Portable Acceleration of CMS Mini-AOD Production with Coprocessors as a Service

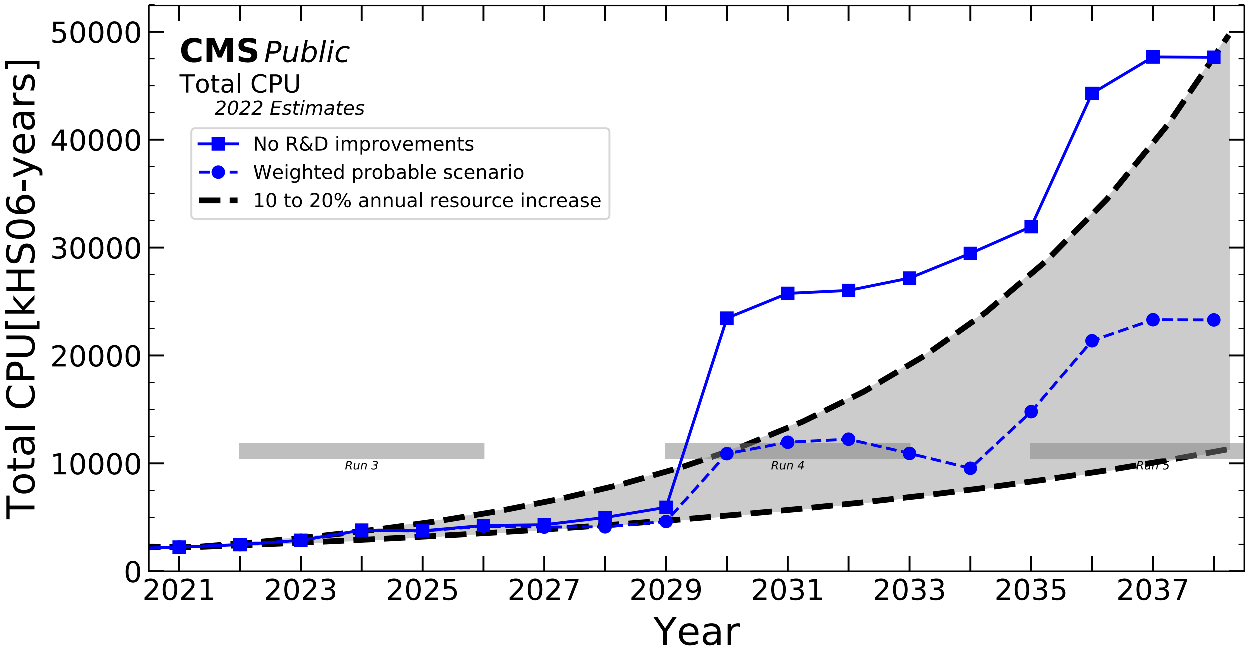
Patrick McCormack (MIT)
For the CMS Collaboration

September 25, 2023

FastML 2023

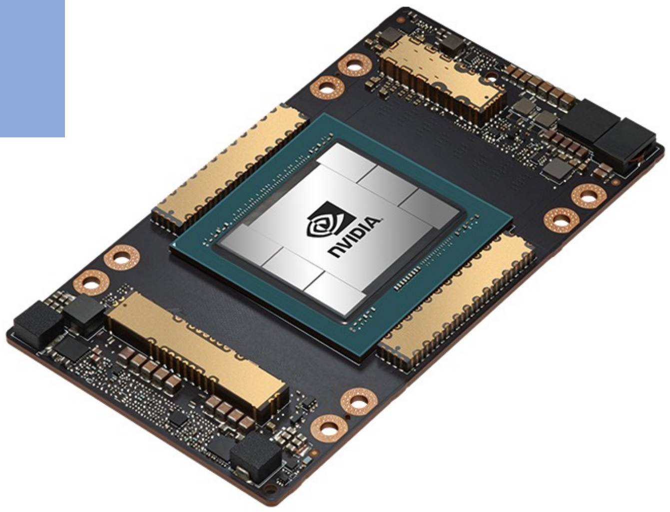
The future of CMS computing

- As the LHC transitions to the High Luminosity LHC, CMS workflow complexity will only increase
- CPU-only capabilities expected to increase, but it would be helpful to pursue additional performance enhancements



