HLT Revolutions: LS2 and Beyond

Sami Kama

SMU

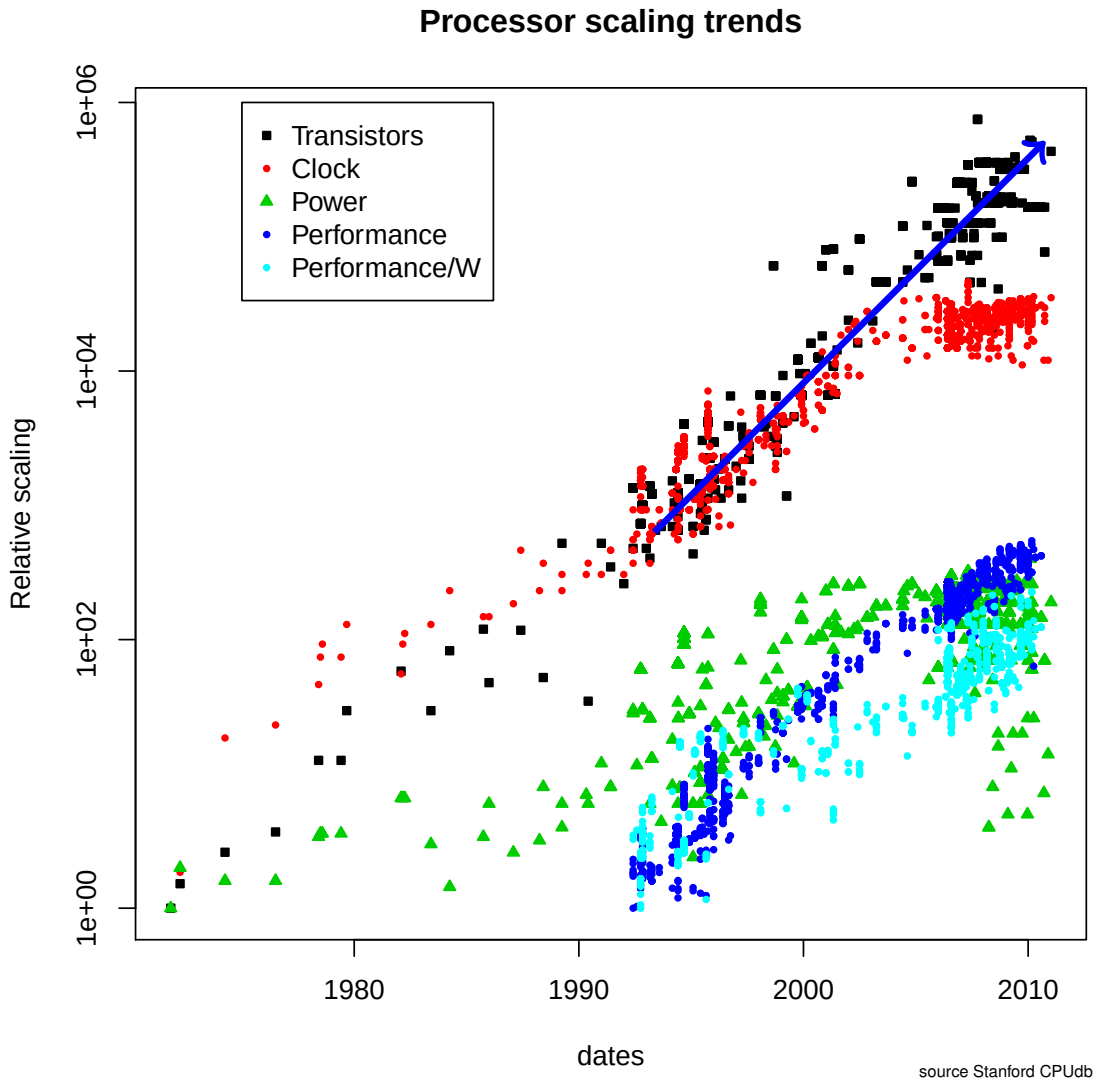
DAQ at LHC workshop, March 2013

HL-LHC aims

- HL-LHC may deliver up to $2.5 \times 10^{35}\ \mathrm{cm}^{-2}\,\mathrm{s}^{-1}$
  - up to ~350 pileup events
- HLT farms need to grow proportionally
- In terms of current processing power, guestimated farm sizes
  - LHCb ~300*k* cores
  - ATLAS ~160*k* cores
  - CMS ~64*k* cores
  - ALICE ~250*k* cores and ~6.4*k* GPUs
- Memory requirements will also increase due to larger events!
- Luminosity leveling might change farm size requirements.

