ATLAS Data Analysis Using a Parallel Workflow on Distributed Cloud-Based Services with GPUs

Jay Sandesara, Rafael Coelho Lopes de Sa, Verena Martinez Outschoorn,

Fernando Barreiro Megino, Johannes Elmsheuser, Alexei Klimentov

on behalf of the ATLAS Computing Activity

Simulation-Based Inference (SBI)

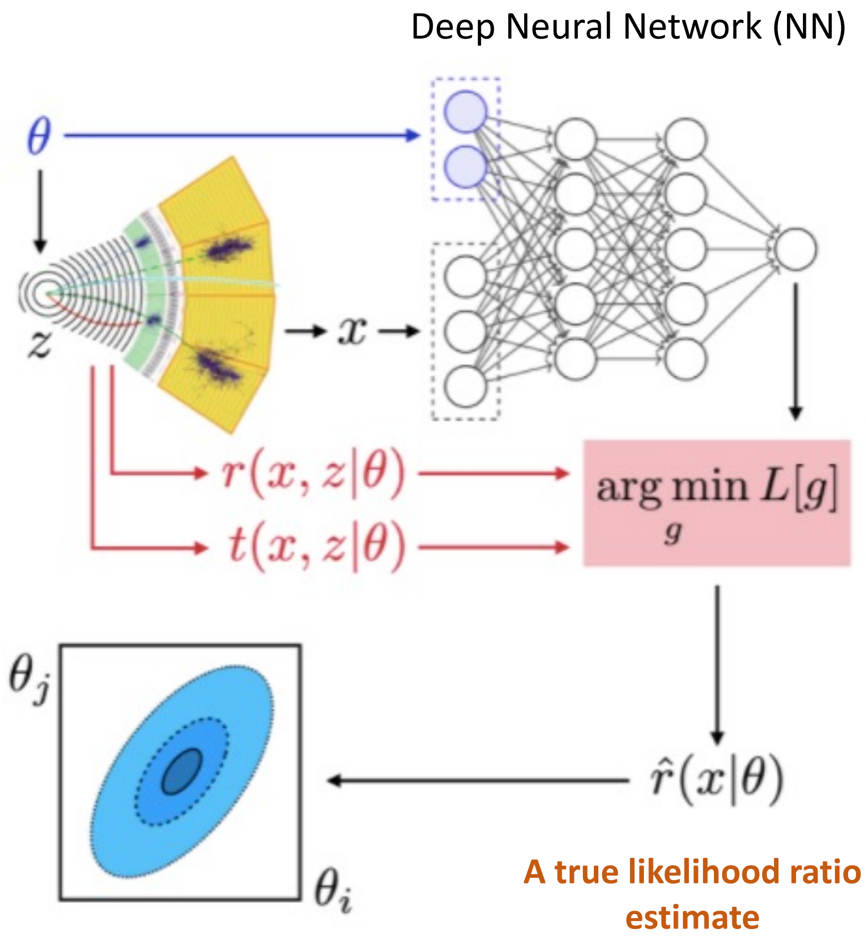
- For HEP experiments, computing exact likelihoods or likelihood ratios **analytically** for an observed event is un-feasible.

Readily available!

$p(x|\theta)$ — Simulation → $x \sim p(x|\theta)$

← Inference

Difficult!



- Simulation-Based Inference refers to a set of Deep Learning techniques used to infer the ***true*** likelihood or likelihood ratio using simulations!

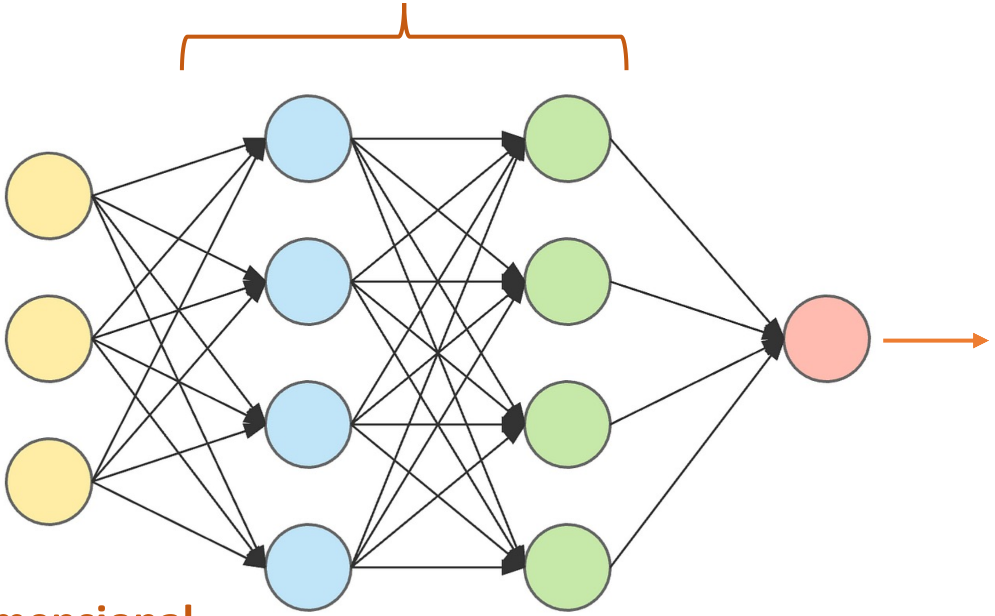
- **Practically, an analysis like this requires large-scale and powerful computing resources.**

Application of the SBI Analysis using ATLAS

- The NNs needed for a well-calibrated, un-biased and low-variance estimate of the likelihood ratio using real experimental data requires an ensemble of very deep and wide NNs.

