

**AI4S**  
BSC AI 4 Science Fellowships

# Profiling and Optimizing AI Trainings

Alexandros Paliouras – AI Engineer  
Guillem Rovira Cortiada – AI Engineer  
*Barcelona Supercomputer Center*



# Agenda

1. Introduction
  - Systems' Topology
  - GPU communication (NCCL)
2. NVIDIA Nsight Systems
  - Nsys command
  - Nsight client
3. Paraver
  - Nsys2prv
  - Paraver client
4. Exercises



# Agenda

## 1. Introduction

- **Systems' Topology**
- **GPU communication (NCCL)**

## 2. NVIDIA Nsight Systems

- Nsys command
- Nsight client

## 3. Paraver

- Nsys2prv
- Paraver client

## 4. Exercises

# Why code profiling?

"You can have 1.000.000 GPU hours for FREE!..."



# Why code profiling?

"You can have 1.000.000 GPU hours for FREE!..."

Only condition:

- Prove that all these hours will be used efficiently





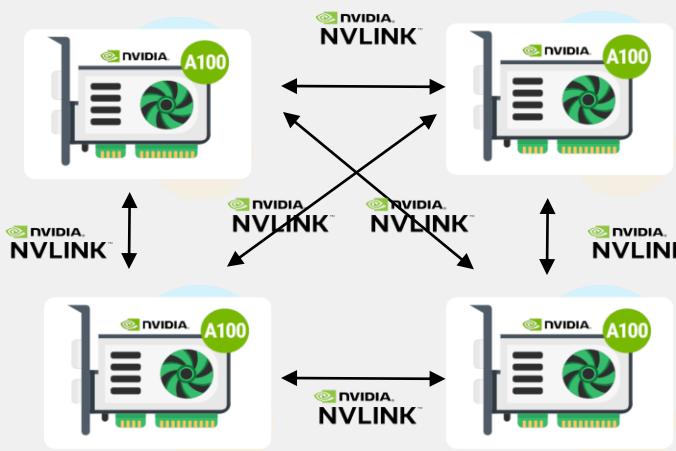
# Questions that need answers...

- Is my GPU "starving" for data ?
  - Maybe the data loader is taking longer than expected
- Is the batch size optimized for memory?
  - Small batch sizes might be underutilizing the available
- How much time is spent on Communication vs. Computation?
  - Communication between gpus is blocking unrelated computations
- Is "Tensor Parallelism" (TP) causing too much intra-node latency?
  - Maybe the TP degree I have chosen is too high my hardware topology, and the available bandwidth is limiting
- Where is the "Bubble" in my Pipeline?
  - Pipeline parallelism (PP) introduces GPU dependencies, slowing down out training

# How do GPUs communicate?

## Intra-Node

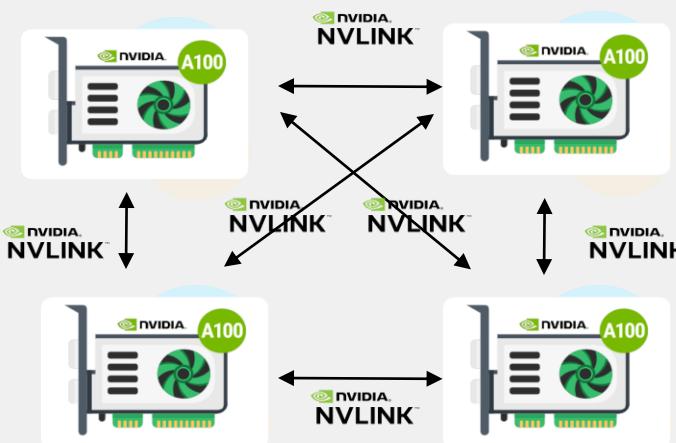
- NVLink cables  
(Leonardo 64GiB HBM2e NVLink 3.0 (200 GB/s))
- NVSwitch (>4 GPUs node) + NVLink
- PCIe (cpu-gpu, slow)



# How do GPUs communicate?

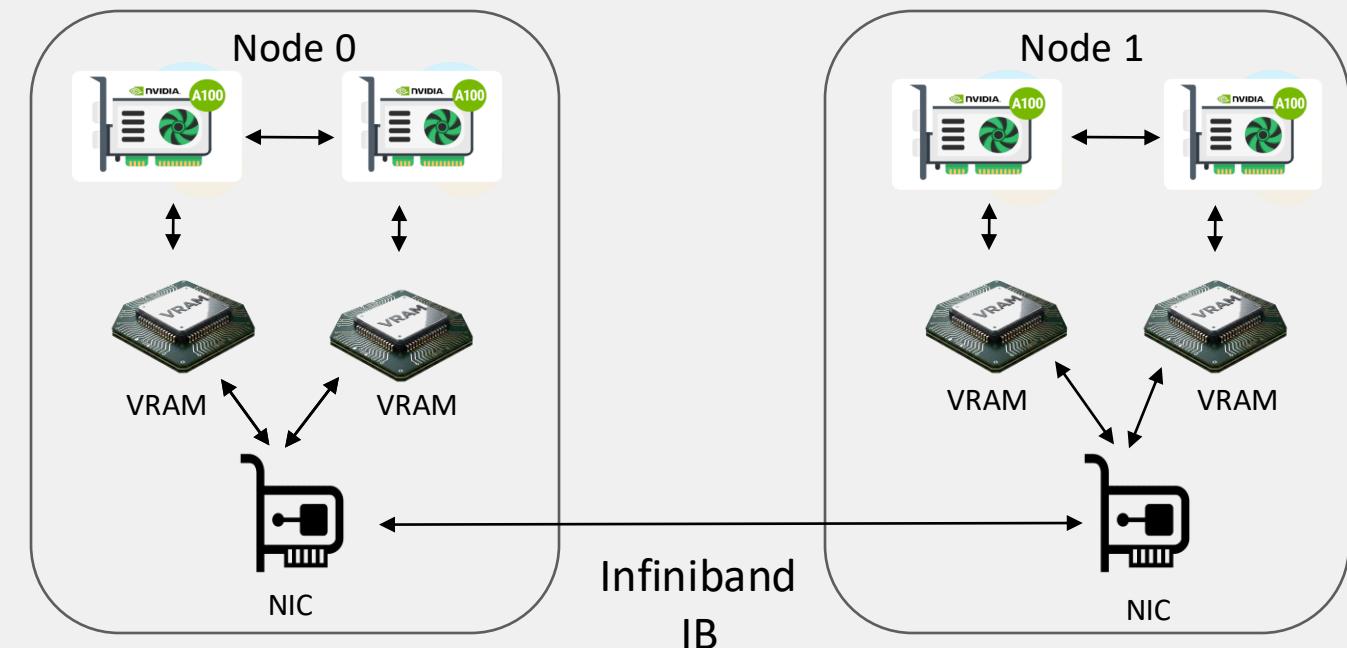
## Intra-Node

- NVLink cables  
(Leonardo 64GiB HBM2e NVLink 3.0 (200 GB/s))
- NVSwitch (>4 GPUs node) + NVLink
- PCIe (cpu-gpu, slow)



## Inter-Node

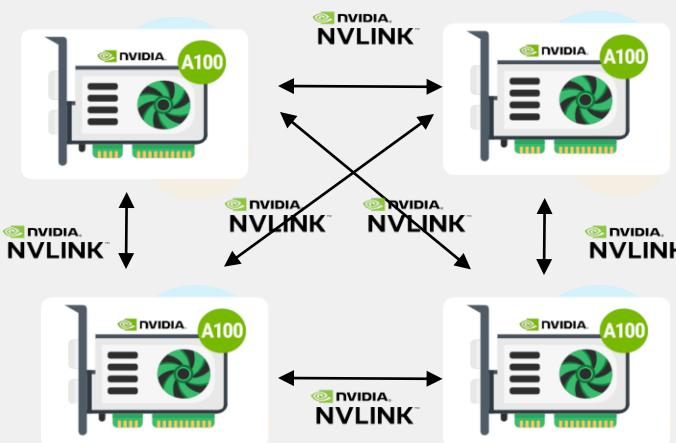
- InfiniBand GPU inter-node Remote Direct Memory Access  
(Leonardo 200 Gbps~25GB/s NVIDIA Mellanox HDR InfiniBand)
- Ethernet (fallback, slow)



