PÁRHUZAMOS AGYI BIOLEKTROMOS JELFELDOLGOZÁS MULTI-GPU KÖRNYEZETBEN A SLICES IMEC INFRASTRUKTÚRA TRANSZNACIONÁLIS HASZNÁLATÁVAL

PARALLEL MULTI-GPU BIOELECTRICAL SIGNAL PROCESSING BY THE TRANS-NATIONAL USE OF THE SLICES IMEC INFRASTRUCTURE

DR. ZOLTÁN JUHÁSZ
UNIVERSITY OF PANNONIA, VESZPRÉM

Dept. Electrical Engineering
and Information Systems

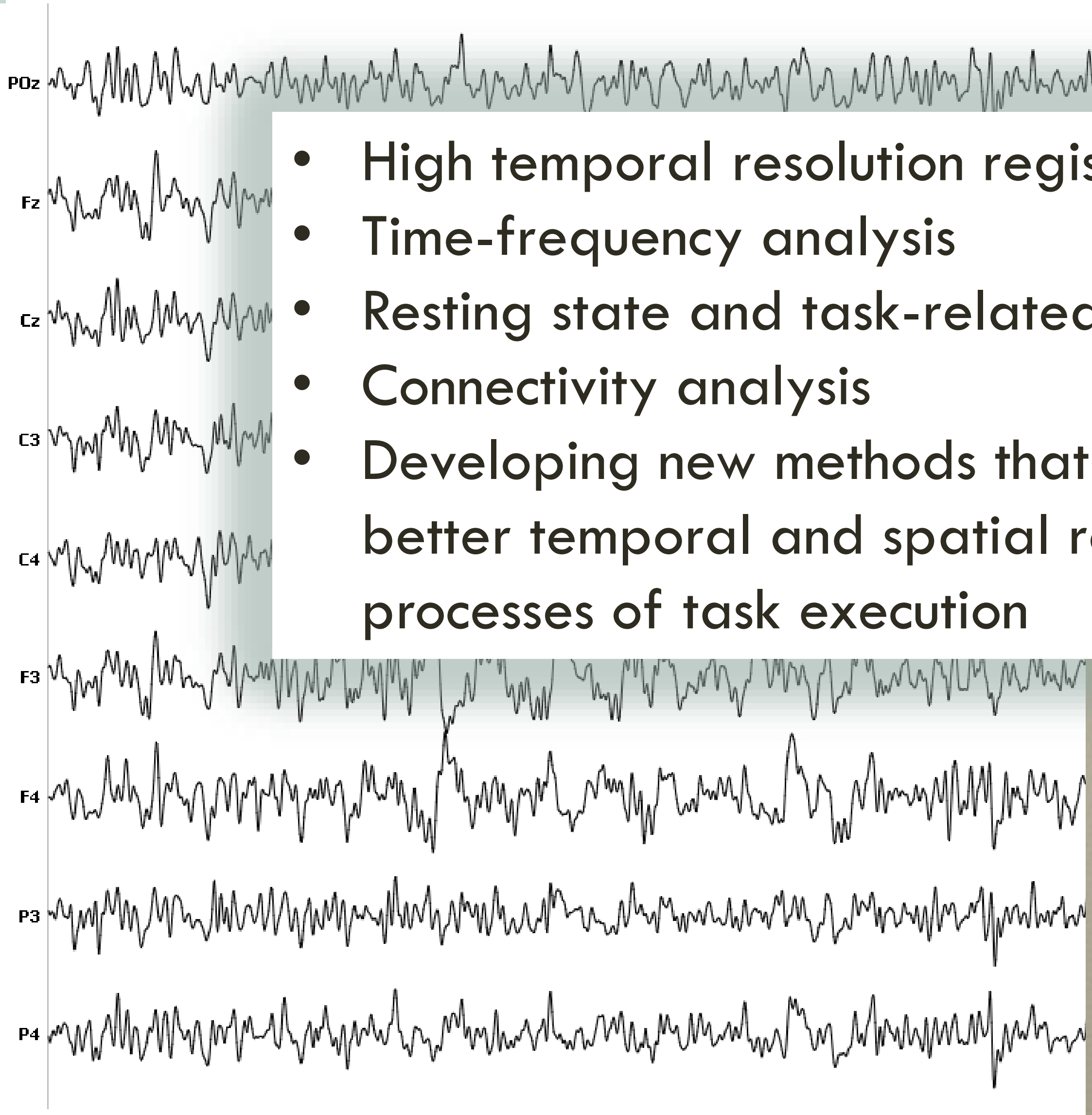
juhasz@virt.uni-pannon.hu

SETTING THE SCENE

Bioelectrical Brain Imaging Laboratory

1. What we do: parallel biomedical signal processing and analysis
2. Why we need high-performance computational resources
3. Problems we normally face: financial, administrative and technical
4. Why we chose the SLICES-SC resources
5. Benefits and results

BIOMEDICAL (EEG) SIGNAL PROCESSING



- High temporal resolution registration of cortical activity
- Time-frequency analysis
- Resting state and task-related analysis
- Connectivity analysis
- Developing new methods that provide new information, better temporal and spatial resolution, uncover sub-processes of task execution