7 Hidden layers of 1.5k neurons

13.5 Million (!) trainable parameters



24 dimensional input vector

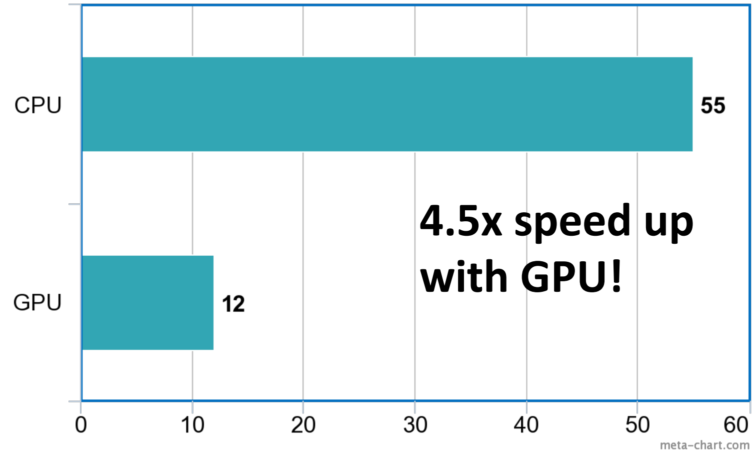
Likelihood Ratio Estimates

Single NN in an ensemble of thousands more

Application of the SBI Analysis using ATLAS

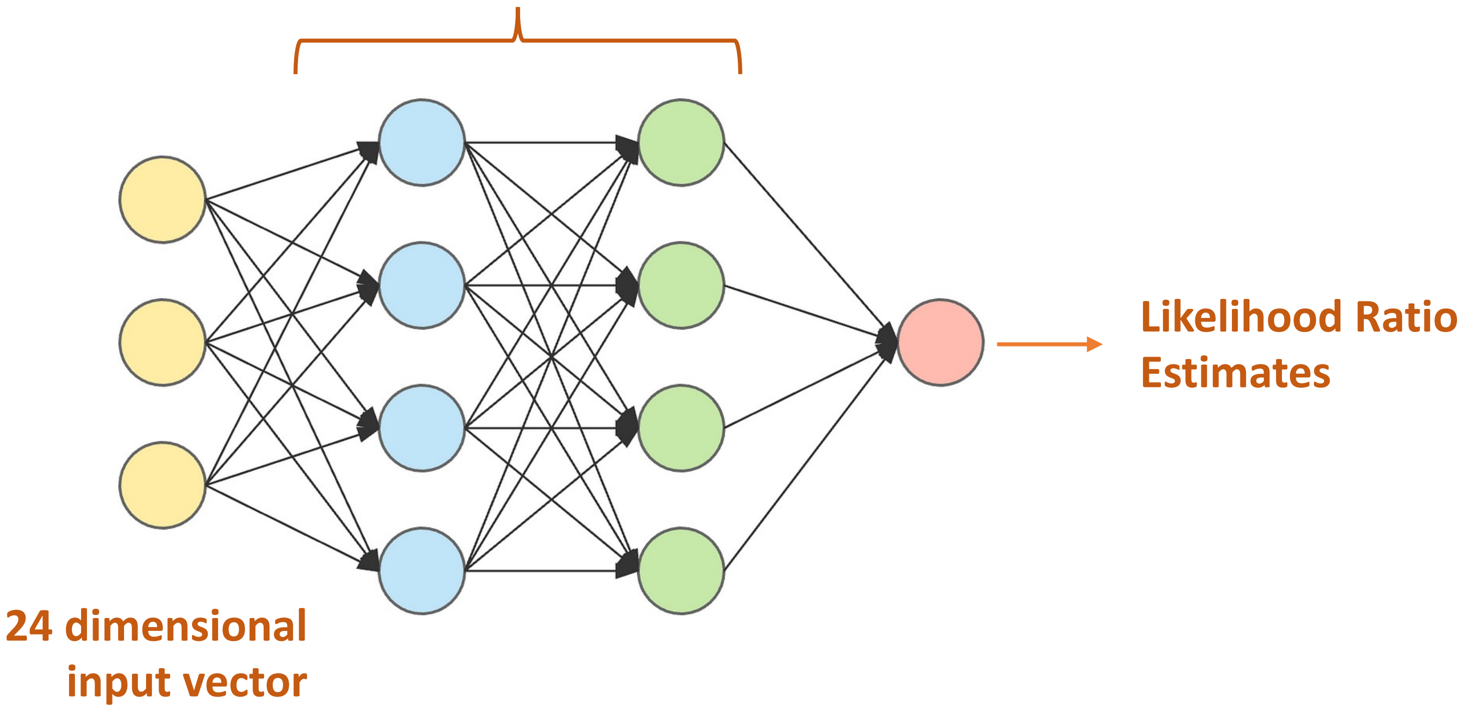
- The NNs needed for a well-calibrated, un-biased and low-variance estimate of the likelihood ratio using real experimental data requires an ensemble of very deep and wide NNs.

Training time for single NN (in hours)



7 Hidden layers of 1.5k neurons

13.5 Million (!) trainable parameters

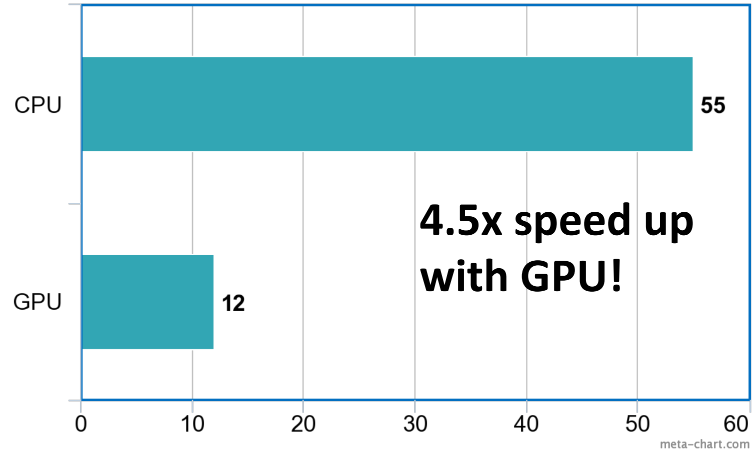


Single NN in an ensemble of thousands more

Application of the SBI Analysis using ATLAS

- The NNs needed for a well-calibrated, un-biased and low-variance estimate of the likelihood ratio using real experimental data requires an ensemble of very deep and wide NNs.