# How do GPUs communicate?

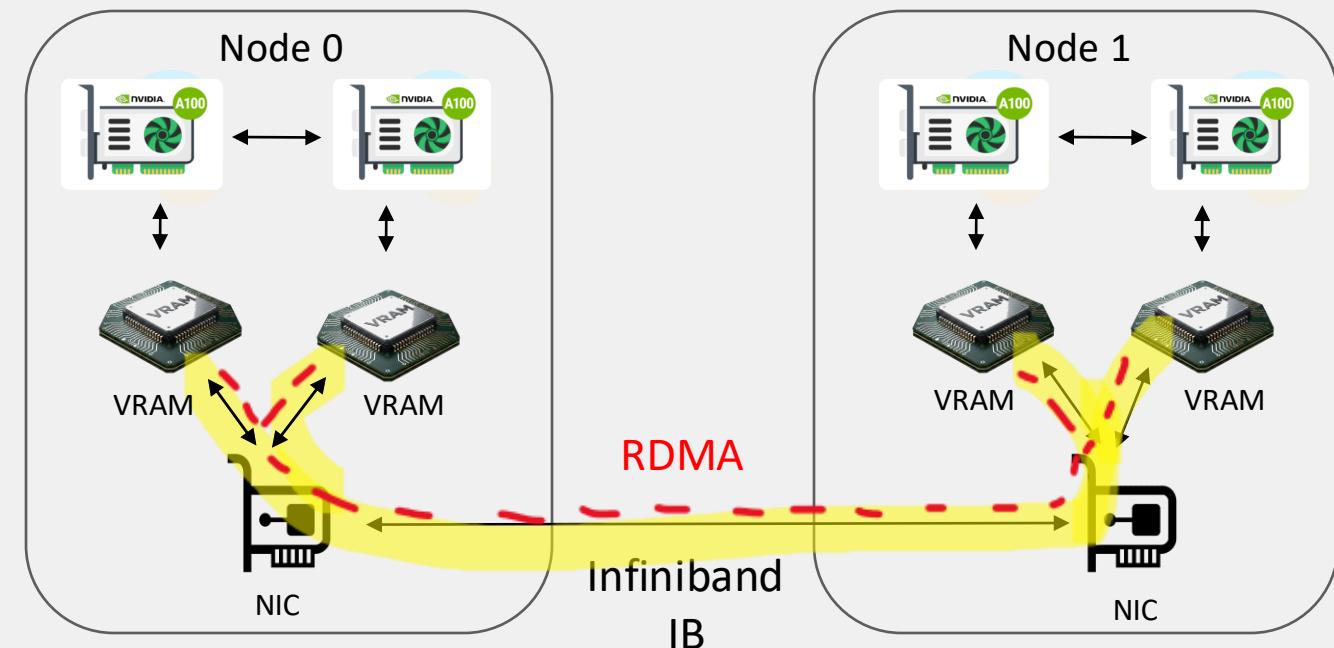
## Intra-Node

- NVLink cables  
(Leonardo 64GiB HBM2e NVLink 3.0 (200 GB/s))
- NVSwitch (>4 GPUs node) + NVLink
- PCIe (cpu-gpu, slow)



## Inter-Node

- InfiniBand GPU inter-node Remote Direct Memory Access  
(Leonardo 200 Gbps~25GB/s NVIDIA Mellanox HDR InfiniBand)
- Ethernet (fallback, slow)



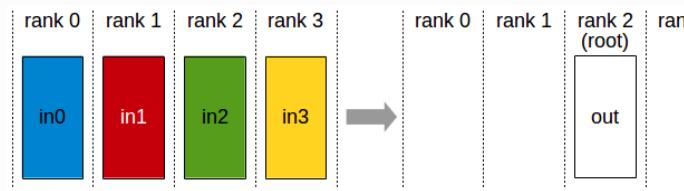
# How do GPUs communicate?

## NCCL: NVIDIA Collective Communication Library

- Implements multi-GPU and multi-node communication primitives optimized for NVIDIA GPUs and networking.

### Reduce

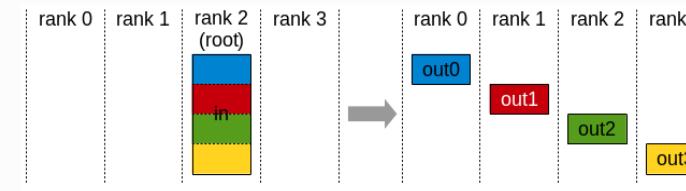
The Reduce operation performs the same operation as AllReduce, but stores the result only in the receive buffer of a specified root rank.



Reduce operation: one rank receives the reduction of input values across ranks.

### Scatter

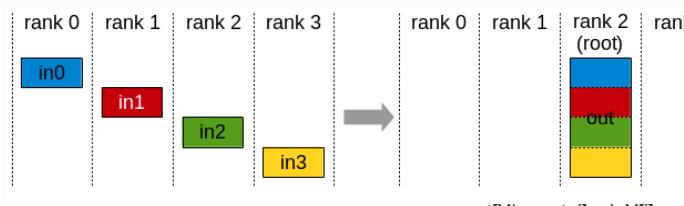
The Scatter operation distributes a total of  $N*k$  values from the root rank to  $k$  ranks, each rank receiving  $N$  values.



Scatter operation: root rank distributes data to all ranks.

### Gather

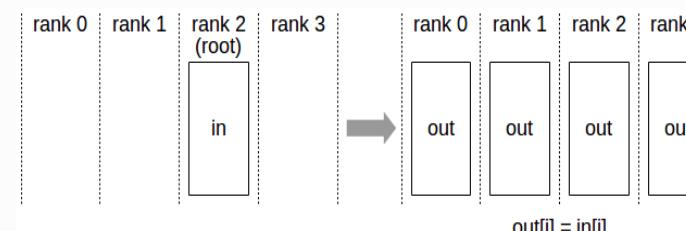
The Gather operation gathers  $N$  values from  $k$  ranks into an output buffer on the root rank of size  $k*N$ .



Gather operation: root rank receives data from all ranks.

### Broadcast

The Broadcast operation copies an  $N$ -element buffer from the root rank to all the ranks.



Broadcast operation: all ranks receive data from a "root" rank.

# How do GPUs communicate?

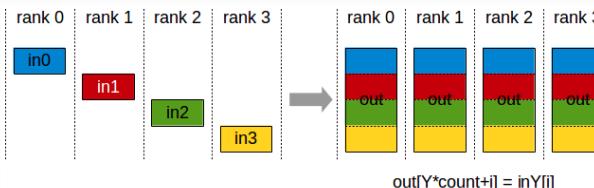
## NCCL: NVIDIA Collective Communication Library

- Implements multi-GPU and multi-node communication primitives optimized for NVIDIA GPUs and networking.

### AllGather

The AllGather operation gathers N values from k ranks into an output buffer of size  $k \times N$ , and distributes that result to all ranks.

The output is ordered by the rank index. The AllGather operation is therefore impacted by a different rank to device mapping.

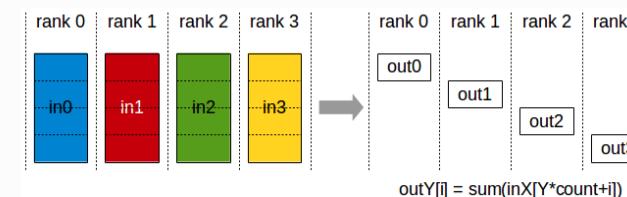


AllGather operation: each rank receives the aggregation of data from all ranks in the order of the ranks.

