VSC Tier-1 compute

Introducing **sofia**

Sam Moors (VUB-HPC)

Hortense: compute@vscentrum.be

sofia: support@vscentrum.be

VLAAMS SUPERCOMPUTER CENTRUM

Vlaanderen is supercomputing

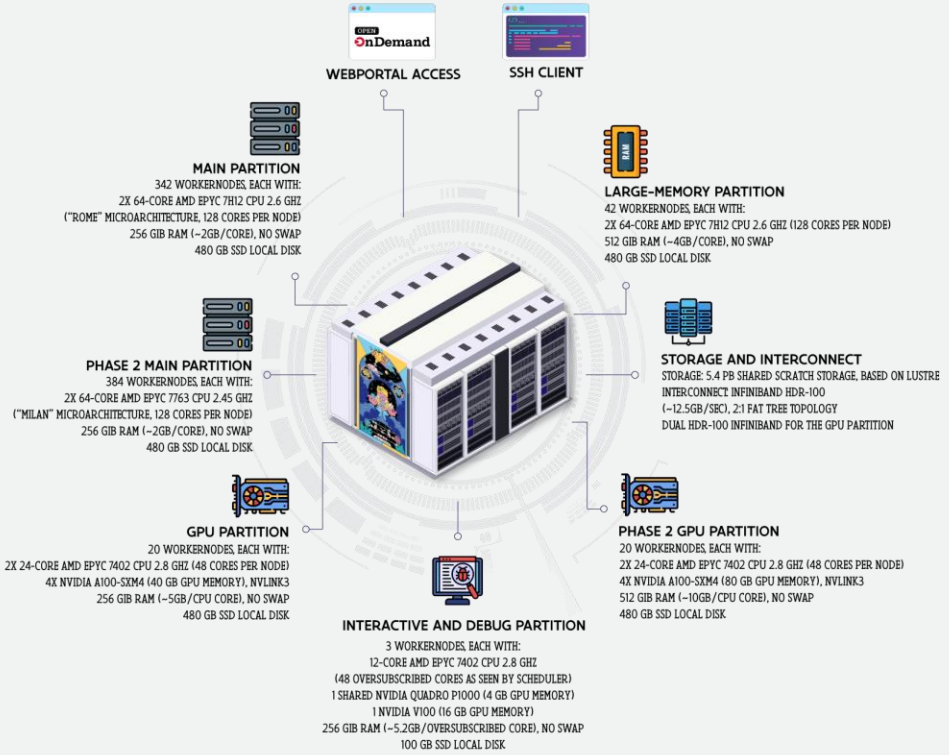

sofia
Powered by
Enlightened by
VUB
VRIJE
UNIVERSITEIT
BRUSSEL

VSC Tier-1 supercomputers

Hortense

- ~ **100,000** CPU cores
- **160** GPUs (NVIDIA A100, 40/80 GB)
- Infiniband interconnect: ~ 12.5 GB/sec
- 5 PB storage
- Operational until end of 2027
- ~50% will shut down ~July 2026
  - No more high-memory nodes
  - A100 40GB GPU nodes: uncertain
  - A100 80GB GPU nodes will stay

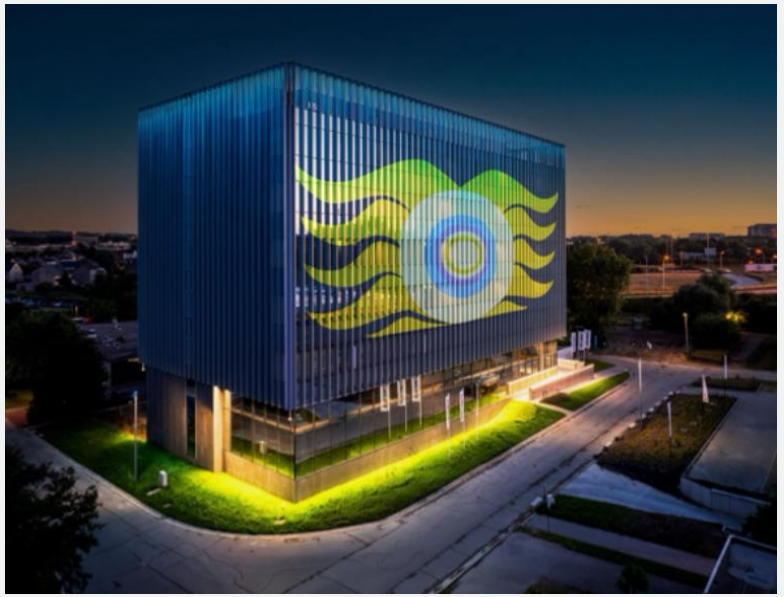
https://docs.vscentrum.be/gent/tier1_hortense.html



