PySONIC for LHC Data Analysis

Raghav Kansal on behalf of the PySONIC team

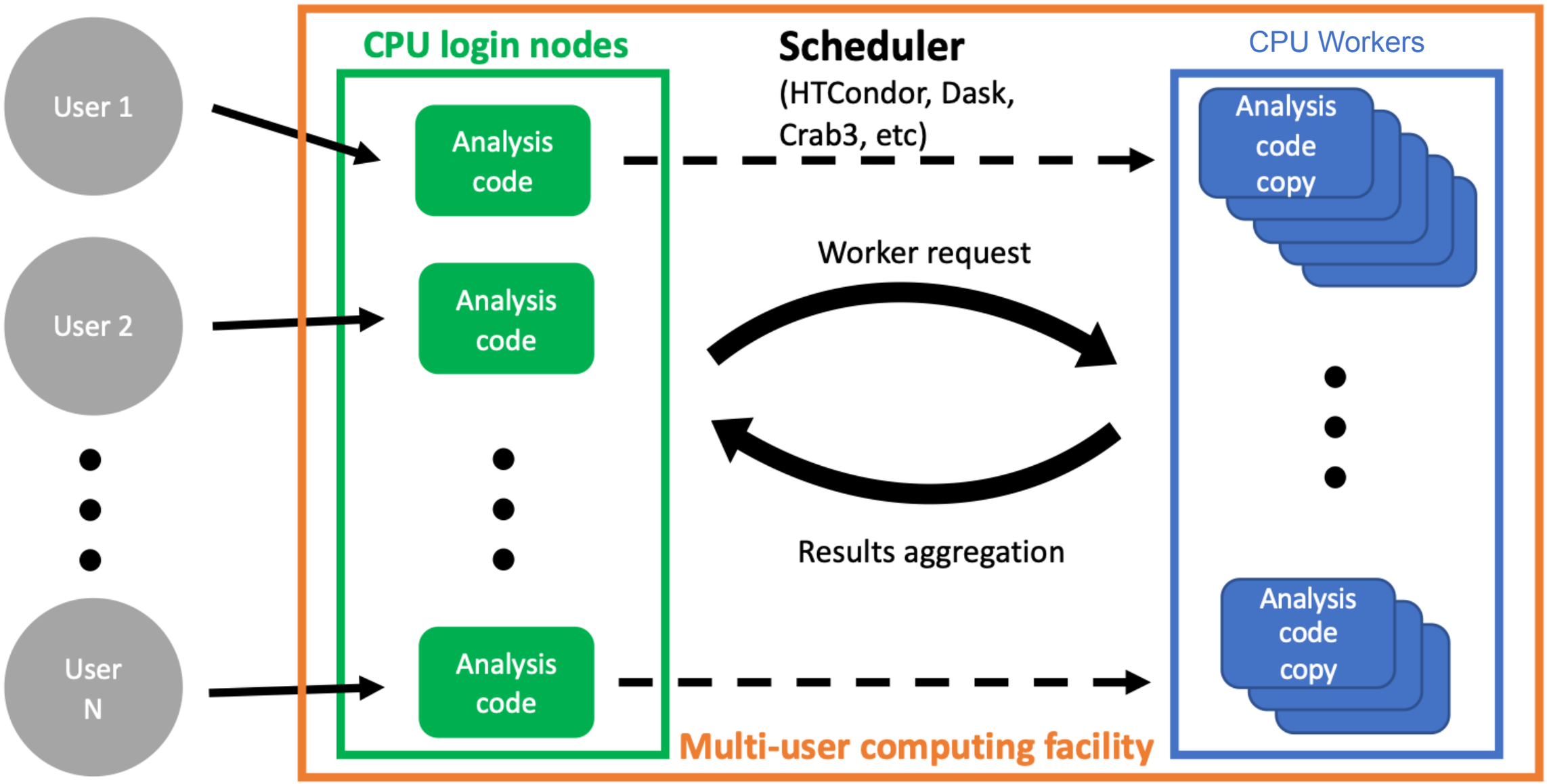
SONIC Workshop

1st March 2024

Why PySONIC?

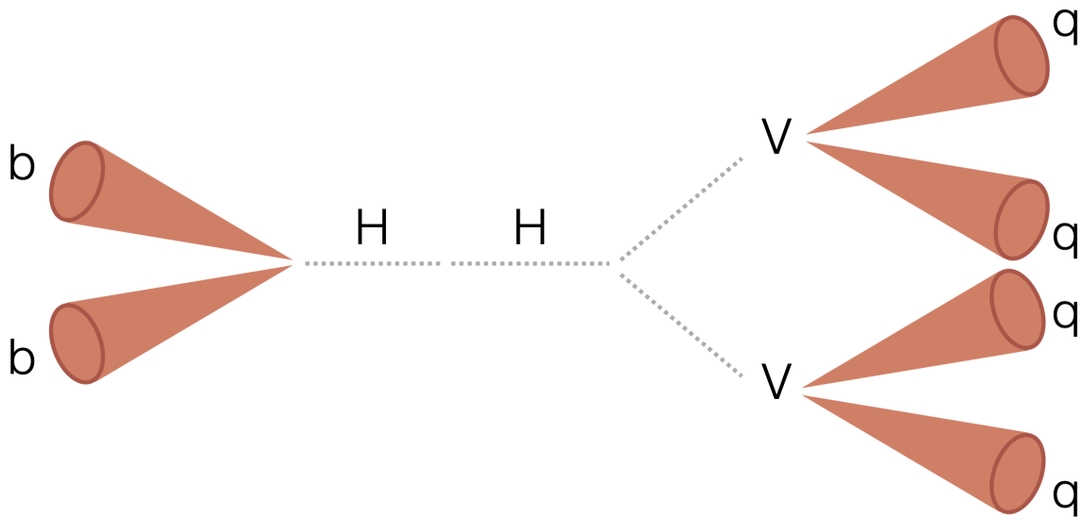
Source: C. Savard et al. arxiv:2312.06838

