Introduction to SONIC + Triton (ML inference as a Service)

Yongbin Feng, Kevin Pedro (Fermilab)

Machine Learning HATS@LPC

July 12th, 2021

Introduction

- Now that you have trained a neural network. **How to deploy the model and run a large scale production (‘inference’) on some datasets?**
  - ✤ CMSSW supports TF1, and started supporting TF2 since CMSSW_11_1_X (cmssw#28711, cmsdist#5525), and ONNX, so that you can run direct inference on CPUs with TF1, TF2, and ONNX models.
  - ✤ What if you have a Pytorch model instead?
    - ➡ Converting it to an ONNX model and run inference with CMSSW is an option
  - ✤ What if you have a GPU (or some other coprocessors), and you want to accelerate the inference with it?
    - ➡ What if the GPU is remote, not directly connected with the CPU clusters?

PYTORCH

TensorFlow

ONNX RUNTIME

CPUs

GPUs

FPGAs

ASICs

Introduction

- Now that you have trained a neural network. **How to deploy the model and run a large scale production (‘inference’) on some datasets?**

  ✤ You can use SONIC + Triton, which will help you solve problems such as:

  ✤ What if you have a Pytorch model instead?

    ➡ Converting it to an ONNX model and run inference with CMSSW is an option

  ✤ What if you have a GPU (or some other coprocessors), and you want to accelerate the inference with it?

    ➡ What is the GPU is remote, not directly connected with the CPU clusters?

