Nvidia Triton Demo: Incorporating AI Inference Into Workflows

NERSC Data Day

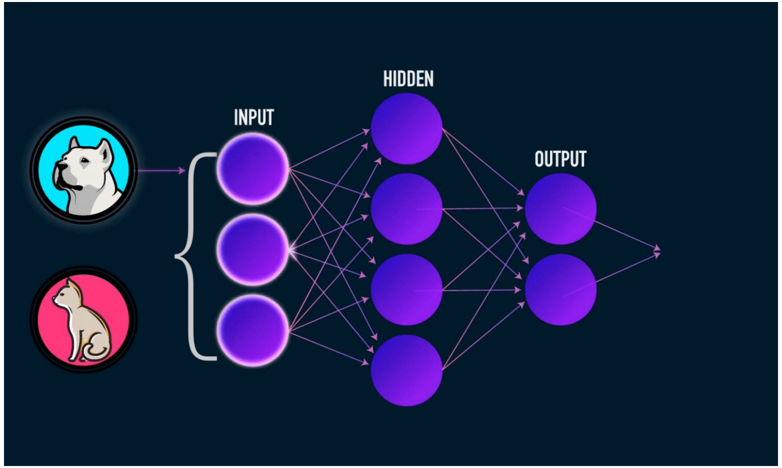
Andrew Naylor
NERSC Postdoc
Feb 21, 2024

Overview

- Inference
- Triton
- Science use case
- Demonstration

Inference

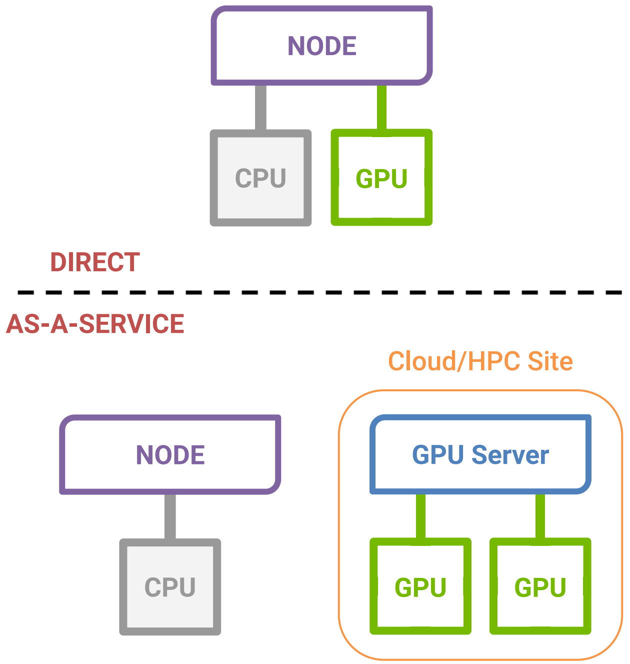
- The process of running new data through a trained AI model to make a prediction or solve a task
- “*Up to 90% of an AI-model’s life is spent in inference mode*” - IBM



Credit: Pallawi - Medium

Using as-a-service vs direct

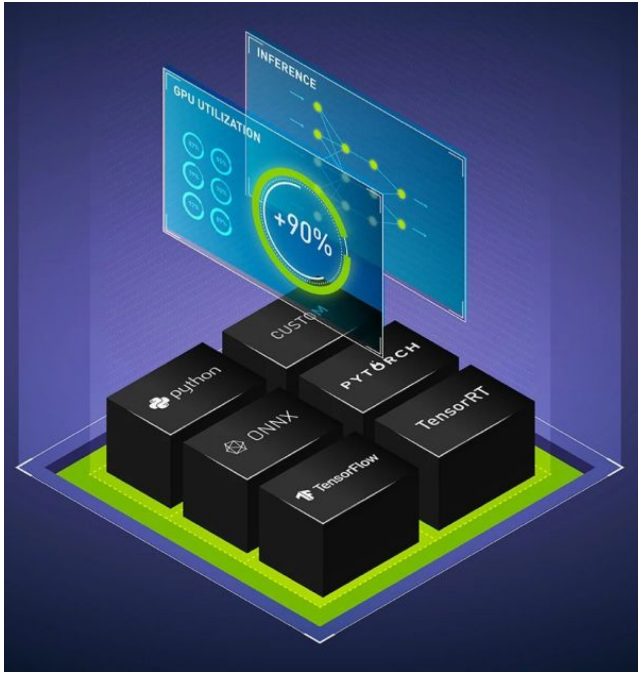
- Factorising out ML frameworks
- Allows for easier integration with workflows
- Potential for better resource utilisation and scalability



**NB: If your application just needs a single node Nvidia TensorRT might work better for you*

Nvidia Triton Inference Server

- Open-source software platform for deploying AI models for inference
- Popular model serving tool
- Supports many deep learning frameworks



Credit: Nvidia

How does it work?