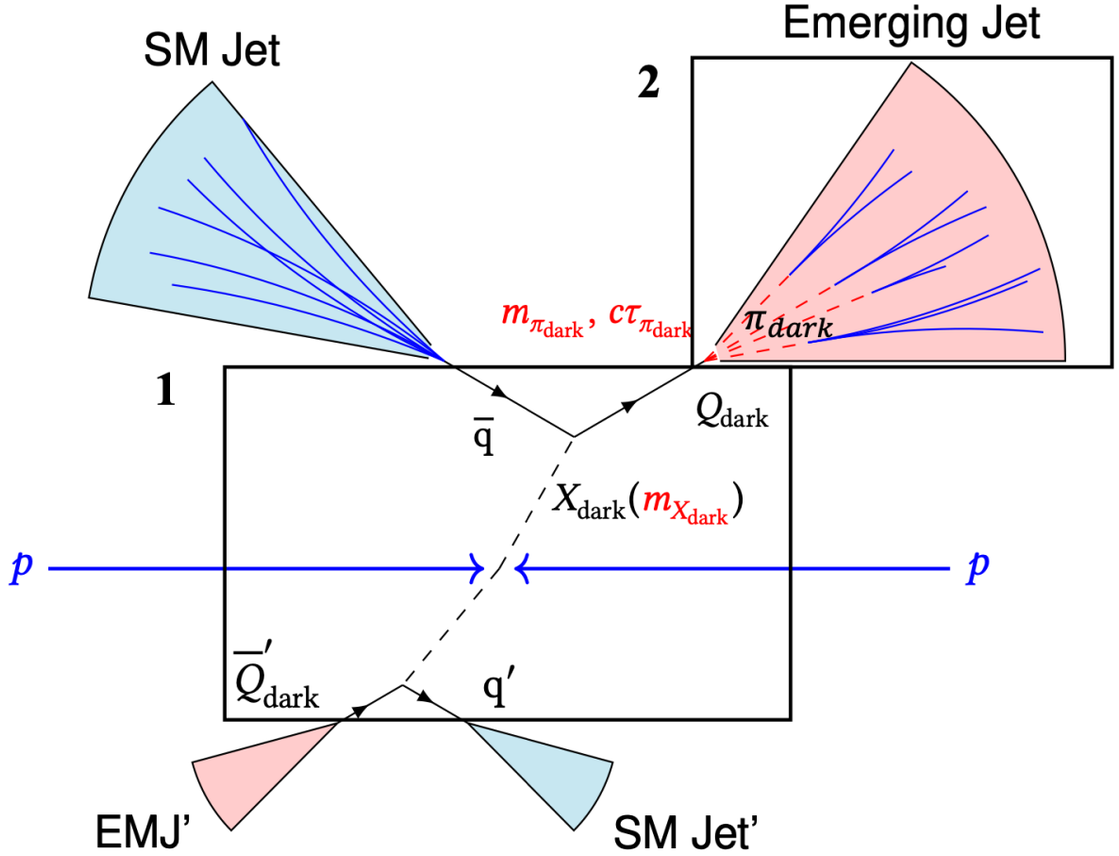
- Traditional LHC computing paradigm:
  - Parallel processing of O(1M-100M) events per user on CPUs



Problem (1/2)

- Innovative new analyses use sophisticated, complex deep learning models e.g.
  - Deep transformers for measuring double-Higgs production (CMS-HIG-23-012, *in prep*)
  - Graph neural networks for searches for dark matter (CMS-EXO-22-015)
  - … and many more!

