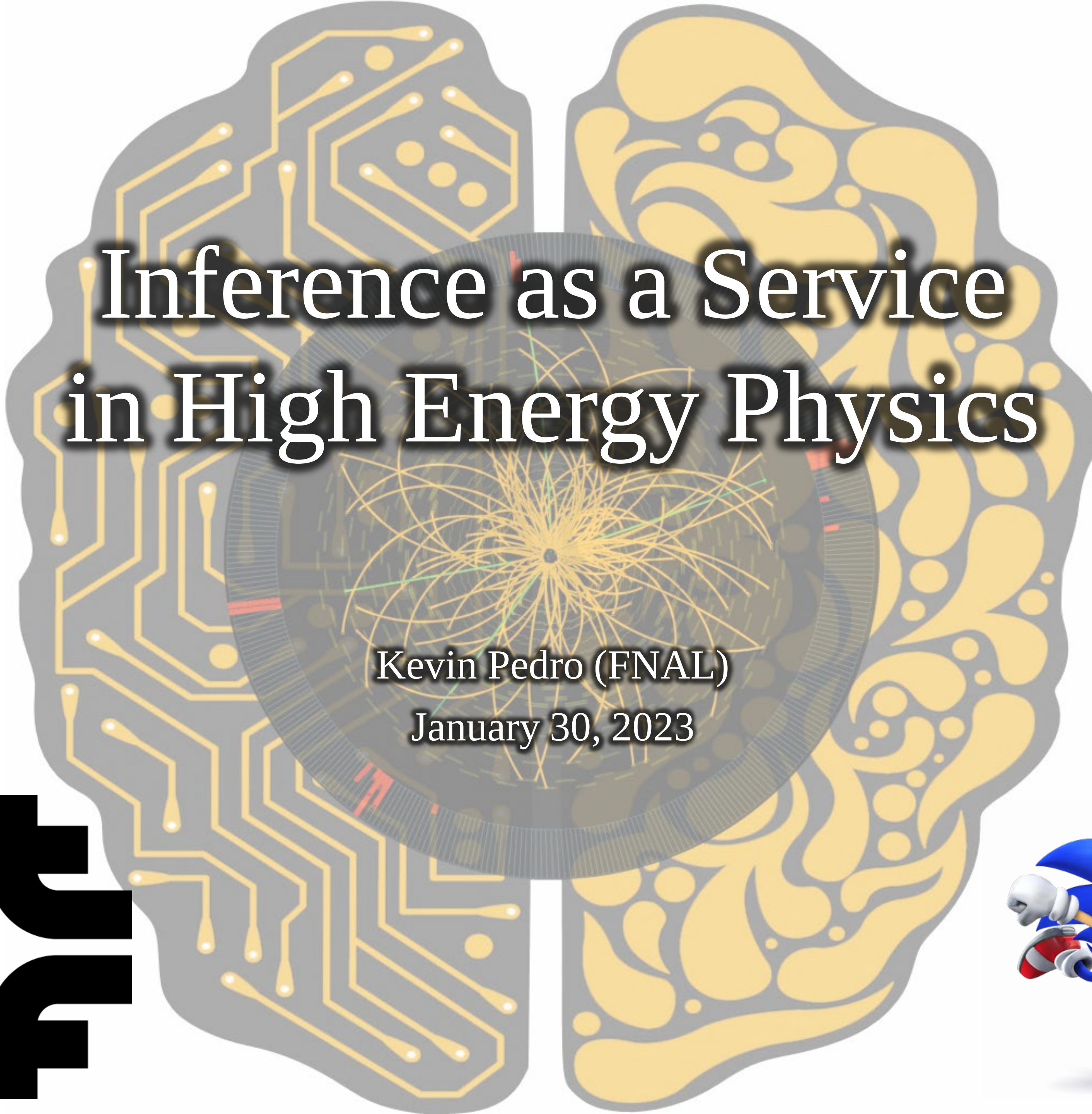
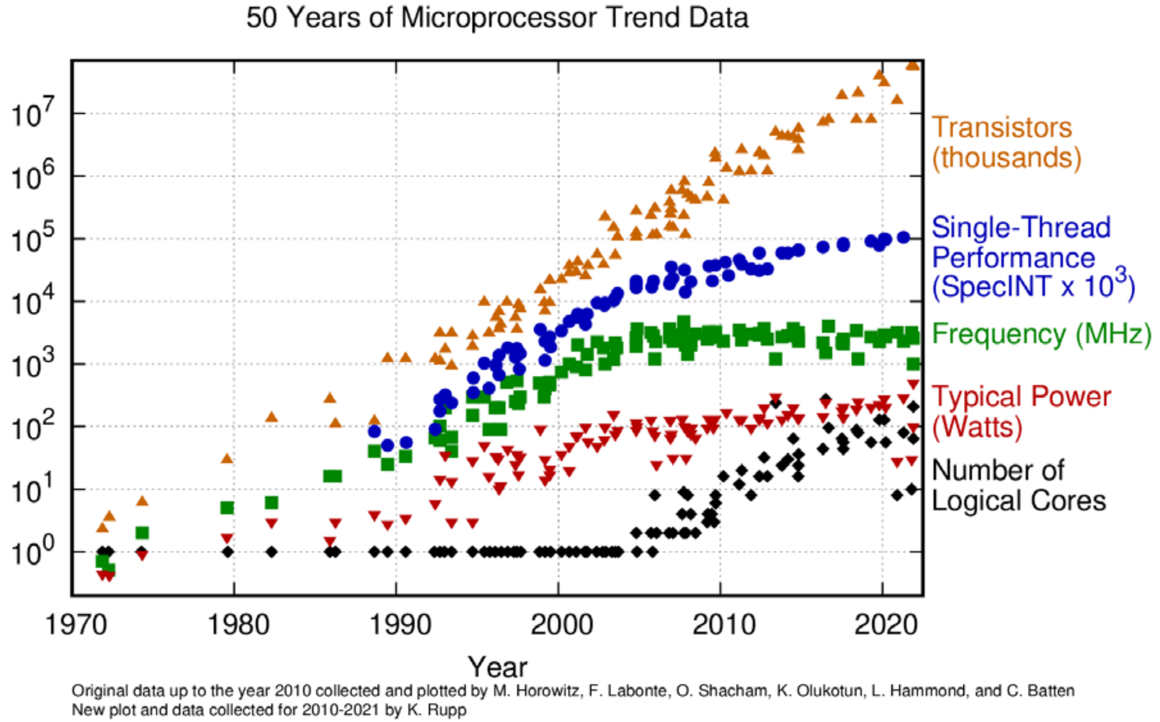


