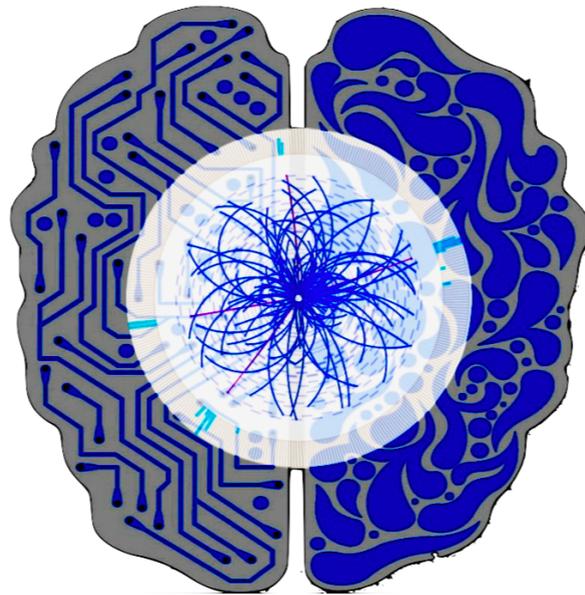




Fast ML Inference as-a-Service (aaS)

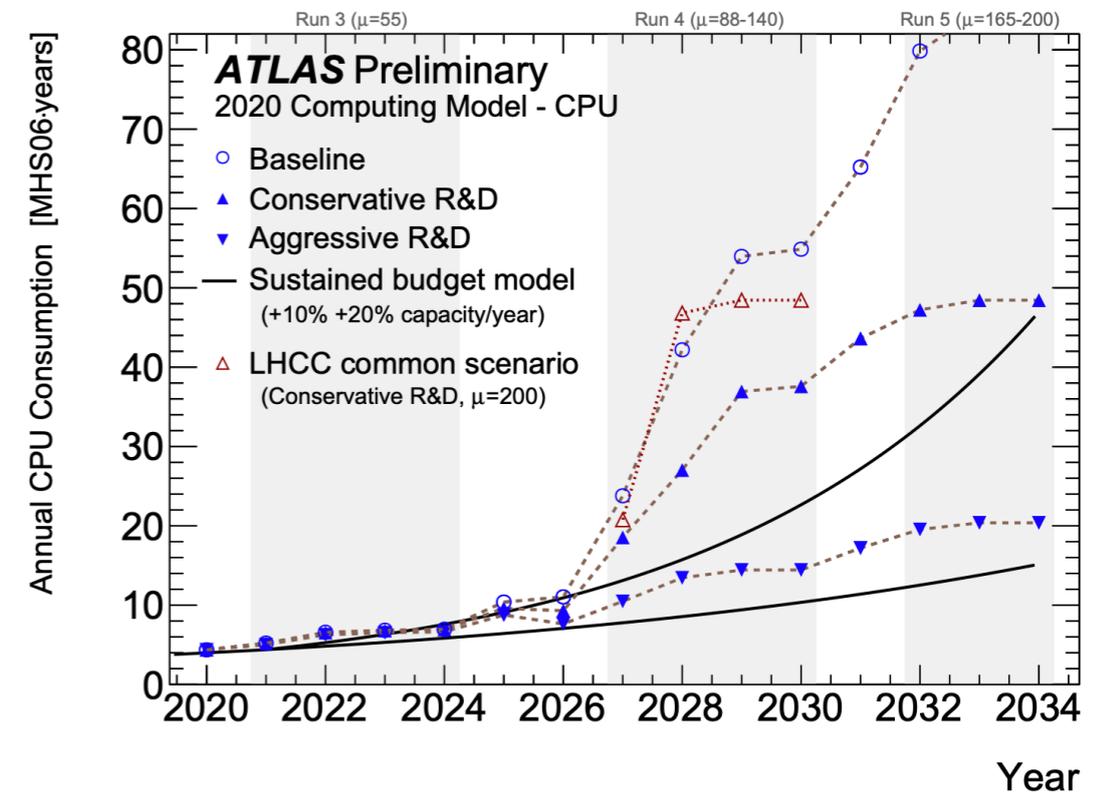
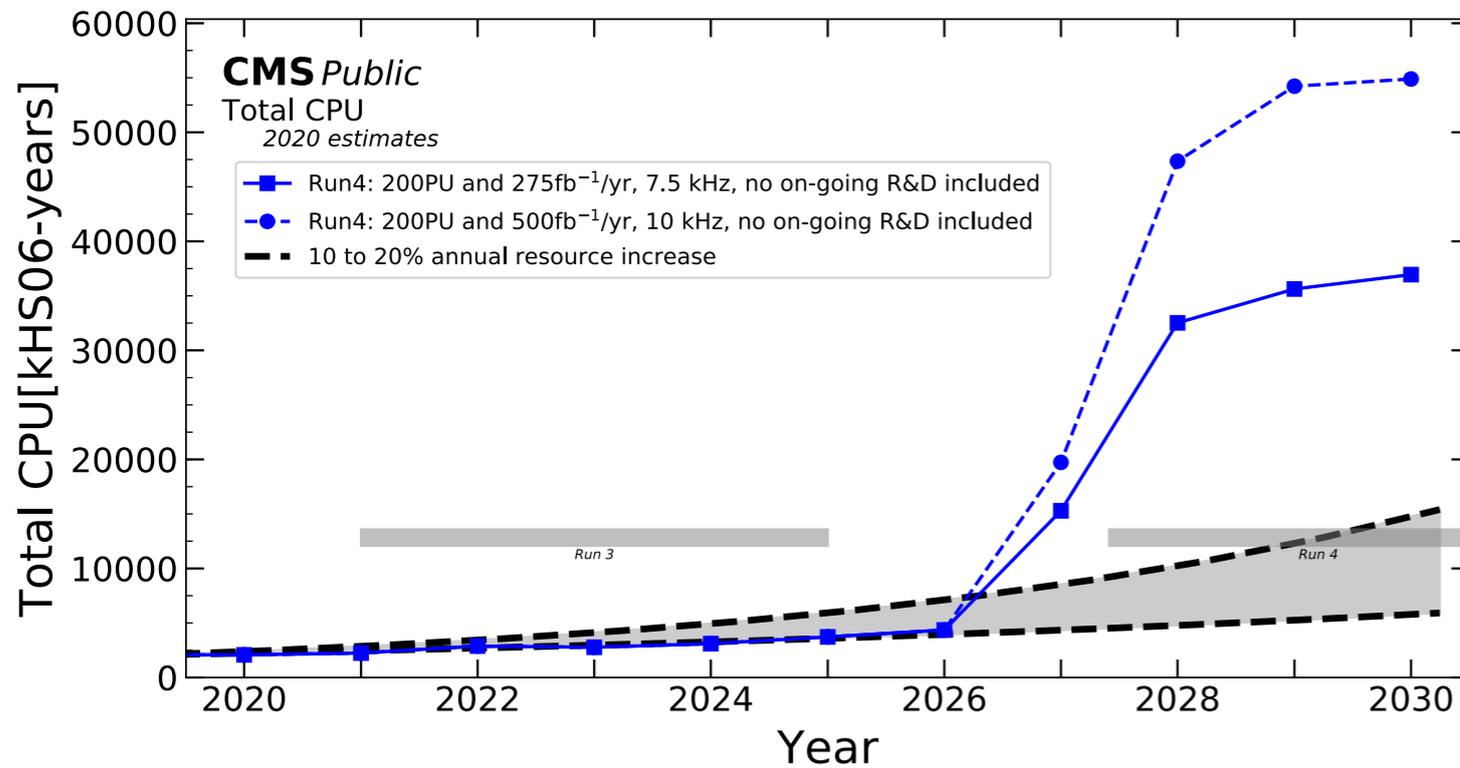


Dylan Rankin (MIT)
On behalf of the FastML group

December 4th, 2020

Introduction

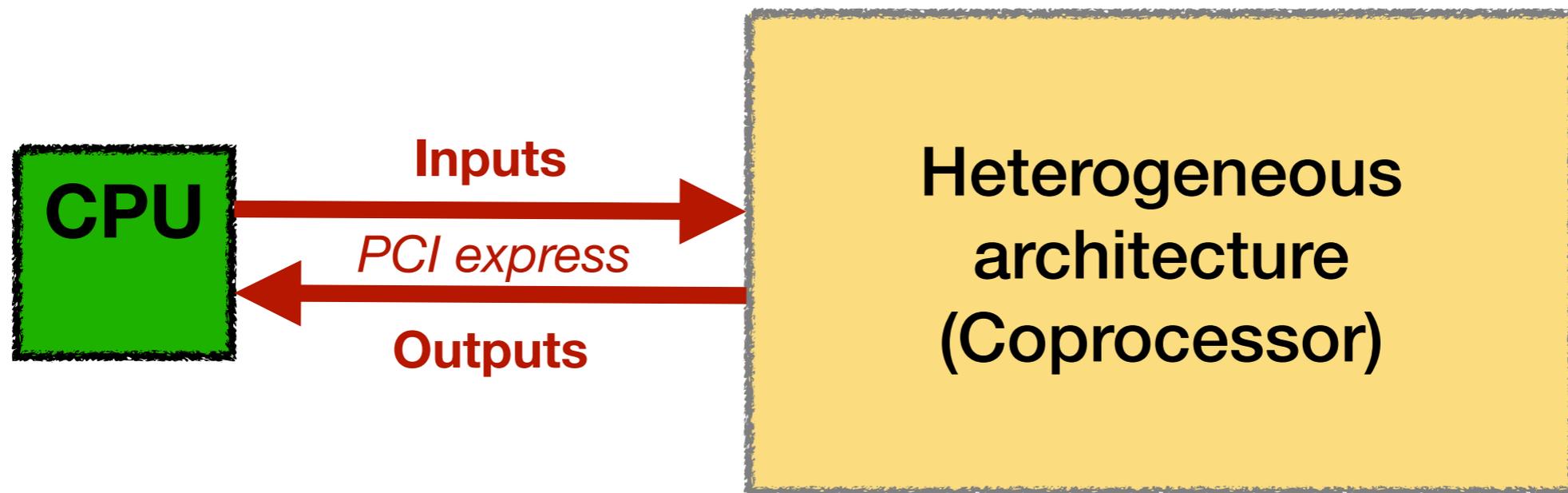
- Computing projections for high energy physics (HEP) greatly outpace CPU growth, interest in ML rapidly increasing



- **Coprocessors (GPU, FPGA, ...)** offer possible solution
- Natural marriage with ML
- Speedups at large:
 - Batch size and/or complexity

Direct Connect

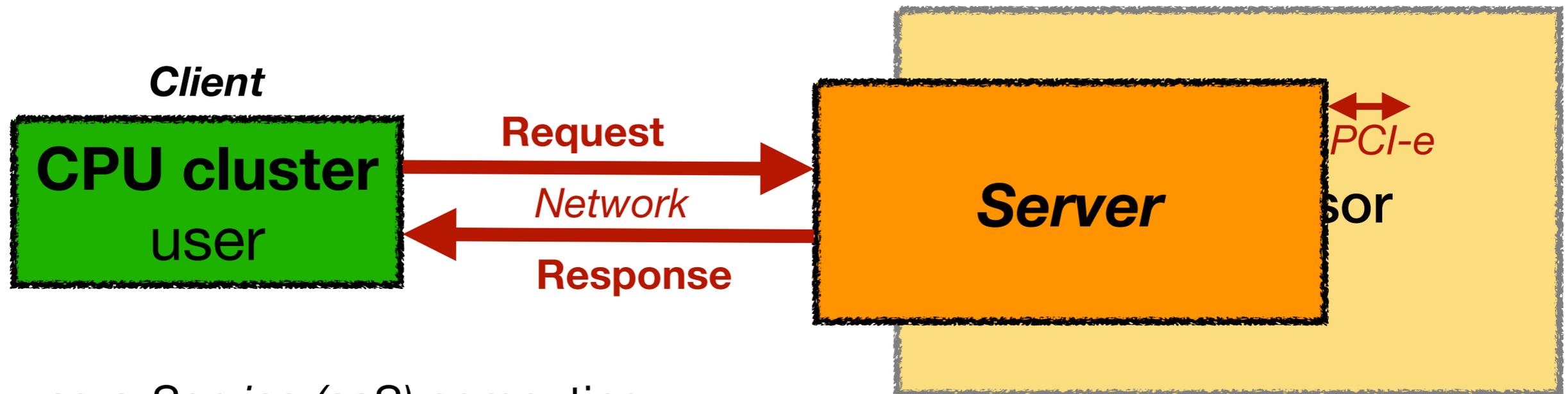
- Simplest form of heterogeneous architecture implementation is *direct connect*