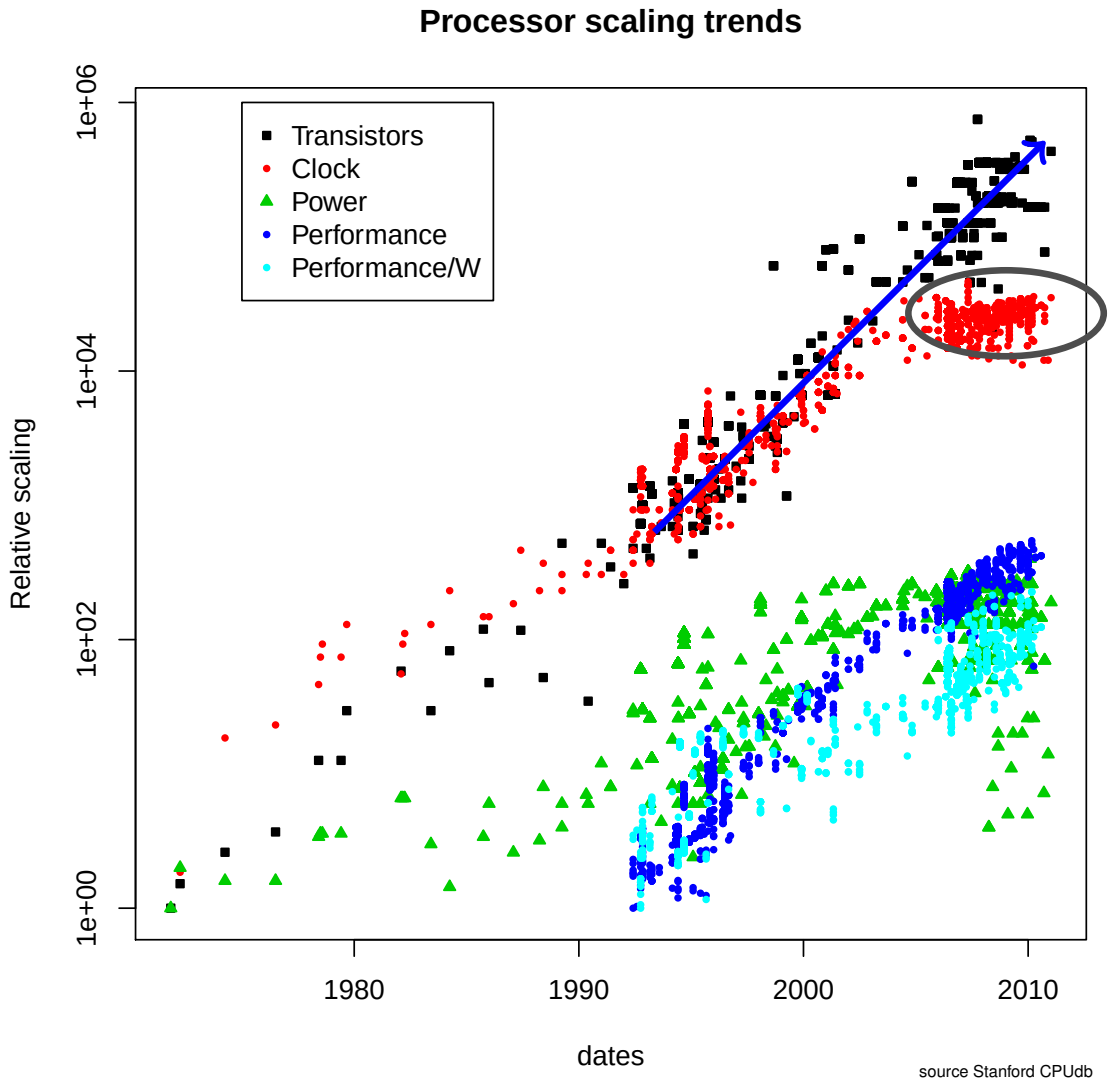
Moore's Law to the Rescue?

- Moore's law is for transistors. May hold for a couple more years.
- Clock speed seems to be saturated
- Processing power is still increasing due to increasing number of cores per CPU and wider vector units.



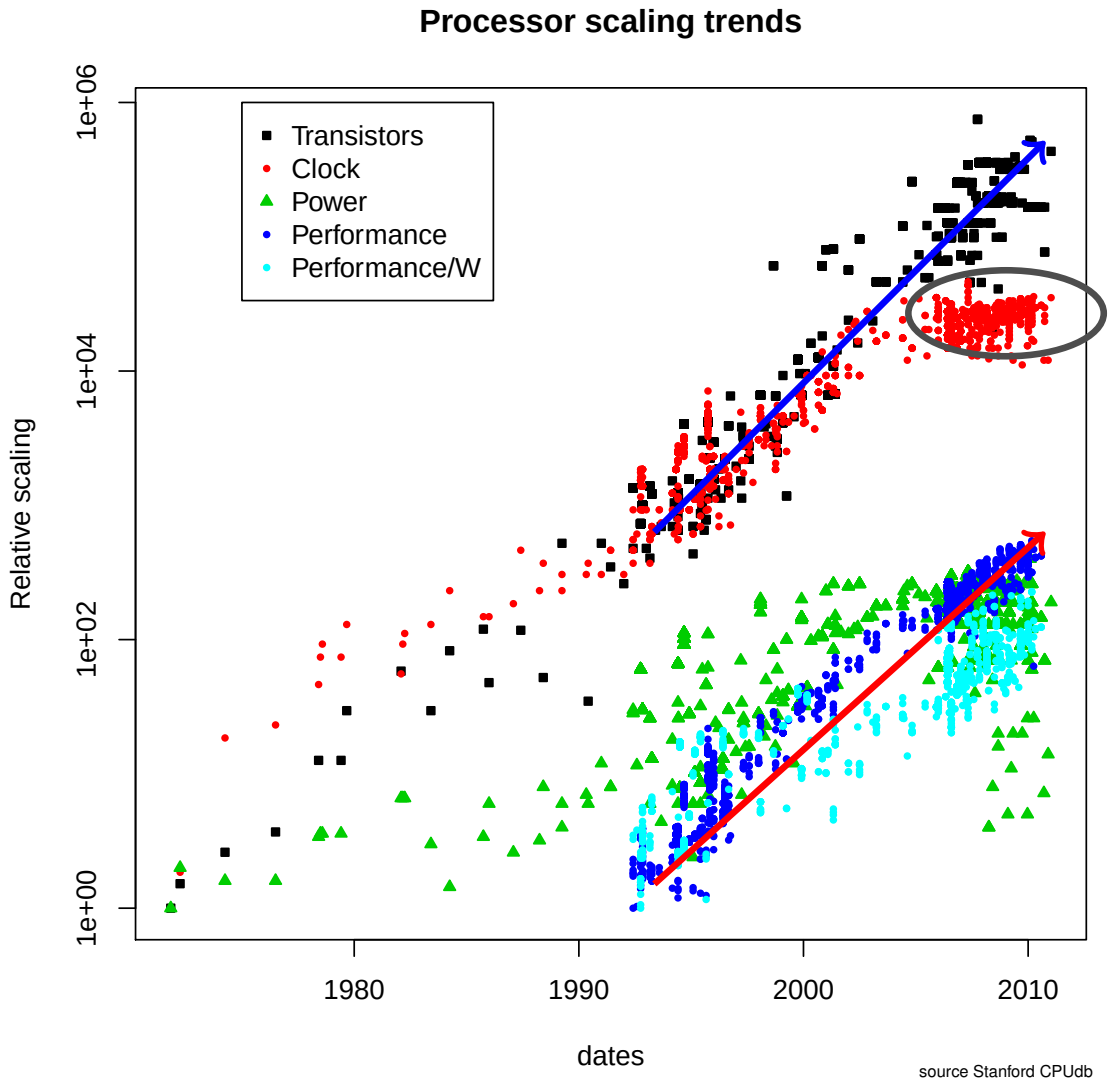
Moore's Law to the Rescue?

