University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Uniwersytet Warszawski

Architecture of large projects in bioinformatics (ADP)

Łukasz P. Kozłowski

Warsaw, 2024

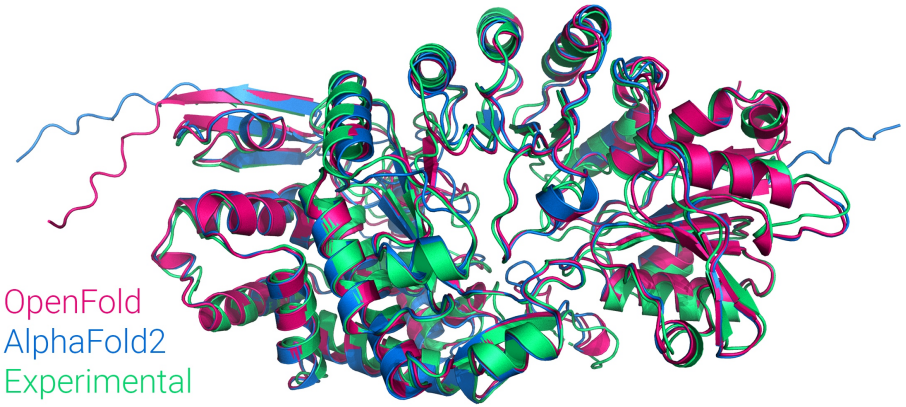
lukaskoz@mimuw.edu.pl

HPC

High-Performance Computing

OpenFold: Retraining AlphaFold2 yields new insights into its learning mechanisms and capacity for generalization

Gustaf Ahdritz[*1,2], Nazim Bouatta[*3], Christina Floristean[1], Sachin Kadyan[1], Qinghui Xia[1], William Gerecke[3], Timothy J O'Donnell[4], Daniel Berenberg[5,6], Ian Fisk[7], Niccolò Zanichelli[8], Bo Zhang[9], Arkadiusz Nowaczynski[10], Bei Wang[10], Marta M Stepniewska-Dziubinska[10], Shang Zhang[10], Adegoke Ojewole[10], Murat Efe Guney[10], Stella Biderman[11,12], Andrew M Watkins[5], Stephen Ra[5], Pablo Ribalta Lorenzo[10], Lucas Nivon[13], Brian Weitzner[14], Yih-En Andrew Ban[15], Peter K Sorger[3], Emad Mostaque[16], Zhao Zhang[17], Richard Bonneau[5], and Mohammed AlQuraishi[1]



"total training time is approximately 50,000 GPU hours"

With 128 NVIDIA A100 GPUs takes over 8 days, 27% faster than AlphaFold2

Model has ~93M parameter

Must read:

https://www.biorxiv.org/content/10.1101/2022.11.20.517210v3.full.pdf

Must watch: OpenFold

https://www.youtube.com/watch?v=EnKqDD8fSZY

RESEARCH ARTICLE | STRUCTURE PREDICTION

Evolutionary-scale prediction of atomic-level protein structure with a language model

ZEMING LIN, HALIL AKIN, ROSHAN RAO, BRIAN HIE, ZHONGKAI ZHU, WENTING LU, NIKITA SMETANIN, ROBERT VERKUIL, ORI KABELI, [...], AND ALEXANDER RIVES +5 authors Authors Info & Affiliations

SCIENCE · 16 Mar 2023 · Vol 379, Issue 6637 · pp. 1123-1130 · DOI: 10.1126/science.ade2574

CURRENT ISSUE

"we scale language models from 8 million parameters up to 15 billion parameters"

an improvement in speed of up to 60×

Folding 617 million sequences from Mgnify

~28,000 GPU days on V100 GPU cards

ESM Atlas
Metagenomics
Protein Dataset

Meta AI

FastFold: Optimizing AlphaFold Training and Inference on GPU Clusters

Shenggan Cheng*
National University of Singapore
shenggan@comp.nus.edu.sg

Xuanlei Zhao
HPC-AI Tech
xlzhao@luchentech.com

Guangyang Lu
HPC-AI Tech
lugy@hpcaitech.com

Jiarui Fang
HPC-AI Tech
fangjr@luchentech.com

Tian Zheng
Xi'an Jiaotong University
zt12389@stu.xjtu.edu.cn

Ruidong Wu
HeliXon
ruidong@helixon.com

Xiwen Zhang
HeliXon
xiwen@helixon.com

Jian Peng
HeliXon
jianpeng@helixon.com

Yang You
National University of Singapore
youy@comp.nus.edu.sg

FastFold can efficiently scale to more GPUs using DAP and reduces overall training time from 11 days to 67 hours and achieves 7.5 ~ 9.5× speedup for long-sequence inference.

Implementation	Framework	Training Process	Hardware	Step Time	Training Time	Resource
AlphaFold	JAX[12]	Initial training Fine-tuning	128 × TPUv3	/ /	11 days	33792 TPU hours
OpenFold	PyTorch	Initial training Fine-tuning	128 × A100	6.186 s 20.657 s	8.39 days	25774 GPU hours
FastFold	PyTorch	Initial training Fine-tuning	128 × A100	3.869 s 11.832 s	5.10 days	**15679 GPU hours**
FastFold-DAP	PyTorch	Initial training Fine-tuning	256 × A100 512 × A100	2.487 s 4.153 s	**2.81** days	20738 GPU hours

ScaleFold: Reducing AlphaFold Initial Training Time to 10 Hours

Feiwen Zhu*
NVIDIA
Shanghai, China
mzhu@nvidia.com

Arkadiusz Nowaczynski*
NVIDIA
Warsaw, Poland
anowaczynski@nvidia.com

Rundong Li
NVIDIA
Shanghai, China
davidli@nvidia.com

Jie Xin
NVIDIA
Shanghai, China
jxin@nvidia.com

Yifei Song
NVIDIA
Shanghai, China
yifeis@nvidia.com

Michal Marcinkiewicz
NVIDIA
Warsaw, Poland
michalm@nvidia.com

Sukru Burc Eryilmaz
NVIDIA
Santa Clara, United States
seryilmaz@nvidia.com

Jun Yang
NVIDIA
Beijing, China
juney@nvidia.com

Michael Andersch
NVIDIA
Berlin, Germany
mandersch@nvidia.com

ScaleFold: Reducing AlphaFold Initial Training Time to 10 Hours

Feiwen Zhu*
NVIDIA
Shanghai, China
mzhu@nvidia.com

Arkadiusz Nowaczynski*
NVIDIA
Warsaw, Poland
anowaczynski@nvidia.com

Rundong Li
NVIDIA
Shanghai, China
davidli@nvidia.com

Jie Xin
NVIDIA
Shanghai, China
jxin@nvidia.com

Yifei Song
NVIDIA
Shanghai, China
yifeis@nvidia.com

Michal Marcinkiewicz
NVIDIA
Warsaw, Poland
michalm@nvidia.com

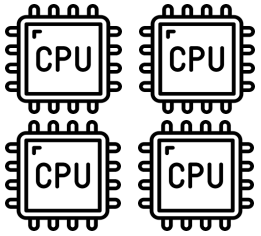
Sukru Burc Eryilmaz
NVIDIA
Santa Clara, United States
seryilmaz@nvidia.com

Jun Yang
NVIDIA
Beijing, China
juney@nvidia.com

Michael Andersch
NVIDIA
Berlin, Germany
mandersch@nvidia.com

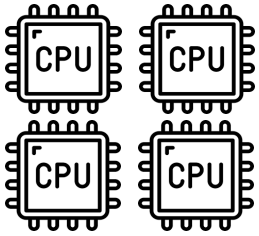
ScaleFold successfully scaled the AlphaFold training to **2080 NVIDIA H100 GPUs** with high resource utilization

CPUs vs GPUs speed

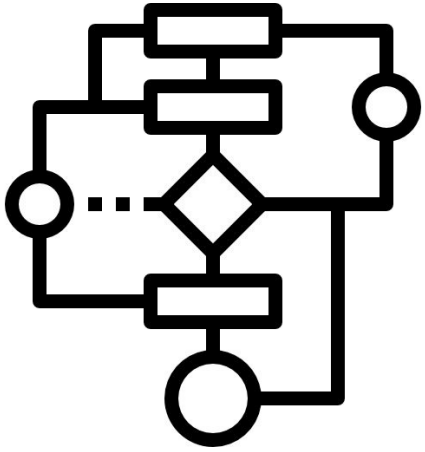


100 CPUs = 1 GPU

CPUs vs GPUs speed

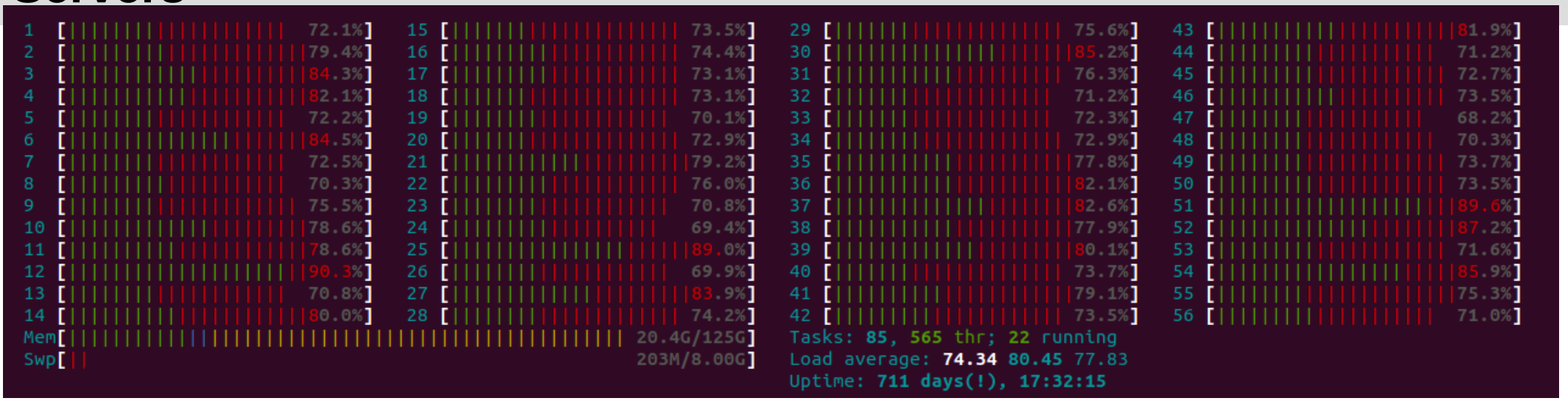


100 CPUs = 1 GPU



Servers

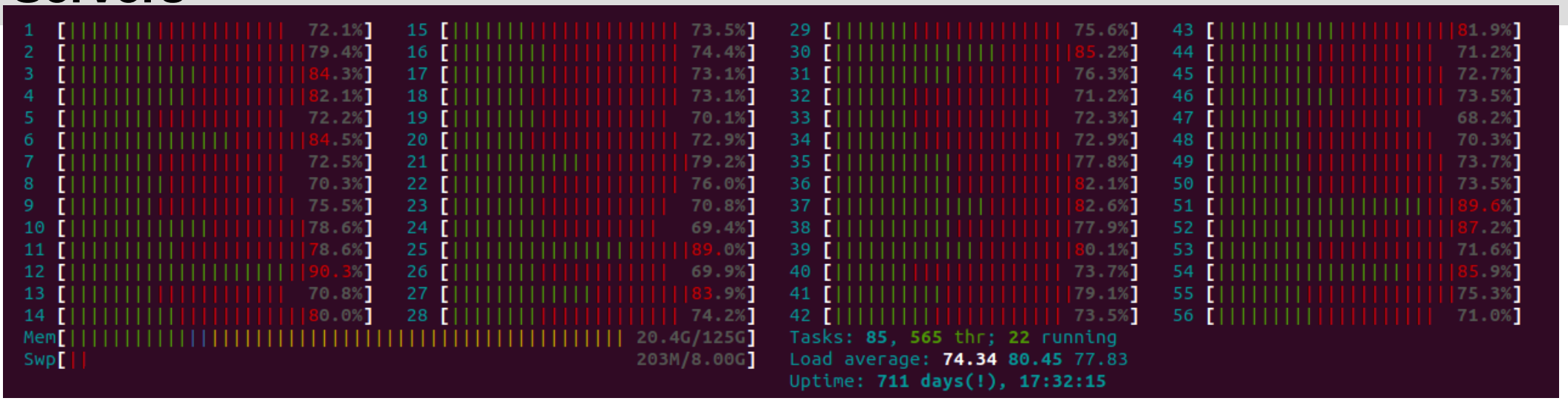
ADP



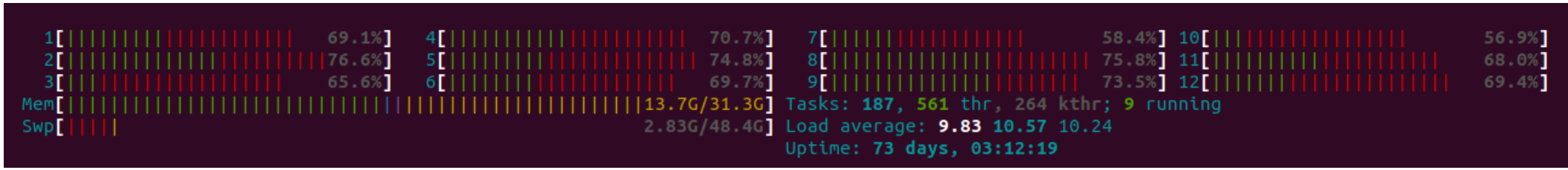
Natrix (56 CPU, 128 GB, 64 GB *tmpfs, 4 GPUs, 50TB)*

Servers

ADP



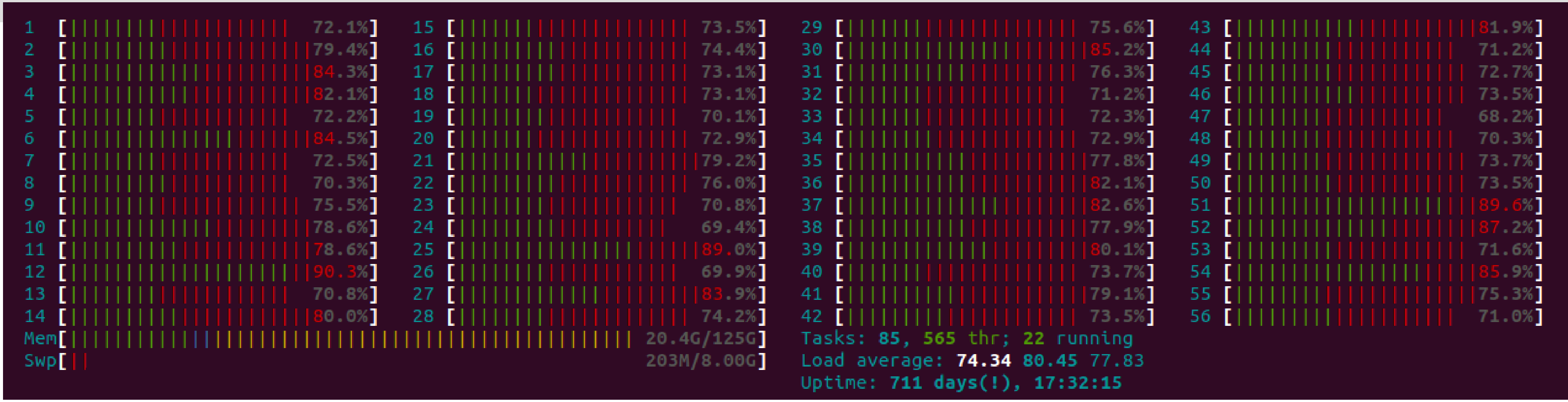
Natrix (56 CPU, 128 GB, 64 GB *tmpfs, 4 GPUs, 50TB)*



Desktop (12 CPU, 32 GB, *1 GPUs)*

10 TB free space

Servers

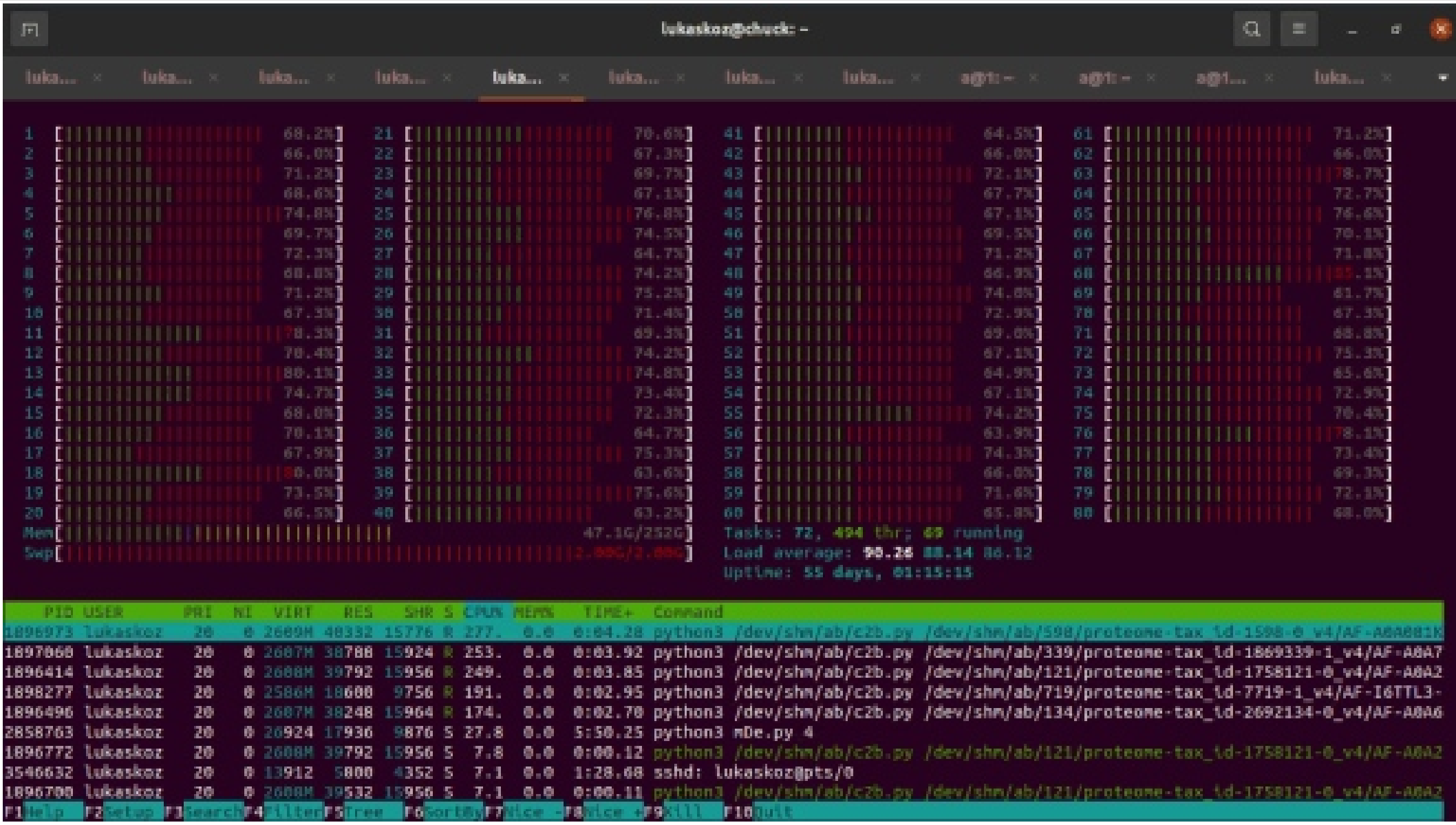


Natrix (56 CPU, 128 GB, 64 GB *tmpfs, 4 GPUs)*

NARODOWE CENTRUM NAUKI

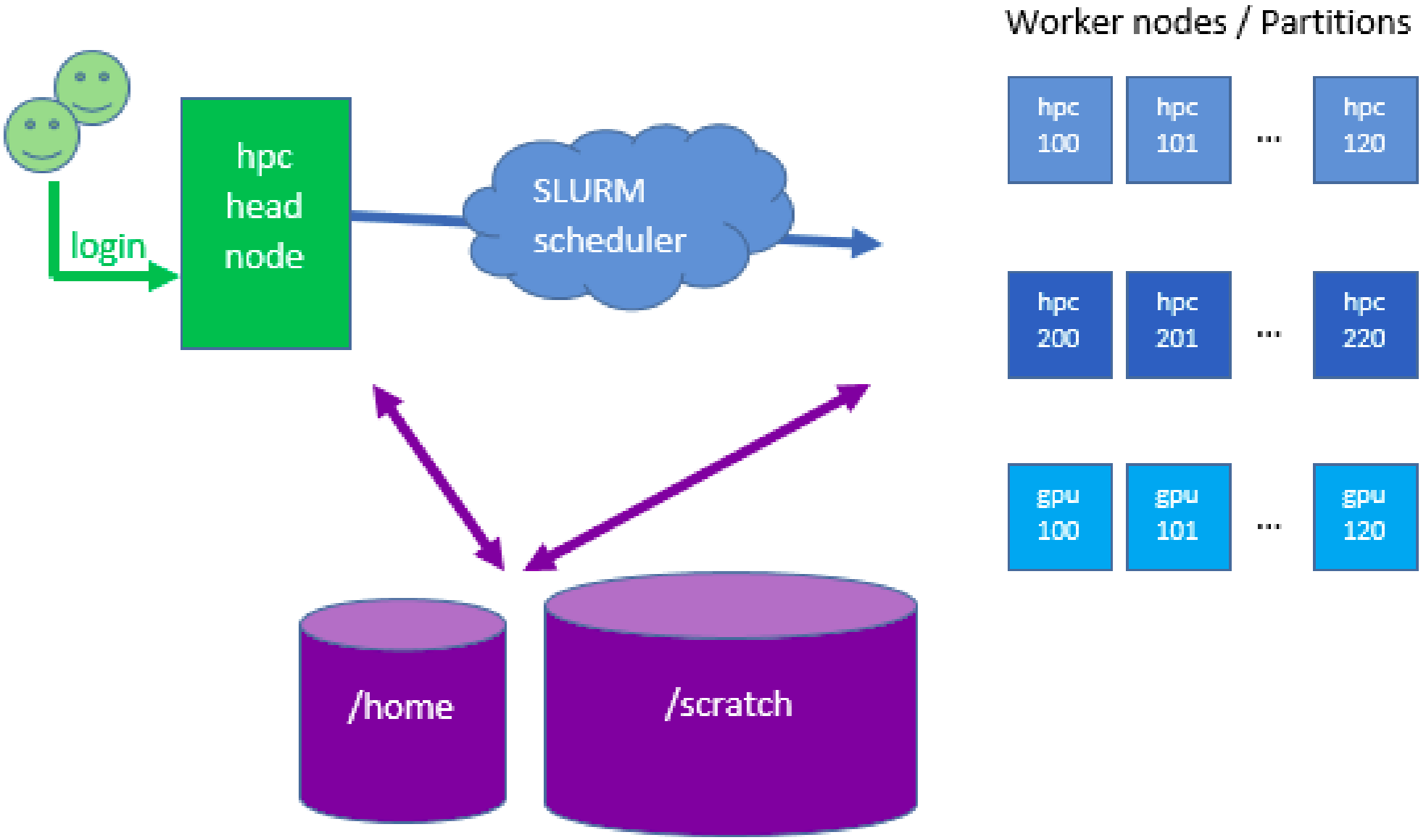
Opus

Funding: National Science Centre, Poland [2018/29/B/NZ2/01403]