Inference as a Service in High Energy Physics

Kevin Pedro (FNAL)

January 30, 2023

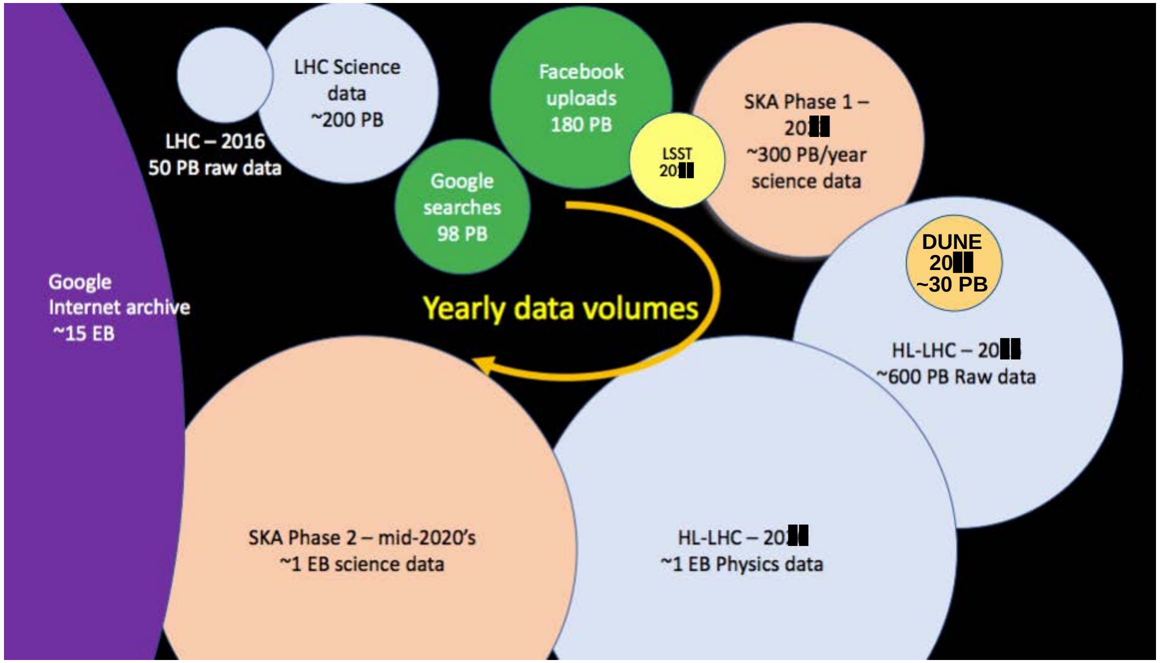
Computing in the Time of x86



- Modern era of computing has been dominated by x86-architecture CPUs
  - Enabled by Moore's Law (transistors double every ~2 years) *and* Dennard scaling (power proportional to transistor area)
  - The former continues, but the latter has broken down

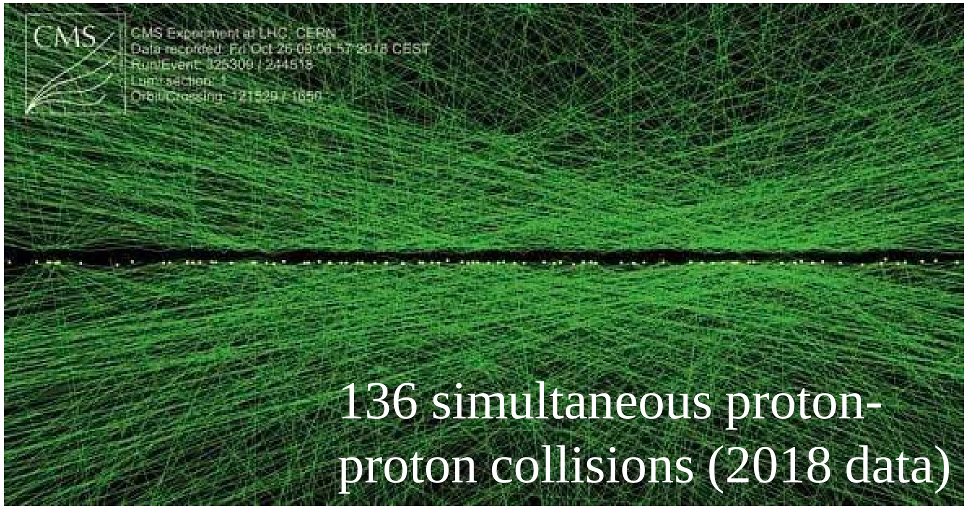
➢ Single thread performance has *stagnated*

More Data, More Problems

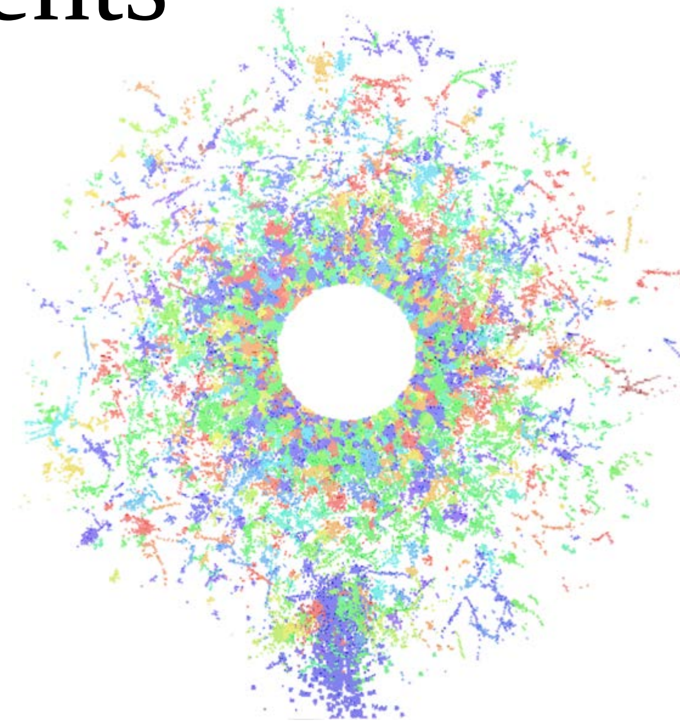


- Next-generation experiments will once again outpace industry data volumes
- Need to process 10× or more data
  - o …with only minor increases in general-purpose computing power
- *And*: data volumes alone don't tell the whole story