- Moore's law is for transistors. May hold for a couple more years.
- Clock speed seems to be saturated
- Processing power is still increasing due to increasing number of cores per CPU and wider vector units.



Moore's Law to the Rescue?

- Moore's law is for transistors. May hold for a couple more years.
- Clock speed seems to be saturated
- Processing power is still increasing due to increasing number of cores per CPU and wider vector units.

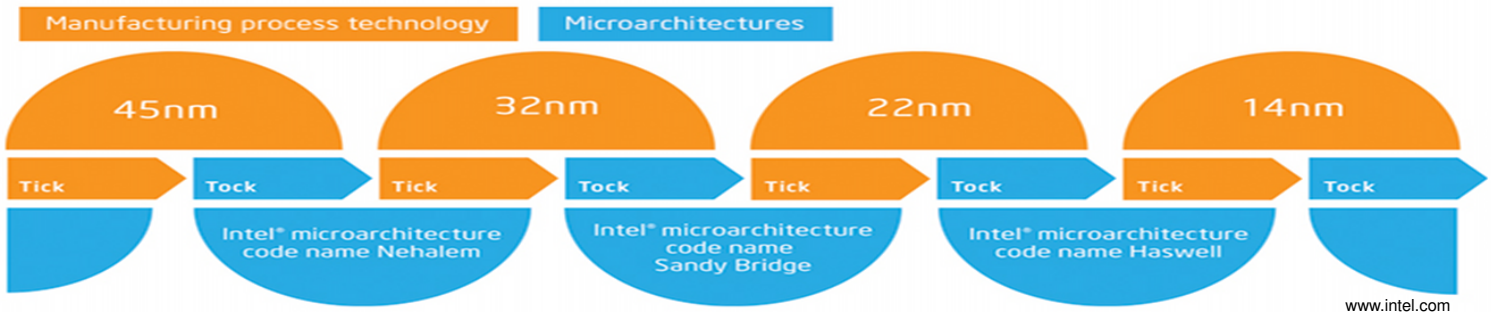


Current Hardware

- Intel Haswell
  - Better SIMD vectors(AVX2 and FMA) and parallelism
  - Up to 2x theoretical improvement over Sandy Bridge, at least 10% over Ivy Bridge (i.e. $1.1\times\text{Ivy Bridge} \leq \text{Haswell} < 2\times\text{Sandy Bridge}$)
  - Includes a GPU with 20 or 40 execution units
- Intel Xeon-Phi Co-processor (aka MIC)
  - 61 Pentium-like cores @ 1.1 GHz, 512bit wide vectors, 8GB RAM
  - 1 TFLOP peak processing power
- ARM CPUs
  - Low-power RISC chips, powering most mobile devices
  - 64-bit prototype servers already available, backed by big companies
  - Up to 4 cores, SIMD support
- NVIDIA Kepler
  - 14x192 Cores, up to 32 simultaneous tasks, dynamic kernels
  - 6GB memory, 3.95 TFLOP SP, 1.31 TFLOP DP peak processing