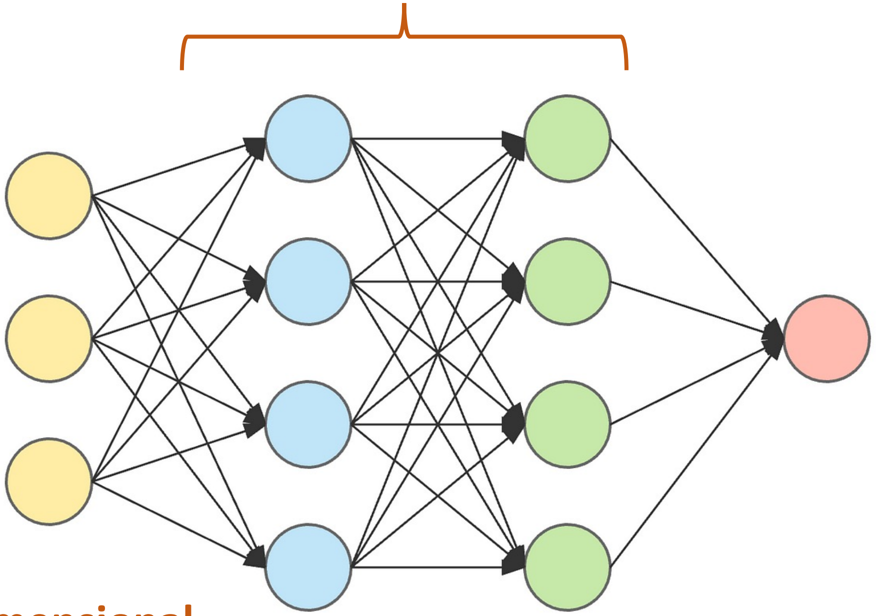
Training time for single NN (in hours)



Large scale GPU infrastructure is essential!

7 Hidden layers of 1.5k neurons

13.5 Million (!) trainable parameters



24 dimensional input vector

Likelihood Ratio Estimates

Single NN in an ensemble of thousands more

Why Use Cloud-Based Services?

- Access to modern and powerful CPU and GPU infrastructure on-demand.
- Possible to scale out large deployments as per analysis requirements, for required periods of time.
- Integrated with the available distributed computing framework (PanDA and Rucio in ATLAS) – make use of existing software tools alongside powerful new infrastructure!

PanDA

RUCIO

×

Google Cloud

An extensive range of powerful computing infrastructure available on-demand!

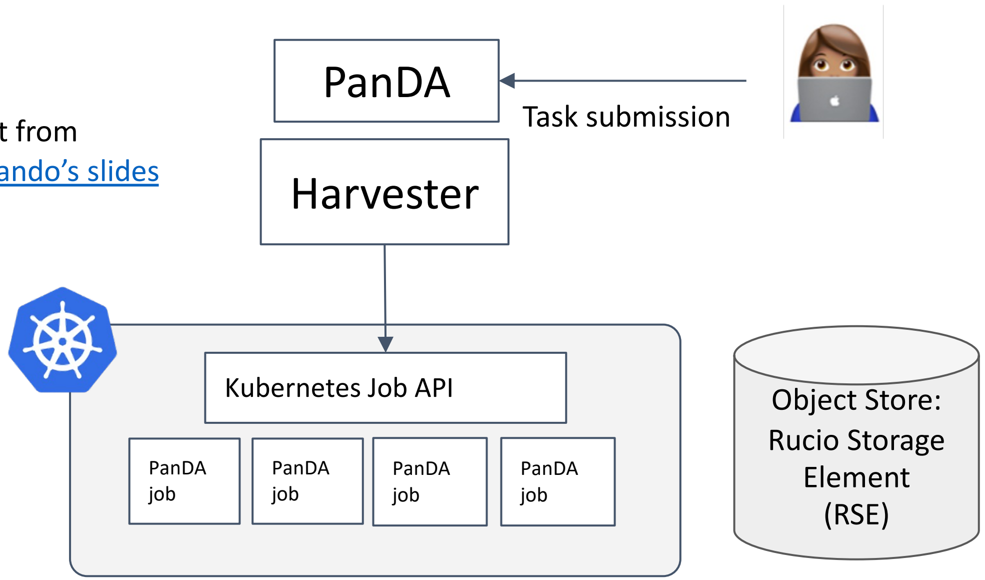
Technical Implementation:

[F. Megino, Accelerating science: the usage of commercial clouds in ATLAS Distributed Computing]