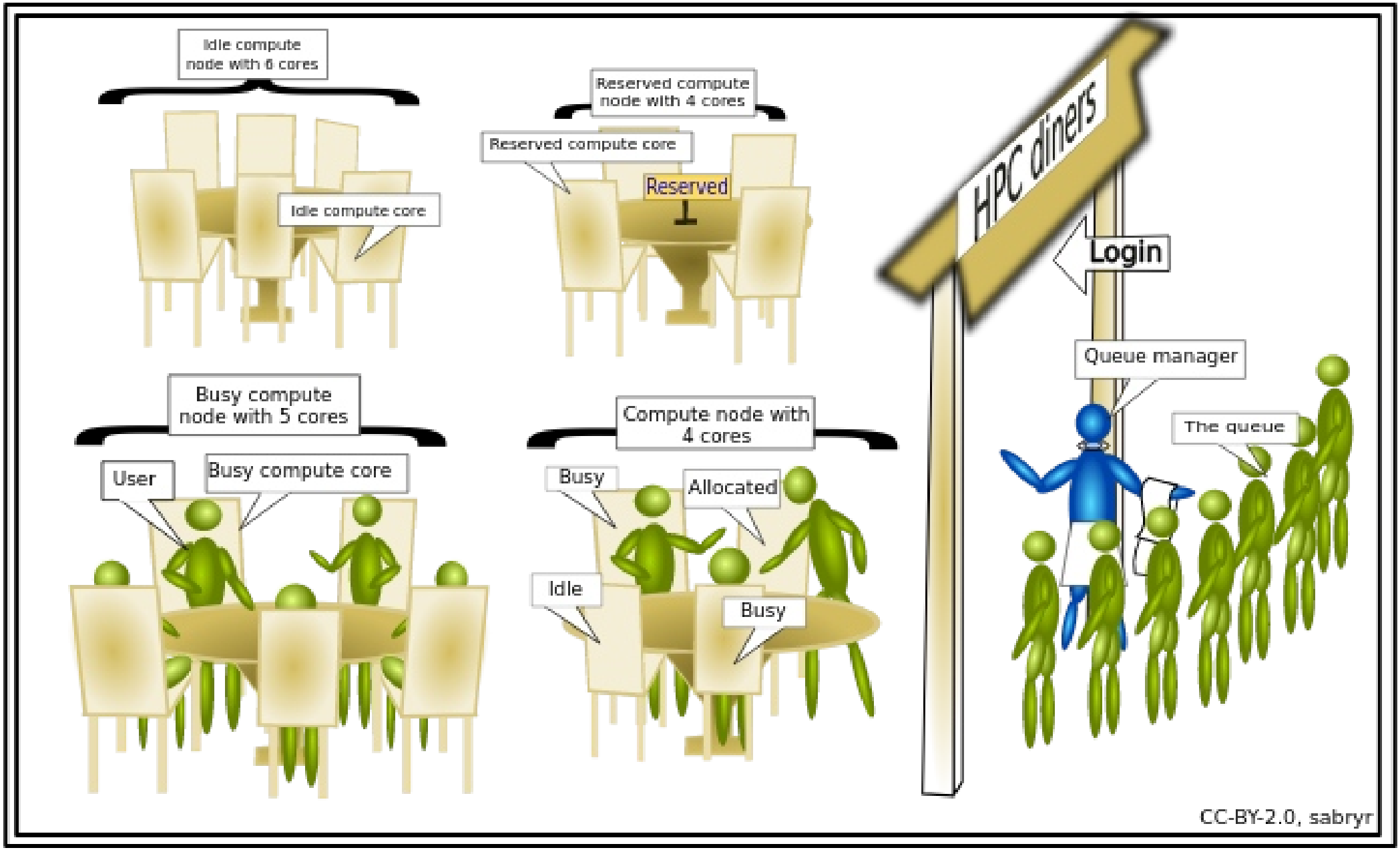


Chuck (80 CPU, 256GB RAM + 128GB *tmpfs, NO GPUs)*

Chuck (80 CPU, 256GB RAM + 128GB *tmpfs, NO GPUs)*



https://ucthpc.uct.ac.za/index.php/hpc-cluster/


Idle compute node with 6 cores
Idle compute core
Reserved compute node with 4 cores
Reserved compute core
Reserved
HPC diners
Login
Queue manager
The queue
Busy compute node with 5 cores
User
Busy compute core
Compute node with 4 cores
Busy
Allocated
Idle
Busy
CC-BY-2.0, sabryr

Some useful links:

https://slurm.schedmd.com/documentation.html

https://kdm.icm.edu.pl/Tutorials/HPC-intro/slurm_intro/

https://entropy-doc.mimuw.edu.pl/submittingjobs.html

https://www.carc.usc.edu/user-information/user-guides/hpc-basics/slurm-cheatsheet

https://docs.lumi-supercomputer.eu/runjobs/scheduled-jobs/lumic-job/

Cheat sheet for slurm

https://slurm.schedmd.com/pdfs/summary.pdf

slurm®
workload manager

Job Submission

salloc - Obtain a job allocation.
sbatch - Submit a batch script for later execution.
srun - Obtain a job allocation (as needed) and execute an application.

--array=<indexes> (e.g. "--array=1-10")	Job array specification. (sbatch command only)
--account=<name>	Account to be charged for resources used.
--begin=<time> (e.g. "--begin=18:00:00")	Initiate job after specified time.
--clusters=<name>	Cluster(s) to run the job. (sbatch command only)
--constraint=<features>	Required node features.
--cpus-per-task=<count>	Number of CPUs required per task.
--dependency=<state:jobid>	Defer job until specified jobs reach specified state.
--error=<filename>	File in which to store job error messages.
--exclude=<names>	Specific host names to exclude from job allocation.
--exclusive[=user]	Allocated nodes can not be shared with other jobs/users.
--export=<name[=value]>	Export identified environment variables.
--gres=<name[:count]>	Generic resources required per node.
--input=<name>	File from which to read job input data.
--job-name=<name>	Job name.
--label	Prepend task ID to output. (srun command only)
--licenses=<name[:count]>	License resources required for entire job.
--mem=<MB>	Memory required per node.
--mem-per-cpu=<MB>	Memory required per allocated CPU.
-N<minnodes[-maxnodes]>	Node count required for the job.
-n<count>	Number of tasks to be launched.
--nodelist=<names>	Specific host names to include in job allocation.
--output=<name>	File in which to store job output.
--partition=<names>	Partition/queue in which to run the job.
--qos=<name>	Quality Of Service.
--signal=[B:]<num>[@time]	Signal job when approaching time limit.
--time=<time>	Wall clock time limit.
--wrap=<command_string>	Wrap specified command in a simple "sh" shell. (sbatch command only)

Accounting

sacct - Display accounting data.

--allusers	Displays all users jobs.
--accounts=<name>	Displays jobs with specified accounts.
--endtime=<time>	End of reporting period.
--format=<spec>	Format output.
--name=<jobname>	Display jobs that have any of these name(s).
--partition=<names>	Comma separated list of partitions to select jobs and job steps from.
--state=<state_list>	Display jobs with specified states.
--starttime=<time>	Start of reporting period.

SchedMD
Slurm Support and Development

sacctmgr - View and modify account information.
Options:

--immediate	Commit changes immediately.
--parseable	Output delimited by '|'

Commands:

add <ENTITY> <SPECS> create <ENTITY> <SPECS>	Add an entity. Identical to the **create** command.
delete <*ENTITY*> where <*SPECS*>	Delete the specified entities.
list <ENTITY> [<SPECS>]	Display information about the specific entity.
modify <*ENTITY*> where <*SPECS*> set <*SPECS*>	Modify an entity.

Entities:

account	Account associated with job.
cluster	*ClusterName* parameter in the *slurm.conf*.
qos	Quality of Service.
user	User name in system.

Job Management

sbcast - Transfer file to a job's compute nodes.

sbcast [options] SOURCE DESTINATION

--force	Replace previously existing file.
--preserve	Preserve modification times, access times, and access permissions.

scancel - Signal jobs, job arrays, and/or job steps.

--account=<name>	Operate only on jobs charging the specified account.
--name=<name>	Operate only on jobs with specified name.
--partition=<names>	Operate only on jobs in the specified partition/queue.
--qos=<name>	Operate only on jobs using the specified quality of service.

--reservation=<name>	Operate only on jobs using the specified reservation.
--state=<names>	Operate only on jobs in the specified state.
--user=<name>	Operate only on jobs from the specified user.
--nodelist=<names>	Operate only on jobs using the specified compute nodes.

squeue - View information about jobs.

--account=<name>	View only jobs with specified accounts.
--clusters=<name>	View jobs on specified clusters.
--format=<spec> (e.g. "--format=%i %j")	Output format to display. Specify fields, size, order, etc.
--jobs<job_id_list>	Comma separated list of job IDs to display.
--name=<name>	View only jobs with specified names.
--partition=<names>	View only jobs in specified partitions.
--priority	Sort jobs by priority.
--qos=<name>	View only jobs with specified Qualities Of Service.
--start	Report the expected start time and resources to be allocated for pending jobs in order of increasing start time.
--state=<names>	View only jobs with specified states.
--users=<names>	View only jobs for specified users.

sinfo - View information about nodes and partitions.

--all	Display information about all partitions.
--dead	If set, only report state information for non-responding (dead) nodes.
--format=<spec>	Output format to display.
--iterate=<seconds>	Print the state at specified interval.
--long	Print more detailed information.
--Node	Print information in a node-oriented format.
--partition=<names>	View only specified partitions.
--reservation	Display information about advanced reservations.
-R	Display reasons nodes are in the down, drained, fail or failing state.
--state=<names>	View only nodes specified states.

scontrol - Used view and modify configuration and state. Also see the **sview** graphical user interface version.

--details	Make show command print more details.
--oneliner	Print information on one line.

Commands:

create *SPECIFICATION*	Create a new partition or .
delete *SPECIFICATION*	Delete the entry with the specified SPECIFICATION
reconfigure	All Slurm daemons will re-read the configuration file.
requeue JOB_LIST	Requeue a running, suspended or completed batch job.
show ENTITY ID	Display the state of the specified entity with the specified identification
update *SPECIFICATION*	Update job, step, node, partition, or reservation configuration per the supplied specification.

Environment Variables

SLURM_ARRAY_JOB_ID	Set to the job ID if part of a job array.
SLURM_ARRAY_TASK_ID	Set to the task ID if part of a job array.
SLURM_CLUSTER_NAME	Name of the cluster executing the job.
SLURM_CPUS_PER_TASK	Number of CPUs requested per task.
SLURM_JOB_ACCOUNT	Account name.
SLURM_JOB_ID	Job ID.
SLURM_JOB_NAME	Job Name.
SLURM_JOB_NODELIST	Names of nodes allocated to job.
SLURM_JOB_NUM_NODES	Number of nodes allocated to job.
SLURM_JOB_PARTITION	Partition/queue running the job.
SLURM_JOB_UID	User ID of the job's owner.
SLURM_JOB_USER	User name of the job's owner.
SLURM_RESTART_COUNT	Number of times job has restarted.
SLURM_PROCID	Task ID (MPI rank).
SLURM_STEP_ID	Job step ID.
SLURM_STEP_NUM_TASKS	Task count (number of MPI ranks).

Daemons

slurmctld	Executes on cluster's "head" node to manage workload.
slurmd	Executes on each compute node to locally manage resources.
slurmdbd	Manages database of resources limits, licenses, and archives accounting records.

SchedMD
Slurm Support and Development

slurm®
workload manager

http://www.schedmd.com

Using srun

https://entropy-doc.mimuw.edu.pl/submittingjobs.html

The `srun` command allows for running a single job on the cluster. Each time a valid `partition` and `qos` needs to be specified. If not redirected, all program output will be printed to the standard output.

1. Running a single command and printing the results to the standard output.

```
$ srun --partition=common --qos=1gpu1h --time=10 --gres=gpu:1 nvidia-smi -L

GPU 0: TITAN V (UUID: GPU-6426f3d6-4cec-9167-5035-4e9129551d9b)
GPU 1: TITAN V (UUID: GPU-bcaaee86-bd21-4735-edc2-d18b5fed40a7)
GPU 2: TITAN V (UUID: GPU-109e5f3c-c2e8-3a9d-486a-0df29fb6c905)
GPU 3: TITAN V (UUID: GPU-e3d1f883-02b2-1da6-80e1-32efd4ab7453)
```

2. Running a single command with a specific node selected and printing the results to the standard output.

```
$ srun --nodelist arnold --partition=common --qos=1gpu1h --time=20 --gres=gpu:1 nvidia-

GPU 0: TITAN V (UUID: GPU-6426f3d6-4cec-9167-5035-4e9129551d9b)
GPU 1: TITAN V (UUID: GPU-bcaaee86-bd21-4735-edc2-d18b5fed40a7)
GPU 2: TITAN V (UUID: GPU-109e5f3c-c2e8-3a9d-486a-0df29fb6c905)
GPU 3: TITAN V (UUID: GPU-e3d1f883-02b2-1da6-80e1-32efd4ab7453)
```

https://entropy-doc.mimuw.edu.pl/submittingjobs.html

Using sbatch

Using `sbatch` involves writing a script with all needed details for job submission. Passing all required parameters is similar to the `#DEFINE` stanzas known in the C language. Slurm uses `#BATCH`. In the batch mode, defining the `--output` file is mandatory.

1. Running a single command.

```
1   $ cat job.sh
2
3   #!/bin/bash
4   #
5   #SBATCH --job-name=test_job_username
6   #SBATCH --partition=common
7   #SBATCH --qos=1gpu1d
8   #SBATCH --gres=gpu:1
9   #SBATCH --time=1-0
10  #SBATCH --output=test_job.txt
11
12  nvidia-smi -L
13
14  $ sbatch job.sh
```

entropy

Short alias commands:

University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

entropy_account_info
entropy_gpuminutes
entropy_job_history
entropy_queue
entropy_usage_report
entropy_disk_quota
entropy_gpuminutes_test
entropy_my_queue
entropy_show_qos

```
lukaskoz@asusgpu0:~$ entropy_usage_report


 _______________
< Usage Report >
 ---------------
   \
    \   \_\_    _/_/
     \      \__/
            (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||

  # Historical GPU and CPU usage report (estimated).
  Please do not use the _sreport_ command!
  The _sreport_ does not provide sensible reports!

  The current implementation of the report is based on the ElapsedRaw
  counter, which returns elapsed time of a job in seconds. This value is
  multiplied by the number of GPUs allocated for a task. The values presented
  below are the sum of GPU Minutes for all jobs.

+--------------+---------------------------+---------------------------+
| User         | Used GPU Minutes          | Used CPU Minutes          |
+--------------+---------------------------+---------------------------+
| lukaskoz     | 96274 [66 GPU Days]       | 385099 [267 CPU Days]     |
+--------------+---------------------------+---------------------------+
```

Quality of service (QOS)

https://entropy-doc.mimuw.edu.pl/queuesresources.html

The **QOS** defines sets of limits imposed on each user (it complements partition limits in certain hierarchy). Each user has been assigned at least one **qos**, which defines the user's capabilities regarding available resources.

The **qos** defined in the cluster can be displayed using the following command:

```
clusteradm@asusgpu0:/usr/local/bin$ entropy_show_qos


 __________
< QoS list >
 ----------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

      Name                                                Flags                MaxTRESPU
---------- ---------------------------------------------------- ------------------------
    normal
   1gpu30m      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu1h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu2h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu3h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu4h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    ...         ...                                                          ...
```

Quality of service (QOS)

https://entropy-doc.mimuw.edu.pl/queuesresources.html

The **QOS** defines sets of limits imposed on each user (it complements partition limits in certain hierarchy). Each user has been assigned at least one **qos**, which defines the user's capabilities regarding available resources.

The **qos** defined in the cluster can be displayed using the following command:

```
clusteradm@asusgpu0:/usr/local/bin$ entropy_show_qos


 __________
< QoS list >
 ----------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

      Name                                                Flags                MaxTRESPU
---------- ---------------------------------------------------- ------------------------
    normal
   1gpu30m      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu1h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu2h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu3h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    1gpu4h      DenyOnLimit,OverPartQOS,NoDecay,UsageFactorSafe              gres/gpu=1
    ...         ...                                                          ...
```

User associations

https://entropy-doc.mimuw.edu.pl/queuesresources.html

Both **QOS** and **queue** (with two other, but immutable parameters) form **associations**. Associations define the ways a user can use the cluster by showing all combinations of available queues and `qos` vales. To display associations available to a user use `entropy_account_info` command:

```
kmwil@asusgpu0:~$ entropy_account_info
 _______________
< Account Info >
 ---------------
  \
   \   \_\_    _/_/
    \      \__/
           (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||

# GrpTRESMins is the cumulative limit for the GPU usage.

   Account              User    Partition          QOS          GrpTRESMins
---------- ---------------- ------------ ------------ --------------------
       mim            kmwil       common       3gpu1d       gres/gpu=10000

---
```

srun --overlap --pty --jobid=6488773 $SHELL
srun --overlap --pty --jobid=6177585 -w nid002955 $SHELL

Monitoring the jobs

srun --overlap --pty --jobid=6488773 $SHELL
srun --overlap --pty --jobid=6177585 -w nid002955 $SHELL

```
kozowski@uan04:~> srun --overlap --pty --jobid=6564364 $SHELL
kozowski@nid001614:~> top

top - 13:44:24 up 11 days, 20:53,  0 users,  load average: 35,28, 19,86, 8,36
Tasks: 2779 total,  36 running, 2743 sleeping,   0 stopped,   0 zombie
%Cpu(s): 13,1 us,  0,6 sy,  0,0 ni, 86,3 id,  0,0 wa,  0,0 hi,  0,0 si,  0,0 st
MiB Mem : 257210,2+total, 206122,4+free, 61374,66+used, 19665,40+buff/cache
MiB Swap:    0,000 total,    0,000 free,    0,000 used. 195835,5+avail Mem

   PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
186245 kozowski  20   0 2065776 999648   4220 R 100,0 0,380   0:26.75 ipc-2.0
193815 kozowski  20   0  138088 109324 101672 R 99,67 0,042   0:03.11 ipc-2.0
193550 kozowski  20   0  142736 118556 104096 R 99,35 0,045   0:05.26 ipc-2.0
193914 kozowski  20   0  136420 105296  99352 R 68,95 0,040   0:02.11 ipc-2.0
193921 kozowski  20   0  133968 104884  99476 R 66,67 0,040   0:02.04 ipc-2.0
193928 kozowski  20   0  133988 102320  96796 R 65,03 0,039   0:01.99 ipc-2.0
193935 kozowski  20   0  134208 102332  97692 R 61,76 0,039   0:01.89 ipc-2.0
193949 kozowski  20   0  133812  98720  95200 R 56,21 0,037   0:01.72 ipc-2.0
193963 kozowski  20   0  134556 103488  98312 R 50,33 0,039   0:01.54 ipc-2.0
193991 kozowski  20   0  131892  98248  94844 R 40,52 0,037   0:01.24 ipc-2.0
194012 kozowski  20   0  130480  97972  95836 R 39,22 0,037   0:01.20 ipc-2.0
194006 kozowski  20   0  136676 112400 104304 R 38,89 0,043   0:01.19 ipc-2.0
194019 kozowski  20   0  132988 102284  97732 R 37,58 0,039   0:01.15 ipc-2.0
194032 kozowski  20   0  115956  94008  91804 R 36,27 0,036   0:01.11 ipc-2.0
194033 kozowski  20   0  132316 102116  98320 R 35,95 0,039   0:01.10 ipc-2.0
```

https://entropy-doc.mimuw.edu.pl/queuesresources.html

```
lukaskoz@asusgpu0:~$ squeue
 JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
  5618      a100 baseline jkrajews PD       0:00      1 (QOSMaxGRESPerUser)
  5749      a100 baseline crewtool PD       0:00      1 (Priority)
  5748      a100 baseline crewtool PD       0:00      1 (Priority)
  5747      a100 baseline crewtool PD       0:00      1 (Resources)
  5750      a100 baseline crewtool PD       0:00      1 (Priority)
  5656      a100 switch_t kkrol_a1  R    2:11:43      1 4124gs0
  5746      a100 baseline crewtool  R      29:44      1 4124gs0
  5616      a100 baseline jkrajews  R 1-00:12:41      1 4124gs0
  5156      a100 baseline jkrajews  R 4-18:07:33      1 4124gs0
  5647     a6000 SD_train juliajoa PD       0:00      1 (QOSMaxGRESPerUser)
  5505     a6000 SD_train juliajoa  R   16:15:26      1 bruce
  5646    common     bash  cyranka  R    2:10:26      1 asusgpu6
5625_6    common jw006-da     asia  R    8:30:57      1 asusgpu3
5625_7    common jw006-da     asia  R    8:30:57      1 asusgpu5
5625_8    common jw006-da     asia  R    8:30:57      1 asusgpu2
5625_9    common jw006-da     asia  R    8:30:57      1 asusgpu4
5625_5    common jw006-da     asia  R    8:37:56      1 asusgpu1
  5672    common mini-maz  maksgro  R   10:12:52      1 steven
  5667    common mini-maz  maksgro  R   10:19:52      1 arnold
```

srun --overlap --pty --jobid=6488773 $SHELL

srun --overlap --pty --jobid=6177585 -w nid002955 $SHELL

srun --partition=common --nodelist asusgpu1 --qos=32gpu14d --time 5-0 --pty /bin/bash -l

srun --partition=common --nodelist asusgpu1 --qos=32gpu14d --gres=gpu:1 --time 5-0 --pty /bin/bash -l

Hardware

entropy

The *entropy* cluster consists of 12 compute nodes.