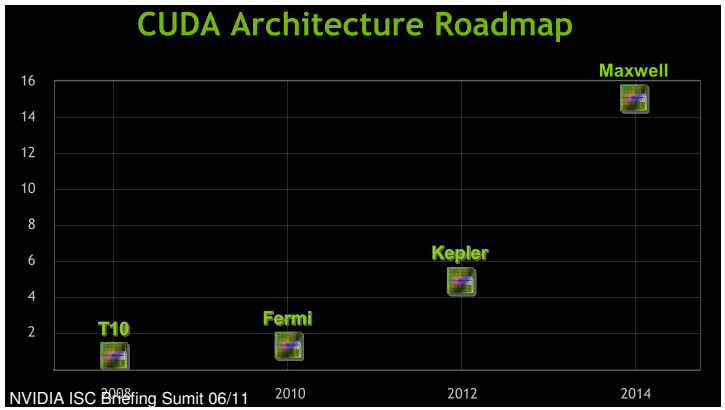
Roadmaps

Intel Roadmap



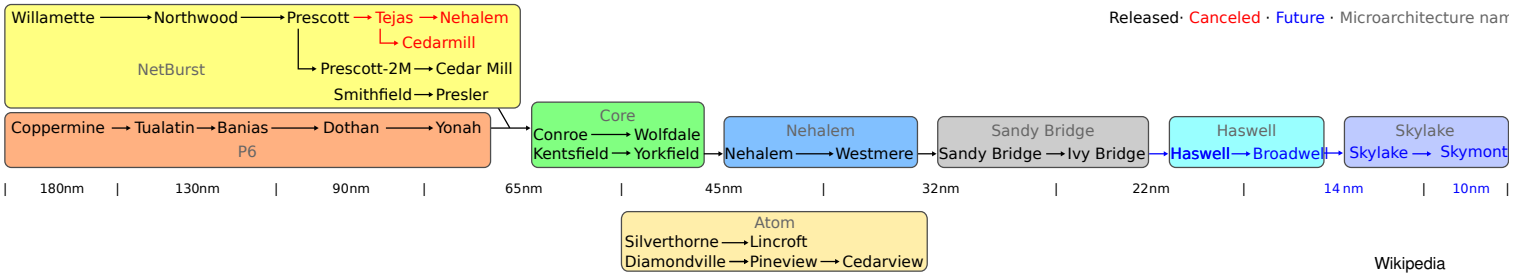
- Costs, profits and technology modifies roadmaps
- Unfortunately they are not always realized
- According to 1990's roadmaps we should have $O(10\ \mathrm{GHz})$ CPUs around

NVIDIA Roadmap



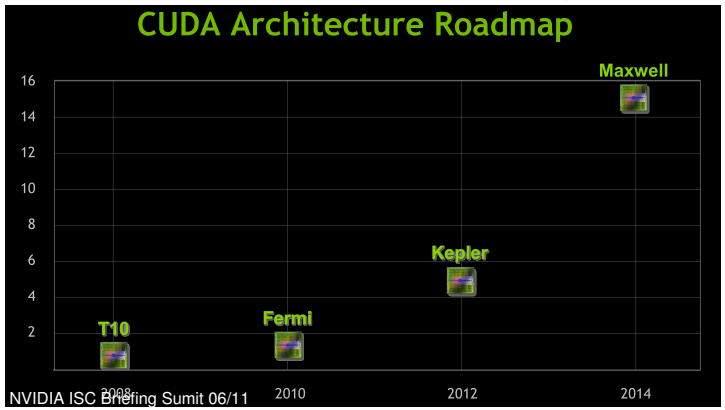
Roadmaps

Intel Roadmap



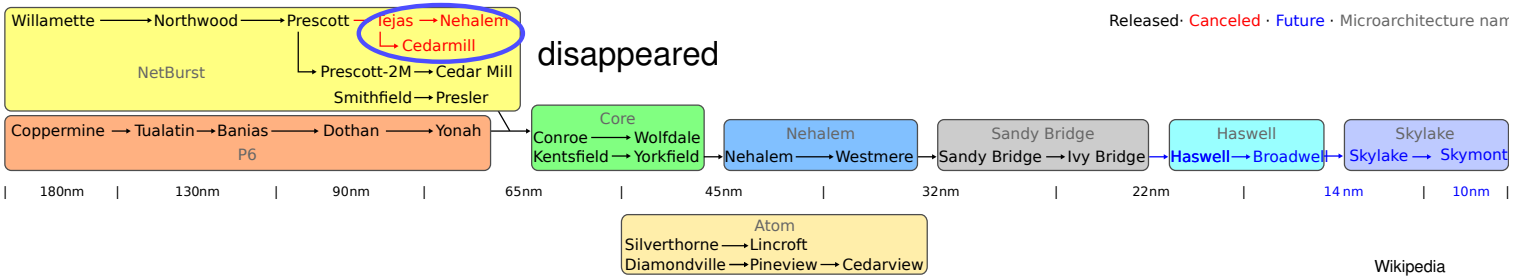
- Costs, profits and technology modifies roadmaps
- Unfortunately they are not always realized
- According to 1990's roadmaps we should have $O(10\ \mathrm{GHz})$ CPUs around

