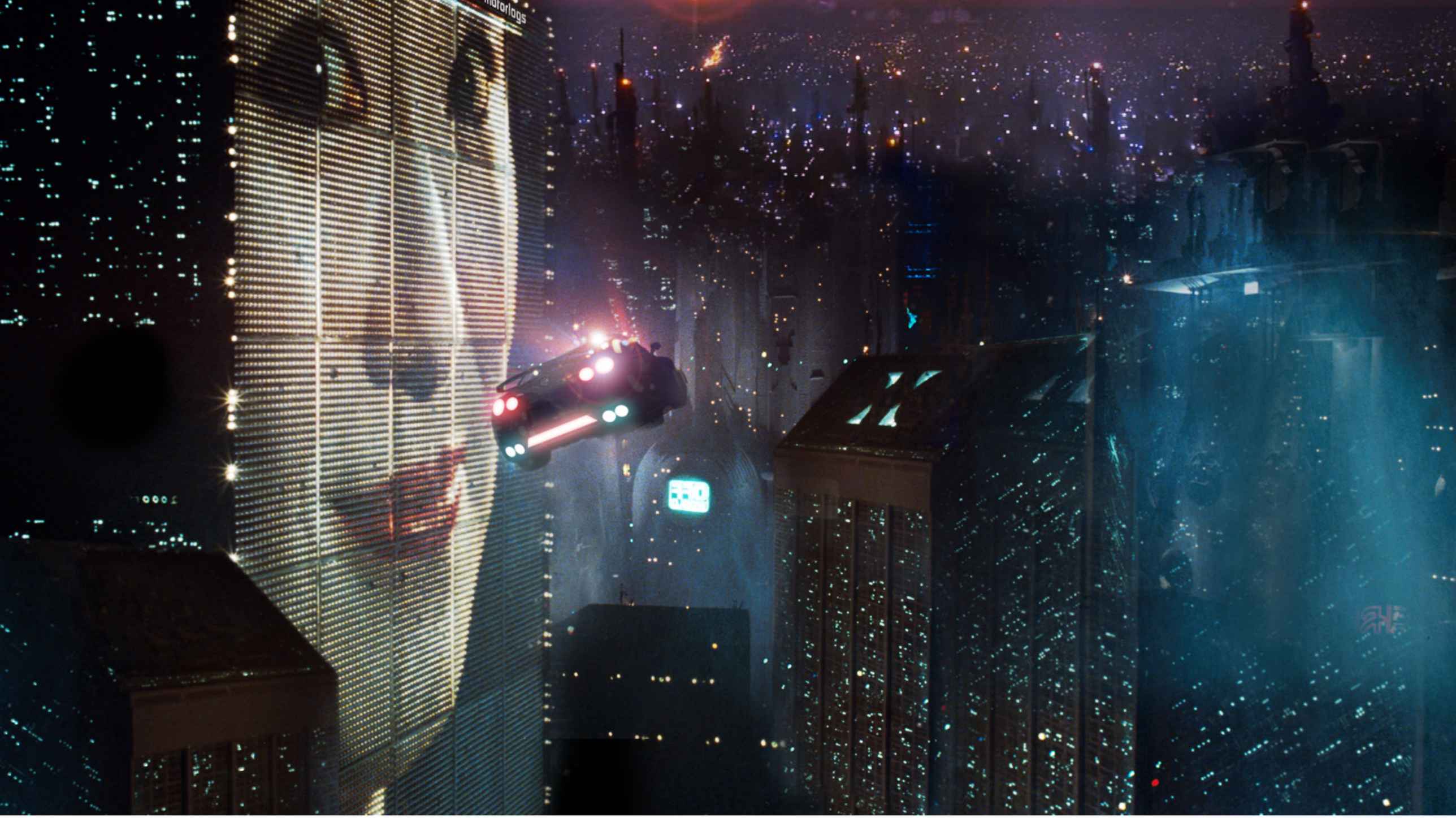
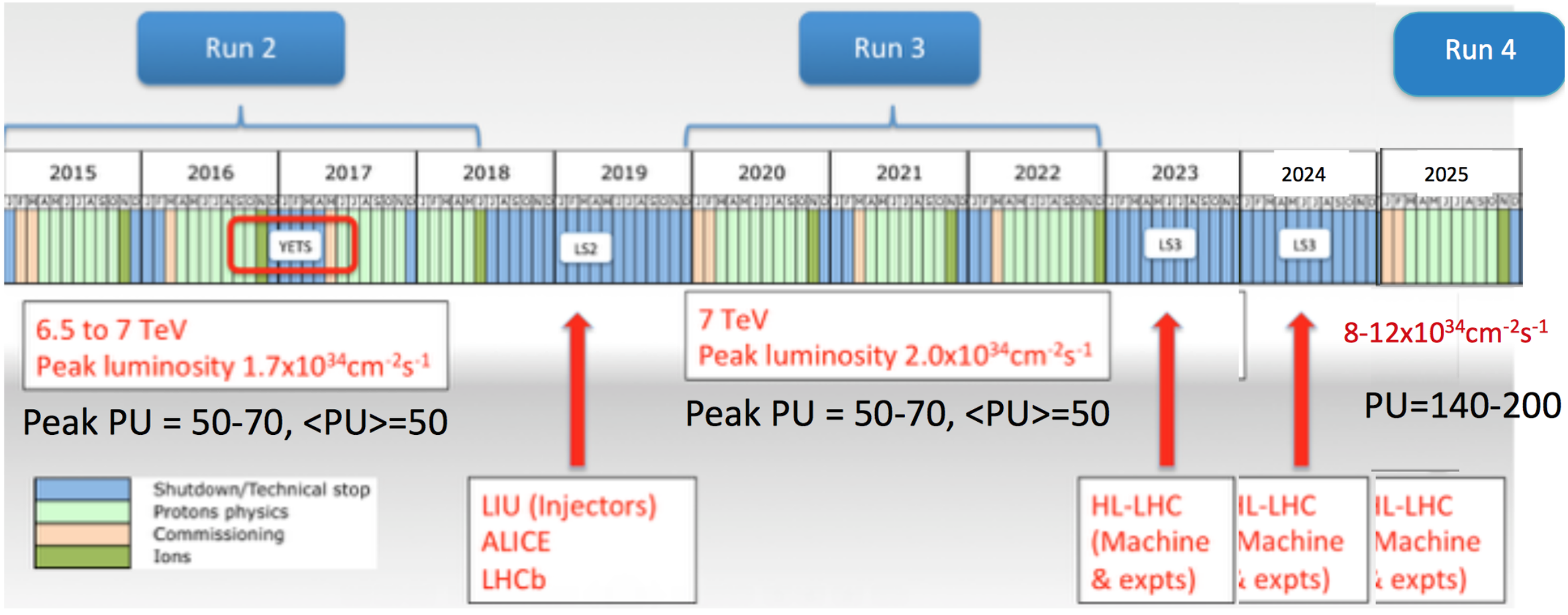


