Olivia Introduction

Hardware,Storage,Data staging,Backup

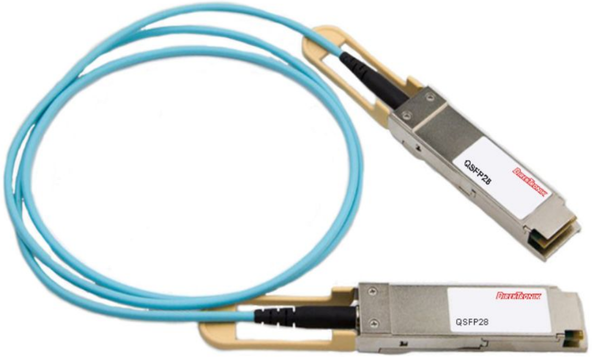
08.12.2025 Saerda

- Interconnect - Slingshot
- Olivia Hardware overview
- Olivia Filesystem layout
- Data staging and workflow
- Localscratch on CPU and GPU

Slingshot interconnect

switch

cables



Slingshot interconnect

Slingshot is a high-performance interconnect fabric developed by HPE/Cray for supercomputing, designed to handle high-bandwidth and low-latency needs for applications like AI and High-Performance Computing (HPC)

- Ethernet-based
- High bandwidth
- Low latency
- Adaptive routing
- Congestion control
- RDMA support

Slingshot network

Rosetta Switch 64-port, 200 Gbps

Cassini NIC 200Gbps

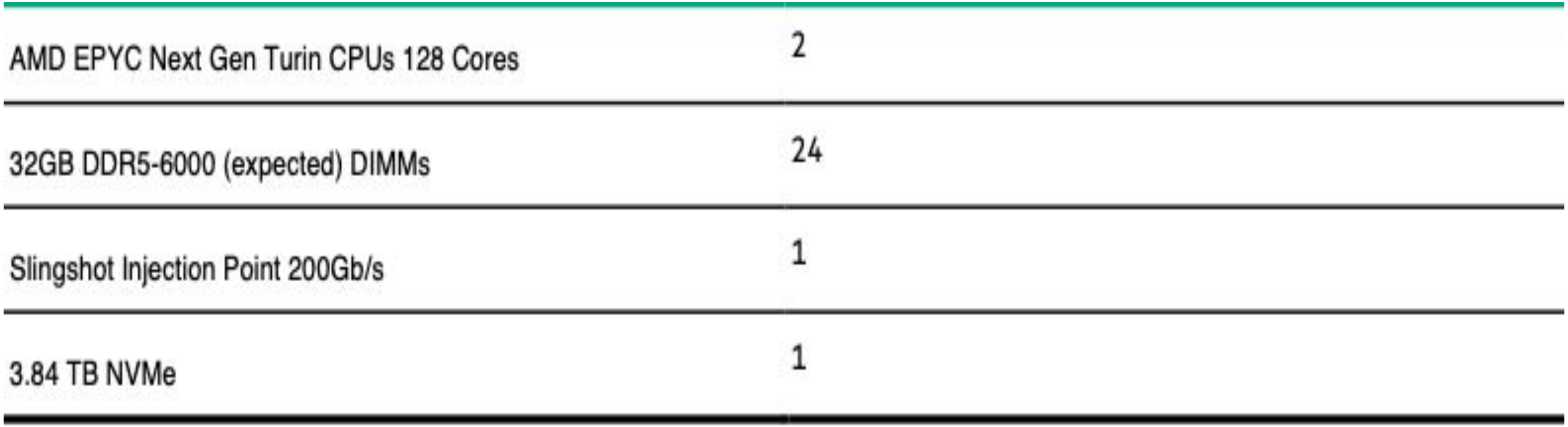
1 in cpu node, 4 in GPU node,1 service node , 2 IO node

Dragonfly topology

All compute connected to switch , and all switch are connected all switch.

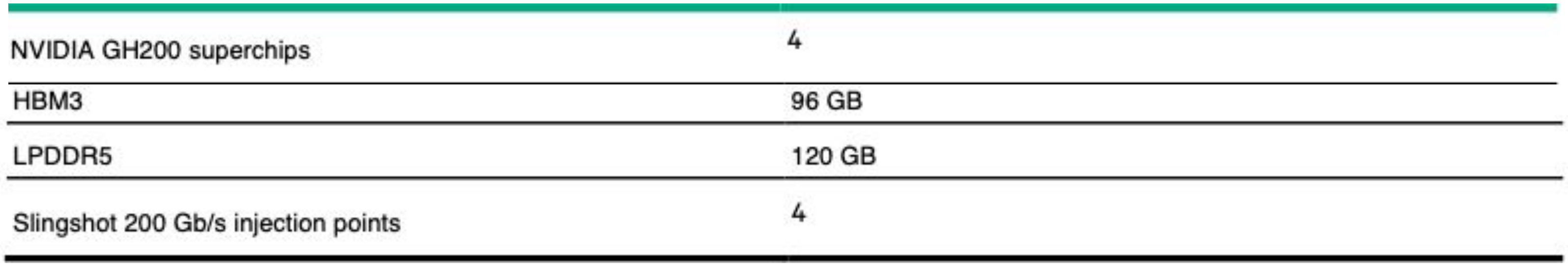
Olivia Hardware specification

Compute node 252 (4 node per balde)