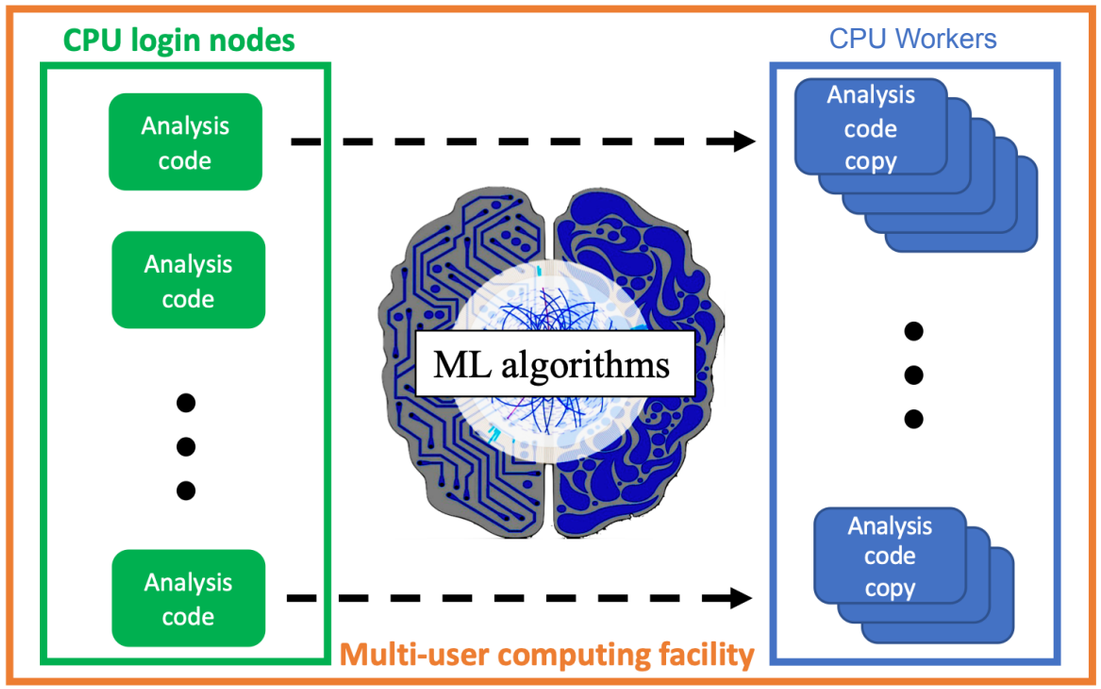




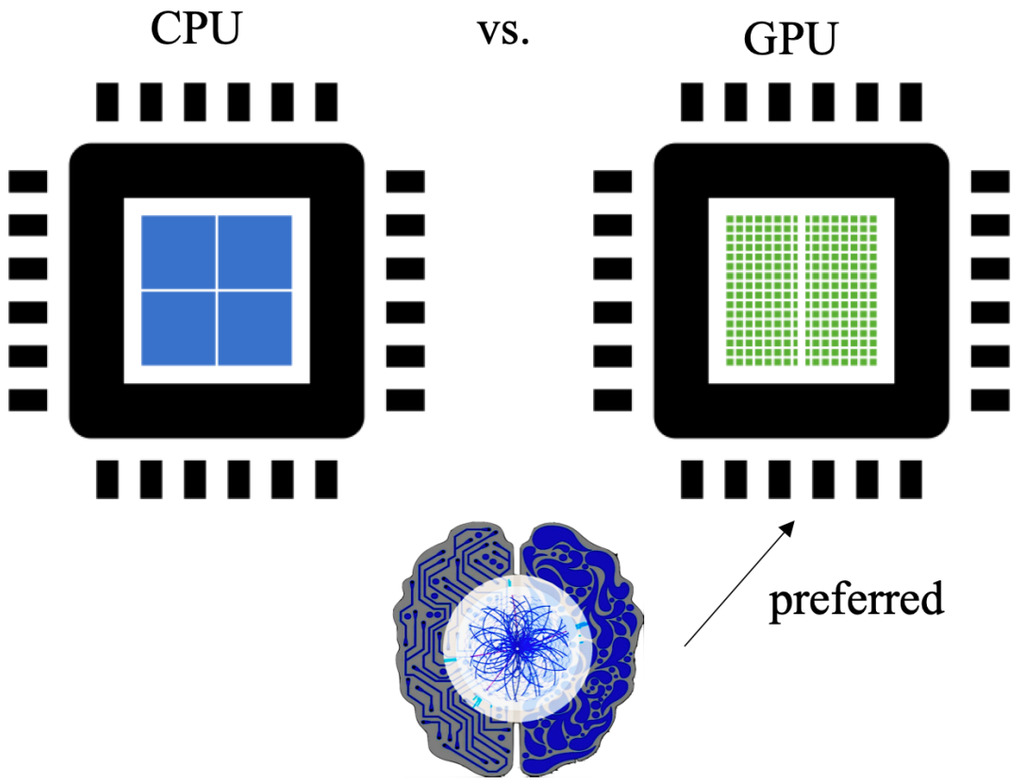
Problem (2/2)

Source: C. Savard et al. Fast ML

- Such models are highly CPU memory- and time-intensive
- Computationally intractable to run inference on CPU workers

