

Machine Learning at TACC Distributed Training

July 2024

PRESENTED BY:

Sikan Li

Research Engineering, Scalable
Computational Intelligence

sli@tacc.utexas.edu

Distributed Training: Motivation

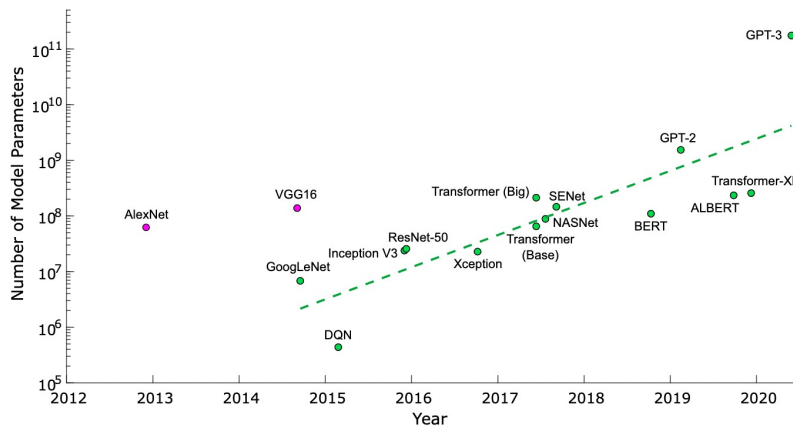


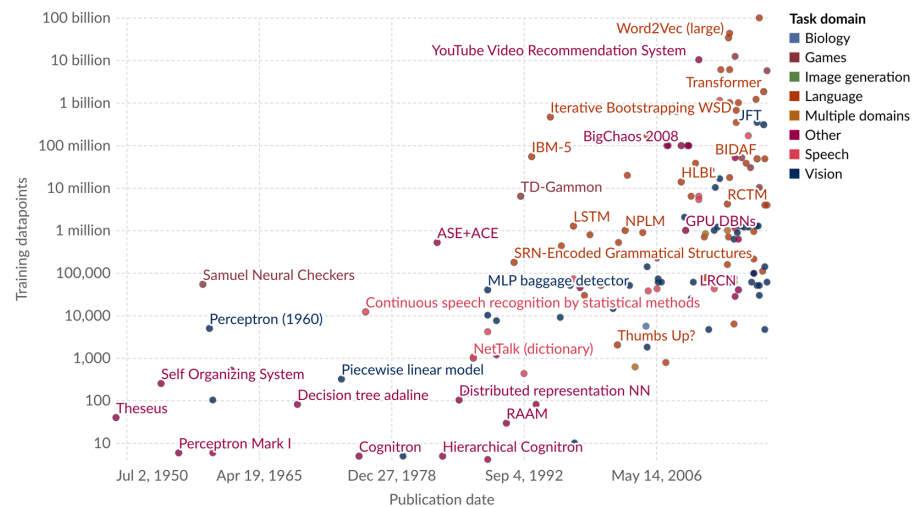
Figure 1. Number of parameters, i.e., weights, in recent landmark neural networks ^{1,2,31-43} (references dated by first release, e.g., on arXiv). The number of multiplications (not always reported) is not equivalent to the number of parameters, but larger models tend to require more compute power, notably in fully-connected layers. The two outlying nodes (pink) are AlexNet and VGG16, now considered over-parameterized. Subsequently, efforts have been made to reduce DNN sizes, but there remains an exponential growth in model sizes to solve increasingly complex problems with higher accuracy.

[Image Credit](#)

Datapoints used to train notable artificial intelligence systems

Each domain has a specific data point unit; for example, for vision it is images, for language it is words, and for games it is timesteps. This means systems can only be compared directly within the same domain.

Our World
in Data



Data source: Epoch (2024)