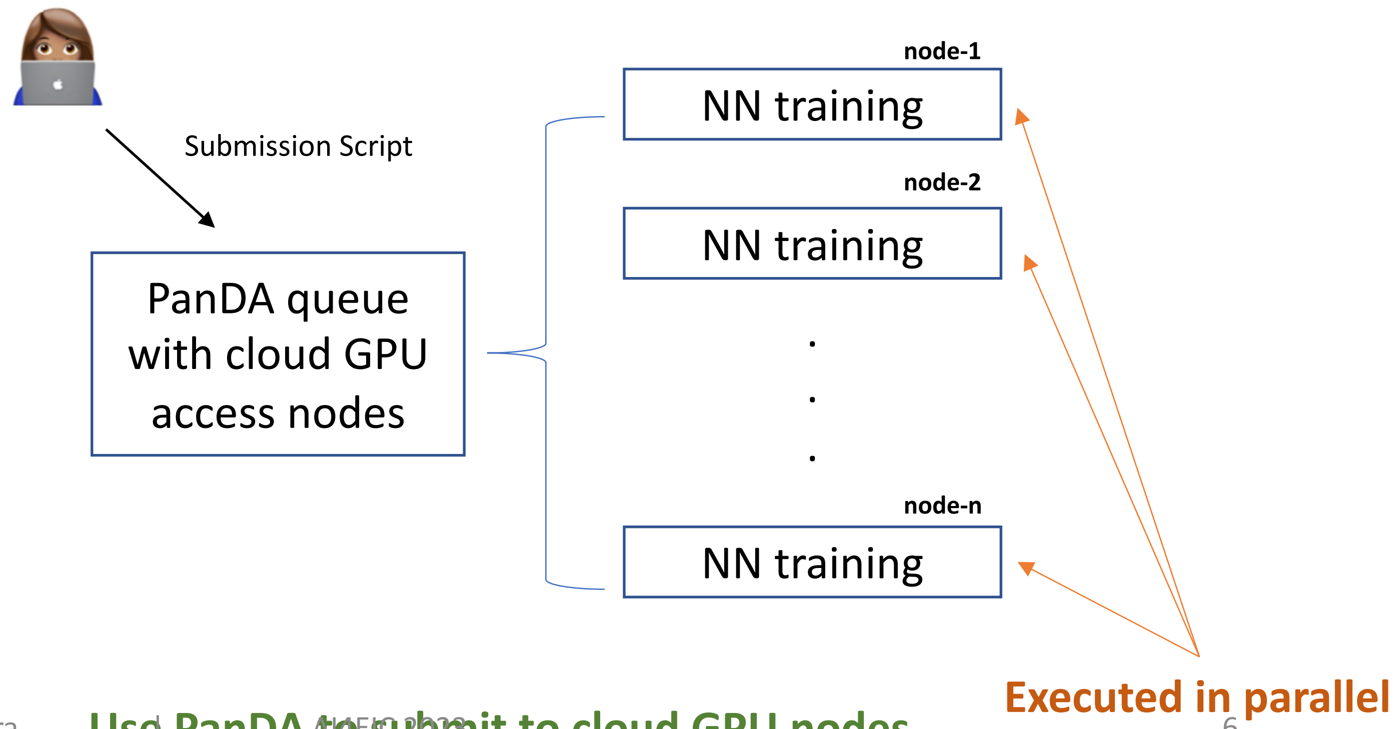
Basic Workflow

- One can submit many simultaneous NN training jobs to individual PanDA nodes that have access to cloud GPU resources.
- The NNs are then trained in parallel, one per node, and the results can be analyzed using another diagnostic script.

Chart from [Fernando's slides]