- Create a model repository

```
$ #create a folder with your AI models and Triton configuration file
```

- Launch Triton

```
$ shifter --module=gpu --image=nvcr.io/nvidia/tritonserver:<xx.yy>-py3
tritonserver --model-repository=$MODEL_FOLDER
```

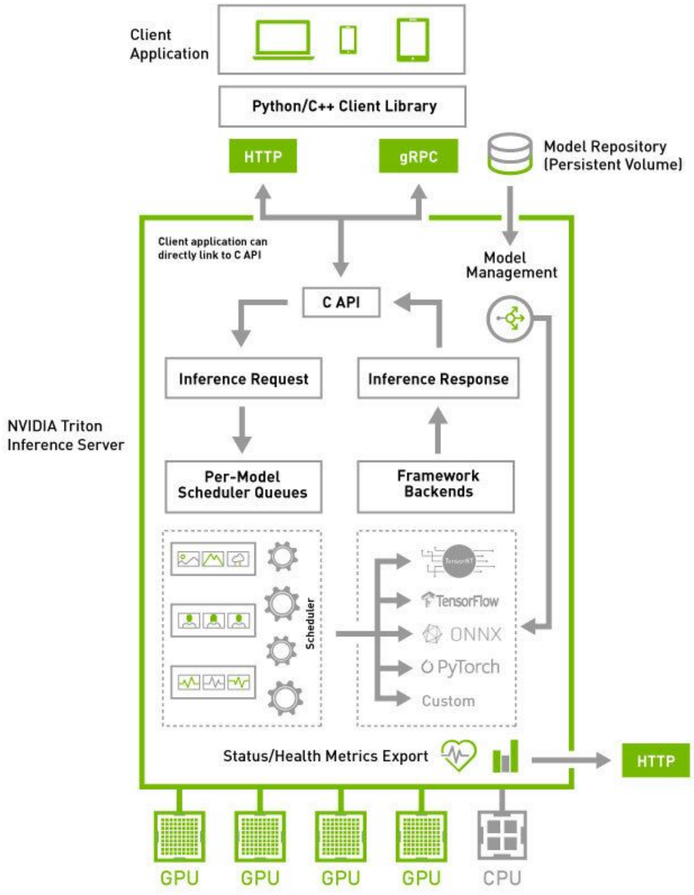
- Send inference request (via HTTP/REST or GRPC)

```
$ /workspace/bin/image_client -m densenet_onnx -c 2 -s INCEPTION -u
<server:8000> /images/mug.jpg
Request 0, batch size 1
Image '/images/mug.jpg':
    15.346230 (504) = COFFEE MUG
    13.224326 (968) = CUP
```

Triton features

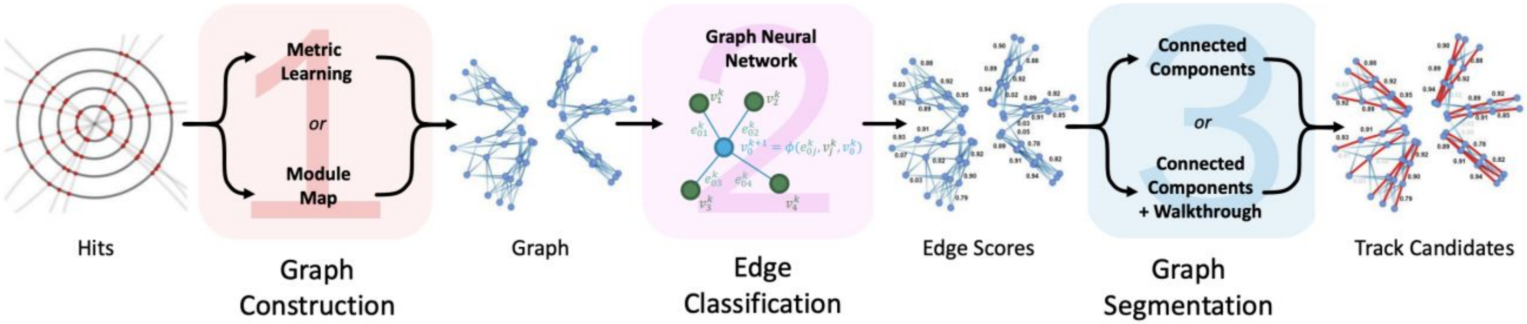
- C++/Python/Java clients*
- Can link to C API directly
- Multiple concurrent models & versions
- Custom backends and pre/post-processing operations
- Runs on CPUs & GPUs
- Dynamic batching
- Monitoring capabilities
- Model analyzer tool to optimize