OurWorldInData.org/artificial-intelligence | CC BY

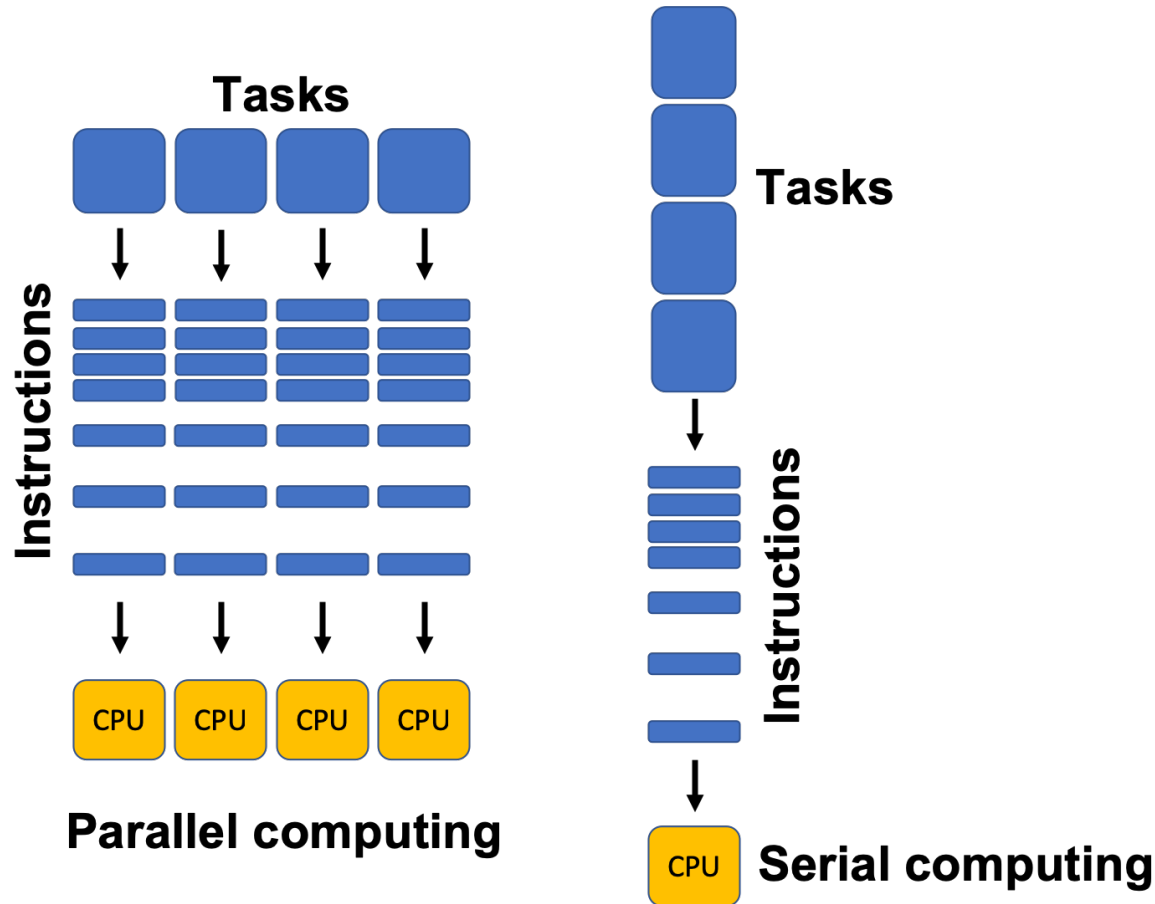
[Image Credit](#)

Outline

- Introduction to Parallel Computing on HPC
- Basics of Distributed Data Parallel
- Fault Tolerance & Torchrn
- MPI and SLURM
- Hands-on

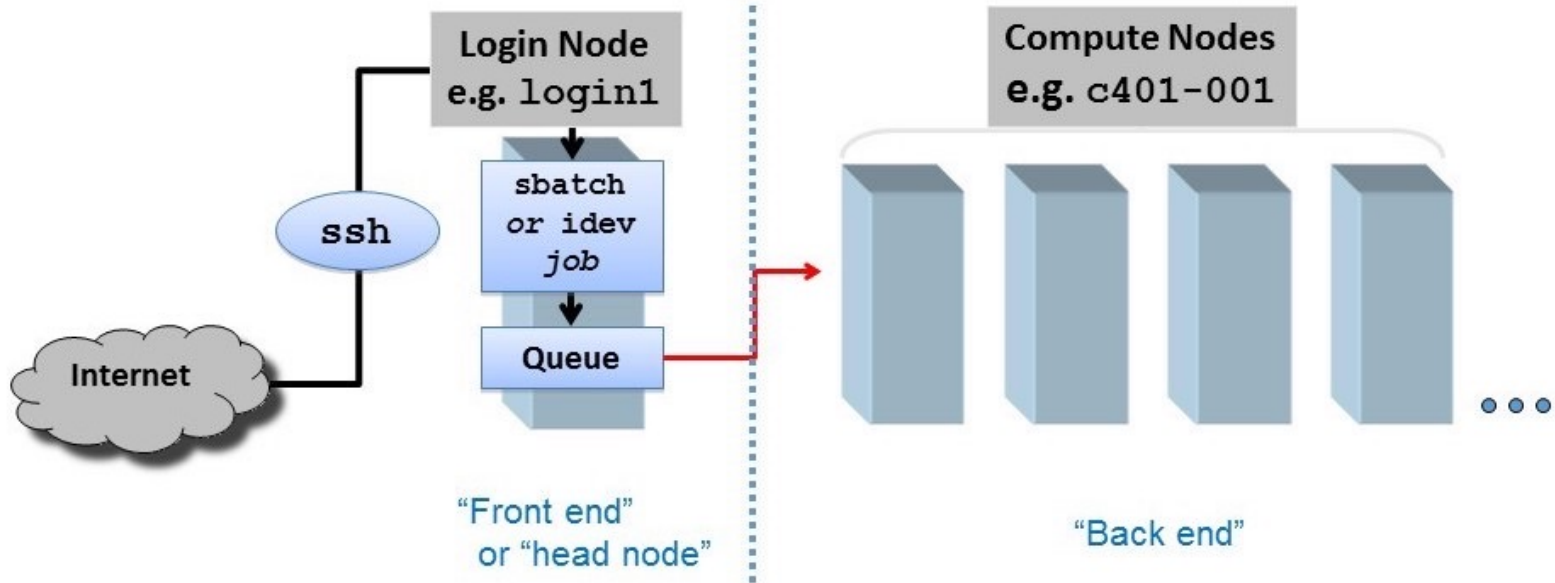
Introduction to parallel computing on High Performance Computing (HPC)