NEED FOR HIGH-PERFORMANCE COMPUTATIONAL RESOURCES

- EEG research is mainly conducted using MATLAB scripts
- tendency to use **limited number of channels, low sampling frequency, short measurements,** to keep execution time within practical limits
- Most sophisticated experiments and analysis methods are excluded!
  - 2k samples/channel/second $\rightarrow$ 512k samples for 256 channels/sec $\rightarrow$ 10^9 samples for 30 min measurement
  - at 1k ops per sample $\rightarrow$ 10^{12} ops (1 Top/s)
- MATLAB execution times are in the order of **hours per subject**
- limited parallelism support, mainly clusters

Research goal:

- Massively parallel GPU algorithms to reduce time from hours/days to seconds
- single-GPU and multi-GPU systems, price/performance ratio

TYPICAL HPC-RELATED DIFFICULTIES

Options for increasing computational capacity:

- in-house cluster (CPU and/or GPU) – very high upfront cost, only viable if used for 24/7 production use
- Cloud – cost, still limited HPC and GPU support, mainly single node architecture
- Supercomputers – fine if access is granted, high threshold to entry
  - current situation in Hungary: there is no suitable system available to us
  - existing systems (Leo) are outdated (GPU model no longer supported)
  - Komondor (A100 GPUs, NVLink!) is still in test mode
  - PRACE resources: possible, but mainly for production runs
- SLICES-SC project – fast access to resources for **experimentation** / **development**

EEG PROCESSING FUNDAMENTALS

Typical operations

- high-pass, low-pass filtering, mean removal, segmenting into trials
- trial averaging and quantification, statistical tests
- spectrum estimation (global or time-varying); FFT, wavelets
- artefact removal: Independent Component Analysis
- source localisation
- connectivity estimation, calculation of graph metrics

Typically performed for subject/patient populations of size 20-100

OPPORTUNITIES FOR PARALLEL EXECUTION

Fortunately, EEG processing lends itself to parallel execution

- subjects: “embarrassingly parallel”
- channels:
- samples:
- spatial ops: one time point but all electrodes
- algorithmic level:

High degree of parallelism, ideal for GPU execution

One GPU is a huge improvement: 200-2000x speedup over MATLAB

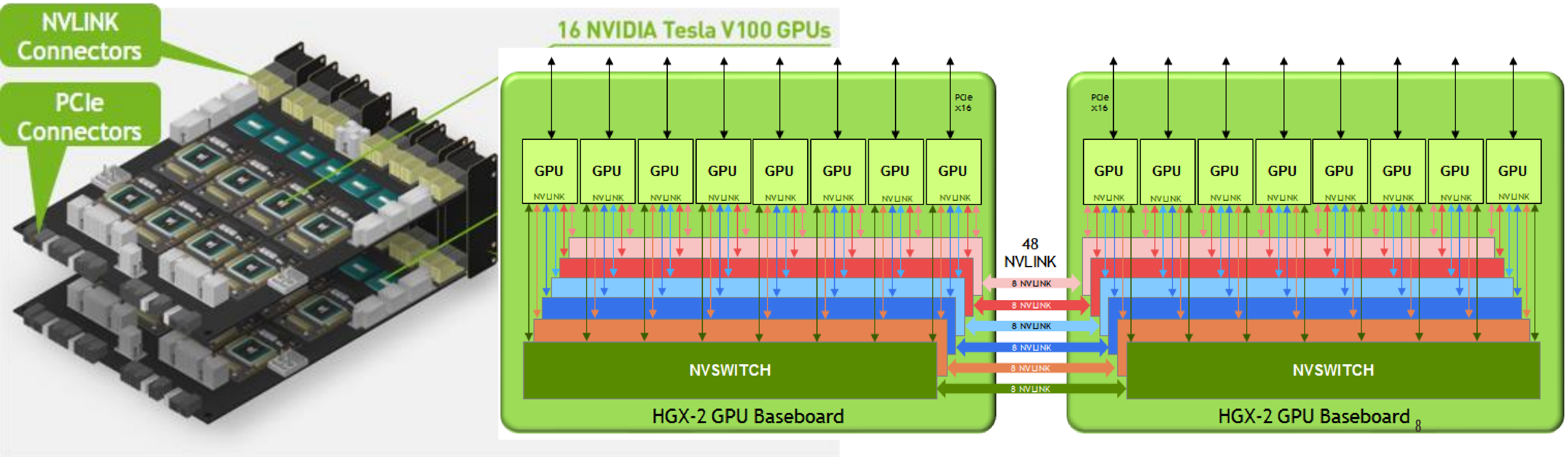
What about multiple GPUs? Can we utilise them effectively?