- Scaling this up can be difficult
- CPU needs some knowledge of coprocessor details

as-a-Service Computing

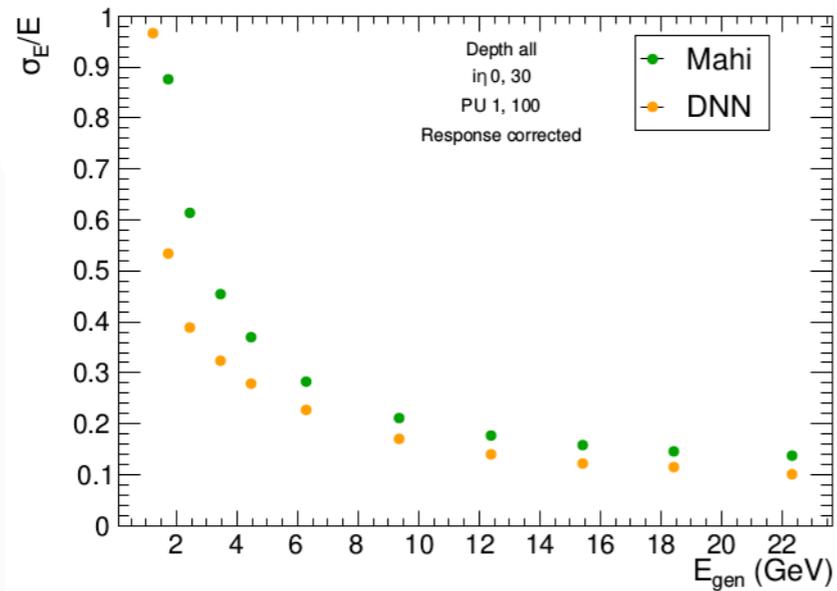
- As a user, I just want my workflow to run quickly



- *as-a-Service (aaS)* computing
 - Client communicates with server CPU, server CPU communicates with coprocessor
- Many tools available
 - *Some details in the backup, also Kevin's talk next*

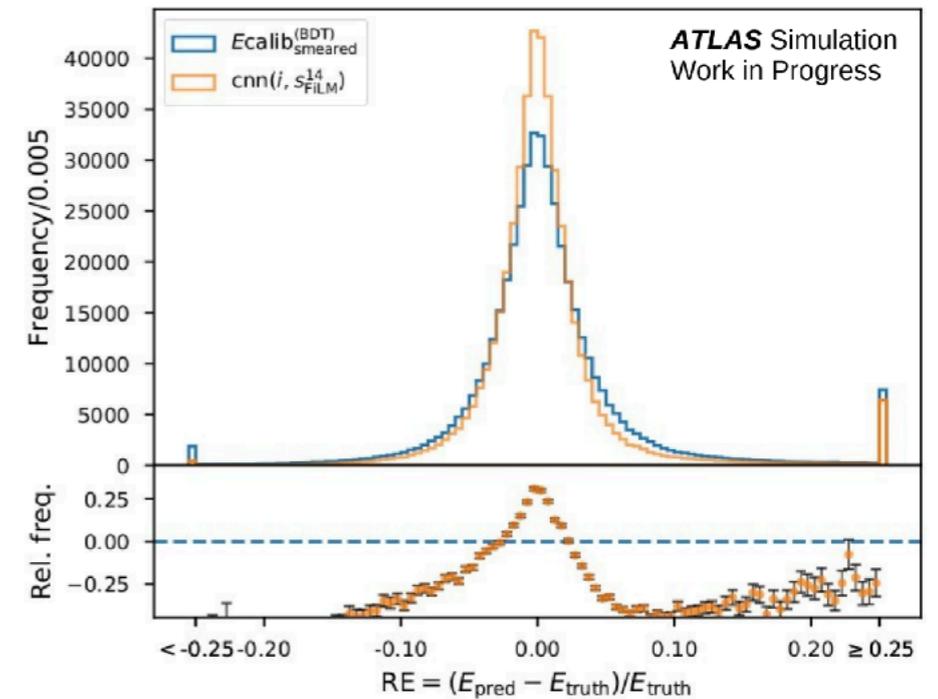
Benchmarks

FACILE



Calorimeter energy regression

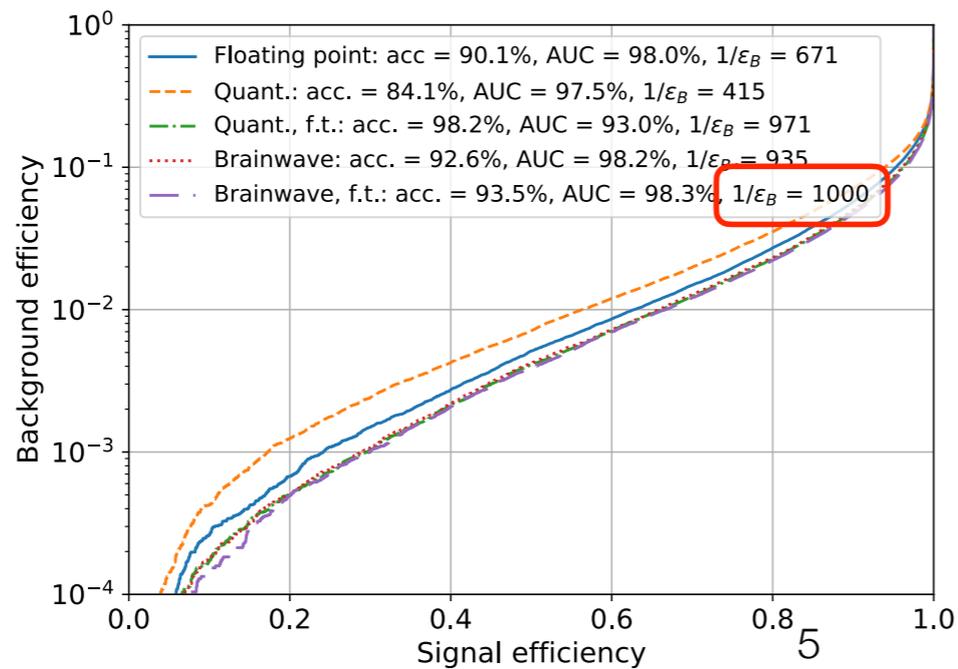
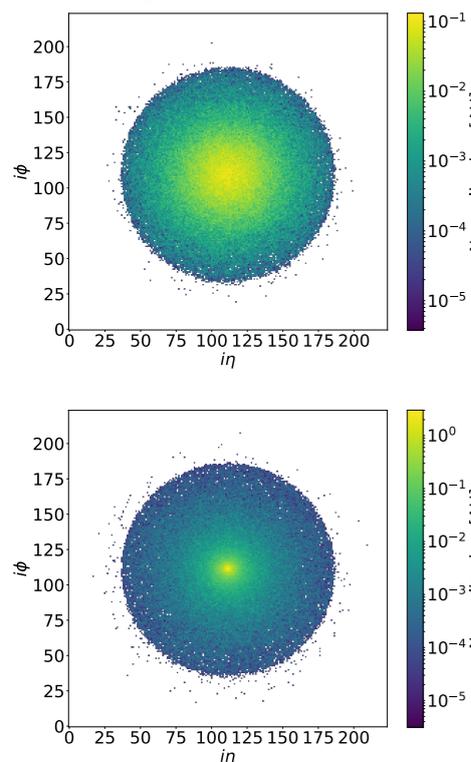
DeepCalo



Cluster energy regression

ResNet top quark image classification

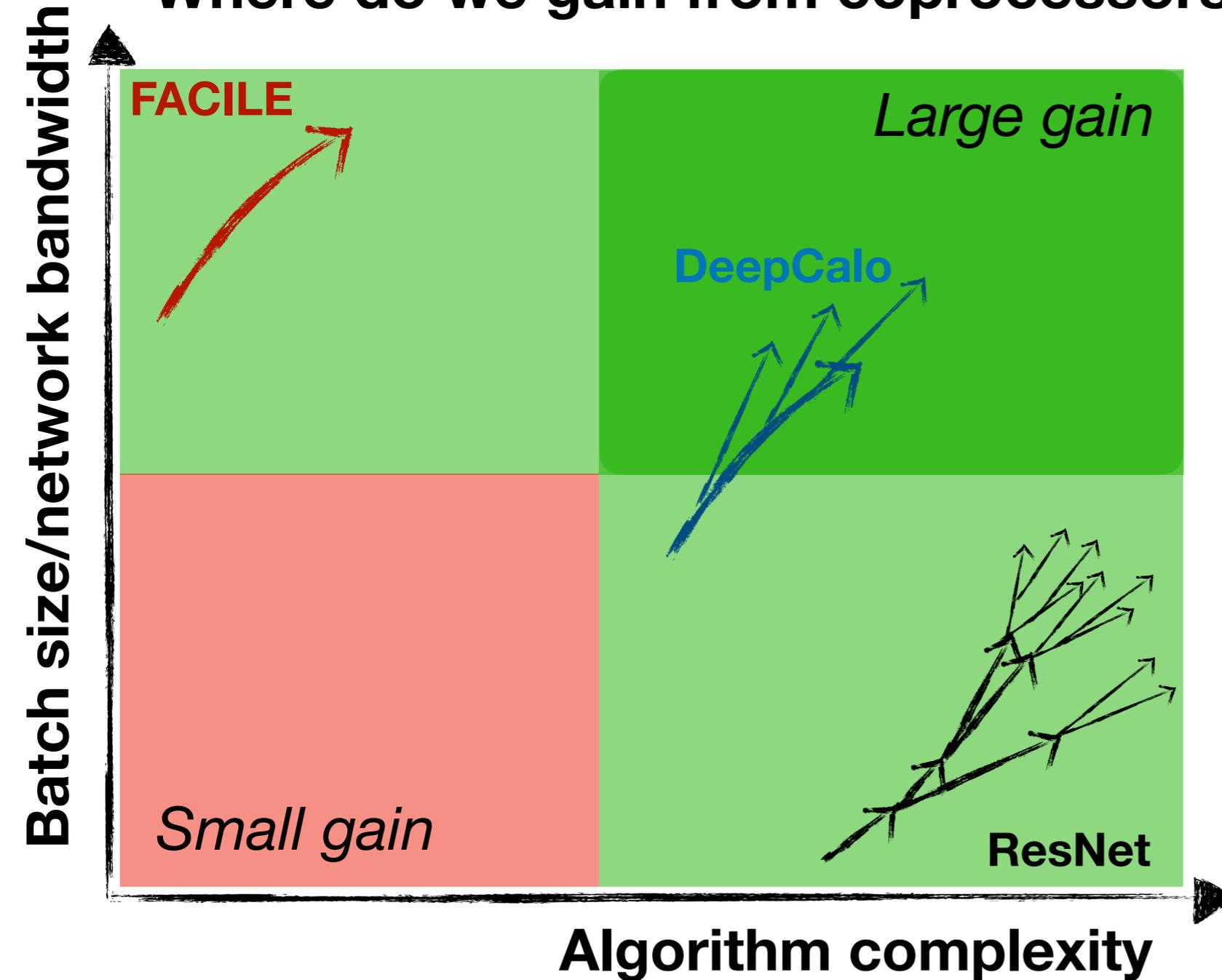
Averaged over 1000 jets



- **FACILE** (batch 16000) **2k parameters**
- **DeepCalo** (batch 10) **2M parameters**
- **ResNet** (batch 10) **10M parameters**