VSC Tier-1 supercomputers

sofia

- The next Tier-1 managed by VUB-HPC team
- Located in Nexus datacenter in Research Park Zellik
  - Max. energy efficiency
    - Heat recovery
    - Cooling with rainwater
- Currently in **pilot** phase
  - 22 April – 15 June 2026
- First Tier-1 projects on sofia: 1 June 2026
  - https://vscentrum.be/compute
- No starting grants yet, ask for a pilot project
  - support@vscentrum.be

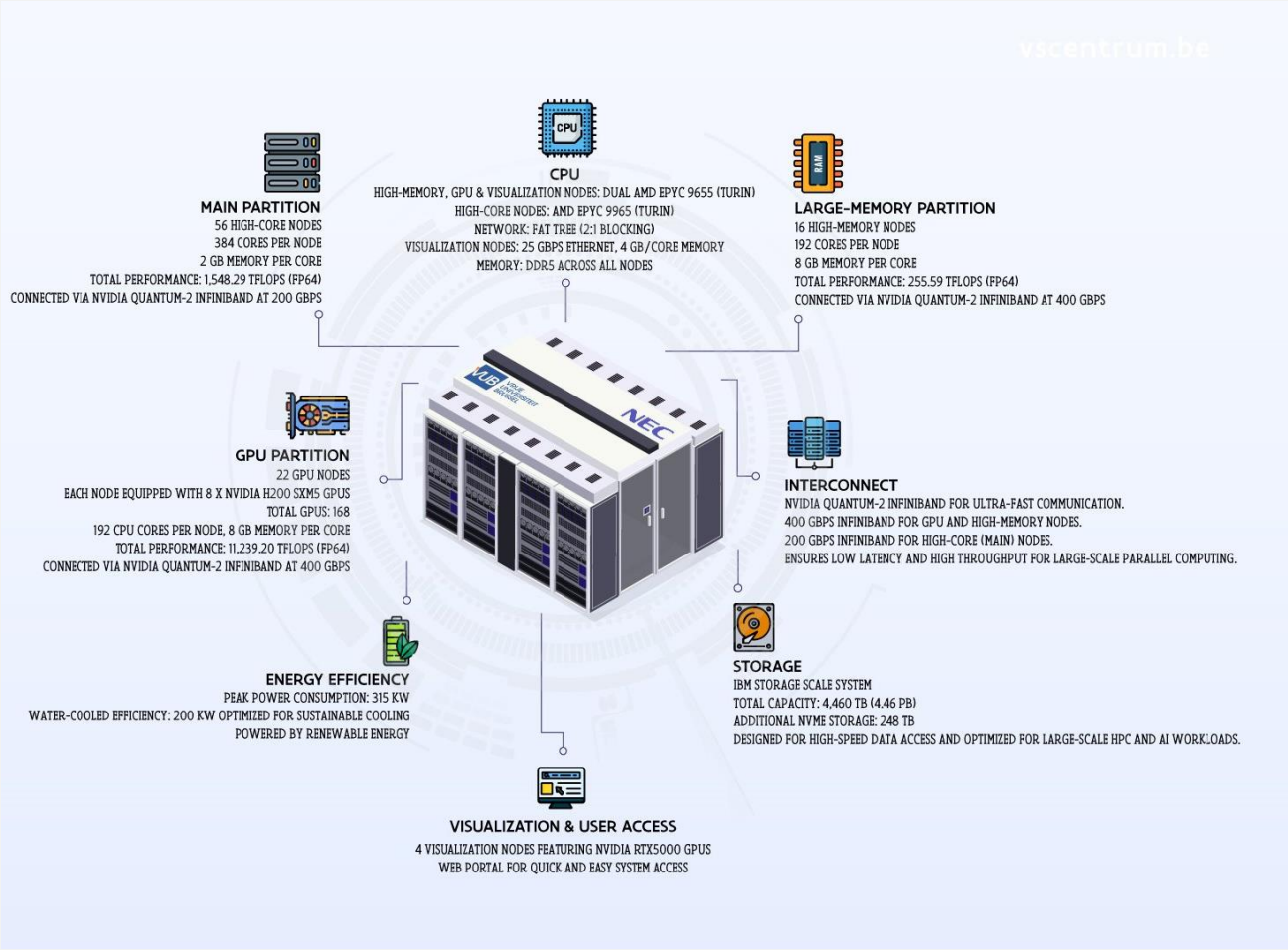


VSC Tier-1 supercomputers

sofia

- Initial investment of 8,6 million
  - ~20% expansion in 2027
- Focus on AI: **176** H200 GPUs
- **21,504** CPU cores with 2 GB of memory/core
- **3,072** CPU cores with 8 GB of memory/core
- Infiniband interconnect: 200/400 Gbps
- 4.3 PiB storage