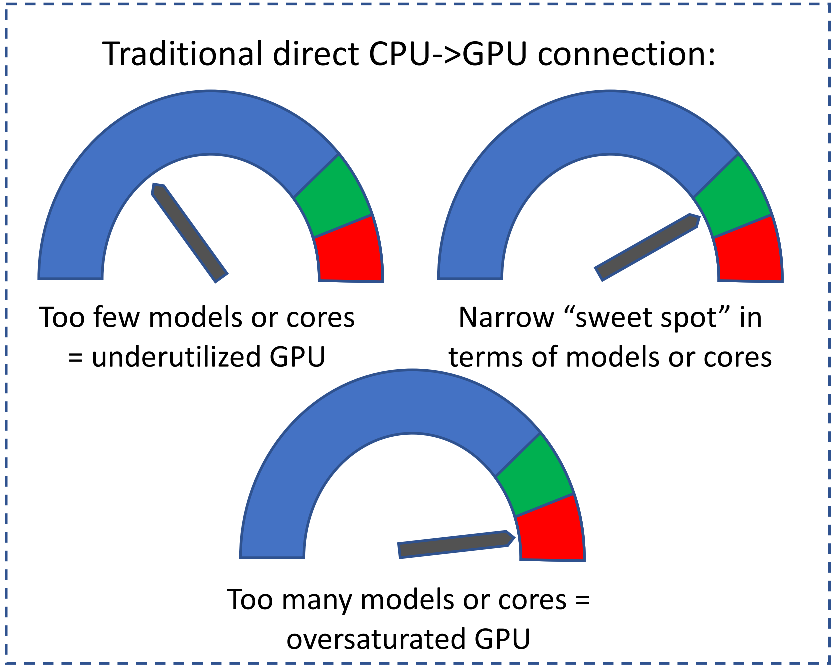
Hardware-based acceleration

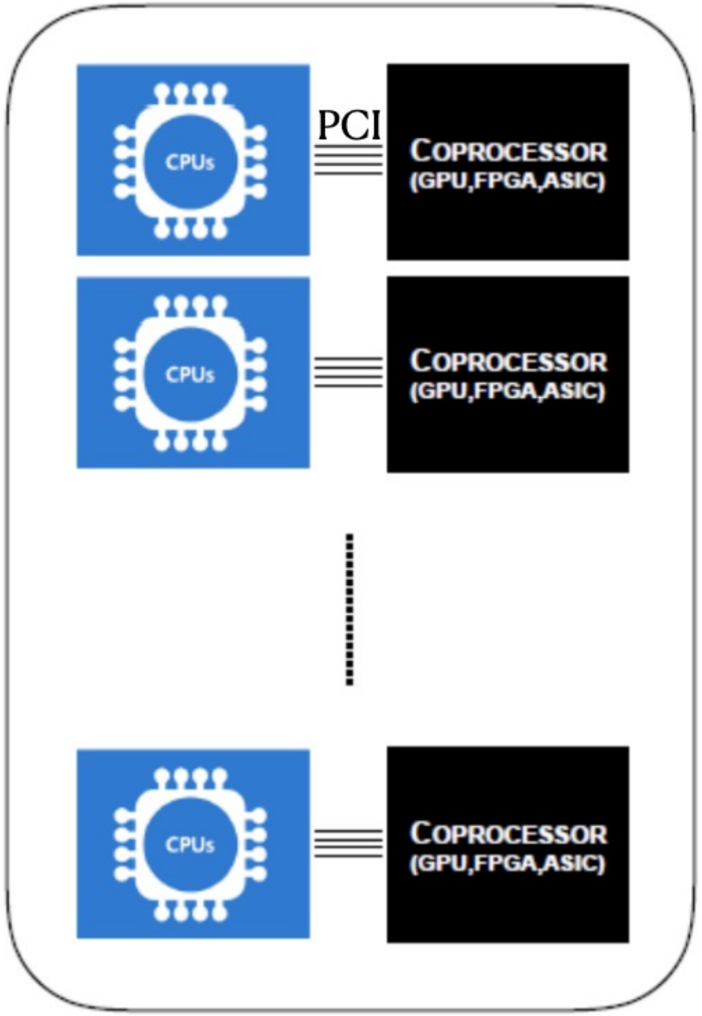
- Machine learning (ML) based algorithms are becoming increasingly common in CMS workflows
- Luckily, inference for ML algorithms (and some domain algorithms) can be accelerated dramatically by running on coprocessors
  - E.g. GPUs, FPGAs, and Intelligence Processing Units (IPUs)



Heterogeneous computing

- The most straightforward way to deploy algorithms on coprocessors is to run workflows on machines with coprocessors
- This “Direct connection” can be inefficient:



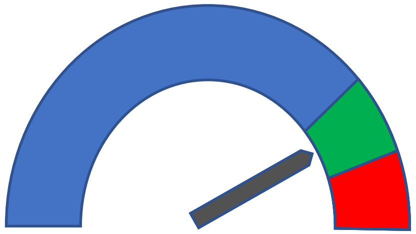


Also: workflows can only take advantage of acceleration if they run on a machine with a coprocessor – expensive at large scales!