Gains

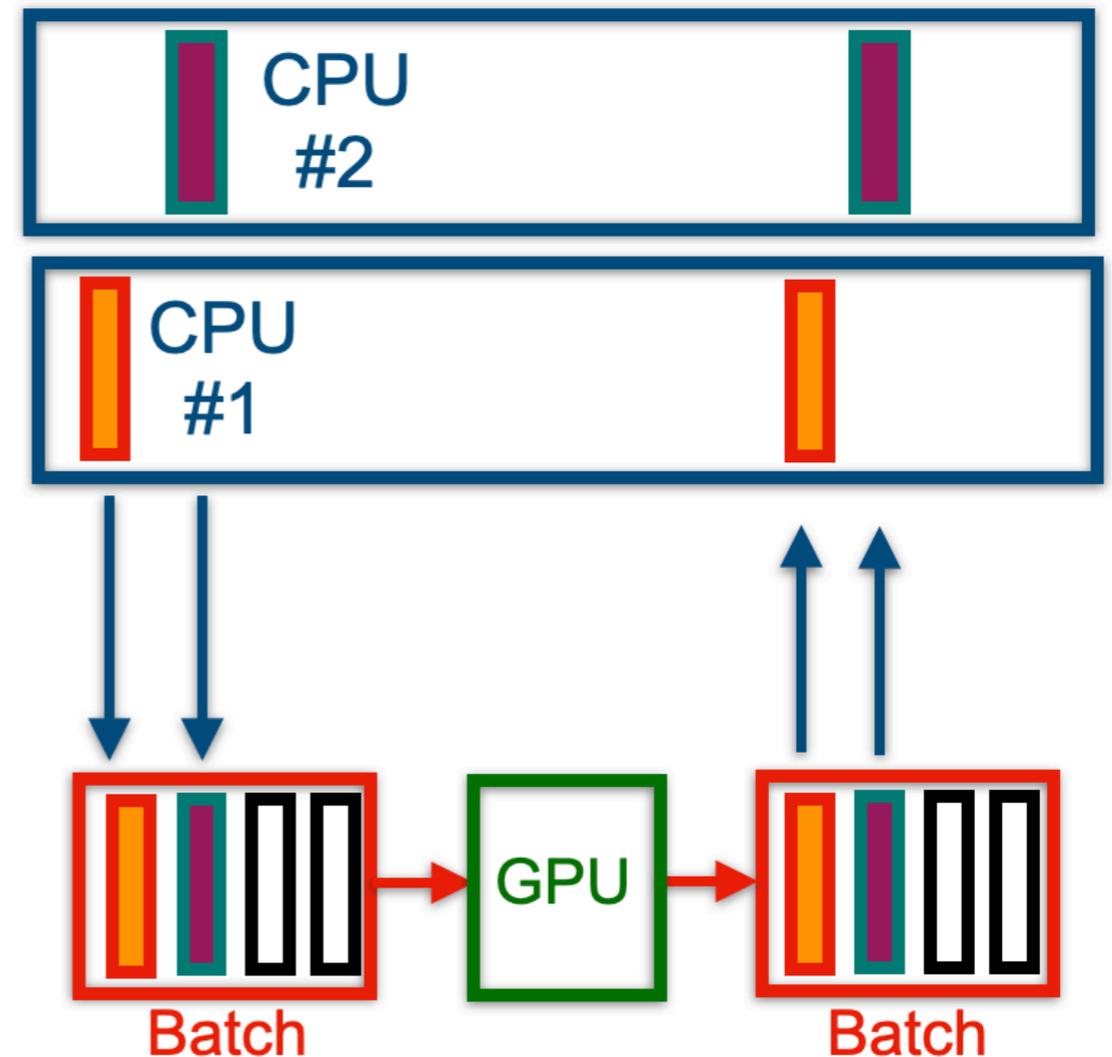
Where do we gain from coprocessors?



GPU/FPGA aaS	Gain w.r.t. CPU
2 ms (GPU) 0.2 ms (FPGA)	8x (GPU) 80x (FPGA)
0.1 ms (GPU) in progress (FPGA)	750x
1-2 ms (GPU/FPGA)	500x

Dynamic Batching

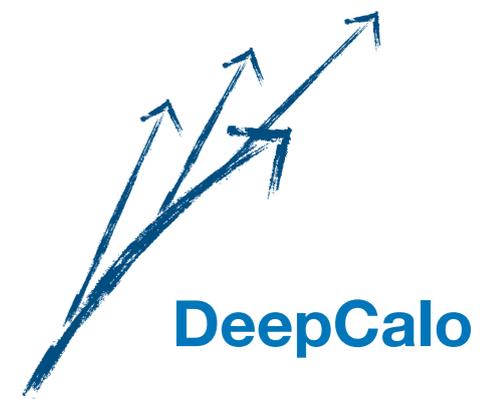
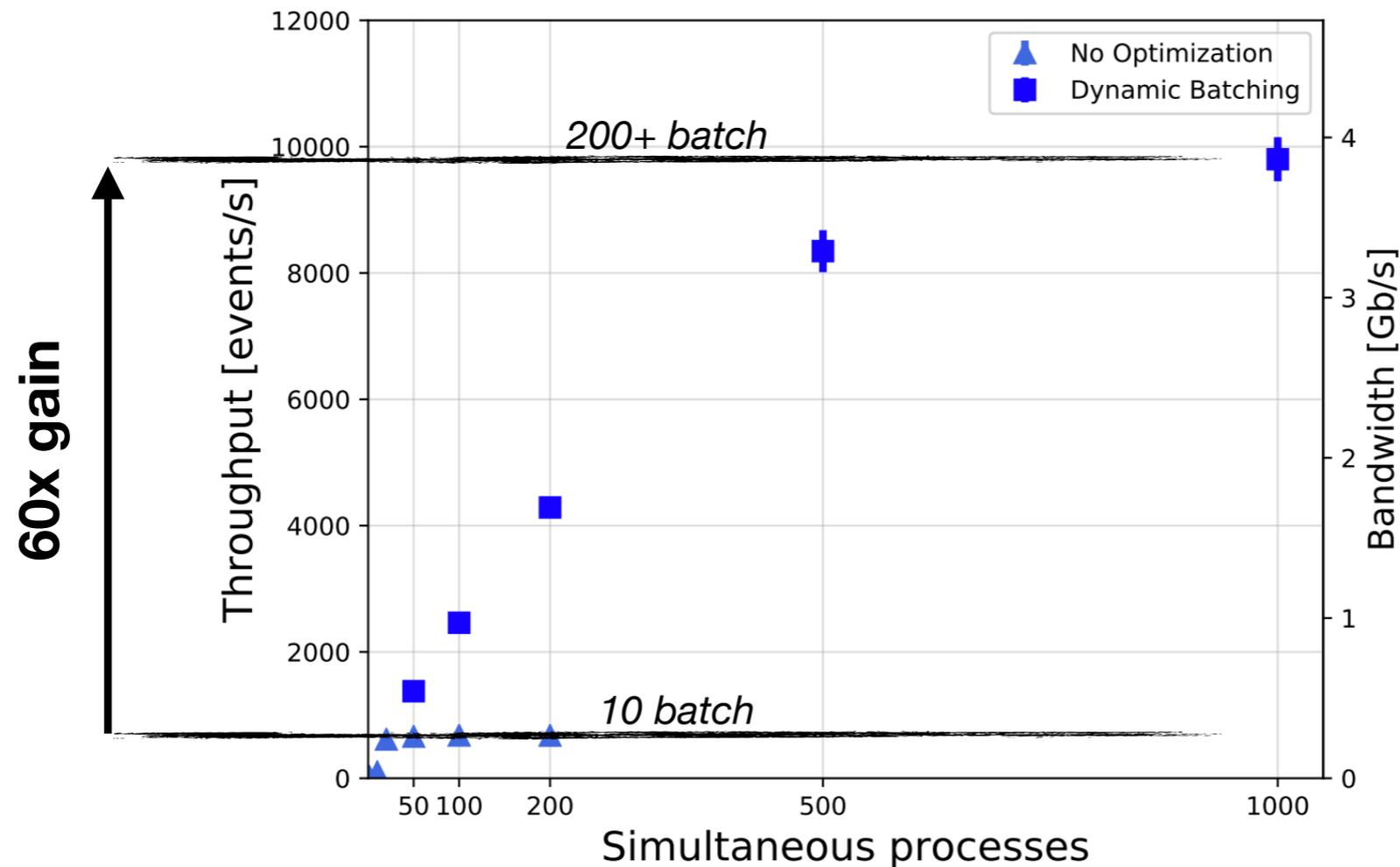
- Allows server to wait for requests to build up
- Most beneficial for small-batch algorithms
- Can extend event-by-event processing to multi-event processing
 - Transparent to user
- Single-line change to server configuration



```
dynamic_batching {  
  preferred_batch_size: [ 100 ]  
}
```

Can also specify max wait time

Dynamic Batching

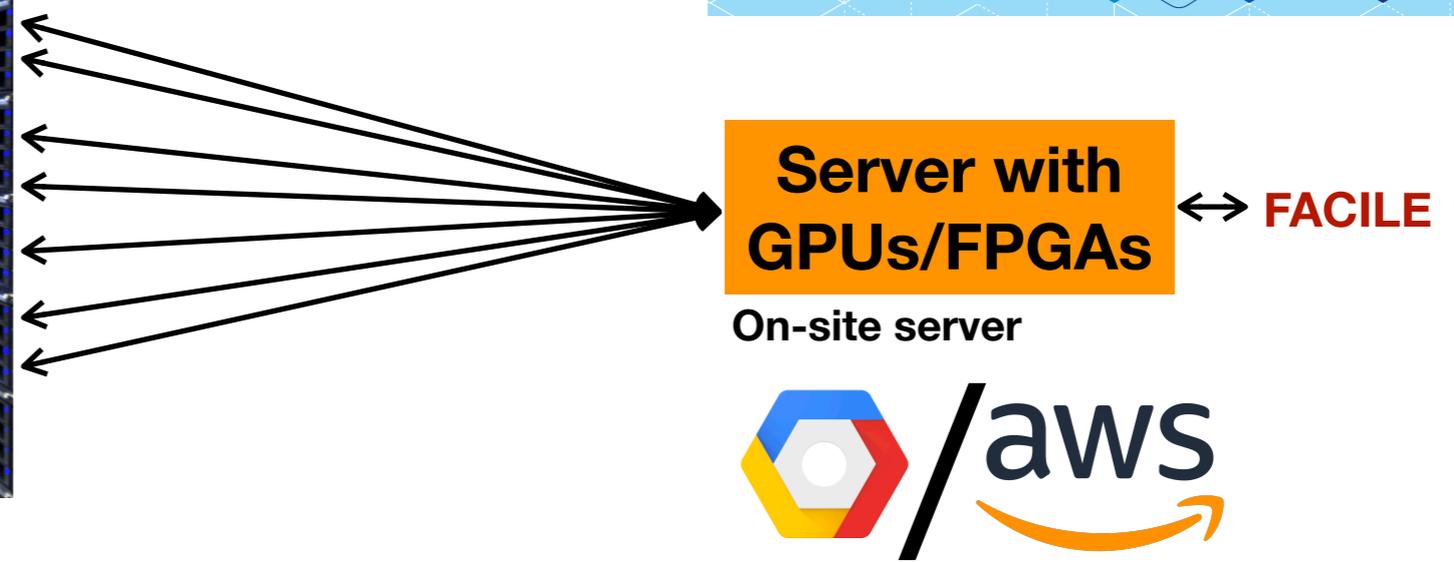


- 60x throughput gain in this case
 - 10k events/s for 1M weight model
 - ~1000 simultaneous clients to saturate single GPU