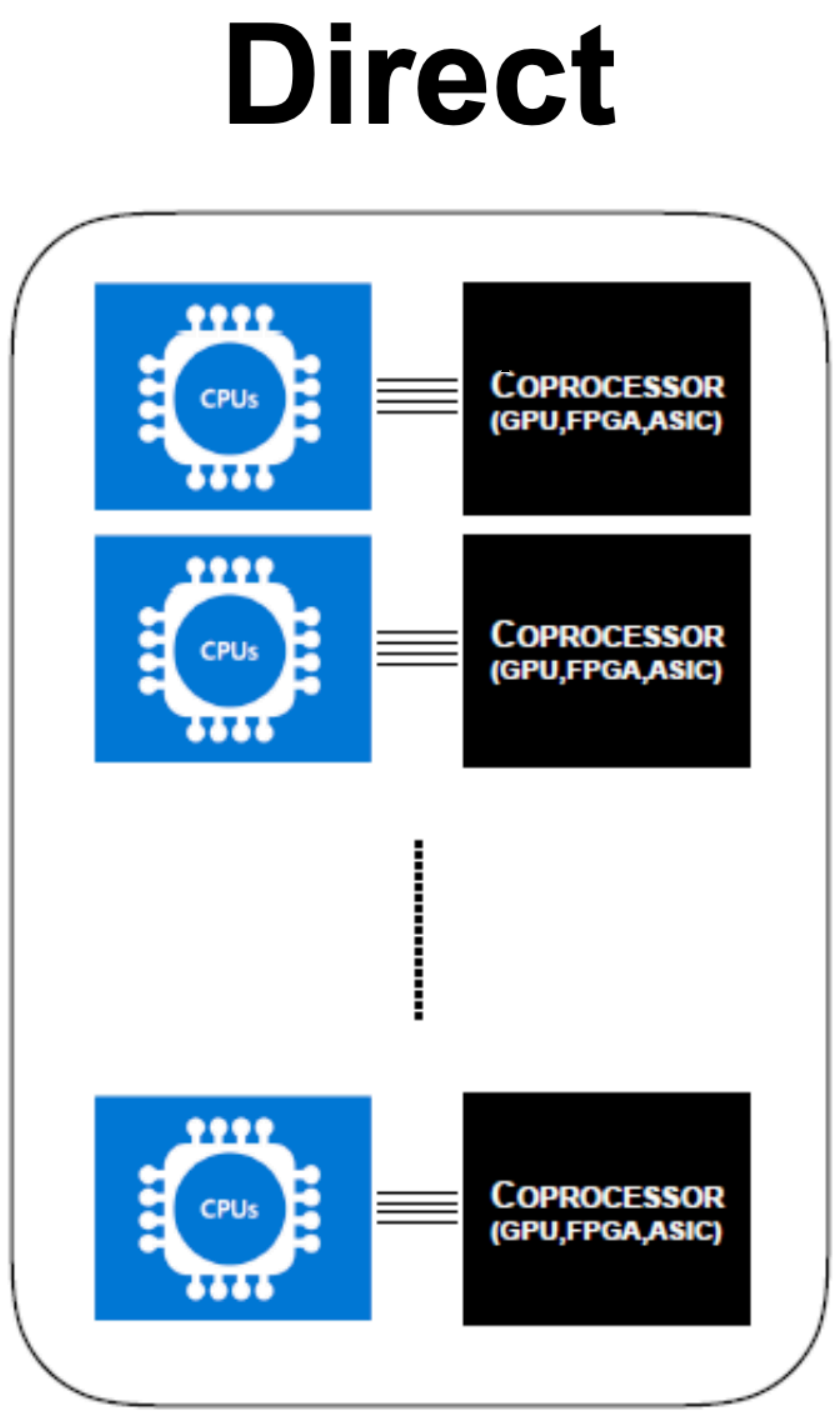
PYTORCH TensorFlow ONNX RUNTIME

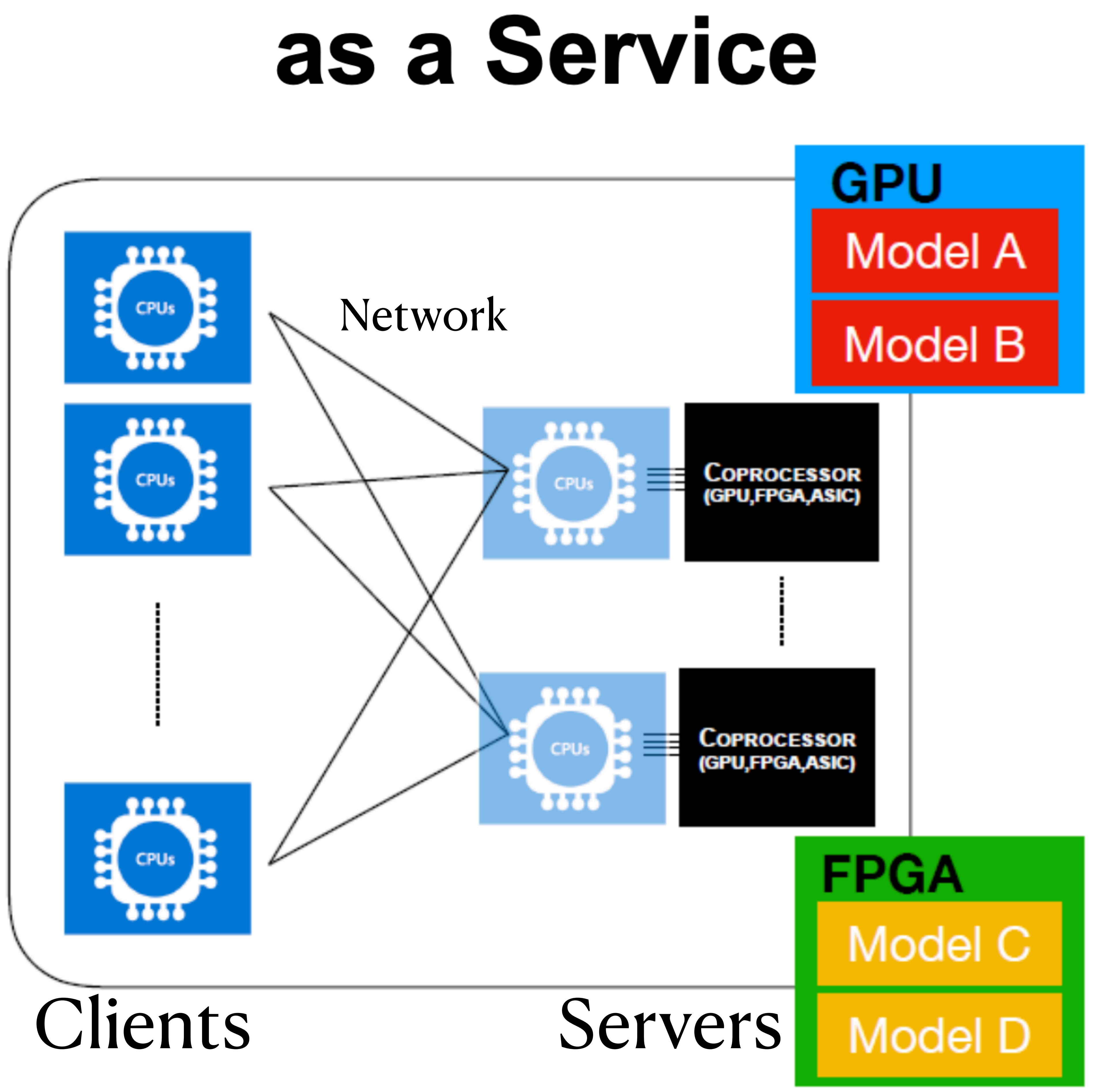
CPUs GPUs FPGAs ASICs

ML Inference Infrastructure

Two ML Inference Infrastructures:

- **Directly connect CPUs and coprocessors**
  - ✤ Inference running on the coprocessors directly connected to the CPU
  - ✤ Simple connection; no network load
- **Inference as a service (aaS)**
  - ✤ Clients communicate with the server, prepare the model inputs to the server and receive model outputs from the server
  - ✤ Server directs the coprocessor for model inference



