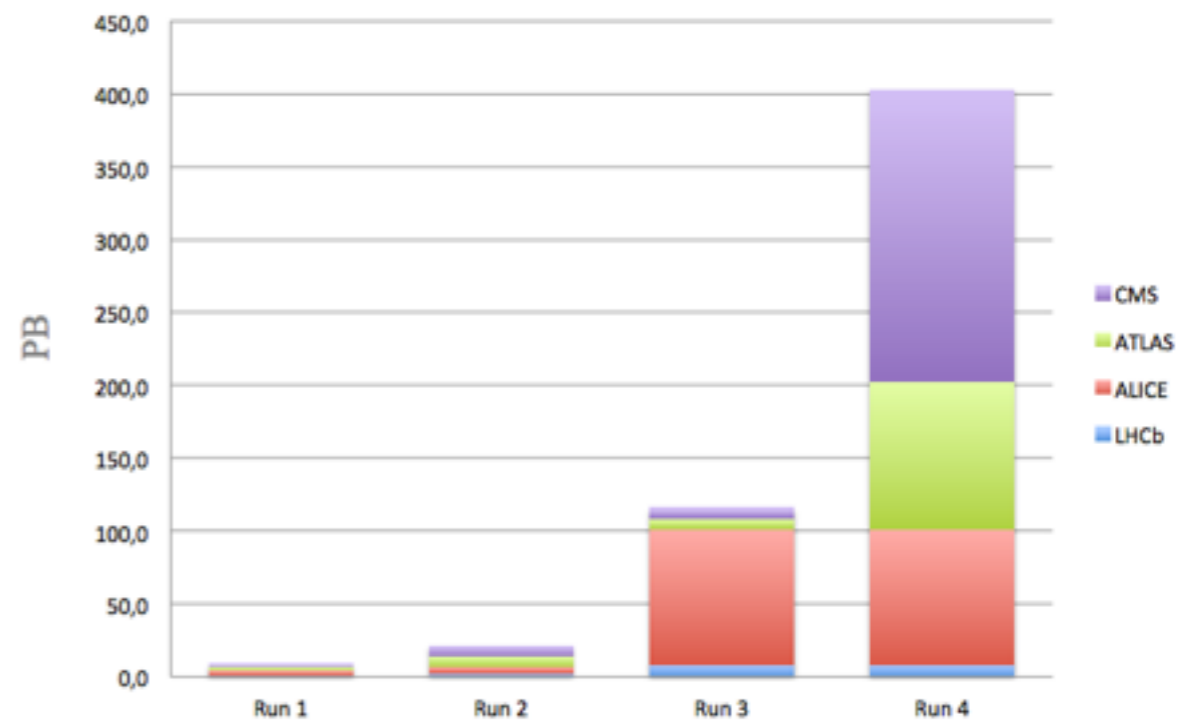
HL-LHC

HL-LHC: The Challenge



HL-LHC In Numbers

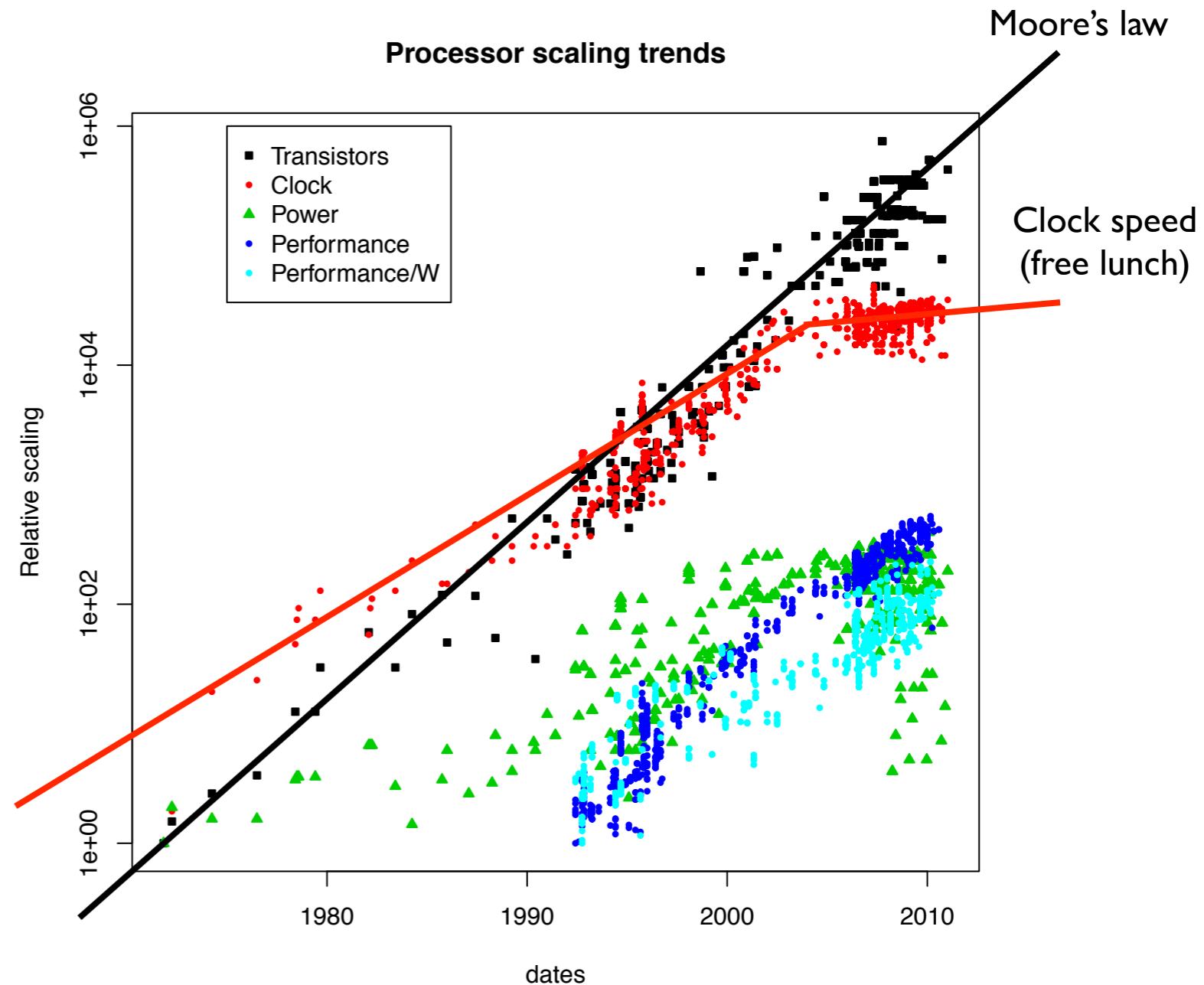
- Pileup likely to be about 150 instead of $\langle\mu\rangle = 25$ in Run 1
 - Exceeding difficult conditions for tracking
- Readout rate will be x10 higher than Run 1
 - So data rate will be much higher
- Storage, archive, processing loads go up



Total RAW data recorded

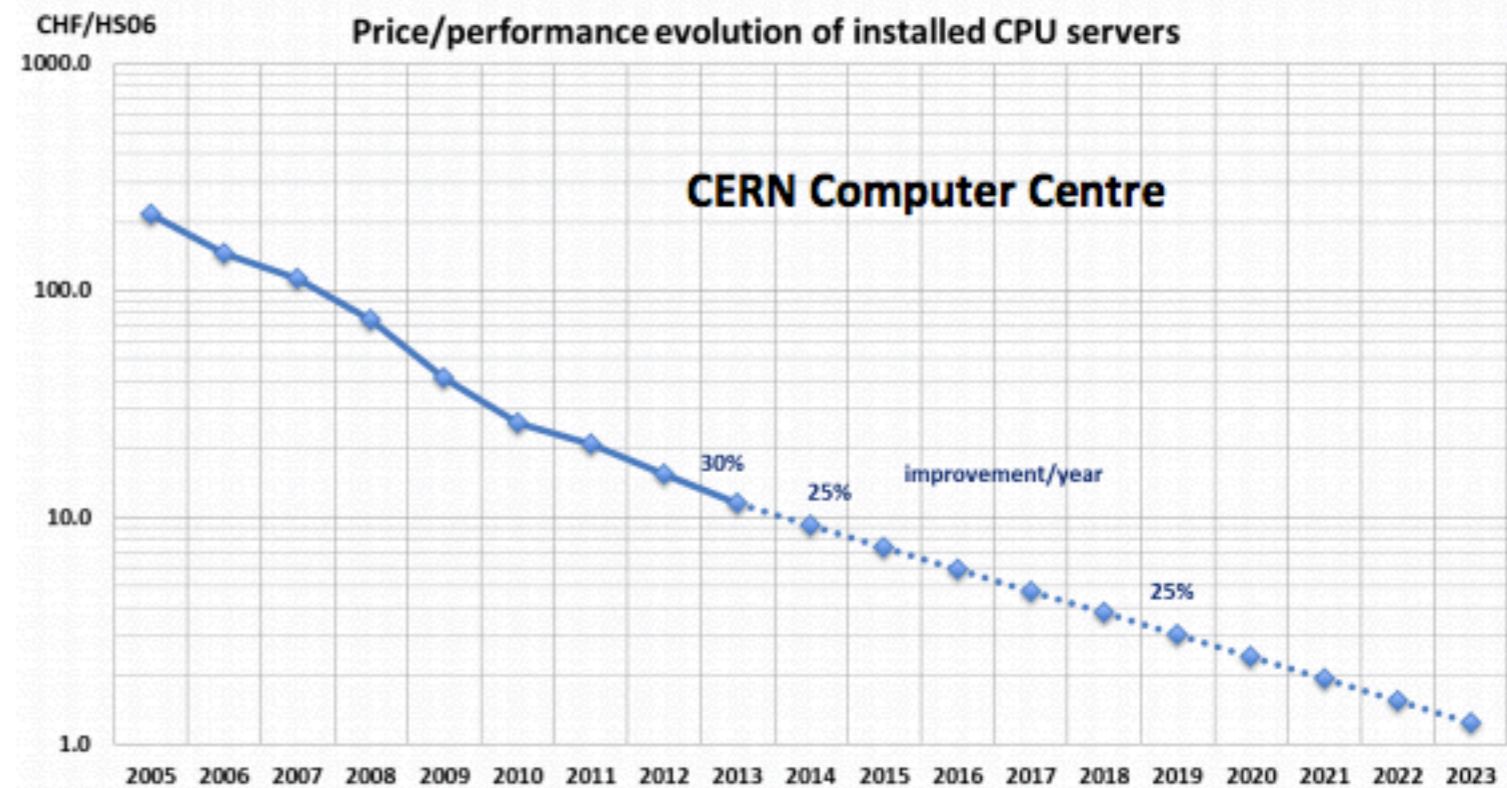
Processor Landscape

- Moore's law - alive and well
 - This means doubling time of transistors on a chip stays at about the same (2 years)
 - But this does not translate directly into the price/performance of servers
- Clock scaling stalled a long time ago