Scalability



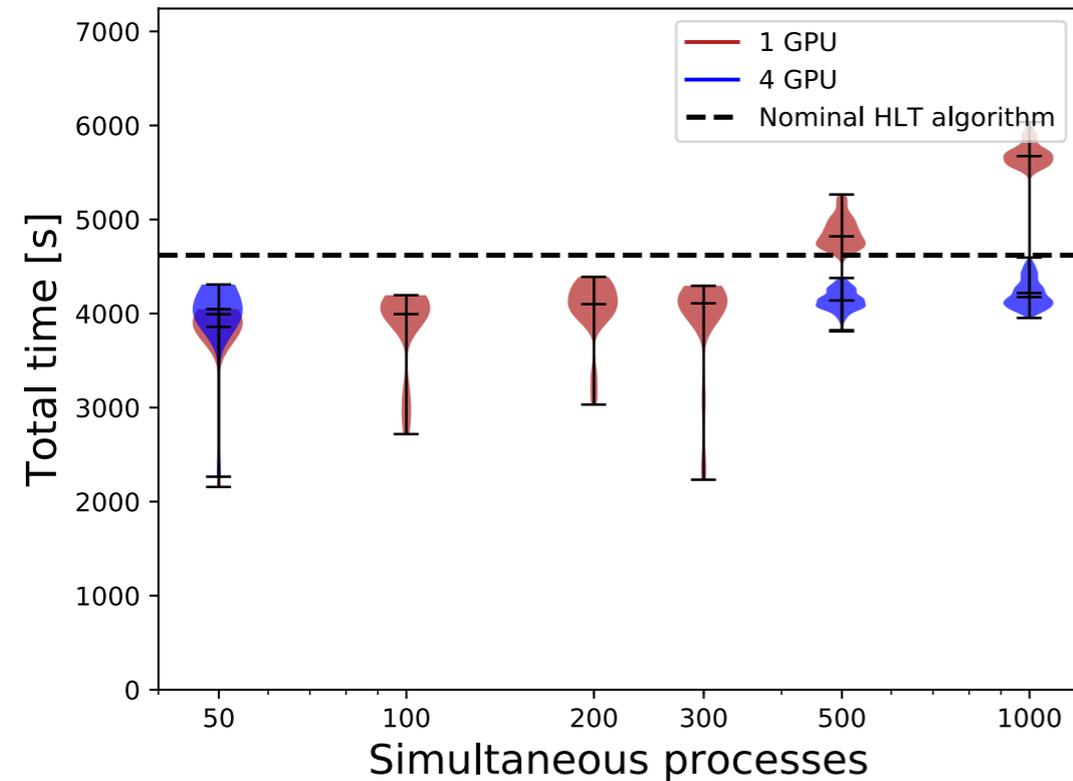
HLT emulation



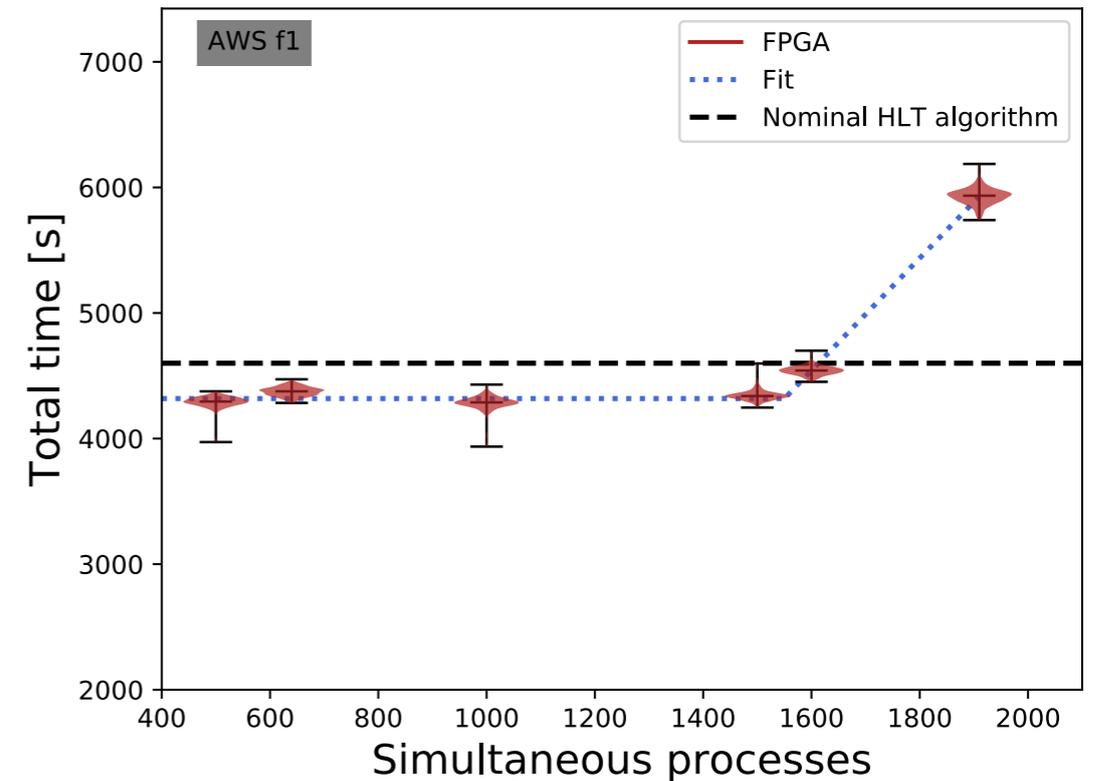
- Used FACILE in CMS HLT workflow to test as-a-service model in realistic computing environment
 - Replaced current (non-ML) MAHI algo
- Use of cloud resources allows at-scale test

Coprocessor Scalability

V100 GPU



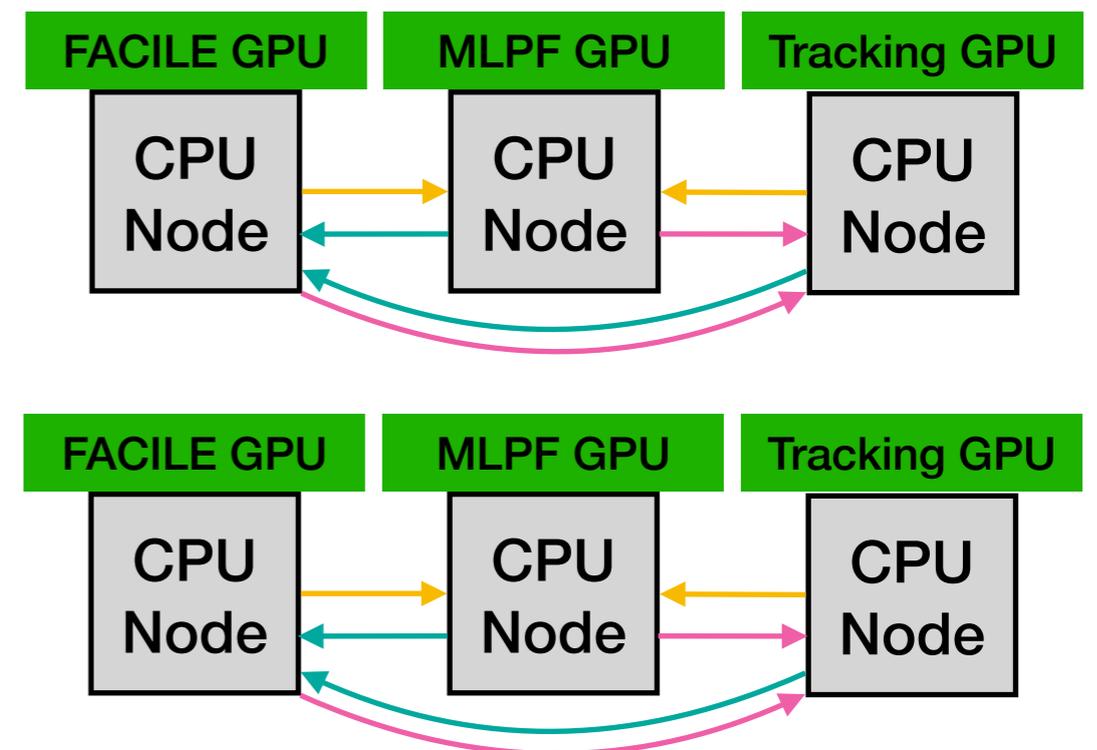
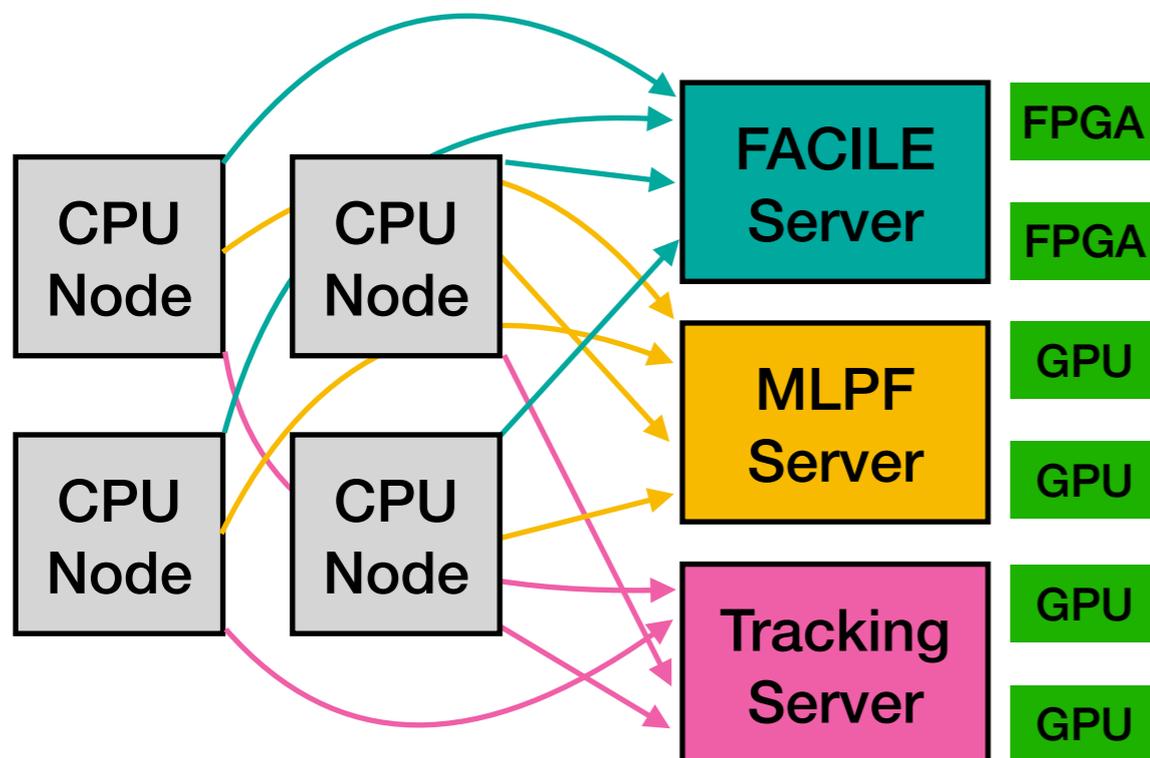
AWS f1 FPGA (VU9P)



- **10% reduction in computing time operating as-a-service**
 - Consistent with fraction of time spent on HCAL local reco w.r.t total HLT time
 - → Maximal achievable reduction for this single algorithm
- No increase in latency until 300/1500 clients (GPU/FPGA)
 - Single device can service 300/1500 HLT instances

Summary

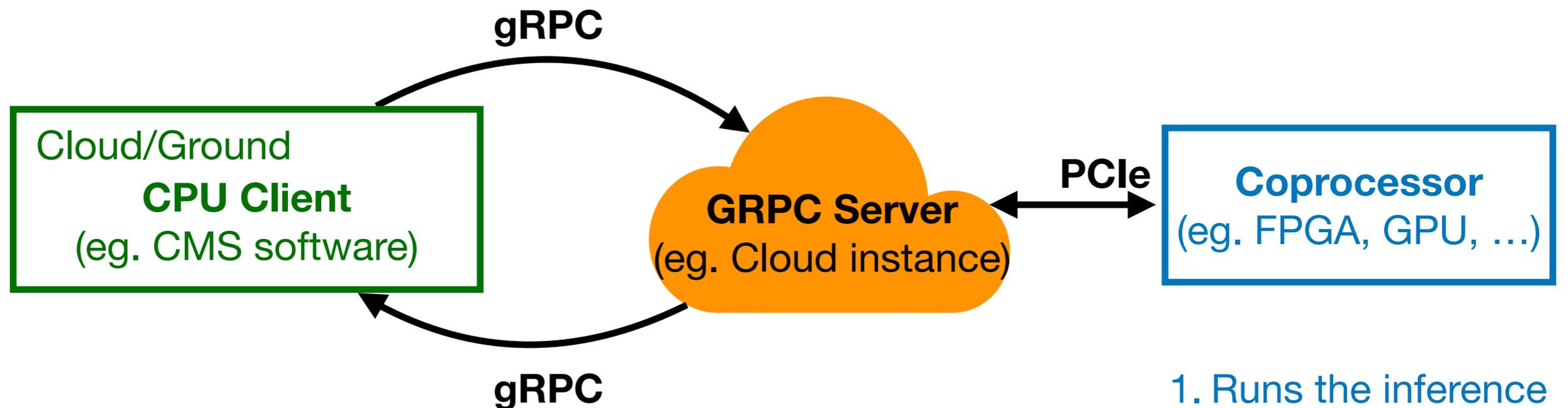
- As-a-service computing has many existing tools that we can leverage to address HEP computing challenges
 - Very cohesive with ML usage, extremely simple for end user
- Papers detailing GPUaaS ([2007.10359](#)), FPGAaaS ([2010.08556](#))
- Many options to consider for implementation