https://docs.vscentrum.be/brussel/tier1_sofia.html

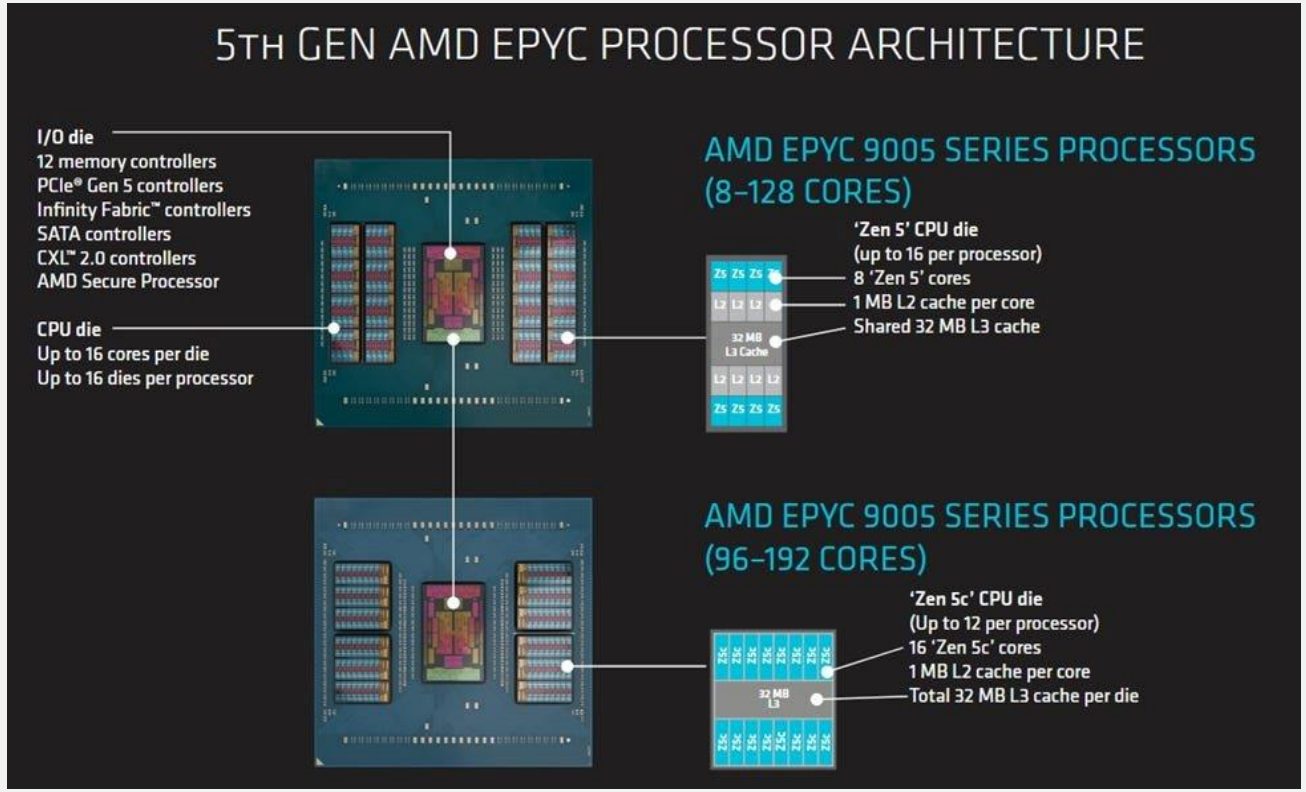


sofia hardware

CPUs

- Zen4: double-pumped 256-bit AVX-512
- Zen5: full 512-bit AVX-512

- **zen5_dense** (zen5c)
  - 2 * 192 cores
  - 2 * 12 CCDs
  - 2 * 4 NUMA nodes
  - 2 * 384 MB L3 cache
- **zen5_himem** (zen5c) and **zen4_h200** (zen4)
  - 2 * 96 cores
  - 2 * 12 CCDs
  - 2 * 4 NUMA nodes
  - 2 * 384 MB L3 cache