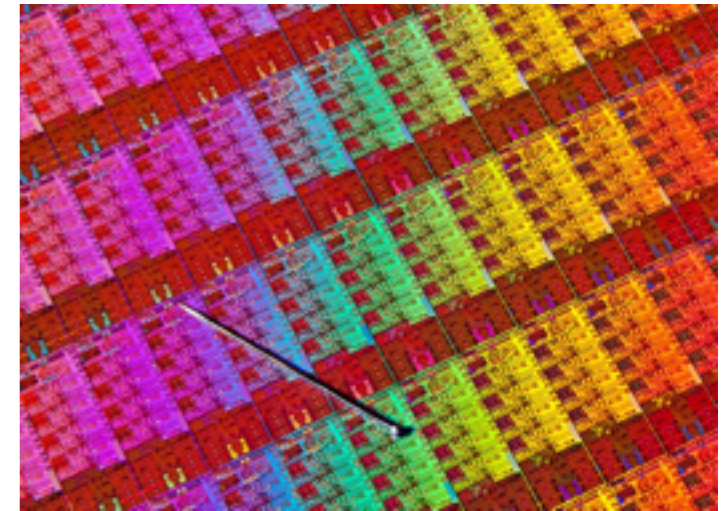
Server Cost Trends

- Translating Moore's law into a built server box we see about 25% improvement per year
- N.B. $\pm 5\%$ per year would lead to factor 1.63 in 10 years

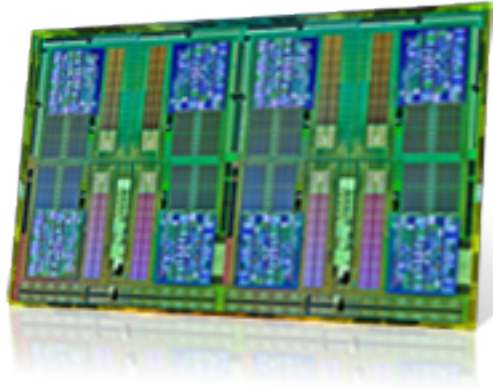


- In addition CPU architectures are becoming more difficult to exploit over time

Xeon server cores



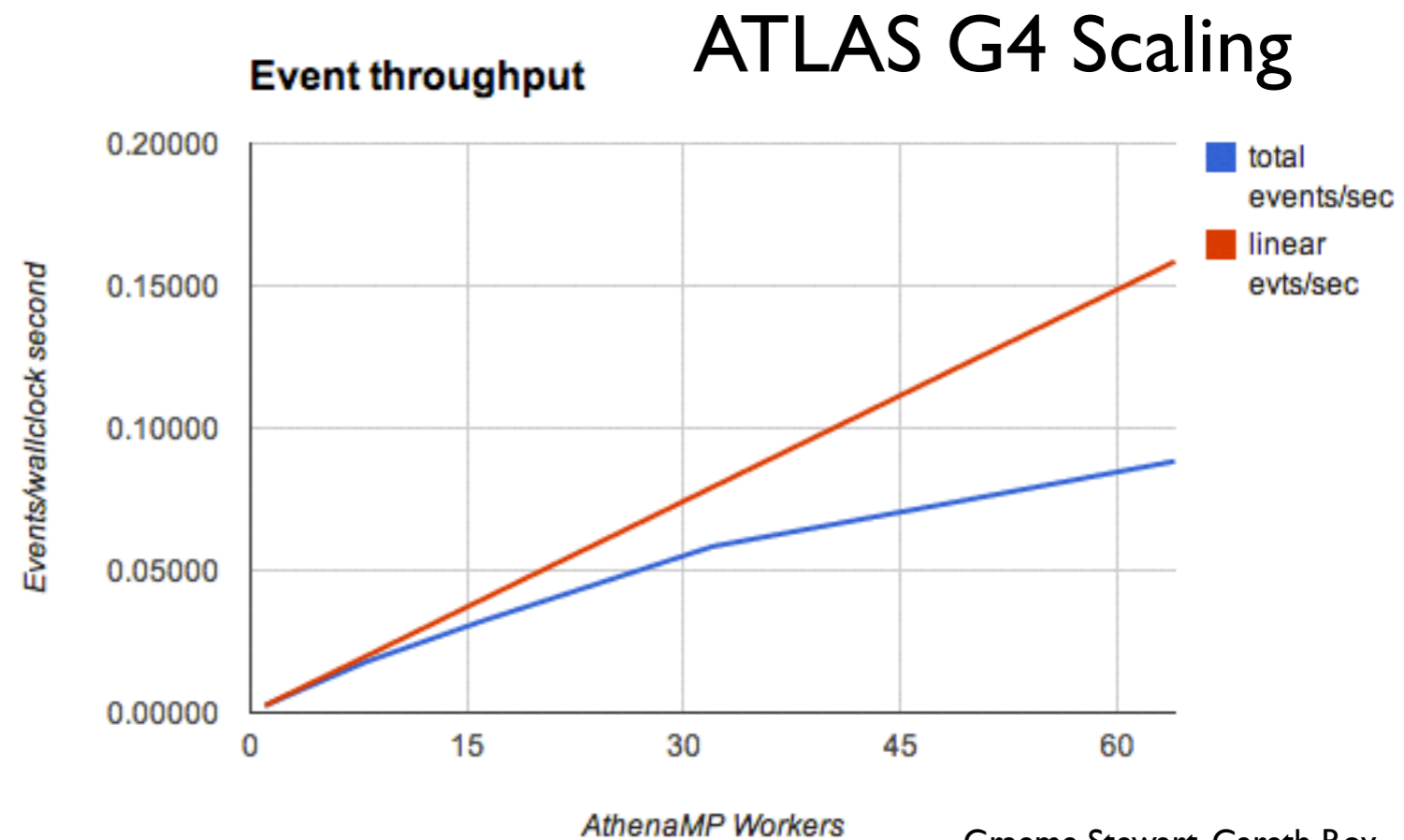
- Today's grid workhorse
- Continuing to improve
 - Intel sticking to tick/tock model of development:
 - *tick* - shrink the process size
 - *tock* - update the microarchitecture
- Skylake with 10nm process (aka Cannonlake) is ~2016
- No sense that there is a serious scaling limit on this architecture, but...
 - Vector registers will get wider
 - Gentle increase in core count (linear)
 - Memory?
 - As much as you can pay for



AMD



- Yes, they are still around
 - Probably in long term withdrawal from the server market
 - We see many of the same problems we'll see on Intel architectures with a large number of weaker cores
- Speculation seems to hint that they will move more into 64 bit ARM servers as well as maintain GPU interests
 - (This is definitely a bad situation - no real challenger to Intel)



AthenaMP Workers

Graeme Stewart, Gareth Roy

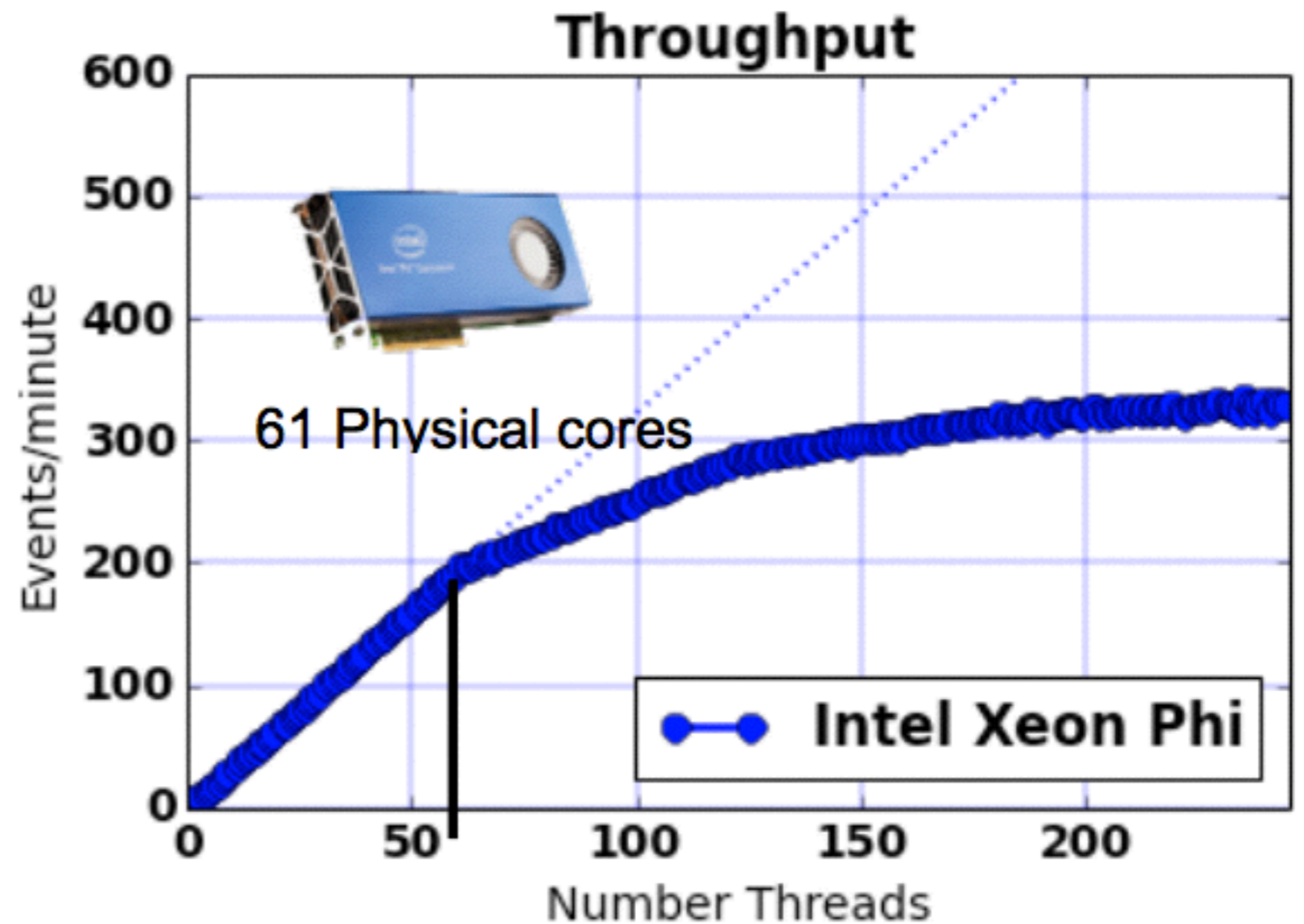
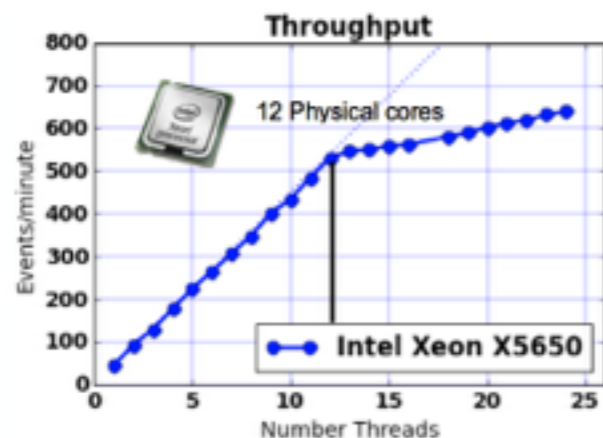
Intel Many Integrated Cores