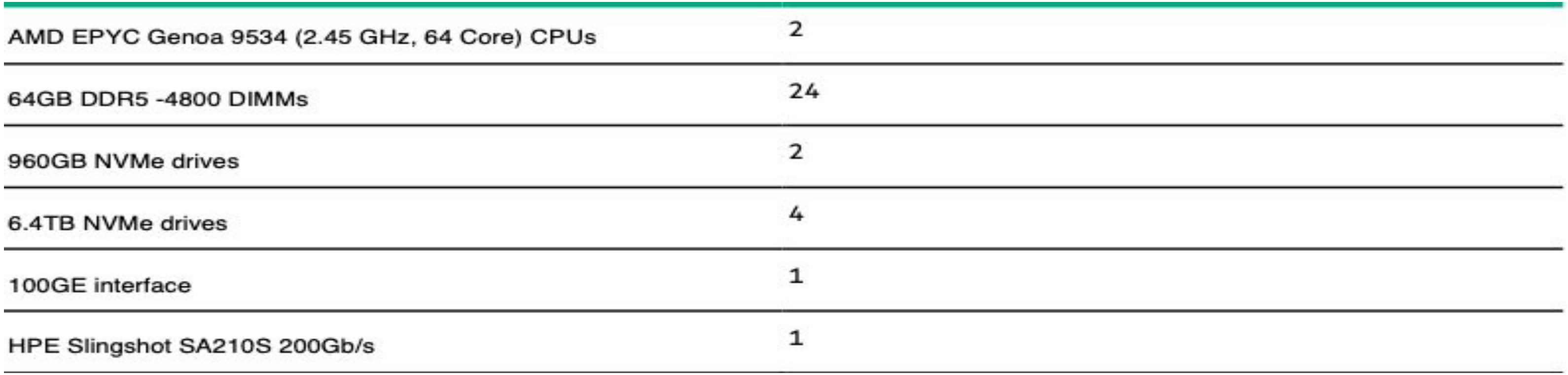
AMD EPYC Next Gen Turin CPUs 128 Cores	2
32GB DDR5-6000 (expected) DIMMs	24
Slingshot Injection Point 200Gb/s	1
3.84 TB NVMe	1

Accelerator node 76 (72 cores, ArmCPU)

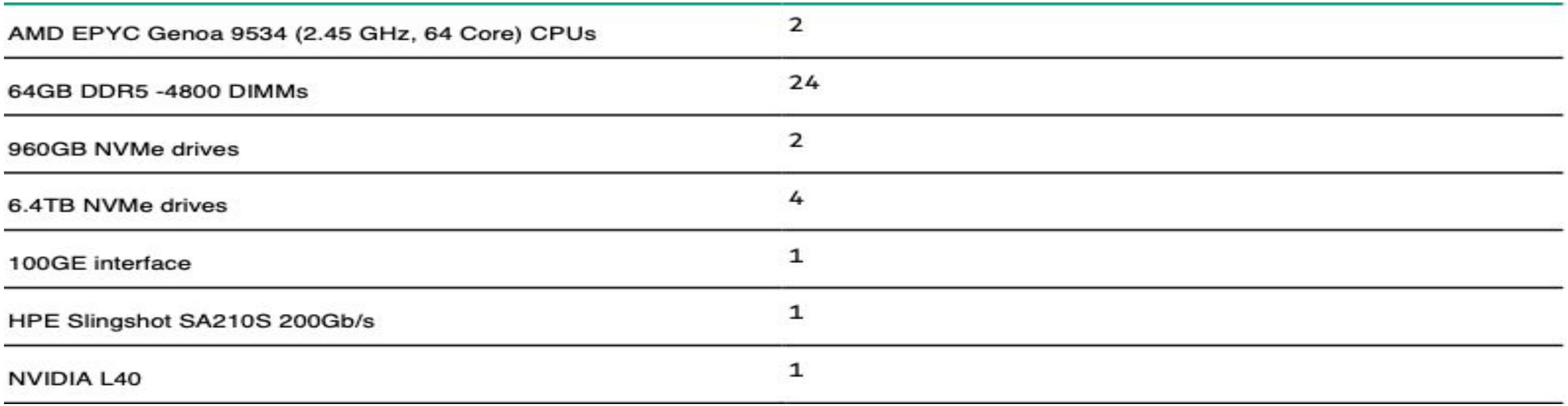


NVIDIA GH200 superchips	4
HBM3	96 GB
LPDDR5	120 GB
Slingshot 200 Gb/s injection points	4