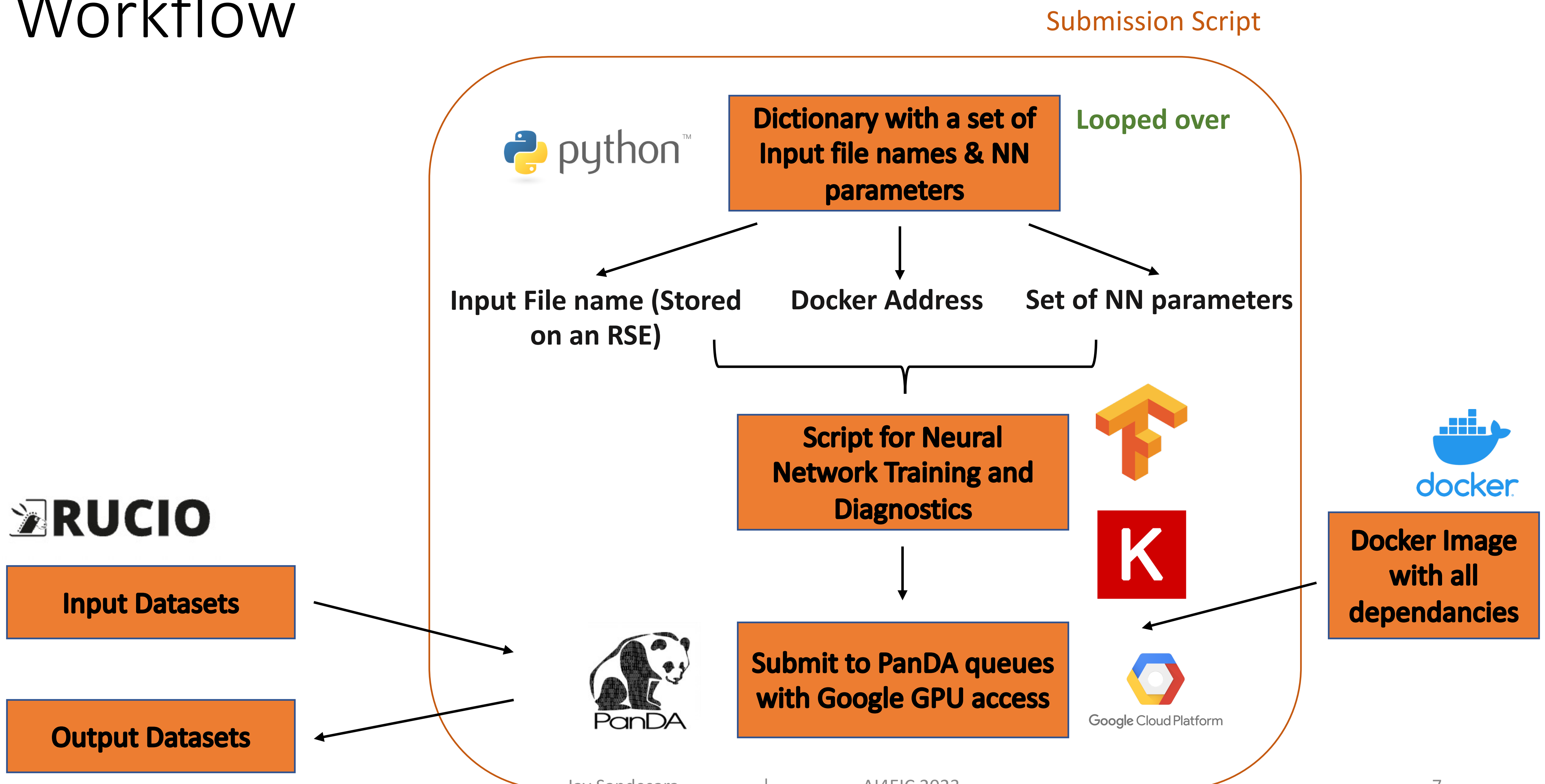


General PanDA submission overview



Use PanDA to submit to cloud GPU nodes

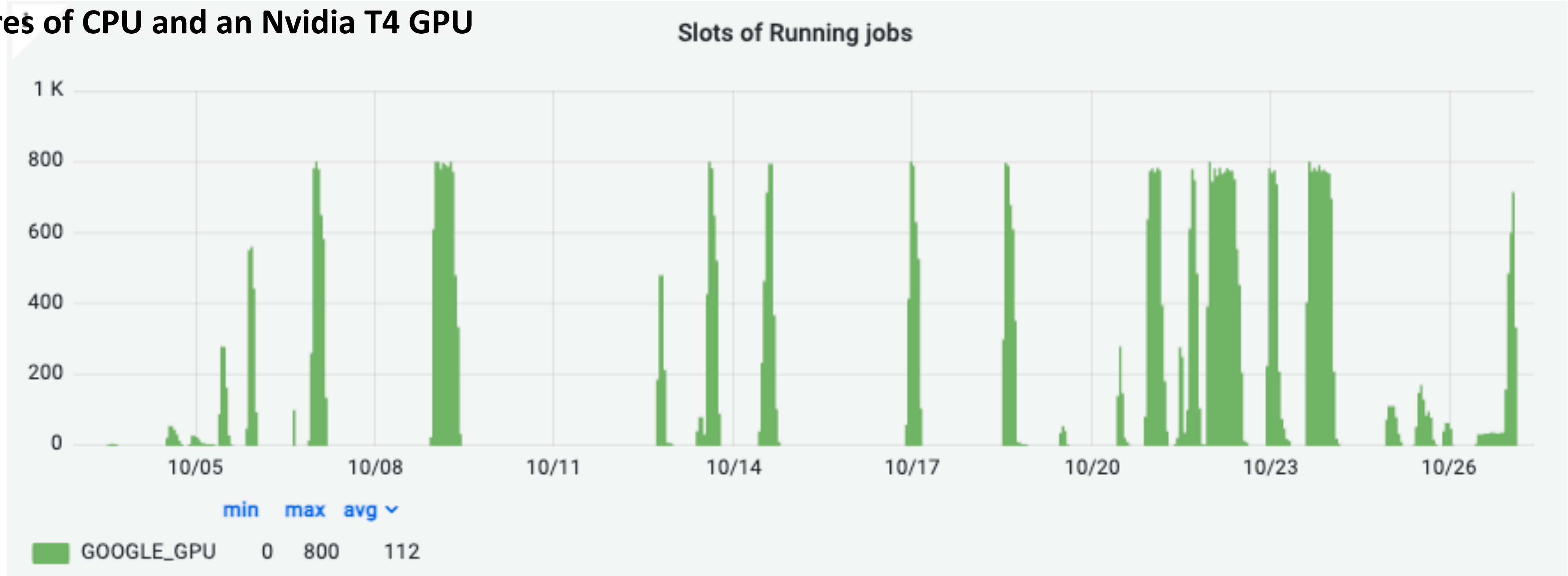
Workflow


Submission Script
python
Dictionary with a set of Input file names & NN parameters
Looped over
Input File name (Stored on an RSE)
Docker Address
Set of NN parameters
Script for Neural Network Training and Diagnostics
RUCIO
Input Datasets
Output Datasets
docker
Docker Image with all dependancies
PanDA
Submit to PanDA queues with Google GPU access
Google Cloud Platform

Graph of Real Usage for the SBI analysis

Total of 200 nodes were made available for the analysis, each with four cores of CPU and an Nvidia T4 GPU

Number of CPU cores in use at given time

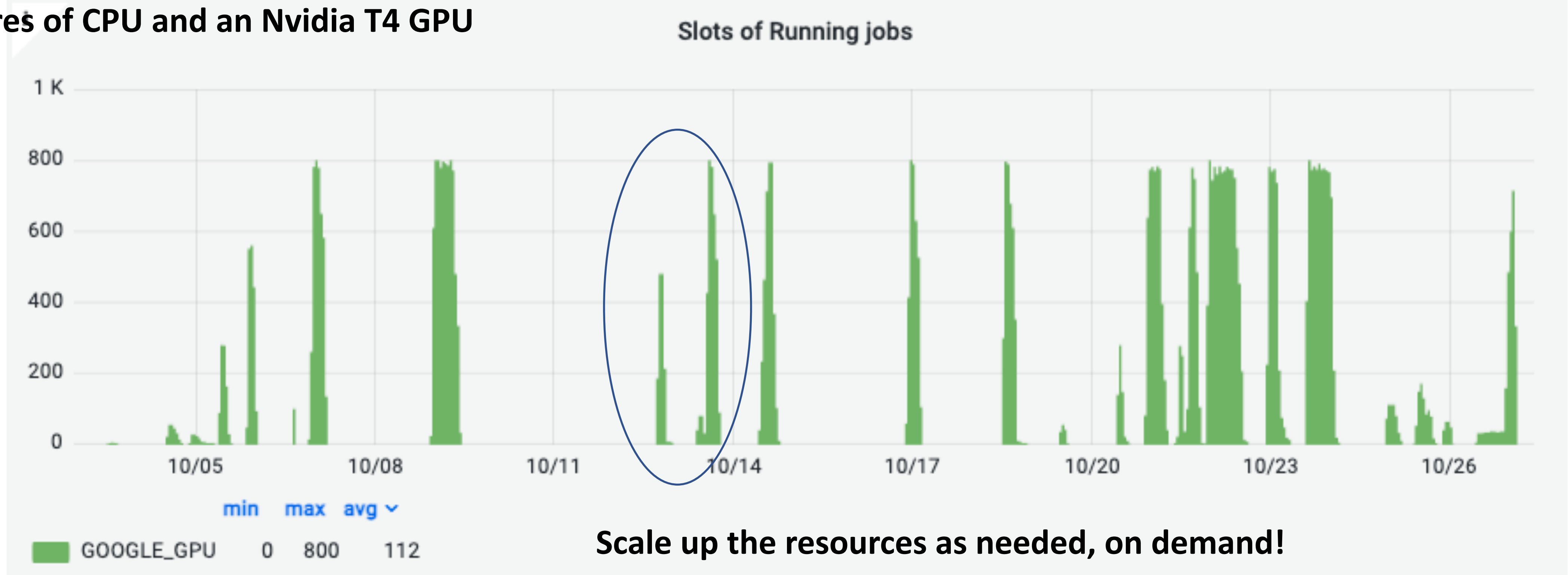


Summary of Cloud resources used for the SBI analysis R&D, in October 2022

Graph of Real Usage for the SBI analysis

Total of 200 nodes were made available for the analysis, each with four cores of CPU and an Nvidia T4 GPU

Number of CPU cores in use at given time



Scale up the resources as needed, on demand!