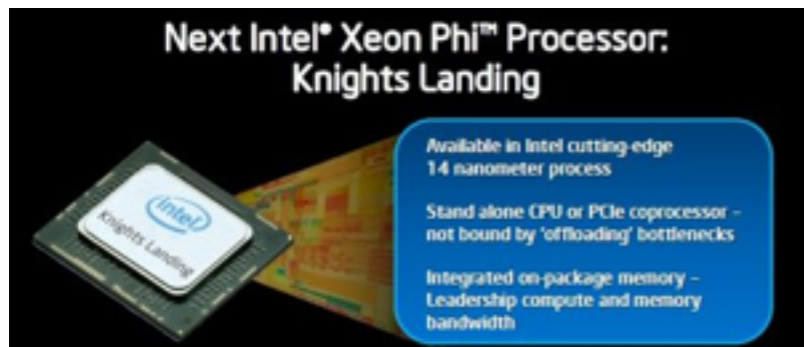
- Intel's Xeon Phi (aka Intel MIC) is in its first generation
 - 61 x86_64 cores @ ~1GHz
 - 16GB of memory
 - Coprocessor architecture
 - Cache coherent, but no out of order execution
 - 512 bit registers (8 double or 16 float)
- Memory per core: 256MB
 - Maximum performance needs 4 threads per core:
64MB per thread

MIC Performance

- Results from Geant4 running many parallel threads on the MIC
- Lower throughput than Xeon server, but more bang per buck



Andrea Dotti



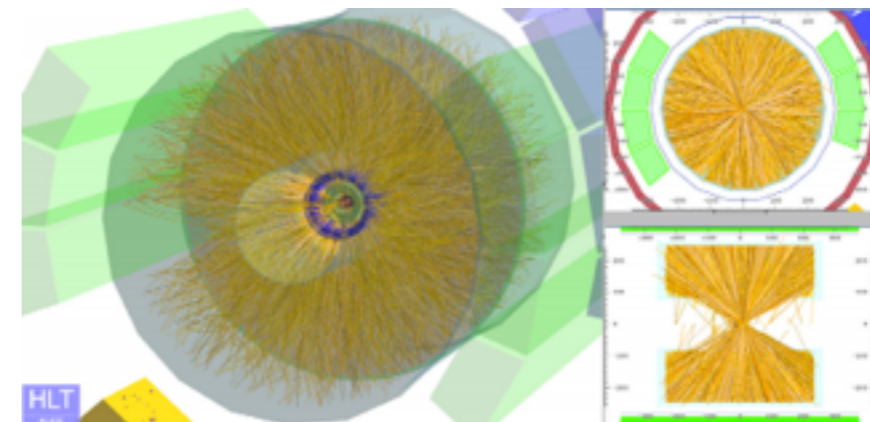
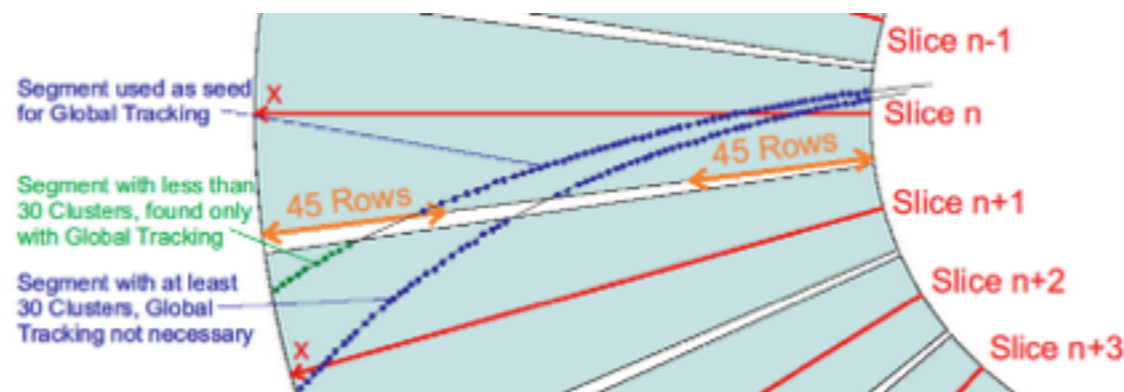
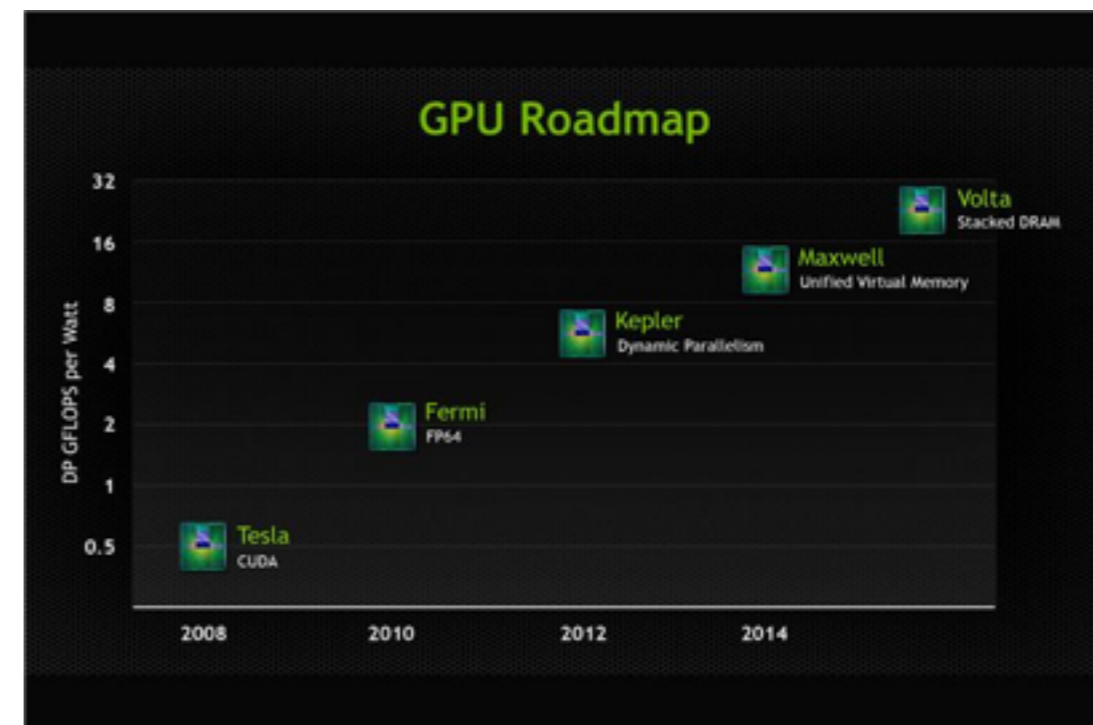
Knights Landing

- Current Phi is (to me) a technology demonstrator in many ways
 - Get people used to an architecture, see how well it flies in the market
- Next generation (Knights Landing) looks rather more impressive:
 - Uses Silvermont cores (from the atom)
 - Out of order execution is back
 - Paired cores share a 1MB L2 cache
 - Up to 16GB of eDRAM (fast onboard memory)
 - DDR4 System memory (max 384GB?)
 - Potential >3TFlops
- Also available in 'standalone' mode



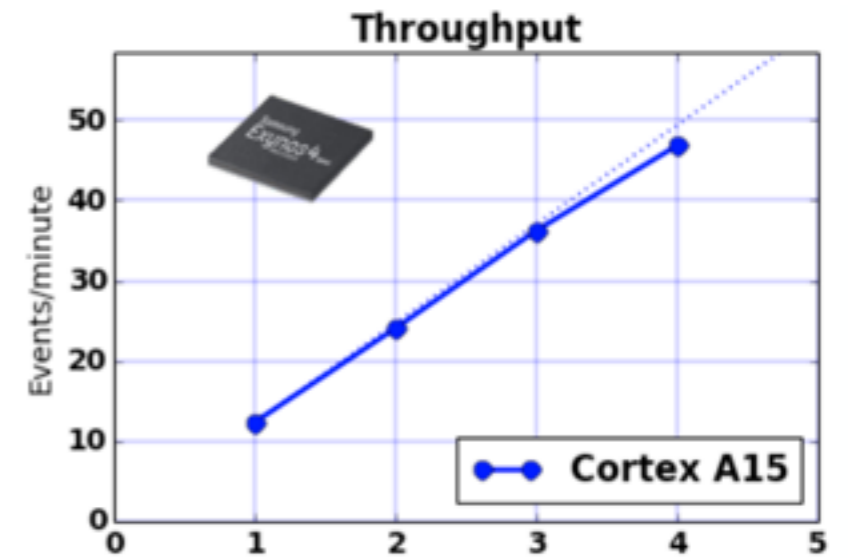
GPGPUs

- Huge potential FLOPs
 - Classic HPC coprocessors
- Expect to see growth here too (next up nvidia Maxwell cards)
 - Worries about the death of desktop graphics?
- Very different programming model
 - OpenMP/CL/ACC will save you
 - But at a price (and the CPU performance of OpenMP sucks)
- Lack of uniformity at sites
 - Better for a closed environment (online, Tier-0?) unless offload can be made extremely flexible
- ALICE plans for GPGPU at their Tier-0 fit their problem particularly well