NVIDIA Roadmap



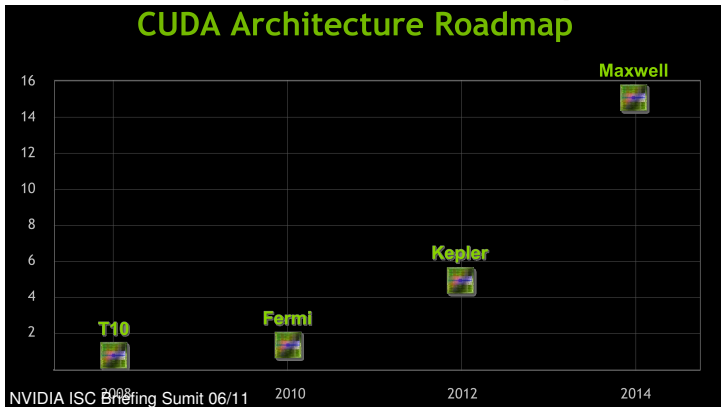
Roadmaps

Intel Roadmap



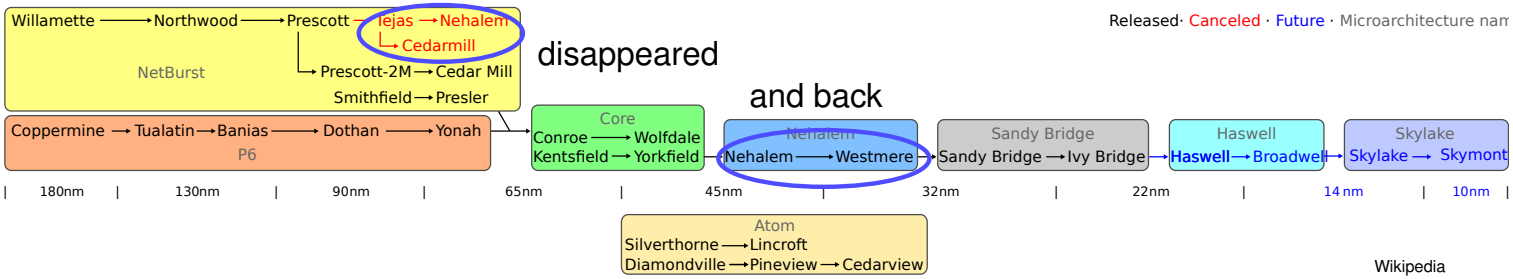
- Costs, profits and technology modifies roadmaps
- Unfortunately they are not always realized
- According to 1990's roadmaps we should have $O(10\ \mathrm{GHz})$ CPUs around

NVIDIA Roadmap



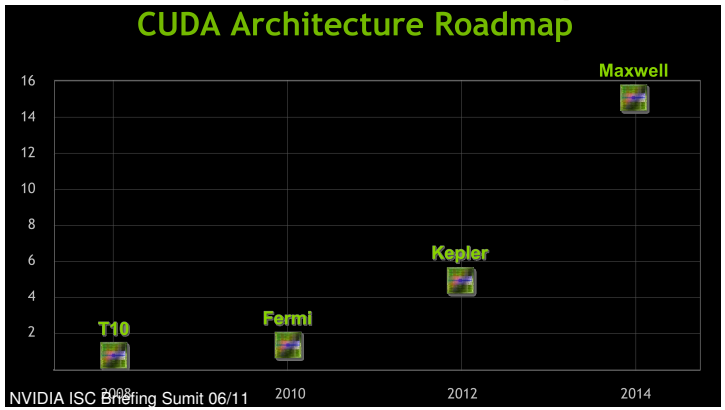
Roadmaps

Intel Roadmap



- Costs, profits and technology modifies roadmaps
- Unfortunately they are not always realized
- According to 1990's roadmaps we should have $O(10\ \mathrm{GHz})$ CPUs around

NVIDIA Roadmap



Guessing Hardware

- 2018 is too hard to extrapolate

Guessing Hardware

- 2018 is too hard to extrapolate
- Producers will try to keep up with Moore's law but silicon technology is at the physical limits
  - Synthetic benchmarks do not reflect real life workloads

Guessing Hardware

- 2018 is too hard to extrapolate
- Producers will try to keep up with Moore's law but silicon technology is at the physical limits
  - Synthetic benchmarks do not reflect real life workloads
- Energy costs and mobile market is driving the designs but gaming and Exascale HPC are still in the game.
  - Increase in the number of cores
  - Wider vector units
  - What will be the next driving force?

Guessing Hardware

- 2018 is too hard to extrapolate
- Producers will try to keep up with Moore's law but silicon technology is at the physical limits
  - Synthetic benchmarks do not reflect real life workloads
- Energy costs and mobile market is driving the designs but gaming and Exascale HPC are still in the game.
  - Increase in the number of cores
  - Wider vector units
  - What will be the next driving force?
- More heterogeneous computing
  - System-on-Chip devices
  - Hosts with accelerator add-on cards
  - Blade mixtures of different types

Guessing Hardware

- 2018 is too hard to extrapolate
- Producers will try to keep up with Moore's law but silicon technology is at the physical limits
  - Synthetic benchmarks do not reflect real life workloads
- Energy costs and mobile market is driving the designs but gaming and Exascale HPC are still in the game.
  - Increase in the number of cores
  - Wider vector units
  - What will be the next driving force?
- More heterogeneous computing
  - System-on-Chip devices
  - Hosts with accelerator add-on cards
  - Blade mixtures of different types
- Will memory keep up?

Designing and maintaining software is a challenge!

Possible designs-1

Single threaded, one process per core, serial processing

- Pros
  - Most CPU efficient approach
  - Only vectorization and performance optimization
  - Each can offload certain tasks to accelerators, reducing memcopy overhead
- Cons
  - Wasted memory, can be reduced with forking and 32 bit ABI
  - Memory/core is already at the limit and most likely to go down.
  - Resource contention limits performance in accelerator offloading.

Unlikely to work due to insufficient memory resources

Possible designs-2

Multiple processes, each process doing a specific task (pipelining)

- Pros
  - Can accommodate heterogeneous systems and use accelerators for special operations
  - Possibly less memory footprint
  - Simpler than threading
- Cons
  - IPC overhead
  - Increased Latency

Might run into problems if memory/core goes down but simplest design for host-accelerator mixture.

Possible designs-3

Multi-threaded process running on whole node

- Pros
  - Most memory efficient
  - Can exploit sub-event level and multi-event parallelism at the same time.
  - Least resource contention for accelerators
- Cons
  - Hardest to program
  - Needs multiple events on flight to utilize maximum resources
  - Can't easily handle heterogeneous systems.