Service nodes has two different specs

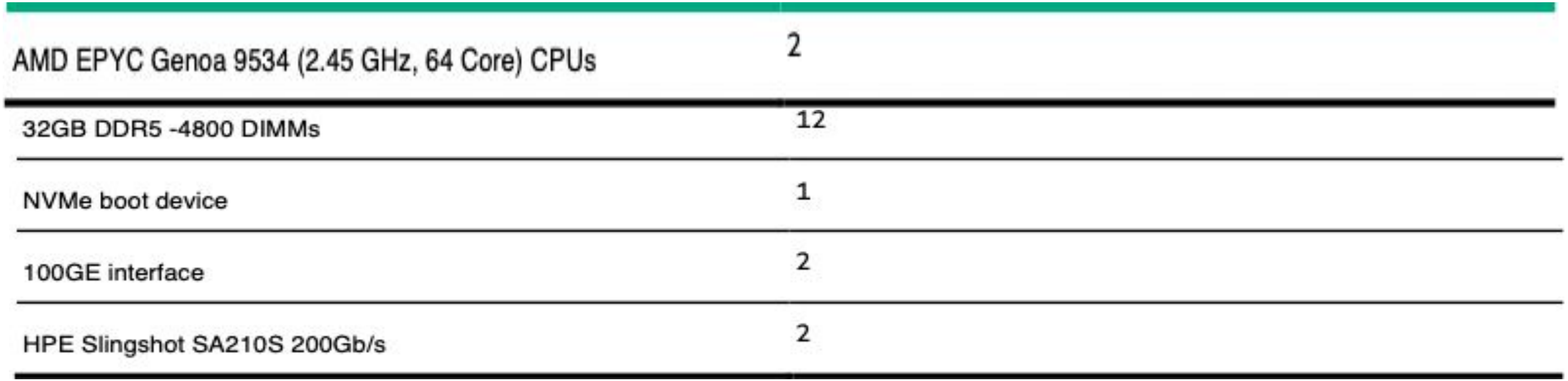


AMD EPYC Genoa 9534 (2.45 GHz, 64 Core) CPUs	2
64GB DDR5 -4800 DIMMs	24
960GB NVMe drives	2
6.4TB NVMe drives	4
100GE interface	1
HPE Slingshot SA210S 200Gb/s	1