ARM and ATOM

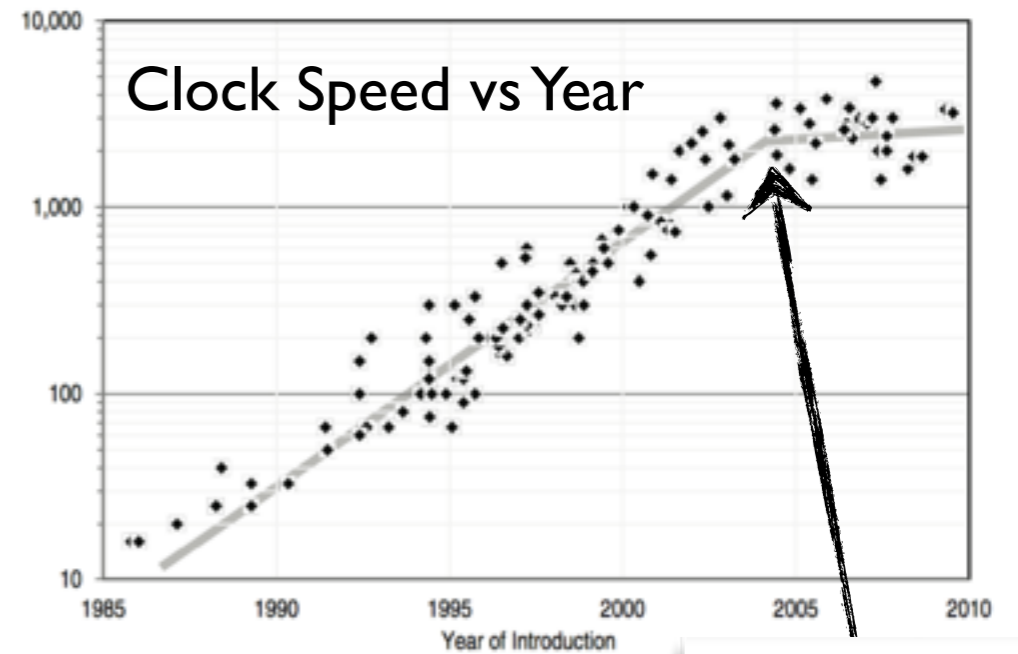
- ARM is now 64 bit (ARMv8)
 - Weaker cores than Xeon
 - But performance per watt is impressive
 - x4-5 improvement
- If data centres go this way up front costs will fall a lot
 - This may become a competitive way of delivering high throughput computing
 - Expect much less memory per core, however
 - Typically, 4 cores with 2GB
- Intel ATOMs have a similar market target, but x86 architecture
 - New Silvermont cores (22nm) are being tested now



Geant4, Andrea Dotti



Some General Considerations with Moore's Law



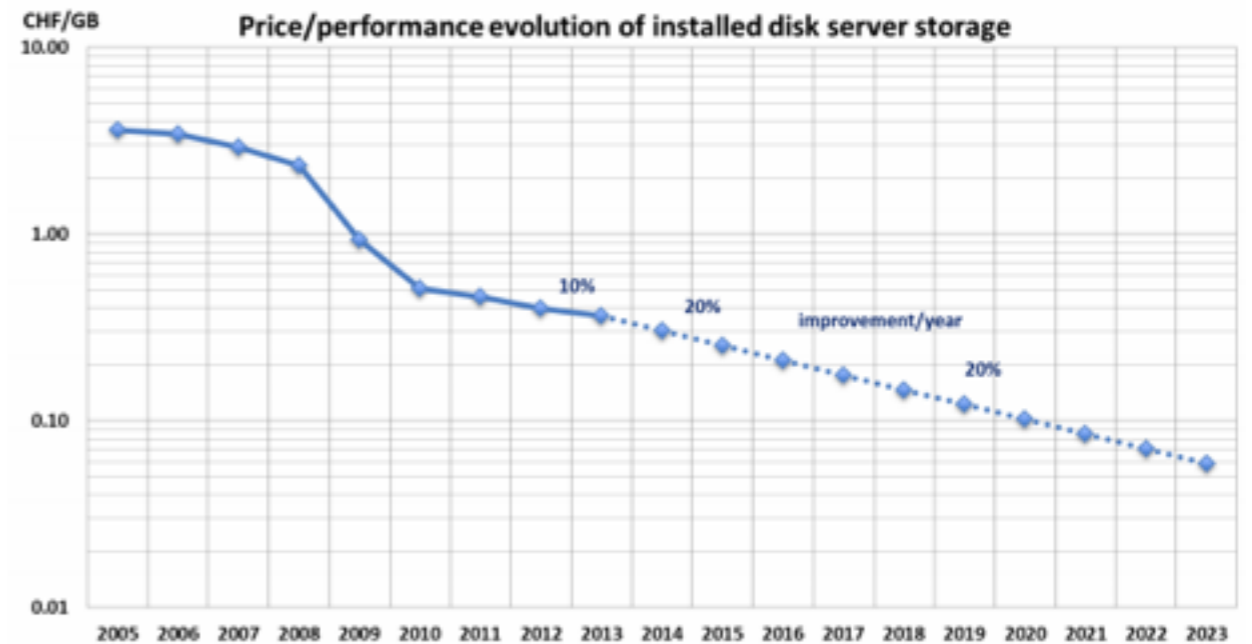
- The issue with Moore's Law is that transistor counts do not equal performance
 - There is now a lot of transistors looking for something to do:
 - Vector registers
 - Out of order execution
 - Multiple Cores
 - Hyperthreading
- All of these techniques increase the theoretical performance of a processor
 - But it is increasingly hard to achieve this performance (or close to it) with HEP applications

Free lunch
ended a
long time
ago

Processor Summary

- Intel Xeons dominate HTC
 - Coprocessors are specialist
 - Low power chips have a high capital cost per HEPSPEC
- Already we're being asked to use HPC resources to make up the compute shortfall
 - Getting access to co-processors 'for free'
 - Volunteer computing at the other end?
- Then in 5 years? 10 years?
 - Server Xeons may become more limited segment of the market
 - Sites will integrate across their customers' needs to decide what to buy
 - We will certainly not be the only customer and we may not even be the biggest

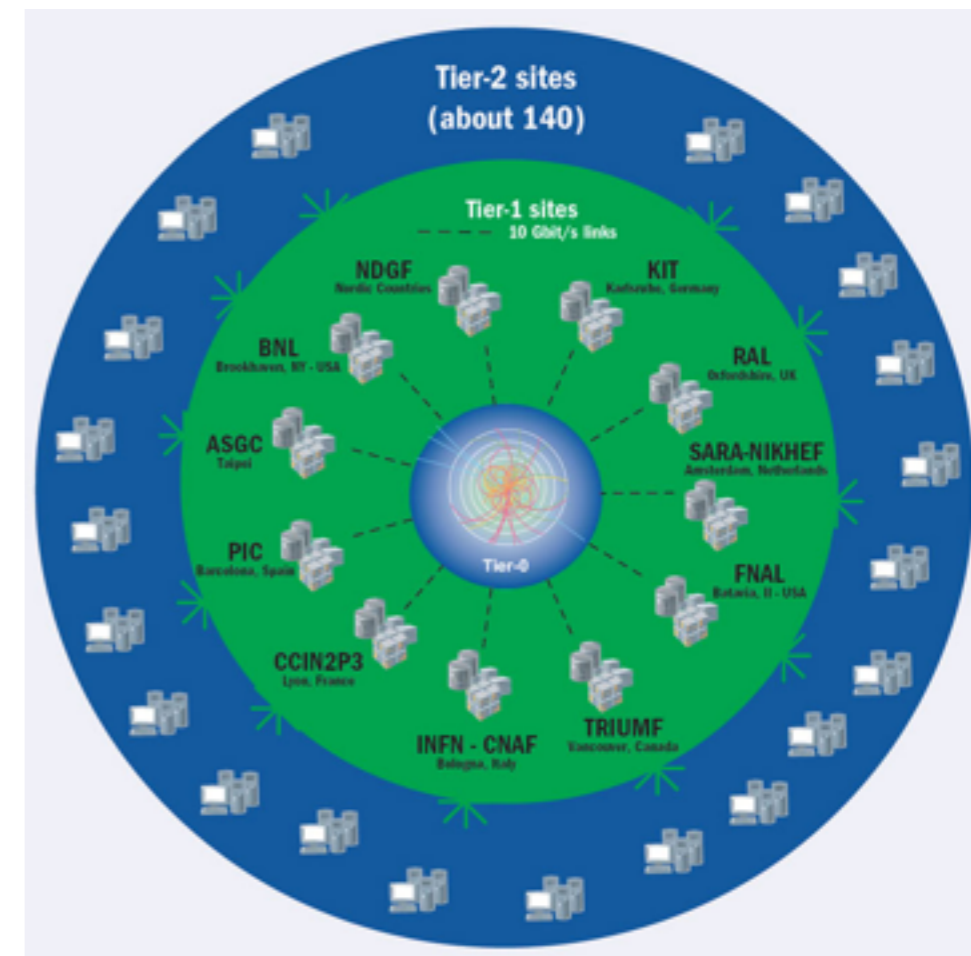
Storage and i/o



- HEP uses tape and disk significantly
 - SSD prices remain too high to store our data volumes at a price we can afford
- Technology projections here are reasonable
- Most worrying aspect is stagnation of i/o rates from disks
- Increasing time to read the whole of a hard drive

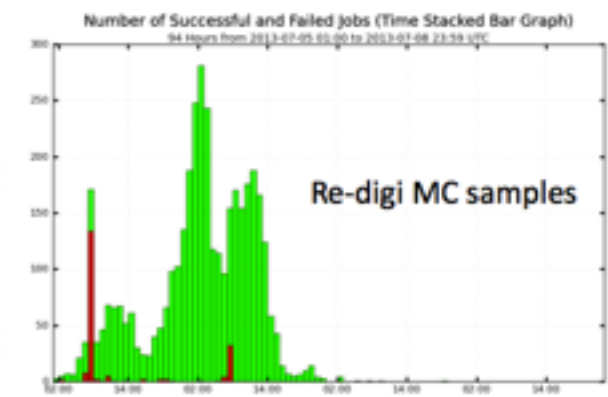
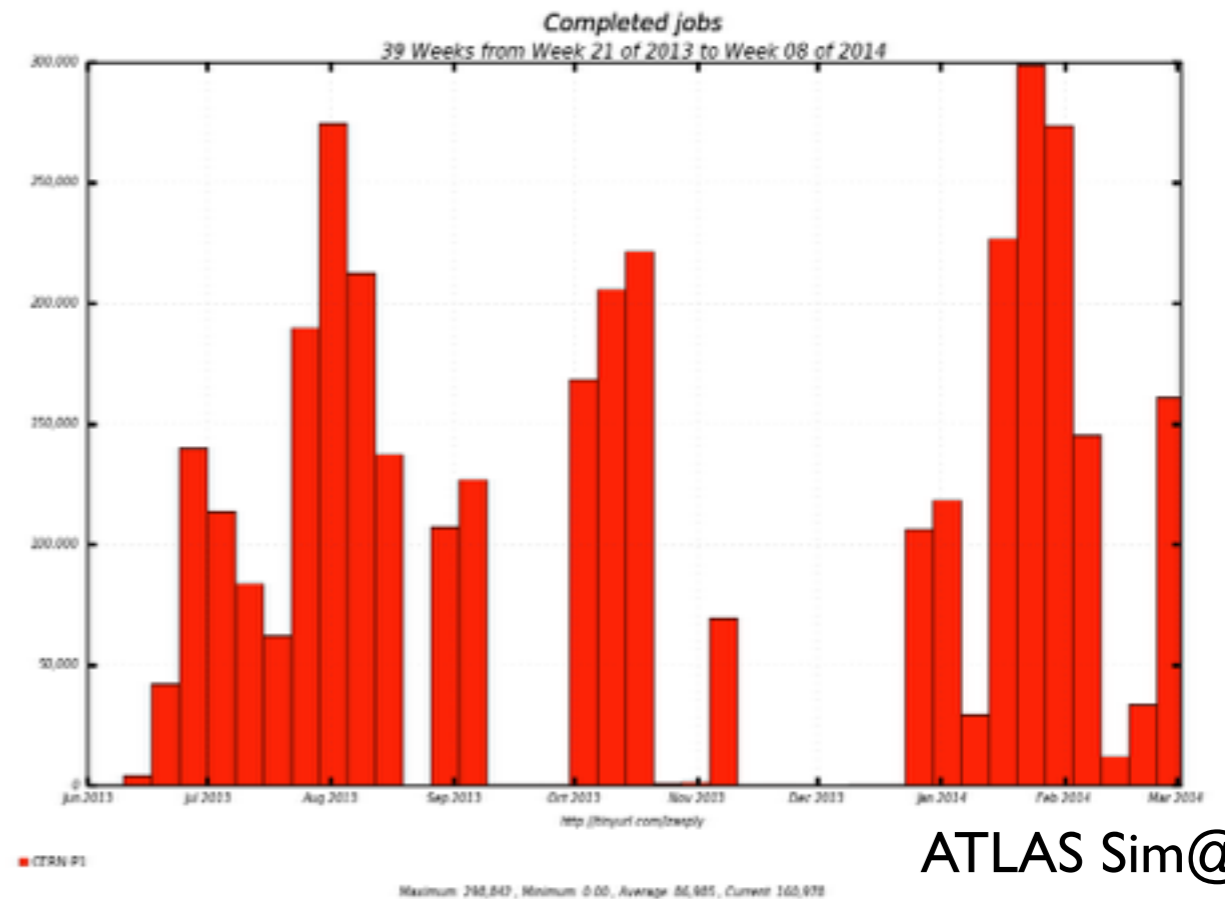
Facilities: Grid and Cloud