This is most suitable approach to current hardware extrapolations.

Accelerator Challenges-1

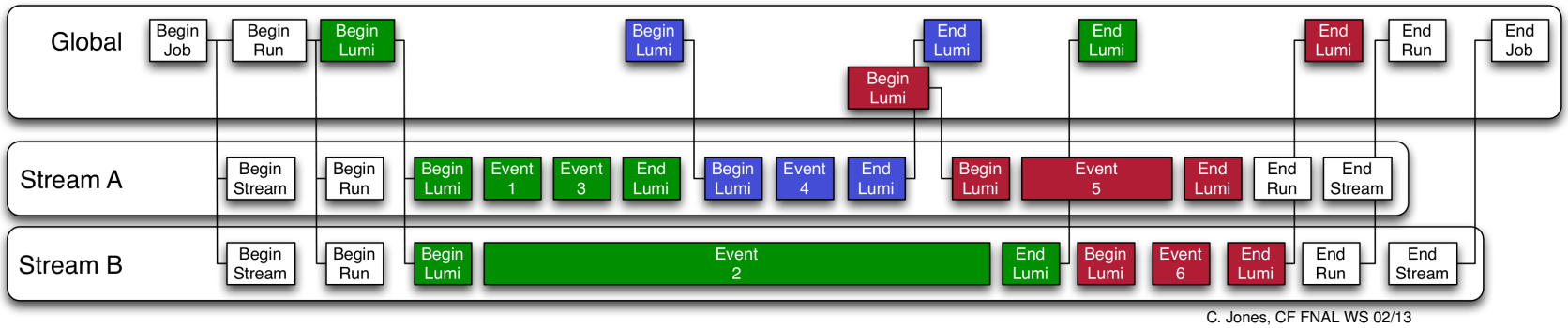
- Each have their own instruction set and require their own optimizations and code
  - NVIDIA $\longrightarrow$ CUDA
  - Xeon-Phi $\longrightarrow$ TBB, Cilk+, OpenMP, icc
  - ARMs $\longrightarrow$ MPI?
- Unfortunately one code can't run them all.
  - OpenCL is platform portable NOT performance portable
  - OpenMP might include Cilk+ and OpenACC to reduce platform dependent codebase
- Each have preferred problem types
  - Xeon-Phi $\longrightarrow$ highly vectorized computationally intensive parallel tasks...
  - NVIDIA $\longrightarrow$ Striped access, least branchy, identical code on large data sets...

Accelerator Challenges-2

- All require data to be copied to the device memory
  - Computational task should be large enough to reduce data copy overhead
  - Parallel portions of our code is usually not enough, need multiple events to improve efficiency
- EDMs used in frameworks are not suitable for accelerators
  - We code in terms of Array of Structs (AoS), computers like Struct of Arrays(SoA)
  - Usually need to convert EDM to SoA and back to use accelerators adding more overhead
- Multi-process approaches need to manage access to accelerators

Multi-Threaded Examples-1

CMS Multi-threaded framework CMSSW

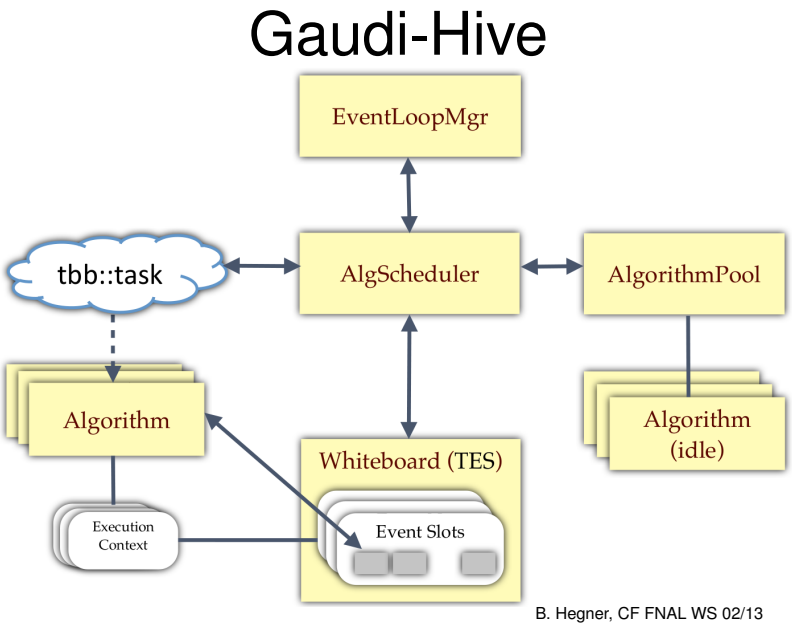


C. Jones, CF FNAL WS 02/13

- Multiple *Streams*, each working on a different event.
- Sub-event level parallelism whenever possible
- Global scope keeps shared modules(algorithms) such as output modules