Computing Model Evolution

Peter Elmer - Princeton University

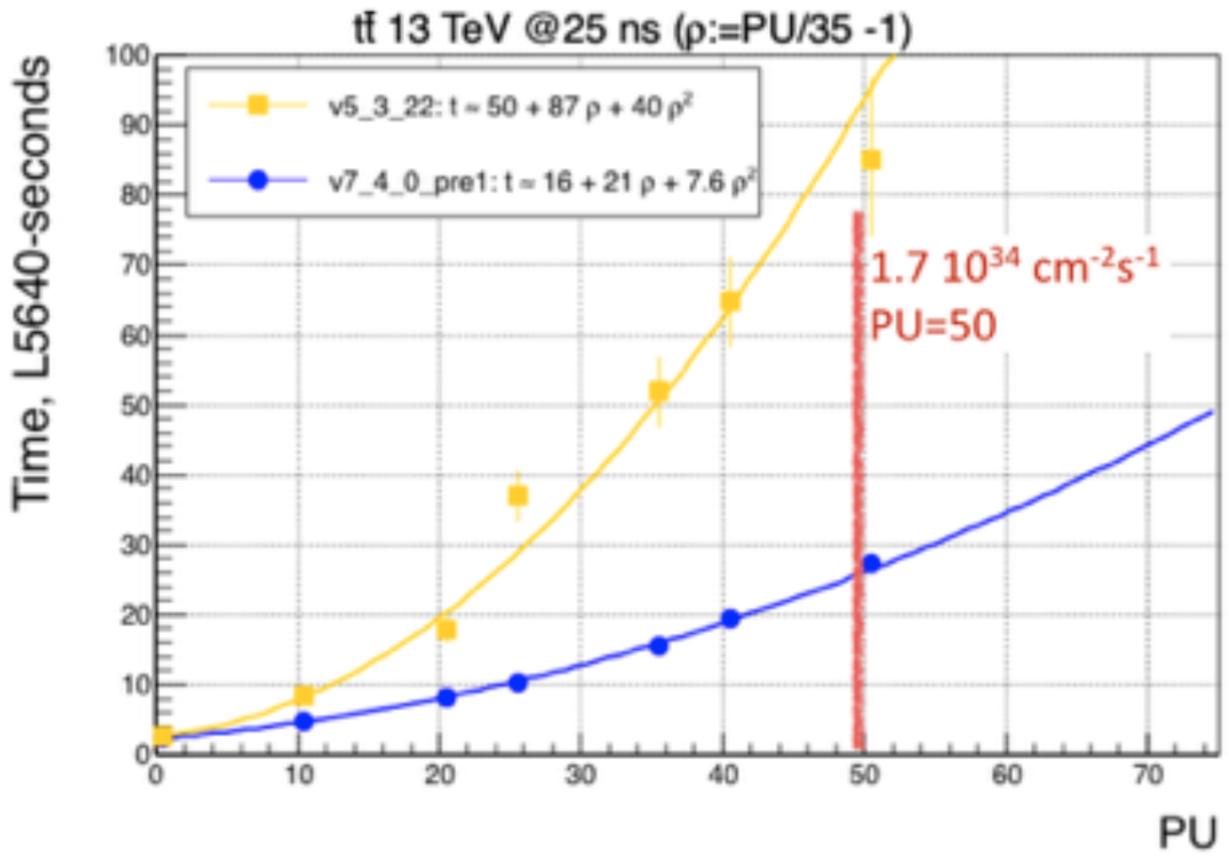
S&C Upgrade Context



- We have 10 years to prepare for the very challenging HL-LHC conditions of Run 4.
- There will be a step function in expected pileup.

S&C Upgrade Context

- Run 4 need another factor of 50…
  - Input pileup of 140 - 200 to the parameterization of latest technical RECO performance.
  - RECO time w/ no improvement: factor 5 compared to Run 2
  - Rate: ~10 kHZ, factor 10 compared to Run 2



- No way we can buy ourselves out of this with computing purchases.
- We will have to revamp the computing model and improve the software.

Background Material

- The US has driven a lot of the larger changes in CMS computing over the years.

- To get a flavor of various ideas USCMS groups find interesting in terms of R&D and possible upgrades, see the S&C session at the recent USCMS Upgrade workshop at Caltech:

- https://indico.cern.ch/event/377754/

- I'll cover some of these topics here

Main themes driving S&C R&D and Upgrades

- Better performance and scalability, achieving Moore's Law or better gains in throughput/unit-cost over time, much more emphasis on power use and density

- More efficient use of existing resources and access to additional (opportunistic, HPC) resources. Simplification and flexibility of managing resources.

- Enabling analysis with larger datasets, higher Pile-Up (more complex events), higher trigger rate, etc.

- **Both evolutionary changes and "What will be the computing model for HL-LHC?"**

Needed R&D and Upgrade activities

- Tracking on heterogeneous many-core resources, on heterogeneous worldwide resources
- Trigger on many-core resources, potentially homogeneous at the pit, heterogeneous elsewhere
- Simulation (G4) on heterogeneous resources, NLO generators
- Resource scheduling, I/O, event processing frameworks for heterogeneous resources, parallel processing models
- Tracking and other reconstruction algorithms at high pile-up
- Tools and infrastructure (profilers, development tools and models, etc.)
- More focus on power use, in addition to raw performance

Power limitations for processors

- Over the past ten years processors have hit power limitations which place significant constraints on "Moore's Law" scaling.
- The first casualty was scaling for single sequential applications, giving birth to multi-core processors.