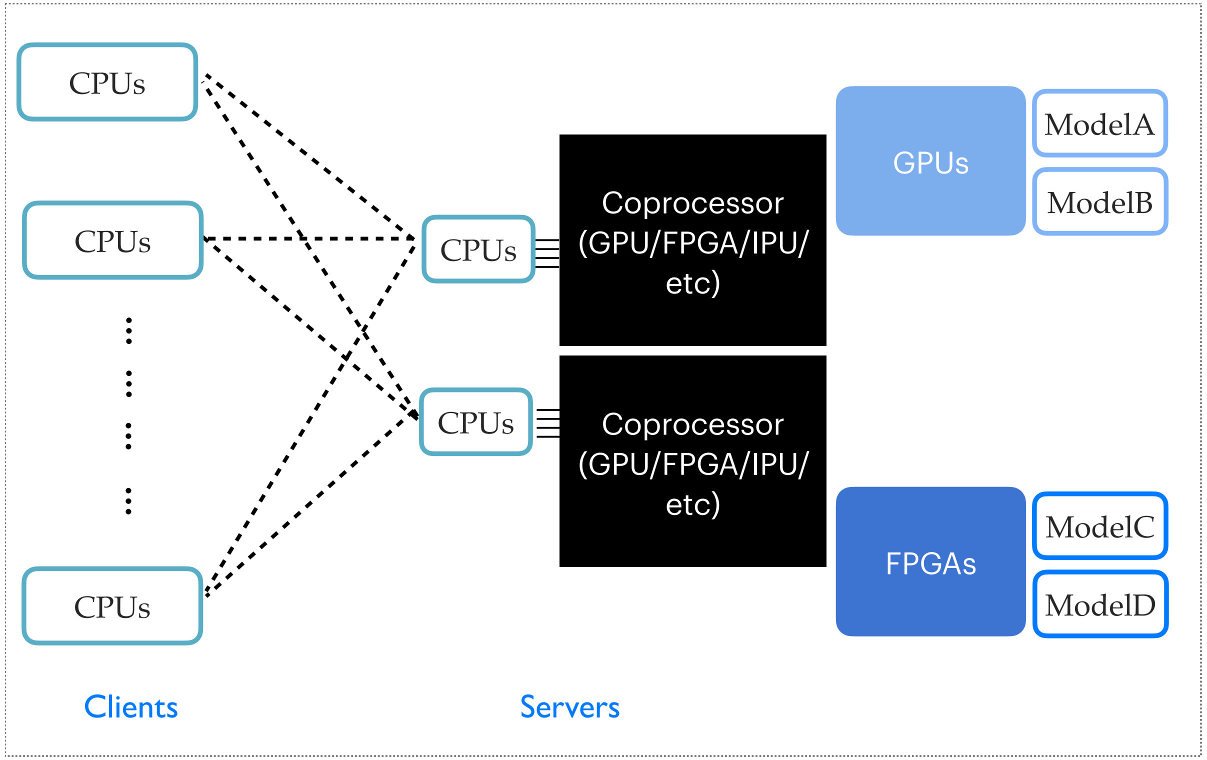
Inference as a Service

- "Inference as a Service" (IaaS): alternate coprocessor deployment scheme where coprocessor-enabled machines host an inference server and remote jobs send inference requests via network connection

Inference as a Service:

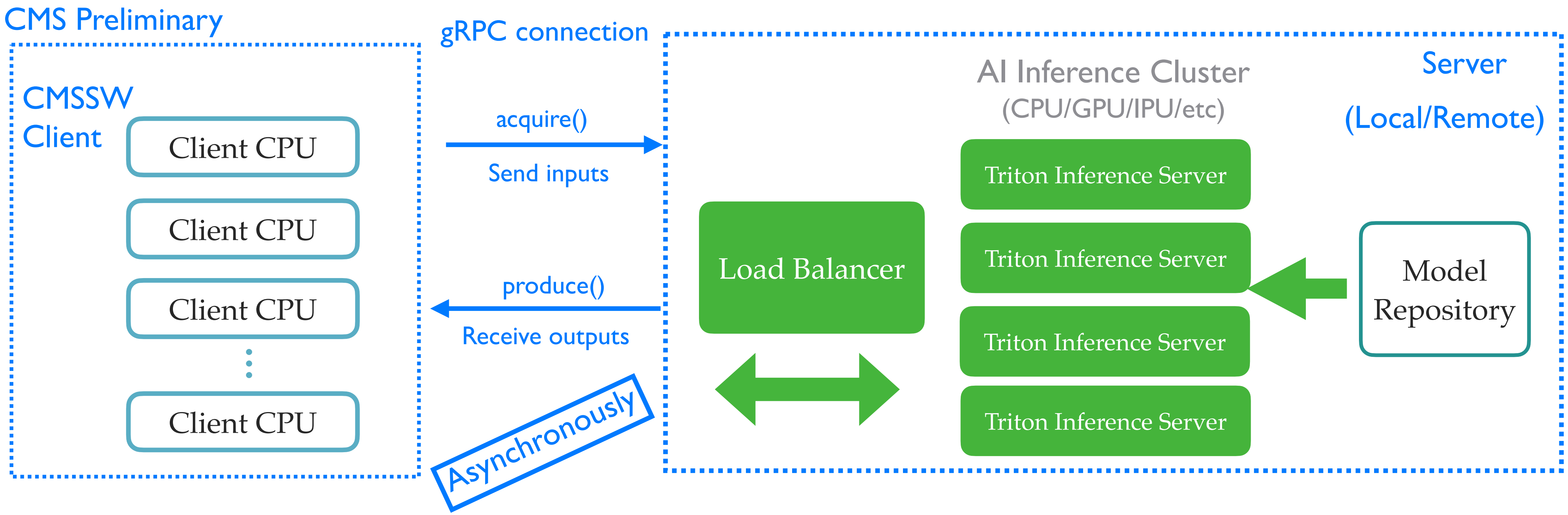


Adjust the number of GPUs per client-CPU core to get as close to the "sweet spot" as possible



SONIC

- Within CMS software (CMSSW), the IaaS deployment scheme is called "Services for Optimized Network Inference on Coprocessors" (SONIC)