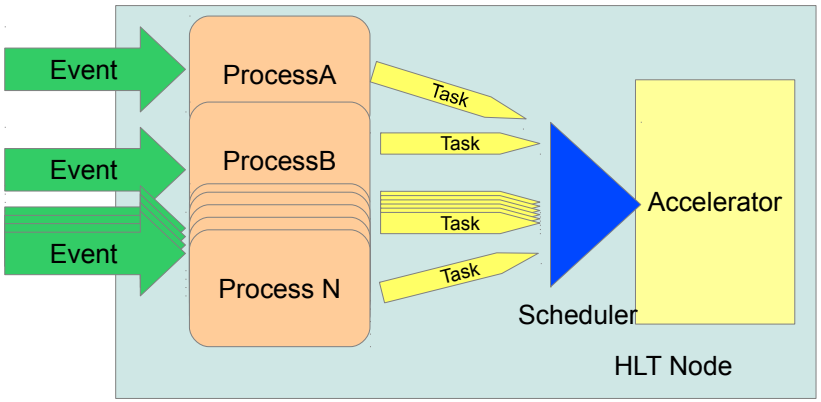
Multi-Threaded Examples-2

- Evolution of Gaudi
- Uses algorithm dependency graphs to exploit parallelism
- Multiple events on flight
- Keeps multiple copies of algorithms/modules (clones) to exploit event level parallelism (if algorithm supports)
  - Analogous to 3-lane highway, 10-lane toll-booth
- Supports sub-event level parallelism in algorithms/modules.



Process Based Examples-1

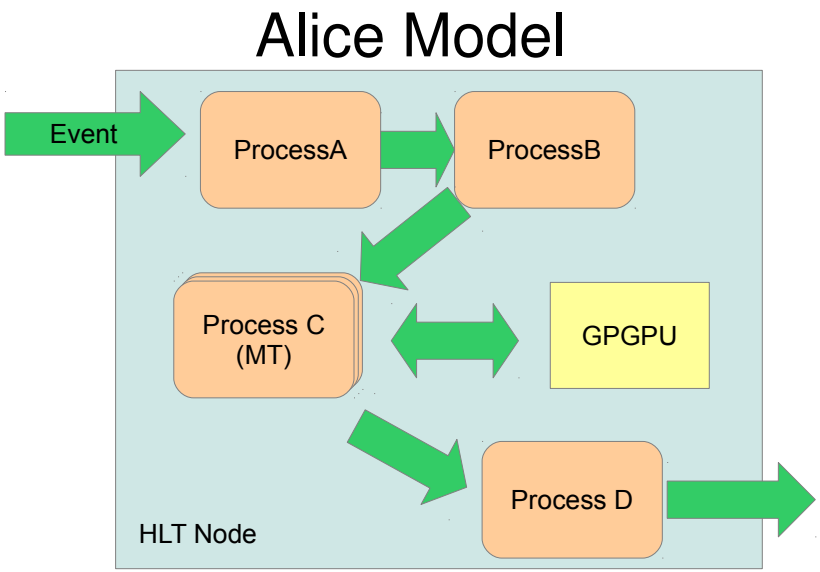
- Multiple processes receive events
- They send specific tasks suitable to accelerator to Scheduler/Gatekeeper process
- Gatekeeper process arranges tasks to suitable chunks and executes them on accelerator and returns results
- Processes keep working other tasks until accelerator results received



- Helps studying algorithm level parallelism
- Being investigated in ATLAS and LHCb

Process Based Examples-2

- Event goes through different processes in the node
- Each process handles a specific task
- One of the processes execute task on accelerator card (GPGPU)
- Results are sent when process chain completes
- Already in production in ALICE
- Might become more important if ARM/Atom servers dominate the market



Compilers

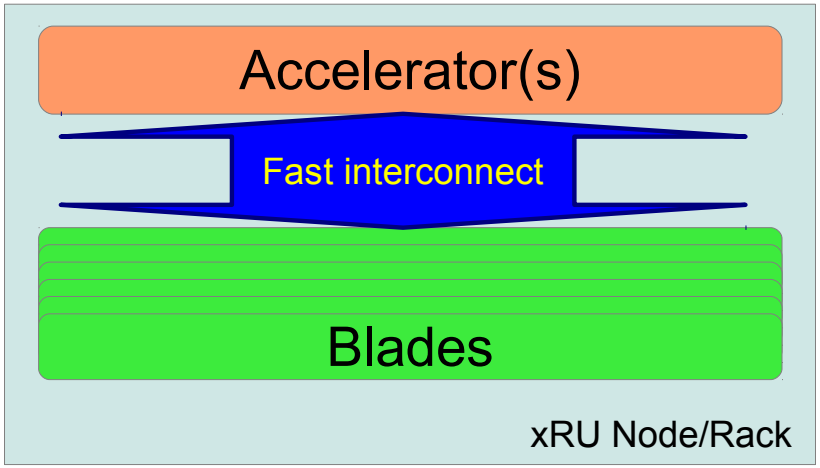
- Hardware manufacturers are pushing for different standards, favoring their own products
  - CUDA, OpenMP extensions, OpenACC, Cilk+,OpenHMPP...
- They are aware of the programming complexity and code portability issues
  - Substantial amount of expertise required to maximize the benefit.
  - Software developers don't want to invest in developing large code bases optimized to a specific hardware
- OpenCL is a couple of years behind the hardware
  - each manufacturer has its own implementation
- Extensions of OpenMP will probably help
  - But it took quite long time for OpenMP itself to mature
  - Might not be ready in time
- Need to keep our eyes open for new compilers (LLVM, CAPS...), libraries, SEJIT-like approaches

Revolutions?

- Hardly any
  - don't expect magical compilers or hardware
- Systems are becoming extremely heterogeneous, more and more specialized units
  - Compilers may help with diversity of the hardware
    - unlikely to convert arbitrary data structures to hardware optimized structures
  - We should look for the commonalities between units
- Our problems are a mixture of different tasks
  - can we exploit specialized hardware to full extent?
- We need to use Array of Struct of Arrays(AoSoA) to maximize the benefit from hardware!
  - All performance metrics are given on operations with flat data arrays
  - We have to reconsider our EDM
    - too much human oriented!