Introduction to parallel computing on HPC



Serial vs. Parallel Computation

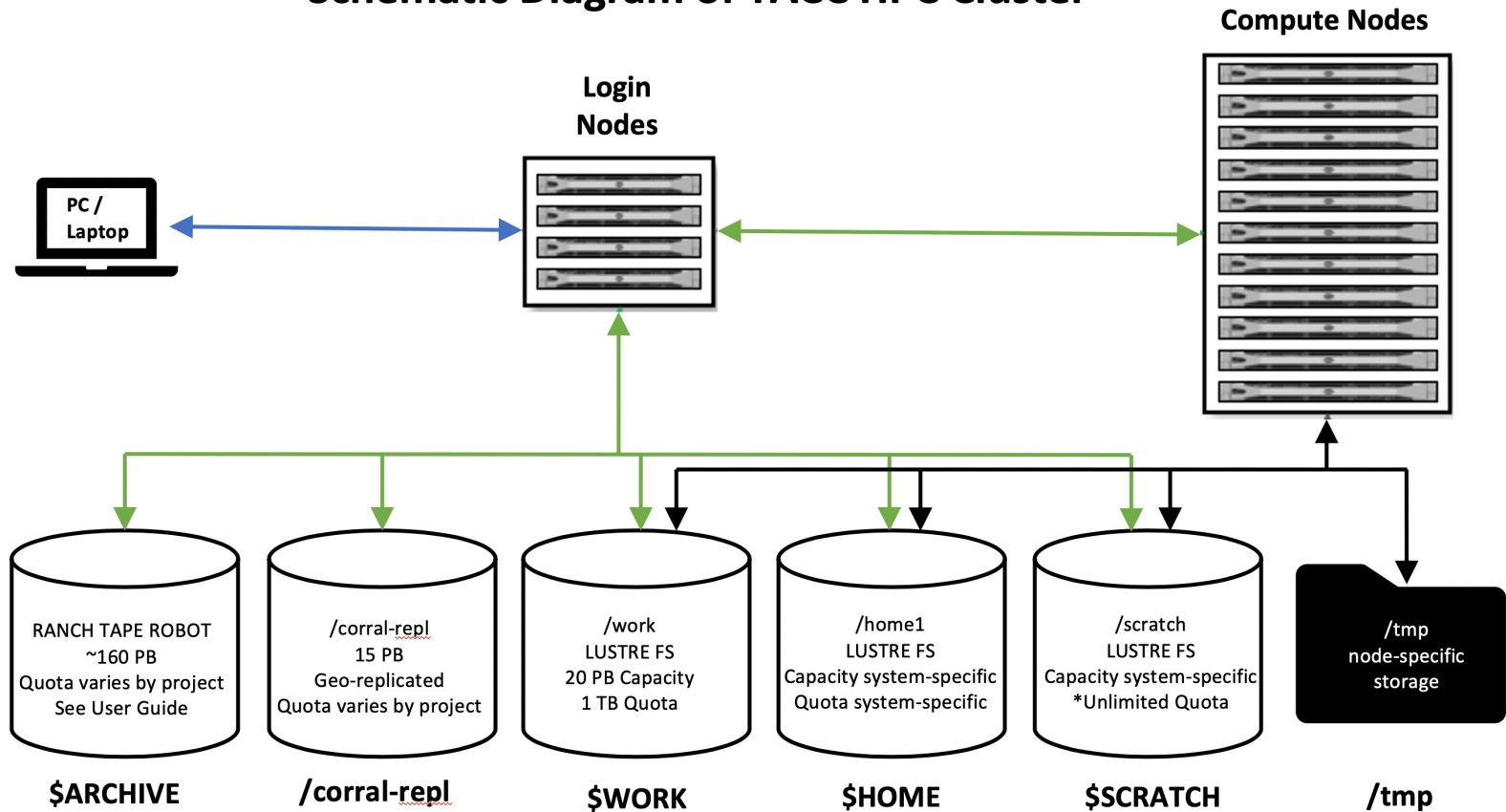
Introduction to Parallel Computing on HPC