**A GRPC API can be generated in a large number of programming languages*



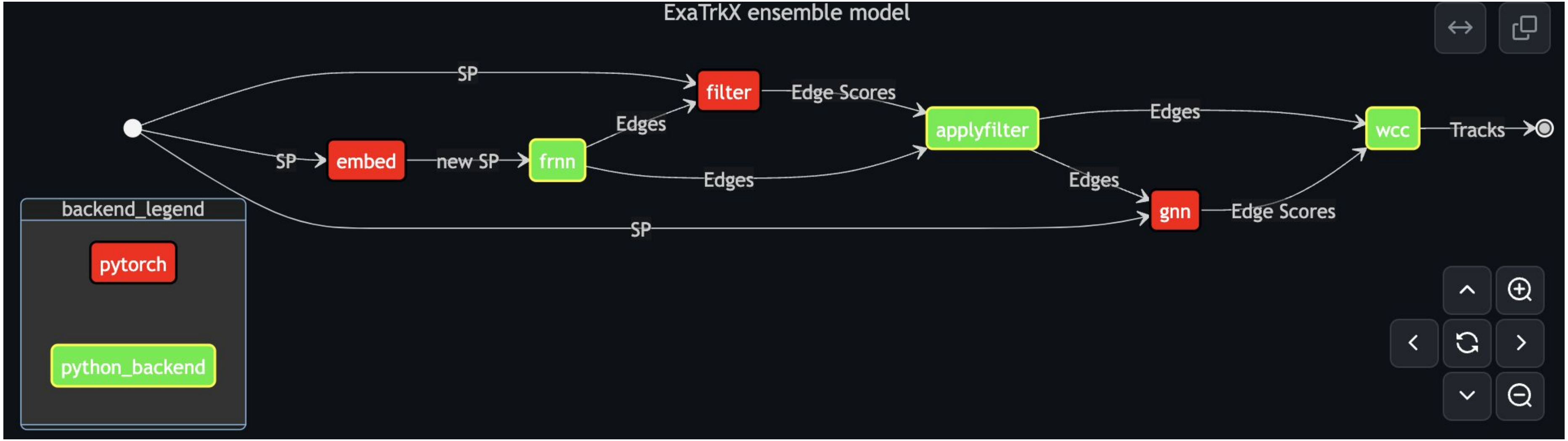
Credit: Nvidia

ExaTrkX: ML-based tracking pipeline



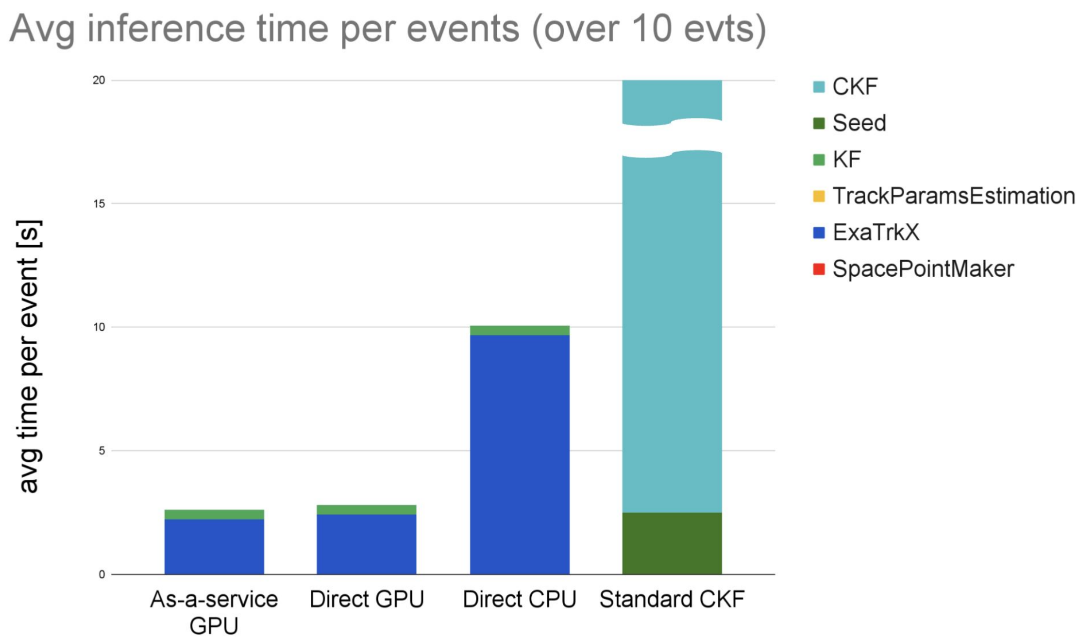
- GNN-based track finding algorithm
- Can be accelerated on different coprocessors to get faster
- ExaTrkX are exploring using NERSC as a GPU server for both local and offsite