University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Compute nodes

Server	CPU	GPU	Memory
4124gs0	AMD EPYC 7713	8x A100	1024 GB
asusgpu1	Intel Xeon E5-2640 v4	8x Titan V	128 GB
asusgpu2	Intel Xeon E5-2660 v4	8x RTX 2080 Ti	128 GB
asusgpu3	Intel Xeon E5-2660 v4	2x RTX 1080 Ti, 6x Titan V	128 GB
asusgpu4	Intel Xeon E5-2660 v4	2x RTX 2080 Ti, 6x Titan X	128 GB
asusgpu5	Intel Xeon E5-2640 v4	2x RTX 2080 Ti, 6x Titan X	128 GB
asusgpu6	Intel Xeon E5-2640 v4	8x RTX 2080 Ti	128 GB
arnold	Intel Xeon E5-2630 v4	4x Titan V	64 GB
bruce	Intel Xeon E5-2630 v4	2x RTX A6000	128 GB
steven	Intel Xeon E5-2630 v4	4x Titan V	64 GB
sylvester	Intel Xeon E5-2630 v4	4x Titan V	64 GB

Hardware

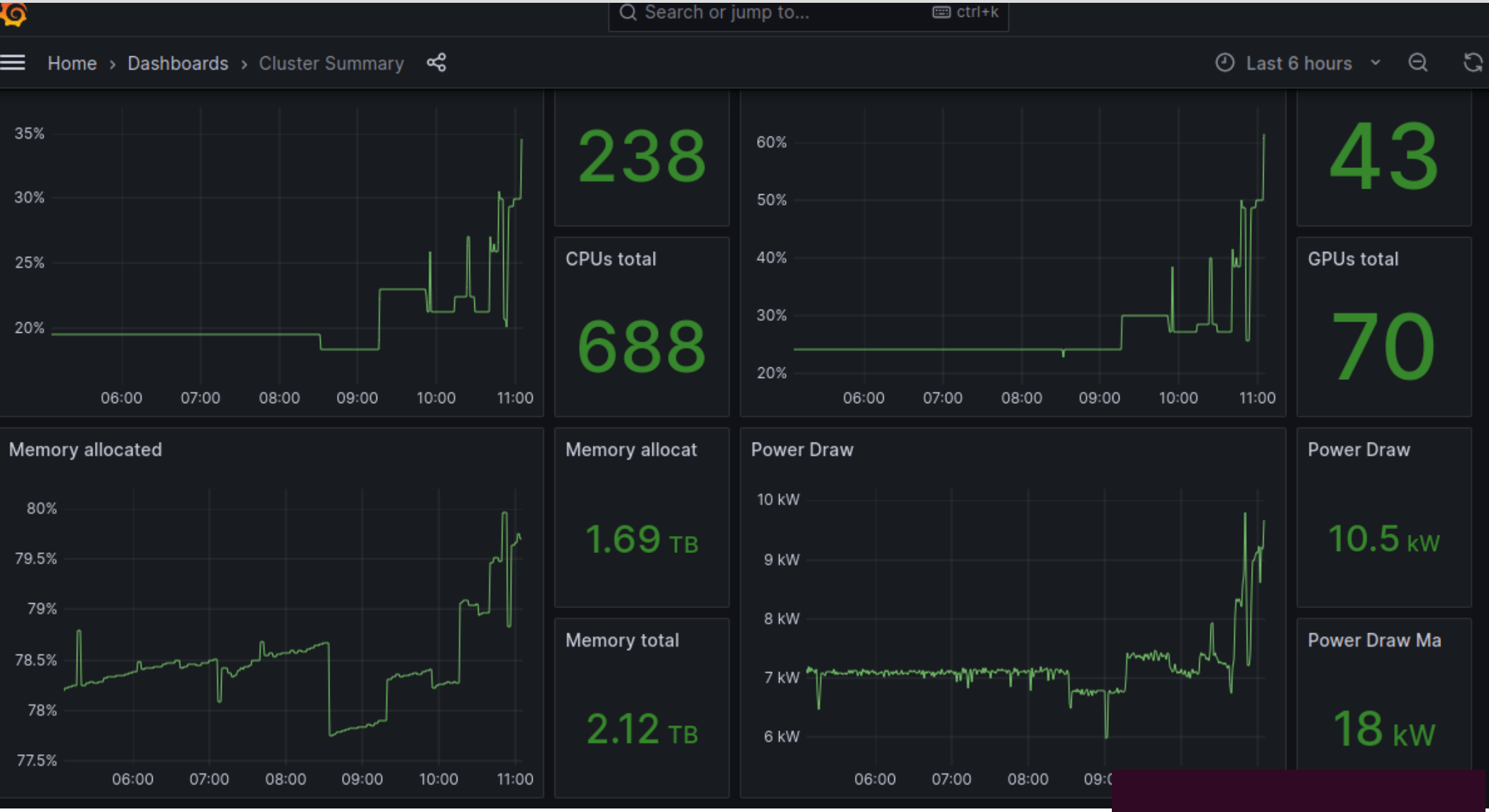
entropy

The *entropy* cluster consists of 12 compute nodes.

University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

Compute nodes

Server	CPU	GPU	Memory
4124gs0	AMD EPYC 7713	8x A100	1024 GB
asusgpu1	Intel Xeon E5-2640 v4	8x Titan V	128 GB
asusgpu2	Intel Xeon E5-2660 v4	8x RTX 2080 Ti	128 GB
asusgpu3	Intel Xeon E5-2660 v4	2x RTX 1080 Ti, 6x Titan V	128 GB
asusgpu4	Intel Xeon E5-2660 v4	2x RTX 2080 Ti, 6x Titan X	128 GB
asusgpu5	Intel Xeon E5-2640 v4	2x RTX 2080 Ti, 6x Titan X	128 GB
asusgpu6	Intel Xeon E5-2640 v4	8x RTX 2080 Ti	128 GB
arnold	Intel Xeon E5-2630 v4	4x Titan V	64 GB
bruce	Intel Xeon E5-2630 v4	2x RTX A6000	128 GB
steven	Intel Xeon E5-2630 v4	4x Titan V	64 GB
sylvester	Intel Xeon E5-2630 v4	4x Titan V	64 GB



University of Warsaw
Faculty of Mathematics, Informatics and Mechanics

entropy

Search or jump to... ctrl+k

Home › Dashboards › Cluster Summary

Last 6 hours

238

CPUs total

688

43

GPUs total

70

Memory allocated

Memory allocat

1.69 TB

Memory total

2.12 TB

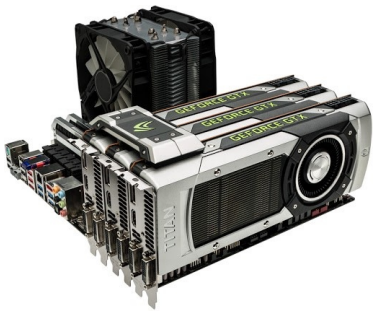
Power Draw

Power Draw

10.5 kW

Power Draw Ma

18 kW



entropy

https://entropy-doc.mimuw.edu.pl

https://entropy.mimuw.edu.pl/grafana/

Servers / GPU

Last 3 hours | 1m

rver | asusgpu4.mimuw.edu.pl | GPU | db133710-a9ba-607e-1e9c-5116ff63f3be

Model

NVIDIA TITAN V

Memory

12.9 GB

Driver Version

510.85.02

GPU Throttle

GPU Idle	Not Active
Application Clock	Not Active
Hardware Power ...	Not Active
Hardware Slowd...	Not Active
Hardware Therm...	Not Active
Software Power ...	Not Active
Software Therma...	Not Active

Temperature

°C
°C
°C

Memory Used

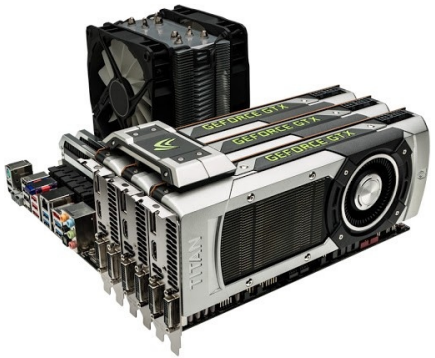
3 GB
2 GB
1 GB
0 B

Power Draw

80 W
60 W
40 W

Clocks

1.50 GHz
1 GHz
500 MHz
0 Hz
16:00 17:00 18:00
Graphics Memory SM



entropy

https://entropy-doc.mimuw.edu.pl

https://entropy.mimuw.edu.pl/grafana/

```
#!/bin/bash -l

#SBATCH --job-name=8502_8829
#SBATCH --qos=32gpu14d
#SBATCH --partition=common      # Partition (queue) name

#SBATCH --gres=gpu:1
#SBATCH --mem=12000

#SBATCH --time=7-0              # 7 days (max)

#SBATCH --output="/home/lukaskoz/logs/8502_8829.out"
#SBATCH --error="/home/lukaskoz/logs/8502_8829.err"

cd /home/lukaskoz/
source /home/lukaskoz/venv_ProtBert/bin/activate
srun python BFD_range_runner_gpu.py 8502 8829
deactivate
```

srun -J 8502_8829 -o ~/logs/8502_8829.out -e ~/logs/8502_8829.errr --nodelist sylvester --partition=common --qos=32gpu14d --time 7-0 --mem 10000 /bin/bash ~/sruns/sub.sh & *(one line)*

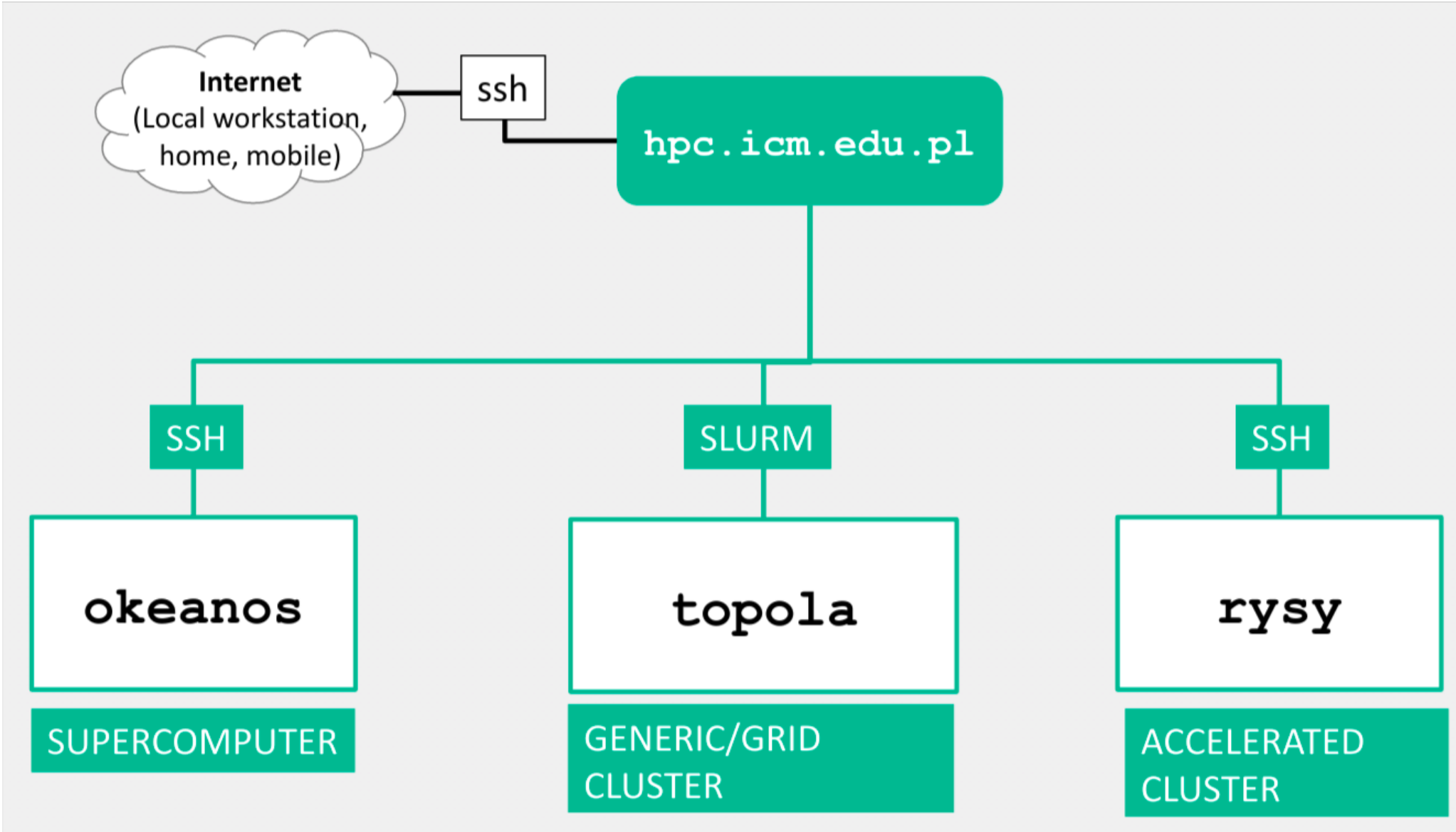
Where sub.sh is:

#!/bin/bash

cd /home/lukaskoz/
source /home/lukaskoz/venv_ProtBert/bin/activate
srun python BFD_range_runner_gpu.py 8502 8829
deactivate

https://kdm.icm.edu.pl/

Nazwa	Typ	Architektura	Liczba i nazwa węzłów obliczeniowych	Parametry węzła obliczeniowego
Okeanos	Superkomputer	Intel Haswell Cray XC40	1084	24 rdzenie, 128 GB pamięci RAM
Topola	Klaster HPC, Klaster PL-Grid	Intel Haswell Huawei E9000	223	28 rdzeni, 64 lub 128 GB pamięci RAM
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	6	36 rdzenie, 380 GB pamięci RAM, 4x GPU V100 32GB
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	1	48 rdzeni, 1500 GB pamięci RAM, 8x GPU V100 16GB

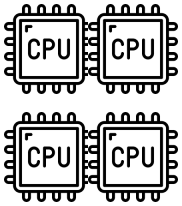

Internet
(Local workstation,
home, mobile)
ssh
hpc.icm.edu.pl
SSH
SLURM
SSH
okeanos
topola
rysy
SUPERCOMPUTER
GENERIC/GRID
CLUSTER
ACCELERATED
CLUSTER

UNIWERSYTET WARSZAWSKI

icm
icm.edu.pl

https://kdm.icm.edu.pl/

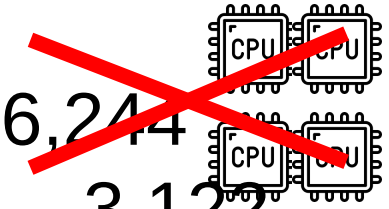
TOPOLA 6,244



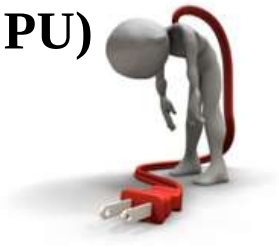
https://kdm.icm.edu.pl/

Nazwa	Typ	Architektura	Liczba i nazwa węzłów obliczeniowych	Parametry węzła obliczeniowego
Okeanos	Superkomputer	Intel Haswell Cray XC40	1084	24 rdzenie, 128 GB pamięci RAM
Topola	Klaster HPC, Klaster PL-Grid	Intel Haswell Huawei E9000	223	28 rdzeni, 64 lub 128 GB pamięci RAM
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	6	36 rdzenie, 380 GB pamięci RAM, 4x GPU V100 32GB
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	1	48 rdzeni, 1500 GB pamięci RAM, 8x GPU V100 16GB

TOPOLA ~~6,244~~ 3,122



QOS 64 (~128 CPU)

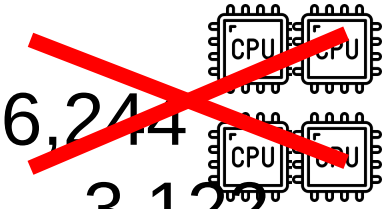


https://kdm.icm.edu.pl/

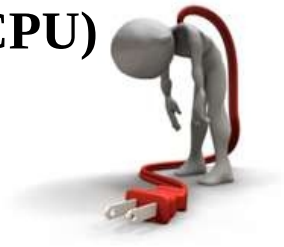
Nazwa	Typ	Architektura	Liczba i nazwa węzłów obliczeniowych	Parametry węzła obliczeniowego
Okeanos	Su			128 GB pamięci RAM
Topola	Kla Gri			4 lub 128 GB pamięci
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	6	36 rdzenie, 380 GB pamięci RAM, 4x GPU V100 32GB
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	1	48 rdzeni, 1500 GB pamięci RAM, 8x GPU V100 16GB

Total budget in 2023
40 mln CPUh

TOPOLA ~~6,244~~ 3,122



QOS 64 (~128 CPU)



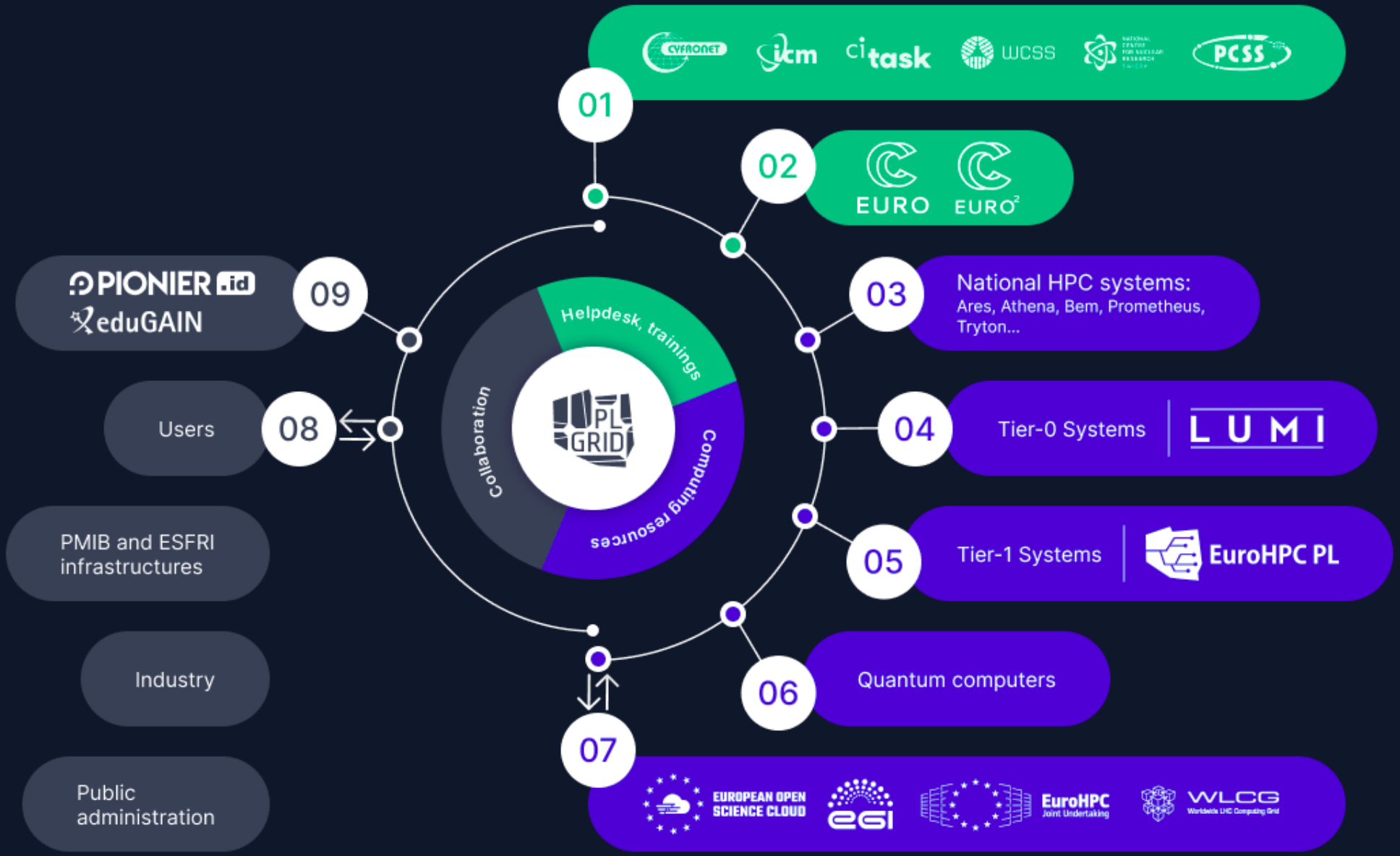
https://kdm.icm.edu.pl/

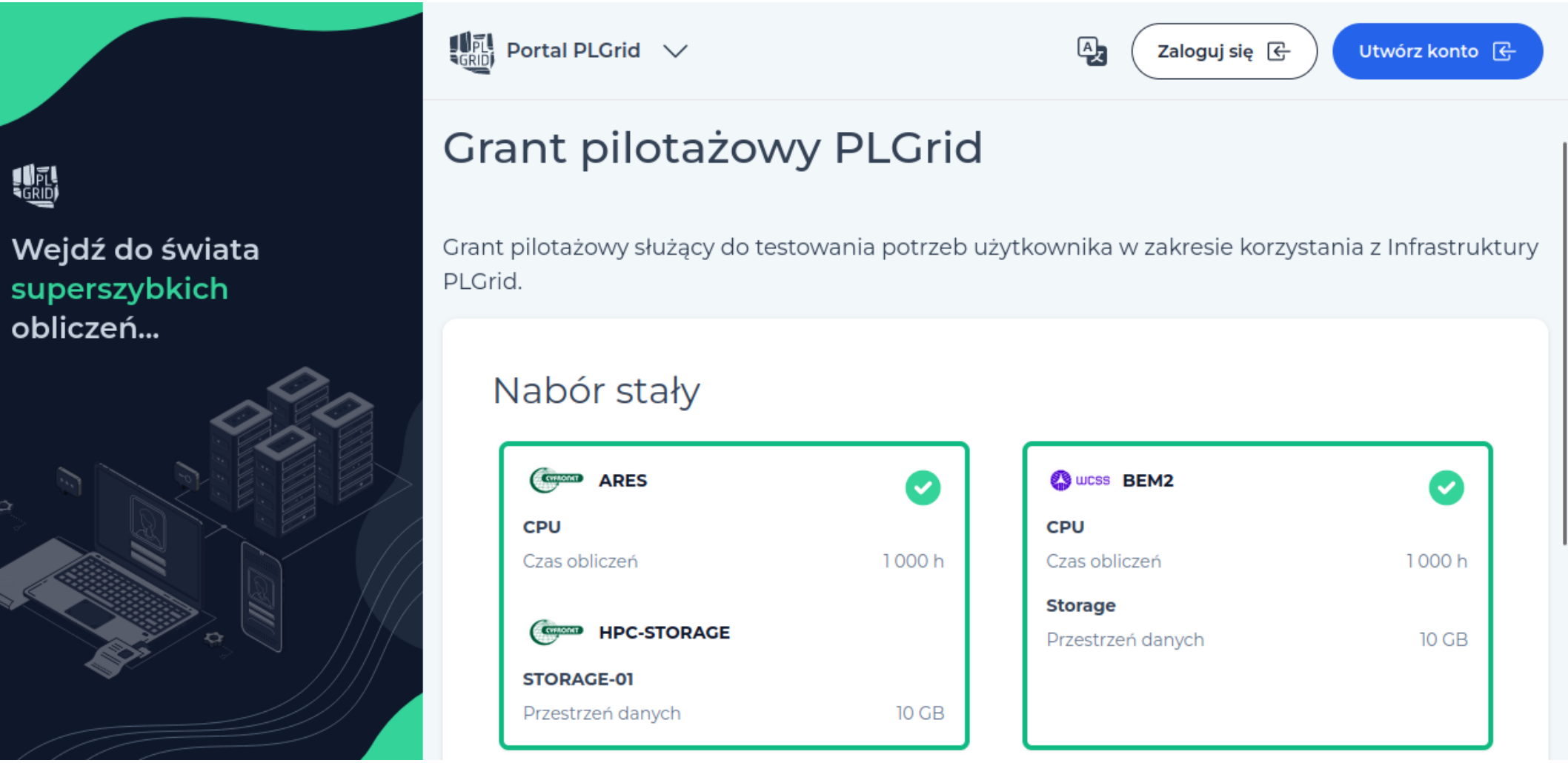
Nazwa	Typ	Architektura	Liczba i nazwa węzłów obliczeniowych	Parametry węzła obliczeniowego
Okeanos	Su			128 GB pamięci RAM
Topola	Kla Gri			4 lub 128 GB pamięci
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	6	36 rdzenie, 380 GB pamięci RAM, 4x GPU V100 32GB
Rysy/GPU	Klaster GPU,	Intel Skylake, NVIDIA Volta	1	48 rdzeni, 1500 GB pamięci RAM, 8x GPU V100 16GB