Login vs Compute Nodes

Introduction to Parallel Computing on HPC

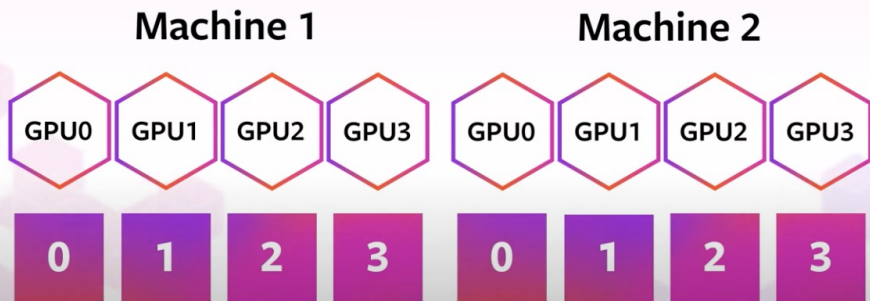
Schematic Diagram of TACC HPC Cluster



Introduction to Parallel Computing on HPC

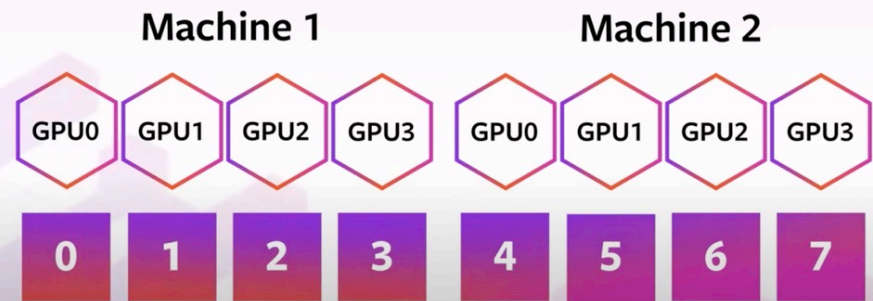
- Local Rank & Global Rank
- World Size

Local Rank



Local Rank

Global Rank



Global Rank

Introduction to Parallel Computing on HPC

Challenge: Communication

$$A = B * C$$

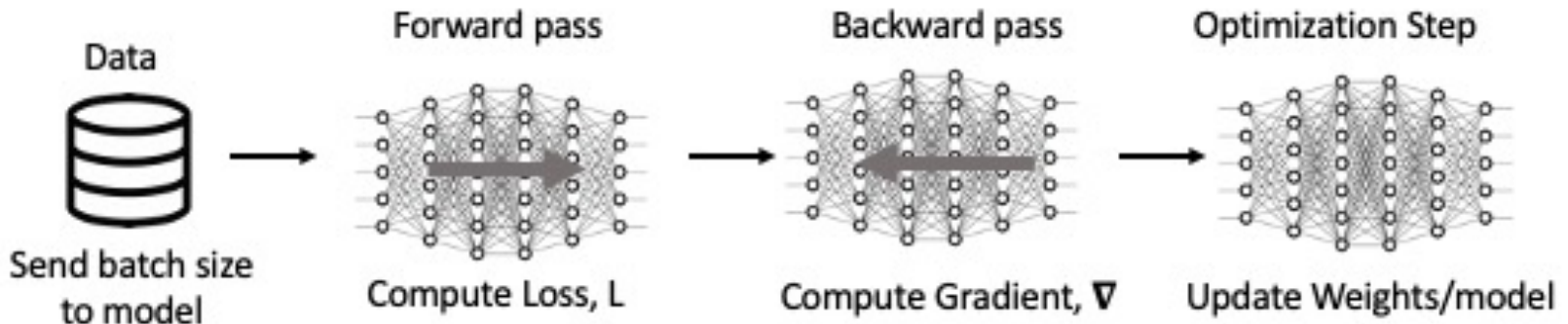
$$D = E * F$$

$$G = A * D$$

- Synchronization
- Data Movement
- Collective Computation

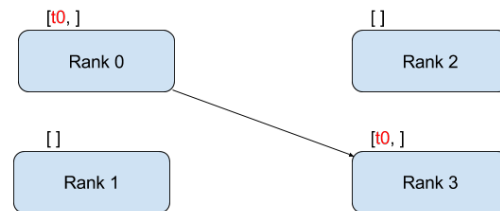
Basics of Distributed Data Parallel (DDP)

Basics of DDP



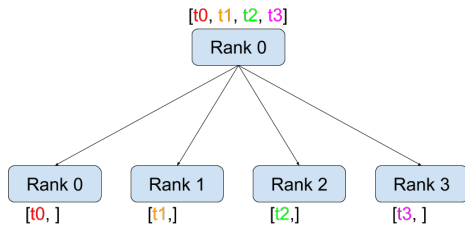
Visual representation of one step in training a neural network with one GPU.