ExaTrkX models



- This chain of models (ensemble) in Triton had poor performance
- Moved to a custom backend which executed c++ code

ExaTrkX performance



- Moving to custom backend gave significantly improved performance
- Tested on Perlmutter GPU node (using 1 A100 GPU)

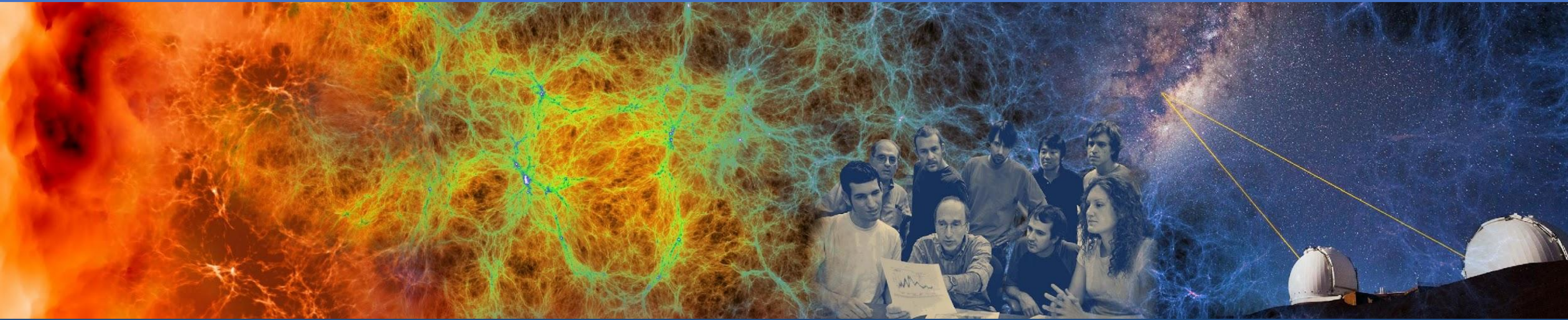
CMS

- CMS is a detector at the LHC
- Testing Triton with asynchronous requests as part of the CMS data processing pipeline
- Performed large scale tests at Purdue, Fermilab and on Google Cloud Platform
- They were able to achieved:
    - Increased throughput
    - Optimisable GPU-to-CPU ratios
    - Flexible algorithm design
- “Portable Acceleration of CMS Production Workflow with Coprocessors as a Service”

Performance

- I ran a simple Triton test on Perlmutter via gRPC
    - Resnet50 (Image classification) with PyTorch backend
    - 1 GPU node (4 x A100’s 40 GB)
    - With 3 concurrent request 649.132 infer/sec, latency 5.253 ms
- Deploy a load balancer on spin connecting from offsite to GPU server
    - With 1 concurrent request 124.481 infer/sec, latency 8.033 ms
    - Had challenges with multiple concurrent requests
    - Needs to be explored
- Nvidia MLPerf Resnet50 Triton benchmark for A100 (80 GB)
    - 1 x A100 achieved 39k infer/sec
    - 8 x A100 achieved 316k infer/sec
- Issues and performance at NERSC are not yet ironed out as early days

Demonstration



notebook.ipynb - JupyterLab

jupyter.nersc.gov/user/asnaylor/perlmutter-login-node-base/lab/tree/global/homes/a/asnaylor/projects/triton_nersc_data_day_demo/notebook.ipynb