- CPUs, disks, tapes networks have to be hosted somewhere
- Tiered structure for HEP has worked very well for Run I
- Cloud technology will percolate into WLCG
- Networks are getting better and MonARCH model is breaking down (e.g. FAX)
 - But they are not an infinite resource
- HPC facilities are also of interest
 - Scavenging CPU cycles for now...



Flexibility - Trigger Farm Use in LSI

- A good example of increasing flexibility was the use of trigger farms during LSI
- Cloud interfaces used to setup large computing sites in an agile way



Fat to Thin

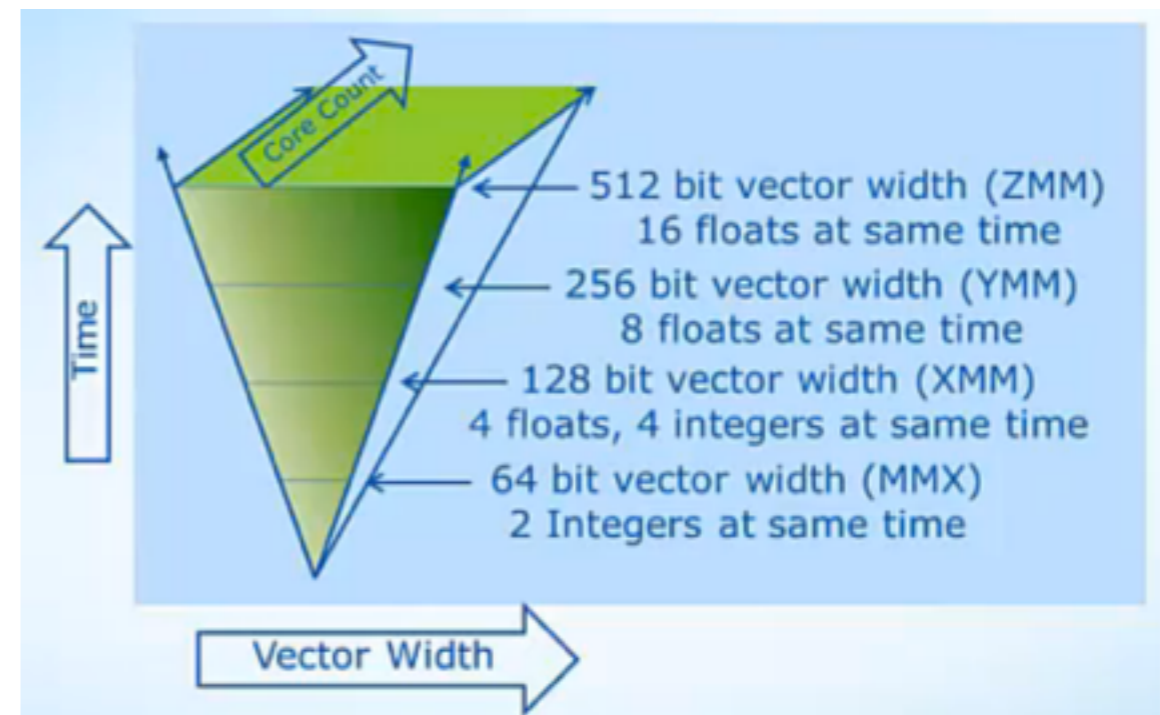
- Coprocessors to mobile computing
 - Different weights of cpu/network/io
- Target different platforms at different stages of the data analysis chain
- But in all cases we must try to take advantage of what is there
 - We can't be fussy about what compute we can use
- Fortunately the recipe to exploit heterogeneous resources has not really changed...



Performance Scaling

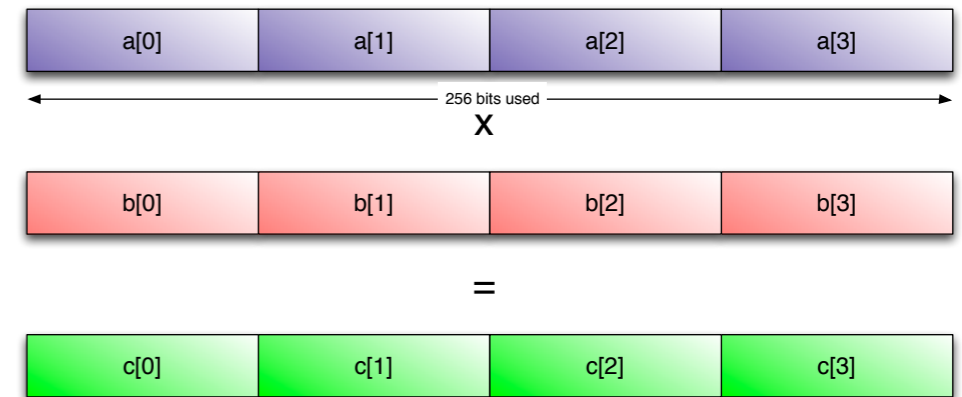
Performance Aspect
Clock Speed
SIMD/Vectorisation
Instruction Pipelines and Latency
Hardware Threads
Multi-Cores
Multi-Sockets
Multi-Node

- This is a multi-dimensional space
- Each aspect we do not optimise loses x2-x10 in potential performance



SIMD

AVX 256 Bit Registers: $c[] = a[] \times b[]$



- Single Instruction, Multiple Data
 - AKA vectorised operations
 - Parallelise calculations across data
 - Works very well *if* your problem is amenable to being expressed like this
 - Two approaches
 - SIMD of the calculations done for a single object (track, cluster, etc.)
 - Relatively simpler, but scaling issues due to data starvation in very wide registers (512bit \rightarrow 16 floats or 8 doubles)
 - SIMD by doing calculations for multiple objects in parallel (a.k.a. data parallelism)
 - Harder to do, but abundant parallelism in hadron colliders

Using SIMD

- Vectorise libraries are great where they fit into the workflow:
 - Linear algebra: Eigen, Smatrix, Blaze, MKL
 - Transcendental functions: Intel Maths Library (libimf), VDT
 - but vectorised libraries only give a speed up where the library writer intersects with our code
- Autovectorisation is improving, but what the compiler can manage is still very far away from what our code looks like
 - It's also a constant maintenance operation - any update to code/compiler can break it
- Exotic compilers, like SPMD, emulate GPGPU like behaviour on CPUs
 - Relatively easy way to express concurrency, but this is a left field development - too exotic for mainstream use
- C++ language support for SIMD is not on the table yet, and will probably not even make it into C++17
- OpenMP 4.0 seems to hold out hope:
 - `#pragma` to tell the compiler what you know as the programmer
 - Vectorised functions can execute down SIMD lanes, c.f. ISPC

Memory Locality and Latency

- Memory locality is extremely important
 - Vector loads and stores are vastly more efficient and cache misses cost hundreds of cycles
 - Put like things together
 - Structures of arrays not arrays of structures
- Simple tests of sigmoid functions (for neural networks) with contiguous and random memory access

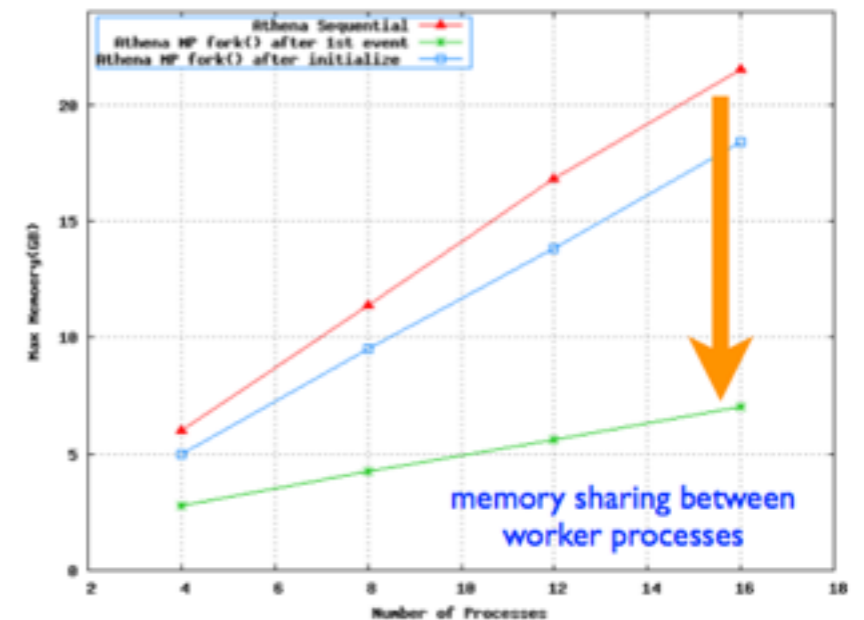
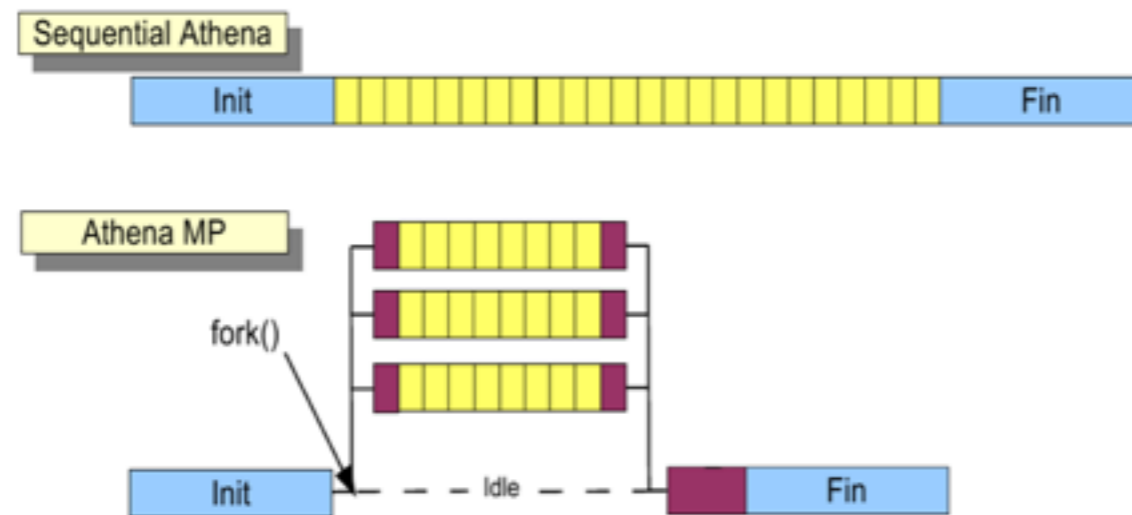
Function	Contiguous	Random	Ratio
Logistical Fn	2400ms	9700ms	÷4
Fast sqrt Fn	560ms	7900ms	÷14
Ratio	x4.3	x1.2	