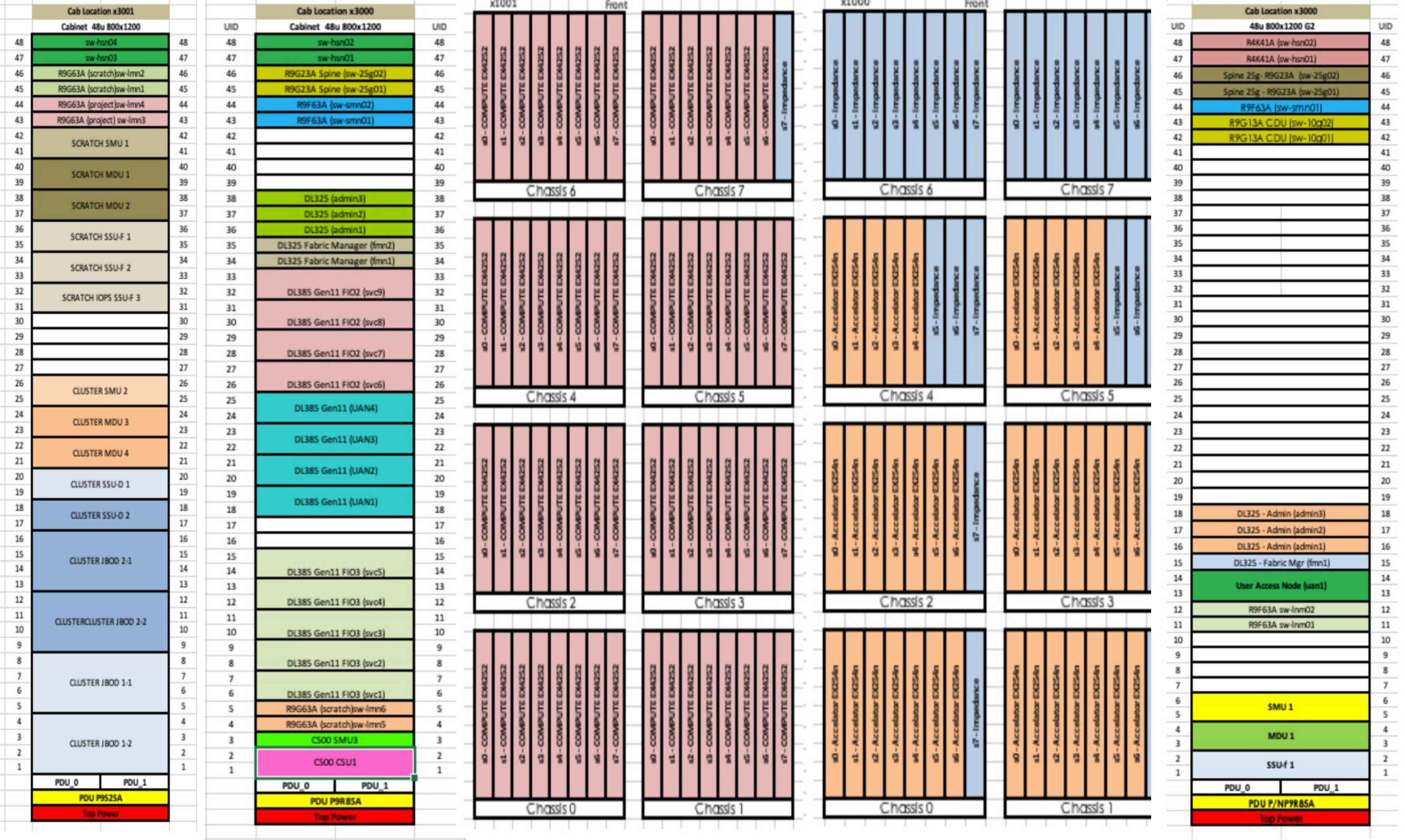


AMD EPYC Genoa 9534 (2.45 GHz, 64 Core) CPUs	2
64GB DDR5 -4800 DIMMs	24
960GB NVMe drives	2
6.4TB NVMe drives	4
100GE interface	1
HPE Slingshot SA210S 200Gb/s	1
NVIDIA L40	1

I/O nodes



AMD EPYC Genoa 9534 (2.45 GHz, 64 Core) CPUs	2
32GB DDR5 -4800 DIMMs	12
NVMe boot device	1
100GE interface	2
HPE Slingshot SA210S 200Gb/s	2



	Cab Location x3001		
	Cabinet 48u 800x1200		
48	sw-hsn04		48
47	sw-hsn03		47
46	R9G63A (scratch)sw-lmn2		46
45	R9G63A (scratch)sw-lmn1		45
44	R9G63A (project)sw-lmn4		44
43	R9G63A (project) sw-lmn3		43
42	SCRATCH SMU 1		42
41			41
40	SCRATCH MDU 1		40
39			39
38	SCRATCH MDU 2		38
37			37
36	SCRATCH SSU-F 1		36
35			35
34	SCRATCH SSU-F 2		34
33			33
32	SCRATCH IOPS SSU-F 3		32
31			31
30			30
29			29
28			28
27			27
26	CLUSTER SMU 2		26
25			25
24	CLUSTER MDU 3		24
23			23
22	CLUSTER MDU 4		22
21			21
20	CLUSTER SSU-D 1		20
19			19
18	CLUSTER SSU-D 2		18
17			17
16	CLUSTER JBOD 2-1		16
15			15
14			14
13			13
12	CLUSTERCLUSTER JBOD 2-2		12
11			11
10			10
9			9
8	CLUSTER JBOD 1-1		8
7			7
6			6
5			5
4	CLUSTER JBOD 1-2		4
3			3
2			2
1			1
	PDU_0	PDU_1	
	PDU P9S25A		
	Top Power		