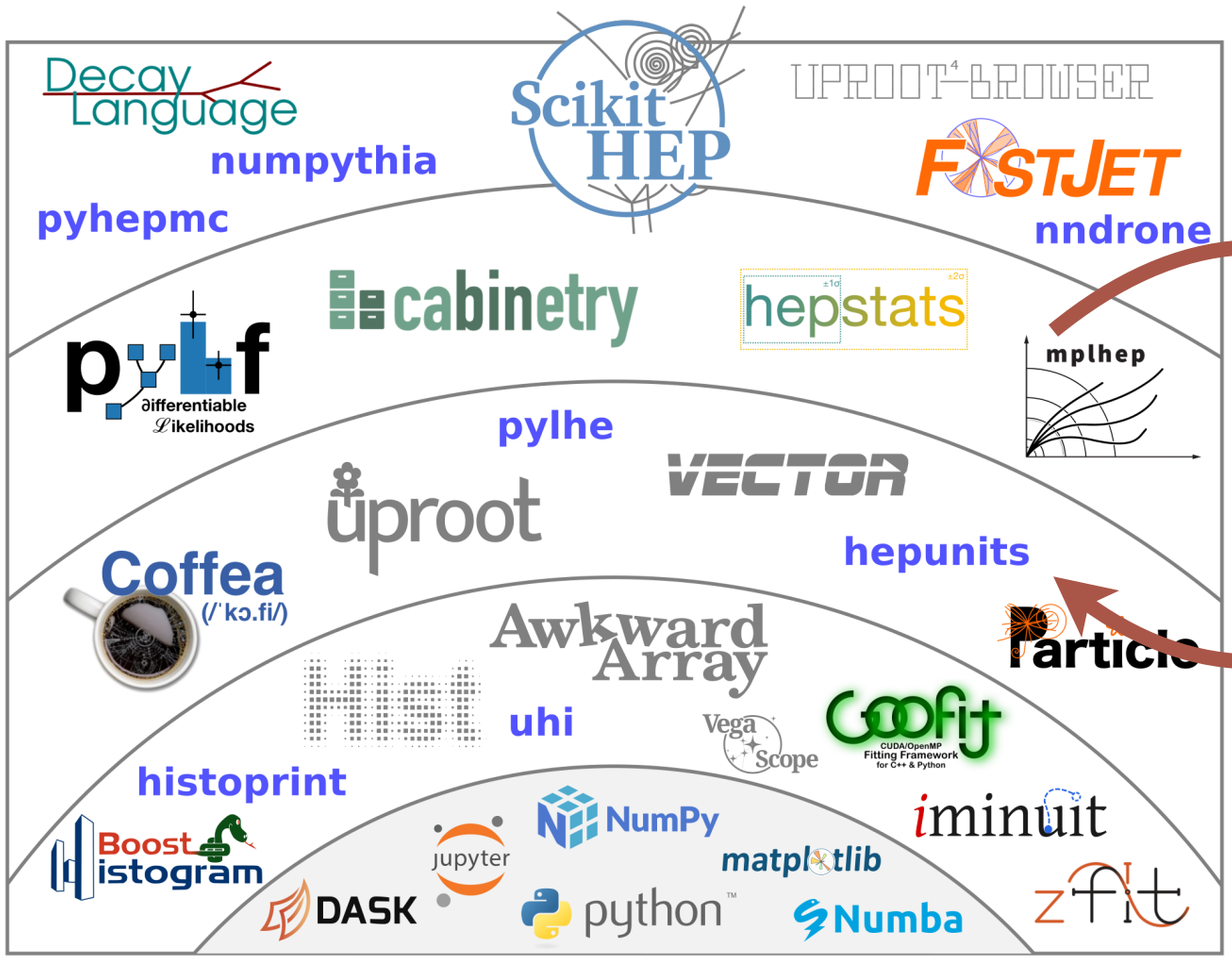


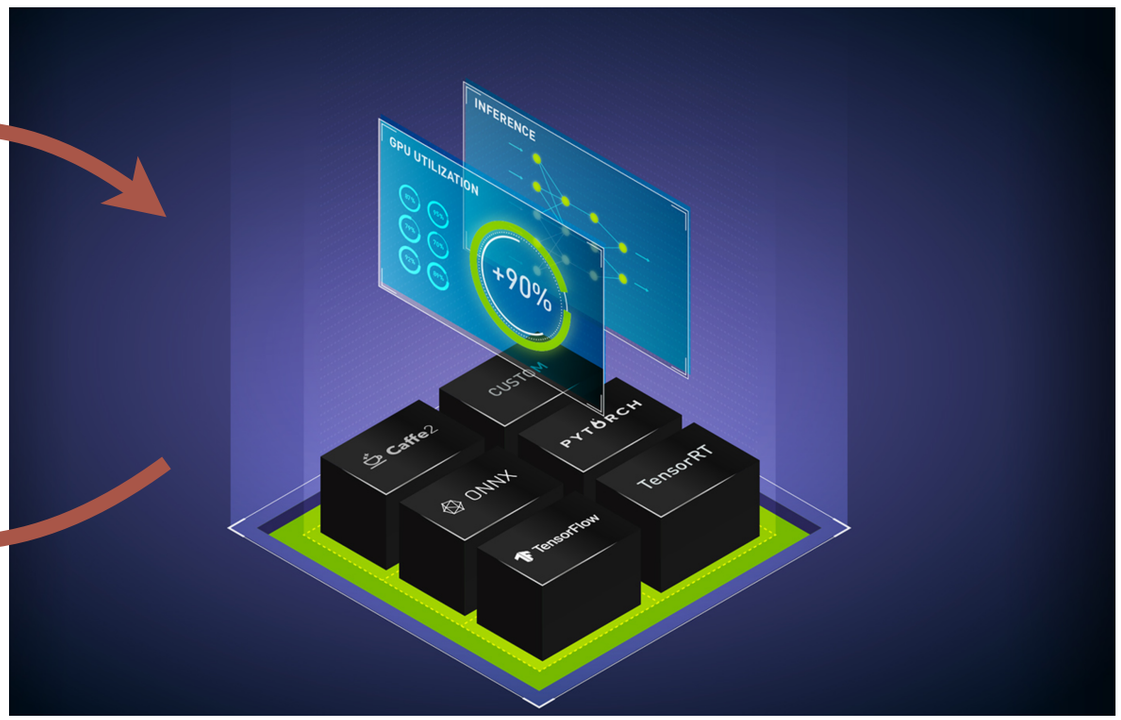
- Can we use GPUs?
  - Ideal for parallelized ML inference
  - CPUs for file I/O, transfer, storage, general workflow



What is PySONIC?

- Integrating modern Python (CPU) based analysis tools…
  - Numpy, Pandas, Scikit-HEP, Awkward Array, Uproot, Coffea etc.