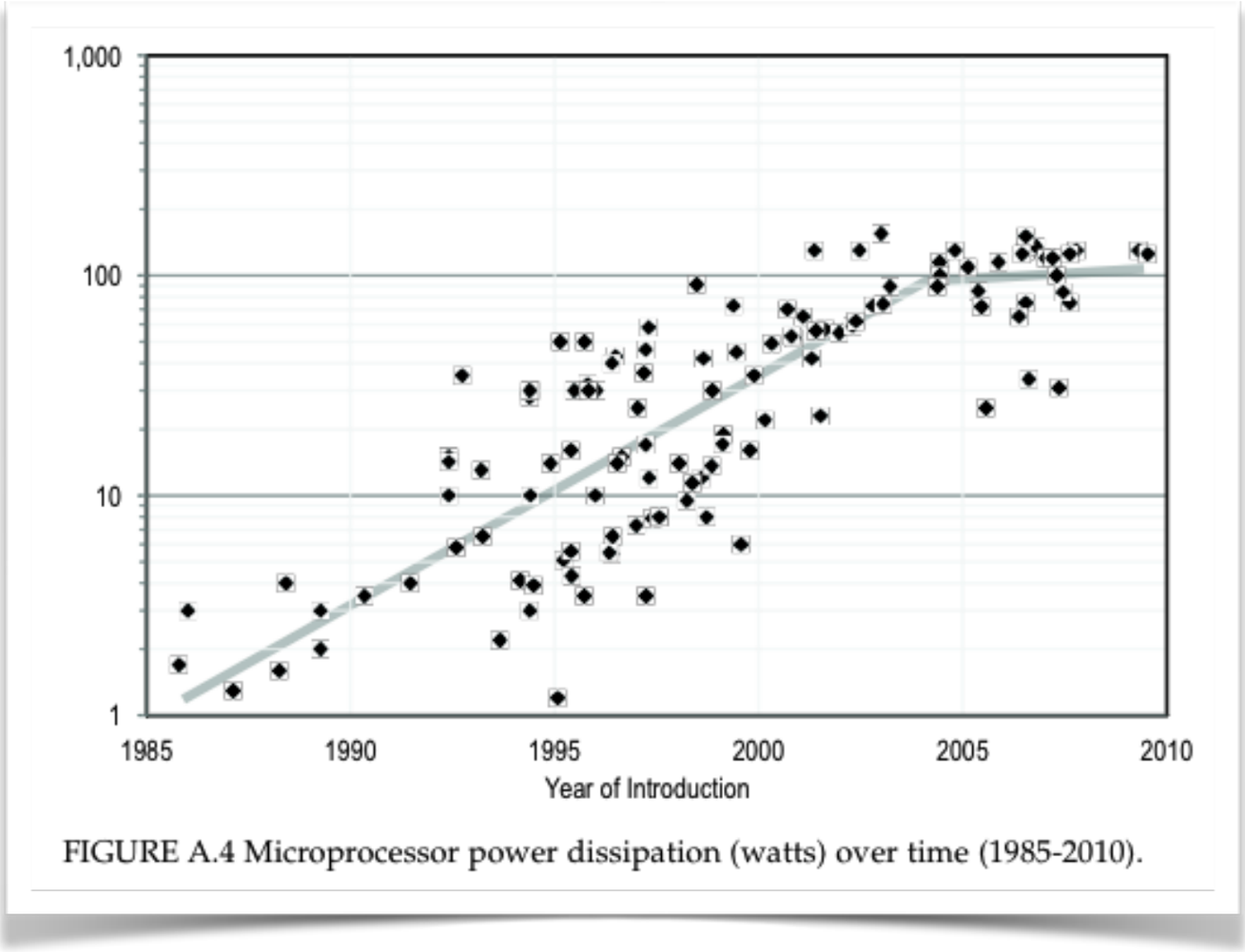


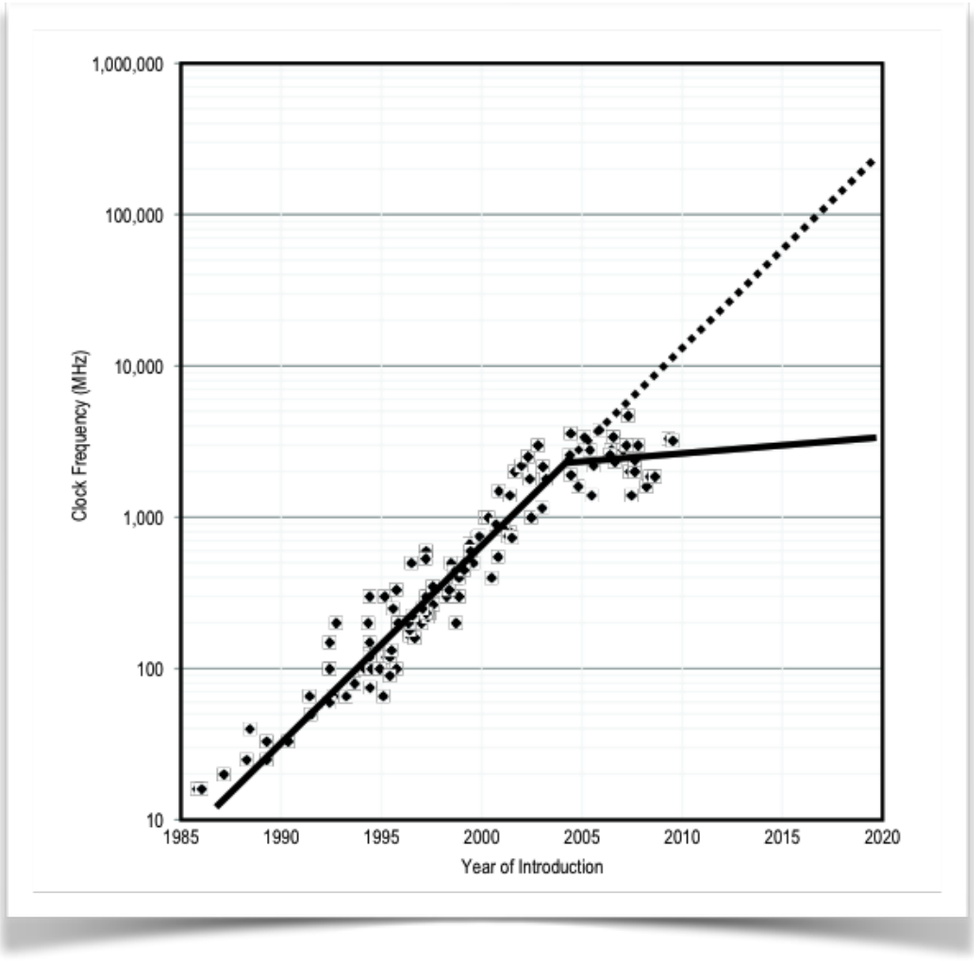
FIGURE A.4 Microprocessor power dissipation (watts) over time (1985-2010).

From: "The Future of Computing Performance: Game Over or Next Level?"

The Future of Moore's Law

- Even multi-core, implemented with large "aggressive" cores is just a stop-gap. The power limitations remain. The focus is shifting to performance/watt, not just performance/price.

- **Overall performance/$$$ growth dropped from 40+%/year before 2005 to 20-25%/year in more recent years**



From: "The Future of Computing Performance: Game Over or Next Level?"

arXiv.org > hep-ex > arXiv:1308.1247

High Energy Physics - Experiment

The Need for an R&D and Upgrade Program for CMS Software and Computing

Peter Elmer, Salvatore Rappoccio, Kevin Stenson, Peter Wittich

(Submitted on 6 Aug 2013)