	Cab Location x3000		
UID	Cabinet 48u 800x1200		UID
48	sw-hsn02		48
47	sw-hsn01		47
46	R9G23A Spine (sw-25g02)		46
45	R9G23A Spine (sw-25g01)		45
44	R9F63A (sw-smn02)		44
43	R9F63A (sw-smn01)		43
42			42
41			41
40			40
39			39
38	DL325 (admin3)		38
37	DL325 (admin2)		37
36	DL325 (admin1)		36
35	DL325 Fabric Manager (fmn2)		35
34	DL325 Fabric Manager (fmn1)		34
33	DL385 Gen11 FIO2 (svc9)		33
32			32
31	DL385 Gen11 FIO2 (svc8)		31
30			30
29	DL385 Gen11 FIO2 (svc7)		29
28			28
27	DL385 Gen11 FIO2 (svc6)		27
26			26
25	DL385 Gen11 (UAN4)		25
24			24
23	DL385 Gen11 (UAN3)		23
22			22
21	DL385 Gen11 (UAN2)		21
20			20
19	DL385 Gen11 (UAN1)		19
18			18
17			17
16			16
15	DL385 Gen11 FIO3 (svc5)		15
14			14
13	DL385 Gen11 FIO3 (svc4)		13
12			12
11	DL385 Gen11 FIO3 (svc3)		11
10			10
9	DL385 Gen11 FIO3 (svc2)		9
8			8
7	DL385 Gen11 FIO3 (svc1)		7
6			6
5	R9G63A (scratch)sw-lmn6		5
4	R9G63A (scratch)sw-lmn5		4
3	C500 SMU3		3
2	C500 CSU1		2
1			1
	PDU_0	PDU_1	
	PDU P9R85A		
	Top Power		

x1001 Front

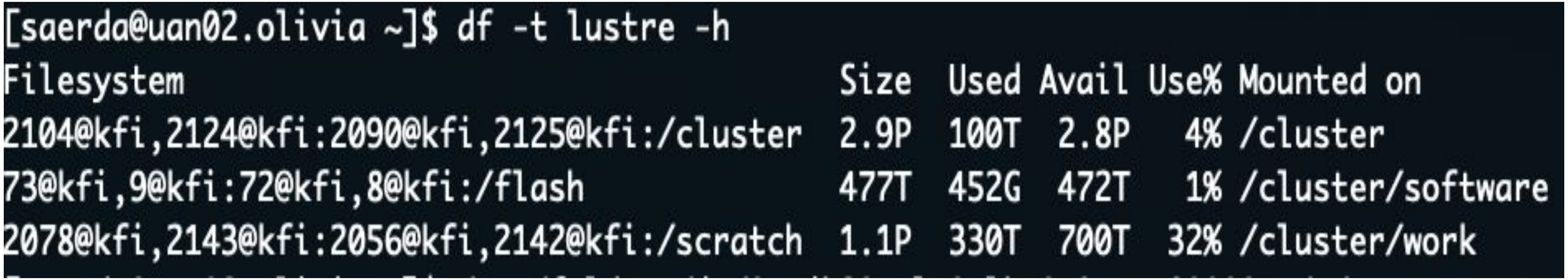
Chassis 6	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - COMPUTE EX4252
Chassis 7	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - Impedance
Chassis 4	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - COMPUTE EX4252
Chassis 5	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - COMPUTE EX4252
Chassis 2	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - COMPUTE EX4252
Chassis 3	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - COMPUTE EX4252
Chassis 0	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - COMPUTE EX4252
Chassis 1	s0 - COMPUTE EX4252	s1 - COMPUTE EX4252	s2 - COMPUTE EX4252	s3 - COMPUTE EX4252	s4 - COMPUTE EX4252	s5 - COMPUTE EX4252	s6 - COMPUTE EX4252	s7 - COMPUTE EX4252

x1000 Front

Chassis 6	s0 - Impedance	s1 - Impedance	s2 - Impedance	s3 - Impedance	s4 - Impedance	s5 - Impedance	s6 - Impedance	s7 - Impedance
Chassis 7	s0 - Impedance	s1 - Impedance	s2 - Impedance	s3 - Impedance	s4 - Impedance	s5 - Impedance	s6 - Impedance	
Chassis 4	s0 - Accelator EX254n	s1 - Accelator EX254n	s2 - Accelator EX254n	s3 - Accelator EX254n	s4 - Accelator EX254n	s5 - Impedance	s6 - Impedance	s7 - Impedance
Chassis 5	s0 - Accelator EX254n	s1 - Accelator EX254n	s2 - Accelator EX254n	s3 - Accelator EX254n	s4 - Accelator EX254n	s5 - Impedance	s6 - Impedance	
Chassis 2	s0 - Accelator EX254n	s1 - Accelator EX254n	s2 - Accelator EX254n	s3 - Accelator EX254n	s4 - Accelator EX254n	s5 - Accelator EX254n	s6 - Accelator EX254n	s7 - Impedance
Chassis 3	s0 - Accelator EX254n	s1 - Accelator EX254n	s2 - Accelator EX254n	s3 - Accelator EX254n	s4 - Accelator EX254n	s5 - Accelator EX254n	s6 - Accelator EX254n	
Chassis 0	s0 - Accelator EX254n	s1 - Accelator EX254n	s2 - Accelator EX254n	s3 - Accelator EX254n	s4 - Accelator EX254n	s5 - Accelator EX254n	s6 - Accelator EX254n	s7 - Impedance
Chassis 1	s0 - Accelator EX254n	s1 - Accelator EX254n	s2 - Accelator EX254n	s3 - Accelator EX254n	s4 - Accelator EX254n	s5 - Accelator EX254n	s6 - Accelator EX254n	