Complex Events

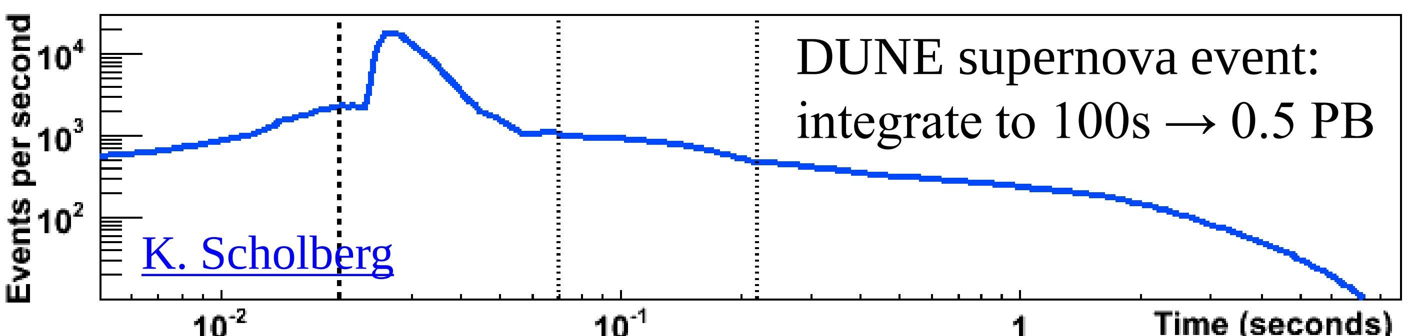


136 simultaneous proton-proton collisions (2018 data)



One event in CMS High Granularity Calorimeter w/ 200 simultaneous pp collisions

- Event complexity increases with detector size & complexity, beam intensity, new multimessenger frontiers
- But still… CPUs stay the same

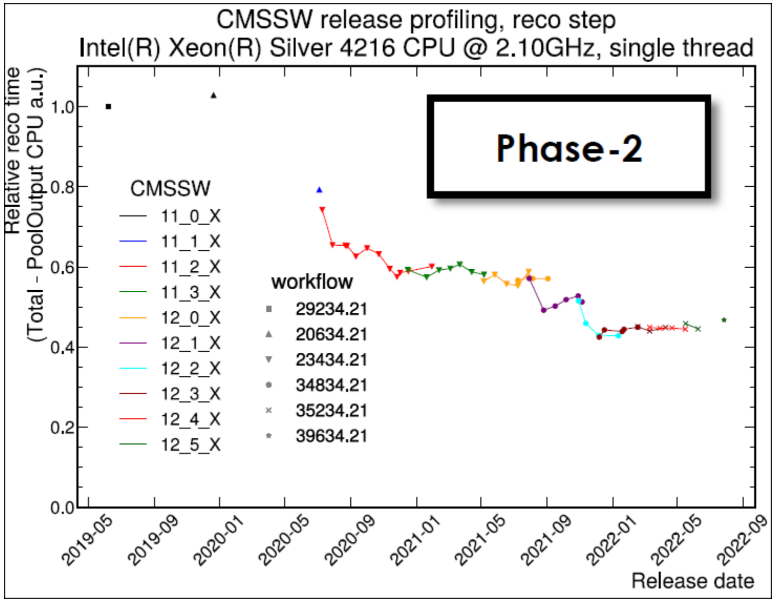


DUNE supernova event: integrate to 100s → 0.5 PB

Saved by Software?

Configuration	Relative CPU usage Minbias	$t\bar{t}$
No optimizations	1.00	1.00
Static library	0.95	0.93
Production cuts	0.93	0.97
Tracking cut	0.69	0.88
Time cut	0.95	0.97
Shower library	0.60	0.74
Russian roulette	0.75	0.71
FTFP_BERT_EMM	0.87	0.83
All optimizations	0.21	0.29

[CMS simulation, circa 2018]

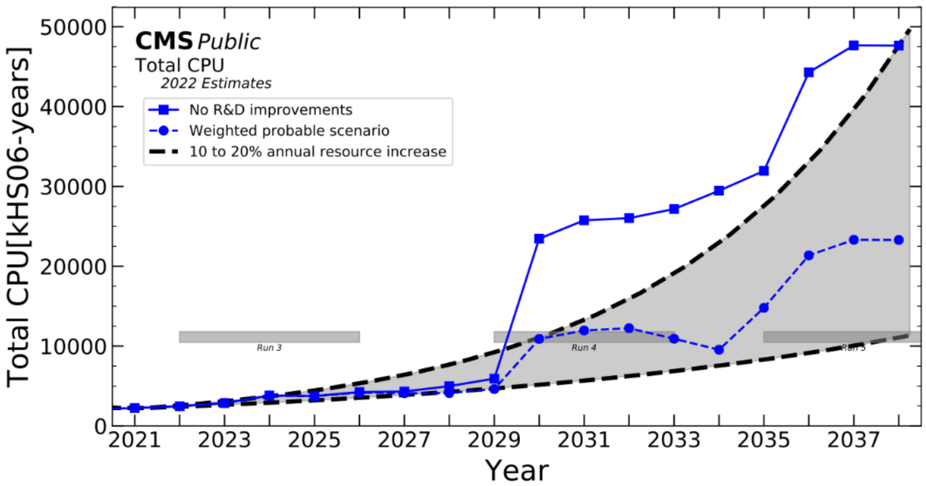


[CMS reconstruction, 200 pp collisions]

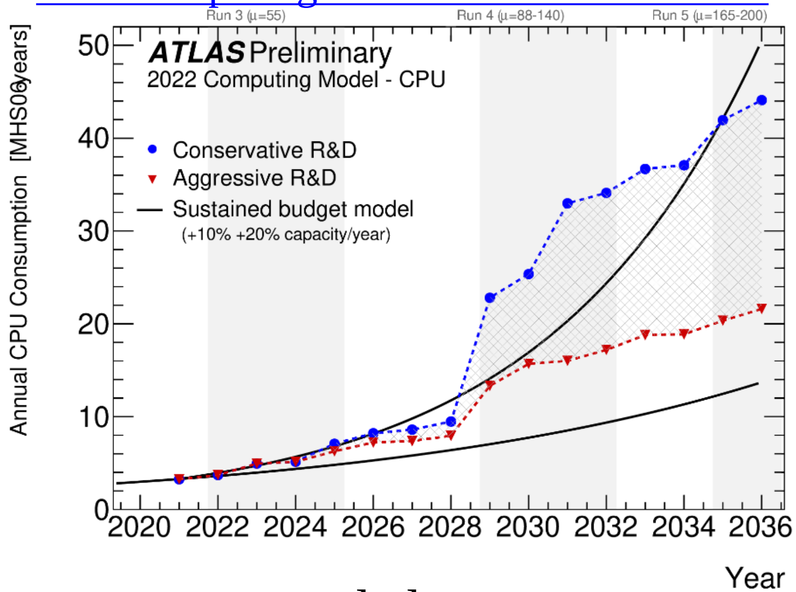
- Impressive and sustained effort to increase CPU efficiency
  - ➢ Process more events and execute more algorithms without buying more or better CPUs
- Low-hanging fruit gradually being picked
  - o Techniques like autovectorization, cache optimization, etc. can only be applied once…