Moore's Law scaling and LHC challenges

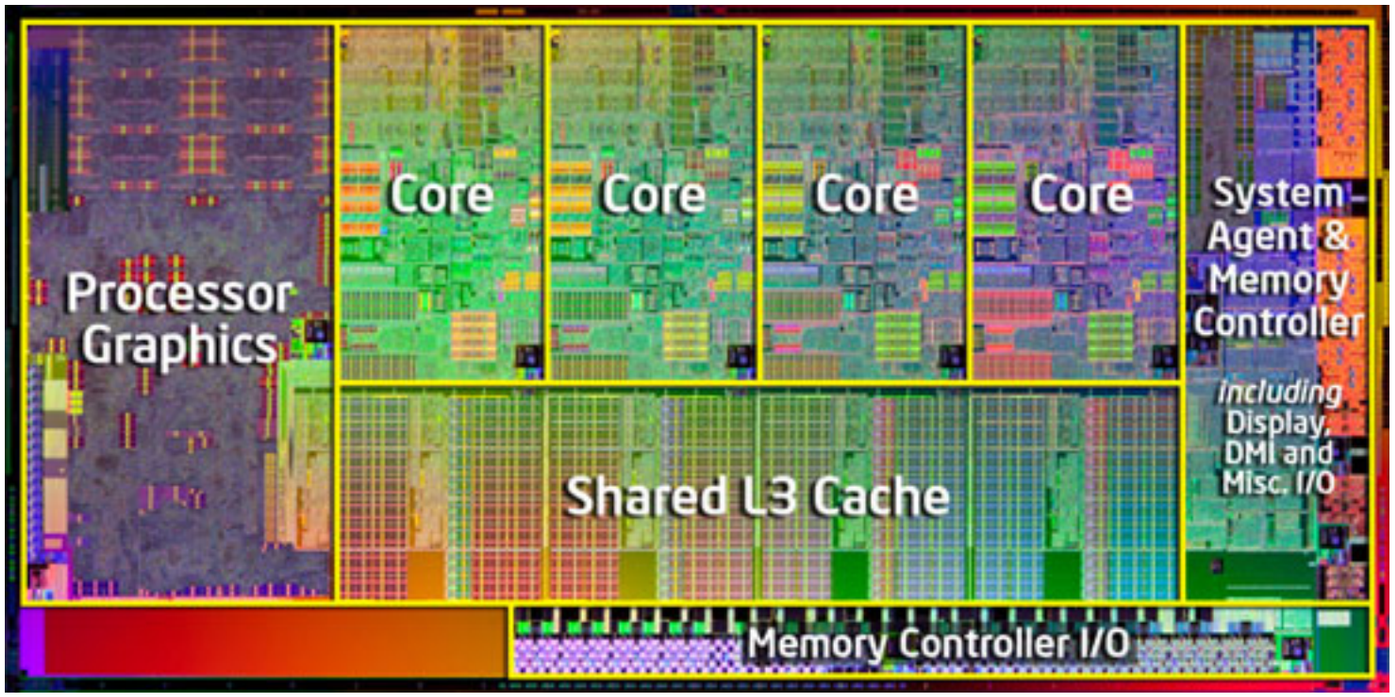
- Through about 2005, processor performance per unit cost increased at about 40% per year. CERN IT quotes 20-25% per year in recent years. This is the difference between a factor of ~6 and a factor of ~30 over 10 years.

- Taking into account increased offline reconstruction times with PU, increased trigger rate, etc. one easily finds CPU needs growing by 10^3-10^4, driven largely by tracking.

- Thus both the technology and the LHC needs push towards investigating higher perf/$ and perf/Watt technologies

The Future of Moore's Law

- Even multi-core, implemented with large "aggressive" cores is just a stop-gap. The power limitations remain. The focus is shifting to performance/watt, not just performance/price.



- On top of this integration of components on-die will continue.

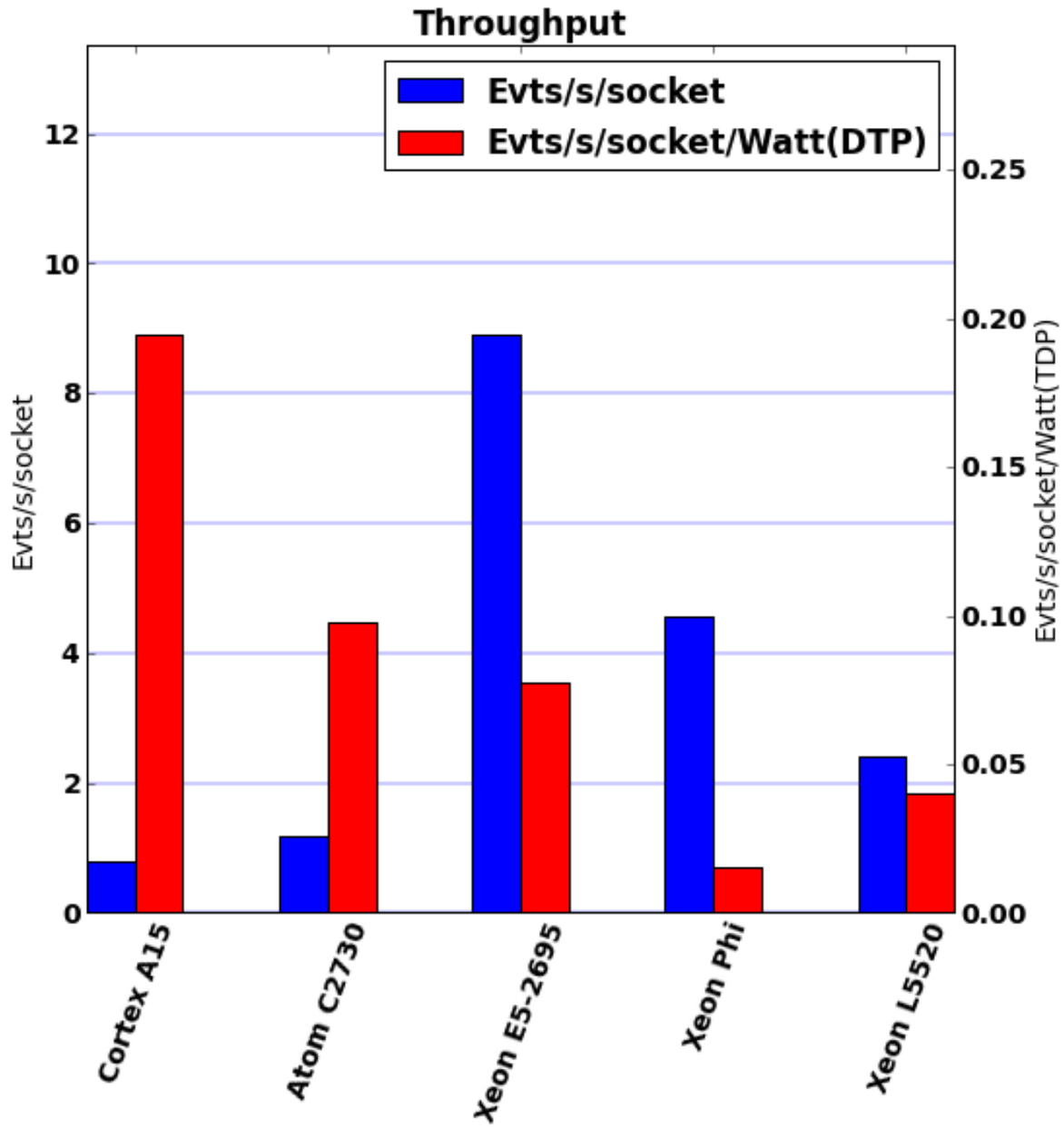
Indicative technologies today

- ARM processors - low power, higher performance/watt, simpler processor cores

- Graphics processing units (GPU) - e.g. Tesla from NVidia, coprocessors

- Xeon Phi (Intel MIC architecture) - coprocessor with ~60 lightweight cores, each with big vector units

- Likely that the next step will be heterogenous mixes, like mobile



ARM processors

- The different IP Model and mobile market dominance could imply an eventual "commodity" challenge to Intel in the way that x86 ran over the RISC workstation players (DEC, SGI, Sun, ...) 15 years ago?