File Edit View Run Kernel Git Tabs Settings Help

notebook.ipynb | Terminal 1

Code | git | NERSC Python

Exa.TrkX Triton Data Day Demo

Start up the Triton server

Start up a slurm job:

```
salloc --exclusive --account=nstaff -N 1 -C gpu -G 4 -q interactive -t 00:30:00
```

Run Triton container (also works with `podman-hpc`):

```
shifter --module=gpu --image=${TRITON_SERVER_IMAGE} tritonserver --model-repository=${MODEL_REPO} --log-verbose=3
```

Check server is running

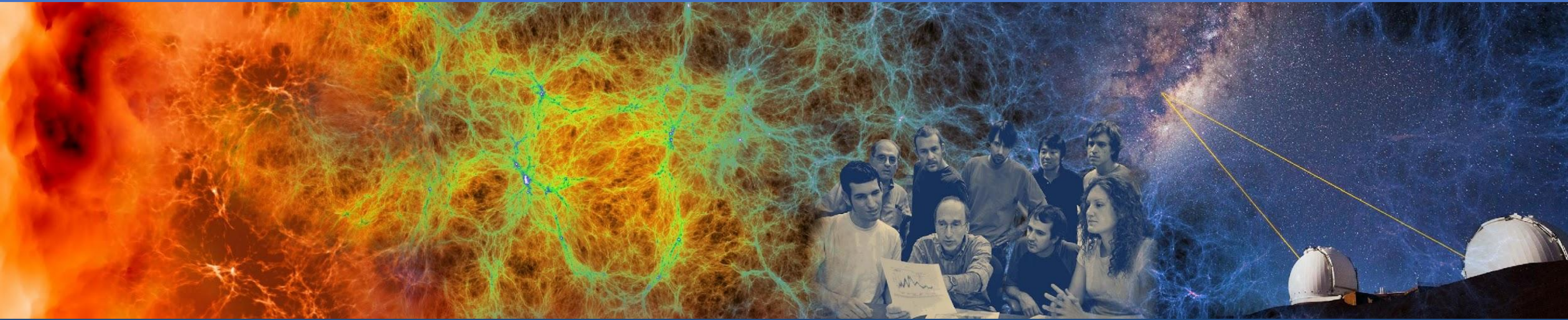
[13]:
```
# Get Tritonsever address
with open('install/.tritonserver_address', 'r') as file:
    triton_address = file.read().rstrip()
```

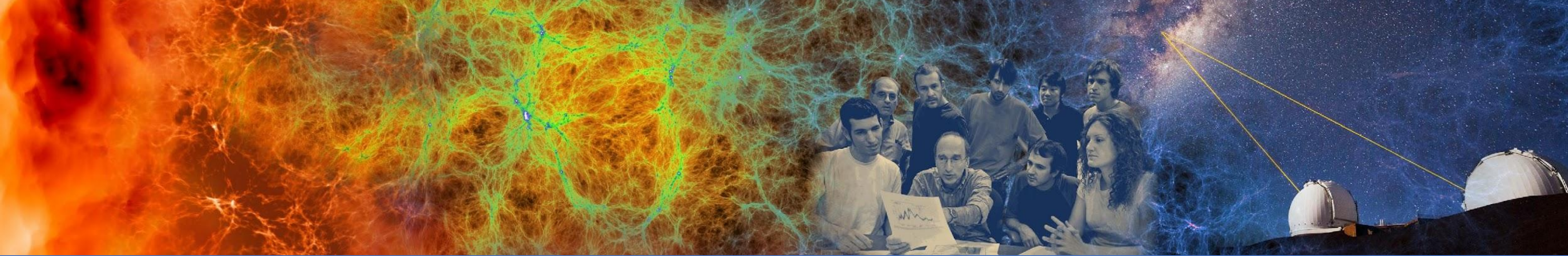
[14]:
```
!curl -v {triton_address}:8000/v2/health/ready
```

```
*   Trying 128.55.84.206:8000...
* Connected to 128.55.84.206 (128.55.84.206) port 8000 (#0)
> GET /v2/health/ready HTTP/1.1
> Host: 128.55.84.206:8000
> User-Agent: curl/8.0.1
> Accept: */*
>
```

Simple 1 1 main NERSC Python | Idle Mem: 346.00 MB Saving started Mode: Command Ln 13, Col 12 notebook.ipynb 0

Please let us know if you have a compelling use case





Thank you for listening

Any questions?