### ReduceScatter

The ReduceScatter operation performs the same operation as Reduce, except that the result is scattered in equal-sized blocks between ranks, each rank getting a chunk of data based on its rank index.

The ReduceScatter operation is impacted by a different rank to device mapping since the ranks determine the data layout.

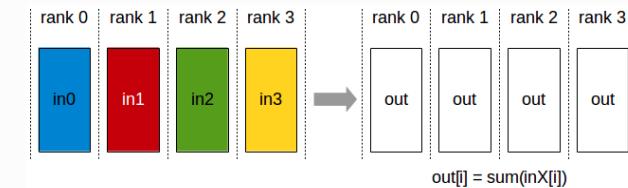


Reduce-Scatter operation: input values are reduced across ranks, with each rank receiving a subpart of the result.

### AllReduce

The AllReduce operation performs reductions on data (for example, sum, min, max) across devices and stores the result in the receive buffer of every rank.

In a sum allreduce operation between k ranks, each rank will provide an array in of N values, and receive identical results in array out of N values, where  $out[i] = in0[i] + in1[i] + \dots + in(k-1)[i]$ .

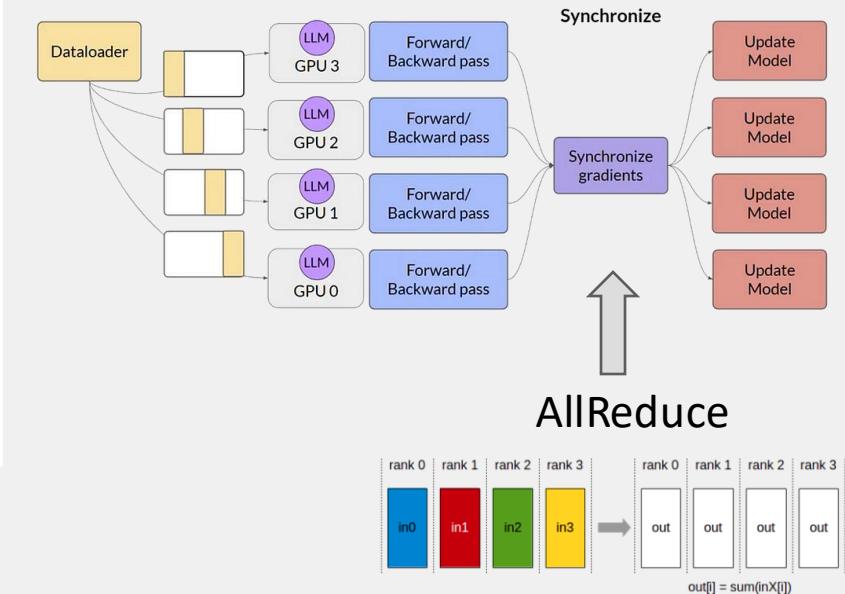


All-Reduce operation: each rank receives the reduction of input values across ranks.

# DDP Communication

Collective	Role
AllReduce	Aggregates gradients across all GPUs after backward pass. This is the default and the most performance-critical operation in DDP.
Broadcast	Used at initialization to broadcast model weights from a master rank to all other ranks, ensuring consistent start states before training.

## Distributed Data Parallel (DDP)

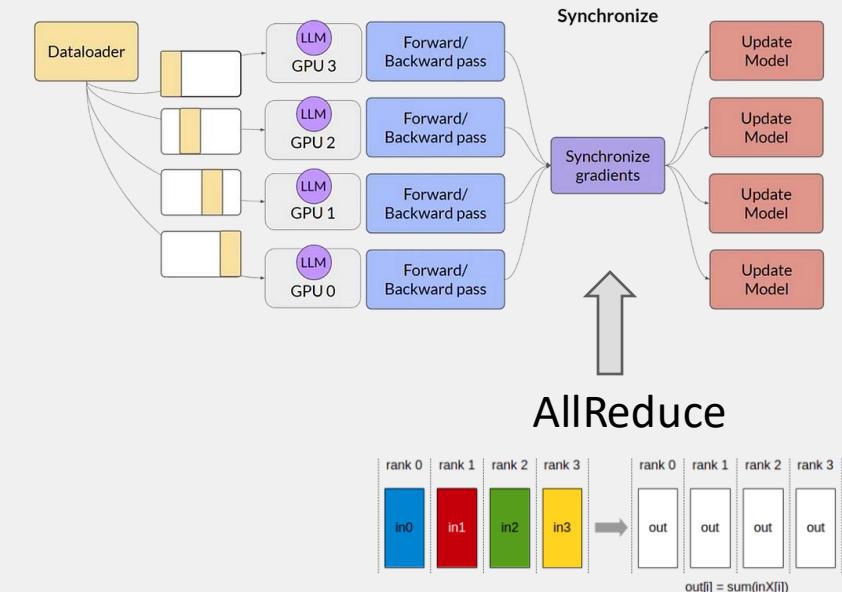




# DDP Communication

Collective	Role
AllReduce	Aggregates gradients across all GPUs after backward pass. This is the default and the most performance-critical operation in DDP.
Broadcast	Used at initialization to broadcast model weights from a master rank to all other ranks, ensuring consistent start states before training.

## Distributed Data Parallel (DDP)



Trace Pattern:

Forward GEMM → Backward GEMM → AllReduce (bucketed)

\*GEMM: CUDA kernel for General Matrix Multiplication



# Larger Model VRAM requirements - example

- **Training static states:**
  - Weights
  - Gradients
  - Optimizer states
- **ADAM optimizer:**
  - Master Weights (FP32)
  - Momentum & variance (FP32)  
for each model parameter
- **Dynamic or live states:**
  - Activations (depends batch size, hidden layers, sequence length, etc. )
  - Activation gradients
  - Communication & precision casting buffers
  - Etc.

Component	Per Billion Params	Mistral 7B Total	Equal partitions (4 GPUs)
<b>Model weights</b>	x2bytes → 2 GB (FP16/BF16)	14 GB	3.5 GB / GPU
<b>Gradients</b>	x2bytes → 2 GB (FP16/BF16)	14 GB	3.5 GB / GPU
<b>Optimizer States</b>	3x4bytes → 12 GB (Adam)	84 GB	21 GB / GPU
<b>Total Static</b>	16 GB	112 GB	<b>28 GB / GPU</b>

Dynamic states can vary in this example 8-20GB or more



# FSDP Communication

## Collective

## Role

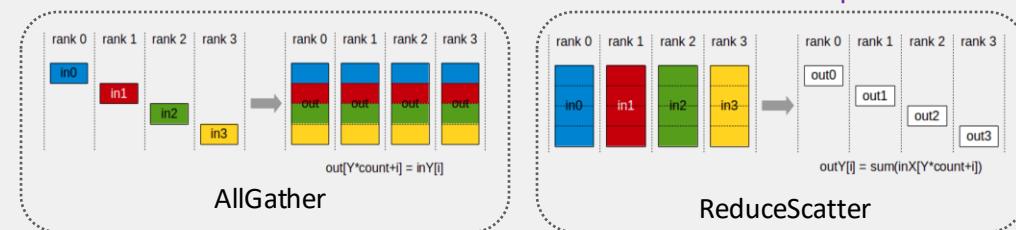
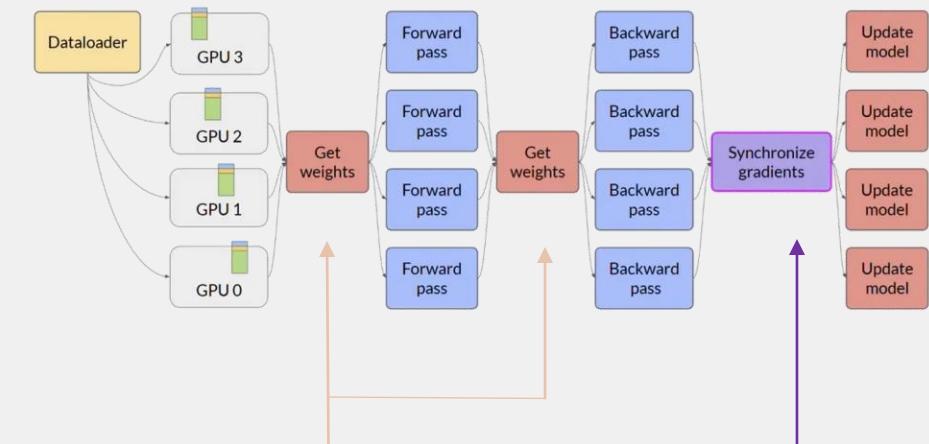
### AllGather