Total budget in 2023
40 mln CPUh

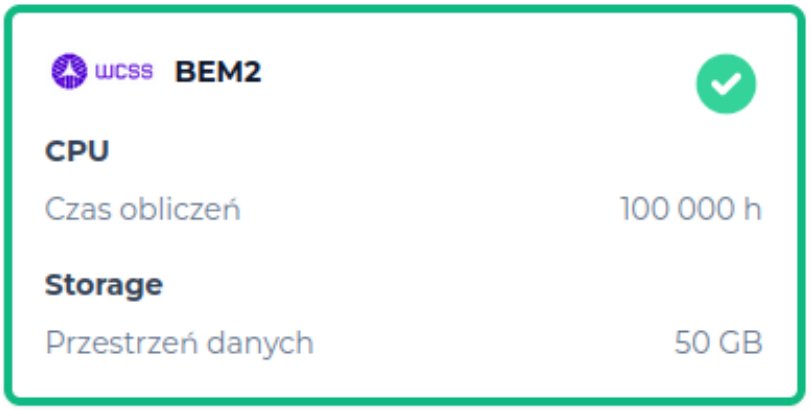
Do not ask for GPUs

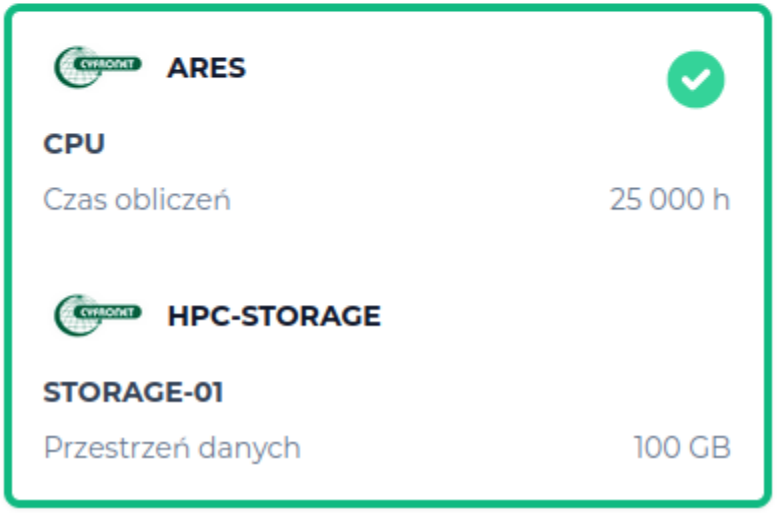
https://plgrid.pl


01
CYFRONET
icm
citask
WCSS
PCSS
02
EURO
EURO²
03
National HPC systems:
Ares, Athena, Bem, Prometheus, Tryton...
04
Tier-0 Systems
LUMI
05
Tier-1 Systems
EuroHPC PL
06
Quantum computers
07
EUROPEAN OPEN SCIENCE CLOUD
EGI
EuroHPC Joint Undertaking
WLCG Worldwide LHC Computing Grid
08
Users
PMIB and ESFRI infrastructures
Industry
Public administration
09
PIONIER.id
eduGAIN
Helpdesk, trainings
Computing resources
Collaboration
PL GRID



https://plgrid.pl





https://plgrid.pl

ACK Cyfronet Ares

Nowoczesny superkomputer o wydajności obliczeniowej ponad 3,5 PetaFlopsów (CPU) i 500 TFlopsów (GPU). Opis techniczny klastra Ares znajdziesz tutaj. Jeżeli chcesz dowiedzieć się więcej, przejdź do dokumentacji PLGrid.

CPU - Aktywny plglspfegpu-cpu

Zasoby dostępne
od 20.12.2023 do 30.03.2024

Renegocjuj zasoby

Zużycie zasobów (Ostatnia aktualizacja: 17.03.2024 14:38)

Zużyto: 950 311 h

0% 100%

OTRZYMANO NASTĘPUJĄCE ZASOBY

Czas Obliczeń	1 250 000 h
Maksymalny Czas Wykonania Zadania	72 h

https://plgrid.pl

ACK Cyfronet Ares

Nowoczesny superkomputer o wydajności obliczeniowej ponad 3,5 PetaFlopsów (CPU) i 500 TFlopsów (GPU). Opis techniczny klastra Ares znajdziesz tutaj. Jeżeli chcesz dowiedzieć się więcej, przejdź do dokumentacji PLGrid.

GPU - Aktywny plglspfegpu-gpu

Zasoby dostępne od 31.03.2023 do 30.03.2024

Renegocjuj zasoby

Zużycie zasobów (Ostatnia aktualizacja: 16.03.2024 22:13)

OTRZYMANO NASTĘPUJĄCE ZASOBY

Czas Obliczeń Na Kartach GPU	50 000 h

https://plgrid.pl

ACK Cyfronet Athena

Najszybszy superkomputer w Polsce, osiągający moc obliczeniową ponad 7,7 PetaFlopsów. Opis techniczny klastra Athena znajdziesz tutaj. Jeżeli chcesz dowiedzieć się więcej, przejdź do dokumentacji PLGrid.

GPU-A100 - Aktywny plglspfegpu-gpu-a100

Zasoby dostępne
od 27.08.2023 do 30.03.2024

Renegocjuj zasoby

Zużycie zasobów (Ostatnia aktualizacja: 16.03.2024 22:13)

Zużyto: 424 071 h

0% 100%

OTRZYMANO NASTĘPUJĄCE ZASOBY

Czas Obliczeń Na Kartach GPU	425 000 h

https://plgrid.pl

Ares:
3,5 PFlops of CPUs
500 Tflops of GPUs
11 PB disk

532 nodes
(48x CPUs, 192 GB RAM)

256 nodes
(48x CPUs, 384 GB RAM)

9 nodes
(32 Cpus, 384 GB RAM,
8 NVIDIA Tesla V100)

In total:
37 824 CPUs
72 GPUs

Documentation: https://docs.cyfronet.pl/display/~plgpawlik/Ares

404. position in TOP500 (2023XI)

ARES

PL GRID

CPU CPU CPU CPU

37k CPUs

TOP500 SUPERCOMPUTER

CYFRONET

Ares:
3,5 PFlops of CPUs
500 Tflops of GPUs
11 PB disk

532 nodes
(48x CPUs, 192 GB RAM)

256 nodes
(48x CPUs, 384 GB RAM)

9 nodes
(32 Cpus, 384 GB RAM,
8 NVIDIA Tesla V100)

In total:
37 824 CPUs
72 GPUs

Documentation: https://docs.cyfronet.pl/display/~plgpawlik/Ares

404. position in TOP500 (2023XI)

ARES

PL GRID

CPU CPU CPU CPU

37k CPUs

NO LIMITS !!!
... for number of jobs

TOP500 SUPERCOMPUTER

CYFRONET

Athena

20 mln zł

7,7 PFlops of CPUs
240 Pflops of GPUs

48 nodes
(128x CPUs, 1 TB RAM,
8x GPU NVIDIA A100 **40GB**)

In total:
6 144 CPUs
364 GPUs

291. position in
TOP500 (2023XI)

Documentation: https://docs.cyfronet.pl/display/~plgpawlik/Athena

364 x

Athena

20 mln zł

7,7 PFlops of CPUs
240 Pflops of GPUs

48 nodes
(128x CPUs, **1 TB RAM**,
8x GPU NVIDIA A100 **40GB**)

In total:
6 144 CPUs
364 GPUs

291. position in
TOP500 (2023XI)

Documentation: https://docs.cyfronet.pl/display/~plgpawlik/Athena

364 x

RAM file system

Filesystem	Size	Used	Avail	Use%	Mounted on
udev	63G	0	63G	0%	/dev
tmpfs	13G	2,9M	13G	1%	/run
/dev/sda3	196G	70G	116G	38%	/
tmpfs	63G	8,3G	55G	14%	/dev/shm
tmpfs	5,0M	4,0K	5,0M	1%	/run/lock
tmpfs	63G	0	63G	0%	/sys/fs/cgroup
/dev/sda2	9,8G	344M	9,0G	4%	/boot
/dev/sda1	511M	6,7M	505M	2%	/boot/efi
/dev/sda4	5,3T	4,7T	315G	94%	/mnt/sda4
tmpfs	13G	0	13G	0%	/run/user/1000

INPUT/OUTPUT (I/O)

RAM file system

```
Filesystem      Size  Used Avail Use% Mounted on
udev             63G     0   63G   0% /dev
tmpfs            13G  2,9M   13G   1% /run
/dev/sda3       196G   70G  116G  38% /
tmpfs            63G  8,3G   55G  14% /dev/shm
tmpfs           5,0M  4,0K  5,0M   1% /run/lock
tmpfs            63G     0   63G   0% /sys/fs/cgroup
/dev/sda2       9,8G  344M  9,0G   4% /boot
/dev/sda1       511M  6,7M  505M   2% /boot/efi
/dev/sda4       5,3T  4,7T  315G  94% /mnt/sda4
tmpfs            13G     0   13G   0% /run/user/1000
```

RamFS & TmpFS

INPUT/OUTPUT (I/O)

Athena


```
[athena][plglukaskoz@login01 plglukaskoz]$ squeue -S V|grep 'R'|tail
            436436 plgrid-gp 1260-126 plglukas  R       12:40      1 t0011
            436437 plgrid-gp 3205-321 plglukas  R       12:40      1 t0011
            436438 plgrid-gp 1265-127 plglukas  R       12:40      1 t0011
            436439 plgrid-gp 3210-321 plglukas  R       10:28      1 t0029
            436440 plgrid-gp 1270-127 plglukas  R       10:15      1 t0032
            436442 plgrid-gp 3215-322 plglukas  R       10:05      1 t0039
            436443 plgrid-gp 1275-128 plglukas  R       10:01      1 t0030
            436444 plgrid-gp 3220-322 plglukas  R        5:22      1 t0037
            436445 plgrid-gp 1280-128 plglukas  R        4:11      1 t0020
            436446 plgrid-gp 3225-323 plglukas PD        0:00      1 (Resour
[athena][plglukaskoz@login01 plglukaskoz]$ squeue -S V|grep 'PD'|tail
            436476 plgrid-gp 3300-330 plglukas PD        0:00      1 (Priori
            436477 plgrid-gp 1360-136 plglukas PD        0:00      1 (Priori
            436478 plgrid-gp 3305-331 plglukas PD        0:00      1 (Priori
            436479 plgrid-gp 1365-137 plglukas PD        0:00      1 (Priori
            436480 plgrid-gp 3310-331 plglukas PD        0:00      1 (Priori
            436481 plgrid-gp 1370-137 plglukas PD        0:00      1 (Priori
            436482 plgrid-gp 3315-332 plglukas PD        0:00      1 (Priori
            436483 plgrid-gp 1375-138 plglukas PD        0:00      1 (Priori
            436484 plgrid-gp 3320-332 plglukas PD        0:00      1 (Priori
            436485 plgrid-gp 1380-138 plglukas PD        0:00      1 (Priori
[athena][plglukaskoz@login01 plglukaskoz]$ squeue -S V|grep 'R'|wc
    304    2432   22513
```

Helios

PRICE: 86 mln zł

To be released at 04.2024

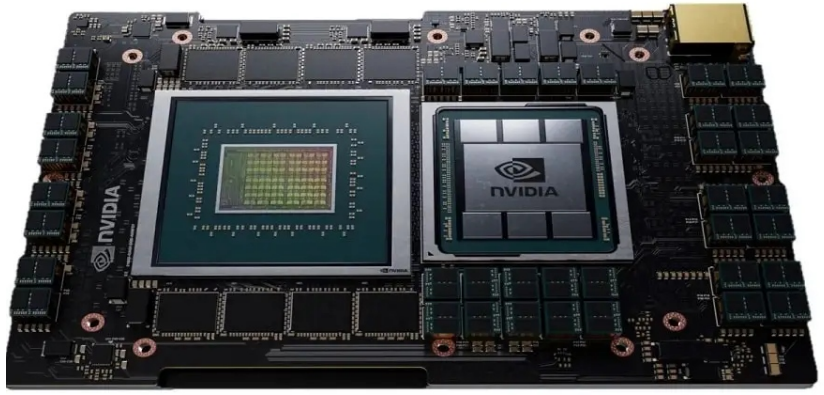
35 PFlops of CPUs

In total:

75 264 CPUs

200 TB DDR5 RAM

24 acelerators NVIDIA H100



440 NVIDIA Grace Hopper GH200

155. position in TOP500 (2023XI)

TOP500 SUPERCOMPUTER

CYFRONET

Helios

PRICE: 86 mln zł

To be released at 04.2024

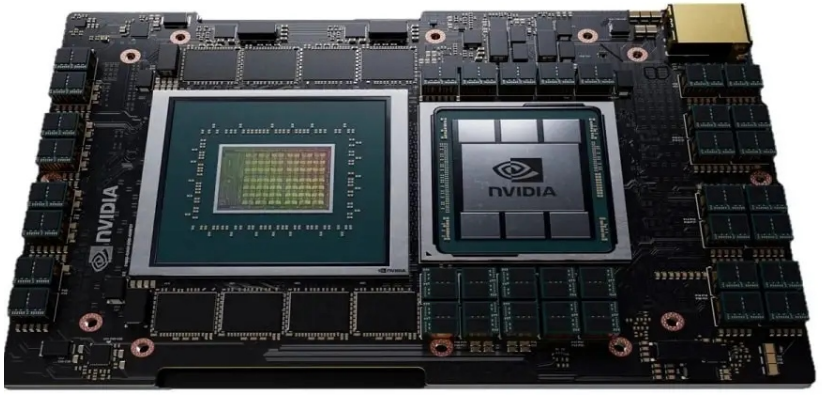
35 PFlops of CPUs

In total:

75 264 CPUs

200 TB DDR5 RAM

24 acelerators NVIDIA H100



440 NVIDIA Grace Hopper GH200

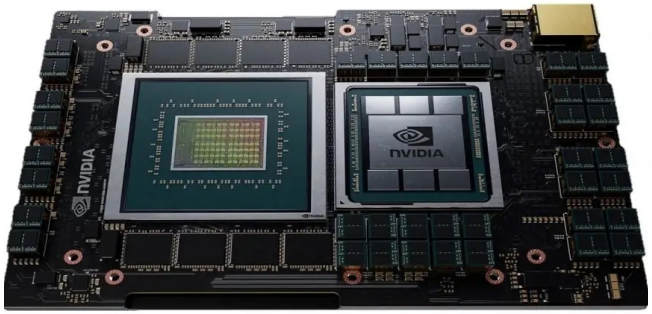
155. position in TOP500 (2023XI)

TOP500 SUPERCOMPUTER

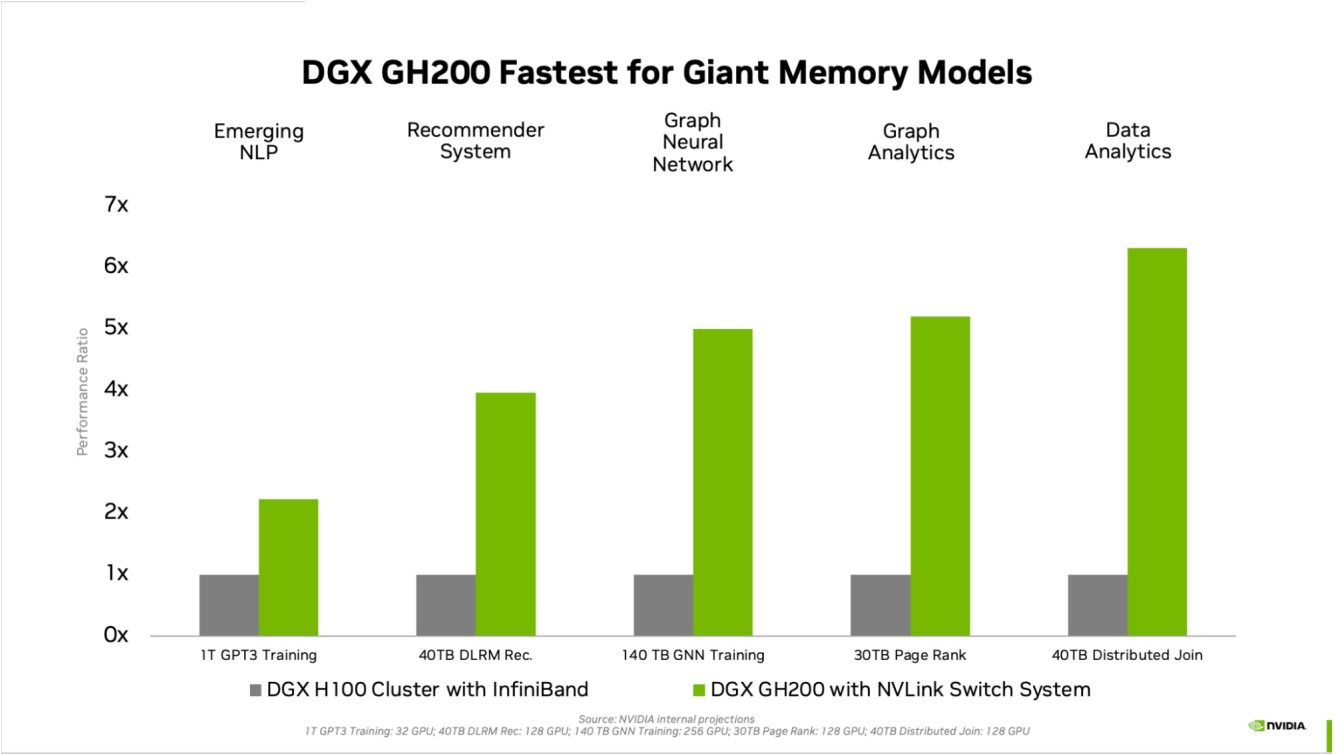
CYFRONET

NVIDIA Grace Hopper GH200

Starting price 43 500 USD



Feature	Description
Grace CPU cores (number)	Up to 72 cores
CPU LPDDR5X bandwidth (GB/s)	Up to 500GB/s
GPU HBM bandwidth (GB/s)	4TB/s HBM3 4.8TB/s HBM3e
NVLink-C2C bandwidth (GB/s)	900GB/s total, 450GB/s per direction
CPU LPDDR5X capacity (GB)	Up to 480GB
GPU HBM capacity (GB)	96GB HBM3 141GB HBM3e
PCIe Gen 5 Lanes	64x



LUMI

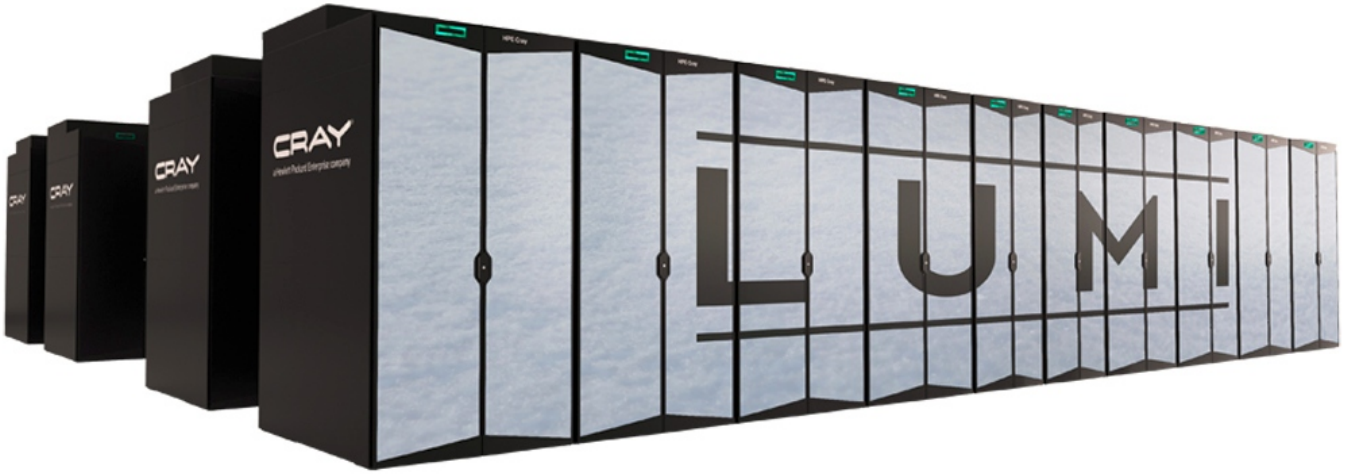
1 SYSTEM
380 Petaflop/s
PEAK PERFORMANCE

145 mln euro

In total:

362k CPUs

12k GPUs



Power consumtion: up to 8.5 MW (hydroenergy)

5. position in TOP500 (2023XI)
the fastest supercomputer in Europe and the fifth fastest globally

https://lumi-supercomputer.eu/

https://docs.lumi-supercomputer.eu

EuroHPC
Joint Undertaking

PL GRID

LUMI

LUMI Consortium

- Unique consortium of 10 countries with strong national HPC centers
- The resources of LUMI will be allocated per the investments
- The share of the EuroHPC JU (50%) will be allocated by a peer-review process (cf. PRACE Tier-0 access) and available for all European researchers
- The shares of the LUMI partner countries will be allocated by local considerations and policies – seen and handled as extensions to national resources

EuroHPC Joint Undertaking

PL GRID

LUMI

https://www.dpconline.org/docs/miscellaneous/events/2021-events/2524-dpc-csc-lumi-presentation-11112021-val/file

LUMI Consortium

EuroHPC
Joint Undertaking

PL GRID

LUMI

LUMI: Excess Heat Utilization Process Overview

Annual CO_2 savings 12 400 tonnes

DRY AIR COOLING FOR BACK UP ~ 10 MW

DISTRICT HEATING NETWORKS ~ 10 MW
-Renforsin Ranta Business Park
-CITY of Kajaani

Service demarcation point for the excess heat utilization

HEAT EXCHANGERS

HEAT PUMPS

Heat pumps use renewable energy

HPC load

In addition of Direct Liquid Cooling there is approximately 1 MW of capacity for the air-cooled servers (e.g. storage and management servers). This load is free-cooled.