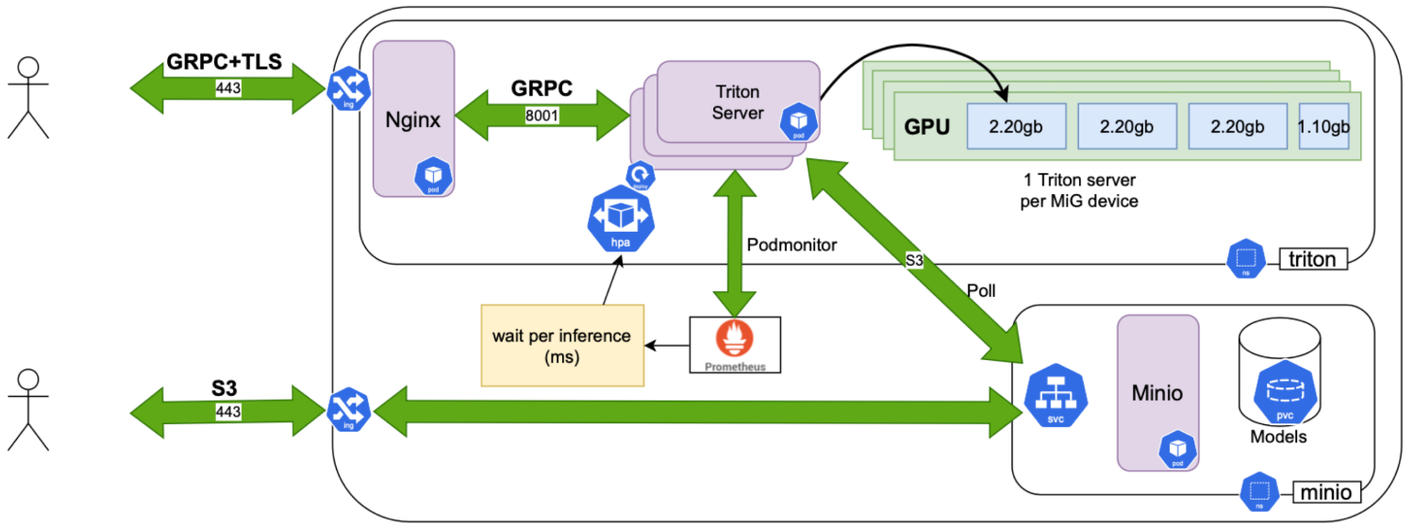




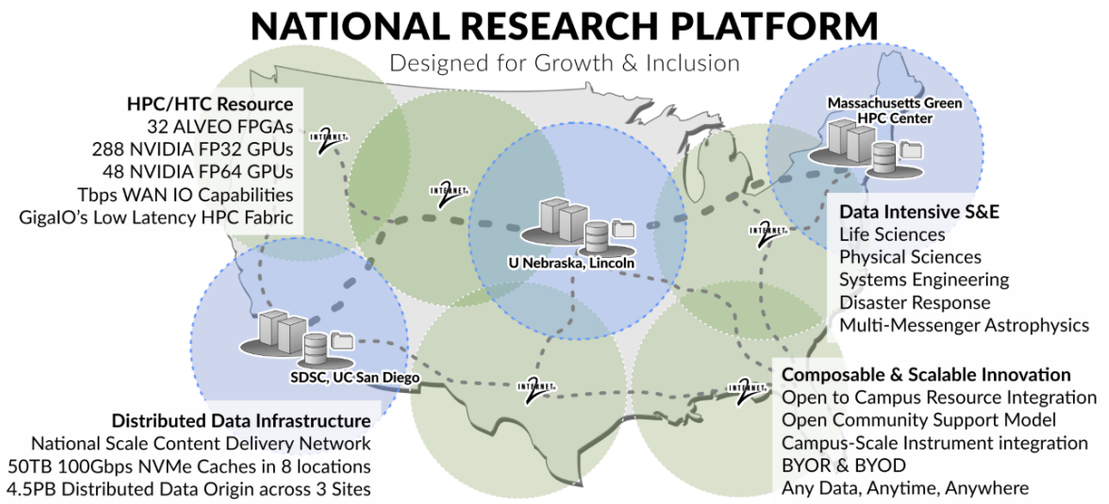
- …with the Triton inference server for real-time GPU inference-as-a-service

(Some) Current Facilities

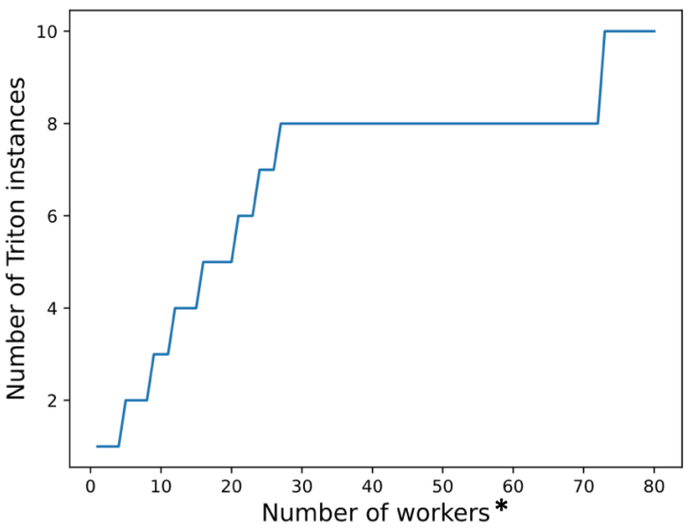
- Fermilab Elastic Analysis Facility (EAF)

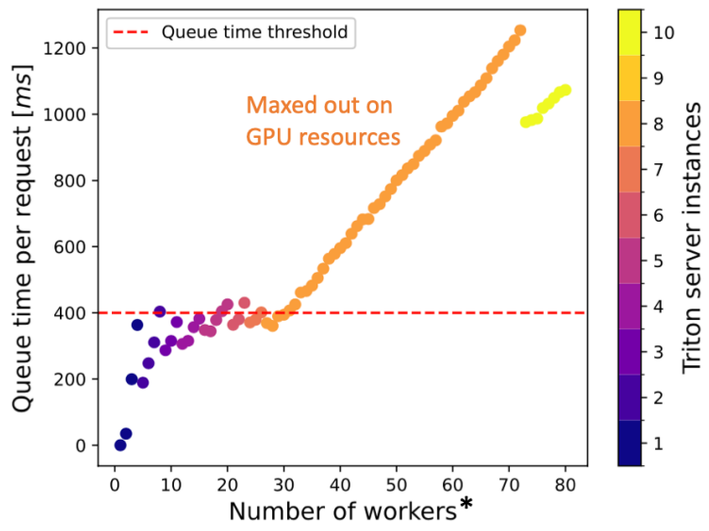


- National Research Platform (NRP) / San Diego Supercomputer Center



- Auto-scaling of server instances





C. Savard et al. arxiv:2312.06838

- Kubernetes deployments + load balancing to scale to 1000s of workers

How to use PySONIC?