- Also ARM is a key ingredient in the strategies of other players like AMD and NVIDIA, and perhaps big players for computer centers. Interesting technology evolution even for general purpose cores: big.LITTLE.

- This is a relatively easy one, requiring only a software port (done for ARMv7 and ARMv8 already done, we are also building a small demonstrator cluster)

GFlops/year by processor type

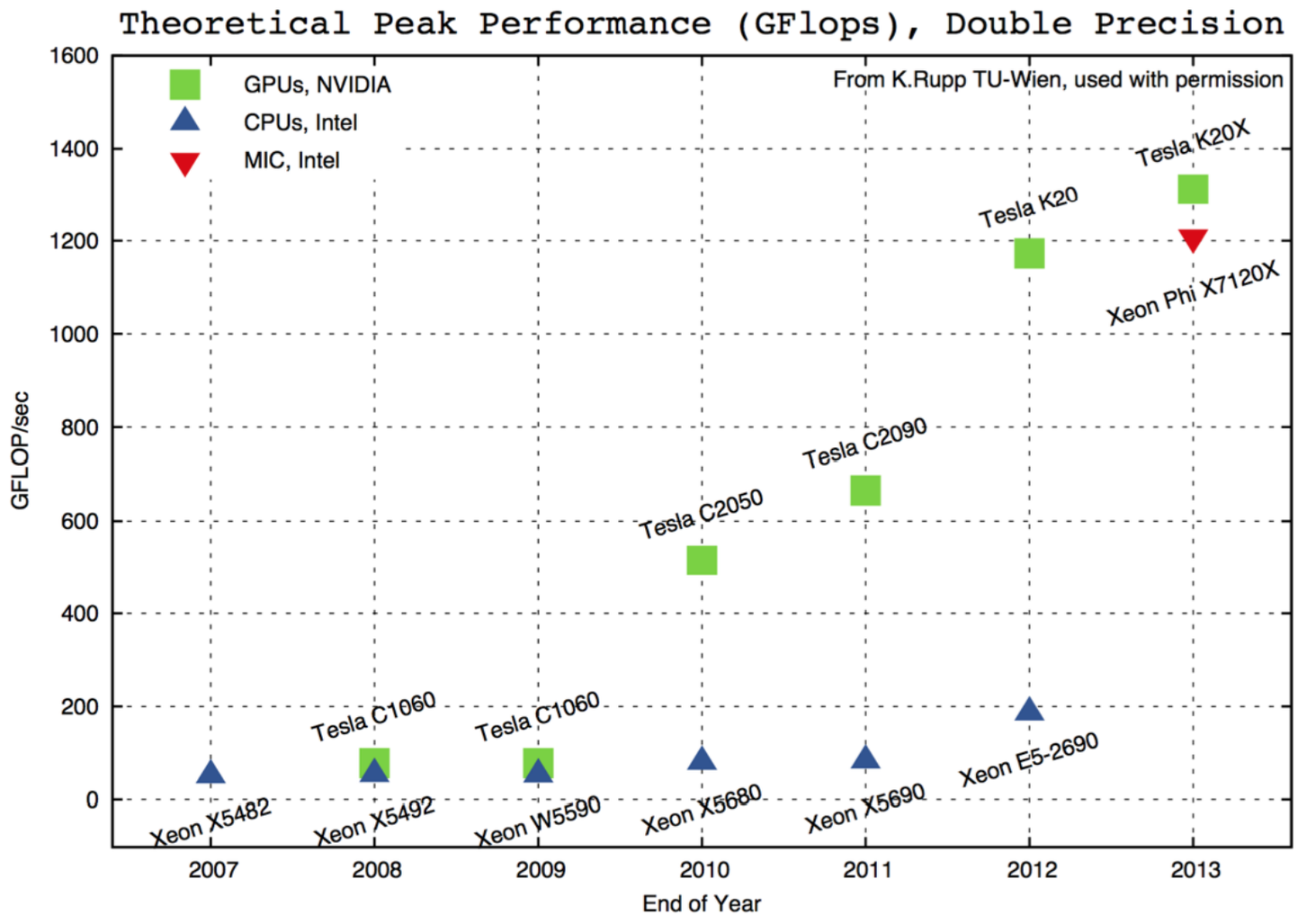


Figure 2: Time evolution of performance (GFlops) for several high-end CPUs, GPU models and the Intel Xeon Phi. (Note growth rate of Intel CPUs vs accelerators.)

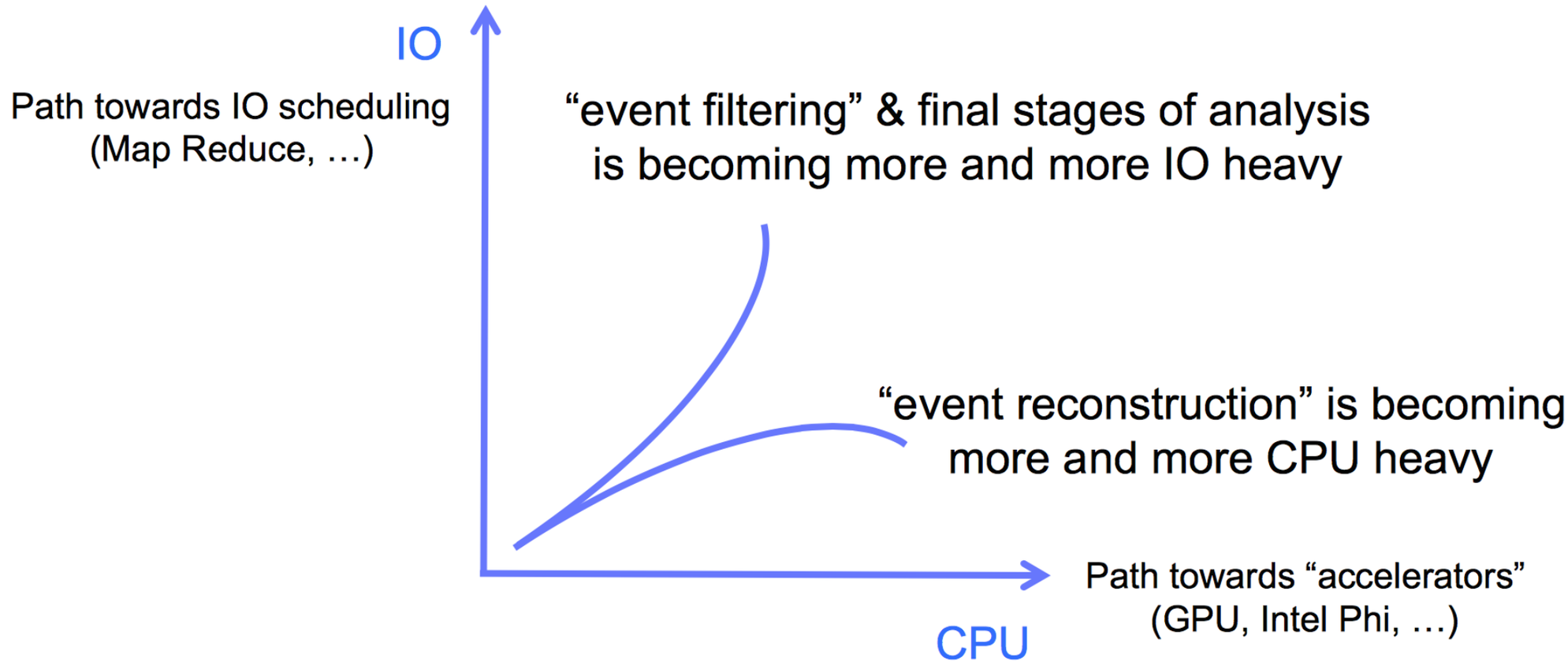
Event Processing Framework

- We are currently deploying the multicore-capable framework. This is an important step forward in terms of exploiting multicore processors efficiently, but is also an important step towards eventual exploitation of heterogeneous processor configurations.