HL-LHC Projections

[CMSOfflineComputingResults]



[AtlasComputingandSoftwarePublicResults]



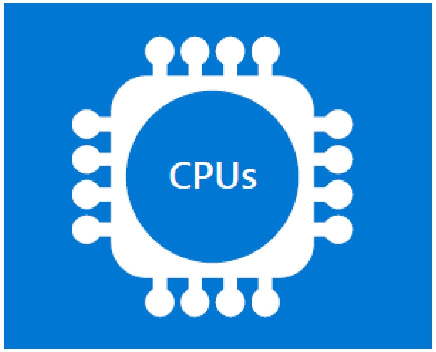
- Substantial continued software R&D improvements are needed
    - Even to fit into somewhat optimistic resource increase model
- Similar story for disk and tape
    - Potentially even more constrained: can delay CPU processing tasks, but once a disk is full, it's full
- Memory, network: projections more uncertain, but undeniably finite resources

Heterogeneous Revolution

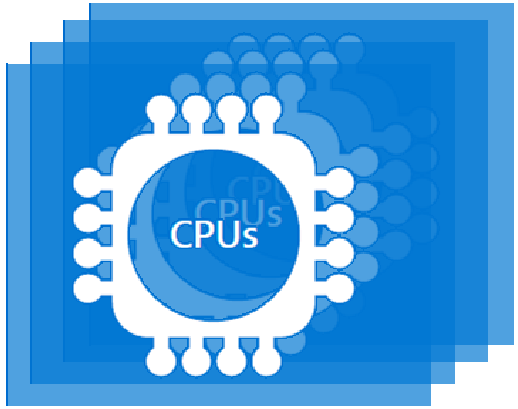
- Rise of **coprocessors**: *specialized hardware* attached to general-purpose CPUs, dedicated to *specific tasks*
  - **GPUs**: single instruction, multiple data → accelerate simple mathematical operations like matrix multiplication on batches of data
  - **FPGAs**: arrange reconfigurable hardware gates to perform tasks → *spatial computing*, apply multiple instructions in parallel to input data
  - **ASICs**: even more specialized than FPGAs, but not reconfigurable → faster, but costlier

- *Growing taxonomy*: more specialized processors emerging
  - **IPUs** (intelligence processing units): multiple-instruction, multiple-data chips aimed at machine learning applications

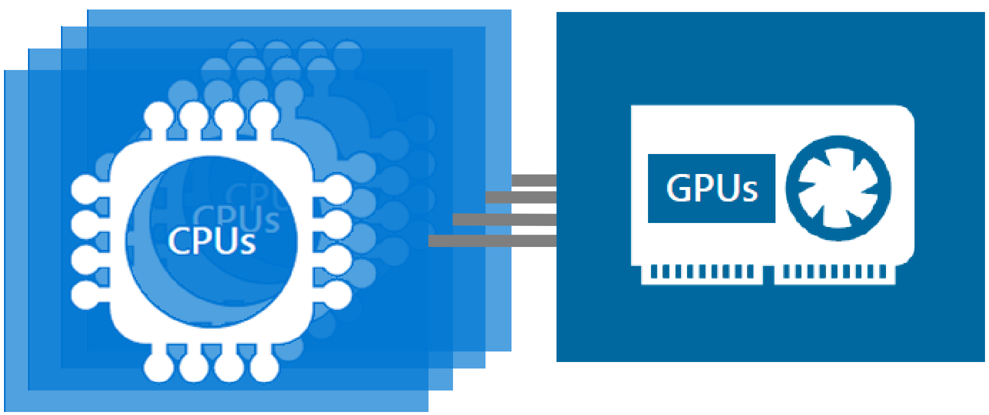
Software Evolves with Hardware



LHC Run 1: single-core



LHC Run 2: multithreading

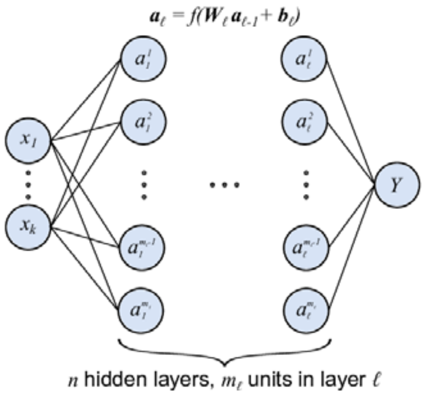


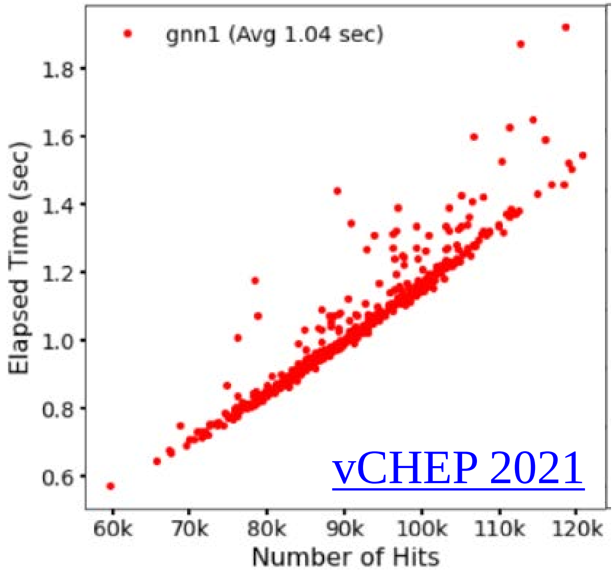
LHC Run 3: direct-connect offloading

- Progression: use more resources more efficiently
- Costs:
  - Multithreading: need thread-safe code
    - Framework changes
    - Partial rewrites of C++ code
  - GPU direct connect: need GPU-friendly algorithms
    - More framework changes
    - Full rewrites of C++ code into CUDA
- How to continue progression without incurring repeated costs (rewrites)?