GPULAB RESOURCES

Several GPU clusters with different types of GPU cards

What we needed was Cluster 6:

- NVIDIA HGX-2 with **16 Tesla V100 GPUs,** 96 2.7Ghz vCPU cores with 1.5TB RAM
- NVLink switch connection fabric



ACCESS AND USAGE

We used the GPULab system both in interactive and CLI batch mode

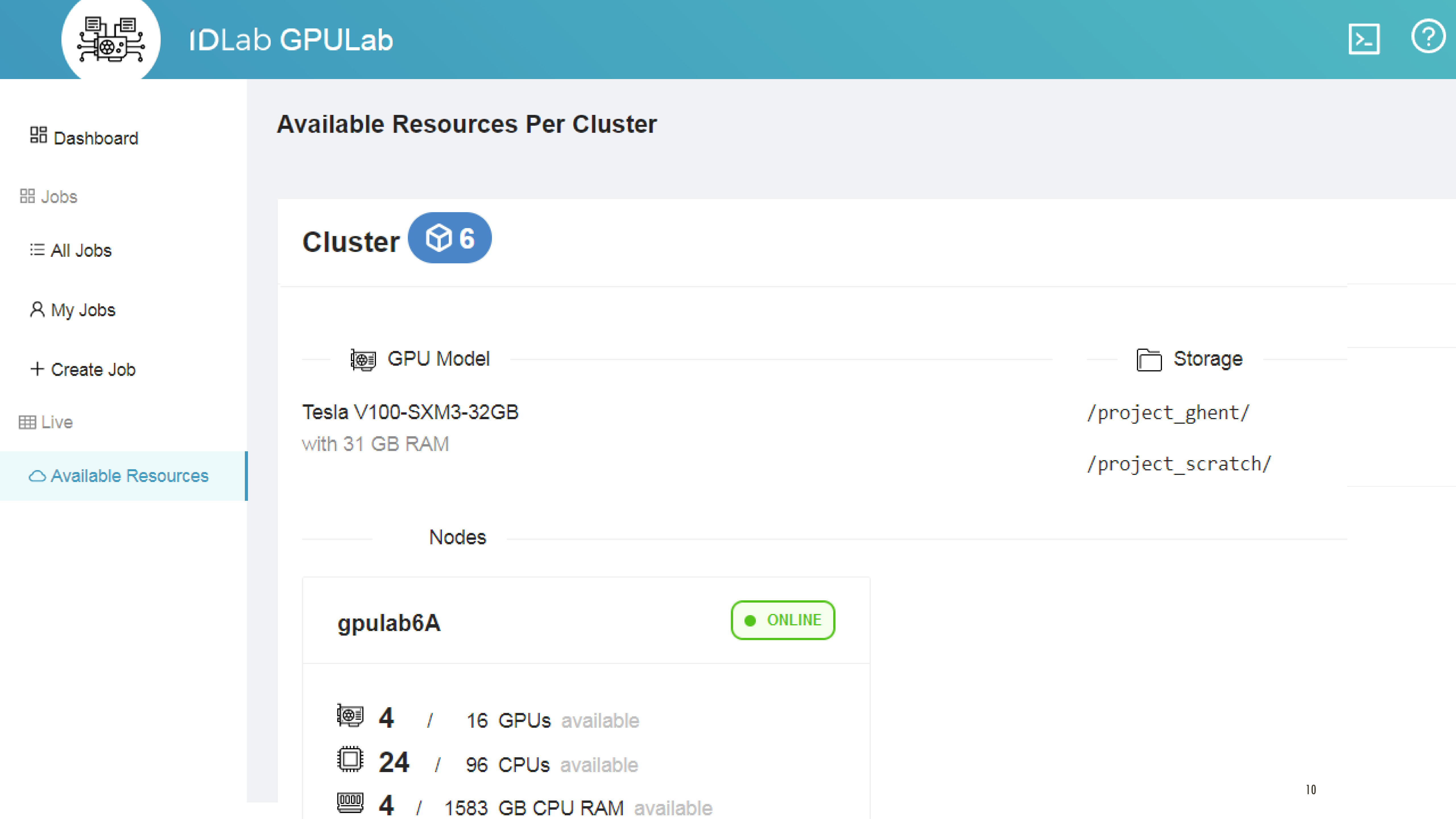
Interactive: mainly for checking resources, job status and uploading files -- JupyterHub