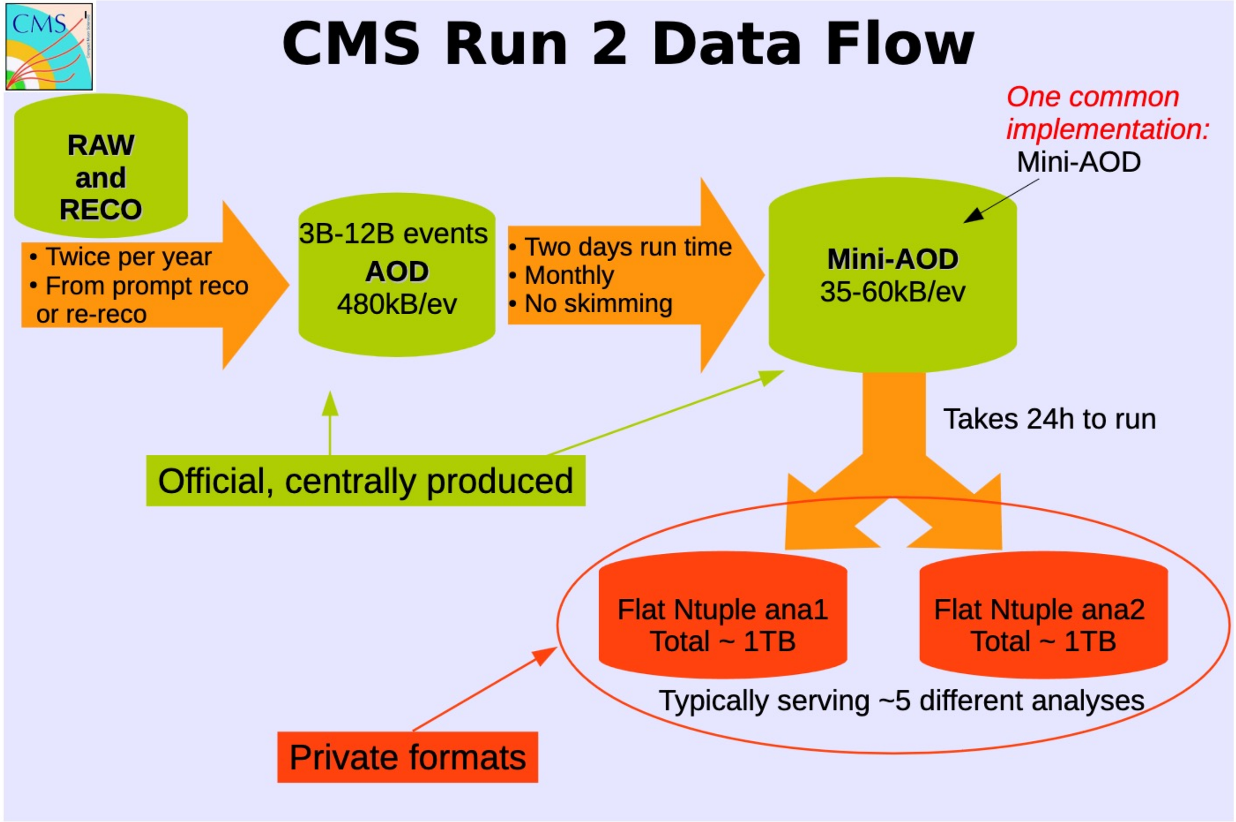
SONIC

- SONIC uses [NVIDIA Triton inference servers]
- CMSSW only handles preprocessing and I/O, not inference framework
  - Triton supports many ML backends: ONNX, TensorFlow, PyTorch, Scikit-Learn, etc.
  - Improves model-building flexibility
- Makes asynchronous inference requests

Studying SONIC at scale

[1702.04685]

- As a testbed for SONIC-enabled deployment, we created a MiniAOD demonstrator workflow
  - Runs a refinement and slimming step of CMS data processing
  - Full MiniAOD processing workflow typically run ~monthly



Mini-AOD production typically takes about 0.5 seconds per event on production grid nodes

Studying SONIC at scale

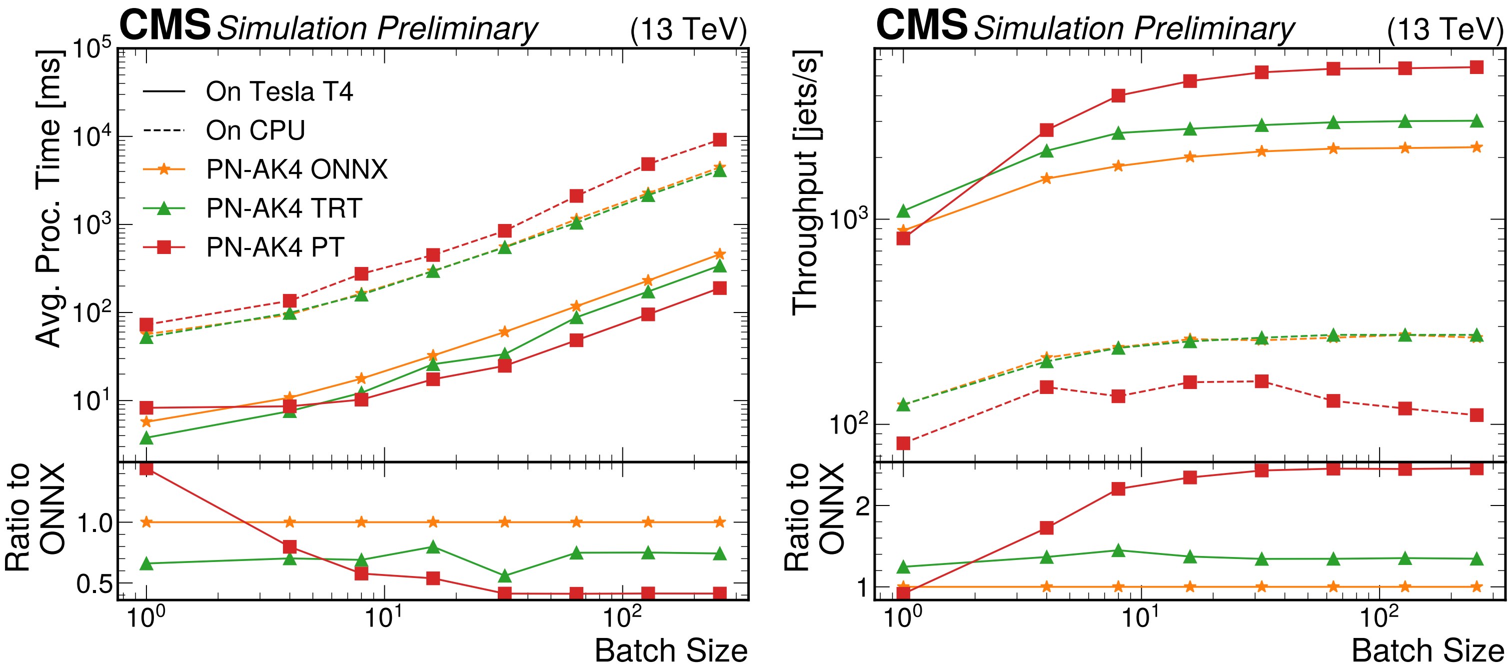
- Inferences for three classes of algorithms were run through SONIC:
  - ONNX-based jet tagger
  - TensorFlow based missing energy calculation
  - TensorFlow based CNN for tau lepton ID
- These algorithms consume about 10% of total workflow latency