During forward and backward pass, before computing a layer, the full parameter shard is assembled across GPUs, so that each GPU can compute activations, attention matrices, etc.  
( Q-K-V projection weights, embeddings, MLP weights)

### ReduceScatter

After backward pass, gradient shards are reduced and scattered back across GPUs.

## Fully Sharded Data Parallel (FSDP)





# FSDP Communication

Collective	Role
AllGather	During forward and backward pass, before computing a layer, the full parameter shard is assembled across GPUs, so that each GPU can compute activations, attention matrices, etc. ( Q-K-V projection weights, embeddings, MLP weights)
ReduceScatter	After backward pass, gradient shards are reduced and scattered back across GPUs.

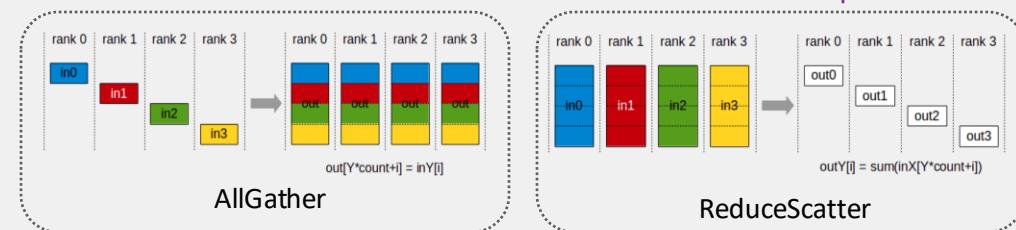
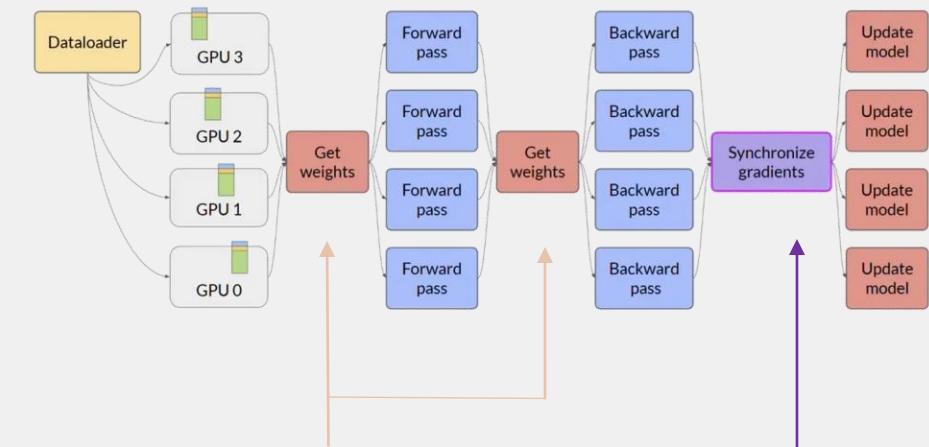
Trace Pattern:

AllGather → Forward GEMM → AllGather → Backward GEMM → ReduceScatter

FSDP compared to DDP:

- More frequent collectives
- Smaller collective sizes
- Higher sensitivity to latency
- More opportunity for communication-computation overlap

## Fully Sharded Data Parallel (FSDP)





# Deepspeed Communication - ZeRO1

Feature	ZeRO Stage 1
Parameter storage	Replicated
Gradient storage	Replicated
Optimizer state	Sharded
Forward communication	None
Backward communication	AllReduce
Optimizer communication	Minimal
Memory efficiency	Medium
Communication frequency	Low

Trace Pattern:

ZeRO1: Backward GEMM → AllReduce (large buckets) [Similar to DDP]

# Deepspeed Communication - ZeRO2

Feature	ZeRO Stage 1	ZeRO Stage 2
Parameter storage	Replicated	Replicated
Gradient storage	Replicated	Sharded
Optimizer state	Sharded	Sharded
Forward communication	None	None
Backward communication	AllReduce	ReduceScatter
Optimizer communication	Minimal	Minimal
Memory efficiency	Medium	High
Communication frequency	Low	Medium

Trace Pattern:

ZeRO1: Backward GEMM → AllReduce (large buckets) [Similar to DDP]

ZeRO2: Backward GEMM → ReduceScatter



# Deepspeed Communication - ZeRO3

Feature	ZeRO Stage 1	ZeRO Stage 2	ZeRO Stage 3
Parameter storage	Replicated	Replicated	Sharded
Gradient storage	Replicated	Sharded	Sharded
Optimizer state	Sharded	Sharded	Sharded
Forward communication	None	None	AllGather (per layer)
Backward communication	AllReduce	ReduceScatter	AllGather, ReduceScatter
Optimizer communication	Minimal	Minimal	Minimal
Memory efficiency	Medium	High	Very High
Communication frequency	Low	Medium	High

Trace Pattern:

ZeRO1: Backward GEMM → AllReduce (large buckets) [Similar to DDP]

ZeRO2: Backward GEMM → ReduceScatter

ZeRO3: AllGather → Forward GEMM → AllGather → Backward GEMM → ReduceScatter [Similar to FSDP]

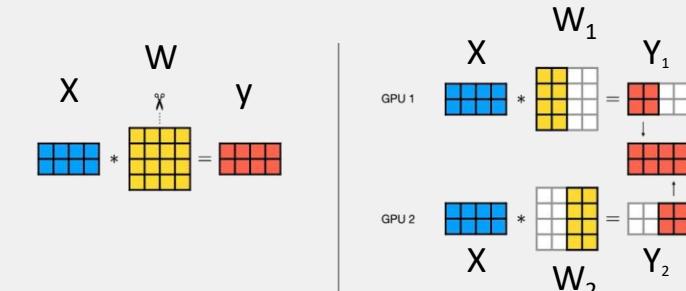
# MegatronLM Communication

Parallelism Type	Collective Used	Forward	Backward
Data Parallel	AllReduce	-	After backward on weight updates
Pipeline Parallel	Send / Recv (P2P)	Transfer activations (outputs) between pipeline stages	Transfer gradients between pipeline stages
Tensor Parallel	-	ColumnParallelLinear -> RowParallelLinear	-
	AllGather	ColumnParallelLinear -> ColumnParallelLinear (rare)	-
	AllReduce	RowParallelLinear -> ColumnParallelLinear	ColumnParallelLinear layer input gradients only * ( all other gradients are computed locally on each rank)

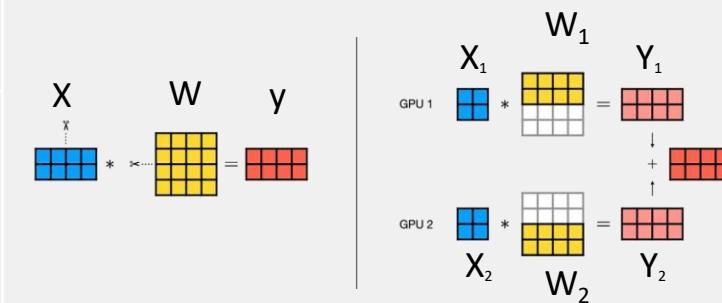
\*  $dX = dY W^T$ ,  $W = [W_1; W_2; \dots; W_{TP-ranks}]$ ,  $Y = [Y_1, Y_2, \dots, Y_{TP-ranks}]$

$$\text{Linear Layer: } Y = W^T X + b$$

Column Parallel Linear



Row Parallel Linear





# Agenda

## 1. Introduction

- Systems' Topology
- GPU communication (NCCL)

## 2. NVIDIA Nsight Systems

- **Nsys command**
- **Nsight client**

## 3. Paraver

- Nsys2prv
- Paraver client

## 4. Exercises



# Nvidia Nsight Systems - Terminology

**Kernel:** function executed on the GPU.

- ampere\_bf16\_gemm → matrix multiplication
- attn\_fwd\_kernel → attention
- ncclDevKernl\_AllReduce → communication

**CUDA API:** CPU-GPU interface, set of functions that allow our program to launch kernels, allocate GPU memory, copy data, synchronize execution, etc.