BACKUP

Setup

- For fast inference we focus on gRPC protocol
 - Open source remote procedure call (RPC) system developed by Google



1. Formats inputs
2. Sends asynchronous, non-blocking gRPC call
3. Interprets response

1. Initializes model on coprocessor
2. Receives and schedules inference request
3. Sends inference request to accelerator
4. Outputs and send results
5. Monitors network/device utilization

1. Runs the inference



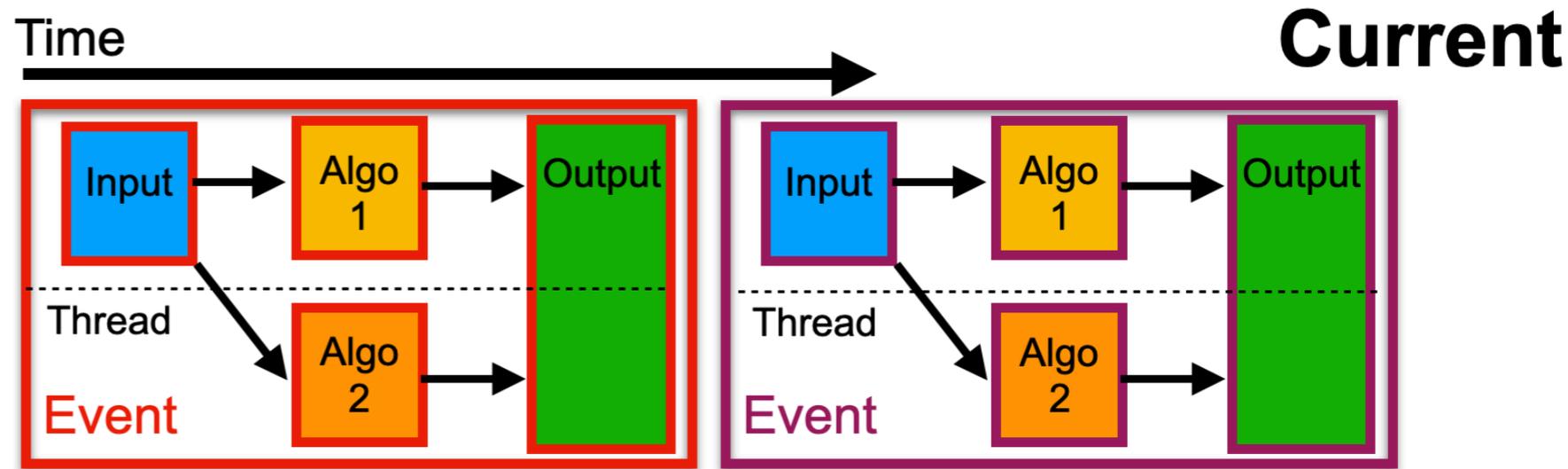
Use NVidia triton inference server for GPU + Customized GCP Kubernetes



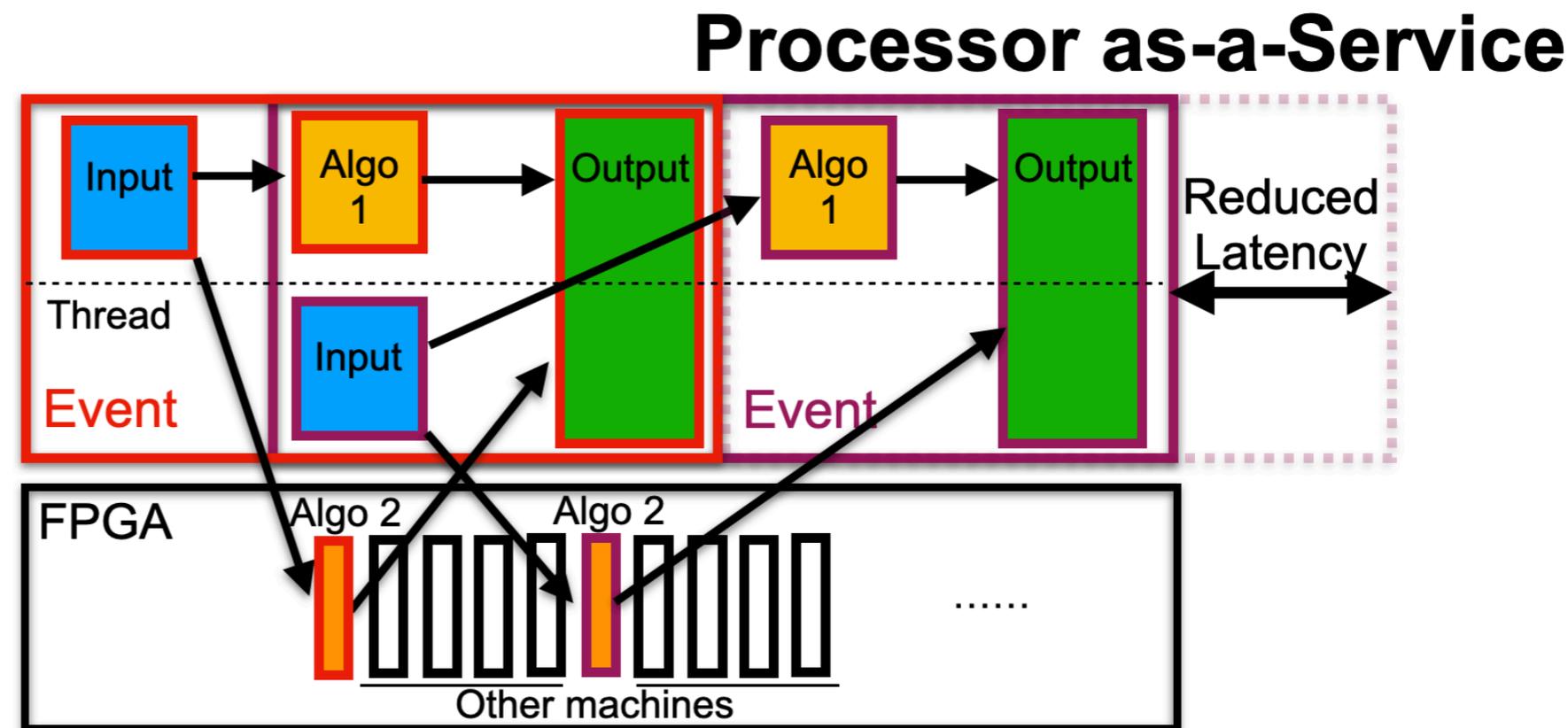
Wrote our own FPGA gRPC inference server

As-a-service Computing

- Can provide large speed up w.r.t traditional computing model
- In principle, as-a-service can be used for any algorithm

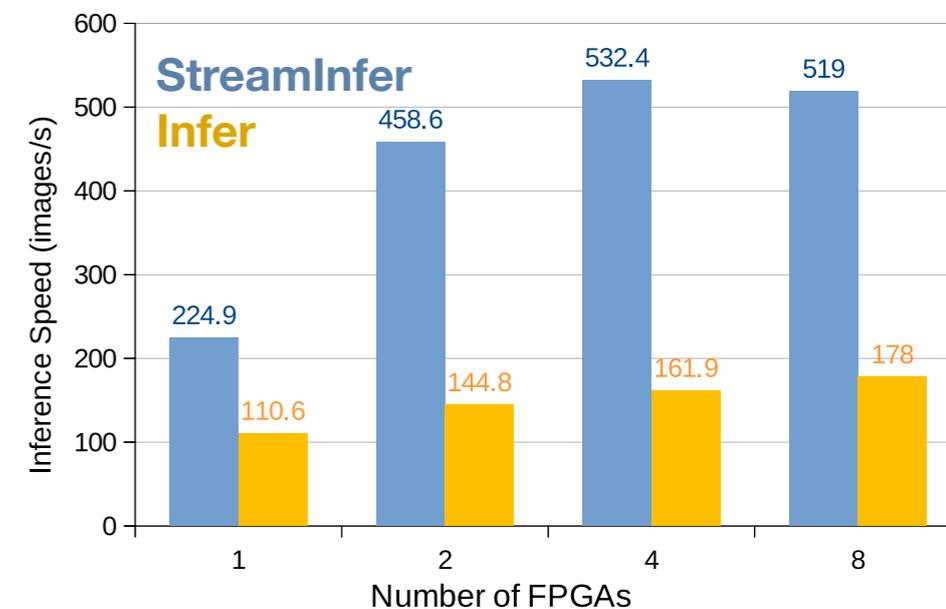
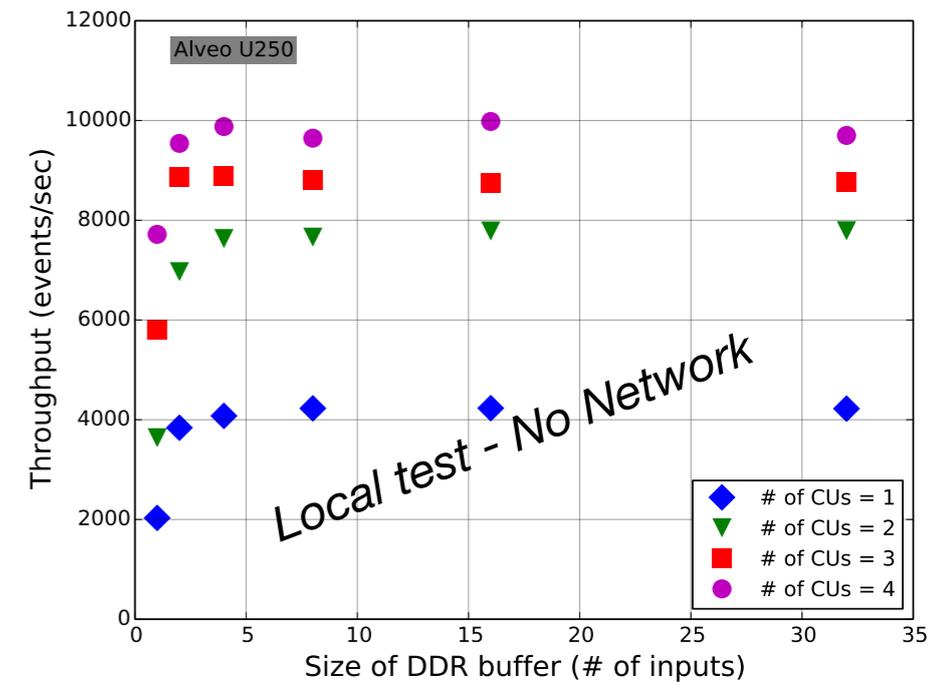


- Simply send all inputs to server, server returns outputs
- Just need server able to accept requests and communicate with GPU or FPGA



FPGA Server Design

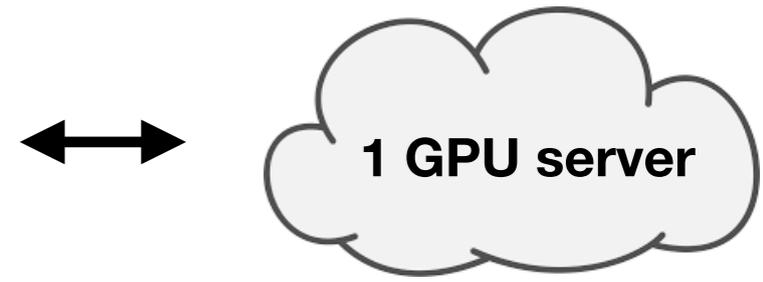
- Same workflow developed for FPGA coprocessors
 - gRPC base (Triton calls), same config as for running on GPU
 - **FACILE:** hls4ml (Alveo U250 & AWS f1)
 - **DeepCalo:** hls4ml (ongoing work)
 - **ResNet:** Xilinx ML Suite (AWS f1)
 - **ResNet:** Microsoft Azure ML Studio (Azure Stack Edge)
- Many design settings to optimize