Batch: program execution using a batch execution system

The system is based on the Docker container system

Many predefined Docker images are available

We created our own image for HPC workloads



Available Resources Per Cluster

Cluster 6

GPU Model

Tesla V100-SXM3-32GB
with 31 GB RAM

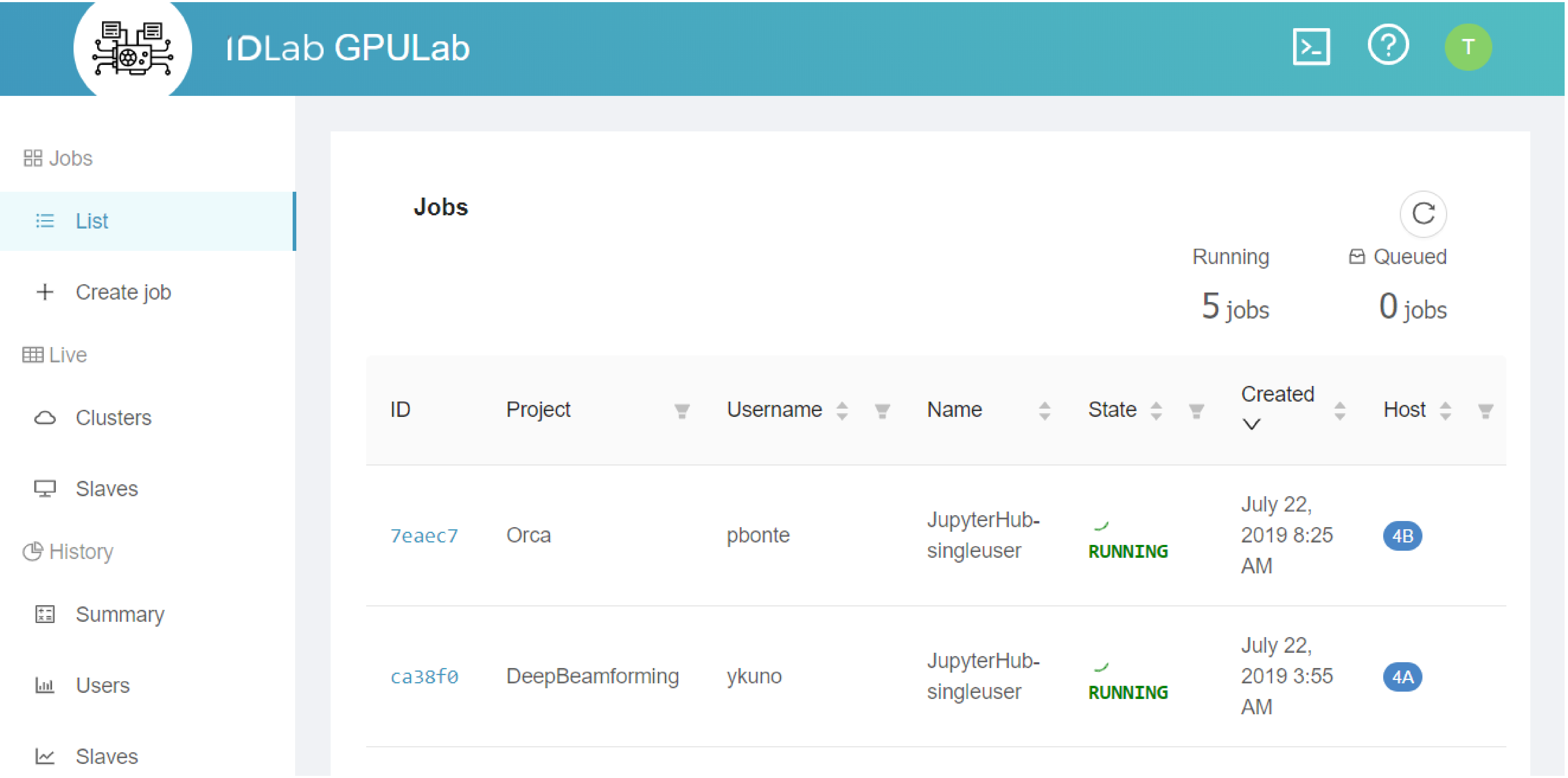
Storage

/project_ghent/
/project_scratch/

Nodes

gpulab6A ● ONLINE

4 / 16 GPUs available
24 / 96 CPUs available
4 / 1583 GB CPU RAM available

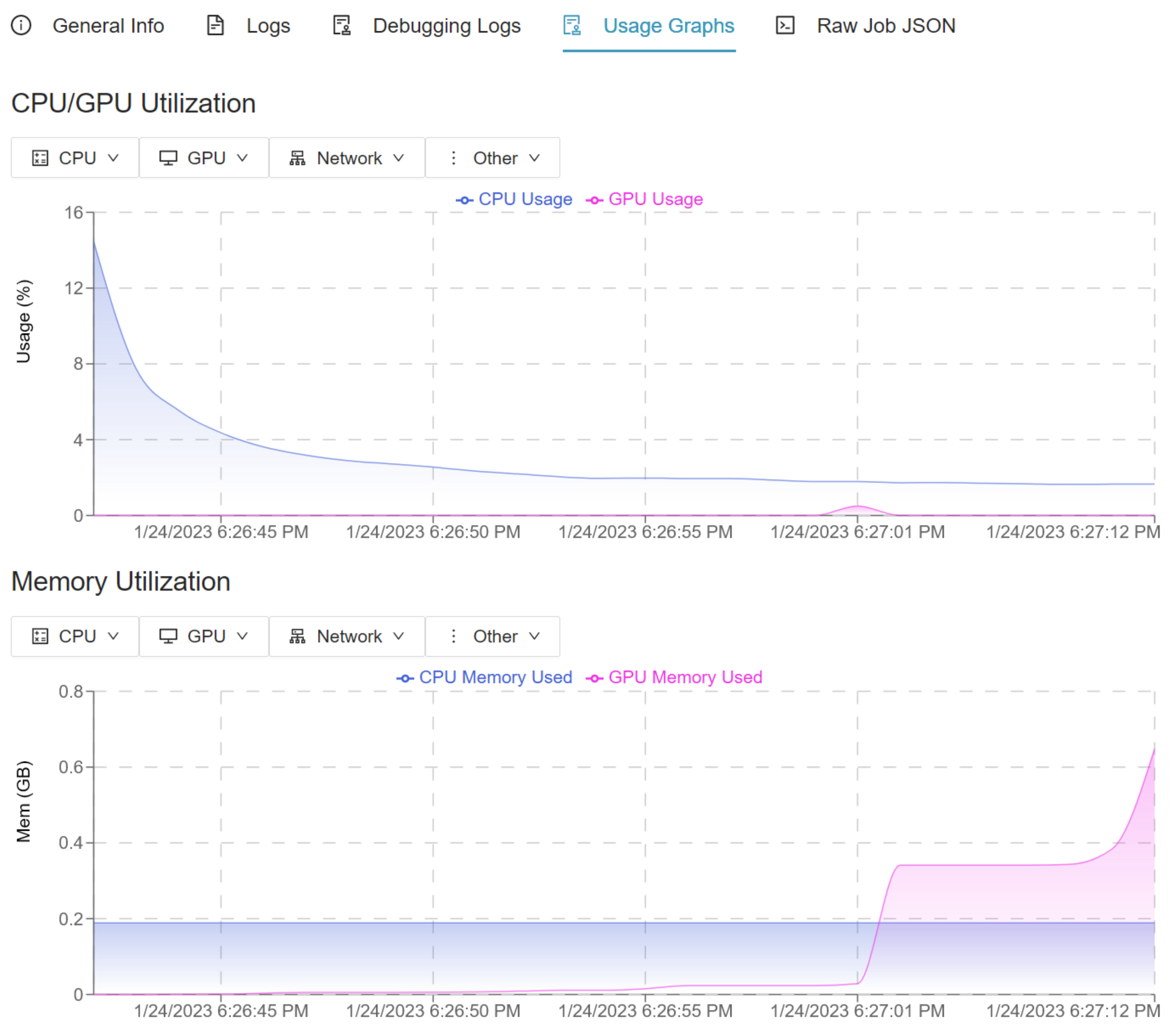


Jobs

Running 5 jobs

Queued 0 jobs

ID	Project	Username	Name	State	Created	Host
7eaec7	Orca	pbonte	JupyterHub-singleuser	RUNNING	July 22, 2019 8:25 AM	4B
ca38f0	DeepBeamforming	ykuno	JupyterHub-singleuser	RUNNING	July 22, 2019 3:55 AM	4A


General Info
Logs
Debugging Logs
Usage Graphs
Raw Job JSON
CPU/GPU Utilization
CPU
GPU
Network
Other
CPU Usage
GPU Usage
Usage (%)
16
12
8
4
0
1/24/2023 6:26:45 PM
1/24/2023 6:26:50 PM
1/24/2023 6:26:55 PM
1/24/2023 6:27:01 PM
1/24/2023 6:27:12 PM
Memory Utilization
CPU
GPU
Network
Other
CPU Memory Used
GPU Memory Used
Mem (GB)
0.8
0.6
0.4
0.2
0
1/24/2023 6:26:45 PM
1/24/2023 6:26:50 PM
1/24/2023 6:26:55 PM
1/24/2023 6:27:01 PM
1/24/2023 6:27:12 PM

Sample job definition file

```
{
  "name": "CUDA_MEMD",
  "description": "Test_CUDA_MEMD",
  "request": {
    "docker": {
      "image": "gitlab.ilabt.imec.be:4567/ilabt/gpu-docker-stacks/code-server-notebook:latest",
      "command": "/project_ghent/startScript.sh",
      "storage": [
        {
          "hostPath": "/project_ghent",
          "containerPath": "/project_ghent"
        }
      ]
    },
    "resources": {
      "clusterId": 6,
      "cpus": 4,
      "gpus": 2,
      "cpuMemoryGb": 16,
      "minCudaVersion": 11
    },
    "scheduling": {
          "maxDuration": "1 min",
      "interactive": false
    }
  }
}
```