Algorithm	Time [ms]	Fraction [%]	Input [MB]
PN-AK4	42.4	4.3	0.04
PN-AK8	11.4	1.1	0.003
DeepMET	13.2	1.3	0.33
DeepTau	21.1	2.1	1.18
ParticleNet+DeepMET+DeepTau	88.1	8.8	1.55
Total	993.3	100.0	—

Computing resources

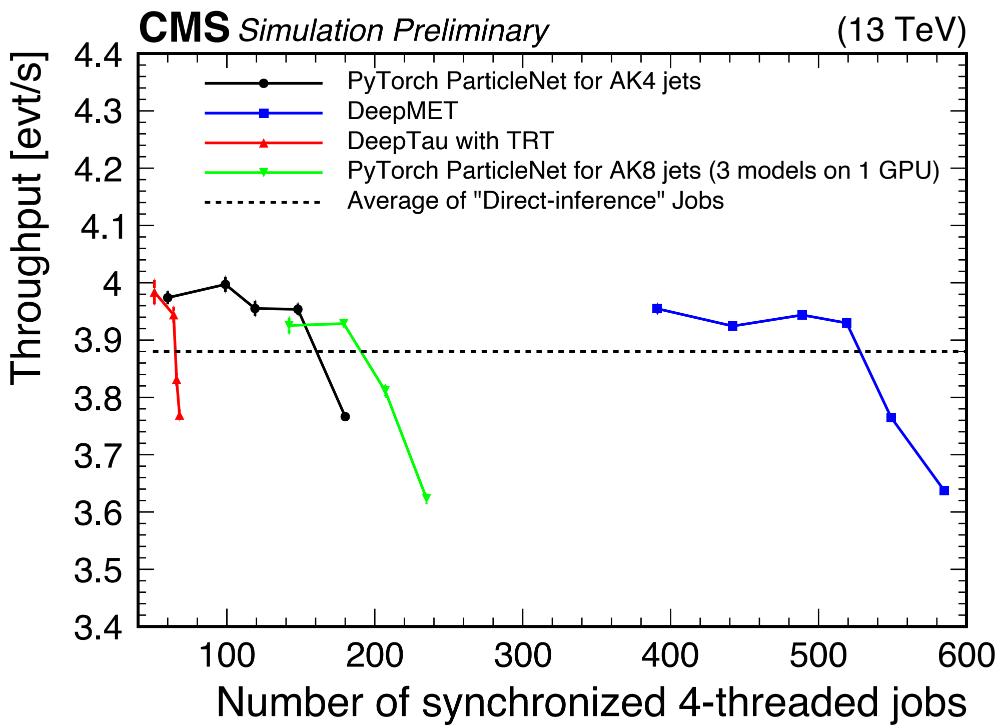
- MiniAOD demonstrator was deployed in multiple computing contexts
  - **Google Cloud (GCP)**: Triton server on cloud VM, with client-side CPUs also in cloud
  - **Purdue computing cluster:** 2 T4s available – client CPUs at Purdue (can also use cloud GPUs)
  - **Fermilab computing:** We had (non-exclusive) access to 2 T4s at Fermilab
- NOTE: Can use CPUs at one site to communicate with GPUs at another site