Improve because
of SIMD

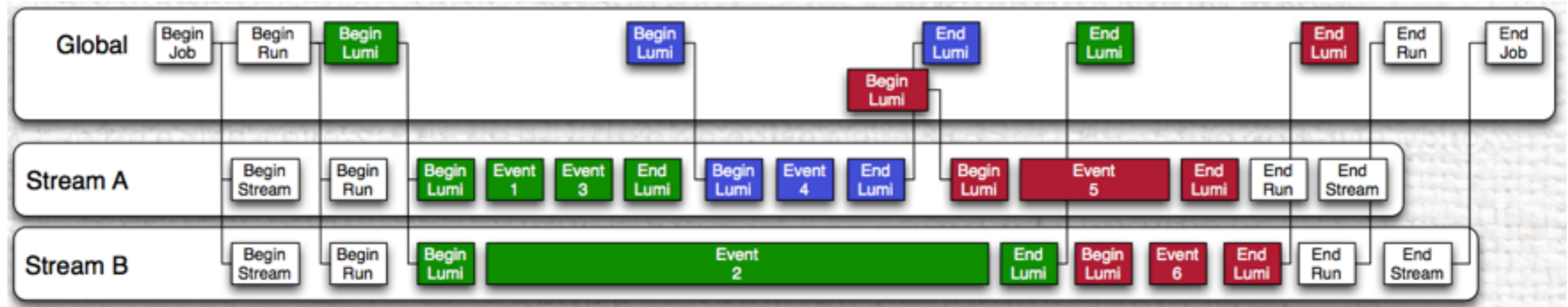
Losses because of
lack of locality

Memory Footprints and Multi-threading

- Memory per core will very likely continue to fall
 - Certainly it does on anything except Xeons
 - (And, in any case, we don't have spare money to buy 16GB per core even if we wanted too)
- ATLAS have used forking after initialisation in AthenaMP to keep the memory under control for Run 2
- After that we need to thread
 - Shared memory space for threads
 - Massive memory saving for parallelising processing of events and algorithms



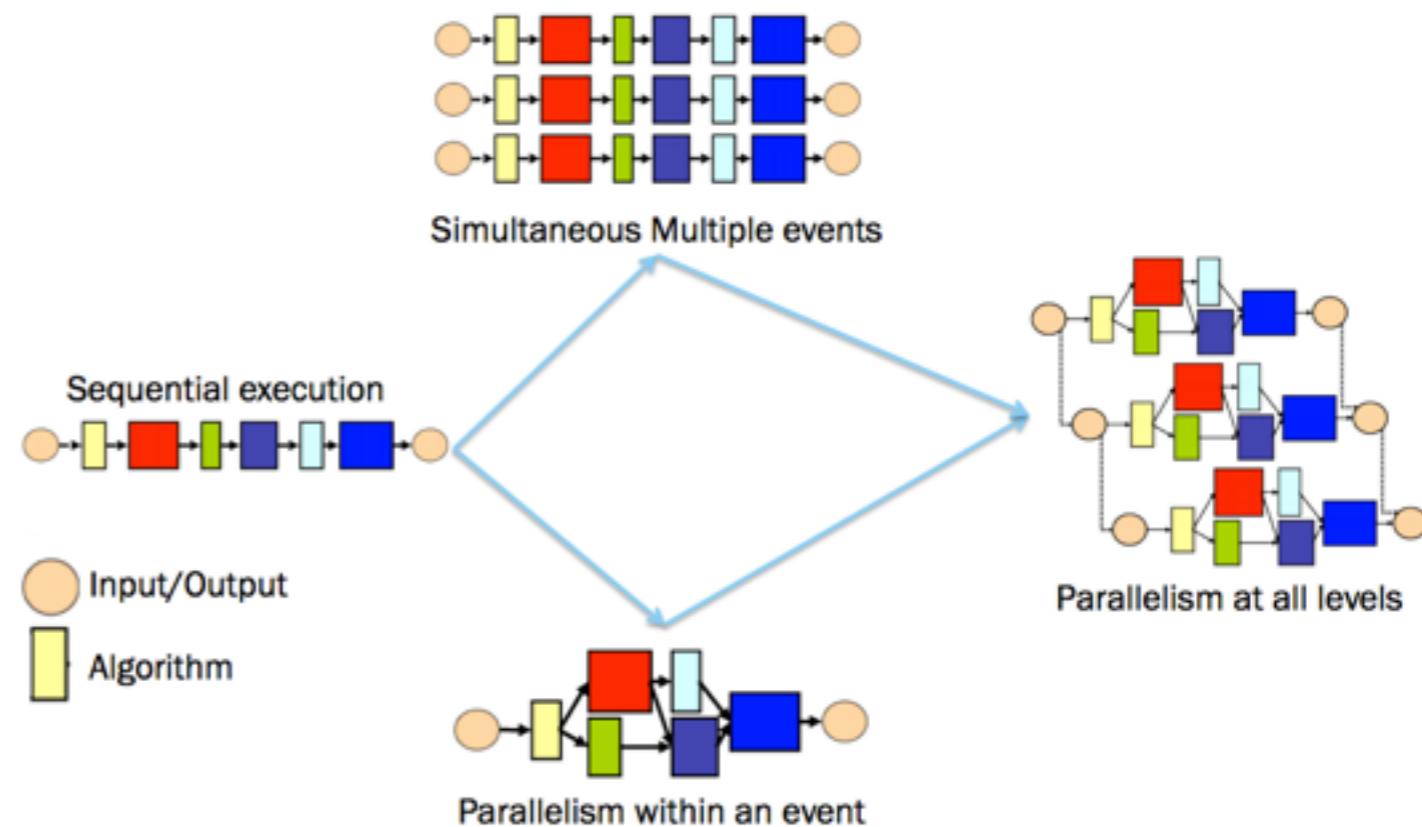
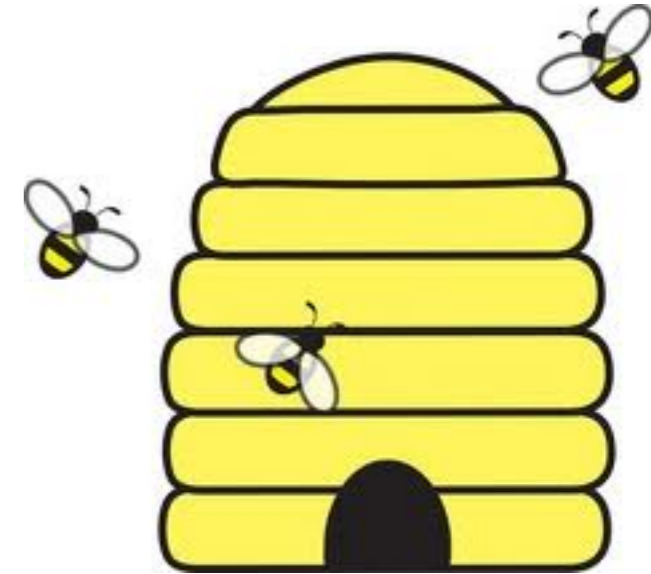
CMS Multithreading



- Process events along parallel streams
 - Put multiple events in flight inside the same process
 - Uses Intel's Threaded Building Blocks for expressing concurrency
 - Heavy weight algorithms can also use parallelism in a coherent way with the framework