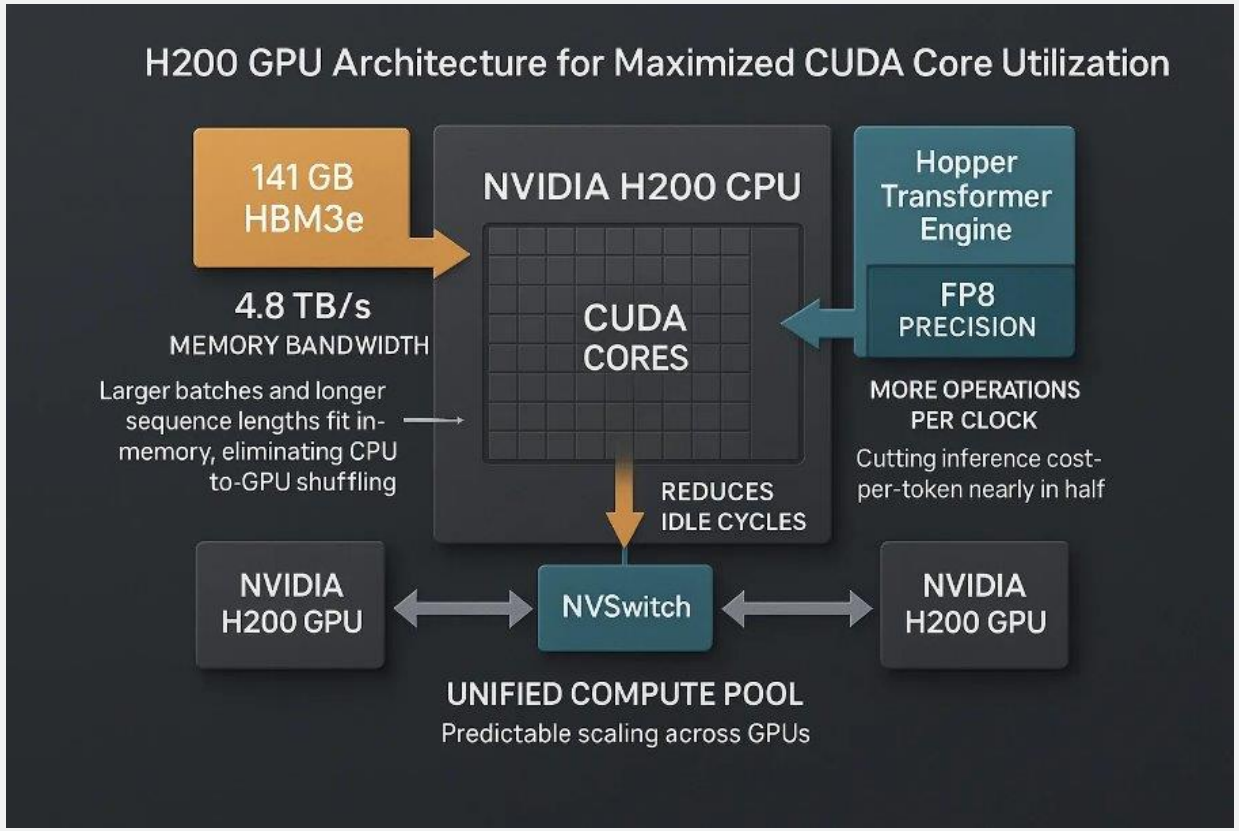


sofia hardware

GPUs

- **zen4_h200** (NVIDIA Hopper H200)
  - 8 GPUs/node
  - 141 GB VRAM
  - 4.8 TB/s memory bandwidth