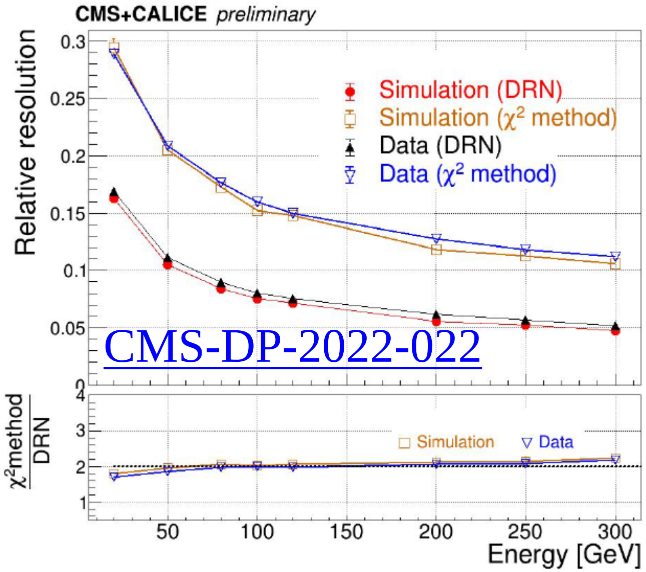
Machine Learning to the Rescue!

- Deep neural networks improve both physics accuracy & computational acceleration potential
  - Limited subset of mathematical operations: perfect for acceleration on GPUs or other coprocessors
  - Already outperform classical/rule-based algorithms for tasks like classification
  - Now being applied to lower-level reconstruction tasks: tracking (sub-quadratic scaling), clustering, calibration (2× resolution improvement vs. rule-based)





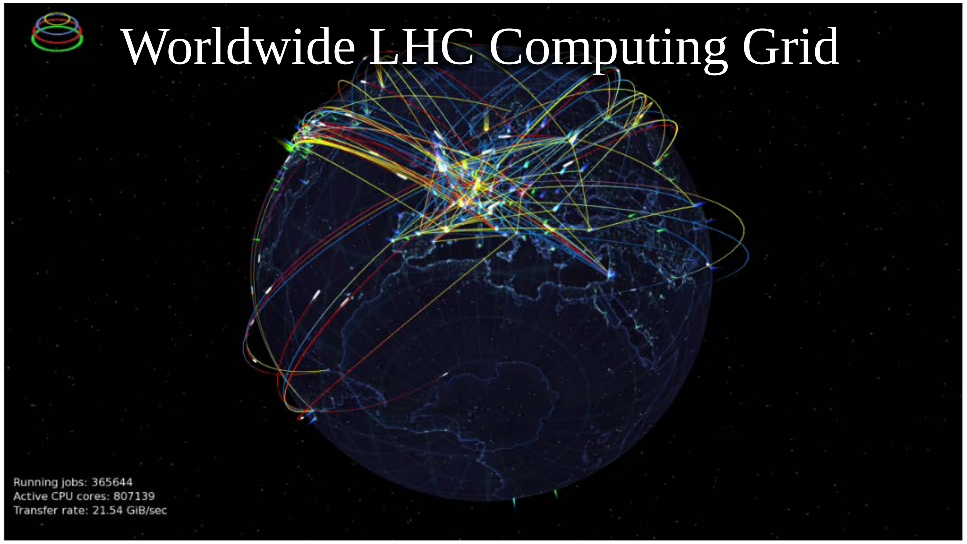
[vCHEP 2021]



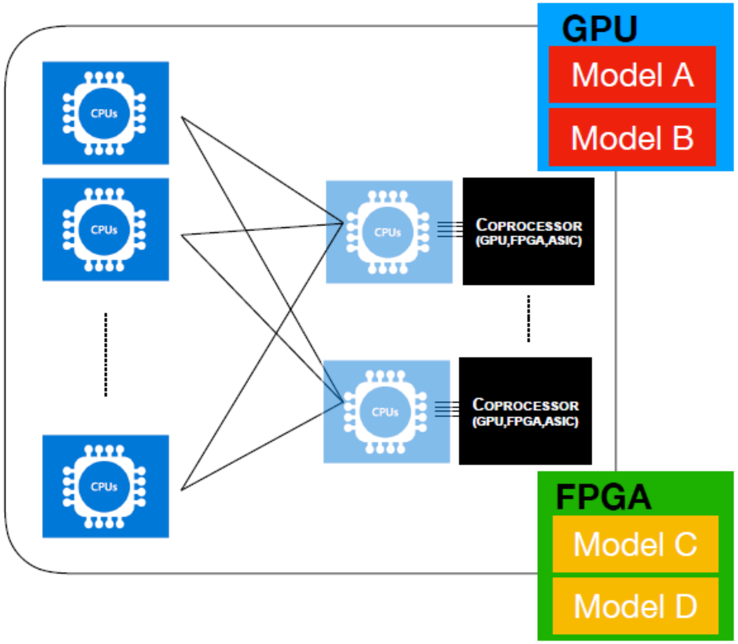
[CMS-DP-2022-022]

Why Accelerate Inference?

- Training is viewed as "hard" part of machine learning
  - Definitely requires time & expensive resources
- But, training happens N times (algorithm evaluated M times per training cycle)
- Inference (using trained DNN) performed for every event → *billions* of times
- N×M << billions → resource needs are smaller and can be concentrated (cloud, HPC, …)
- Training is done by experts & developers; inference is done by everyone → need solutions that scale to worldwide grid



Coprocessors As a Service



- Q: Will every worldwide CPU node have a coprocessor connected to it?
- A: Probably not… coprocessors are expensive!
- ➢ Need a *more general* approach to deploy algorithms on coprocessors
- Abstract CPU-coprocessor connection into communication protocol
- Multiple CPUs can send inference requests to multiple coprocessor servers
- Optimal, flexible, cost-effective use of resources
- Can deploy different algorithms on different coprocessors as desired

Services for Optimized Network Inference on Coprocessors

- SONIC: *design pattern* to implement coprocessors as a service in HEP experiment software frameworks (C++-based)
  - Goal: minimize disruption to existing computing model, minimize hardware dependence, maximize efficiency

- Numerous advantages:
  - *Industry tools*: gRPC, Kubernetes, inference servers
  - *Containerization*: ML frameworks separate from experiment software
  - *Simplicity*: modules only implement input/output conversions
  - *Flexibility*: adjustable deployment strategies when many CPUs connect to many coprocessors
  - *Efficiency*: aggregate work for full utilization of coprocessors (also most cost-effective approach)
  - *Portability*: Swap CPU, GPU, FPGA, IPU, etc. without any code changes
  - *Accessibility*: connect to any available coprocessor anywhere

SONIC Approach

Service

- Keep track of server details: URLs, available models, etc.
- Launch CPU fallback server

Module **Module** **Module**

- Produce, analyze, filter events
- Execute client operations (inference requests)