- cudaMalloc, cudaMemcpy, cudaStreamSynchronize, etc.

**NVTX:** lightweight annotation API to create custom markers and labels in the code. Used in torch.cuda.nvtx.

- "Batch X", "forward", "backward", "data loading X"

**Stream Processors (SM):** A compute unit inside the GPU. The A100 has 108 SMs.

**GPU Thread:** Smallest execution unit

**Warp:** Group of GPU threads that execute together (commonly 32)

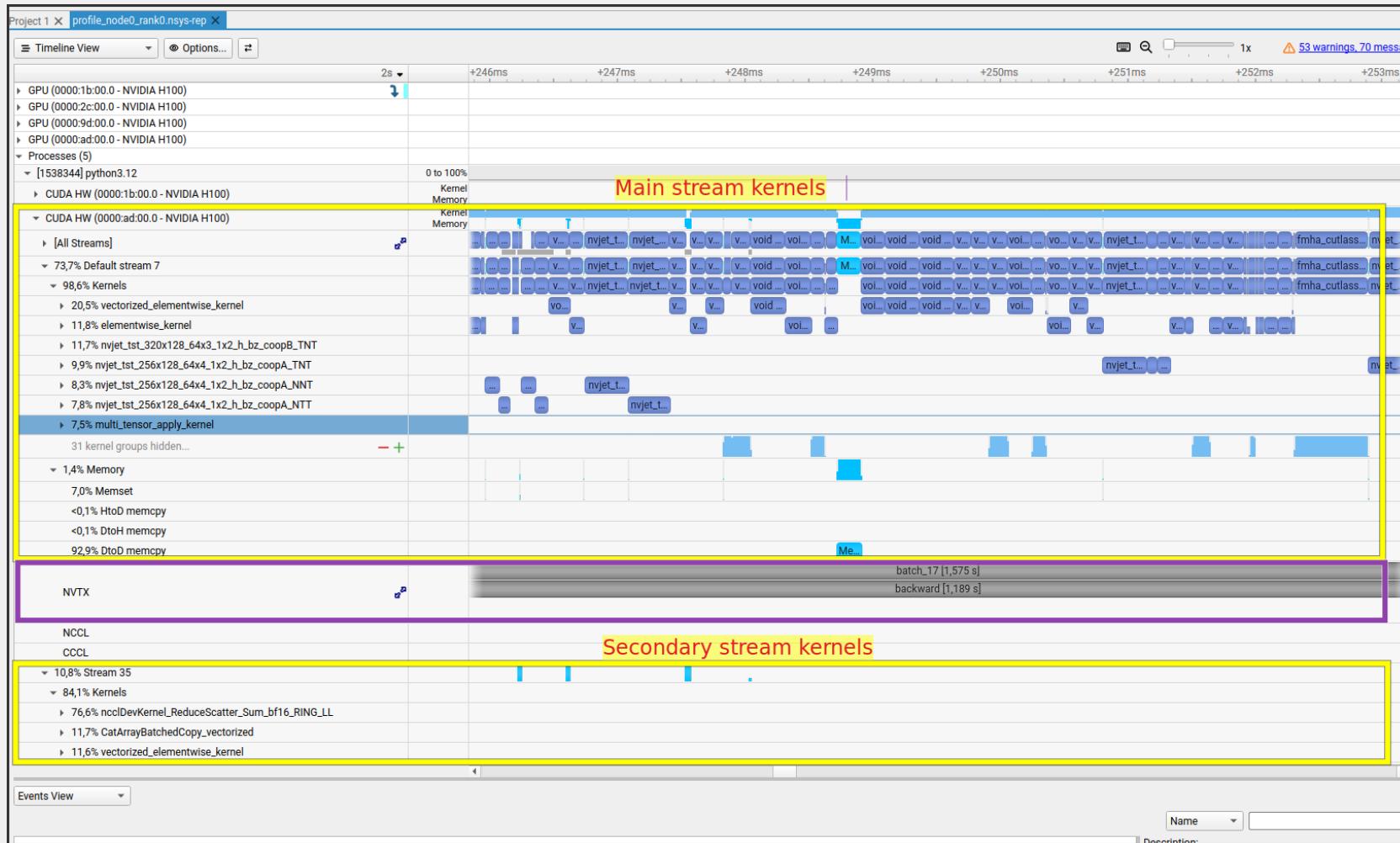
- **SM Active %:** Percentage of time SMs are doing work.
- **SM Instructions Tensor Active:** Tensor core activity for accelerate tensor multiplication
- **Warp Occupancy:** How many warps are ready to run per SM
- **(CUDA) Stream:** Execution queue