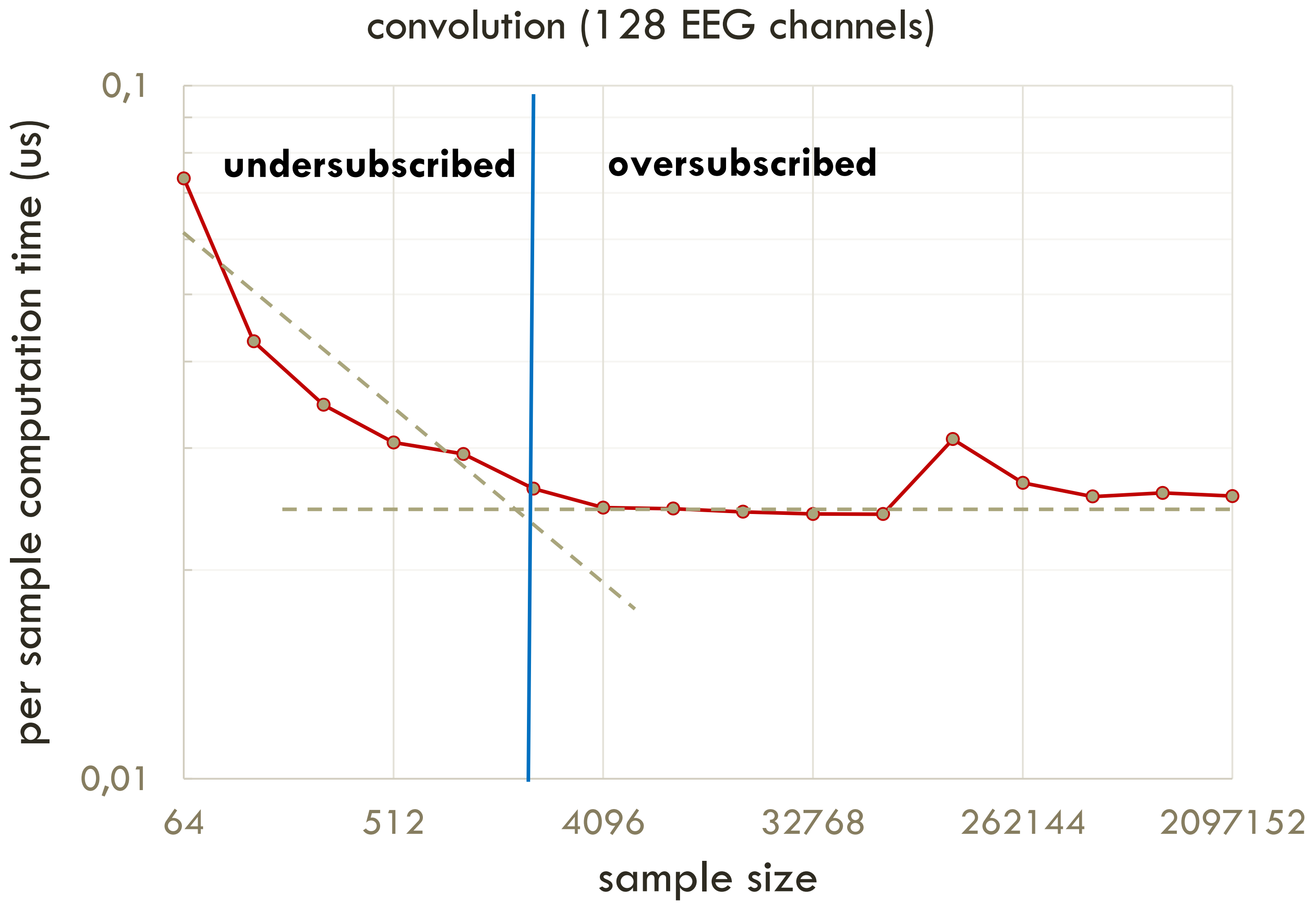
GPU EXECUTION PROPERTIES

GPU uses a fixed number of cores

throughput, not latency optimised

inefficient if there is not enough work

will not run faster if fully loaded



TRADITIONAL MULTI-GPU APPROACH

Distributed, message-passing programming model

1. Distribute data using MPI

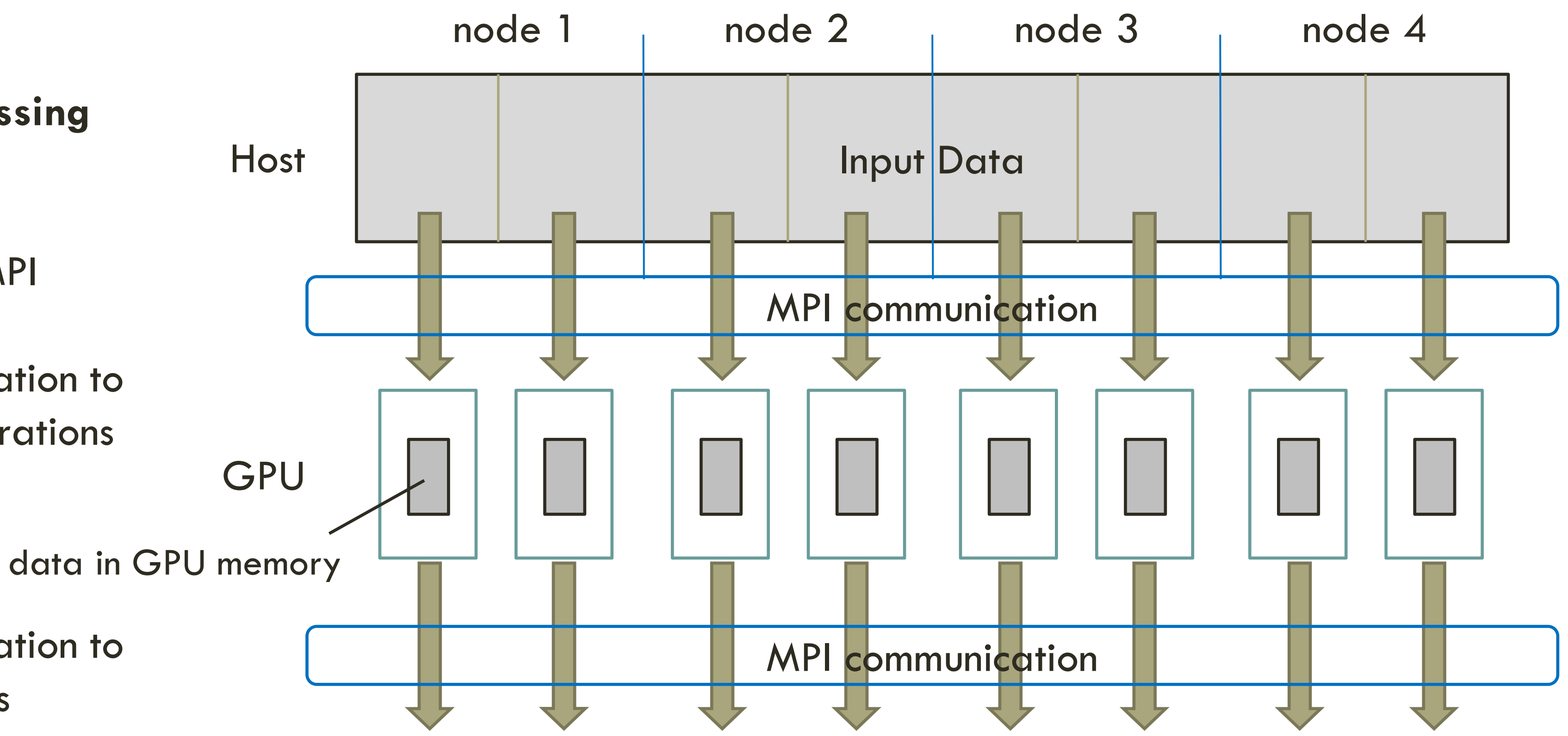
2. Perform MPI communication to collect data for GPU operations