Optimizing performance: server parameters



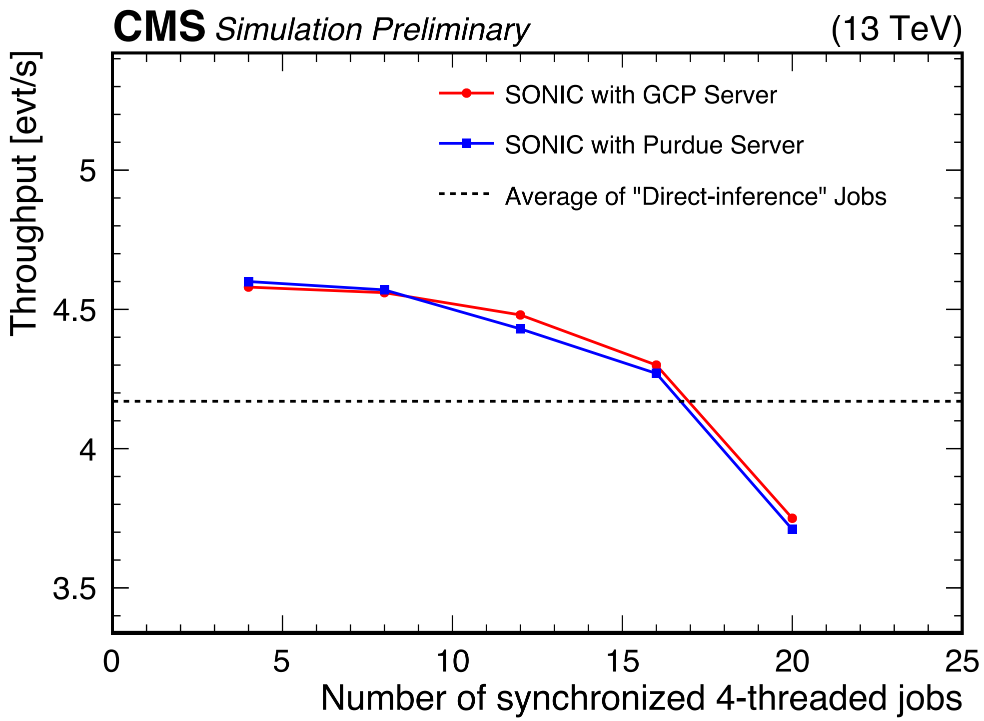
- Triton provides a model analyzer tool to optimize server settings
  - For example, we can adjust parameters like preferred batch size
  - We can also compare different backends if there are multiple versions and try optimization schemes such as TensorRT (TRT)

Optimizing performance: CPU-to-GPU ratio



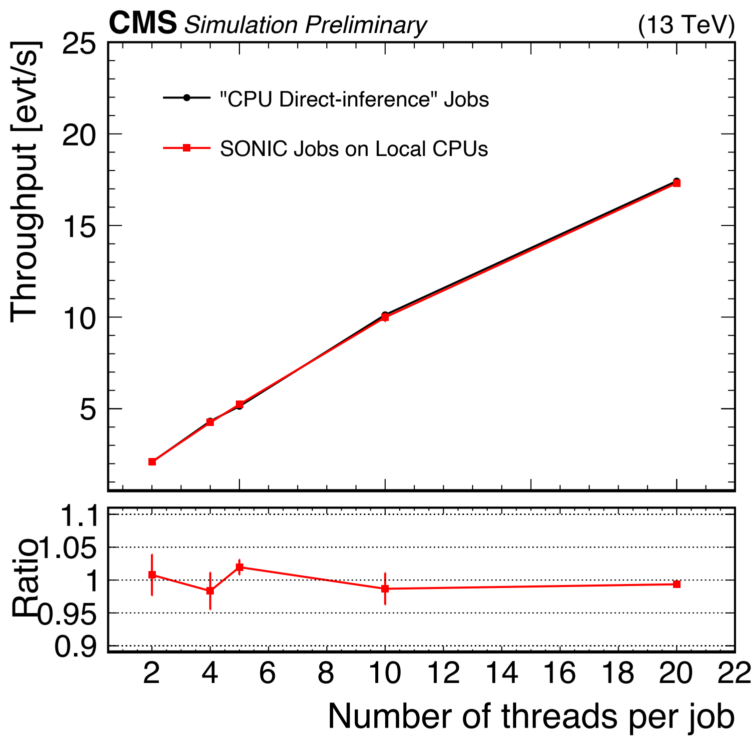
- Having explored server parameters, we can test the number of client jobs that a single GPU can handle
- We perform these tests in the cloud, as we need to synchronize jobs running on O(1000) CPU cores

Testing performance: distance-induced latency?