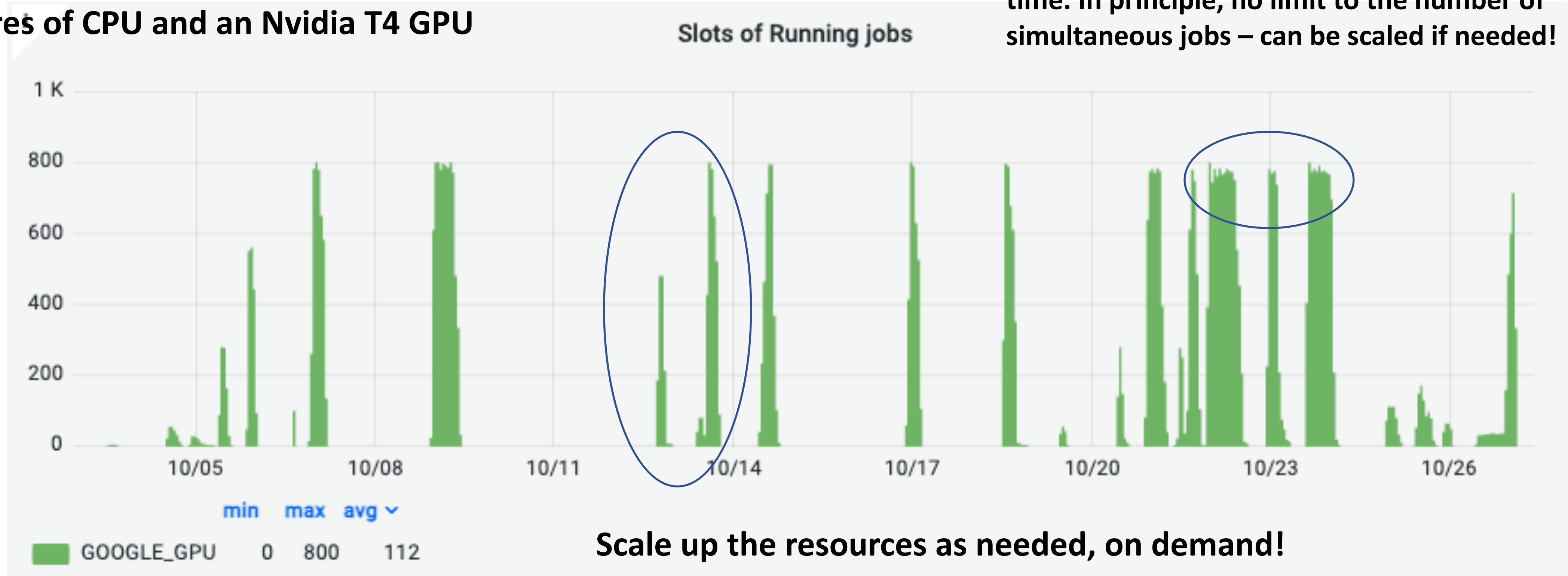
Summary of Cloud resources used for the SBI analysis R&D, in October 2022

Graph of Real Usage for the SBI analysis

Total of 200 nodes were made available for the analysis, each with four cores of CPU and an Nvidia T4 GPU

Simultaneously running 200 training jobs at a time. In principle, no limit to the number of simultaneous jobs – can be scaled if needed!

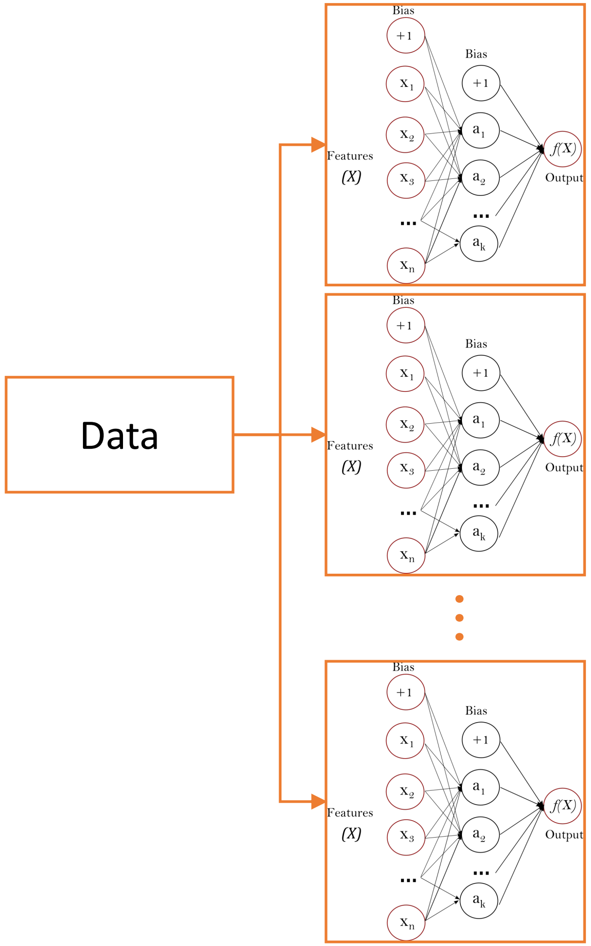
Number of CPU cores in use at given time



Scale up the resources as needed, on demand!

Summary of Cloud resources used for the SBI analysis R&D, in October 2022

Single NN Optimizations



Grid of HyperParameters (HPs) for NN optimization

Less than a day per loop with O(1k) NNs!