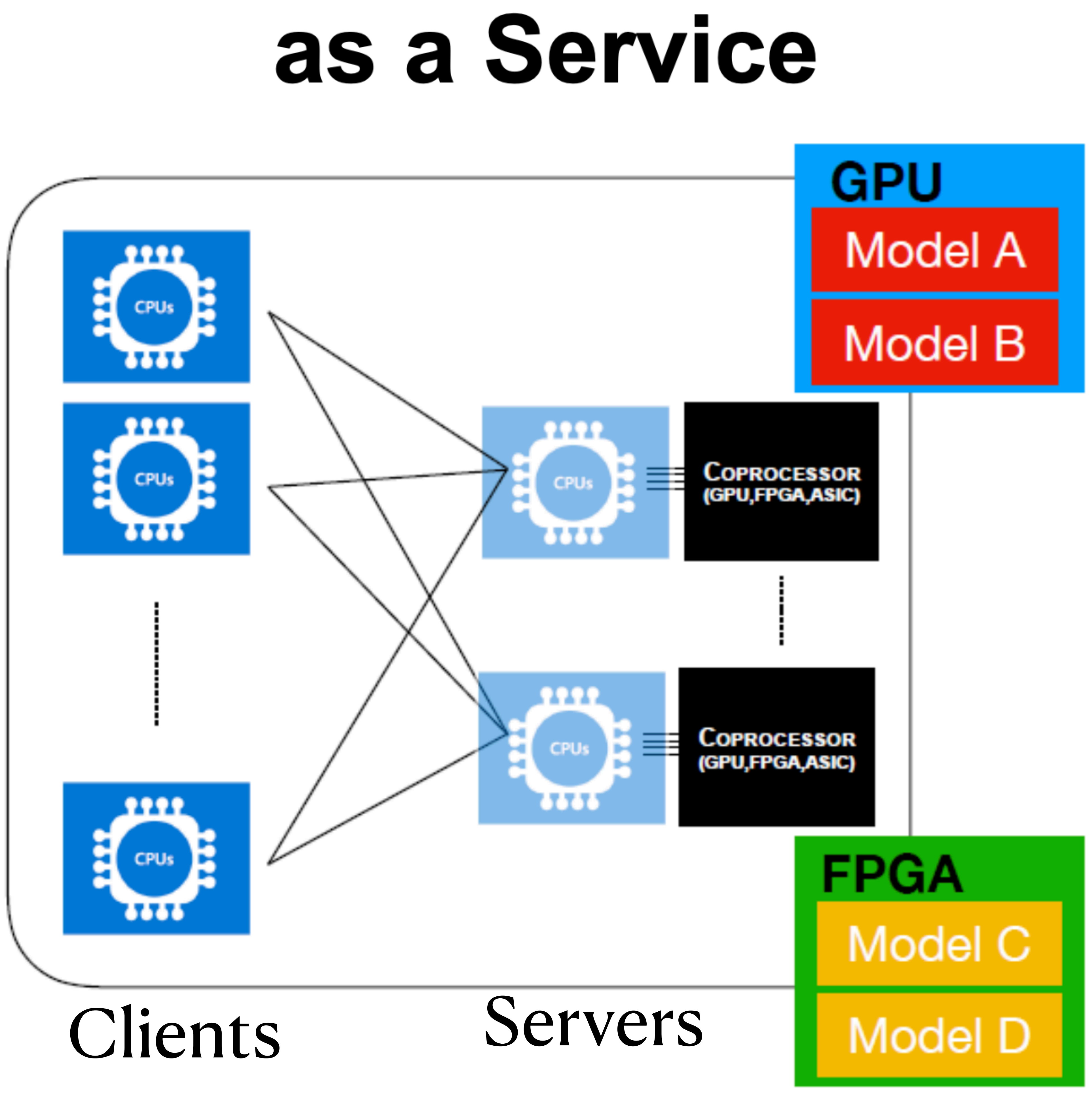


ML Inference aaS

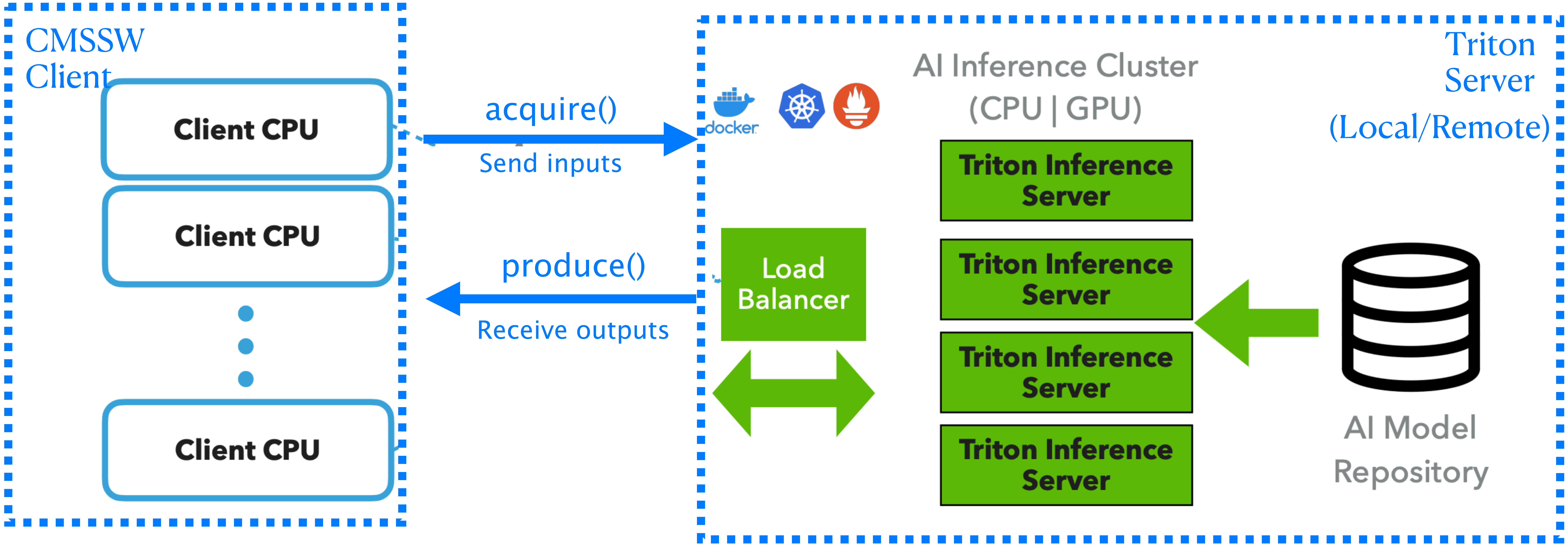
Benefits of running Inference aaS:

- **Factorize the ML framework out of CMSSW**
  - ✤ Only need to handle input and output conversions on the client side (i.e., in CMSSW). Different frameworks supported on the server side.
- **Simple support for different coprocessors:**
  - ✤ No need to rewrite algorithms in coprocessor-specific languages
- **More flexibility**
  - ✤ One coprocessor can serve many CPU clients
  - ✤ ML models can be deployed on different coprocessors simultaneously; choose the best coprocessor for each specific job
- **Can be more efficient in the direct connect case, with GPU shared memory**
- **Is the only option to access remote GPUs (Currently no other way to run GPU code if no local GPU is available)**

Existing tools available from industry, cloud



SONIC in CMSSW



- SONIC (Service for Optimized Network Inference on Coprocessors) available in CMSSW
- The Client in CMSSW sends the inference request with inputs for the model, and receives the outputs from the server
- NVIDIA Triton server runs the inference