1 GPU Server



Google Cloud Platform



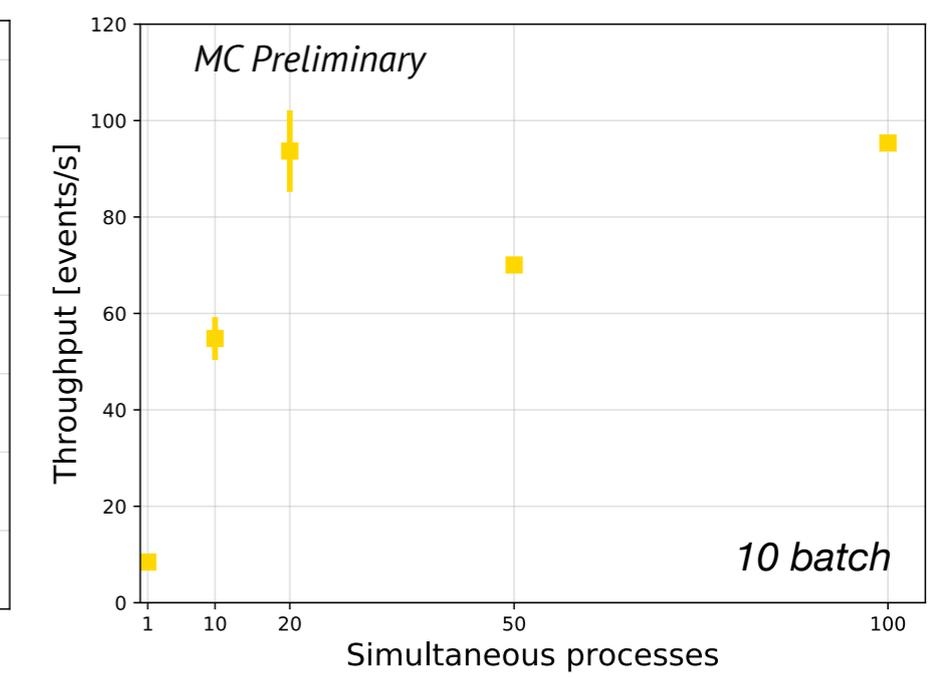
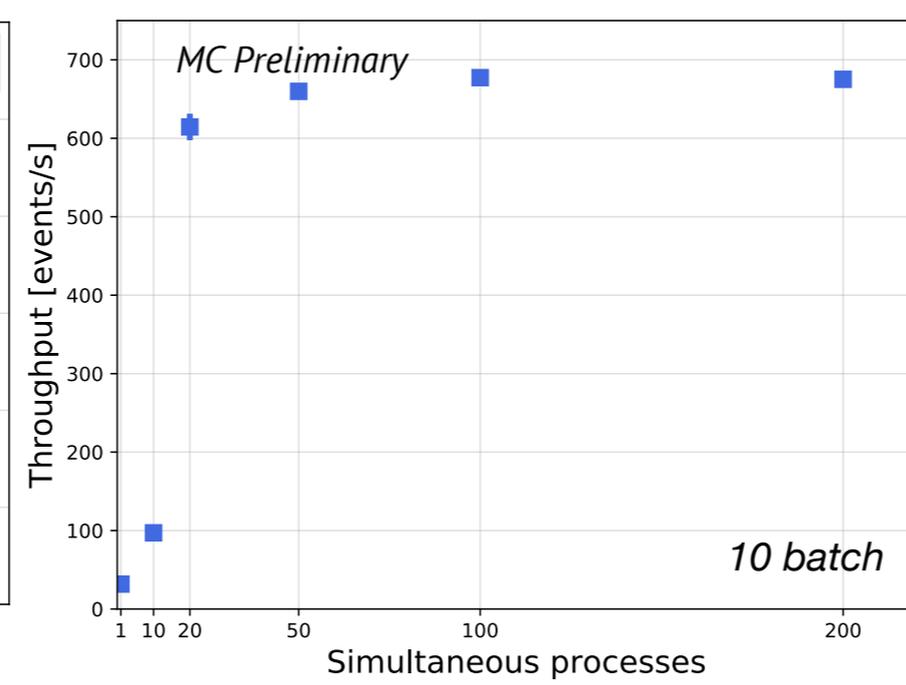
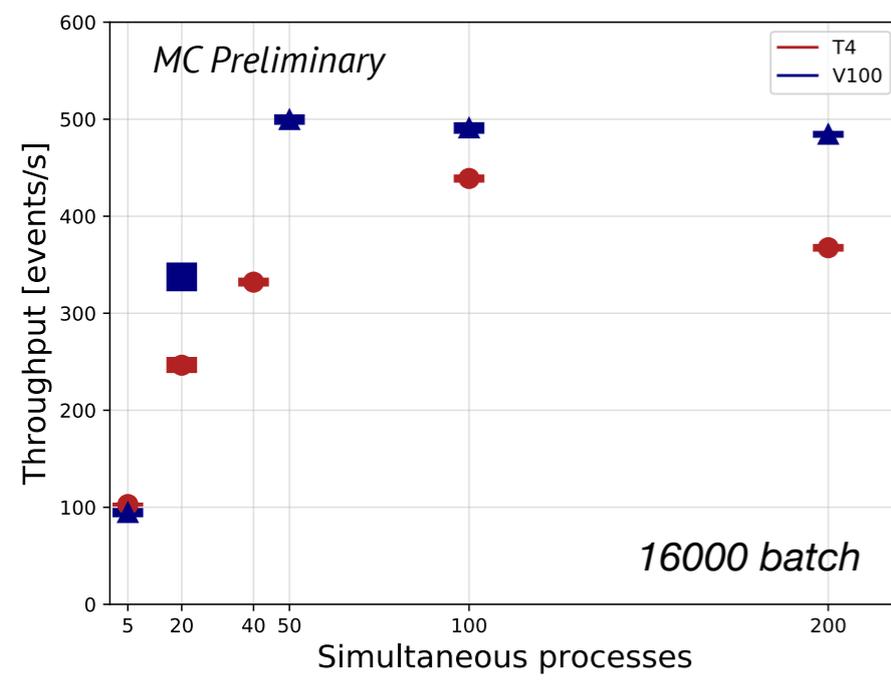
- Inference performed in CMS workflow
- Larger models saturate with fewer clients, lower throughput
- Range of performance for GPUs



FACILE

DeepCalo

ResNet



FPGA Server



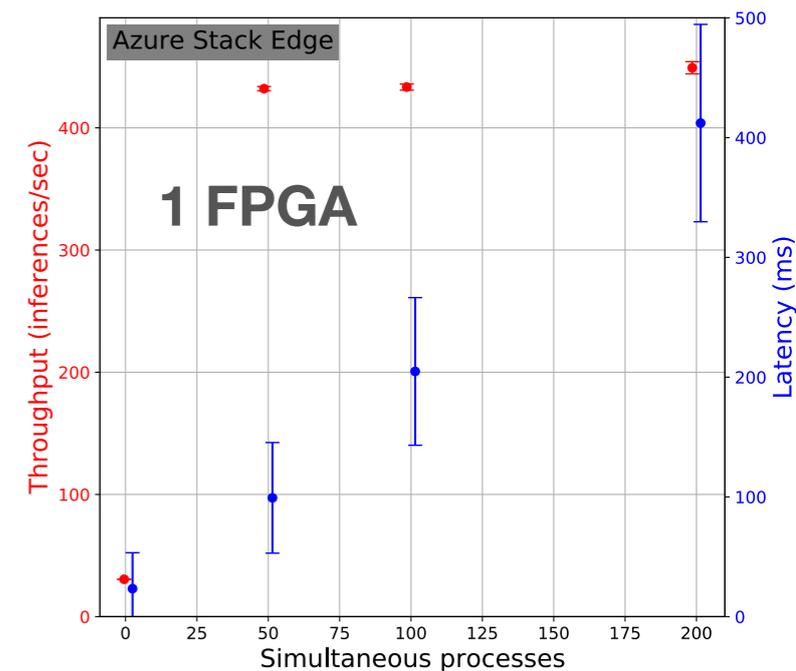
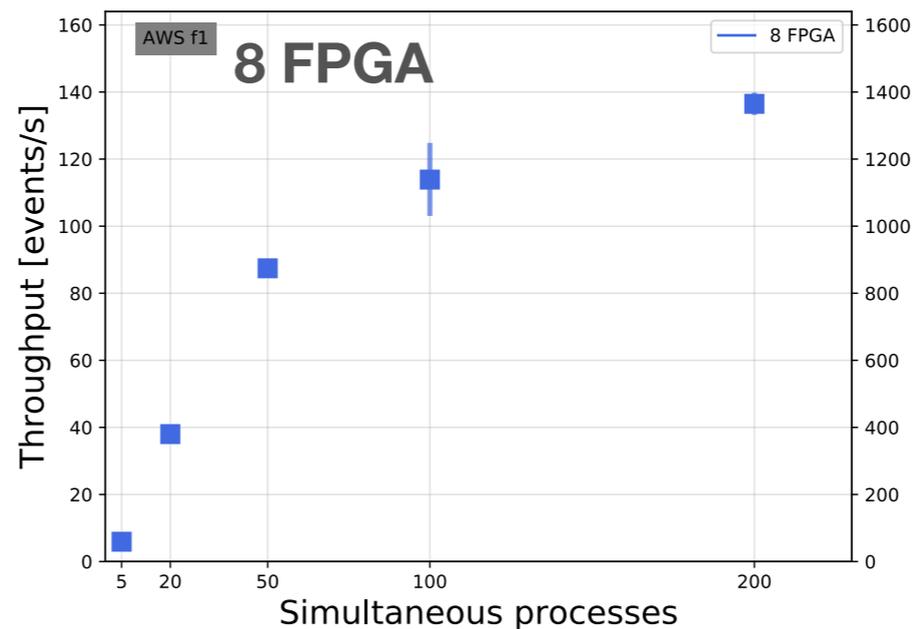
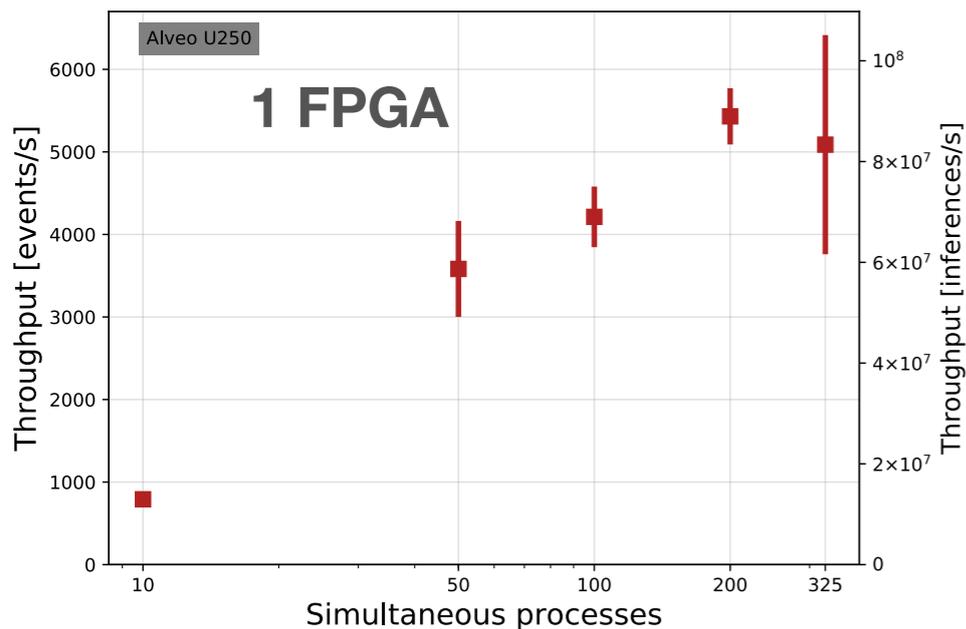
- With small **FACILE** network, major speedup w.r.t. GPU (500 evt/s)
- Limitation from CPU
- For larger **ResNet**, comparable or slightly better throughput w.r.t GPU