	Cab Location x3000		
UID	48u 800x1200 G2		UID
48	R4K41A (sw-hsn02)		48
47	R4K41A (sw-hsn01)		47
46	Spine 25g- R9G23A (sw-25g02)		46
45	Spine 25g - R9G23A (sw-25g01)		45
44	R9F63A (sw-smn01)		44
43	R9G13A CDU (sw-10g02)		43
42	R9G13A CDU (sw-10g01)		42
41			41
40			40
39			39
38			38
37			37
36			36
35			35
34			34
33			33
32			32
31			31
30			30
29			29
28			28
27			27
26			26
25			25
24			24
23			23
22			22
21			21
20			20
19			19
18	DL325 - Admin (admin3)		18
17	DL325 - Admin (admin2)		17
16	DL325 - Admin (admin1)		16
15	DL325 - Fabric Mgr (fmn1)		15
14	User Access Node (uan1)		14
13			13
12	R9F63A sw-lnm02		12
11	R9F63A sw-lnm01		11
10			10
9			9
8			8
7			7
6	SMU 1		6
5			5
4	MDU 1		4
3			3
2	SSU-f 1		2
1			1
	PDU_0	PDU_1	
	PDU P/NP9R85A		
	Top Power		

File System on Olivia



```
[saerda@uan02.olivia ~]$ df -t lustre -h
Filesystem                                       Size  Used Avail Use% Mounted on
2104@kfi,2124@kfi:2090@kfi,2125@kfi:/cluster  2.9P  100T  2.8P   4% /cluster
73@kfi,9@kfi:72@kfi,8@kfi:/flash              477T  452G  472T   1% /cluster/software
2078@kfi,2143@kfi:2056@kfi,2142@kfi:/scratch  1.1P  330T  700T  32% /cluster/work
```

File System on Olivia

Cluster file system

- mounted as /cluster
- 424x10TB SAS HDD disk
- Has separate MDS and OSS server
- Serving mainly Home, projects
- Home will be in backup.

Flash file system

mounted as /cluster/software

24x 30 TB NVMe

Has separate MDS and OSS server

Serving mainly software , gpujobscrach

Scratch filesystem

mounted as /cluster/work

48x30TB NVMe

24x7.68 NVMe

Has separate MDS and OSS server

Serving mainly Work projects(userwork) , support

Subjected to clean up at 70%,80%,90% usage

Olivia: File staging

On olivia not all projects have dedicated project storage,most of the projects will have to keep data on NIRD, data has to be moved in between NIRD and Olivia, to achieve this we have two alternatives

- Manual staging from NIRD to Olivia

- Staging via slurm as batch job

Manual staging

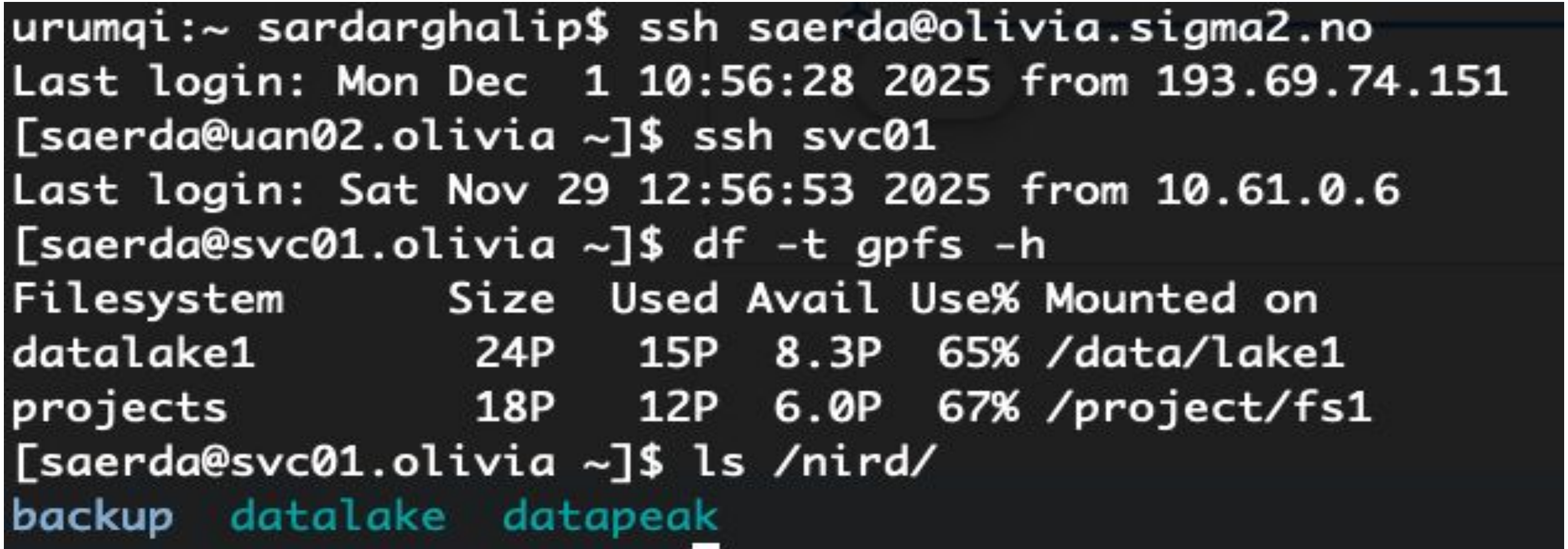
- We need to use IO nodes, svc0[1-5]
- IO nodes can be accessed from login nodes
- IO nodes are physically connected to NIRD network and NIRD filesystem is mounted
- Each IO nodes has 200Gib/s connection to NIRD