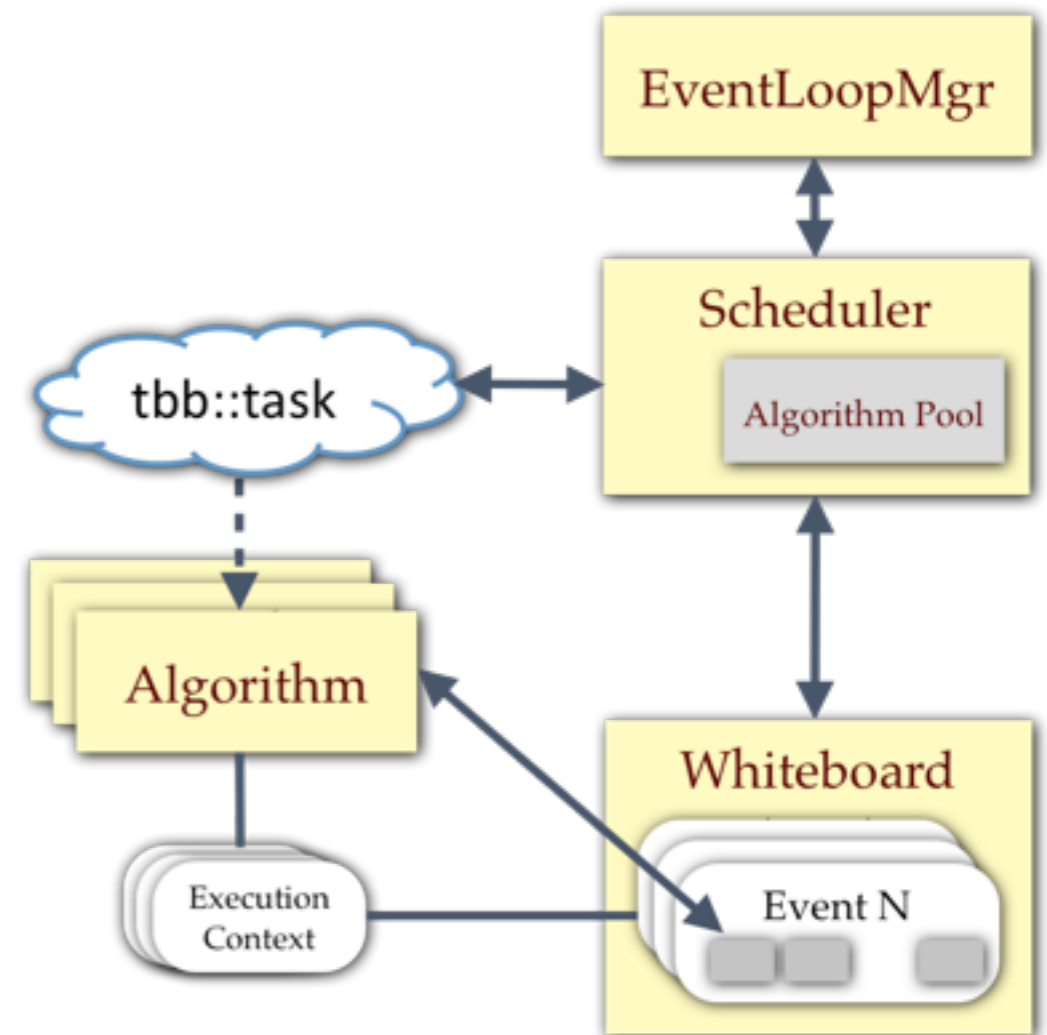
GaudiHive

- Attempt to introduce framework parallelisation for HEP
- Event level and algorithm level
- Born out of the HEP Concurrency Forum
- Developed by SFT, ATLAS and LHCb

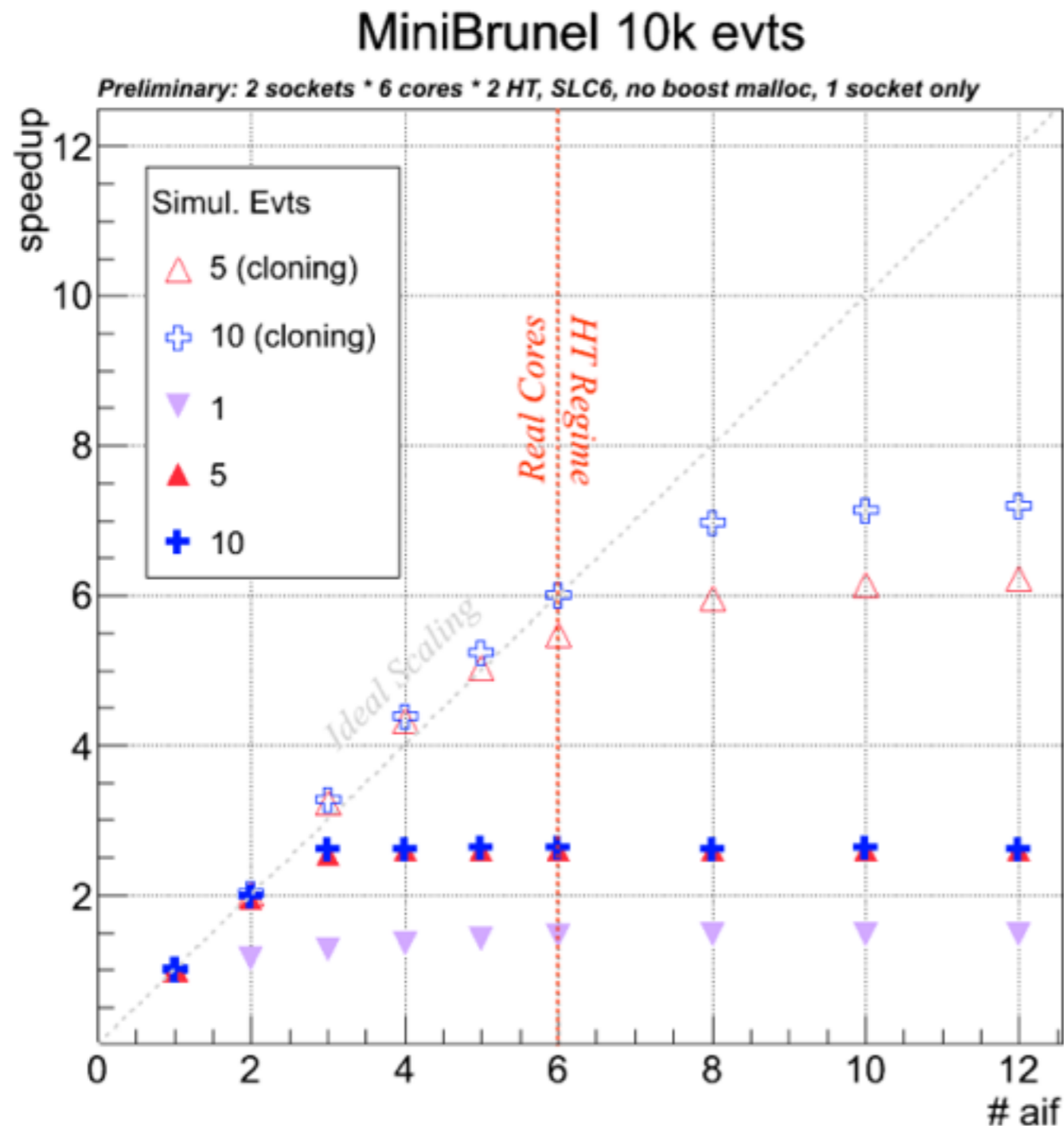


How it works

- Algorithms declare their input dependencies
- AlgScheduler looks at the Whiteboard (StoreGate) for the current data
 - Event dependent!
- Any algorithm that has its input data ready is scheduled as a task
 - Control algs in flight, events in flights, etc.
- When all algorithms have run on all events we're done



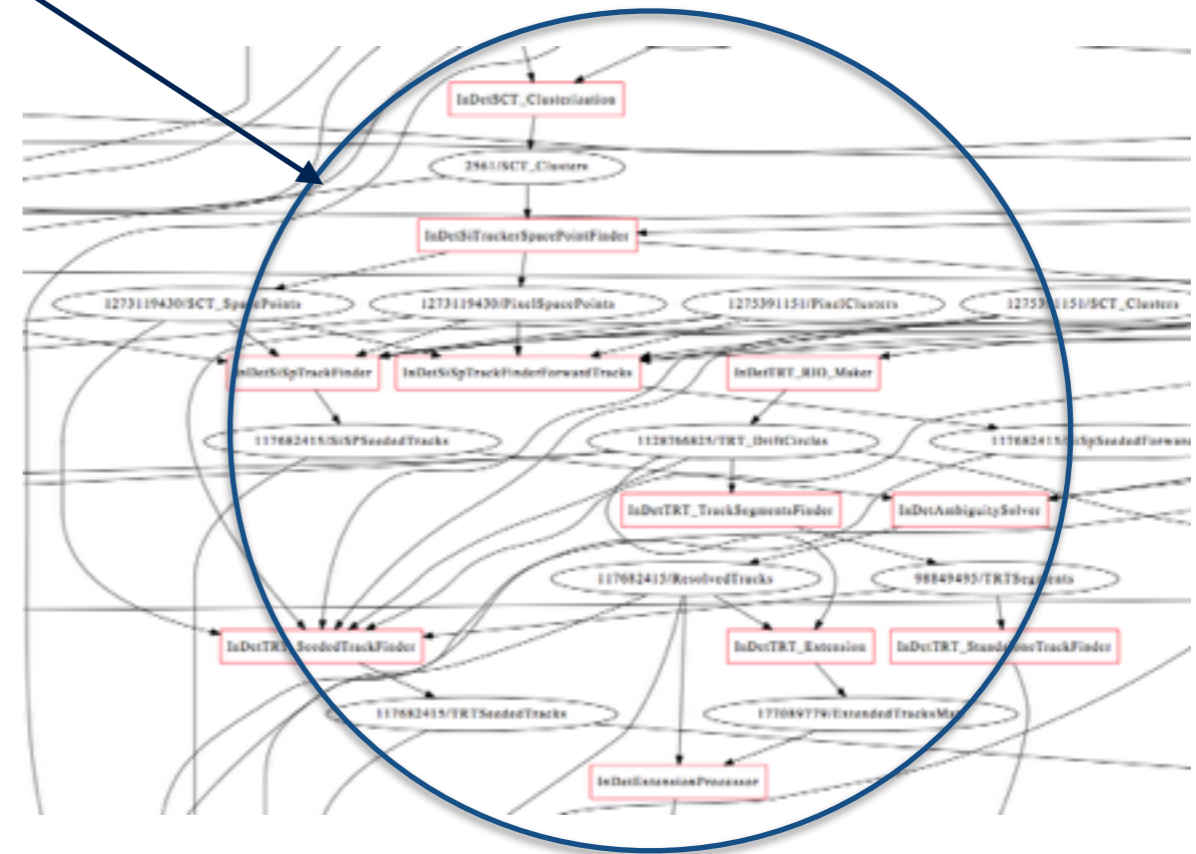
Hive results



- LHCb mini-brunel results presented at CHEP
- Vital to allow expensive algorithms to be cloned (~made thread safe)
- ATLAS have similar results from calorimeter and inner detector reconstruction

Data flow graph

- Data flow for ATLAS RAW to ESD
- Not easy to understand this flow
- To say nothing of dataflow back doors, that don't use the event store



In-algorithm parallelism

- Inner Detector absolutely requires in-algorithm parallelism
- Algorithm parallelism is probably still necessary or Ahmdal's law will kill us:

$$t_{||} = p/n + s$$

- Very interesting to look at how parallelism across events, algorithms and within algorithms behaves
 - Intel's Threaded Building Blocks (TBB) seems to be the weapon of choice

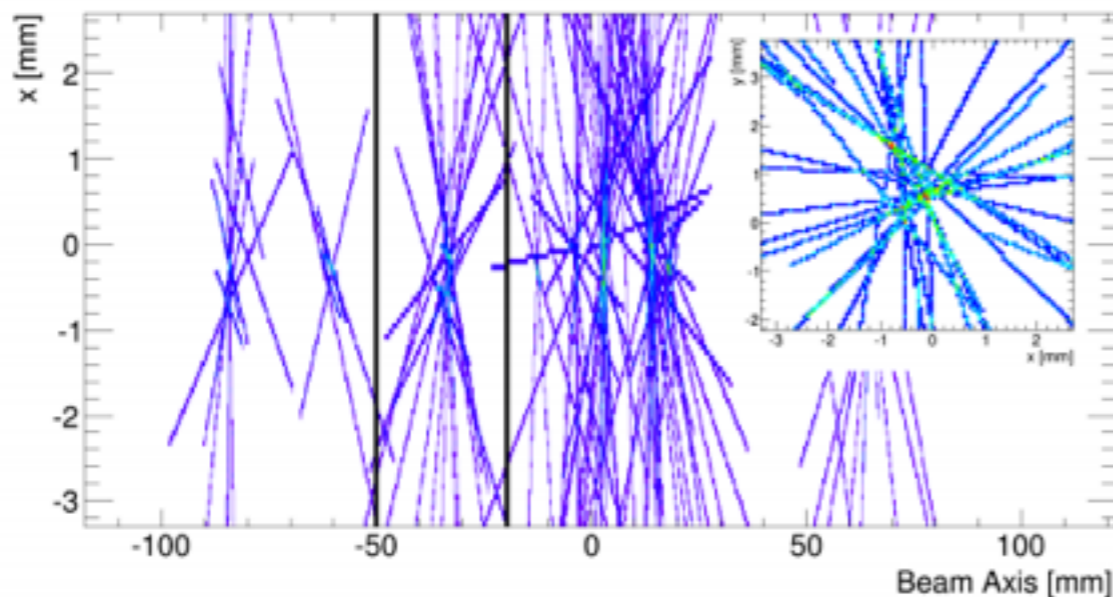
Back to the Drawing Board?