Basics of DDP

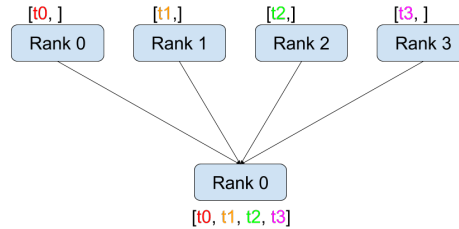


Point-to-Point Communication

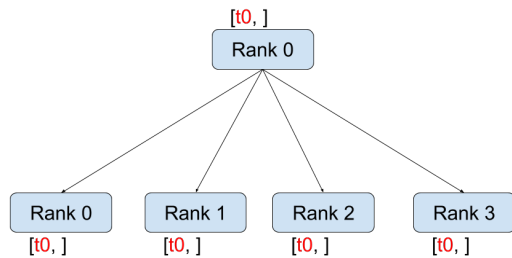
Basics of DDP



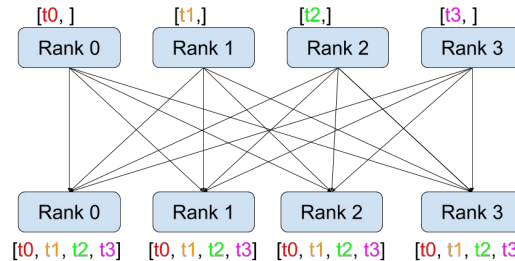
Scatter



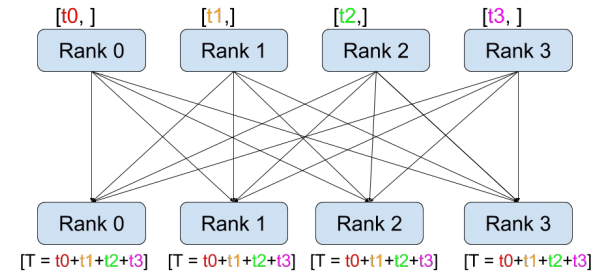
Gather



Broadcast



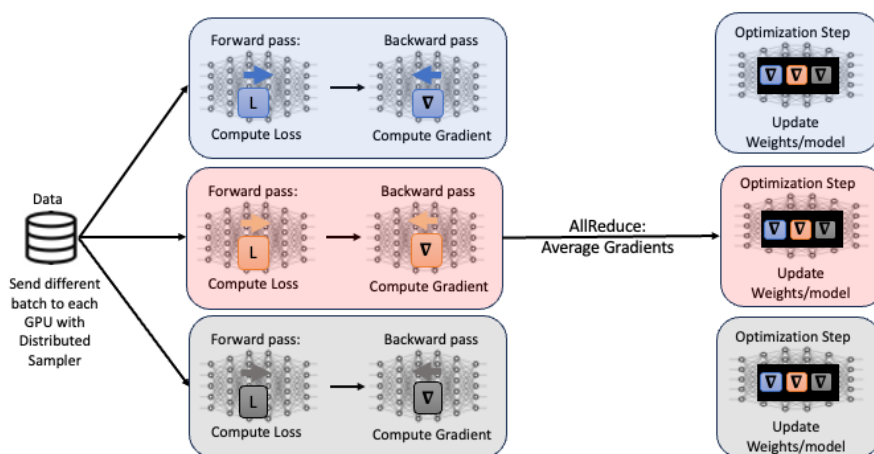
All-Gather



All-Reduce

Collective Communication

Basics of DDP



- Synchronization
- Data Movement
- Collective Computation

Visual representation of one iteration of DDP using 3 GPUs.

Basics of DDP

PyTorch Distributed

DistributedDataParallel

Basics of DDP

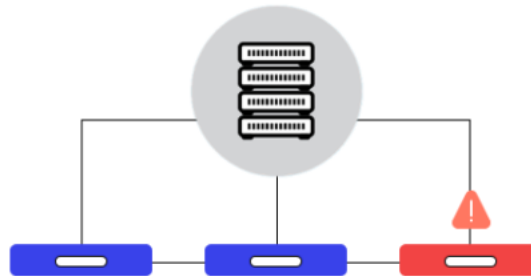
Communication backend for torch.distributed

- Gloo
- NVIDIA Collective Communication Library (NCCL)
- Message Passing Interface (MPI)

Fault Tolerance & Torchrun

Fault Tolerance & Torchrun

Fault Tolerance



Fault Tolerance

Fault Tolerance & Torchrun



Visual of checkpointing. CP refers to a point in time when a checkpoint is saved.

Fault Tolerance & Torchrun

Torchrun:

`torch.distributed.launch +`

- worker failures
- environment variables
- elasticity



Message Passing Interface (MPI) & Simple Linux Utility for Resource Management (SLURM)

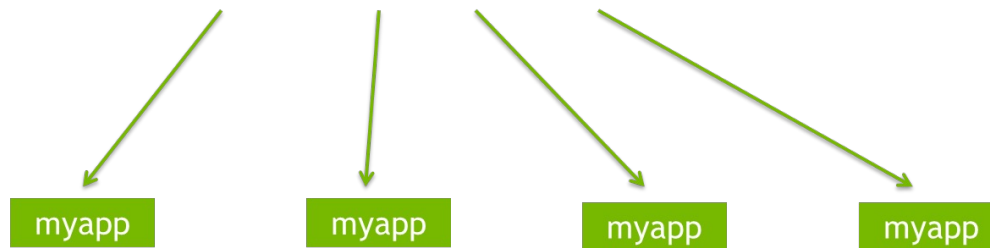
MPI & SLURM

Message Passing Interface (MPI)

MPI is a standardized and portable message-passing standard designed to function on parallel computing architectures.