Client

- Implement client-server API, make requests
- Modes: Sync, Async, PseudoAsync

Data

- Handle I/O consistency checks, other operations

MemResource

- Abstraction for local vs. remote differences (gRPC vs. shared memory)

Timeline of SONIC

2019 (CMS)
- FPGA (Intel Altera)
- TensorFlow Serving (Microsoft Brainwave)

2020 (CMS)
- GPU (Nvidia)
- Triton Inference Server (Nvidia)

2020 (DUNE)
- GPU (Nvidia)
- Triton Inference Server (Nvidia)

2020 (CMS)
- FPGA (Xilinx)
- FPGA-as-a-Service Toolkit (FaaST)

2022 (CMS)
- GPU (Nvidia)
- Triton Inference Server (Nvidia)

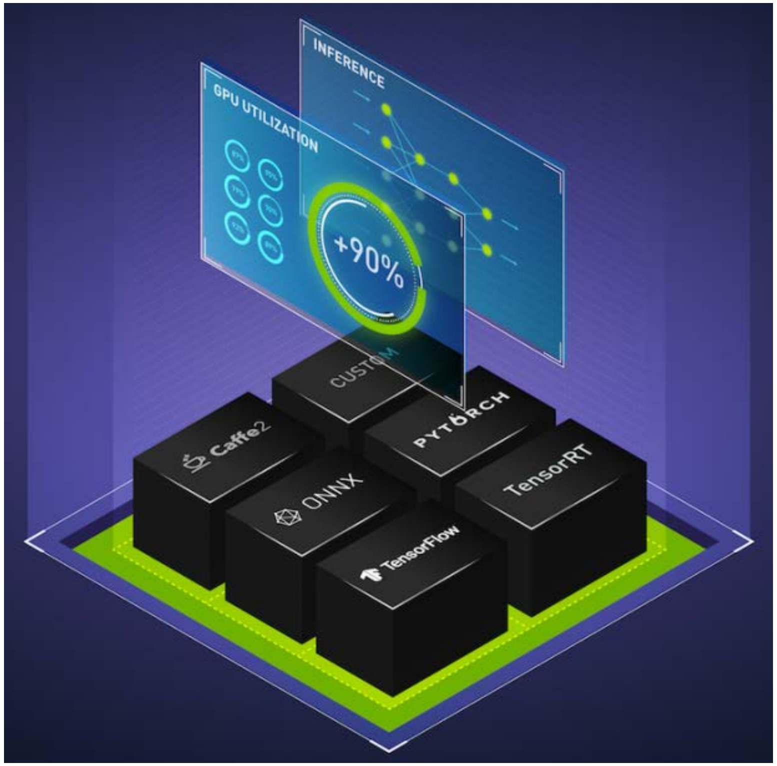
2022 (CMS)
- IPU (Graphcore)
- Custom Triton backend

2023 (DUNE)
- GPU (Nvidia)
- Triton Inference Server (Nvidia)

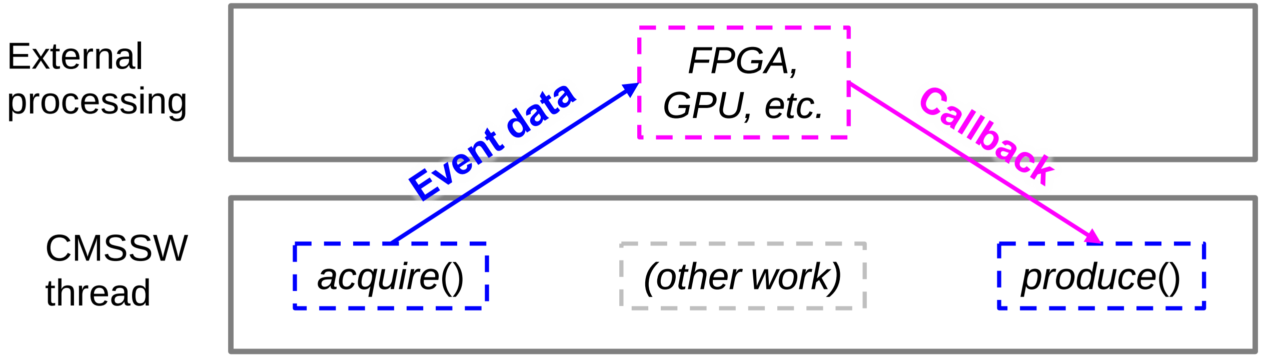
- In all cases, per-algorithm speedups of at least 10× were observed; sometimes more than 100×!
  - Depending on CPU and coprocessor used, ML framework versions and optimizations, etc.
- Variety of hardware, experiments, server technologies, communication protocols
- Now being deployed at analysis facilities!

Converging on Triton

- Triton Inference Server:
  - Free open source software from Nvidia
  - gRPC communication
    - Extension of standard KServe protocols
  - Supports all ML backends
    - + non-ML algorithms, non-Nvidia GPUs through custom backend
  - Dynamic batching: process events together to increase GPU utilization & throughput
  - And more: load balancing, compression, optimization, deployment tools…
- Has already been extended to FPGAs (FaaST, custom server implementing same protocols) and IPUs (custom backend)



Asynchronous



[Eur. Phys. J. Web Conf. 245 (2020) 05009]

- Most efficient method to access coprocessors: asynchronous, non-blocking
  - Enabled by ExternalWork mechanism in CMS software
    - On top of task-based multithreading
  - CPU does other work while coprocessor request is ongoing
    - ➢ Minimizes impact of network latency in aaS paradigm
- Especially important in collider reconstruction case: 100s of algorithms/event
  - No single dominant contributor

Synchronous

- If asynchronous functionality not available (not implemented, no task-based multithreading, etc.): can still benefit w/ synchronous, blocking calls

➢ Need to consider latency in performance projections

$$t_{\text{SONIC}} = (1-p) \times t_{\text{CPU}} + t_{\text{GPU}} \left[1 + \max\left(0, \frac{N_{\text{CPU}}}{N_{\text{GPU}}} - \frac{t_{\text{ideal}}}{t_{\text{GPU}}}\right)\right] + t_{\text{latency}}$$