3. Execute GPU code

4. Perform MPI communication to send results to other GPUs

5. If necessary, collect data on host and re-distribute results among nodes

…



DIRECT MULTI-GPU APPROACH

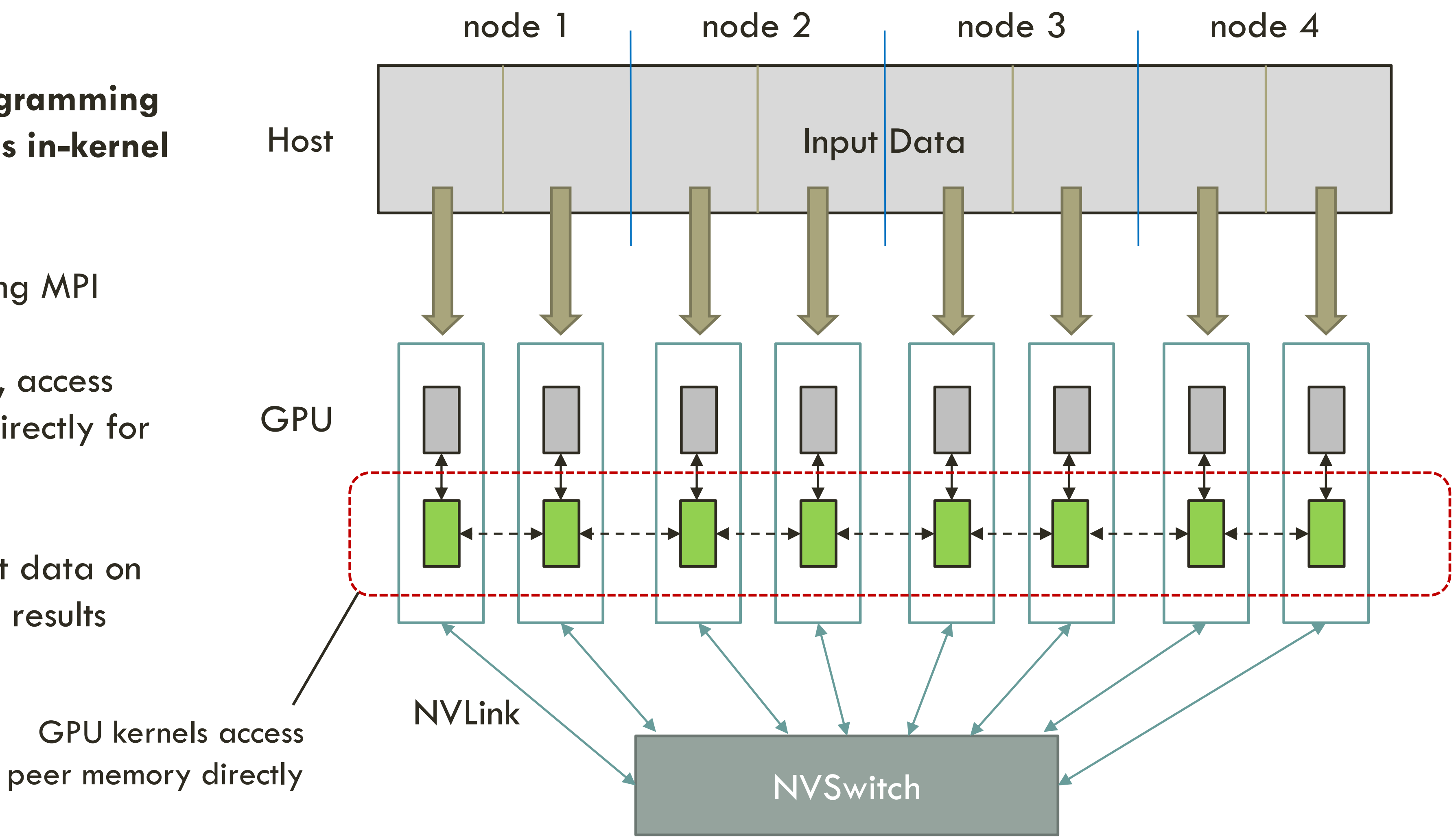
Shared-memory programming model, asynchronous in-kernel communication