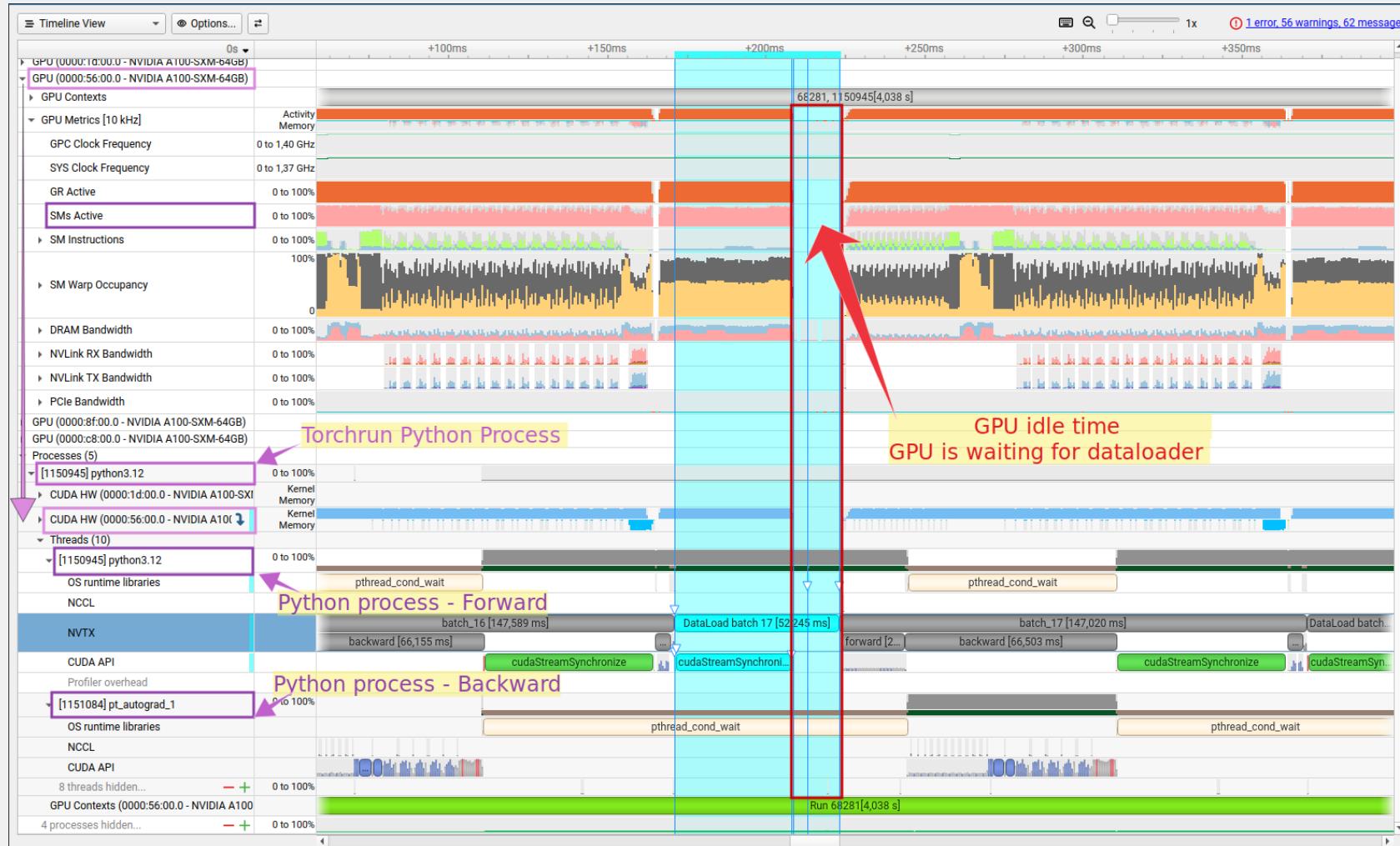
# Nvidia Nsight Systems - Events View



# Nvidia Nsight Systems - Events View



# Nvidia Nsight Systems – Idle time





# Nvidia Nsight Systems – Stats View

# CUDA API Summary

## NVTX Push/Pop Range Summary

CUDA GPU Kernel Summary



# Nvidia Nsight Systems – nsys command

## 'nsys' command

```
[apalior@login05 ~]$ module purge
[apalior@login05 ~]$ module load cuda/12.6
[apalior@login05 ~]$ which nsys
/leonardo/prod/opt/compilers/cuda/12.6/none/bin/nsys
[apalior@login05 ~]$ nsys
usage: nsys [l--version] [--help] <command> [<args>] [application] [<application args>]

The most commonly used nsys commands are:
  profile      Run an application and capture its profile into a nsys-rep file.
  launch       Launch an application ready to be profiled.
  start        Start a profiling session.
  stop         Stop a profiling session and capture its profile into a nsys-rep file.
  cancel       Cancel a profiling session and discard any collected data.
  service      Launch the Nsight Systems data service.
  stats        Generate statistics from an existing nsys-rep or SQLite file.
  status       Provide current status of CLI or the collection environment.
  shutdown     Disconnect launched processes from the profiler and shutdown the profiler.
  sessions list List active sessions.
  export       Export nsys-rep file into another format.
  analyze     Identify optimization opportunities in a nsys-rep or SQLite file.
  recipe      Run a recipe for multi-node analysis.
  nvprof      Translate nvprof switches to nsys switches and execute collection.

Use 'nsys --help <command>' for more information about a specific command.

To run a basic profiling session:  nsys profile ./my-application
For more details see "Profiling from the CLI" at https://docs.nvidia.com/nsight-systems
```

## 'nsys profile' example

```
nsys profile
  -trace=cuda,nvtx,osrt,cudnn,ucx
  -cuda-memory-usage=true
  -gpuctxsw=true
  -gpu-metrics-devices=all
  -gpu-metrics-frequency=10000
  -capture-range=cudaProfilerApi
  -capture-range-end=stop
  -cudabacktrace=kernel
  -stats=true
  -force-overwrite=true
  -output=/exercise_2_DeepSpeed/profiler/Mistral-7B-v0.1-34563983-n1-q4-mbs4-gas16-mixed1-actckpt0-hpz4-ZeR03-badcomm1/nsys/profile_node%Q{SLURM_NODEID}
singularity exec
  -n
  --bind /leonardo
  --bind /Leonardo
  /leonardo_work/tra26_minwinc/containers/ai-profiling-workshop.sif
    accelerate launch
      --config-file ./exercise_2_DeepSpeed/accelerate_config.yaml-H80Gt1r6yDfrb
      --rdzy-backend=c10d
      --machine_rank 0
      -m exercise_2_DeepSpeed.train
      --data-path /leonardo_work/tra26_minwinc/DATA/alpaca-cleaned/alpaca_data_cleaned.json
      --model-path /leonardo_work/tra26_minwinc/models/Mistral-7B-v0.1
      --epochs 1
      --no-validation
      --profile
      --data-sample 5000
      --dataloader-num-workers 8
      --batch-size 4
      --gradient-accumulation-steps 16
      -deepspeed-config ./exercise_2_DeepSpeed/ds_configs/stage-3/ds_config_mixed-precision_no-activation-checkpointing_comm-overhead.json
```

## 'nsys profile' sub-command

```
[apalior@login05 ~]$ nsys profile --help

usage: nsys profile [<args>] [application] [<application args>]

# =====#
# NSYS Configuration
# =====#
# Recommended NSYS options for ML training profiling:
#   - --trace=cuda,nvtx,osrt,cudnn,ucx,cublas : Trace GPU kernels, NVTX markers, OS runtime, cuDNN, UCX, NCCL, cuBLAS
#   - --force-overwrite true                  : Overwrite existing profile files
#   - --cuda-memory-usage=true               : Track CUDA memory allocations and usage
#   - --gpuctxsw=true                       : Track GPU context switches (optional, rarely a bottleneck)
#   - --gpu-metrics-devices=all              : Collect GPU metrics (SM utilization, memory throughput, etc.)
#   - --gpu-metrics-frequency=10000          : Sample GPU metrics at 10KHz for fine granularity
#   - --capture-range=cudaProfilerApi        : Use cudaProfiler start/stop for precise capture (requires NVTX markers in code)
#   - --capture-range-end=stop              : End capture when cudaProfilerStop is called
#   - --sample=cpu                          : CPU sampling for host-side bottlenecks
#   - --backtrace=dwarf                    : Detailed backtraces for CPU samples
#   - --cudabacktrace=kernel               : Collect backtraces for CUDA kernel launches
#   - --stats=true                         : Generate summary statistics
#   - --export=sqlite                      : Export results in SQLite format for advanced analysis
#   - --output=<path>                     : Set output file path (already used)
# =====#
```