NN Training on the Cloud

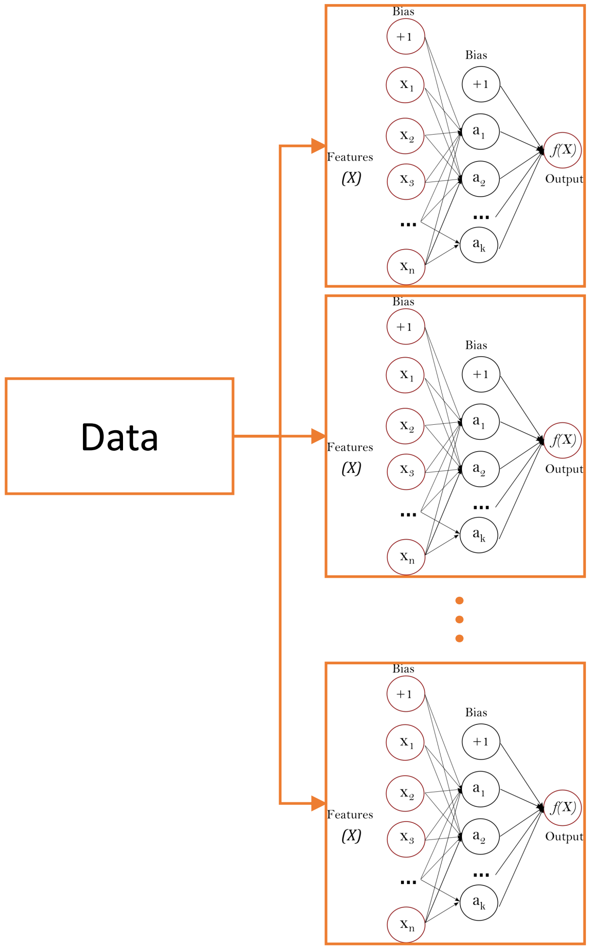
Download the diagnostics and trained NN models

Analyze the diagnostics and train with more HP's if necessary

Central Submission Script

Submit with a grid of HP's for training

Ensemble NN Optimizations



Ensemble with thousands of NNs

Less than a day per loop with O(1k) NNs!

PanDA
Google Cloud Platform

NN Training on the Cloud

RUCIO

Download the diagnostics and trained NN models.

Combine the predictions and plot diagnostics – Resubmit if necessary

docker
python™

Central Submission Script

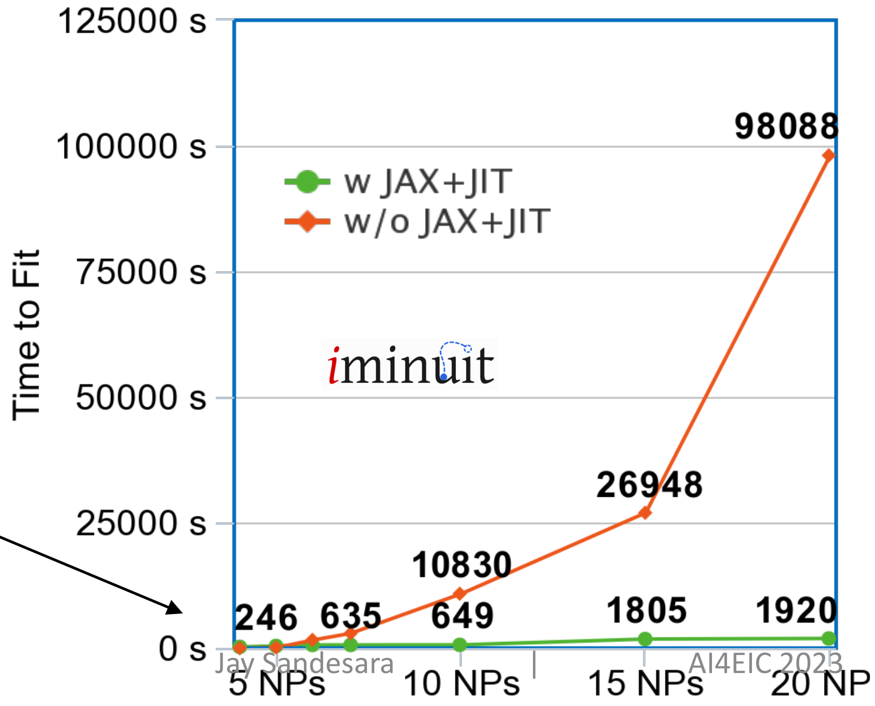
Submit with a different random seed

Final Step - Profile Likelihood Fit

- **With the unbinned SBI analysis, the computation time for profile likelihood fit with the typical $O(100)$ nuisance parameters (NPs) increases significantly** – there are event-by-event likelihood ratio predictions with systematic variations for $O(1-10M)$ number of entries!

- The new analysis **makes use of auto-differentiation and JIT compilation using the JAX library** – decreasing the likelihood fit computation time by several orders of magnitude!

For small number of NPs, JIT compiled fit takes longer because of the compilation step.



As the NPs increase, though, this trend changes dramatically and a JAX optimized, JIT compiled function becomes essential for a practical SBI analysis

Hessian Matrix

- **Proposal**: Calculate both the pull errors and NP impacts using the Hessian matrix at the best fit value, calculated with the JAX auto-diff library – Super efficient and quick!

- **Challenge**: Calculating the second derivative matrix using the full event-by-event data in SBI is a memory-intensive task!

$$\nabla^2(f) : \mathbb{R}^{100} \rightarrow \mathbb{R}^{100 \times 100}$$

Compute a $O(100) \times O(100)$ dimensional Hessian matrix

$$\mathbf{H}_f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1\,\partial x_n} \\ \frac{\partial^2 f}{\partial x_2\,\partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2\,\partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n\,\partial x_1} & \frac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