Manual staging

NIRD filesystems are presented as following on svc nodes :

/nird/datalake

/nird/datapeak



```
urumqi:~ sardarghalip$ ssh saerda@olivia.sigma2.no
Last login: Mon Dec  1 10:56:28 2025 from 193.69.74.151
[saerda@uan02.olivia ~]$ ssh svc01
Last login: Sat Nov 29 12:56:53 2025 from 10.61.0.6
[saerda@svc01.olivia ~]$ df -t gpfs -h
Filesystem      Size  Used Avail Use% Mounted on
datalake1        24P   15P  8.3P  65% /data/lake1
projects         18P   12P  6.0P  67% /project/fs1
[saerda@svc01.olivia ~]$ ls /nird/
backup  datalake  datapeak
```

Manual staging

cp : basic , simple but not efficient.

rsync: recommended, works best with synchronizing folders files, suits well with network transfer

Staging in

```
rsync -avh --progress /nird/datapeak/NSxxxxK/input_data/ /cluster/work/projects/nnxxxxk/job_input/
rsync -avh --progress /nird/datalake/NSxxxxK/input_data/ /cluster/work/projects/nnxxxxk/job_input/
```

Staging out

```
rsync -avh --progress /cluster/work/projects/nnxxxxk/job_output/ /nird/datapeak/NSxxxxK/results/
rsync -avh --progress /cluster/work/projects/nnxxxxk/job_output/ /nird/datalake/NSxxxxK/results/
```

Usa tmux instead of nohup

We recommend to use tmux session.

```
tmux new -s $session

Tmux ls

tmux at -t $session
```

For detaching (ctrl+b)+d

Pros:

- Simple and transparent workflow
- Full user control over data management

Cons:

- Requires manual effort
- High risk of human error
- No automation
- Inefficient for large datasets or batch jobs

Staging files via Slurm

We have configured slurm such that users can integrate the whole data moving process in to slurm script.

Add following lines for staging In:

```
#STAGE IN /nird/datalake/NSxxxxK/some/path /cluster/work/project/nnxxxxk/another/path
```

Staging files via Slurm

Each line should contain one from-path and one to-path.

One can specify a single file, or a directory, which will be copied recursively.

When the job has been submitted, the queue system will copy data,and will wait until the files have been copied before starting the job.

While the file copying is running, the job will be pending with reason **`BurstBufferStageIn.`**

If the copying fails (for instance because the file or directory doesn’t exist), the job will be left pending.

The job must then be cancelled, and resubmitted after fixing the problem.

Staging files via Slurm

Staging out

Add following lines for staging out:

```
#STAGE OUT /cluster/work/project/nnxxxxk/some/path /nird/datalake/NSxxxxK/another/path
```

When a job has finished, the files will be copied to NIRD. While the files are being copied, the job will be in state `completing`

If the copying fails (for instance because the file or directory doesn’t exist), the job will be left in state `STAGE_OUT(SO),` user has to requeue the job before it can be deleted.

Pros:

- Fully automated
- Avoid human error

Cons:

- Can make slurm script little complicated
- Everything is controlled by slurm

Run jobs without staging data

On this approach, jobs are executed directly on the compute nodes, accessing input data stored on NIRD without copying it to the local work directory.