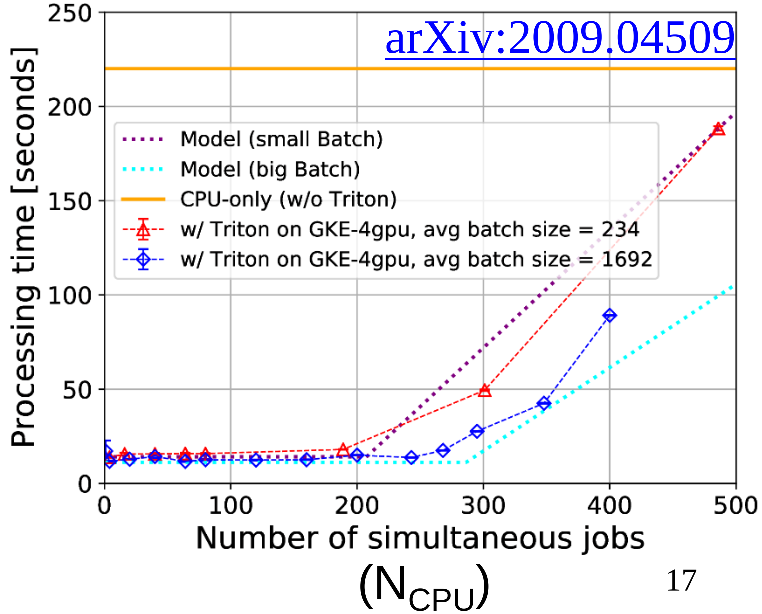
Unsaturated case

$$t_{\text{ideal}} = (1-p) \times t_{\text{CPU}} + t_{\text{GPU}} + t_{\text{latency}}$$

Saturated case

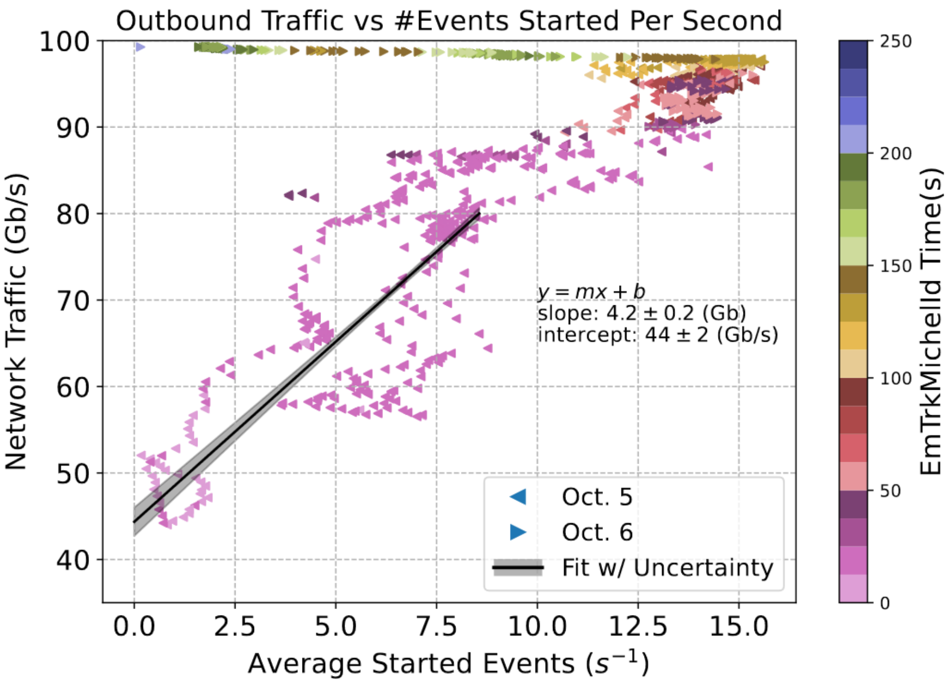
$$t_{\text{SONIC}} = t_{\text{GPU}} \times \frac{N_{\text{CPU}}}{N_{\text{GPU}}}$$

- Still substantial speedup for protoDUNE:
  - o One large CNN dominates reco time
  - o Observed performance agrees w/ above projections



IaaS at Scale

- Resource management becomes more important with IaaS, especially when scaling up
  - GPU not only resource that can saturate: also consider network bandwidth!
  - protoDUNE inputs are large (~4 Gb/image)



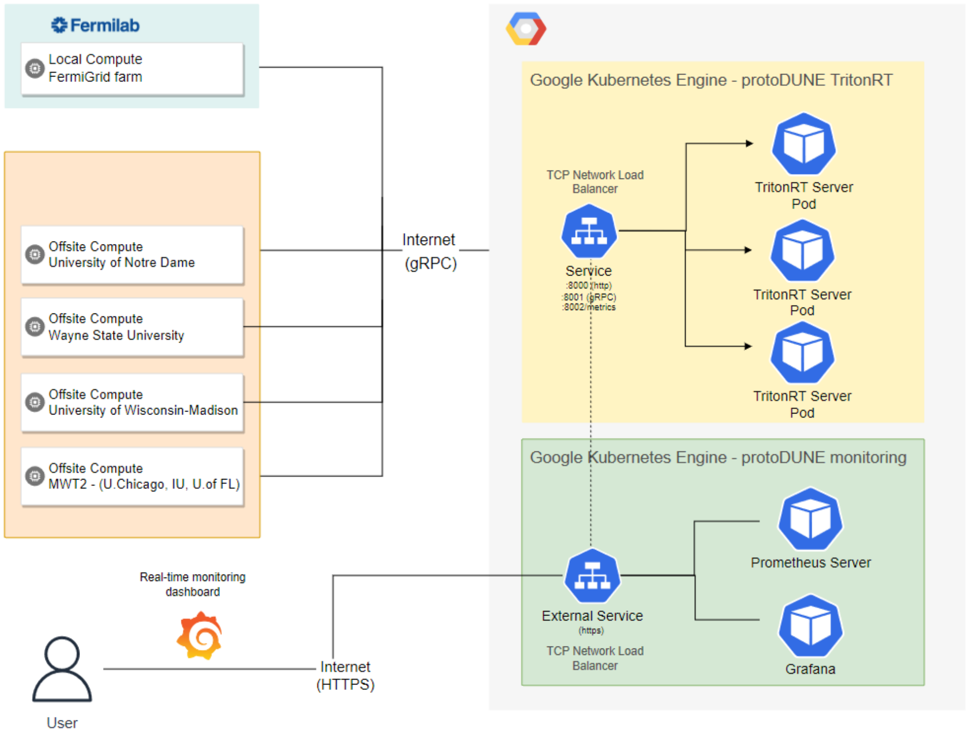
- Overview of resources:
  - **Processing**: client CPU(s), server CPU(s), coprocessor(s)
  - **Network**: both bandwidth and latency matter
  - **Memory**: attached to each processor
  - **Disk**: can also be local or remote
    - **Tape**: very high latency