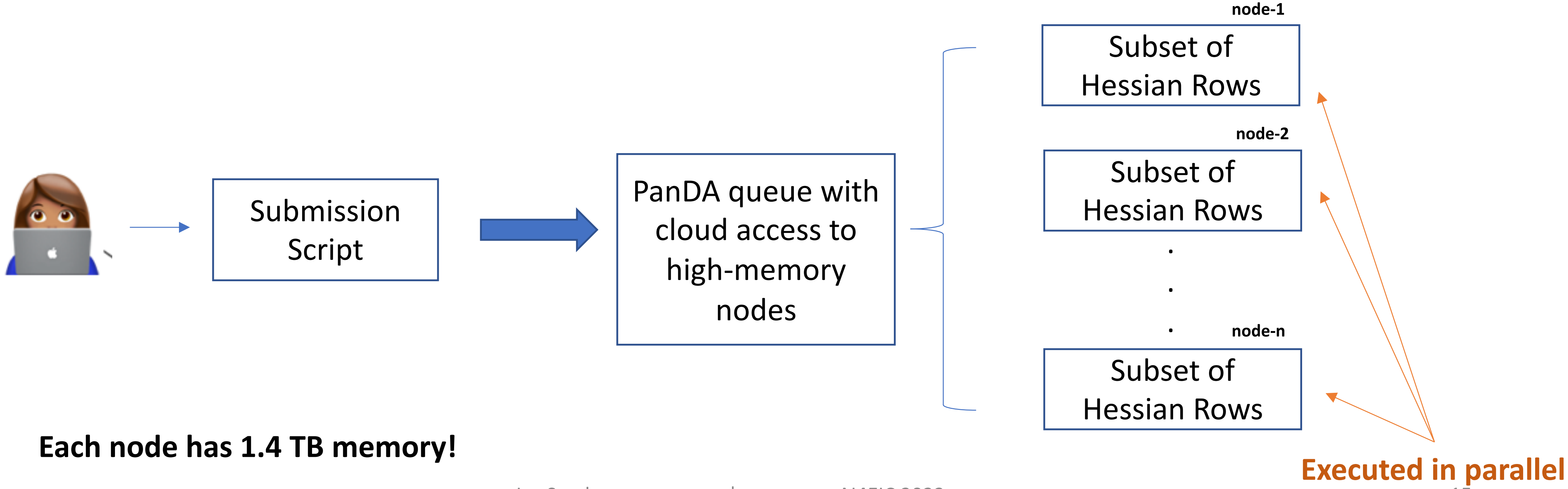
Calculate exact impacts

$$\frac{\partial \hat{\mu}}{\partial \alpha}(\hat{\mu}, \hat{\alpha}) \times \delta\alpha = -\left[\frac{\partial^2 \lambda}{\partial^2 \mu}(\hat{\mu}, \hat{\alpha})\right]^{-1} \frac{\partial^2 \lambda}{\partial \mu \partial \alpha_i}(\hat{\mu}, \hat{\alpha}) \times (\delta\alpha)$$

Cloud to the Rescue

- Google Cloud has several powerful CPU infrastructures with large memory, which was used to compute the full Hessian matrix.

- Since the computation is done row-by-row, this part of the workflow is parallelized for a quick profile likelihood fit – elasticity of using the cloud comes to the advantage!



Hessian Matrix - Challenges

- There is a way to write a memory-efficient solution - We make use the following identity to **compute Hessian vector products** instead of the full Hessian:

$$\nabla^2 f(x)\, v = \nabla[x \to \nabla f \cdot v]$$

Setting v to unit-vectors materializes the full Hessian one row at a time!

```python
def hvp(f, x, v):
    return grad(lambda x: jnp.vdot(grad(f)(x), v))(x)
```

Reducing the problem to estimating gradients of only scalar valued functions $\nabla(f) : \mathbb{R}^{100} \to \mathbb{R}^{100}$

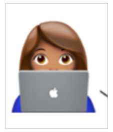
Calculated one row at a time

$$\mathbf{H}_f = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1\,\partial x_n} \\[2.2ex] \dfrac{\partial^2 f}{\partial x_2\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2\,\partial x_n} \\[2.2ex] \vdots & \vdots & \ddots & \vdots \\[2.2ex] \dfrac{\partial^2 f}{\partial x_n\,\partial x_1} & \dfrac{\partial^2 f}{\partial x_n\,\partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix},$$

For unbinned analysis, this still requires hundreds of GBs of RAM for computation! Need specialized hardware.

Can be time consuming to compute one row at a time for $O(100)$ NPs, even with JIT compilation.

Bird's Eye Overview – Full SBI analysis



Submission Script

PanDA queue with cloud GPU access

NN training

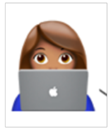
NN training

.
.
.

NN training

Train an ensemble of NN's for likelihood ratio prediction

Set of un-biased and low-variance estimates of POI + NP parametrized likelihood ratios for every event.



Submission Script

PanDA queue with cloud access to high-memory nodes

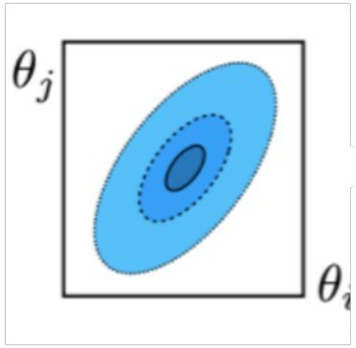
Subset of Hessian Rows

Subset of Hessian Rows

.
.
.

Subset of Hessian Rows

Do a profile likelihood fit and find NP pulls and impacts



Statistical Inference using the SBI technique!

Many More Applications

- Applications that require large-scale GPU and/or high-end CPU infrastructure can benefit from the easy availability and elasticity of cloud-based infrastructure.

OmniFold

Un-binned unfolding technique that requires ensemble NNs for an accurate estimation of density ratios.

https://arxiv.org/abs/1911.09107
Phys. Rev. Lett. 124, 182001 (2020)

FastCaloGAN

Requires training 300 GANs for a very accurate simulation of the calorimeter showers.

https://cds.cern.ch/record/2742369?ln=en

HPO Service ATLAS

Automated optimization of HPs in machine learning models using PanDA+iDDS

https://cds.cern.ch/record/2742369?ln=en

Summary and Outlook

- Scale-able, on-demand GPU and high-memory CPU infrastructure made available using Google Cloud Platform and integrated with the ATLAS distributed computing system has made a full experimental analysis with Simulation-Based Inference practically possible.

- The analysis is still in the approval stage in ATLAS – plan to make public early next year. With the presented workflow and cloud infrastructure, similar types of computationally challenging analysis can be very convenient to pick up.

- All inclusive, the fully parametrized physics likelihood ratios in our ATLAS analysis are described with over a billion NN parameters – first time we are reaching this order of magnitude in ATLAS (or the wider HEP experiment community)!

N.B: HPCs can and have been used as an alternative to cloud-based infrastructure, but the latter is more flexible. Faster on-demand workflow scheduling is also possible with cloud.