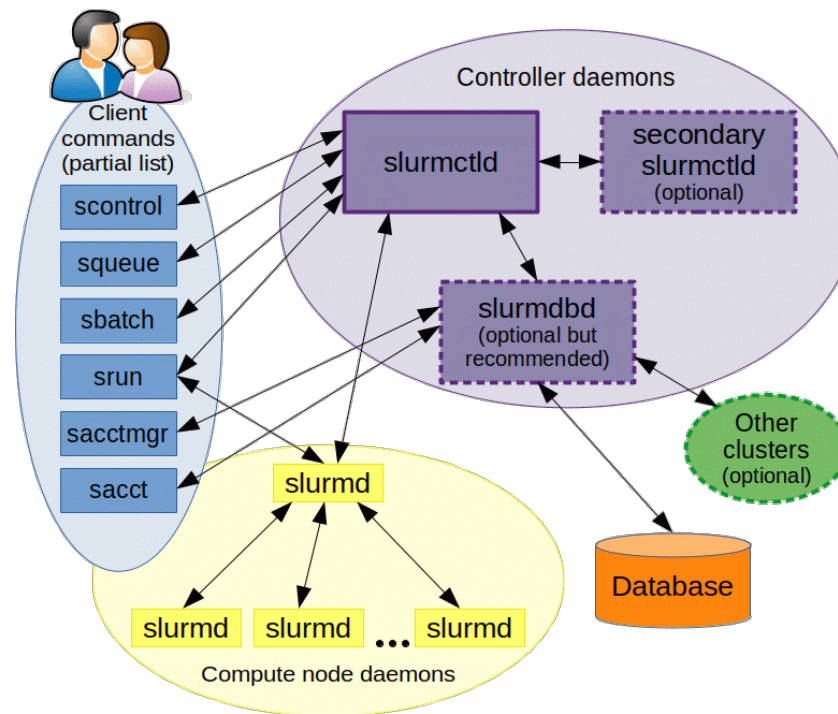
mpirun

```
mpirun -np 4 ./myapp <args>
```



MPI & SLURM

Simple Linux Utility for Resource Management (SLURM)