FACILE

ResNet

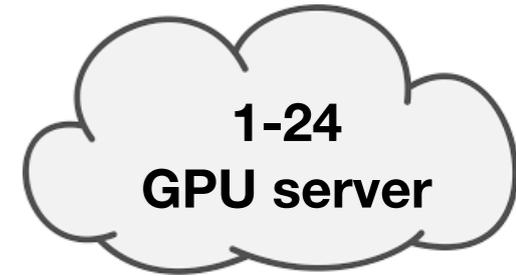
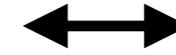
ResNet



Multi-GPU Server

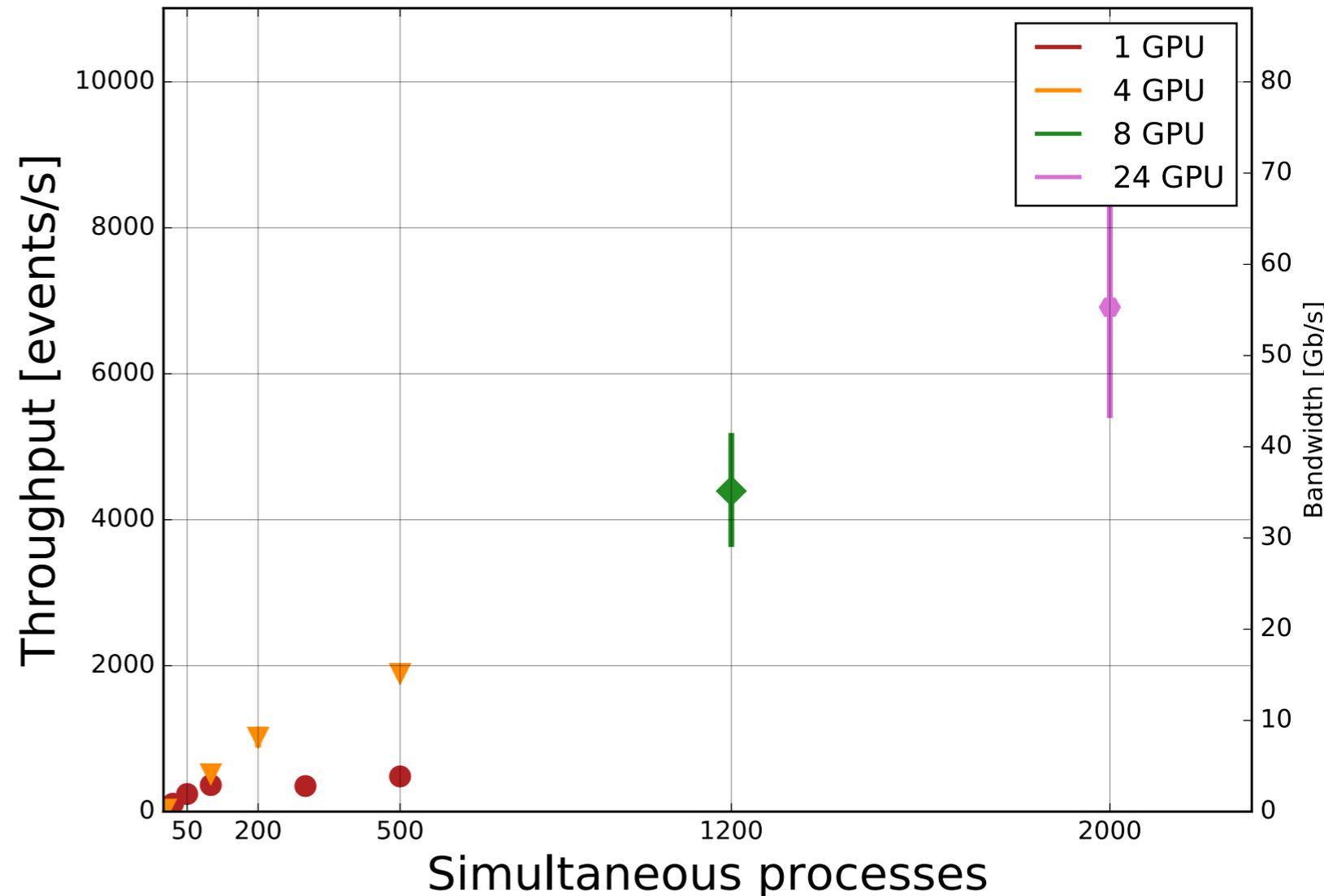


Google Cloud Platform



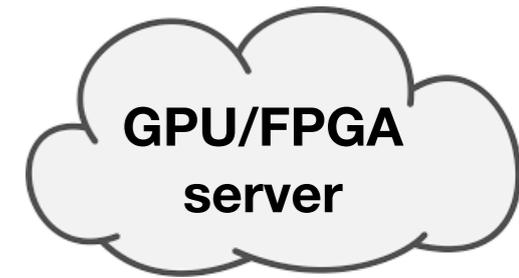
- High bandwidth, long distance (MIT and Google Cloud US-central)
- Linear scaling with # of GPUs
- Throughput saturates at ~60 Gbps (8000 events/s)

FACILE

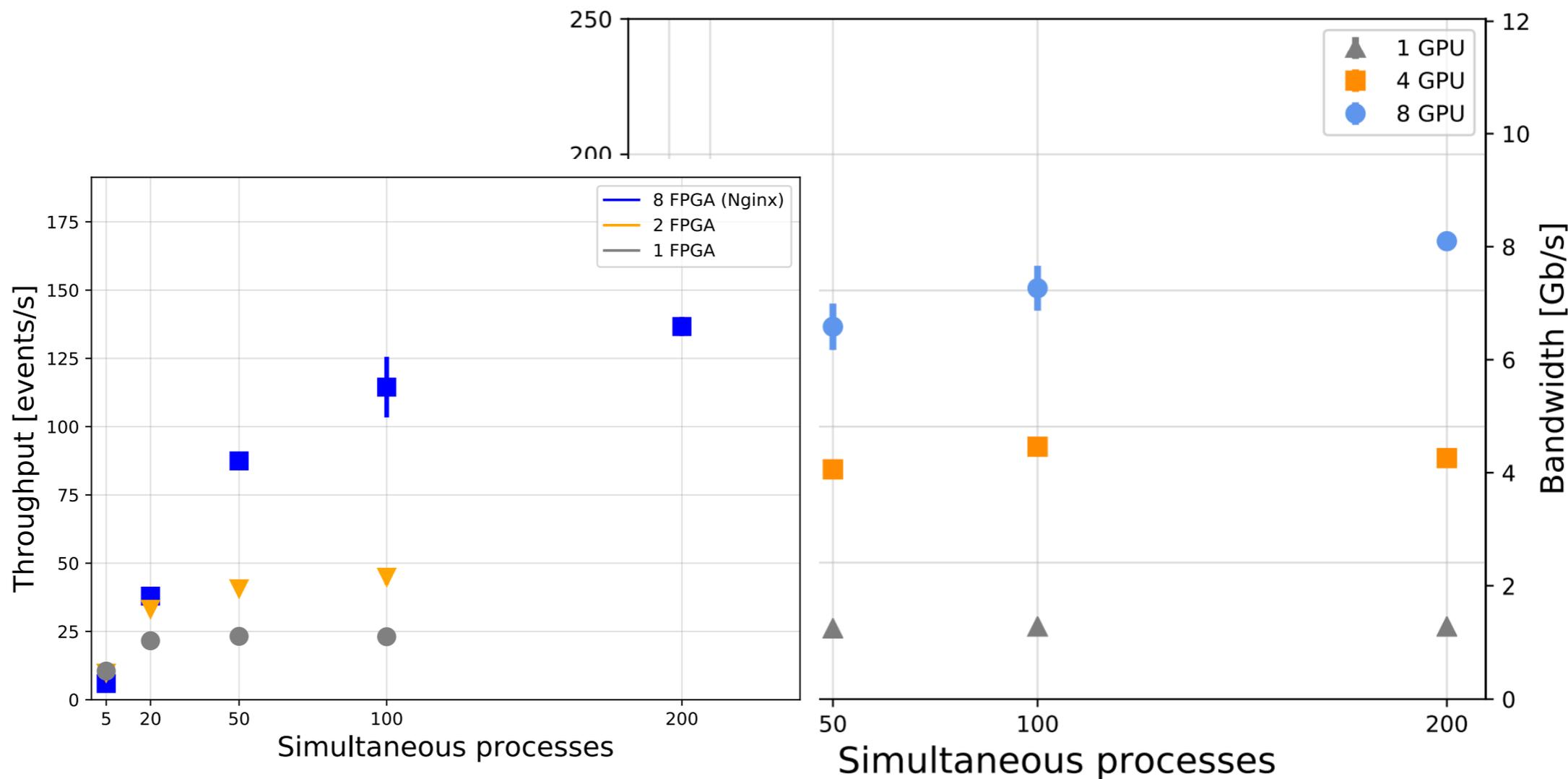
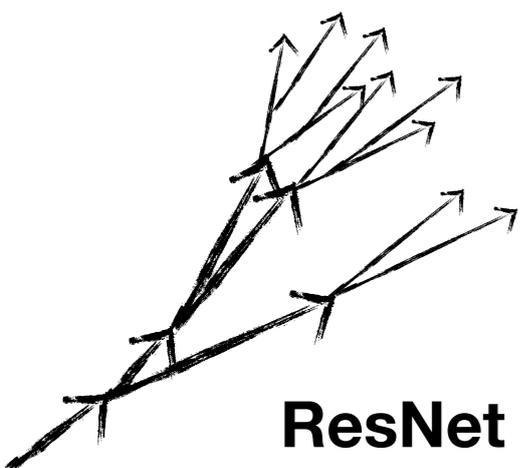