Ideas

- Network and memory bandwidth will most likely to improve
- Even with multiple events on flight, one specialized unit per node might be unfeasible.
  - possibility of a dedicated accelerator per multiple nodes
- Should we keep distributed designs in mind?
- Detector hardware will improve as well
  - Continuous streaming of partially built events?



- Blades are already providing fast interconnects
  - Inter-node communication might catch up with intra-node
- Process level pipelining with specialized processes might become feasible and more efficient

Summary

- Many-core, wide-SIMD and memory limited systems seem to be the hardware we will have in LS2
- Diversity of hardware make programming and decisions complicated
- Multi-threaded event and sub-event level parallel frameworks looks like the way to go but other designs also have benefits
- We should start thinking more computer-like to maximize the benefit from hardware
- Compilers and standards might hide heterogenity partially but will not solve our problems

Thank you for your attention

Thanks to

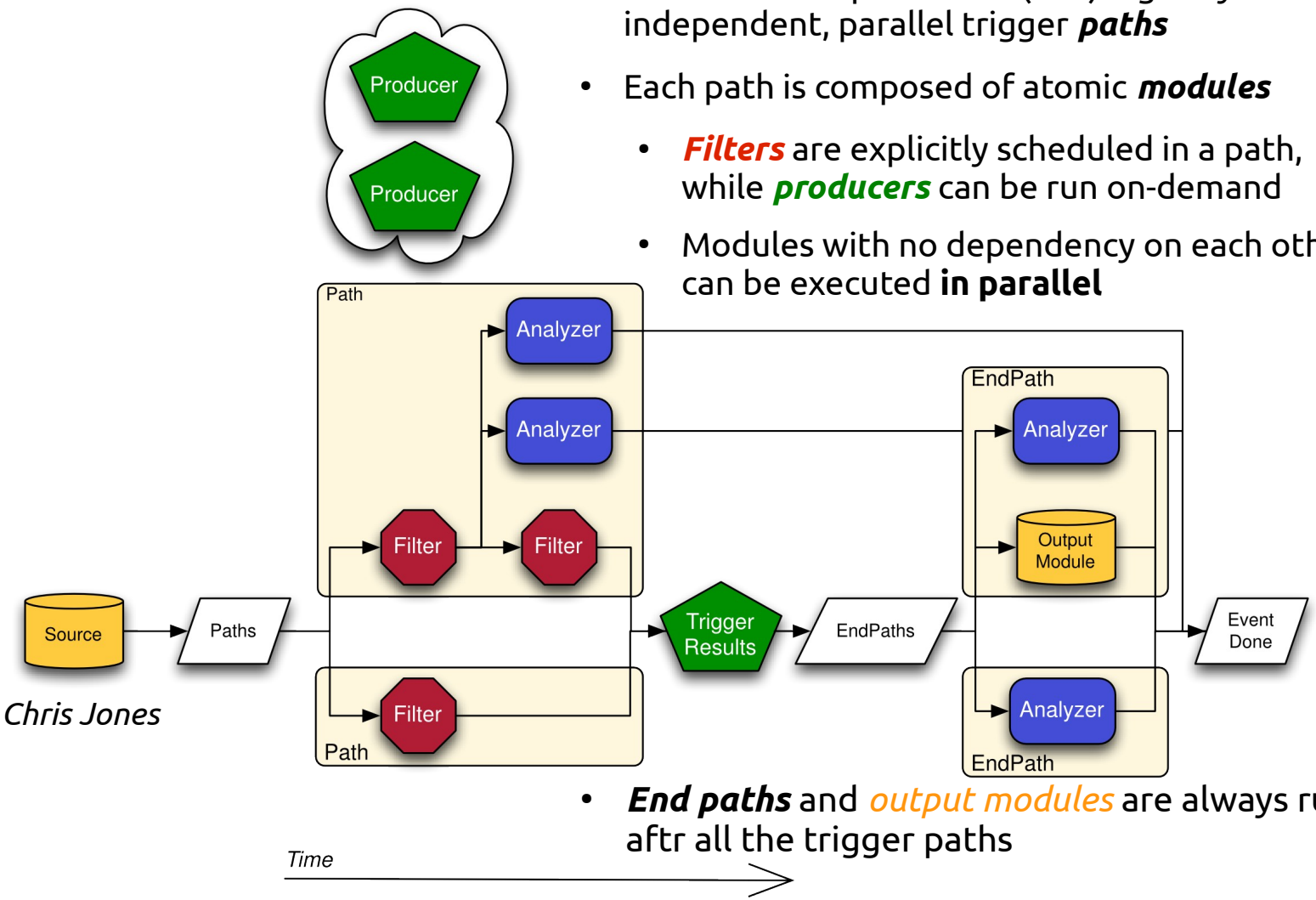
- Andrea Bocci,
- Thorsten Kolleger,
- Niko Neufeld,
- Werner Wiedenmann,

for discussions, explanation of designs, plans and ideas of their respective experiments.

Backup (CMSSW Sub-event Parallelism)

Sub-event parallelism

- The HLT is composed of O(500) logically independent, parallel trigger ***paths***
- Each path is composed of atomic ***modules***
  - ***Filters*** are explicitly scheduled in a path, while ***producers*** can be run on-demand
  - Modules with no dependency on each other can be executed **in parallel**



- ***End paths*** and *output modules* are always run aftr all the trigger paths

Chris Jones