SONIC Framework in CMSSW

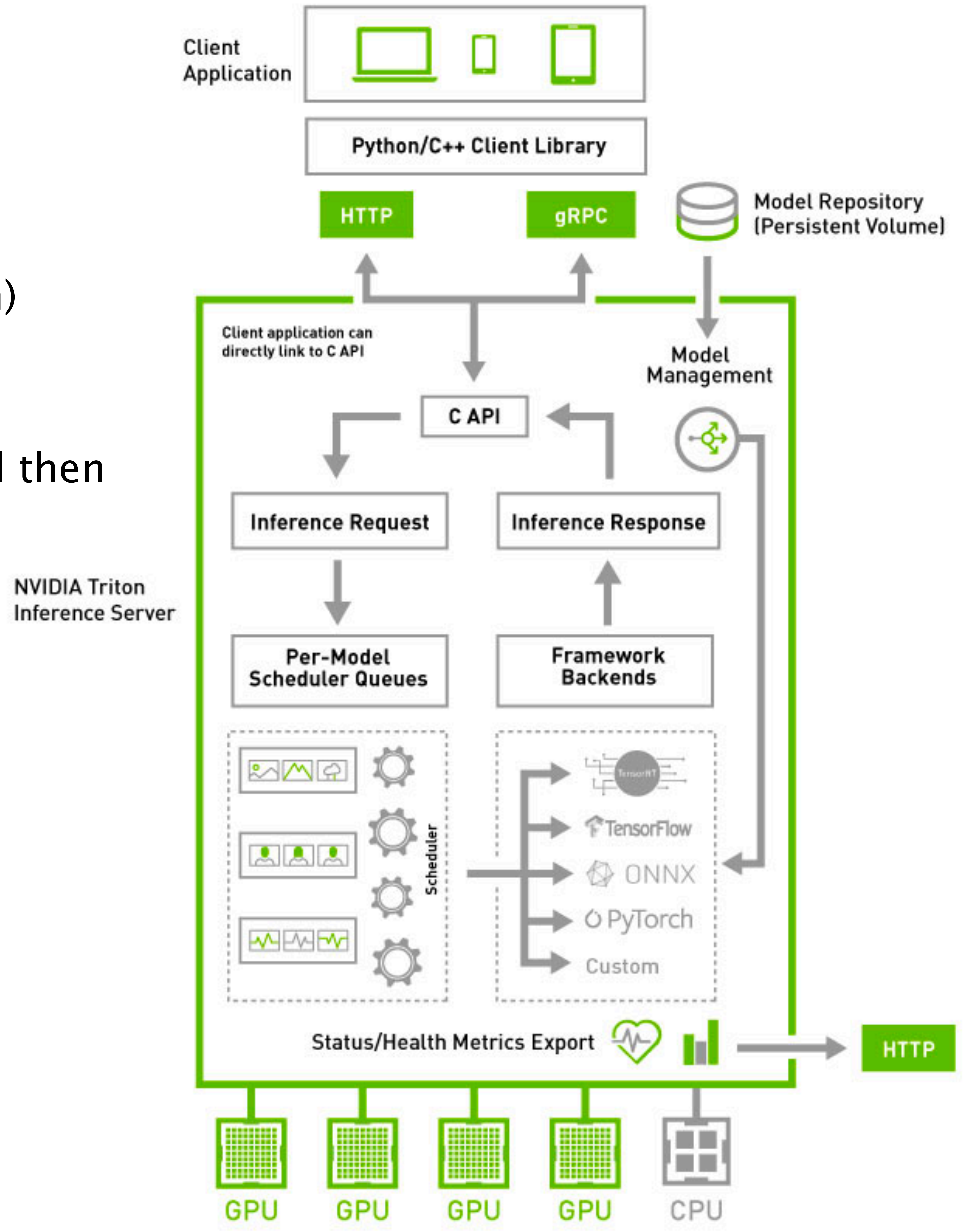
- **SonicCore (repo)**
  - Modules (EDProducer, EDFilter, EDAnalyzer) and client based classes
  - Synchronous and Asynchronous modes for clients
- **SonicTriton (repo)**
  - Modules, clients, data types, and services for Triton inference server
  - cmsTriton script to launch and manage the Triton server via Docker or Singularity

- Examples: ResNet50 and Graph Attention Network available (code)