1. Trained model + config files

Triton Server
MinIO Storage @ EAF
Ceph Persistent Volume @ NRP

2. Link with analysis code

```
## ANALYSIS CODE
# link to triton model
triton =
    wrapped_triton(<path
    to model>)

physics stuff…

# format data
X = …
# inference request
y = triton(X)

more physics stuff…
```

class **coffea.ml_tools.triton_wrapper**(*model_url: str, client_args: Dict | None = None, batch_size=-1*)
[source]

Bases: `nonserializable_attribute` , `numpy_call_wrapper`

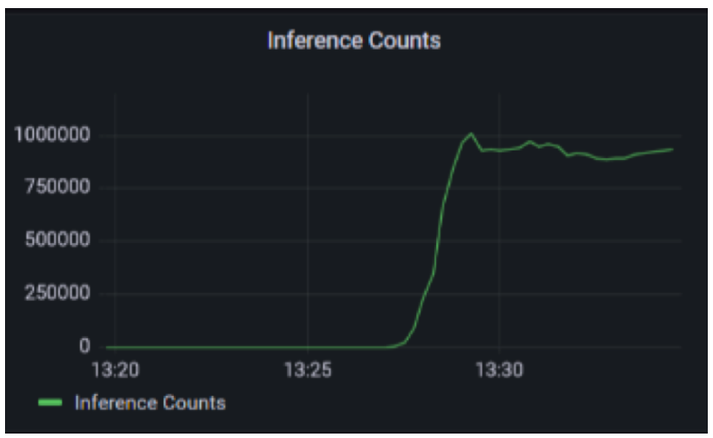
Wrapper for running triton inference.

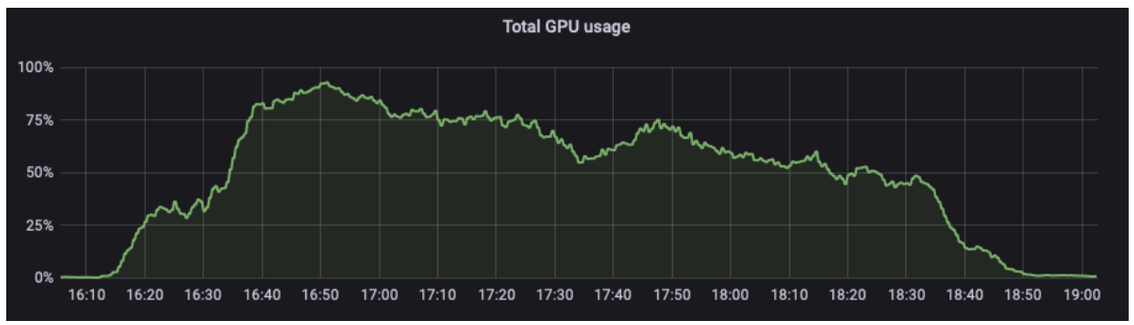
The target of this class is such that all triton specific operations are wrapped and abstracted-away from the users. The users should then only needs to handle awkward-level operations to mangle the arrays into the expected input format required by the the model of interest.

```
from coffea.ml_tools.triton_wrapper import triton_wrapper

# Running the evaluation in lazy and non-lazy forms
tw = triton_wrapper(model_url="triton+grpc://127.0.0.1:8001/pn_test/1")
output = tw(["output"], jets)
```

3. Run and monitor

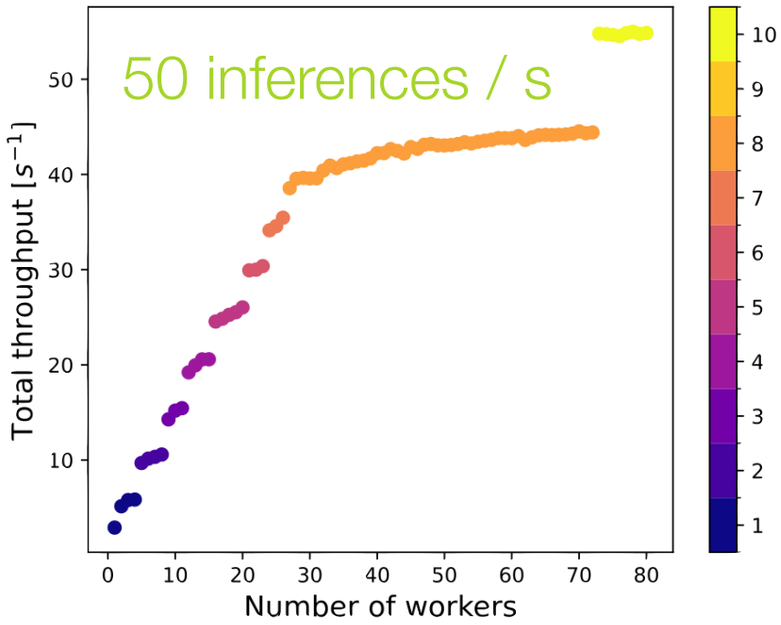




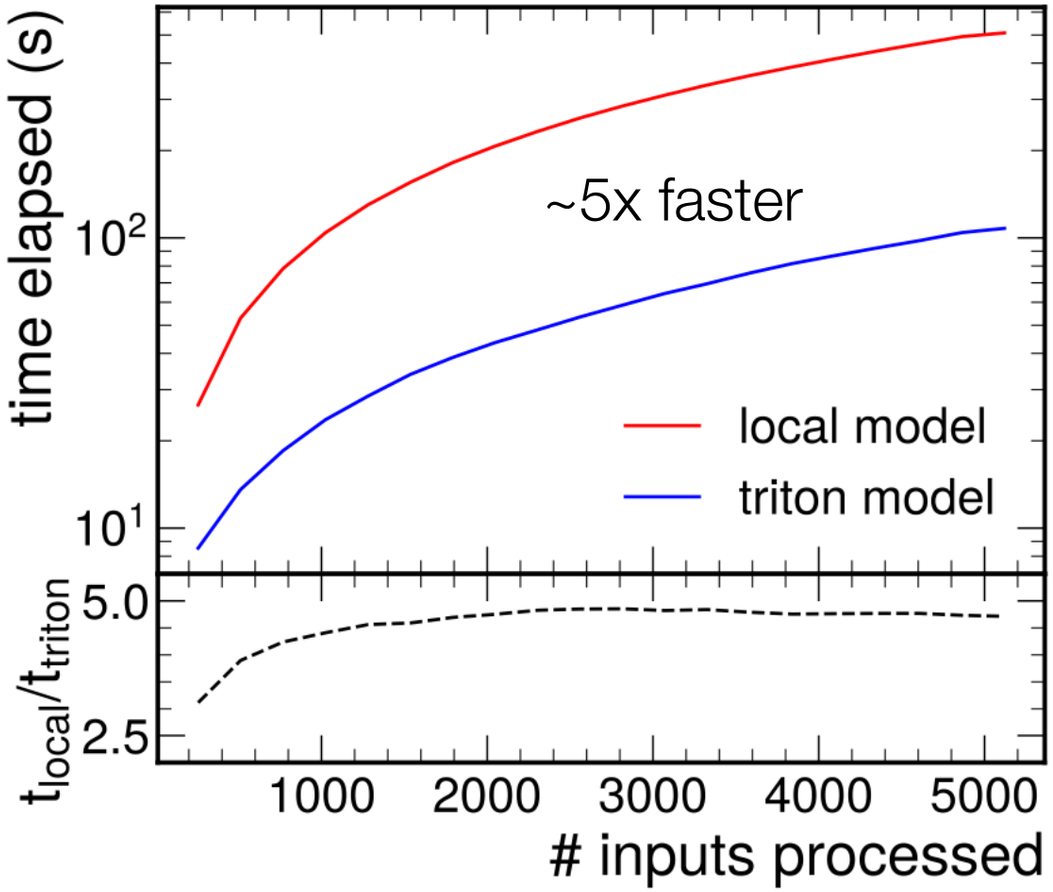
(Some) Results (1/2)