  - NVLink 4 (~900 GB/s)

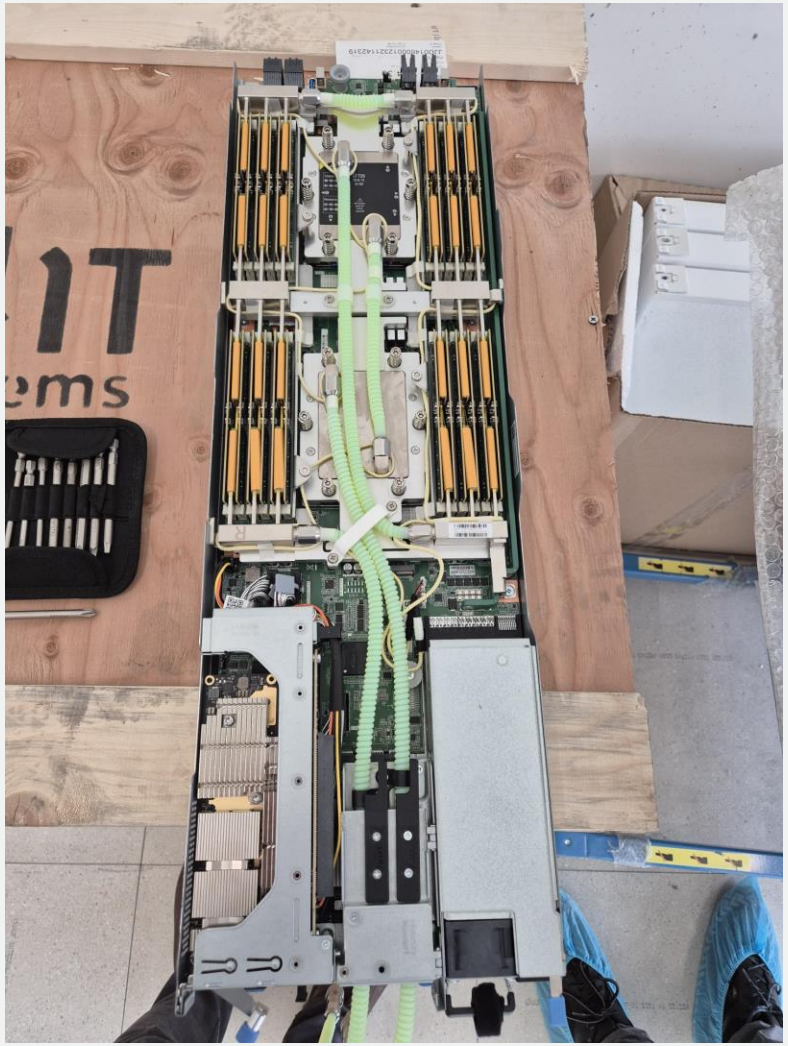
- **zen5_vis** (NVIDIA Ada RTX 5000)
  - 2 GPUs/node
  - 32 GB VRAM
  - Interactive usage (OnDemand)