- Code optimisation and infrastructural improvements will only take us so far
 - These can be introduced incrementally and will help gain experience with improved tools and methods
- Our biggest concerns are to improve the algorithmic performance and optimise our code for high pile-up scenarios
 - In many cases we are developing techniques which are already known, but were not the optimal solution for LHC Run I

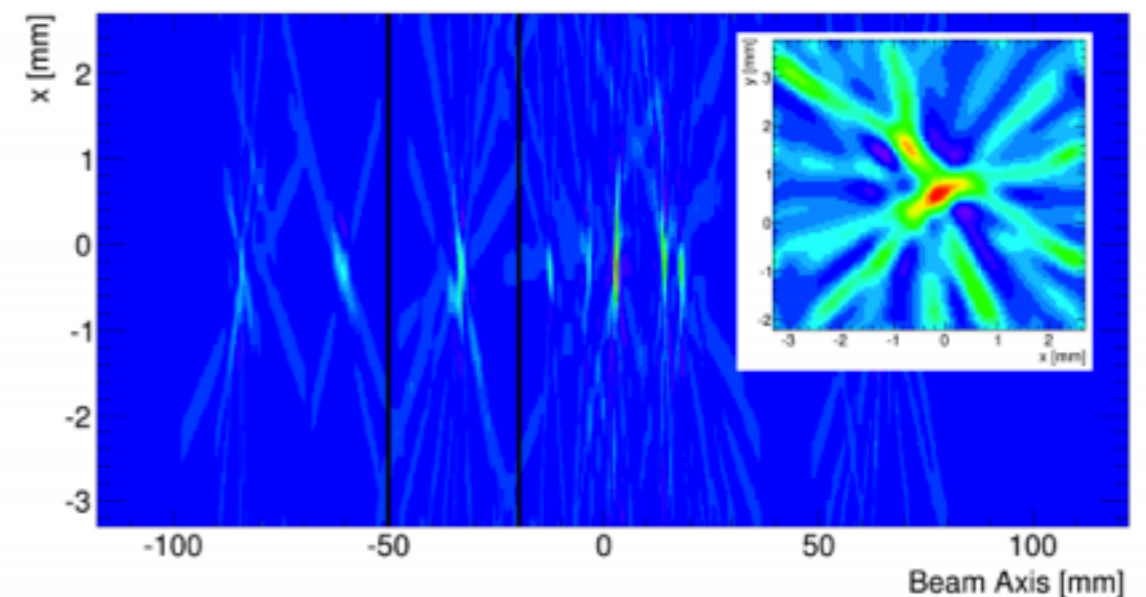
Vertex Finding

- Transformation to Fourier space and filter to make vertices stand out
 - Takes longer than current vertex fitter at current pile-up
 - Promise is in much better scaling with μ

ORIGINAL DENSITY



FILTERED DENSITY

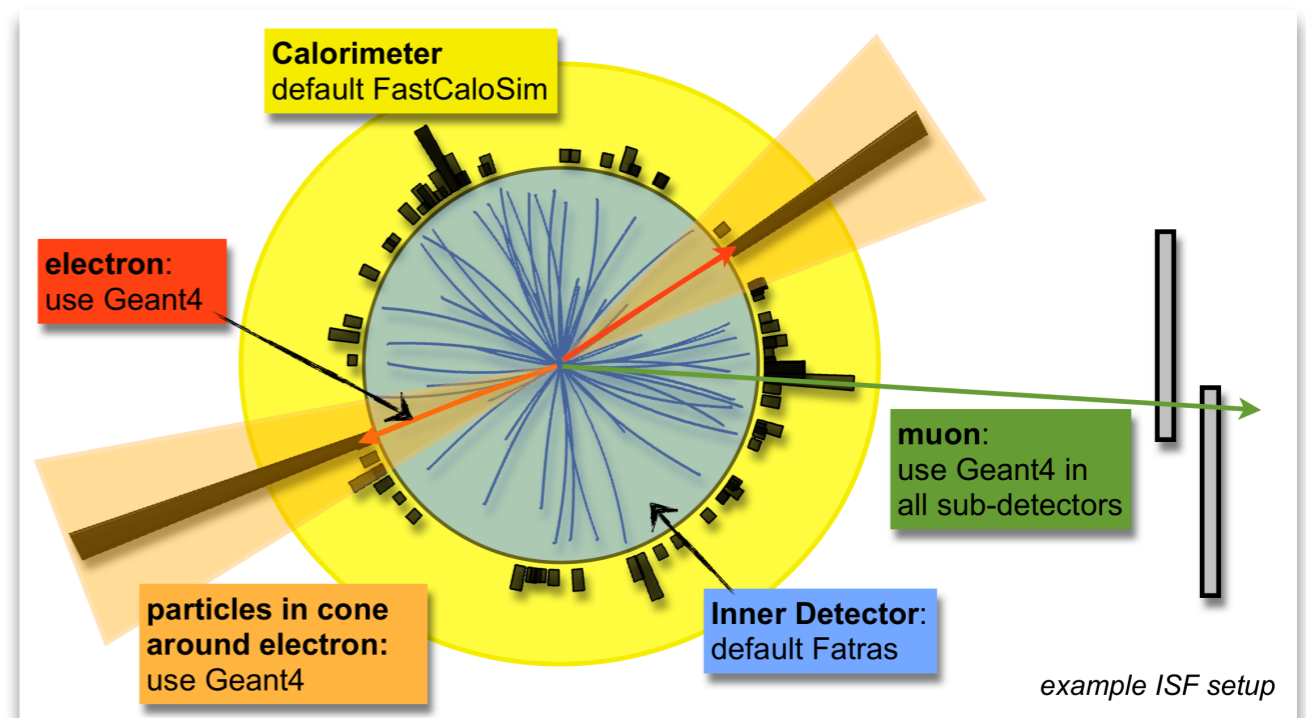
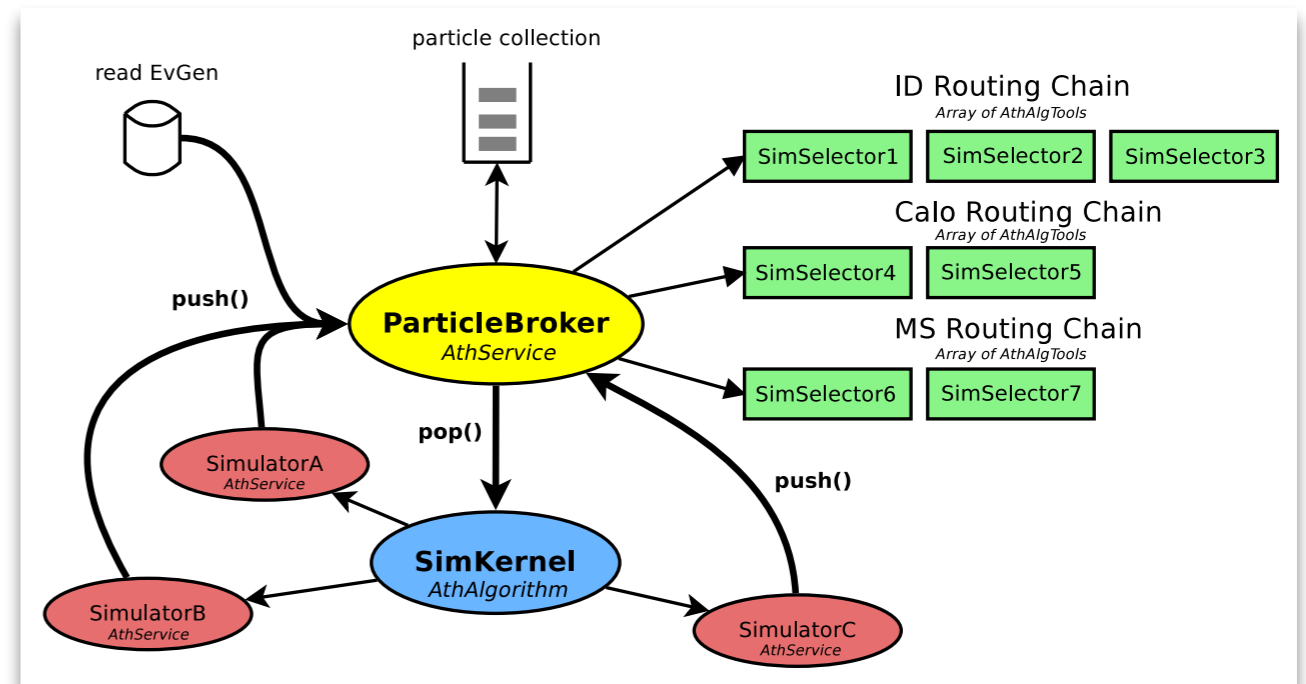


ATLAS Integrated Simulation Framework

- Single framework for simulation
 - Simulation engines act like services
 - Choose engine based on particle type and region of interest
- Mix simulation types within a single event
- Full potential realised when combined with fast digitisation and reconstruction

Tracker	Calo.	Muons	speedup
full	fast	full	~20
fast	fast	fast/full	>100
Rol guided fast/full			~100

Can we extend this to run simulations in parallel?



Conclusions

- Upgrade path of the LHC offers great possibilities for precision Higgs physics and BSM searches
- The main driver for computing complexity is pileup
 - Large in Run 2, very large in Run 3, humungous at HL-LHC
- Significant challenges involved in taking advantage of both current and future hardware
 - Software frameworks need to evolve to allow the experiments to extract the most from the hardware we end up with
 - And the software and interfaces that it runs
- We need to be as agile as possible in computing, from the CPU to the wide area network
- HL-LHC requires really rethinking a lot of our algorithmic approaches to date