EuroHPC Joint Undertaking

PL GRID

LUMI

LUMI

EuroHPC-JU projects | LUMI consortium countries projects

Name	Max walltime	Max jobs	Max resources/job	Hardware partition used
standard-g	2 days	105 (100 running)	512 nodes	LUMI-G
standard	2 days	60 (50 running)	256 nodes	LUMI-C
bench	1 day	n/a	All nodes	LUMI-C and LUMI-G

LUMI

Name	Max walltime	Max jobs	Max resources/job	Hardware partition used
dev-g	3 hours	2 (1 running)	32 nodes	LUMI-G
debug	30 minutes	2 (1 running)	4 nodes	LUMI-C
small-g	3 days	210 (200 running)	4 nodes	LUMI-G
small	3 days	220 (200 running)	4 nodes	LUMI-C
largemem	1 day	30 (20 running)	1 nodes	LUMI-D

```
scontrol show partition <partition-name>
```

LUMI

EuroHPC
Joint Undertaking

LUMI

LUMI

```
kozowski@uan03:/scratch/project_465000991> lumi-workspaces

Quota for your projects:

Disk area                                Capacity(used/max)  Files(used/max)
-----------------------------------------------------------------------------
Personal home folder
Home folder is hosted on lustrep4

/users/kozowski                                  9,8G/22G         3,7K/100K
-----------------------------------------------------------------------------
Project: project_465000961
Project is hosted on lustrep2

/projappl/project_465000961                       36G/54G          26K/100K
/scratch/project_465000961                        10M/55T          836/2,0M
/flash/project_465000961                        4,1K/2,2T            1/1,0M
-----------------------------------------------------------------------------
Project: project_465000991
Project is hosted on lustrep4

/projappl/project_465000991                     4,1K/54G             1/100K
/scratch/project_465000991                       5,7T/55T         1,4M/2,0M
/flash/project_465000991                        4,1K/2,2T            1/1,0M
-----------------------------------------------------------------------------
```

LUMI

```
kozowski@uan03:/scratch/project_465000991> lumi-allocations
Data updated: 2024-03-15 19:22:44
Project                |                          CPU (used/allocated)|
-------------------------------------------------------------------------------
project_465000961      |            94556/50000 (189.1%) core/hours|
project_465000991      |       2777014/9000000  (30.9%) core/hours|
```

```
          GPU (used/allocated)|              Storage (used/allocated)
-------------------------------------------------------------------------------
   1013/1000 (101.3%) gpu/hours|                 31/10 (310.0%) TB/hours
10705/500000   (2.1%) gpu/hours|           3181/10000  (31.8%) TB/hours
```

EuroHPC
Joint Undertaking

LUMI

kozowski@uan03:/scratch/project_465000991> **squeue** | wc
   3163   25304 256268

EuroHPC
Joint Undertaking

kozowski@uan03:/scratch/project_465000991> squeue | wc
  3163 25304 256268

kozowski@uan03:/scratch/project_465000991> **squeue**

```
        JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
      6512730 standard- siesta_g piechota  R    6:50:11      7
nid[006488,006548,007225,007635,007735,007766,007772]
      6512719 standard- siesta_g piechota  R    6:57:05      7
nid[006492,006729,007552,007643,007707,007714,007730]
      6512502 standard- siesta_g piechota  R    7:03:55      6 nid[006493,007119-007120,007271,007285,007640]
      6493766 standard- annealin rluukkon  R   14:56:35     64 nid[006518,006523-007807,007837-007840,007851]
      6510082 standard- md_NCs   fallahzo  R    9:31:19      1 nid007636

very long list ...
```

EuroHPC
Joint Undertaking

kozowski@uan03:/scratch/project_465000991> squeue | wc
  3163   25304 256268

kozowski@uan03:/scratch/project_465000991> squeue
```
       JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
     6512730 standard- siesta_g piechota  R    6:50:11      7
nid[006488,006548,007225,007635,007735,007766,007772]
      6512719 standard- siesta_g piechota  R    6:57:05      7
nid[006492,006729,007552,007643,007707,007714,007730]
      6512502 standard- siesta_g piechota  R    7:03:55      6 nid[006493,007119-007120,007271,007285,007640]
      6493766 standard- annealin rluukkon  R   14:56:35     64 nid[006518,006523-007807,007837-007840,007851]
      6510082 standard- md_NCs   fallahzo  R    9:31:19      1 nid007636
```

kozowski@uan03:/scratch/project_465000991> **squeue --me|wc**
    310   2480   24277

EuroHPC
Joint Undertaking

```
kozowski@uan03:/scratch/project_465000991> squeue | wc
  3163   25304  256268

kozowski@uan03:/scratch/project_465000991> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           6512730 standard- siesta_g piechota  R    6:50:11      7
nid[006488,006548,007225,007635,007735,007766,007772]
           6512719 standard- siesta_g piechota  R    6:57:05      7
nid[006492,006729,007552,007643,007707,007714,007730]
           6512502 standard- siesta_g piechota  R    7:03:55      6 nid[006493,007119-007120,007271,007285,007640]
           6493766 standard- annealin rluukkon  R   14:56:35     64 nid[006518,006523-007807,007837-007840,007851]
           6510082 standard- md_NCs   fallahzo  R    9:31:19      1 nid007636

kozowski@uan03:/scratch/project_465000991> squeue --me|wc
    310    2480   24277

kozowski@uan03:/scratch/project_465000991> squeue --me | tail -n 5
           6498977  standard 09-sse10 kozowski  R 1-01:33:10      1 nid001864
           6498975  standard 08-sse10 kozowski  R 1-01:33:21      1 nid001641
           6498973  standard 07-sse10 kozowski  R 1-01:33:30      1 nid001793
           6498968  standard 06-sse10 kozowski  R 1-01:33:41      1 nid001792
           6498966  standard 05-sse10 kozowski  R 1-01:33:50      1 nid002933
```

EuroHPC
Joint Undertaking

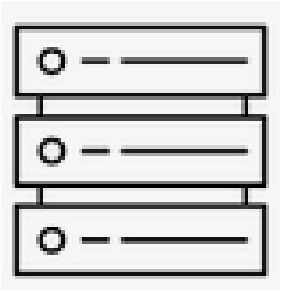
```
kozowski@uan03:/scratch/project_465000991> squeue | wc
  3163   25304  256268

kozowski@uan03:/scratch/project_465000991> squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
           6512730 standard- siesta_g piechota  R    6:50:11      7
nid[006488,006548,007225,007635,007735,007766,007772]
           6512719 standard- siesta_g piechota  R    6:57:05      7
nid[006492,006729,007552,007643,007707,007714,007730]
           6512502 standard- siesta_g piechota  R    7:03:55      6 nid[006493,007119-007120,007271,007285,007640]
           6493766 standard- annealin rluukkon  R   14:56:35     64 nid[006518,006523-007807,007837-007840,007851]
           6510082 standard- md_NCs   fallahzo  R    9:31:19      1 nid007636

kozowski@uan03:/scratch/project_465000991> squeue --me|wc
    310    2480   24277

kozowski@uan03:/scratch/project_465000991> squeue --me | tail -n 5
           6498977  standard 09-sse10 kozowski  R 1-01:33:10      1 nid001864
           6498975  standard 08-sse10 kozowski  R 1-01:33:21      1 nid001641
           6498973  standard 07-sse10 kozowski  R 1-01:33:30      1 nid001793
           6498968  standard 06-sse10 kozowski  R 1-01:33:41      1 nid001792
           6498966  standard 05-sse10 kozowski  R 1-01:33:50      1 nid002933

kozowski@uan03:~> squeue --me|grep ' R' |grep standard|
    100     800    7800
```

EuroHPC
Joint Undertaking

```
kozowski@uan03:/scratch/project_465000991> squeue --me | tail -n 5
           6498977  standard 09-sse10 kozowski  R 1-01:33:10      1 nid001864
           6498975  standard 08-sse10 kozowski  R 1-01:33:21      1 nid001641
           6498973  standard 07-sse10 kozowski  R 1-01:33:30      1 nid001793
           6498968  standard 06-sse10 kozowski  R 1-01:33:41      1 nid001792
           6498966  standard 05-sse10 kozowski  R 1-01:33:50      1 nid002933

kozowski@uan03:~> squeue --me|grep ' R' |grep standard|
    100      800     7800
```

EuroHPC
Joint Undertaking

LUMI



128 = **12.8 k CPUs**

NODE

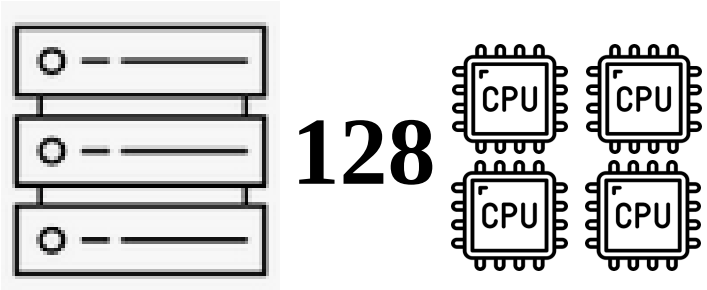
```
kozowski@uan03:/scratch/project_465000991> squeue --me | tail -n 5
           6498977  standard 09-sse10 kozowski  R 1-01:33:10      1 nid001864
           6498975  standard 08-sse10 kozowski  R 1-01:33:21      1 nid001641
           6498973  standard 07-sse10 kozowski  R 1-01:33:30      1 nid001793
           6498968  standard 06-sse10 kozowski  R 1-01:33:41      1 nid001792
           6498966  standard 05-sse10 kozowski  R 1-01:33:50      1 nid002933

kozowski@uan03:~> squeue --me|grep ' R' |grep standard|
    100      800     7800
```

EuroHPC Joint Undertaking

PL GRID

LUMI

LUMI



NODE

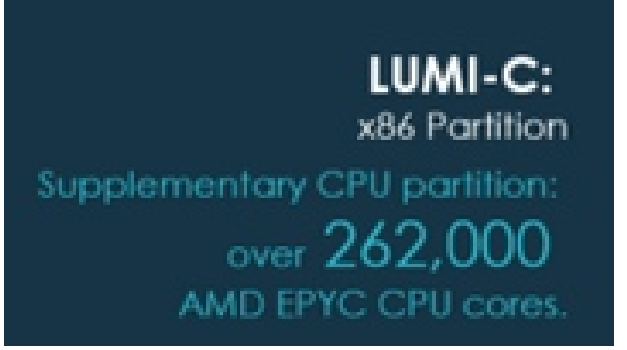
128 CPU x 100N x 48h = 614k CPUh

```
kozowski@uan03:/scratch/project_465000991> squeue --me | tail -n 5
           6498977  standard 09-sse10 kozowski  R 1-01:33:10      1 nid001864
           6498975  standard 08-sse10 kozowski  R 1-01:33:21      1 nid001641
           6498973  standard 07-sse10 kozowski  R 1-01:33:30      1 nid001793
           6498968  standard 06-sse10 kozowski  R 1-01:33:41      1 nid001792
           6498966  standard 05-sse10 kozowski  R 1-01:33:50      1 nid002933

kozowski@uan03:~> squeue --me|grep ' R' |grep standard|
     100      800     7800
```

EuroHPC Joint Undertaking

PL GRID

LUMI

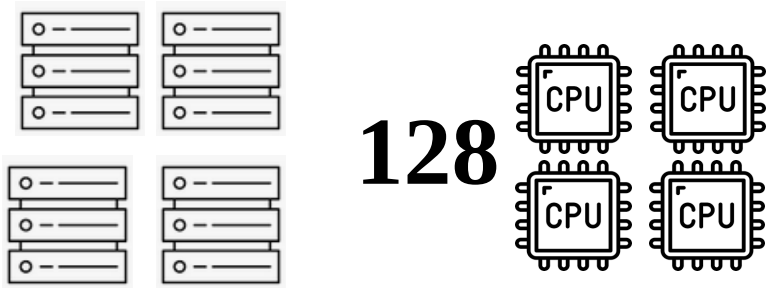


kozowski@uan04:> **scontrol show partition standard**
PartitionName=standard
AllowGroups=ALL AllowAccounts=ALL AllowQos=ALL
AllocNodes=ALL Default=NO QoS=N/A
DefaultTime=00:01:00 DisableRootJobs=YES ExclusiveUser=NO
MaxNodes=256 MaxTime=2-00:00:00 MinNodes=1 LLN=NO
MaxCPUsPerNode=UNLIMITED
Nodes=nid[001512-002023,002696-003047]
PriorityJobFactor=100 PriorityTier=1 RootOnly=NO ReqResv=NO
OverSubscribe=EXCLUSIVE
OverTimeLimit=NONE PreemptMode=OFF
State=UP **TotalCPUs=221184** TotalNodes=864
SelectTypeParameters=NONE
JobDefaults=(null)
DefMemPerCPU=UNLIMITED MaxMemPerNode=UNLIMITED
TRES=cpu=221184,mem=189T,node=864,billing=221184

EuroHPC
Joint Undertaking

LUMI

1 job on many nodes



NODE x 4

EuroHPC Joint Undertaking

PL GRID

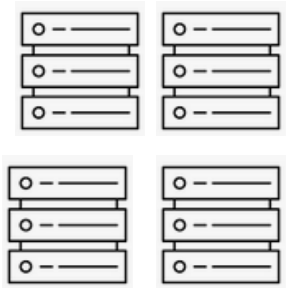
LUMI

1 job on many nodes

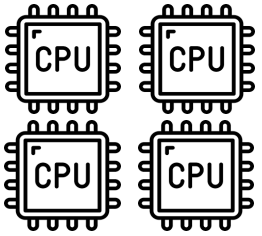
```
#!/bin/bash -l
#SBATCH -J a40_41_42_43
#SBATCH -N 4
#SBATCH --ntasks-per-node=128
#SBATCH --cpus-per-task=1
#SBATCH --time=48:00:00
#SBATCH --account=project_465000991  # Project for billing
#SBATCH --partition=standard  # Partition (queue) name
#SBATCH --output="/scratch/project_465000991/logs/a40_41_42_43.out"
#SBATCH --error="/tmp/a40_41_42_43.err"

module add cray-python/3.9.12.1
cd /scratch/project_465000991/

for ((i=0; i<128; i++)); do
  srun --exclusive -n 1 -c 1 python3 run_LUMI.py 40 $i &
done
for ((i=0; i<128; i++)); do
  srun --exclusive -n 1 -c 1 python3 run_LUMI.py 41 $i &
done
for ((i=0; i<128; i++)); do
  srun --exclusive -n 1 -c 1 python3 run_LUMI.py 42 $i &
done
for ((i=0; i<128; i++)); do
  srun --exclusive -n 1 -c 1 python3 run_LUMI.py 43 $i &
done

wait
```

LUMI

1 job on many nodes



128



=

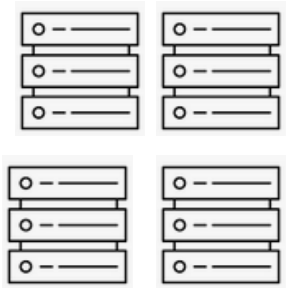
51.2k CPUs

NODE x 4

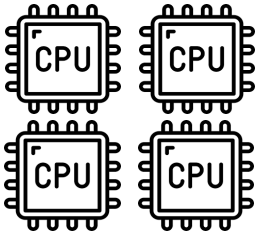
EuroHPC
Joint Undertaking

LUMI

1 job on many nodes



128



=

51.2k CPUs

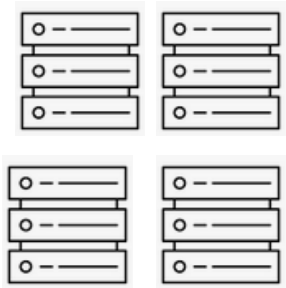
NODE x 4

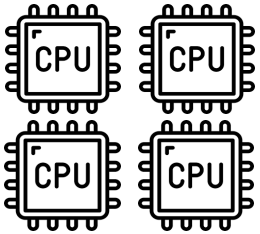
128 CPU x 100 x 4N x 48h = 2.5M CPUh

EuroHPC
Joint Undertaking

LUMI

1 job on many nodes



128 

= **51.2k CPUs**

NODE x 4

128 CPU x 100 x 4N x 48h = 2.5M CPUh

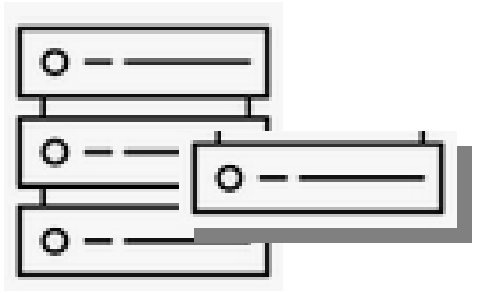
```
kozowski@uan03:~> squeue --me | tail -n 5
  JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
6529497  standard 10_11_12 kozowski  R   14:42:02      4 nid[001988-001991]
6529496  standard 0c_0d_0e kozowski  R   14:42:13      4 nid[001704-001707]
6529495  standard 08_09_0a kozowski  R   14:42:24      4 nid[001667-001670]
6529494  standard 04_05_06 kozowski  R   14:42:35      4 nid[001648-001651]
6529493  standard 00_01_02 kozowski  R   14:43:43      4 nid[001552-001555]

kozowski@uan03:~> squeue --me|grep ' R' |grep standard|wc
100     800    7800
```

EuroHPC
Joint Undertaking

PL GRID

LUMI

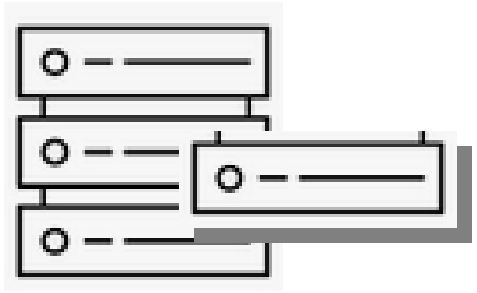
LUMI



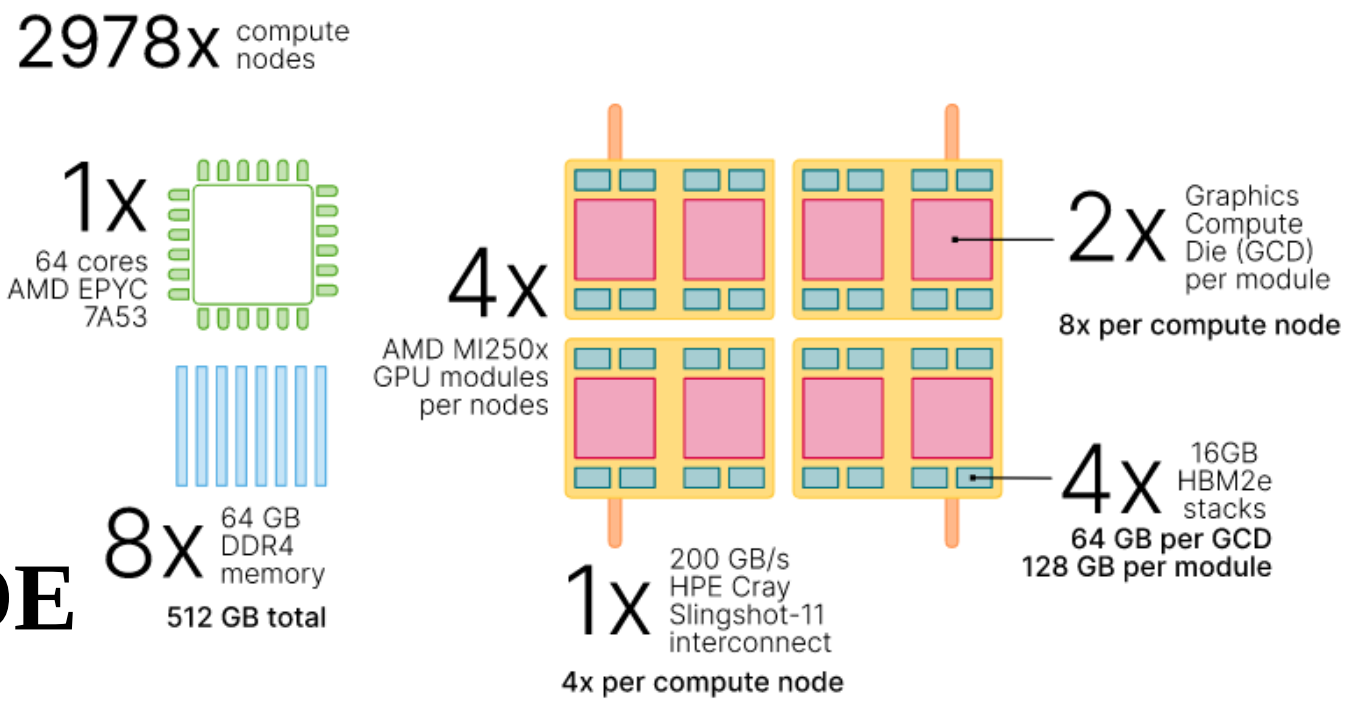
Part of NODE

```
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|wc
    200    1600   15600
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|head
           6440659   small-g  2250_12 kozowski  R 1-20:18:07      1 nid005040
           6440657   small-g 12220_12 kozowski  R 1-20:32:42      1 nid005038
           6440656   small-g 12190_12 kozowski  R 1-20:50:54      1 nid005047
           6440653   small-g 12160_12 kozowski  R 1-20:58:31      1 nid005058
           6440648   small-g 12130_12 kozowski  R 1-21:01:26      1 nid005058
           6440647   small-g 12100_12 kozowski  R 1-21:04:25      1 nid005047
           6440644   small-g 12070_12 kozowski  R 1-21:09:24      1 nid005028
           6440840   small-g 12280_12 kozowski  R 1-20:17:59      1 nid005047
           6445387   small-g 12310_12 kozowski  R 1-18:41:57      1 nid005090
           6490960   small-g 18340_18 kozowski  R 1-04:33:58      1 nid005111
```