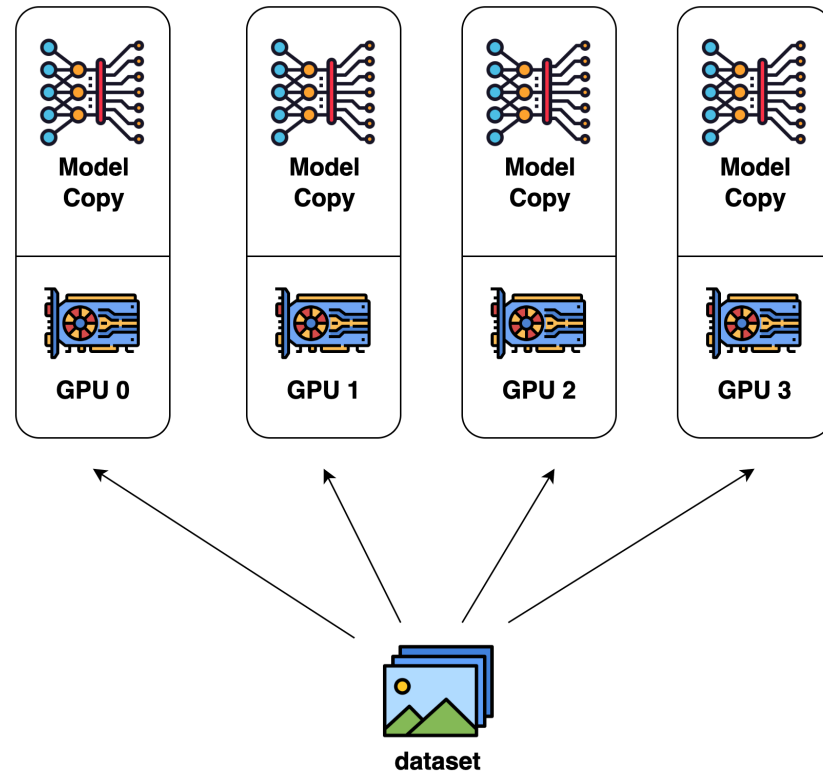
Slurm components

More Parallelization Techniques

- Data Parallel
- Model Parallel
- Pipeline Parallel

More Parallelization Techniques

Data Parallel

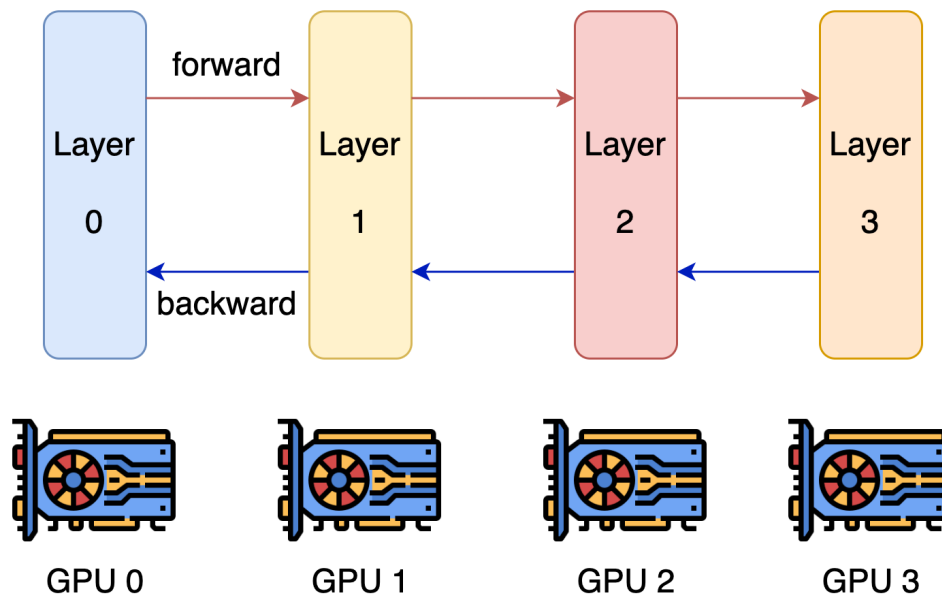


Parameters (Model) are duplicated on all nodes

[Image Credit](#)

More Parallelization Techniques

Pipeline Parallel

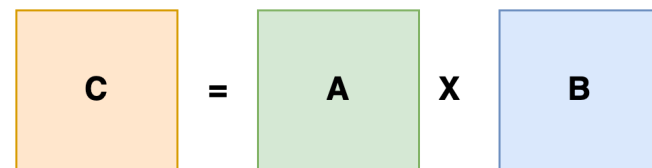


[Image Credit](#)