"nsys profile" options

Application to profile

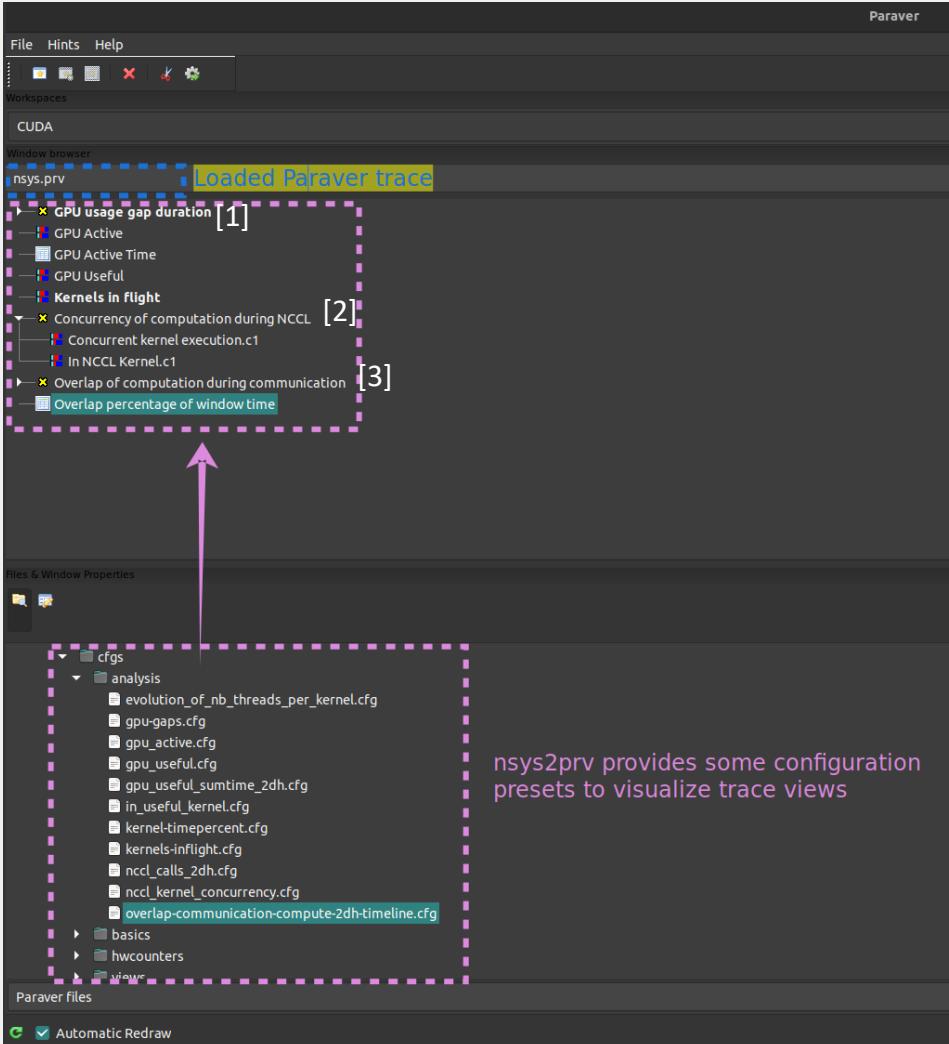


# Agenda

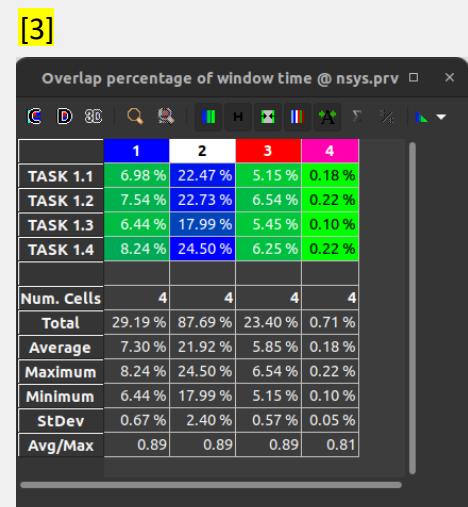
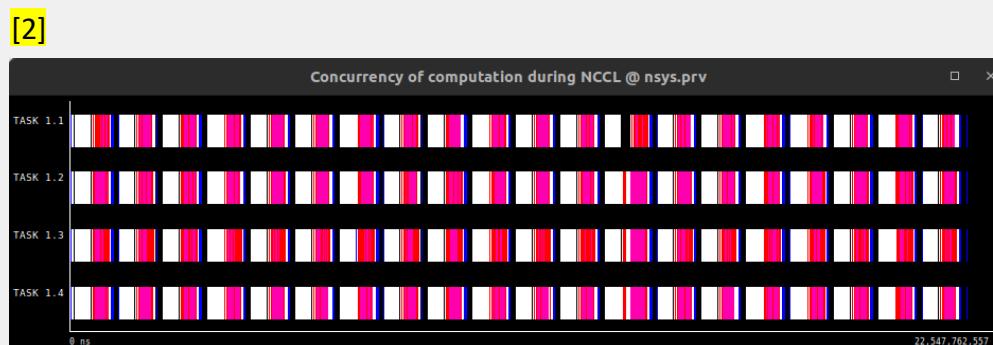
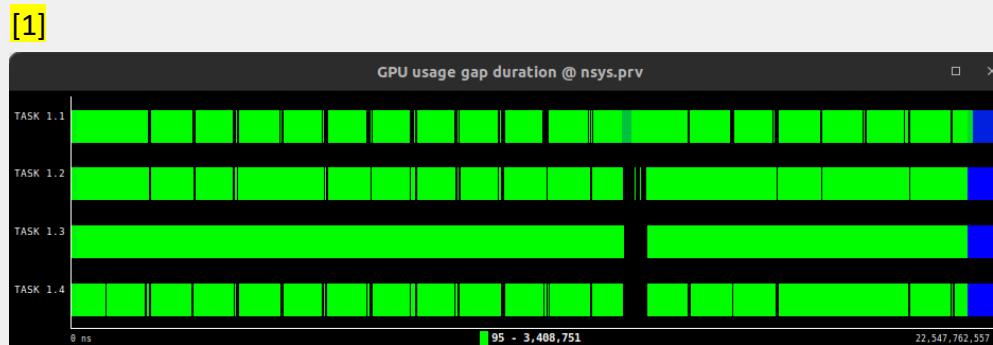
1. Introduction
  - NCCL
  - Systems' Topology
2. NVIDIA Nsight Systems
  - Nsys command
  - Nsight client
3. Paraver
  - **Nsys2prv**
  - **Paraver client**
4. Demo
5. Hands-on Exercises



# Paraver



nsys2prv provides some configuration  
presets to visualize trace views





# nsys2prv – Trace translation

The screenshot shows the GitHub project page for nsys2prv 0.5.8. The page includes a search bar, navigation links (Help, Docs, Sponsors, Log in, Register), and a release section indicating it's the latest version (released Jul 2, 2025). A pip install button is also present. The main content area contains a brief description: "Translate a NVIDIA Nsight System trace to a Paraver trace". The left sidebar has sections for Navigation (Project description, Release history, Download files), Verified details (PyPI verified), Owner (Group for Best Practices for Performance and Programmability of the Barcelona Supercomputing Center), Unverified details (not yet verified by PyPI), Project links (Homepage, Repository), and Meta (License: GNU General Public License v3 (GPLv3) (GPL-3.0-only), Author: Marc Clascà, Requirements: Python >=4.0, >=3.10).

- Python cli tool developed by BSC.
- Translate Nvidia NSYS traces to Paraver traces.

```
> nsys2prv --help
nsys2prv v0.5.8
usage: nsys2prv [-h] [-v] [-f FILTER_NVTX] [-t TRACE] [-m] [--force-sqlite] [-s] [-z] source_rep [source_rep ...] output

Translate a NVIDIA Nsight System trace to a Paraver trace

positional arguments:
  source_rep          Nsight source report file
  output              Paraver output trace name

options:
  -h, --help           show this help message and exit
  -v, --version        Show version and exit. (default: None)
  -f FILTER_NVTX, --filter-nvtx FILTER_NVTX
                      Filter by this NVTX range (default: None)
  -t TRACE, --trace TRACE
                      Comma separated names of events to translate: [mpi_event_trace, nvtx_pushpop_trace, nvtx_startend_trace, cuda_api_trace, gpu_metrics, openacc, nccl, graphs] (default: None)
  -m, --multi-report   Translate multiple reports of the same execution into one trace. (default: False)
  --force-sqlite        Force Nsight System to export SQLite database (default: False)
  -s, --sort            Sort trace at the end (default: False)
  -z, --compress        Compress trace at the end with gzip (default: False)

The nsys executable needs to be in the PATH, or the environment variable NSYS_HOME needs to be set. If using postprocessing, the PARAVER_HOME variable needs to be set.
```



# Agenda

## 1. Introduction

- Systems' Topology
- GPU communication (NCCL)