LUMI

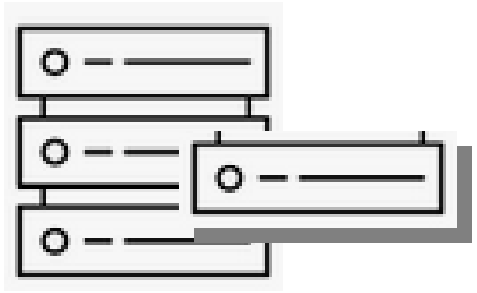


Part of NODE

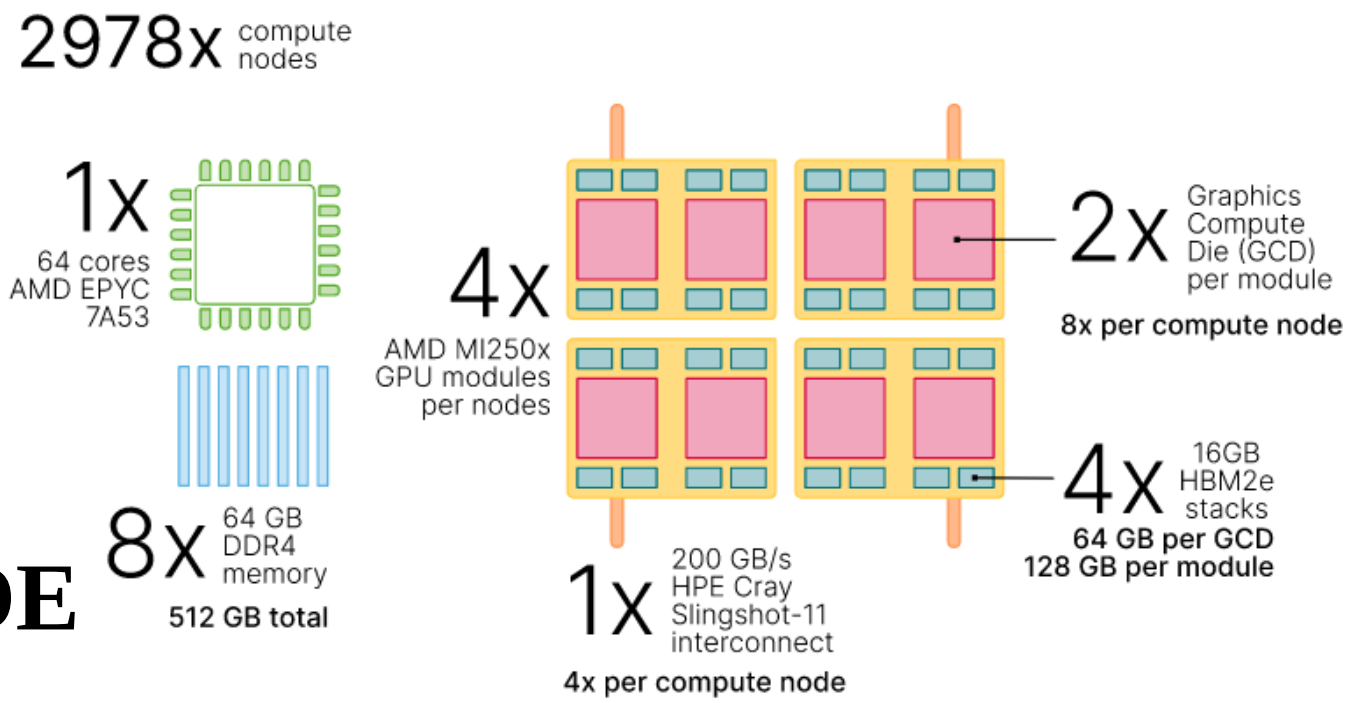



```
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|wc
    200    1600   15600
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|head
           6440659   small-g  2250_12 kozowski  R 1-20:18:07      1 nid005040
           6440657   small-g 12220_12 kozowski  R 1-20:32:42      1 nid005038
           6440656   small-g 12190_12 kozowski  R 1-20:50:54      1 nid005047
           6440653   small-g 12160_12 kozowski  R 1-20:58:31      1 nid005058
           6440648   small-g 12130_12 kozowski  R 1-21:01:26      1 nid005058
           6440647   small-g 12100_12 kozowski  R 1-21:04:25      1 nid005047
           6440644   small-g 12070_12 kozowski  R 1-21:09:24      1 nid005028
           6440840   small-g 12280_12 kozowski  R 1-20:17:59      1 nid005047
           6445387   small-g 12310_12 kozowski  R 1-18:41:57      1 nid005090
           6490960   small-g 18340_18 kozowski  R 1-04:33:58      1 nid005111
```

LUMI



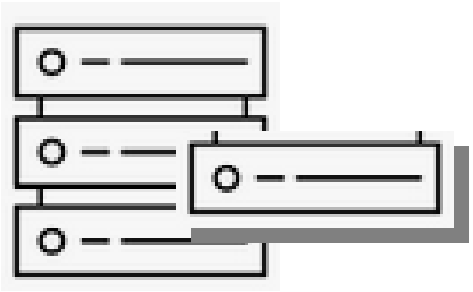
Part of NODE



small-g (allocate by resources, thus “only” 200 GPUs in this batch)

```
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|wc
    200    1600   15600
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|head
           6440659   small-g  2250_12 kozowski  R 1-20:18:07      1 nid005040
           6440657   small-g 12220_12 kozowski  R 1-20:32:42      1 nid005038
           6440656   small-g 12190_12 kozowski  R 1-20:50:54      1 nid005047
           6440653   small-g 12160_12 kozowski  R 1-20:58:31      1 nid005058
           6440648   small-g 12130_12 kozowski  R 1-21:01:26      1 nid005058
           6440647   small-g 12100_12 kozowski  R 1-21:04:25      1 nid005047
           6440644   small-g 12070_12 kozowski  R 1-21:09:24      1 nid005028
           6440840   small-g 12280_12 kozowski  R 1-20:17:59      1 nid005047
           6445387   small-g 12310_12 kozowski  R 1-18:41:57      1 nid005090
           6490960   small-g 18340_18 kozowski  R 1-04:33:58      1 nid005111
```

LUMI

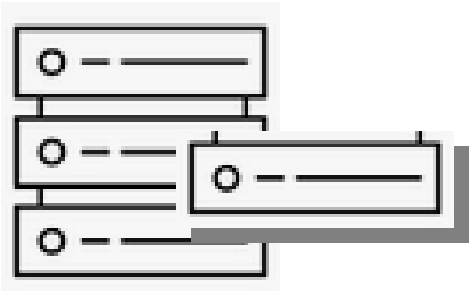


Part of NODE

small-g (allocate by resources, thus “only” 200 GPUs in this batch)

```
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|wc
    200    1600   15600
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|head
           6440659   small-g  2250_12 kozowski  R 1-20:18:07      1 nid005040
           6440657   small-g 12220_12 kozowski  R 1-20:32:42      1 nid005038
           6440656   small-g 12190_12 kozowski  R 1-20:50:54      1 nid005047
           6440653   small-g 12160_12 kozowski  R 1-20:58:31      1 nid005058
           6440648   small-g 12130_12 kozowski  R 1-21:01:26      1 nid005058
           6440647   small-g 12100_12 kozowski  R 1-21:04:25      1 nid005047
           6440644   small-g 12070_12 kozowski  R 1-21:09:24      1 nid005028
           6440840   small-g 12280_12 kozowski  R 1-20:17:59      1 nid005047
           6445387   small-g 12310_12 kozowski  R 1-18:41:57      1 nid005090
           6490960   small-g 18340_18 kozowski  R 1-04:33:58      1 nid005111
```

LUMI



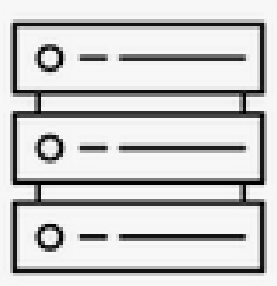
1 GPU * 200 * 72h = 14,400 GPUh

Part of NODE

small-g (allocate by resources, thus “only” 200 GPUs in this batch)

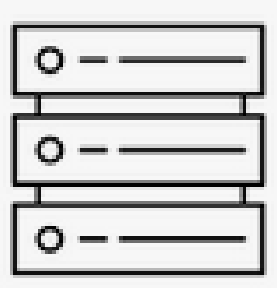
```
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|wc
    200    1600   15600
kozowski@uan03:~> squeue --me|grep ' R' |grep small-g|head
           6440659   small-g  2250_12 kozowski  R 1-20:18:07      1 nid005040
           6440657   small-g 12220_12 kozowski  R 1-20:32:42      1 nid005038
           6440656   small-g 12190_12 kozowski  R 1-20:50:54      1 nid005047
           6440653   small-g 12160_12 kozowski  R 1-20:58:31      1 nid005058
           6440648   small-g 12130_12 kozowski  R 1-21:01:26      1 nid005058
           6440647   small-g 12100_12 kozowski  R 1-21:04:25      1 nid005047
           6440644   small-g 12070_12 kozowski  R 1-21:09:24      1 nid005028
           6440840   small-g 12280_12 kozowski  R 1-20:17:59      1 nid005047
           6445387   small-g 12310_12 kozowski  R 1-18:41:57      1 nid005090
           6490960   small-g 18340_18 kozowski  R 1-04:33:58      1 nid005111
```

LUMI



What is better than 200 GPUs?

LUMI

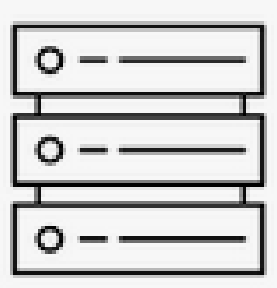


What is better than 200 GPUs?

200 NODEs with 8 GPUs each

```
kozowski@uan03:~> squeue --me|grep ' R' |grep standard-|wc
    200    1600   15600
kozowski@uan03:~> squeue --me|grep ' R' |grep standard-|head
           6712709 standard- 200_20_m kozowski  R   20:37:15      1 nid007090
           6712700 standard- 199_20_m kozowski  R   20:37:45      1 nid007035
           6712686 standard- 198_20_m kozowski  R   20:38:14      1 nid006576
           6712673 standard- 197_20_m kozowski  R   20:38:43      1 nid006674
           6712666 standard- 196_20_m kozowski  R   20:39:17      1 nid006962
           6712660 standard- 195_20_m kozowski  R   20:39:47      1 nid006989
           ...
```

LUMI



8 GPU * 200 * 48h = 76,800 GPUh

What is better than 200 GPUs?

200 NODEs with 8 GPUs each

```
kozowski@uan03:~> squeue --me|grep ' R' |grep standard-|wc
    200    1600   15600
kozowski@uan03:~> squeue --me|grep ' R' |grep standard-|head
           6712709 standard- 200_20_m kozowski  R   20:37:15      1 nid007090
           6712700 standard- 199_20_m kozowski  R   20:37:45      1 nid007035
           6712686 standard- 198_20_m kozowski  R   20:38:14      1 nid006576
           6712673 standard- 197_20_m kozowski  R   20:38:43      1 nid006674
           6712666 standard- 196_20_m kozowski  R   20:39:17      1 nid006962
           6712660 standard- 195_20_m kozowski  R   20:39:47      1 nid006989
           ...
```

Calculate your budget and multiply x2

A back-of-the-envelope calculation shows that you need 100k CPUh
than ask in the grant for 200k CPUh

expect some problems with the cluster, longer queue times, slower hardware/run-times than expected, extra calculations you did not think about earlier, etc.

Calculate your budget and multiply x2

A back-of-the-envelope calculation shows that you need 100k CPUh
than ask in the grant for 200k CPUh

expect some problems with the cluster, longer queue times, slower hardware/run-times than expected, extra calculations you did not think about earlier, etc.

The budget units in most HPC systems are with:

CPU hours & GPU hours & TB hours

only occasionally on smaller systems this could be GPUmin (e.g. ENTROPY@MIMUW) or NODEhours on bigger ones (e.g. LUMI, SUMMIT)

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

No matter what machine you want to use
check ONLY partitions/queue/quota per user

Nobody will give you access to full cluster even for one day

Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

No matter what machine you want to use
check ONLY partitions/queue/quota per user

Nobody will give you access to full cluster even for one day

Nobody will extend the limits for you*

* **you can ask for more CPU/GPU/TB hours, but usually this will take days/weeks**
partitions/queue limits are hard-coded thus do not even bother to ask

Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

No matter what machine you want to use
check ONLY partitions/queue/quota per user

Nobody will give you access to full cluster even for one day

Nobody will extend the limits for you*

* you can ask for more CPU/GPU/TB hours, but usually this will take days/weeks
partitions/queue limits are hard-coded thus do not even bother to ask

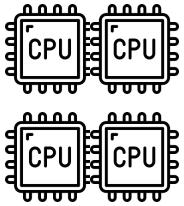
Computing for more than 1-2 weeks in any HPC cluster will only make pain for you and the admins

Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

Example:

6,244 

150k CPUh/24h

Topola	HPC cluster, PL-Grid cluster	Intel Haswell Huawei E9000	223	28 cores, 64/128 GB RAM

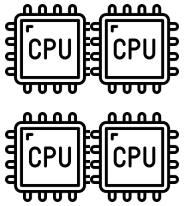
Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

Example:

icm

6,244 

150k CPUh/24h

Topola	HPC cluster, PL-Grid cluster	Intel Haswell Huawei E9000	223	28 cores, 64/128 GB RAM

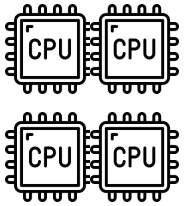
Limit for the user (64 jobs x 2 CPU)

Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

Example:

6,244 

150k CPUh/24h

Topola	HPC cluster, PL-Grid cluster	Intel Haswell Huawei E9000	223	28 cores, 64/128 GB RAM

Limit for the user (64 jobs x 2 CPU)

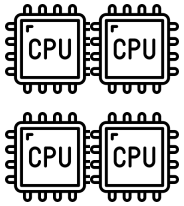
128 CPUs x 24h * 14 days = 43,008 CPUh

Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

Example:

6,244 

150k CPUh/24h

Topola	HPC cluster, PL-Grid cluster	Intel Haswell Huawei E9000	223	28 cores, 64/128 GB RAM

Limit for the user (64 jobs x 2 CPU)

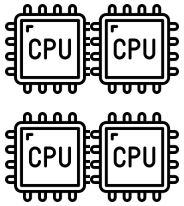
Theoretical calculation
128 CPUs x 24h * 14 days = 43,008 CPUh

Rules of HPC

Rule of thumb for choosing the proper HPC system

Your calculations should be done in one-two weeks

Example:

6,244 

150k CPUh/24h

Topola	HPC cluster, PL-Grid cluster	Intel Haswell Huawei E9000	223	28 cores, 64/128 GB RAM

Limit for the user (64 jobs x 2 CPU)

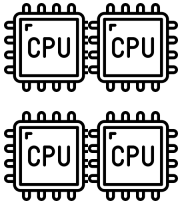
Theoretical calculation
128 CPUs x 24h * 14 days = 43,008 CPUh

Add time for crashes, waiting in the queue
Expected yield: 10-20k CPUh

Rules of HPC

Rule of thumb for choosing the proper HPC system

Reverse estimation of expected yield:



CPUs in the queue x 100

Limit for the user (64 jobs x 2 CPU)

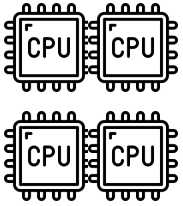
Theoretical calculation
128 CPUs x 24h * 14 days = 43,008 CPUh

Add time for crashes, waiting in the queue
Expected yield: 10-20k CPUh

Rules of HPC

Rule of thumb for choosing the proper HPC system

Reverse estimation of expected yield:



CPUs in the queue x 100

128 x 100 = 12.8k CPUh

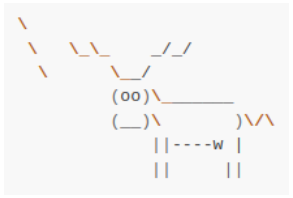
Limit for the user (64 jobs x 2 CPU)

Theoretical calculation
128 CPUs x 24h * 14 days = 43,008 CPUh

Add time for crashes, waiting in the queue
Expected yield: 10-20k CPUh

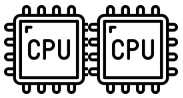
Hitchhiker's Guide to HPC system in Poland and Europe

ADP

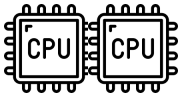


University of Warsaw
MIM
Faculty of Mathematics, Informatics and Mechanics

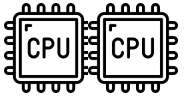
10k-30k CPUh
1-2k GPUh



10-20k CPUh



100k-1M CPUh
10-20k GPUh

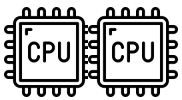


50-100k GPUh

1TB

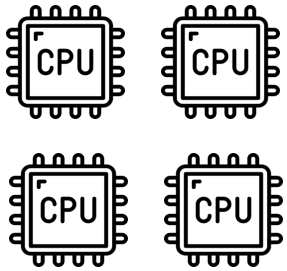
LUMI

up to 20M CPUh
up to 1M GPUh



4TB

HPC vs Cloud



~0.096 CPUh

0.70-0.96 GPUh

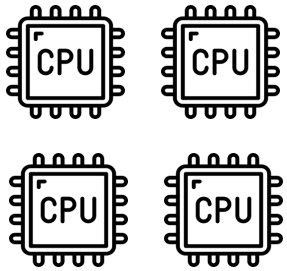
aws

Google Cloud

0.70-0.96 GPUh

	Instance Size	GPU	GPU Memory (GiB)	vCPUs	Memory (GiB)	Storage (GB)	Network Bandwidth (Gbps)	EBS Bandwidth (Gbps)	On Demand Price/hr*	1-yr ISP Effective Hourly (Linux)	3-yr ISP Effective Hourly (Linux)
Single GPU VMs	g6.xlarge	1	24	4	16	1x250	Up to 10	Up to 5	$0.805	$0.499	$0.342
	g6.2xlarge	1	24	8	32	1x450	Up to 10	Up to 5	$0.978	$0.606	$0.416
	g6.4xlarge	1	24	16	64	1x600	Up to 25	8	$1.323	$0.820	$0.562
	g6.8xlarge	1	24	32	128	2x450	25	16	$2.014	$1.249	$0.856
	g6.16xlarge	1	24	64	256	2x940	25	20	$3.397	$2.106	$1.443
	gr6.4xlarge	1	24	16	128	1x600	Up to 25	8	$1.539	$0.954	$0.654
	gr6.8xlarge	1	24	32	256	2x450	25	16	$2.446	$1.517	$1.040
Multi GPU VMs	g6.12xlarge	4	96	48	192	4x940	40	20	$4.602	$2.853	$1.955
	g6.24xlarge	4	96	96	384	4x940	50	30	$6.675	$4.139	$2.837
	g6.48xlarge	8	192	192	768	8x940	100	60	$13.35	$8.277	$5.674

HPC vs Cloud

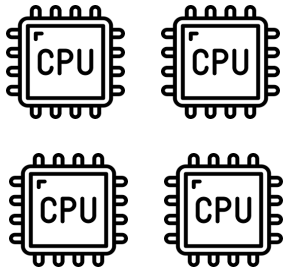


~0.096 CPUh

0.70-0.96 GPUh

aws

Google Cloud



~0.096 CPUh

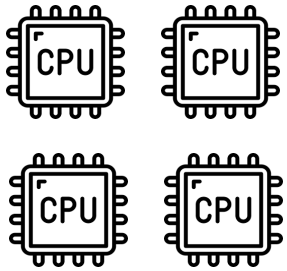
0.70-0.96 GPUh

aws

Google Cloud

In my last bioinformatics project I used:

- **1M GPUh**
- **15M CPUh**



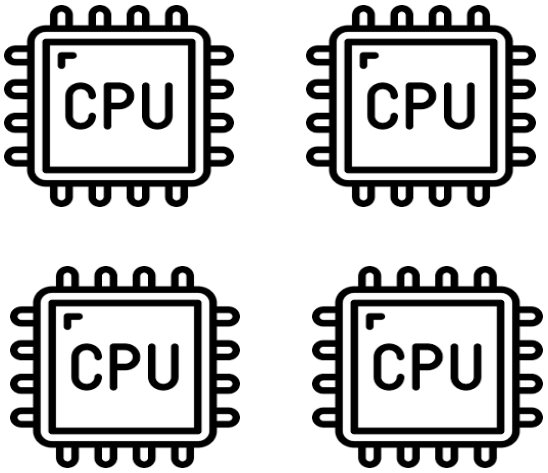
~0.096 CPUh

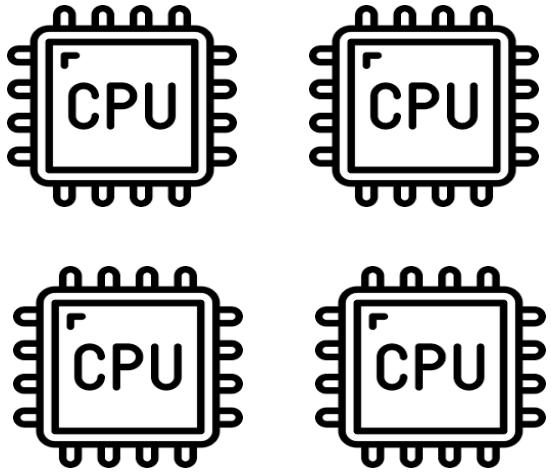
0.70-0.96 GPUh

In my last bioinformatics project I used:

- 1M GPUh
- 15M CPUh

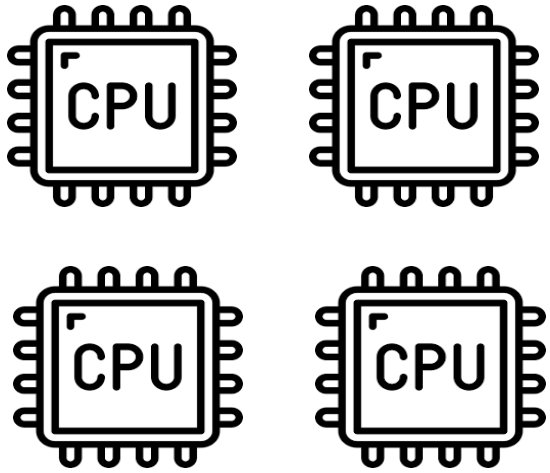
1M * 0.7 + 15M * 0.096 = $2.14M


CPU
CPU
CPU
CPU



AMD

intel

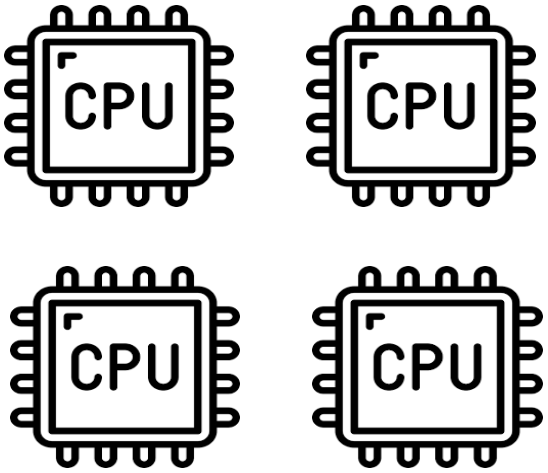

CPU
CPU
CPU
CPU


AMD

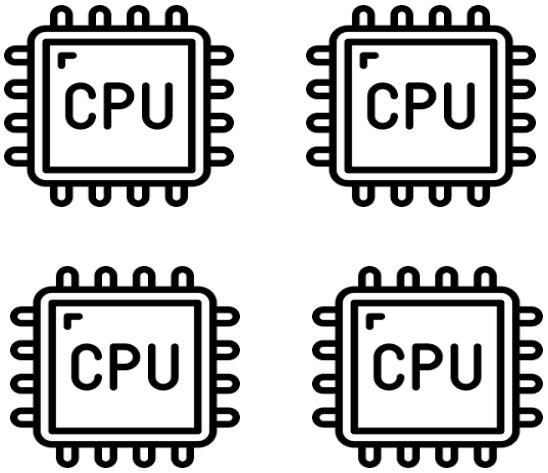

intel


AMD
EPYC


intel
Xeon
processor



AMD


CPU
CPU
CPU
CPU


GPU


AMD

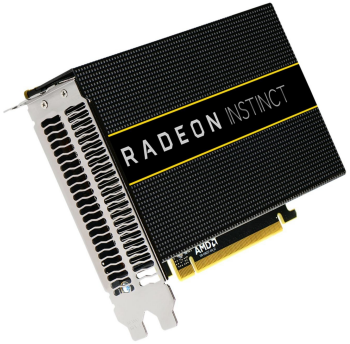

intel


AMD

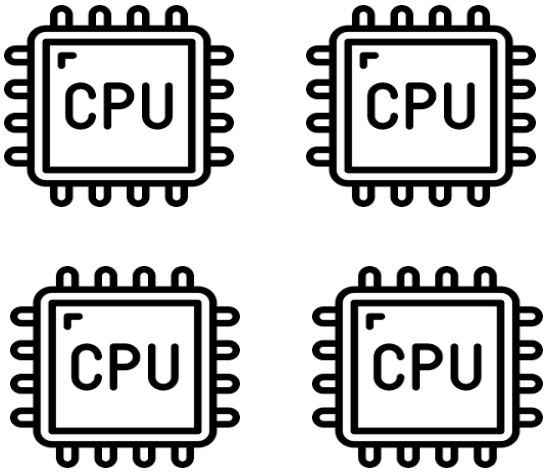

NVIDIA


AMD
EPYC


intel
Xeon
processor


RADEON INSTINCT


NVIDIA
NVIDIA
NVIDIA


CPU
CPU
CPU
CPU


GPU

AMD

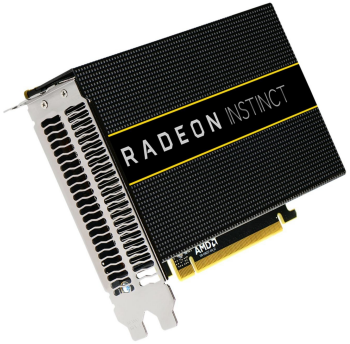

intel


AMD

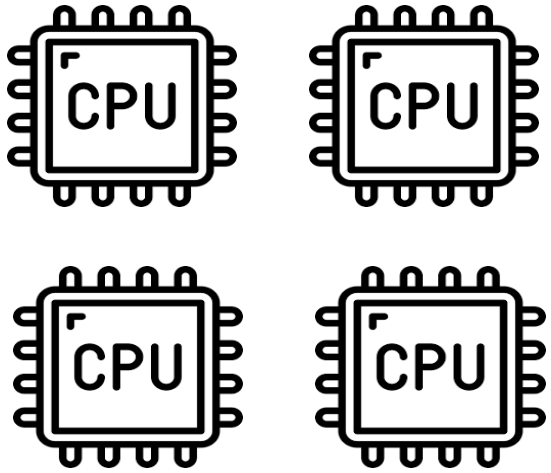
NVIDIA


AMD
EPYC


intel
Xeon
processor


RADEON INSTINCT


NVIDIA
NVIDIA
NVIDIA


CPU
CPU
CPU
CPU


GPU
NVIDIA
CUDA


AMD

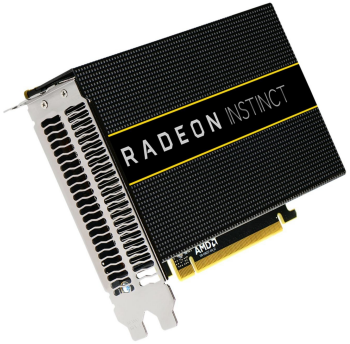

intel


AMD


NVIDIA


AMD
EPYC


intel
Xeon
processor


RADEON INSTINCT

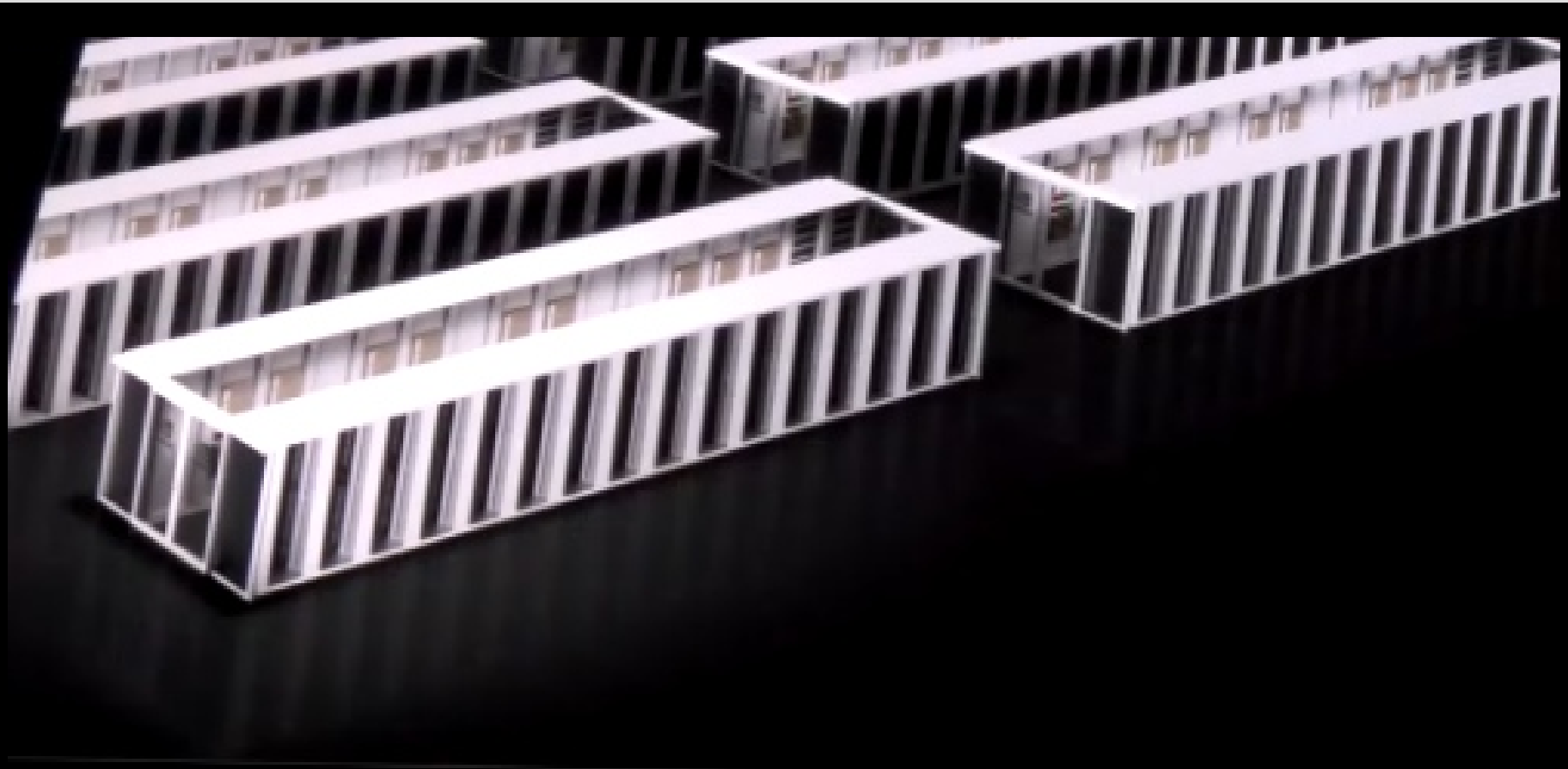

NVIDIA
NVIDIA
NVIDIA

Thank you for your time
and
See you at the next lecture

Any other
questions & comments

lukaskoz@mimuw.edu.pl

Future is NOW



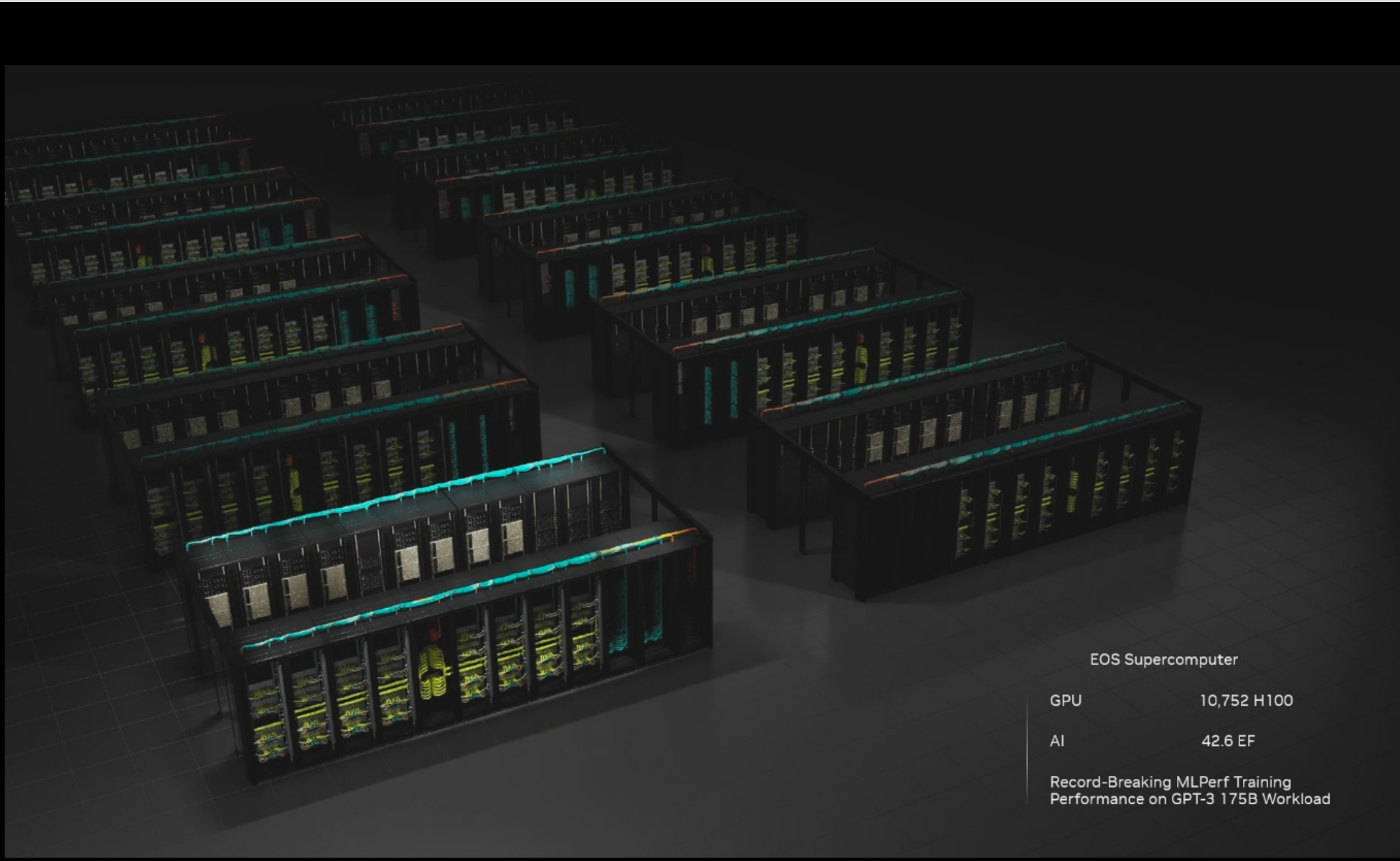
SELENE
2021

4,480 A100 GPUs

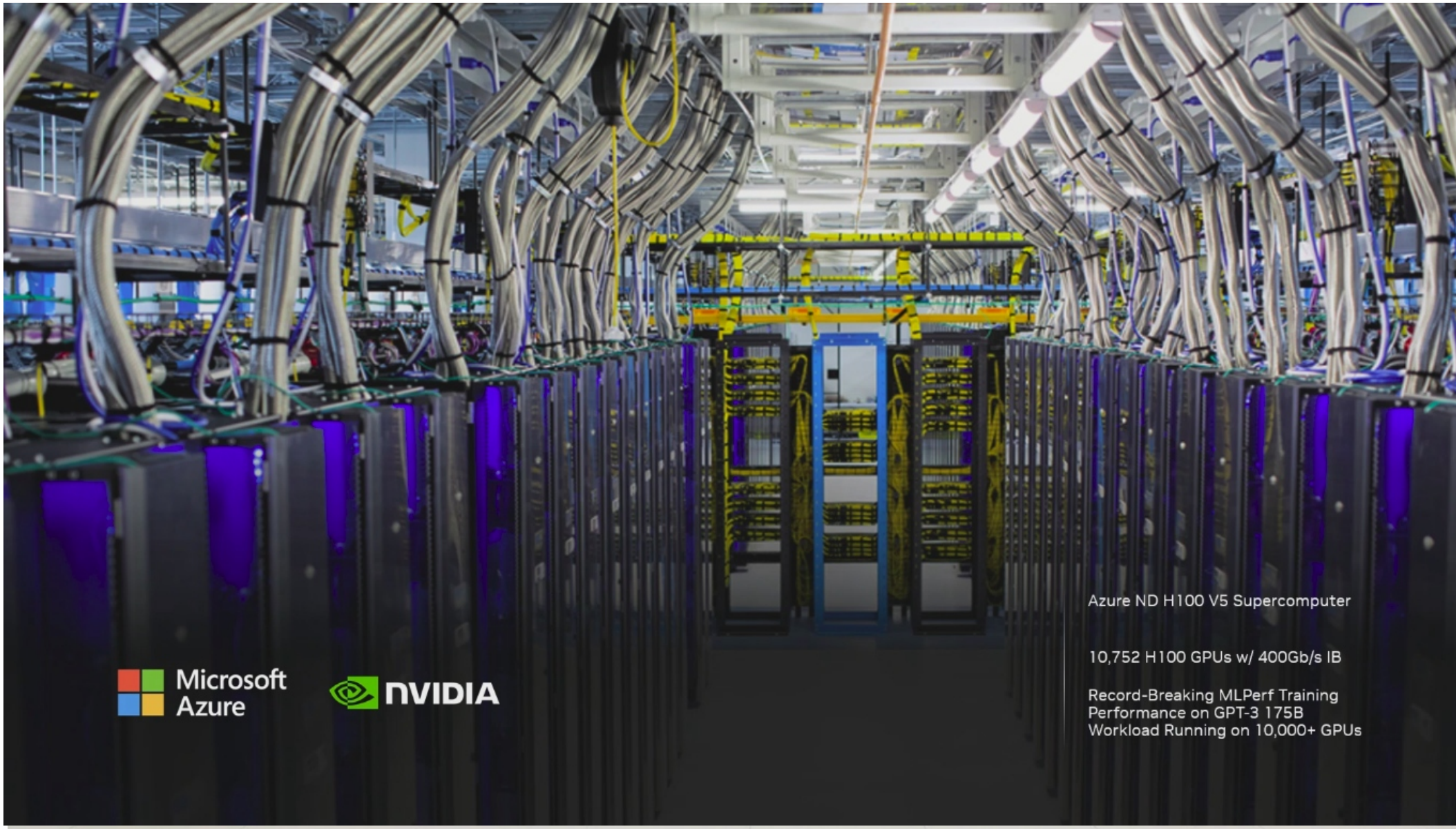
3 EF AI Compute

112 TB/s Interconnect BW

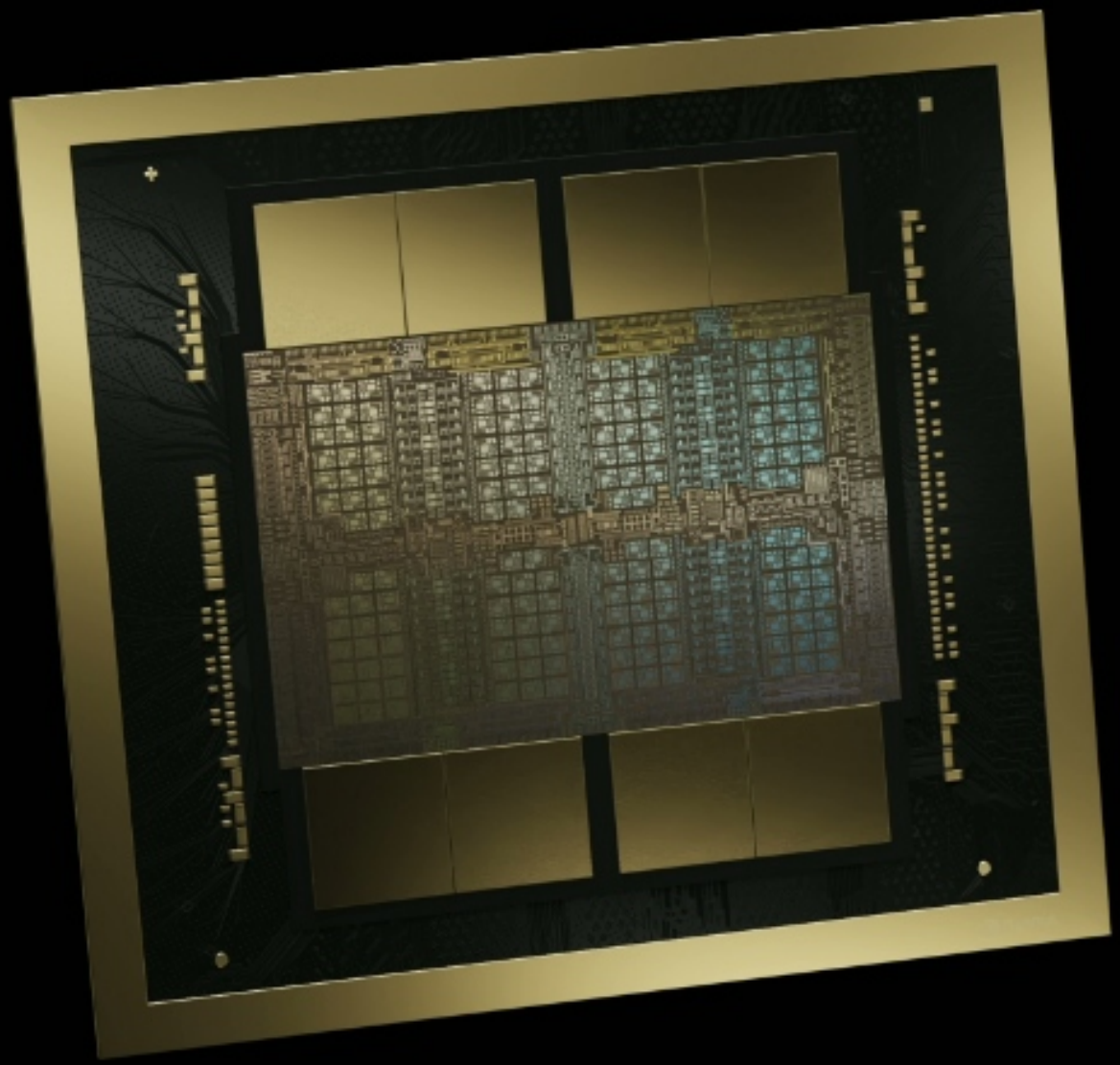
Future is NOW



Future Frontier



Future is NOW



BLACKWELL

THE ENGINE OF THE NEW INDUSTRIAL REVOLUTION

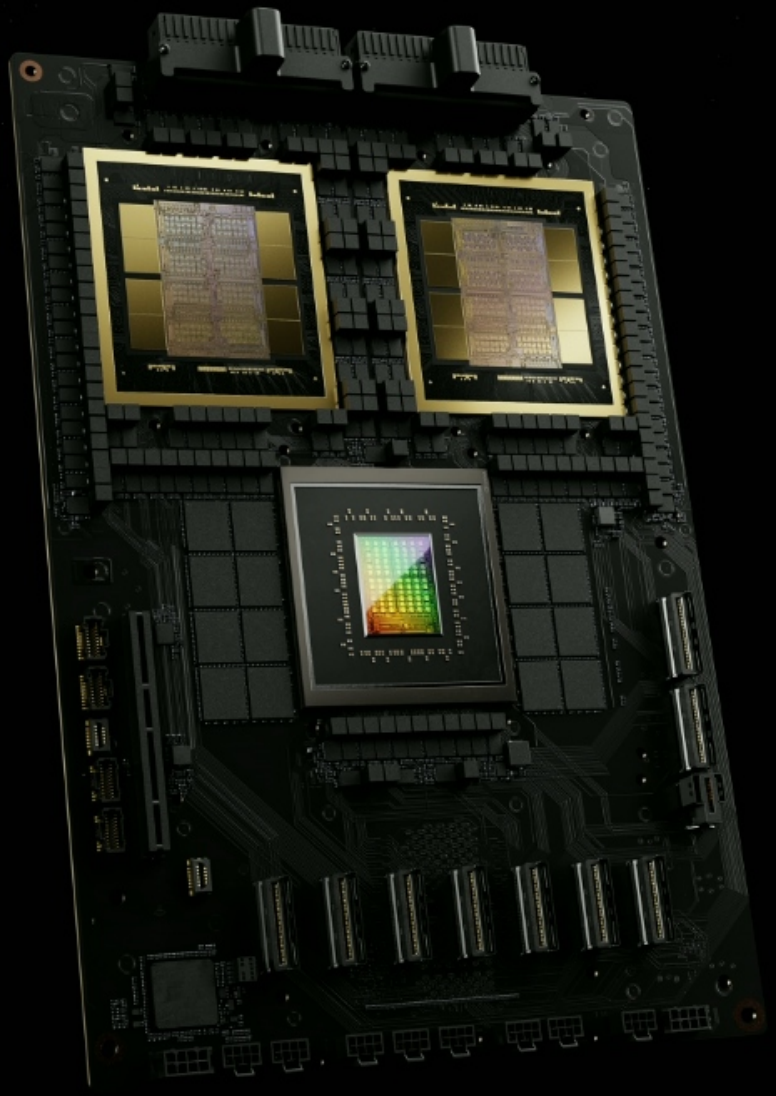
20 petaFLOPS of AI performance

192GB of HBM3e

8TB/s of memory bandwidth

Full stack, CUDA enabled

Future is NOW



	Blackwell GPU	
FP8	20 PFLOPS	2.5X Hopper
NEW FP6	20 PFLOPS	2.5X
NEW FP4	**40 PFLOPS**	**5X**
HBM Model Size	740B param	6X
HBM Bandwidth	34T param/sec	5X
NVLINK All-Reduce with SHARP	7.2 TB/s	4X