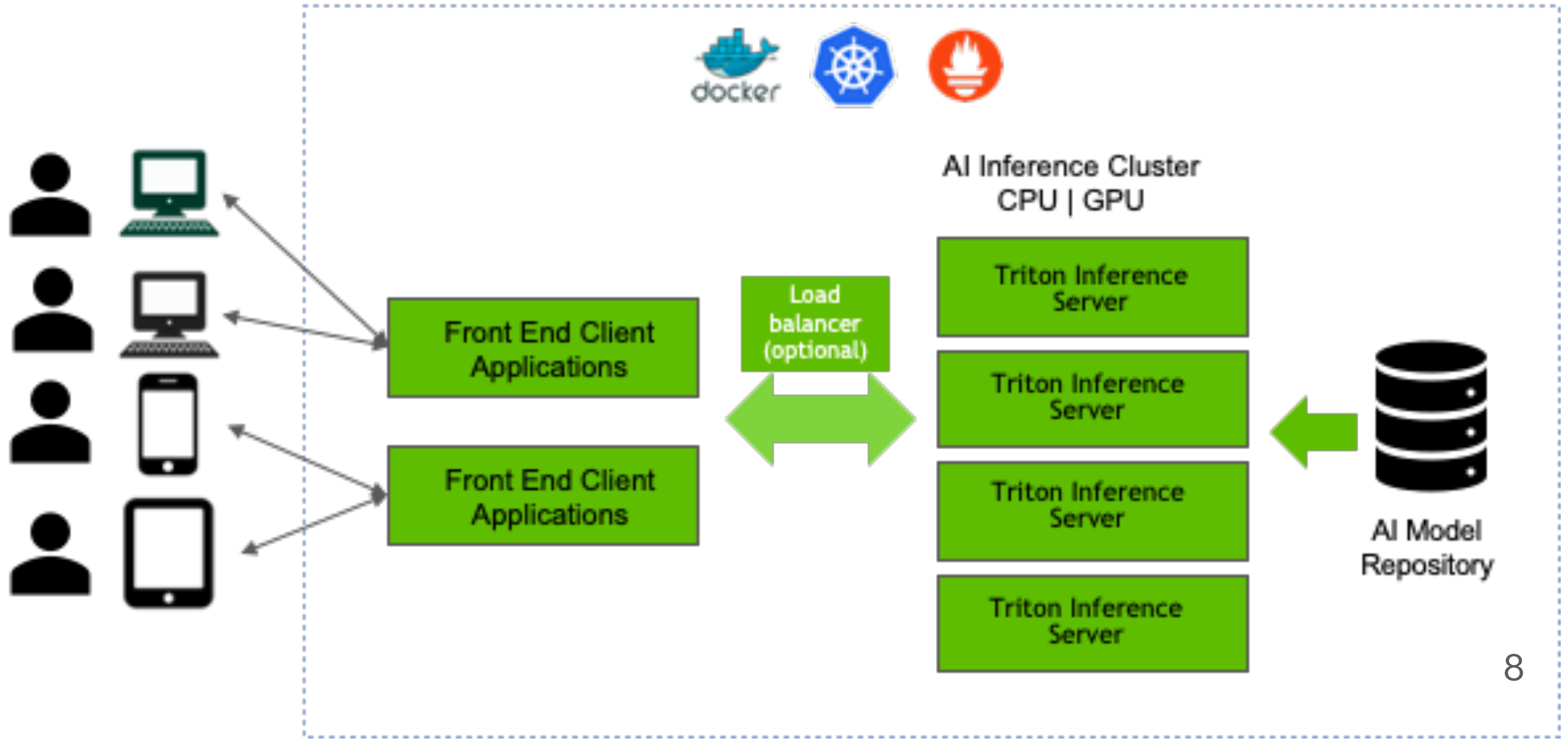
- Besides the Triton server with CPU/GPU, we are also developing the FPGA-as-a-Service Toolkit (FaaST) for FPGAs, which SONIC also supports.