8 GPU/FPGA Server

- Similar performance between GPU and FPGA



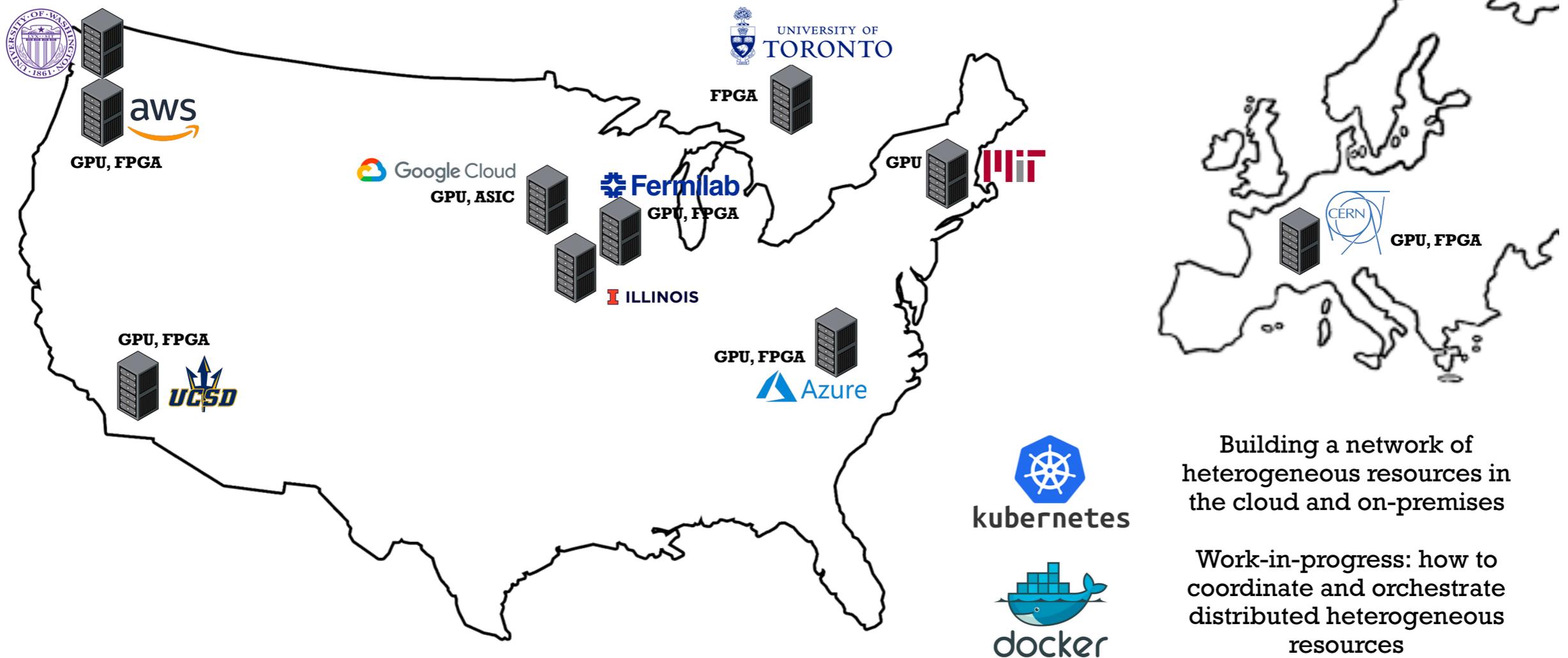
- ~150 evt/s



Tools

Our tools for prototyping CMS reconstruction as-a-service

1. Google Cloud/Amazon Web Services/Microsoft Azure
2. T2/T3 clusters
3. local server/accelerator hardware

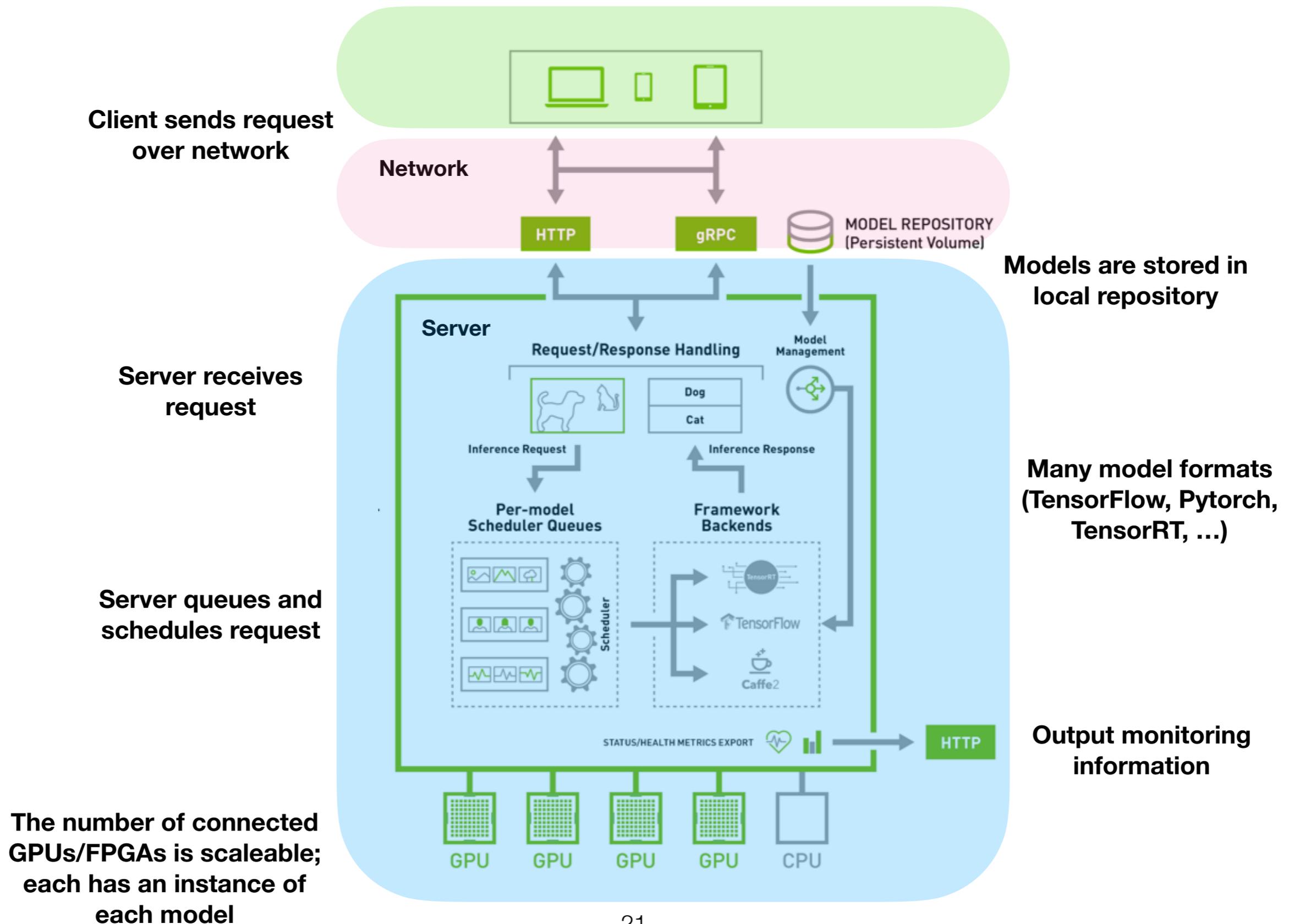


Building a network of heterogeneous resources in the cloud and on-premises

Work-in-progress: how to coordinate and orchestrate distributed heterogeneous resources

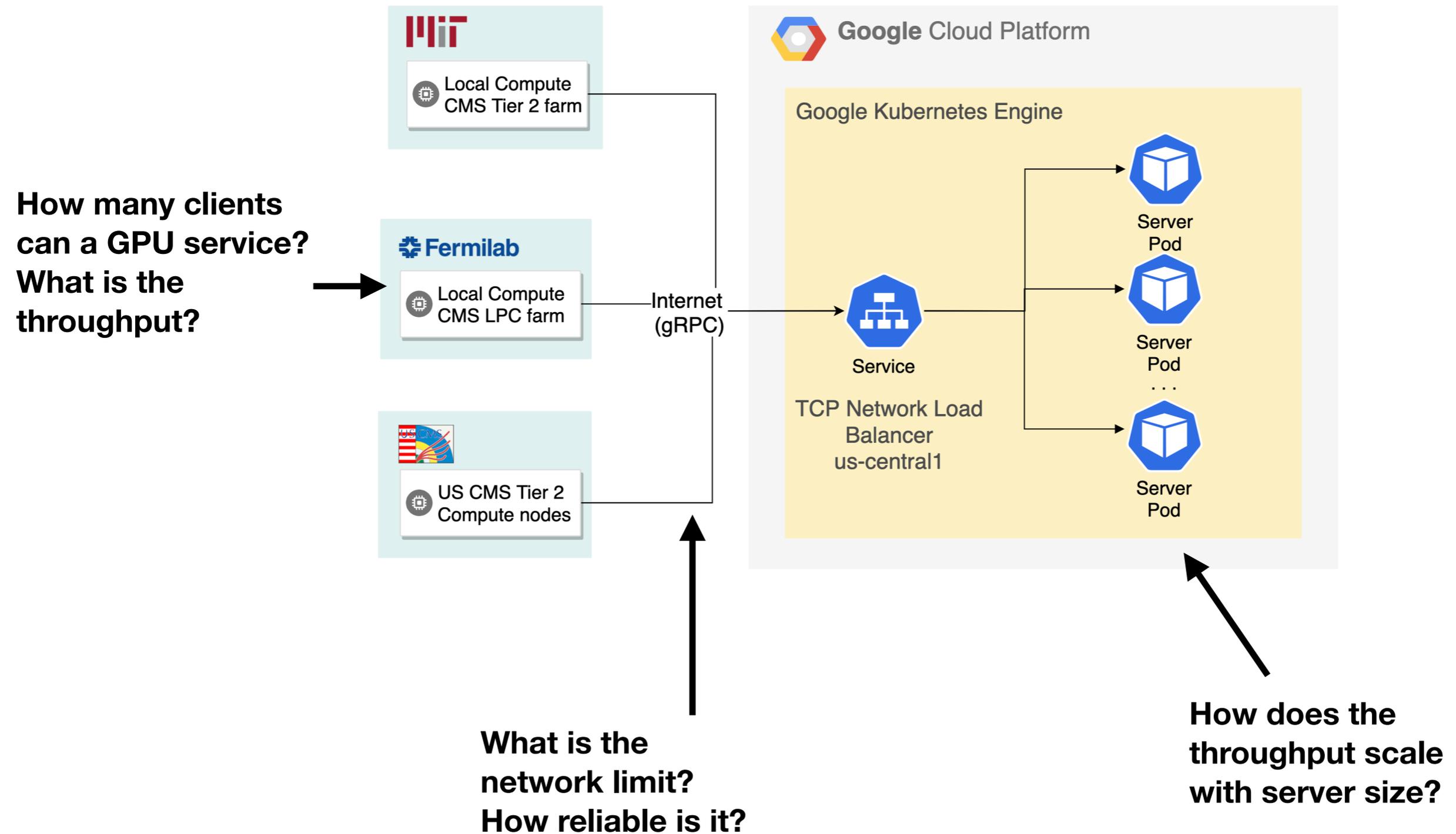
We have a wide network of resources, and perform at-scale tests with many different client-servers configurations, with servers both remote and on-site

Triton Inference Server

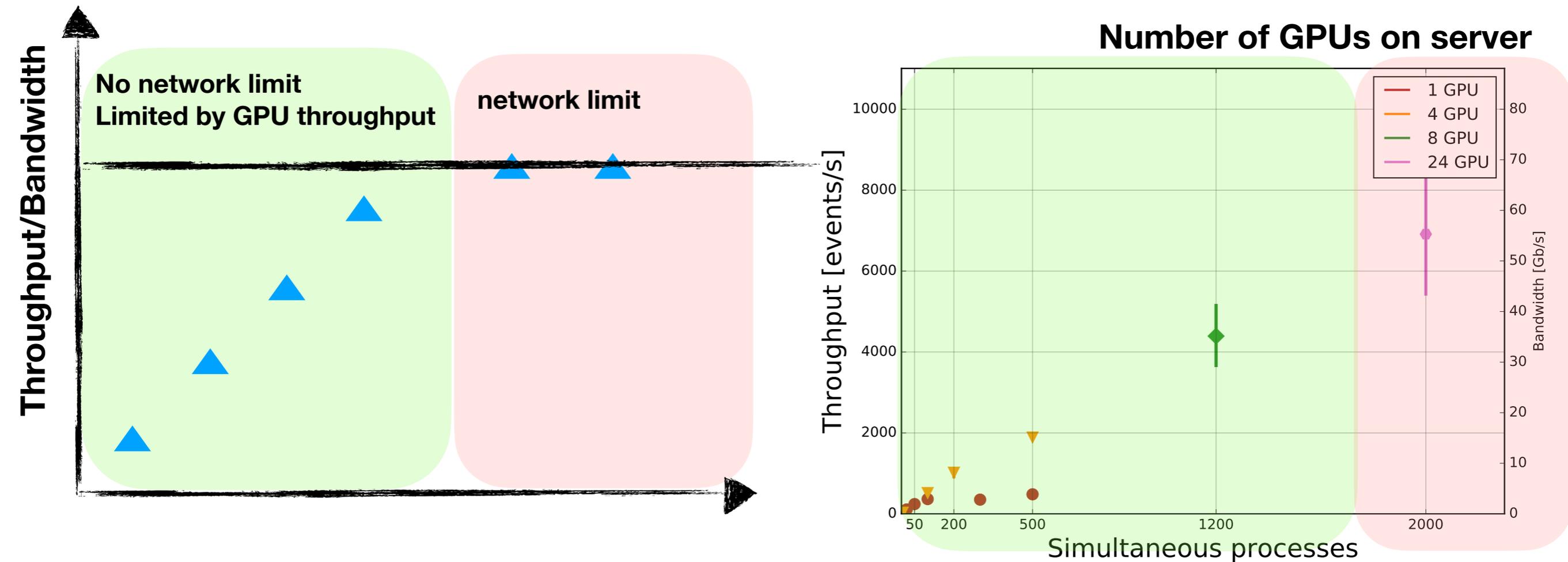


Scalability

A client-server schematic



Network Limit



- Server-on-site: no bandwidth limit found
- Remote server: egress limit at 70 Gb/s for MIT T2
 - Exceeds needs for use cases considered