Future is NOW

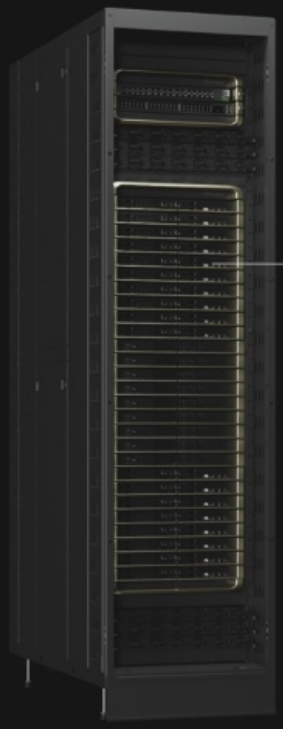
Accelerated Computing Innovation Drives Increase in Application Speedups

Blackwell 2nd generation transformer engine

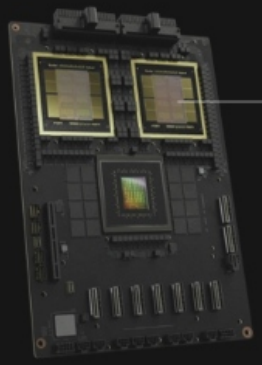
GB200 NVL72

36 GRACE CPUs
72 BLACKWELL GPUs

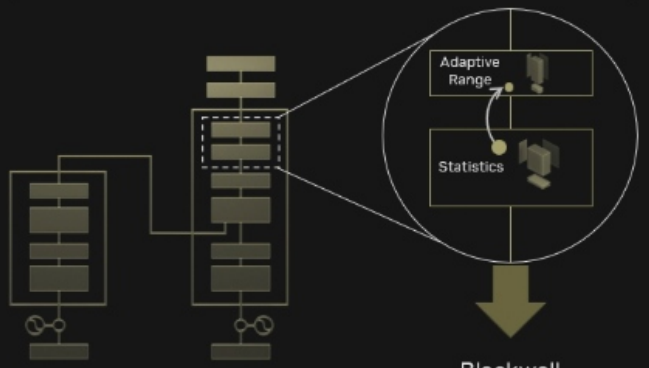
Training	720 PFLOPs
Inference	1,440 PFLOPs



BLACKWELL

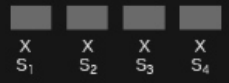


TRANSFORMER ENGINE



Blackwell

Micro-Tensor Scaling



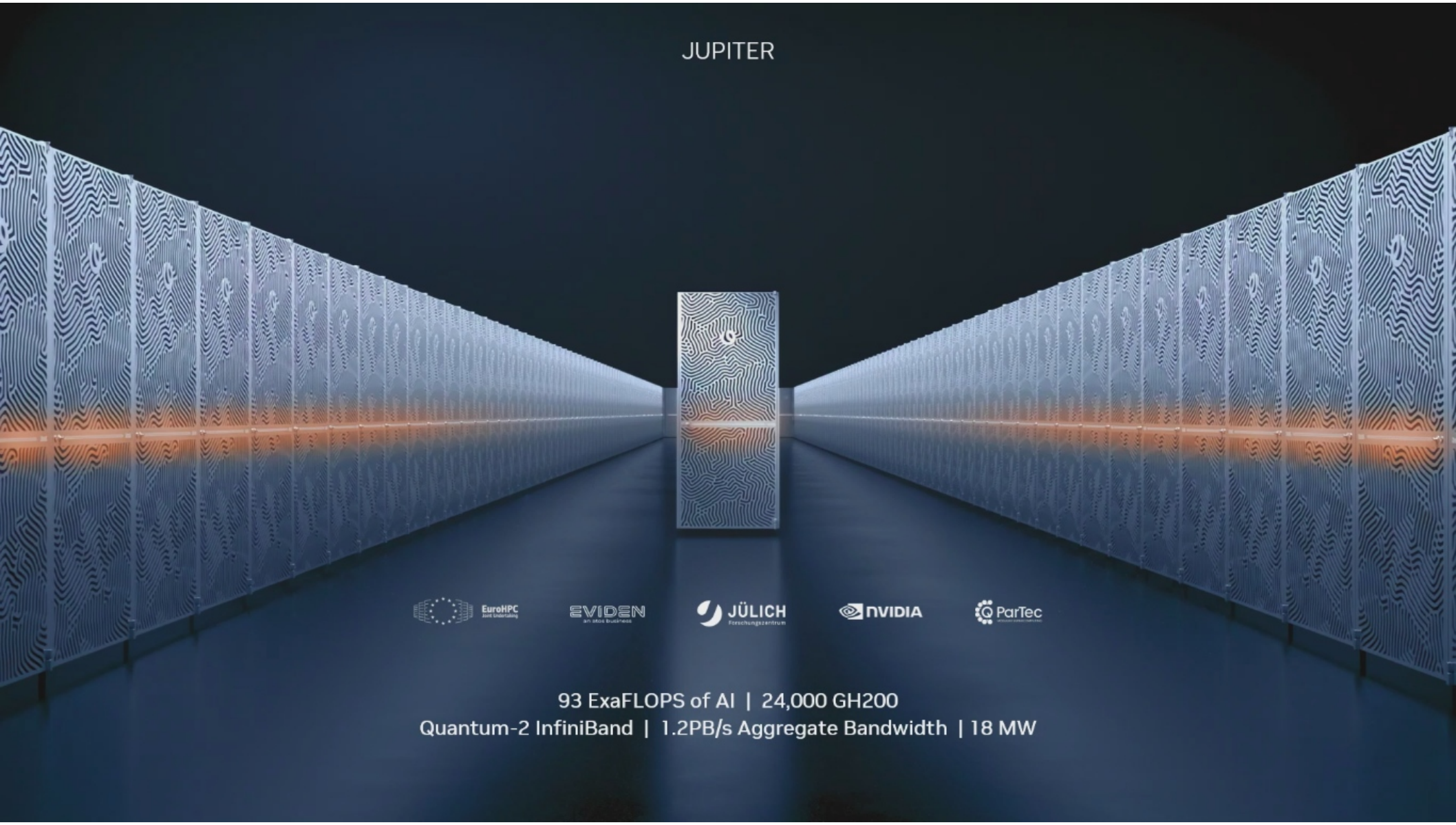
Enabling FP4 AI Inference

2x Compute

NVIDIA

Future Frontier

Future Frontier



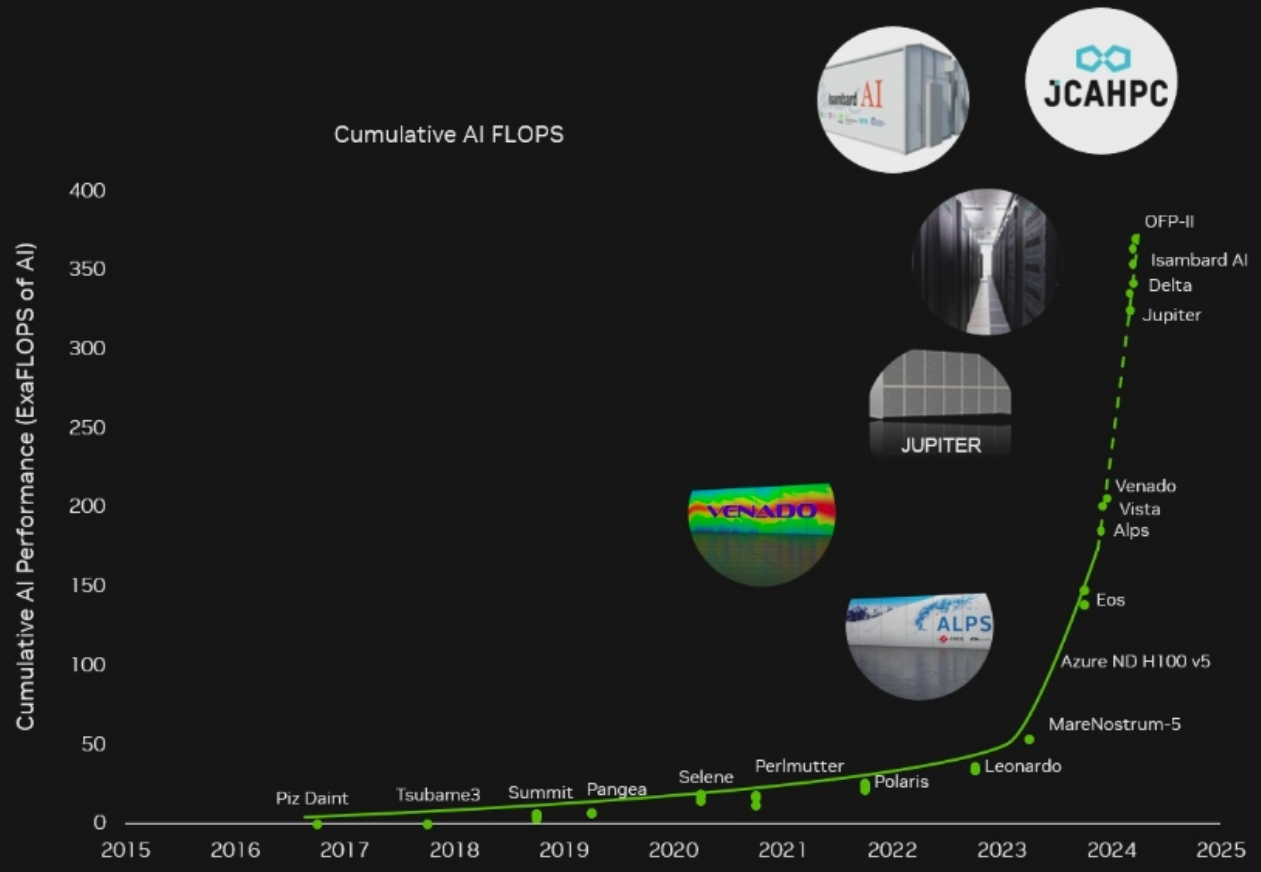
Future Frontier

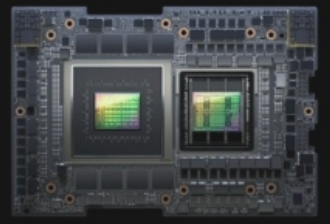
CSCS ALPS Supercomputer
5,000 Grace Hopper Superchips
40 exaFLOPS of AI

Future Frontier

Accelerated Computing Powers AI Supercomputing Datacenters

350 ExaFLOPS of Performance





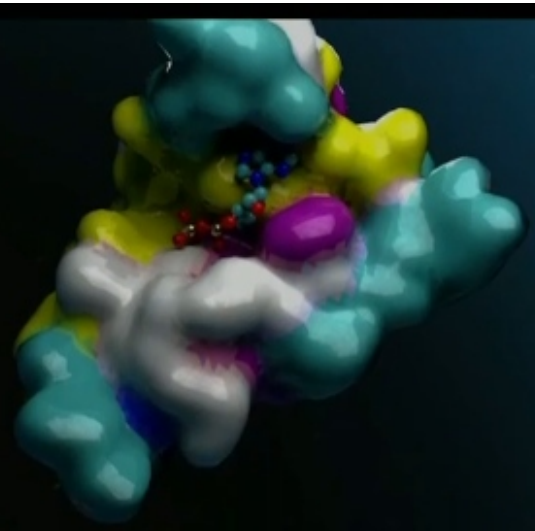
200 Exaflops AI Grace Hopper
Coming online 2024

NVIDIA

NVIDIA BioNeMo

Accelerate early drug discovery with generative AI.

Get Early Access

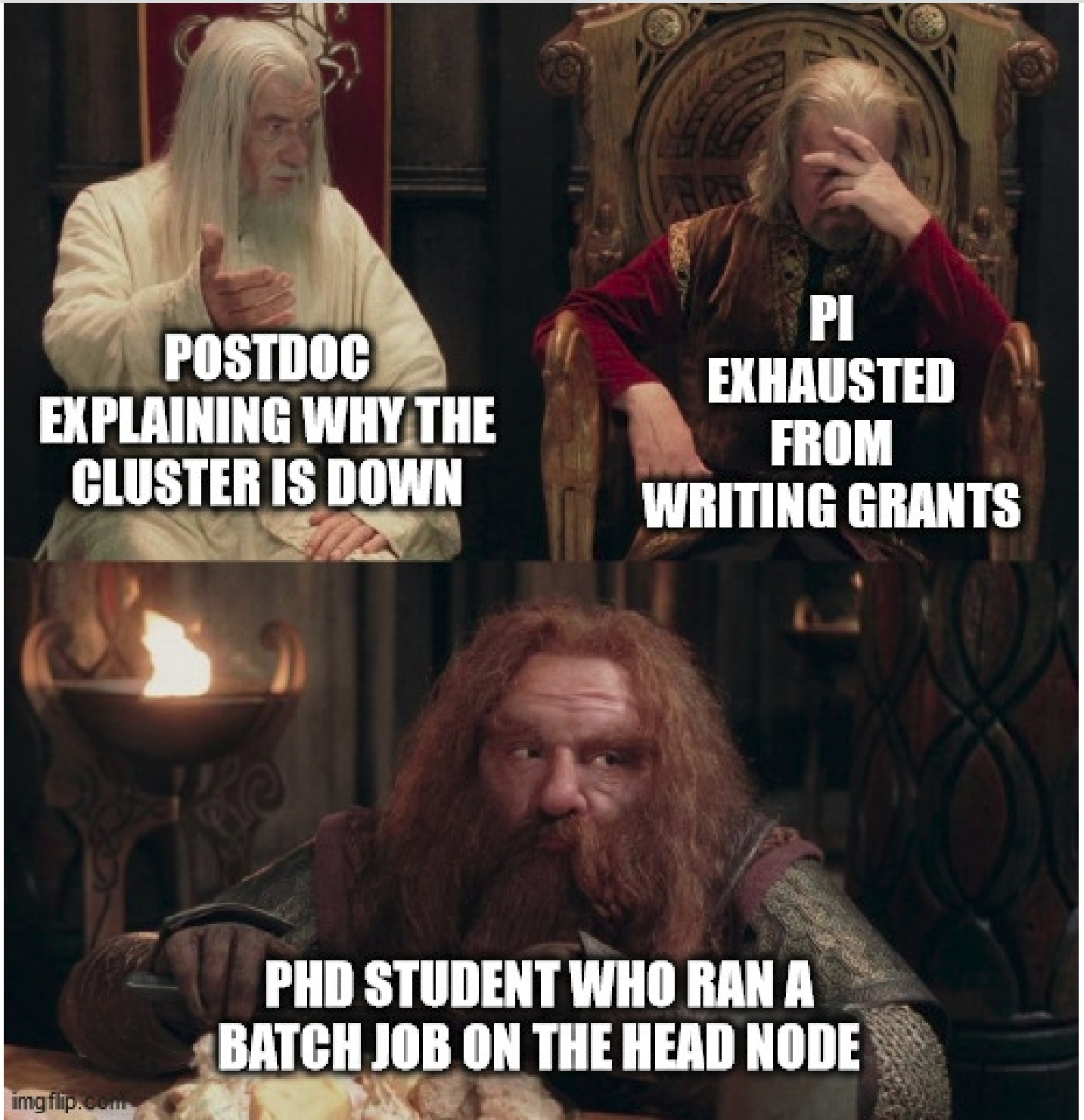


Overview | Features and Benefits | Get Early Access | Resources

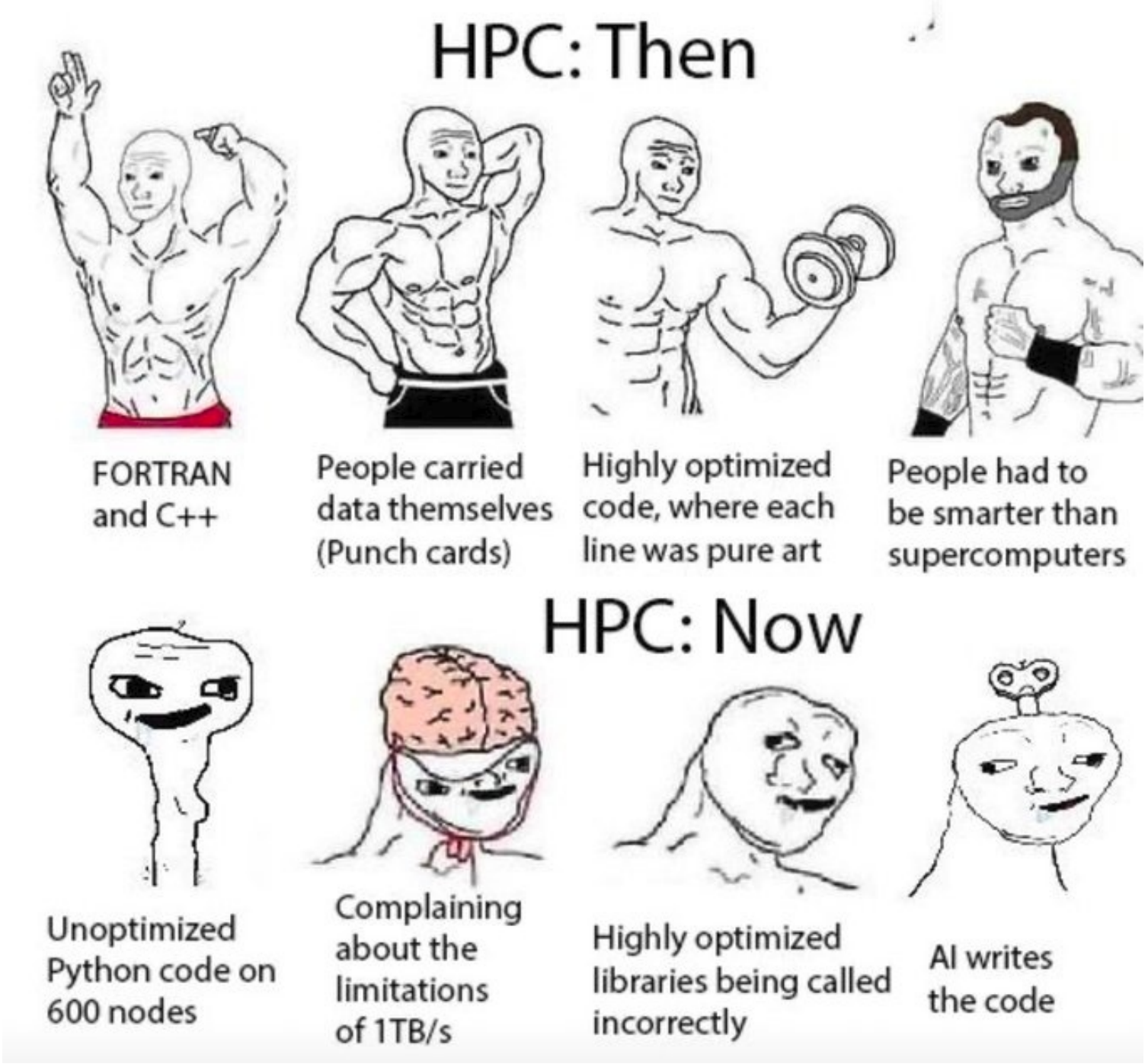
What Is BioNeMo?

NVIDIA BioNeMo™ is a cloud service for generative AI in drug discovery. With NVIDIA cloud APIs, researchers can quickly customize and deploy domain-specific, state-of-the-art generative and predictive biomolecular AI models at scale. BioNeMo enables researchers and developers to use generative AI models to rapidly generate the structure and function of proteins and biomolecules, accelerating the creation of new drug candidates.

HPC JOKES


POSTDOC
EXPLAINING WHY THE
CLUSTER IS DOWN
PI
EXHAUSTED
FROM
WRITING GRANTS
PHD STUDENT WHO RAN A
BATCH JOB ON THE HEAD NODE
imgflip.com

HPC JOKES


HPC: Then
FORTRAN and C++
People carried data themselves (Punch cards)
Highly optimized code, where each line was pure art
People had to be smarter than supercomputers
HPC: Now
Unoptimized Python code on 600 nodes
Complaining about the limitations of 1TB/s
Highly optimized libraries being called incorrectly
AI writes the code