sofia hardware

Liquid cooling

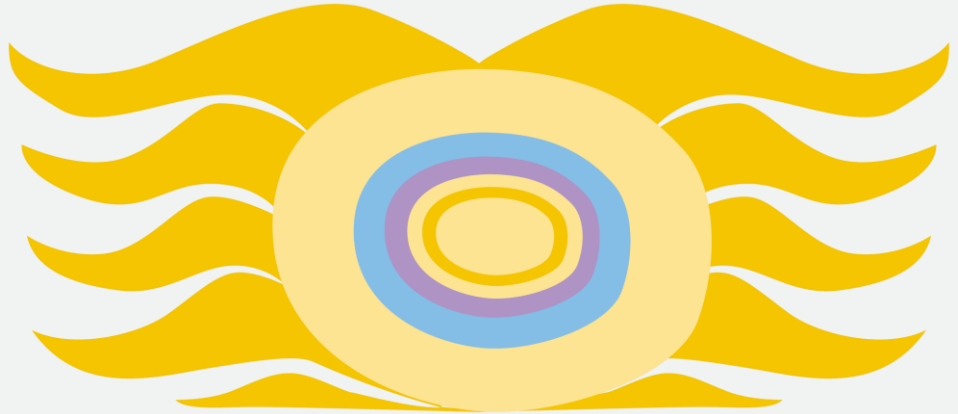
- Liquid cooling: CPUs, GPUs, CPU memory (RAM)
- Liquid-cooled:
  - **zen5_dense**
  - **zen5_himem**
  - **zen4_h200**
- Air-cooled
  - **zen5_vis**
  - Login nodes



sofia for users

General usage

- No *cluster* modules: specify a partition
  - **zen5_dense** (384 cores/node, ~2 GiB/core)
  - **zen5_himem** (192 cores/node, ~8 GiB/core)
  - **zen4_h200** (8 H2OO GPUs/node, 24 cores/GPU, ~8 GiB/core)
  - **zen5_vis** (2 RTX 5000 Ada GPUs/node interactive use)
- Fixed amount of CPU-memory per core
- Fixed #cores per GPU
- No Torque wrappers (*qsub*, *qstat*, *qdel*, …)
  - Use Slurm-native commands **sbatch**, **squeue**, **scancel**, …
  - Interactive job **salloc -p <partition>**
  - Job status **slurm_jobinfo**
- No MPI wrapper (*mympirun* wrapper in Hortense)
  - Use **mpirun** or **srun**
  - Manually control process/thread placement/binding!
- OnDemand portal (TBD)

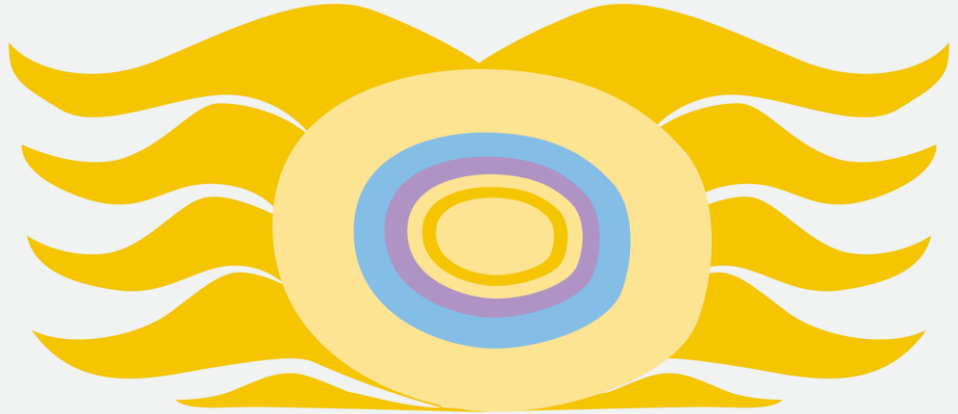


sofia for users

Storage

- Storage: IBM storage scale (GPFS)
  - Same as HPC-UGent Tier-2
- Home on scratch (50 GB)
  - Remains accessible when local Tier-2 is down
  - Don't use home for project data
- Data transfer
  - **$VSC_DATA** and **$VSC_HOME**