- For example, if we want to deploy software capable of using both the general purpose processor cores and specialized cores (e.g. GPU) eventually the framework will need adaption to do some sort of resource scheduling for the different types of cores. Potentially it will need to manage data in different ways and in sufficient quantity to keep specialized cores busy.

- On the time scale of 3-5 years, this is probably possible, but more input is needed from actual algorithmic implementations from R&D projects before defining how the framework should manage them.

Data placement, storage and caching



The impact of this dual trend on “shared commodity clusters”
as well as DHTC in general is as yet unclear.
We’re interested in preparing in both directions.

(Figure swiped from FKW's talk at USCMS upgrade workshop)

Storage

- Recently deployed/developed technologies (AAA/xrootd WAN reads and proxy cache, Data Popularity, Dynamic Data Placement) allow us to move beyond the Run1 model of static placement and strictly local access to storage from jobs.

- The obvious next step would be to evolve the actual (geographical) distribution of storage, and the mix of CPU and storage being purchased. Eventually this could also include an evolution of how many T1's and T2's we have and evolution of their purposes.

Data processing models

- The Run1 model (and historical batch model) has been based on predefined notions of "jobs" as well as their expected connection to input and output from a single execution on a WN.

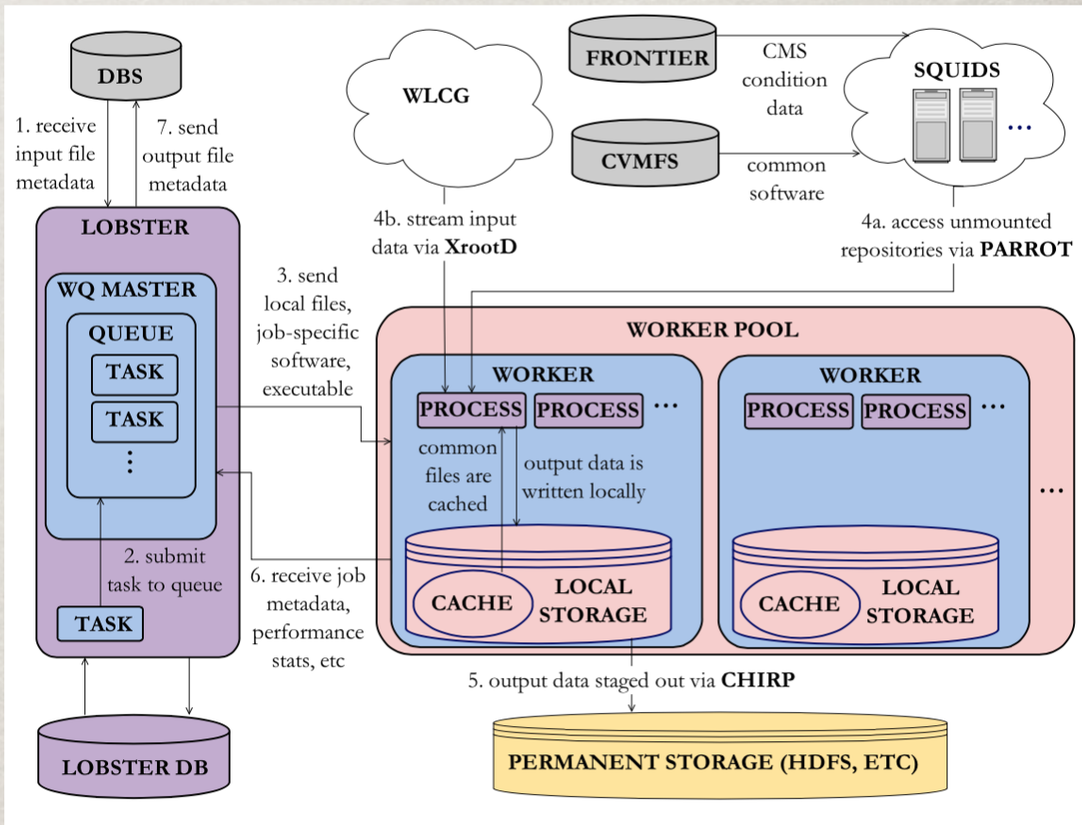
- This simplified some aspects of data management, but the decoupling of single-core wall-clock throughput from Moore's Law in ~2005, as well as ideas for opportunistic use, have made this a somewhat rigid constraint.

- A number of ideas are floating around to change this.

Data processing models - one idea

Lobster Architecture

- Main components
  - Scheduling: schedules and dispatches jobs
  - Data: managing input/output data and software
  - Execution: runs tasks on opportunistic resources
- Master-worker architecture



Checkpointing/Containers

- One additional place where we can be more agile is in the way we use resources, including both cloud and opportunistic resources. Two technologies are available today which could be used to increase flexibility: checkpoint/restart and containers.

- With *checkpoint/restart*, the idea is to pause a process to disk, and then restart from where it was in its processing at a later time (and potentially another machine)

- With *containers*, one can run different processes under the same kernel with different runtimes (e.g. SL6 on Ubuntu)

Checkpoint/Restart

- We have demonstrated in the past fairly efficient checkpoint/restart of CMSSW processes using the DMTCP package. Recent versions have "plugin" feature to properly close/reopen outgoing connections.

- We are currently attempting to get some scale tests done with the HLT farm, as a first place where this could be useful.

- This could easily be deployed sometime in the next 3-5 years, however the WM system would need adaption. We would need more experience at scale to understand the efficiency and reliability.

Containers

- Linux Containers (LXC) are actually a rather mature set of core linux tools used for doing resource (cpu memory) / and filesystem isolation of processes. They are different WRT a VM because the worker unit is still a standard process. Lightweight, low overhead and with almost immediate startup.

- Docker is a set of wrapper scripts around LXC utilities, which simplifies management of containers, introducing a all in one tool for building, deploying and running them. The same company offers also online repository of prebuilt images which users can extend and share (think of it as GitHub for deployable services).