## 2. NVIDIA Nsight Systems

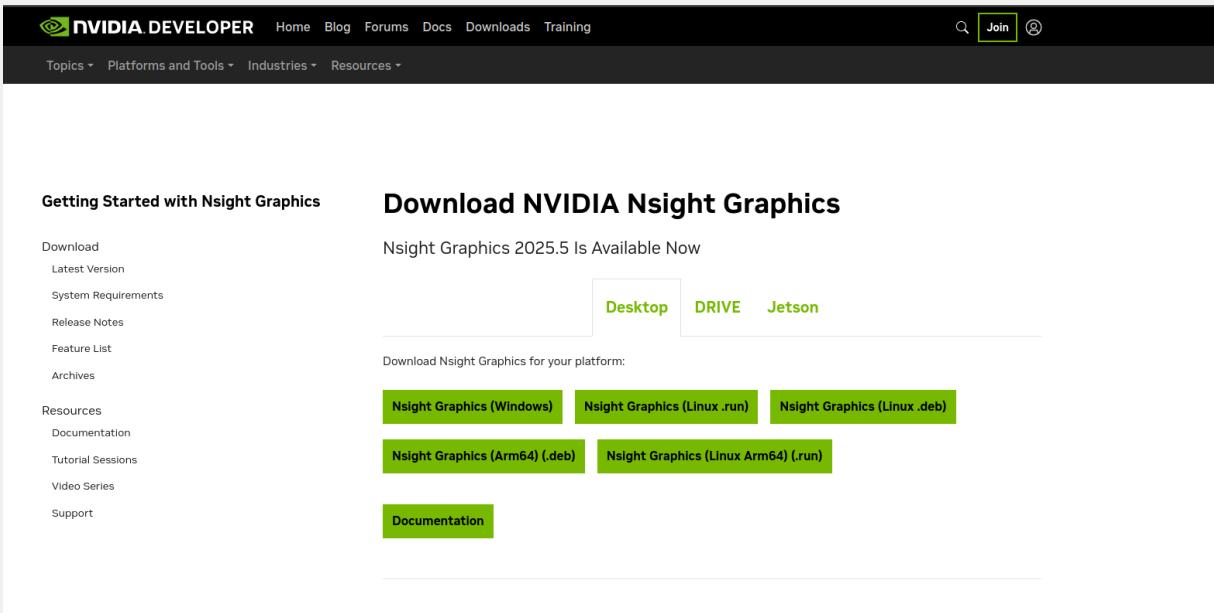
- Nsys command
- Nsight client

## 3. Paraver

- Nsys2prv
- Paraver client

## 4. Exercises

# Download and setup Nsight



The screenshot shows the 'NVIDIA DEVELOPER' website with a dark header. The top navigation bar includes links for Home, Blog, Forums, Docs, Downloads, Training, a search icon, a 'Join' button, and a user profile icon. Below the header, there are dropdown menus for Topics, Platforms and Tools, Industries, and Resources.

The main content area is titled 'Getting Started with Nsight Graphics' and features a large heading 'Download NVIDIA Nsight Graphics'. A sub-section below it says 'Nsight Graphics 2025.5 Is Available Now'. It includes tabs for Desktop, DRIVE, and Jetson. A section for 'Download Nsight Graphics for your platform:' lists several download links:

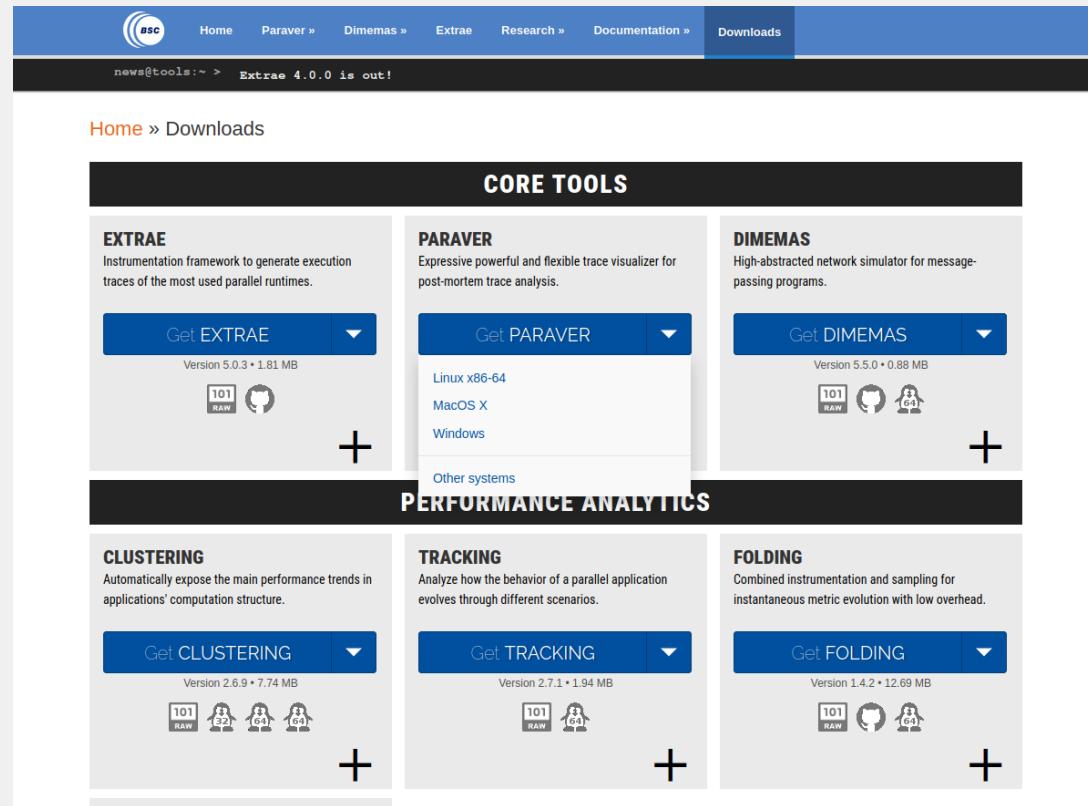
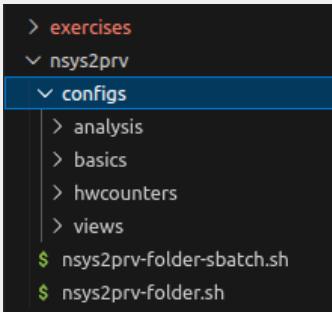
- Nsight Graphics (Windows)
- Nsight Graphics (Linux .run)
- Nsight Graphics (Linux .deb)
- Nsight Graphics (Arm64) (.deb)
- Nsight Graphics (Linux Arm64) (.run)

At the bottom of this section is a 'Documentation' link.

1. Download Nvidia Nsight Systems  
@ <https://developer.nvidia.com/nsight-graphics/get-started>
2. Install on your computer

# Download and setup Paraver

1. Download Paraver  
@ <https://tools.bsc.es/downloads>
2. Install on your computer
3. On Linux/MacOS
  - o [Optional] Move folder to ~/Applications
  - o Run "bin/wxparaver.bin" on terminal
4. On Windows
  - o Run bin/wxparaver.bin.exe
5. Download nsys2prv configuration files stored inside the workshop directory:
  - o sys2prv/configs



A screenshot of the BSC Tools Downloads page. The 'Paraver' section is highlighted. It shows download links for 'EXTRAE', 'PARAVER', and 'DIMEMAS' across various operating systems (Linux x86-64, Mac OS X, Windows, Other systems). Below this, sections for 'CLUSTERING', 'TRACKING', and 'FOLDING' are shown, each with their own download links and system requirements. A 'BASIC ANALYSIS' section is also present.

# Thank you



Co-funded by  
the European Union



**AI4S**  
BSC AI 4 Science Fellowships



This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101182737. The JU receives support from the Digital Europe Programme.