Saturation: slows down

Saturation: maxes out → jobs killed, data lost

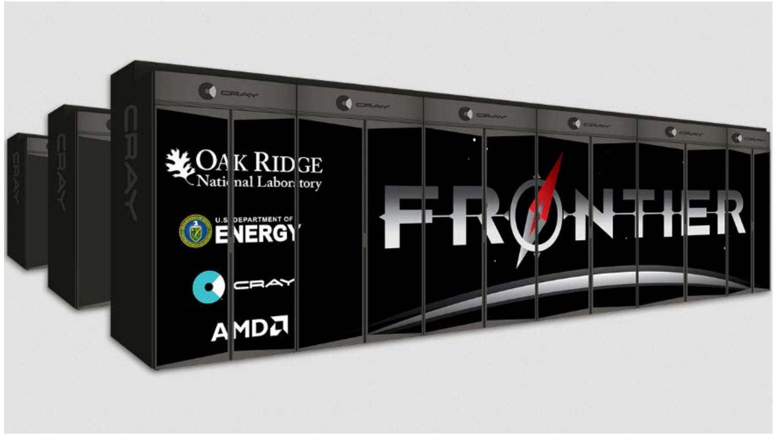
IaaS on the Grid

- LHC experiments and other large collaborations use 100s of computing sites distributed worldwide—and no two are the same…
- A higher level of heterogeneity: choices of storage technology, CPU architecture, coprocessor deployment, etc.
- Need *abstract requirements* for:
  - Server creation
  - Server discovery
  - Server preferences
  - Load balancing
    - Deploying multiple server or framework versions, etc.
  - etc.
- Also need to accelerate adoption of & support for industry tools
- Prime opportunity for further cloud integration in HEP workflows



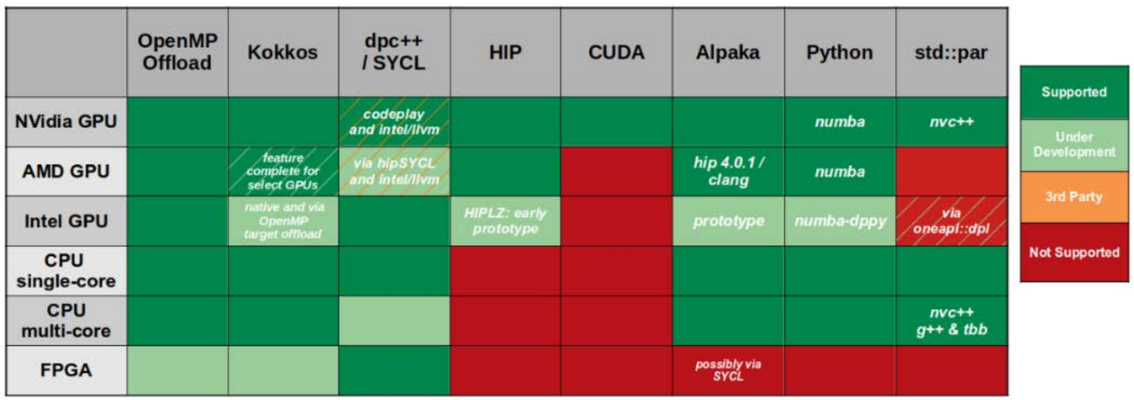
Portability

- Major benefit of ML algorithms:
  - o Automatically portable to new architectures, coprocessors, etc.
    - ➢ Industry does the work for us!
      - ▪ And sometimes we do the work for industry: hls4ml
- vs. standard approach to offload rule-based algorithms: rewrite in coprocessor-specific languages
  - o Fortran → C++ → thread-safe C++ → CUDA → ???
    - ▪ Mid-LHC Run 2: expected to move to many-core systems (Knights Landing, Xeon Phi, …), then canceled by chip companies
- New generation of HPCs: GPU-heavy, but many vendors: CUDA, HIP, SYCL, …
  - o Nvidia (Summit, Perlmutter)
  - o Intel (Aurora)
  - o AMD (Frontier, El Capitan)
- Next generation of HPCs: who knows?



Generalizing aaS

- Even with successes of ML, many rule-based algorithms are worth preserving and lend themselves well to coprocessor acceleration
  - o *Portability languages*: abstraction tool to compile same code to run on different hardware
    - ▪ leading candidate is Alpaka, based on performance & usability



	OpenMP Offload	Kokkos	dpc++ / SYCL	HIP	CUDA	Alpaka	Python	std::par
NVidia GPU			codeplay and intel/llvm				numba	nvc++
AMD GPU		feature complete for select GPUs	via hipSYCL and intel/llvm			hip 4.0.1 / clang	numba	
Intel GPU		native and via OpenMP target offload		HIPLZ: early prototype		prototype	numba-dppy	via oneapi::dpl
CPU single-core								
CPU multi-core								nvc++ g++ & tbb
FPGA						possibly via SYCL		

[arXiv:2203.09945](https://arxiv.org/abs/2203.09945)

- An "Alpaka backend" would further extend utility of SONIC and aaS
  - o Try to be as general and automatic as possible
    - ▪ Need compatibility with, or extraction from, experiment software
  - o If a computer can do the task for you… let it!

Conclusion

- Growing *size* and *complexity* of data in HEP experiments
- Increasing *variety* of computational resources
  - And corresponding constraints and challenges
- Accessing *coprocessors as a service*: most general & flexible approach
  - *SONIC* brings aaS to experiment software frameworks
- Increasing use of *ML algorithms* brings both physics and technical benefits
  - Easy to *accelerate* and very *portable*
  - Benefit from *industry* developments
- Goal: be *forward-looking*
  - Can't plan for every possibility
  - ➢ Instead, plan for *any* possibility

Backup

References

Papers:

- J. Duarte et al., "FPGA-accelerated machine learning inference as a service for particle physics computing", Comp. Soft. Big Sci. 3 (2019) 13, arXiv:1904.08986.
- D. Rankin et al., "FPGAs-as-a-Service Toolkit (FaaST)", Proc. H2RC (2020) 38, arXiv:2010.08556.
- M. Wang, T. Yang, et al., "GPU-accelerated machine learning inference as a service for computing in neutrino experiments", Front. Big Data 3 (2021) 604083, arXiv:2009.04509.
- J. Krupa, K. Lin, et al., "GPU coprocessors as a service for deep learning inference in high energy physics", Mach. Learn. Sci. Tech. 2 (2021) 035005, arXiv:2007.10359.
- T. Cai et al., "Accelerating Machine Learning Inference with GPUs in ProtoDUNE Data Processing", arXiv:2301.04633, January 2023.

Code:

- ToySonic: simple demonstration of interfaces
- SonicCore, SonicTriton: CMSSW version
- NuSonic: LArSoft version
- fastmachinelearning/SonicCMS: FPGA versions
- FaaST: FPGA-as-a-Service Toolkit (server code)