Source: C. Savard et al. arxiv:2312.06838

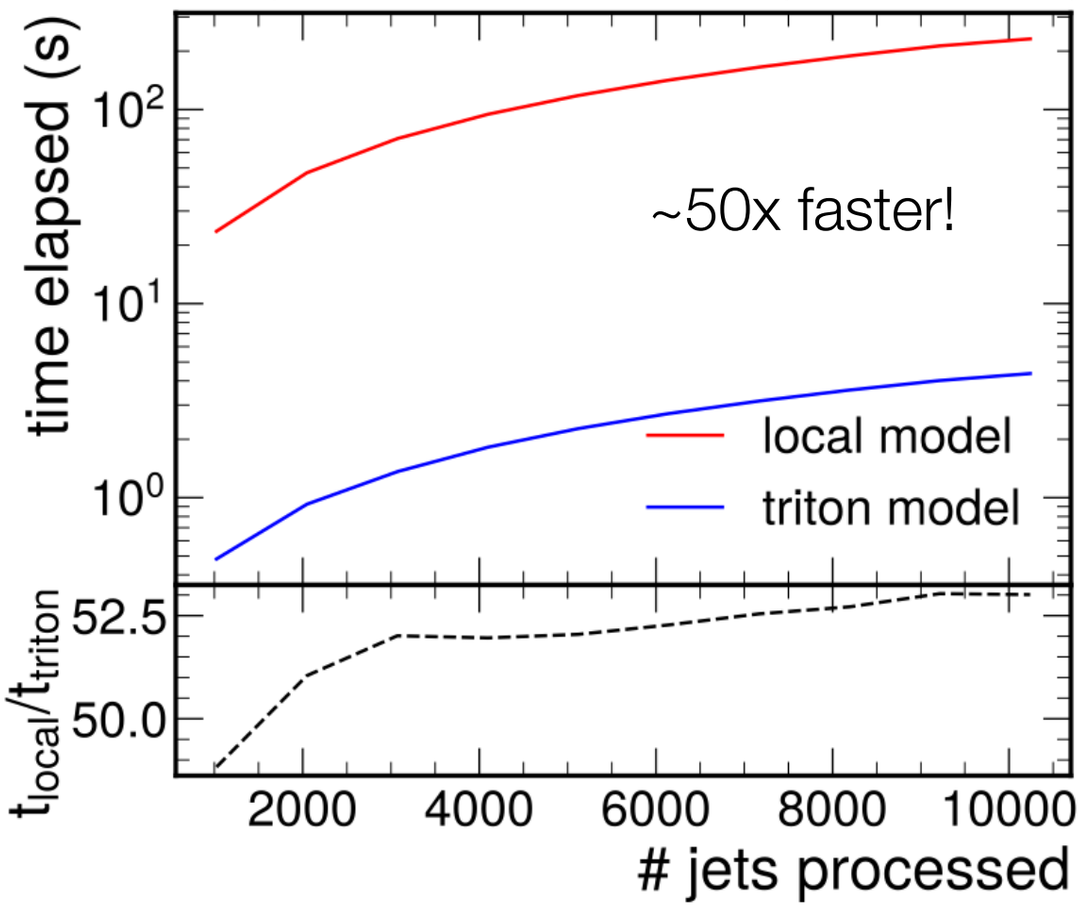
- Benchmarking on EAF:



ResNet50

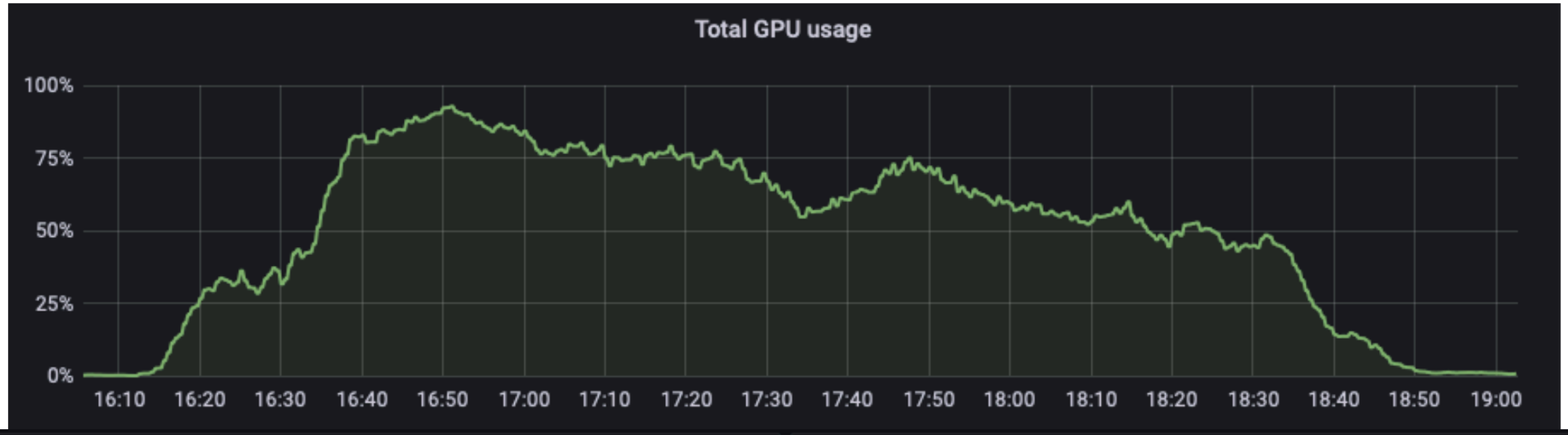


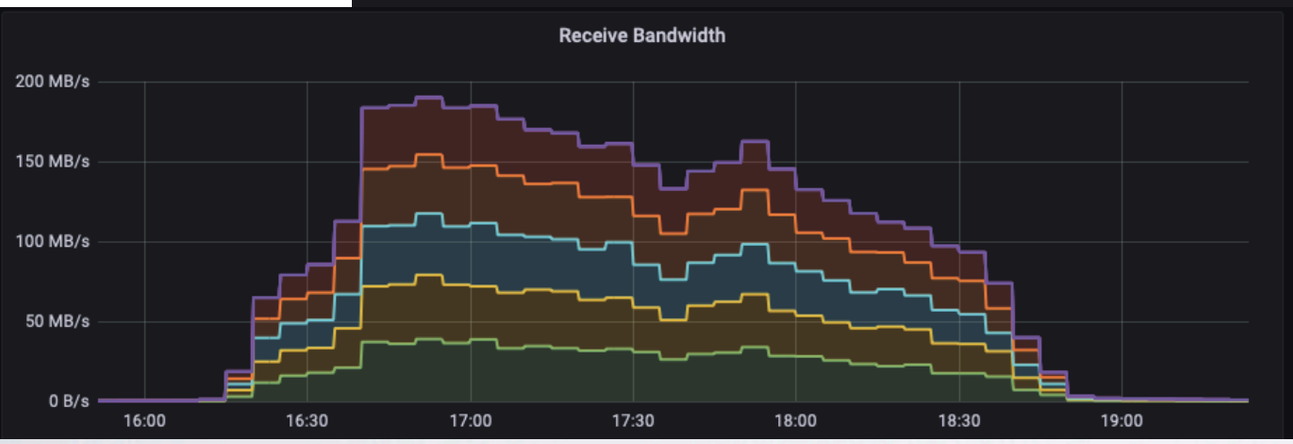
Graph Neural Network (ParticleNet)

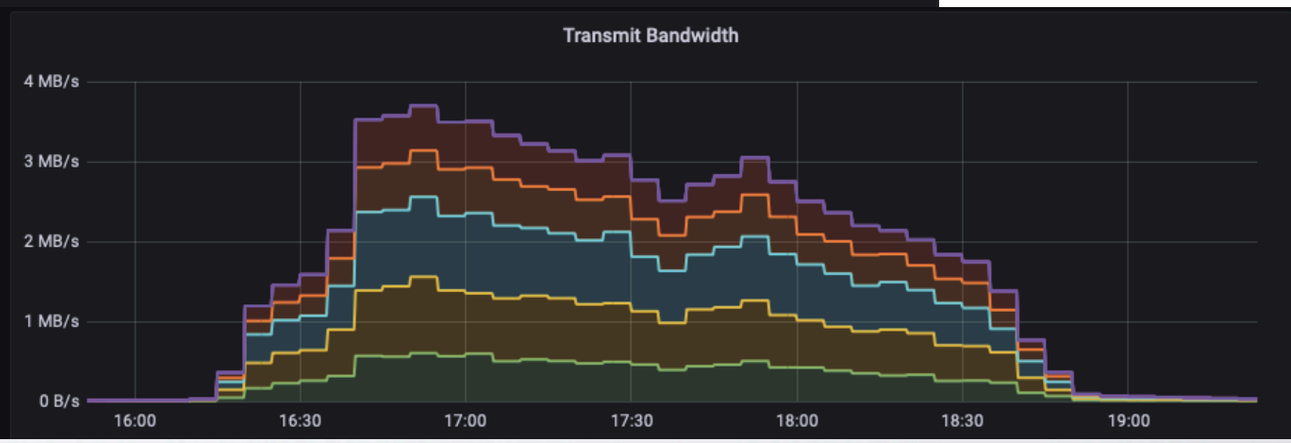


(Some) Results (2/2)

- Example inference run with NRP (10 GPUs)
  - ParticleTransformer classifier for double-Higgs measurement
  - Run over data collected by the CMS experiment in 2018
  - 80M events / 1.3 TB data transmitted / 1000 parallel jobs in 3 hours
  - In comparison: >1 week on CPUs-alone







Conclusion

- Using deep learning to push the boundaries of discoveries at the LHC
- Sophisticated DL methods call for new computing paradigms
- PySONIC leverages modern Python-based data analysis tools + Triton inference server to perform analyses with CPUs ↔ GPUs
- Initial exploration extremely promising, already seeing significant impact
- Future work:
  - Tutorials and training for users (e.g. for EAF)
  - Server/hardware-side: auto-scaling on more facilities, efficient multi-model orchestration, different coprocessors