- Submit directly from login node
- Datapeak and Datalake is mounted read-only on compute.
- Output has to be copied back after the job
- Recommended for small or moderate data volumes and quick analyses where data staging is unnecessary
- Simple workflow (no staging)

IMPORTANT: It is not recommended for high-throughput or I/O-intensive workflows where parallel data access could impact performance.

Best Practices for Data Managemen

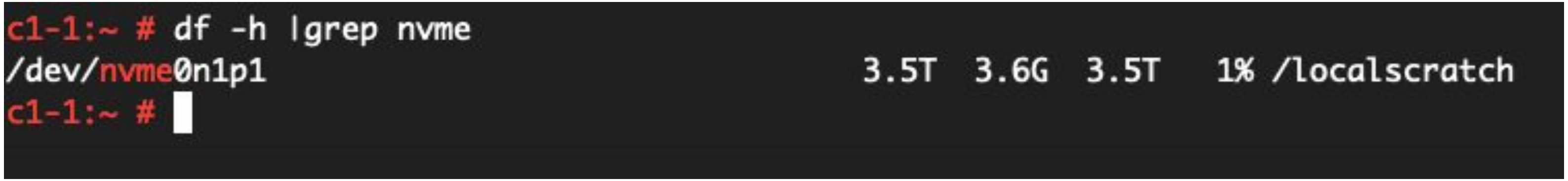
- Use the I/O nodes exclusively for data transfer and preparation tasks.
- Avoid running computationally intensive jobs on these nodes.
- Regularly transfer output data to NIRD Data Lake or NIRD Data Peak for mid-term storage.
- Remove unnecessary temporary data from Olivia (work) to optimize storage usage.

Localscrach on CPU nodes and GPU nodes

CPU nodes has localscrach

Each CPU compute nodes has a local nvme disk which is used by slurm as $scrach, this is only accessible by the user running the job.

`/localscratch/$SLURM_JOB_ID` ($SCRATCH)

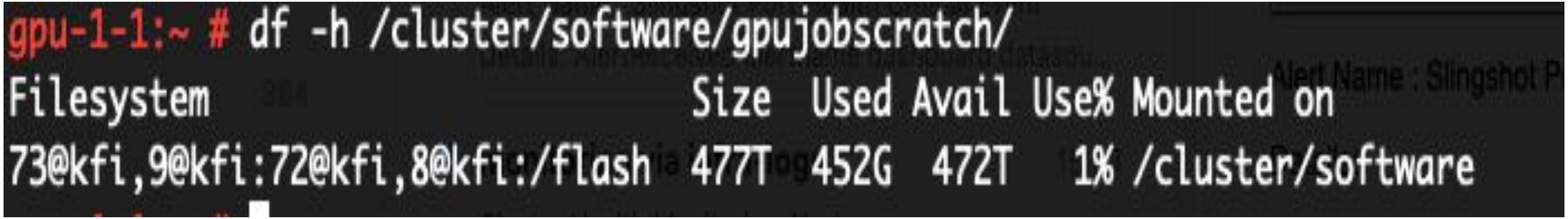


```
c1-1:~ # df -h |grep nvme
/dev/nvme0n1p1                              3.5T  3.6G  3.5T   1% /localscratch
c1-1:~ # 
```

GPU nodes does not have localscrach

GPU nodes does not have local disk, it uses shared file system for slurm $SCRACH

`/cluster/software/gpujobscratch/jobs/$SLURM_JOB_ID` (`$SCRATCH`)



```
gpu-1-1:~ # df -h /cluster/software/gpujobscratch/
Filesystem                              Size  Used Avail Use% Mounted on
73@kfi,9@kfi:72@kfi,8@kfi:/flash  477T  452G  472T   1% /cluster/software
```

Localscrach

All jobs in the *normal* partition (CPU nodes) on **Olivia** get their scratch area on local disk. One does not have to specify a size for this, and all jobs on the node share the available storage. The size of the localscratch disk on Olivia compute nodes is 3.5 TiB.

- less risk of interference from other jobs
- the scripts do not need to clean up temporary files (auto delete)
- It can be hard to debug jobs that fail (auto delete)
- special commands to copy files back in case the job script crashes before it has finished
- If the main node of a job crashes, everything is lost.

Fram home and projects

```
[saerda@svc01.olivia ~]$ ls /nird/backup/fram/home

[saerda@svc01.olivia ~]$ ls /nird/backup/hpc/fram/projects
```

Nird Backup

```
[saerda@svc01.olivia /nird/datapeak/NS9997K/.snapshots]$ ls /nird/datapeak/NS9997K/.snapshots/


[saerda@svc01.olivia /nird/datapeak/NS9997K/.snapshots]$ ls /nird/datalake/NS9997K/.snapshots/
```

Thank you !

Questions?