1. Distribute data using MPI

2. Execute GPU code, access other GPU memory directly for data (load/store)

3. If necessary, collect data on host and re-distribute results among nodes

…



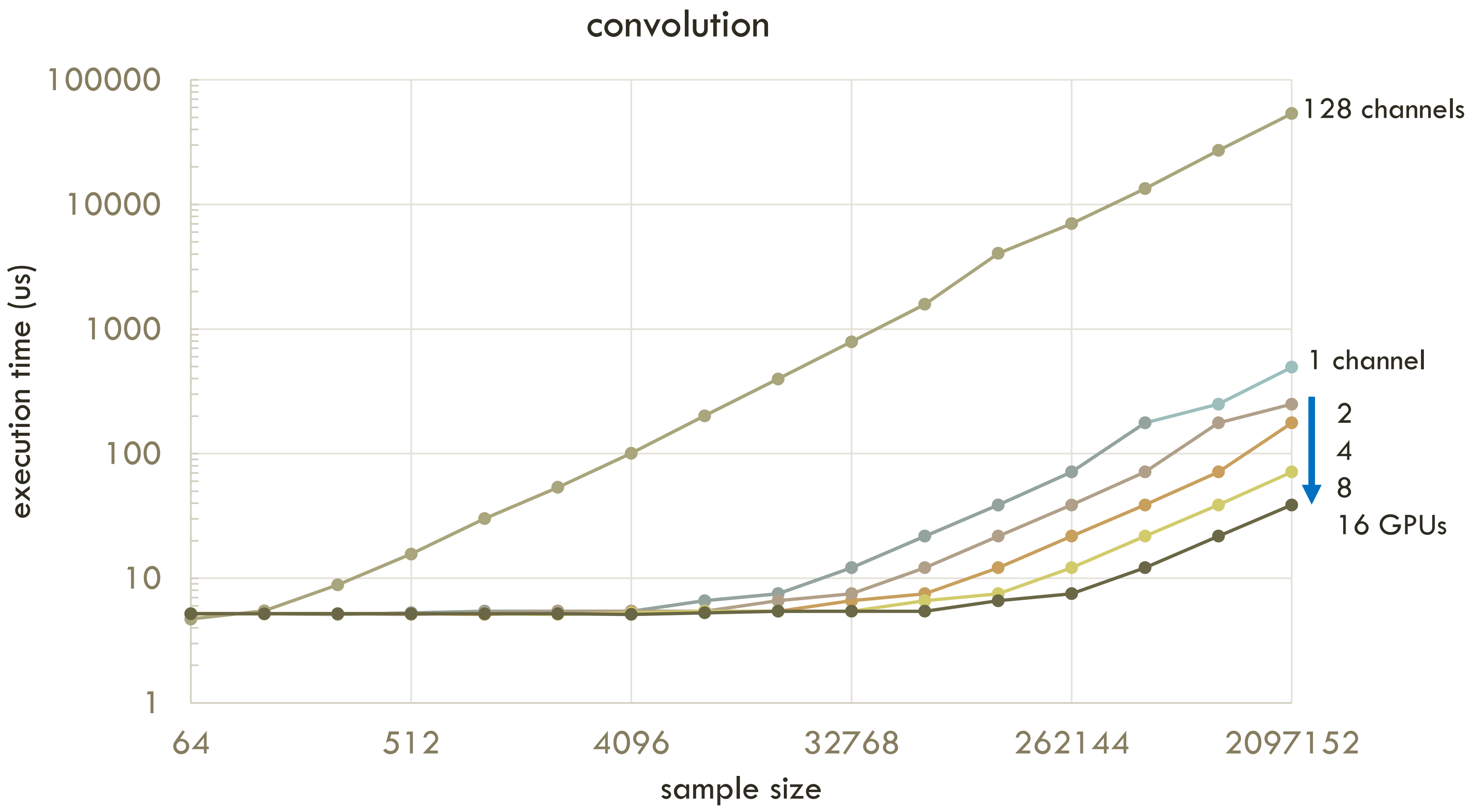
RESULTS

- Single-GPU algorithms tested on the V100 GPU

- simple multi-GPU test programs developed and executed

- multi-GPU EEG algorithms designed, some are tested for functionality

- full scale scalability and performance tests in the coming weeks



CONCLUSIONS

Several multi-GPU EEG processing algorithms have been developed

Full-scale (16 GPU) scalability test is yet to be done

The SLICES-SC project and the GPULab infrastructure gave us state-of-the-art resources for the development

Allowed us to progress with our work until other HPC resources become available