NVIDIA Triton Inference Server

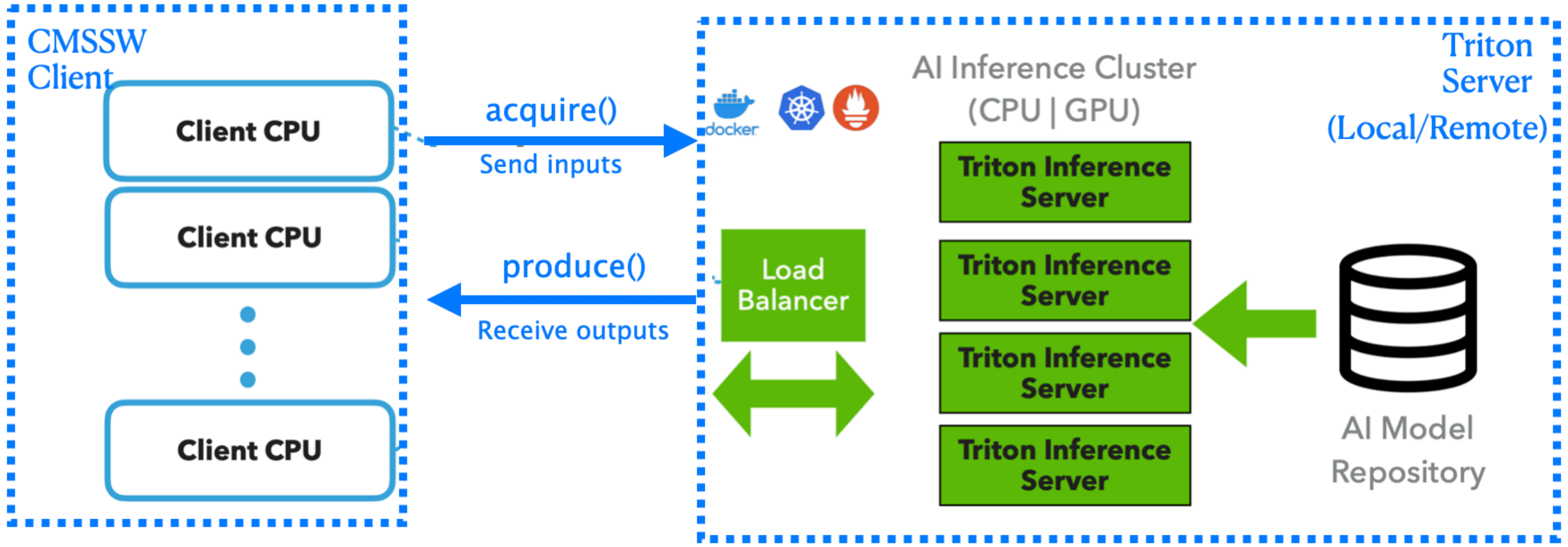
- Open source inference serving software (webpage, repo) that supports numerous backends:
  - TensorFlow, Pytorch, TensorRT, ONNX for ML
  - custom backends for e.g., non-ML algorithms (cpp, CUDA, python)
- Attractive features:
  - Dynamic batching: accumulate requests from multiple events, and then process together
  - Multiple-GPU load balancing



Client Application
Python/C++ Client Library
HTTP
gRPC
Model Repository (Persistent Volume)
Client application can directly link to C API
Model Management
C API
NVIDIA Triton Inference Server
Inference Request
Inference Response
Per-Model Scheduler Queues
Framework Backends
Scheduler
TensorFlow
ONNX
PyTorch
Custom
Status/Health Metrics Export
HTTP
GPU
GPU
GPU
GPU
CPU



Useful Features



Useful features:

- In SONIC Triton producer, the acquire() function prepare inputs to the triton server, and produce() obtains output data from the server
- Input and output tensor shapes can be variable
- TritonService: track all available servers and models
- Local fallback server: automatically launch the local fallback server if the model is not available in the remote servers; can use either CPU or GPU (if available)
- ProcessModifier enableSonicTriton to turn on these features

Useful Links

- Document on the NVIDIA Triton inference server:
  - ✤ https://docs.nvidia.com/deeplearning/triton-inference-server/archives/triton_inference_server_230/user-guide/docs/
- SONIC Core:
  - ✤ https://github.com/cms-sw/cmssw/tree/master/HeterogeneousCore/SonicCore
- SONIC Triton:
  - ✤ https://github.com/cms-sw/cmssw/tree/master/HeterogeneousCore/SonicTriton
- cmsTrion script to launch the triton server:
  - ✤ https://github.com/cms-sw/cmssw/blob/master/HeterogeneousCore/SonicTriton/scripts/cmsTriton
- SONIC + Triton examples:
  - ✤ https://github.com/cms-sw/cmssw/tree/master/HeterogeneousCore/SonicTriton/test