- Docker is de-facto standard for large cloud deployments. Adopted by main industry players including Google, Microsoft and now Amazon. Breaks the "one-VM-one-Service" paradigm, resulting in better resource utilisations and provides faster turnaround by optimising image delivery (via layering of changes on top of each other).

- Given each container is supposed to run one single task, there is growing interests in orchestration of multiple containers and in running them on a cluster of unspecialised machines (e.g. fig, Apache Mesos, CoreOS).

Containers - How HEP could profit

- If batch system were docker aware, users could specify their preferred software setup in for of a docker image without need of a distributed FS or software pre-installation.

- Service deployments could be simplified by simply running containers on top of a unspecialised cluster (maintained, for example, by CERN/IT). Tools are already available to manage high availability, resource aware scheduling and large scale monitoring.

- Developers and Operations would share the same setup, making traditional difference between the two worlds much less of a problem.

Validation/Benchmarking

- Any eventual future heterogeneity of resources implies a set of validation challenges to insure that correct results are obtained independent of processor being used. It also implies greater complexity to benchmark any given set of compute resources to evaluate its potential, as applications will also vary in their effectiveness.

- The current processes for doing this remains some cumbersome and by hand.

- Any code written explicitly for heterogeneity needs to build-in validation tools for comparisons, and how HepSpec evolves will be quite interesting.

Summary

- A number of things are likely to change in the next 5-10 years:

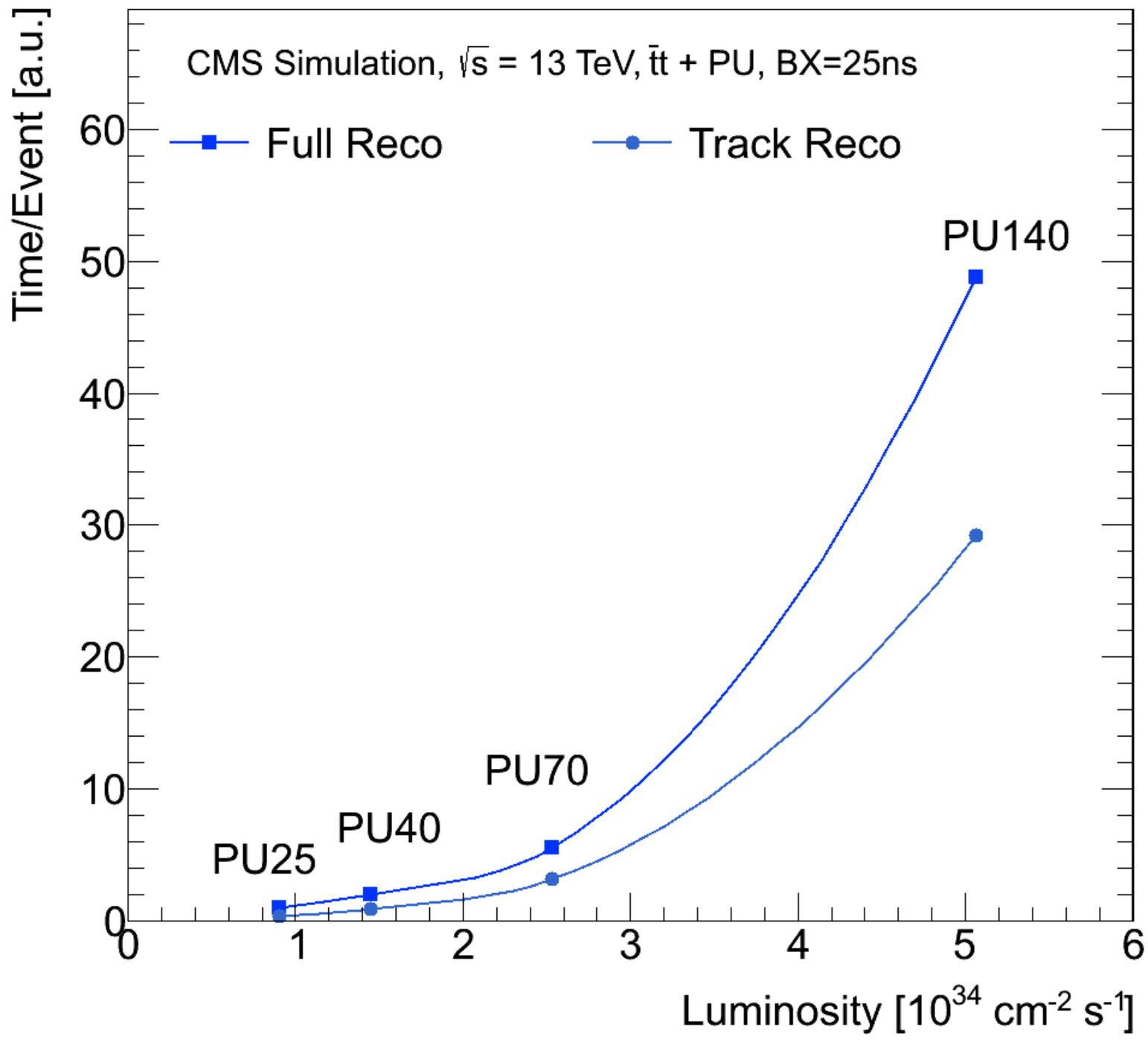
- Processor architectures will evolve to greater heterogeneity and require new programming models

- Storage use and architectures will evolve to be more dynamic and may not map to today's Monarc-ish computing system

- Data Processing models may evolve beyond old style batch

Backup Slides

Tracking Dominates Reconstruction



Track fitting on Xeon Phi (example)

- Exploitation of vectorization
- In this case, on standard Xeon and Xeon Phi
- Example 1M track fits

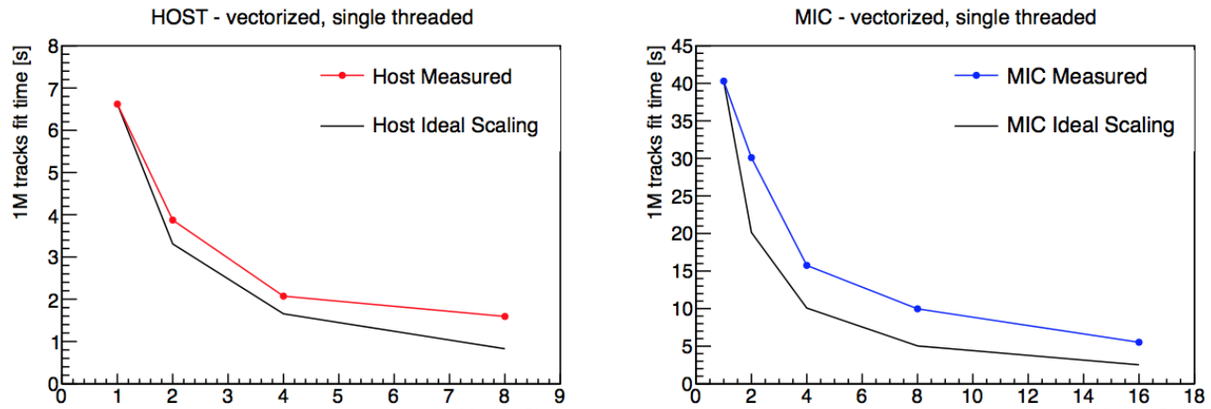


Figure 5. Timing results for vectorized code, as a function of the vector size, for host (left) and MIC (right). The results are compared to an ideal scaling described in the text. Significant speedups compared to serial code are observed.