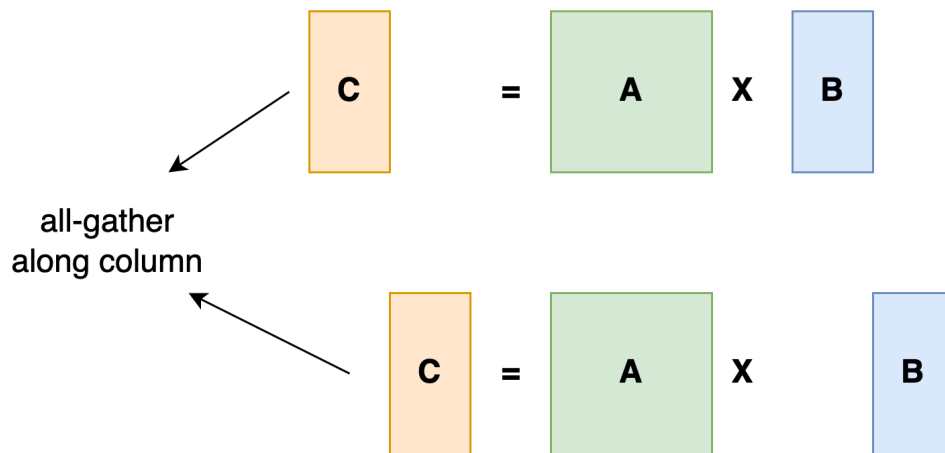
Weights are distributed to all nodes

More Parallelization Techniques

Model Parallel



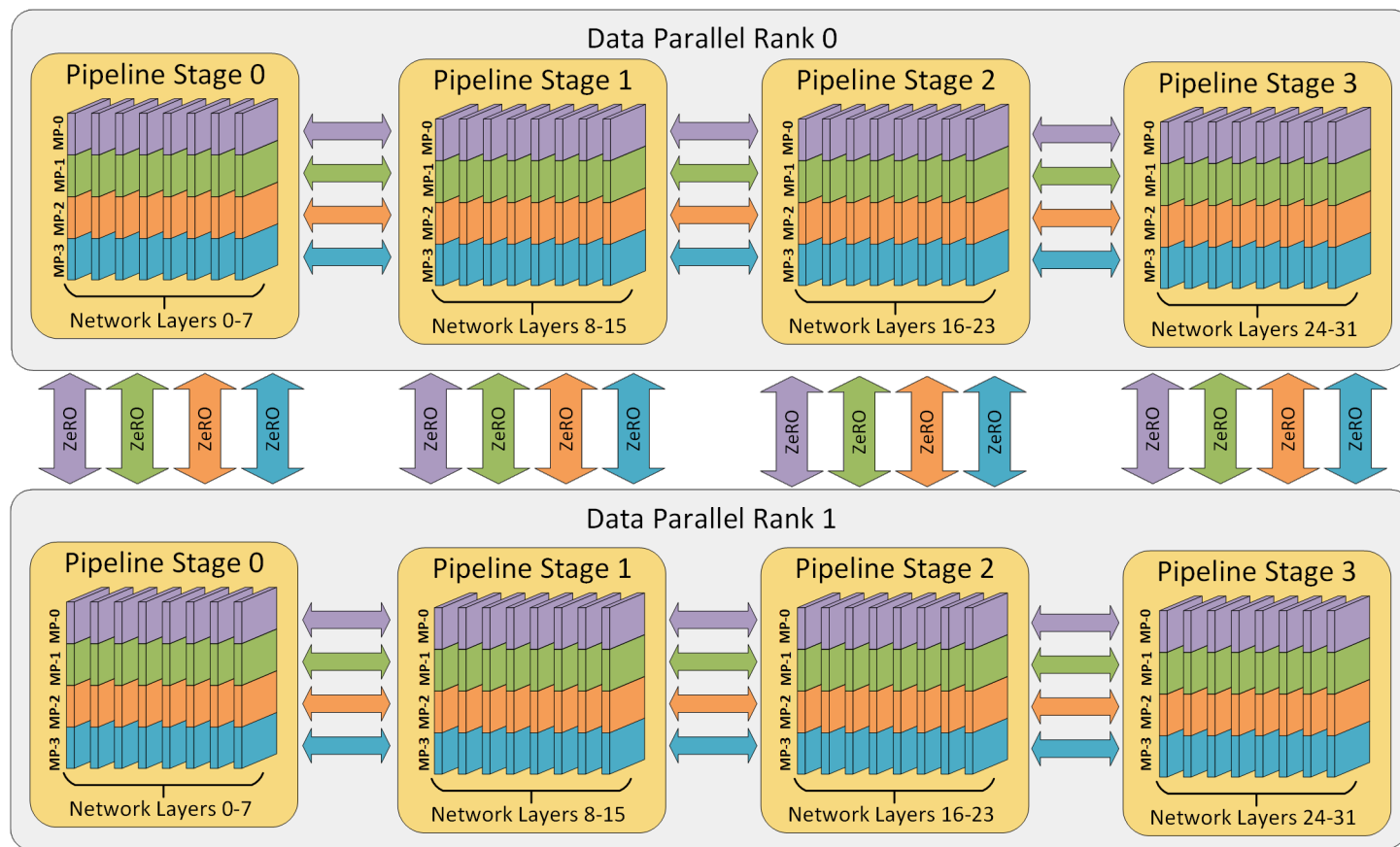
Non-distributed



Column-Splitting Tensor Parallel

Partitions the individual layers.

More Parallelization Techniques



[Image Credit](#)

Hands-on Exercise

Hands-on Exercise

In the three notebooks, we will introduce how to parallelize the training process of a CNN classifier.

We will start with multiple GPUs on a single node, and then approach multi node distributed training.

Hands-on Exercise

Launch a Jupyter Notebook

Resource: Frontera

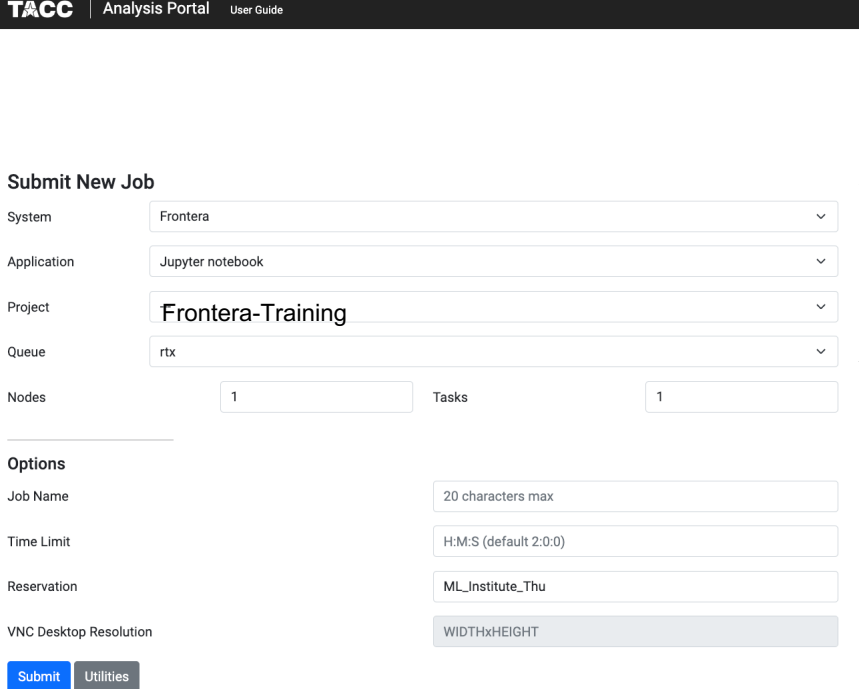
Project: Frontera-Training

Session Type: Jupyter Notebook

Reservation ID: ML_Institute_Thu

Queue: rtx

Time Limit: 04:00:00



The screenshot shows the 'Submit New Job' interface of the TACC Analysis Portal. The form includes dropdown menus for System (Frontera), Application (Jupyter notebook), Project (Frontera-Training), and Queue (rtx). Below these are input fields for Nodes (1) and Tasks (1). An 'Options' section contains fields for Job Name (20 characters max), Time Limit (H:M:S (default 2:0:0)), Reservation (ML_Institute_Thu), and VNC Desktop Resolution (WIDTHxHEIGHT). At the bottom are 'Submit' and 'Utilities' buttons.

TACC | Analysis Portal | User Guide

Submit New Job

System: Frontera

Application: Jupyter notebook

Project: Frontera-Training

Queue: rtx

Nodes: 1 Tasks: 1

Options

Job Name: 20 characters max

Time Limit: H:M:S (default 2:0:0)

Reservation: ML_Institute_Thu

VNC Desktop Resolution: WIDTHxHEIGHT

[Submit](#) [Utilities](#)