    (only on login nodes and during pilot phase)
  - Globus (TBD)
- Local scratch = private and temporary
  - **/tmp**, **$TMPDIR**, **$VSC_SCRATCH_NODE**
  - **/dev/shm** (in-memory storage)
- **sofia** will become inaccessible shortly after your project expires
  - Take regular backups
  - Retrieve data timely



sofia for users

Software

- Software modules
  - Optimized for the different CPU/GPU architectures
- EESSI software stack (TBD)
- Install on your own
  - See VSC training "Getting Scientific Software Installed" https://github.com/vscentrum/gssi-training
- Support for software installations and job troubleshooting
  - Local HPC support teams will be more involved

sofia performance

Weak scaling benchmarks

- Multiple tasks running concurrently on a single node
- More important for **sofia** with 384 cores/node

- Example: weak scaling job script

```
#!/bin/bash
#SBATCH --partition=zen5_dense  # 384 cores per node
#SBATCH --nodes=1
#SBATCH --ntasks=48  # 48 tasks
#SBATCH --cpus-per-task=8  # 8 cores per task

for x in $(seq 48); do
    srun --ntasks=1 --exact <my-prog> &
done
wait
```

sofia performance

Parallel scaling

- Process/thread placement and binding (pinning)
- MPI implementation: OpenMPI, intel-MPI, MPICH
- Compute limited vs. memory bandwidth limited
  - Memory-bw-limited software: OpenFOAM, ANSYS Fluent, …
  - Memory bw is more important with larger system size
  - Memory bw per core: **zen5_himem** > **zen5_dense**
  - Optimize #cores per process (e.g. 2,3,4 cores/process)
- Pure MPI vs. hybrid MPI-threading
- Consult documentation of your software

sofia performance

MPI jobs

- Example: pure MPI with compact process binding

```
#SBATCH --partition=zen5_dense
#SBATCH --nodes=1 --ntasks-per-node=384 --cpus-per-task=1

export OMP_NUM_THREADS=1
mpirun --map-by slot:PE=1 --report-bindings <mpi-prog>
srun -m block:block --cpu-bind=verbose,cores <mpi-prog>
```

- Example: hybrid MPI-OpenMP threading with compact process binding

```
#SBATCH --partition=zen5_dense
#SBATCH --nodes=1 --ntasks-per-node=192 --cpus-per-task=2

export OMP_NUM_THREADS=2
mpirun --map-by slot:PE=2 --report-bindings <mpi-prog>
srun -m block:block --cpu-bind=verbose,cores <mpi-prog>
```

sofia performance

MPI jobs

- Example: pure MPI with compact process binding, skipping every other core (memory bandwidth limited)

```
#SBATCH --partition=zen5_dense
#SBATCH --nodes=1 --ntasks-per-node=192 --cpus-per-task=2

export OMP_NUM_THREADS=1
mpirun --map-by slot:PE=2 --report-bindings <mpi-prog>
srun -m block:block --cpu-bind=verbose,cores <mpi-prog>
```

sofia performance

Multi-threading jobs

- Example: compact OpenMP thread binding

```
#SBATCH --nodes=1 --ntasks=1 --cpus-per-task=16

export OMP_NUM_THREADS=16

export OMP_PLACES=cores
export OMP_PROC_BIND=close
# Intel-compiler/LLVM equivalent:
export KMP_AFFINITY=compact

<my-prog>
```