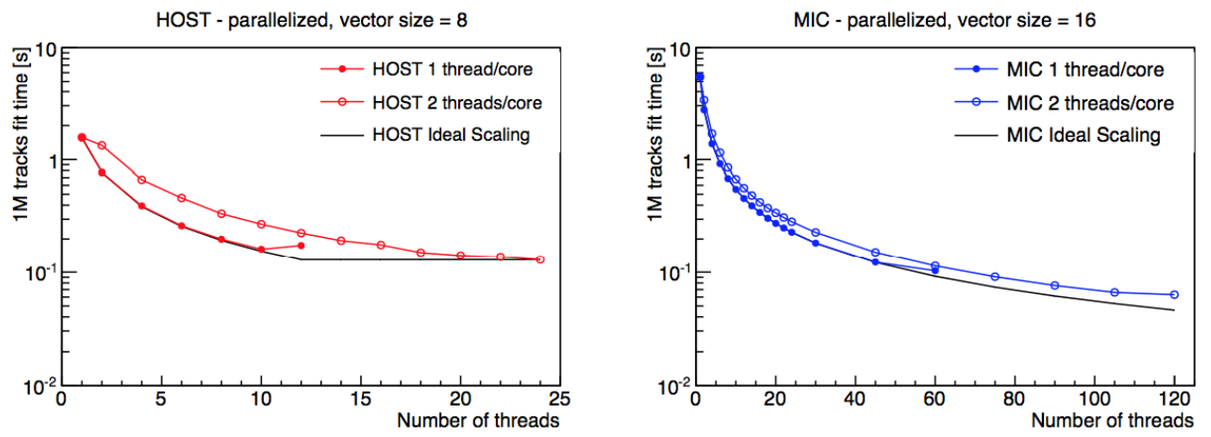
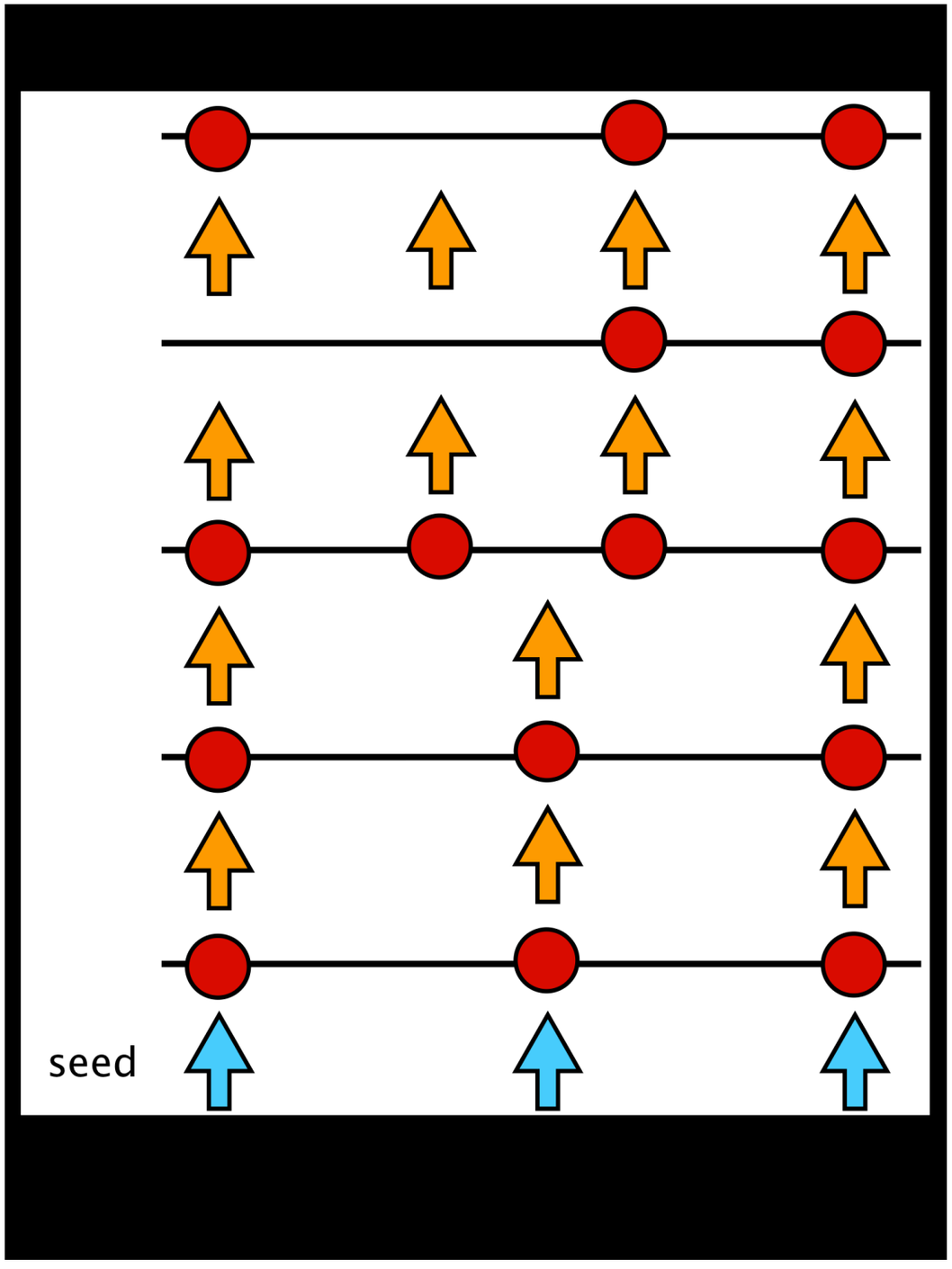


Figure 6. Timing results for parallelization tests, as a function of the number of threads, for host (left) and MIC (right). Two different methods of distributing threads across cores is shown, and compared to an ideal scaling. We observe ideal scaling when distributing one thread/core.

Track Building

- Most time consuming part of tracking.
- Efficient implementation for parallelization and vectorization made difficult by branching.
- With dedicated effort, this could be available for use within 3-5 year time scale, but R&D needed with real geometry to understand gains.



Simulation

- Dominated by Geant4
- Efforts at CERN (Geant V) and FNAL to prototype
- Dedicated efforts on geometry (USolids and VecGeom)
- Unclear to me at least when this will converge to something actually usable "in anger", but in principle could happen in the next 3-5 years

Software Development Model

- The model HEP has used for software development for the past 15-20 years has emphasized widespread contributions from the collaboration to the common software (CMSSW, for example)

- Less emphasis was placed on coding for performance than coding for functionality.

- It is an open question how this model will need to evolve with more complex programming requirements of parallel architectures.

Collaboration Engagement in Software