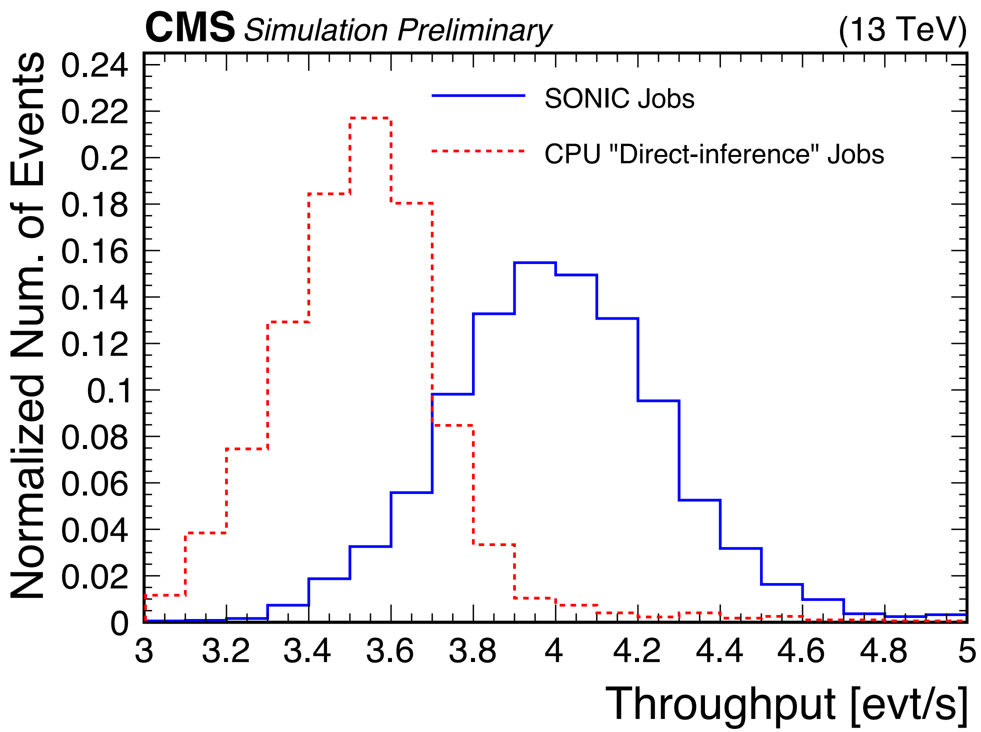
- Because we run algorithms asynchronously, per-event latency should not be negatively affected by client-to-server distance
- This was verified by running client jobs at Purdue that talked with servers either locally at Purdue or in the cloud (physically over 100 miles from the client)

Testing performance: server overhead?



- SONIC deployment accounts for potential server failures by reserving the ability to deploy a "fallback" server based on client-side CPU resources
- Ideally, this would not result in higher latencies relative to running entirely on CPU without SONIC - we do not observe any such slowdowns

Testing performance: running at large scale



- Lastly, we performed a scale-out test at GCP, using 10,000 CPU cores split into 2,500 4-threaded client jobs
- 100 Tesla T4 GPUs were used to host the MiniAOD models with a Kubernetes load balancer to ensure even GPU usage
- Peak network usage was ~15 GB/s (total bandwidth coming into GPU cluster)

Conclusions

- Inference as a Service, implemented as SONIC in CMSSW, can help alleviate CMS computing pressures by accelerating algorithm execution
- With SONIC, we achieve
  - **Increased throughput:** GPUs enable acceleration of ML algorithms
  - **Optimizable GPU-to-CPU ratios:** we can save money if looking to buy GPUs or increase utilization of current resources
  - **Flexible algorithm design:** Not restricted to only supported frameworks in CMSSW
  - **Use of remote GPUs**
- Demonstrated robustness and minimal impact of potential risks
  - Use of fallback servers shown to not impact workflow throughput
  - Network bandwidth requirements not problematic at scales similar to true MiniAOD workflow deployment

Future and challenges

- To run SONIC in full production, we need a GPU resources scouting and server deployment scheme
  - Currently developing a Kubernetes-based framework for dynamic server creation and deletion
- Convert more reconstruction algorithms to ML to take advantage of hardware-based acceleration*
- Can expand to other GPU vendors and coprocessor types with custom backends or interoperable servers

*And potentially improve performance