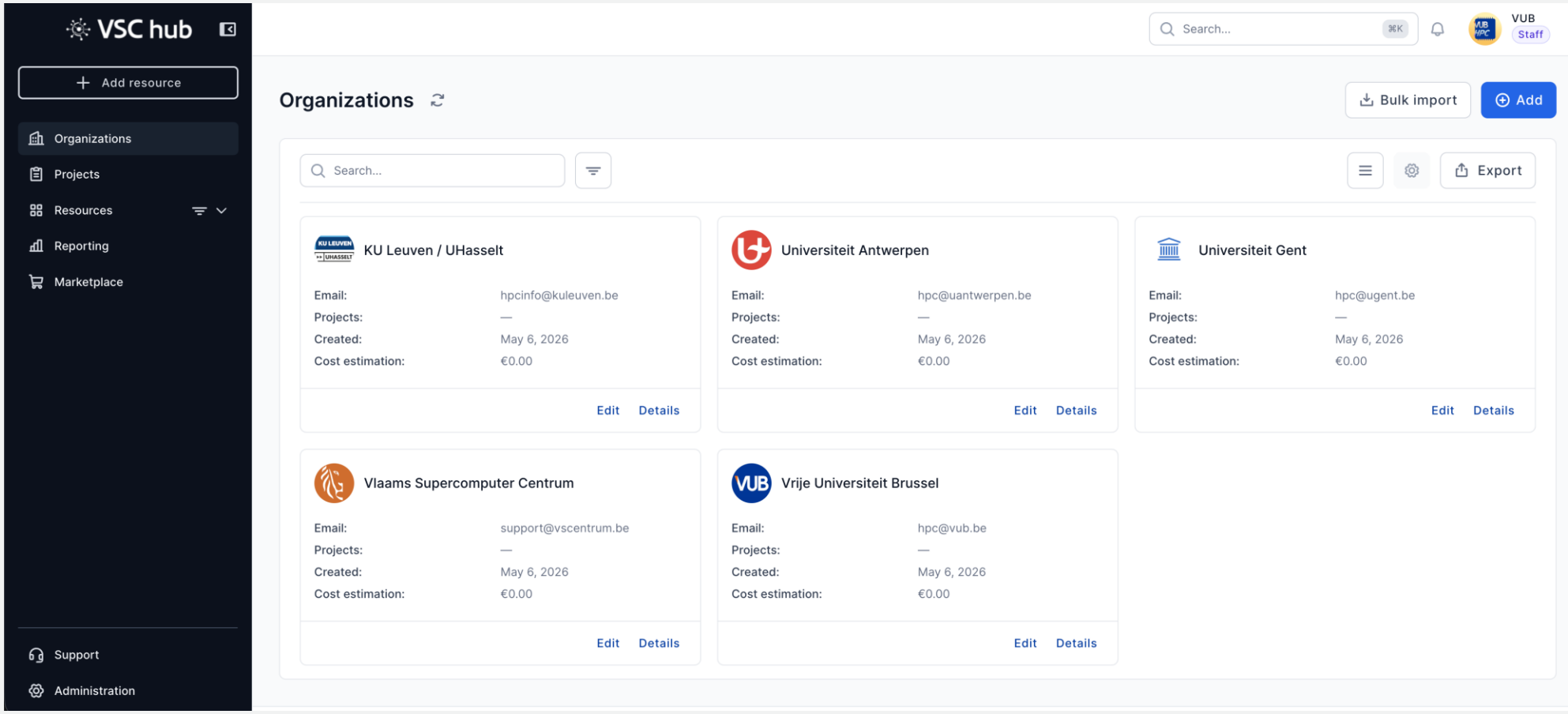
sofia performance

PyTorch multi-GPU jobs

- 1 process per GPU
- Optimize #CPU threads per process (e.g. 16 threads/core)
- Try to keep GPUs busy
  - Pre-load data in memory
  - Load data in parallel
  - Optimize batch size: use all GPU memory
- Use GPU monitoring
  - Weights & Biases (**wandb**)
    - "Free forever for academic research"
    - https://wandb.ai/site/pricing/

sofia project management

Waldur platform (TBD)



sofia project management

Waldur platform (TBD)

- https://hub.vscentrum.be
- Managing Projects, project members
- Resources: CPU/GPU credits, storage
- Reporting: resource usage over time
- Pricing:
  - Industry users
  - Research project accounting
- Also used by
  - EuroHPC Federation Platform (EFP)
  - LUMI (Tier-0)

Thank you!

sofia SUPERCOMPUTER

Powered by

Enlightened by

Supported by

In collaboration with