Efficient data handling and data formats

HPC Cafe, 2024-02-06

HPC Services, RRZE / NHR@FAU, hpc-support@fau.de

NHR@FAU file systems overview

Mount point	Access	Purpose	Technology	Backup	Snap-shots	Data lifetime	Quota
`/home/hpc`	`$HOME`	Source, input, important results	NFS	YES	YES @30 min	Account lifetime	50 GB
`/home/vault`	`$HPCVAULT`	Mid-/long-term storage	NFS	YES	YES @1/day	Account lifetime	500 GB

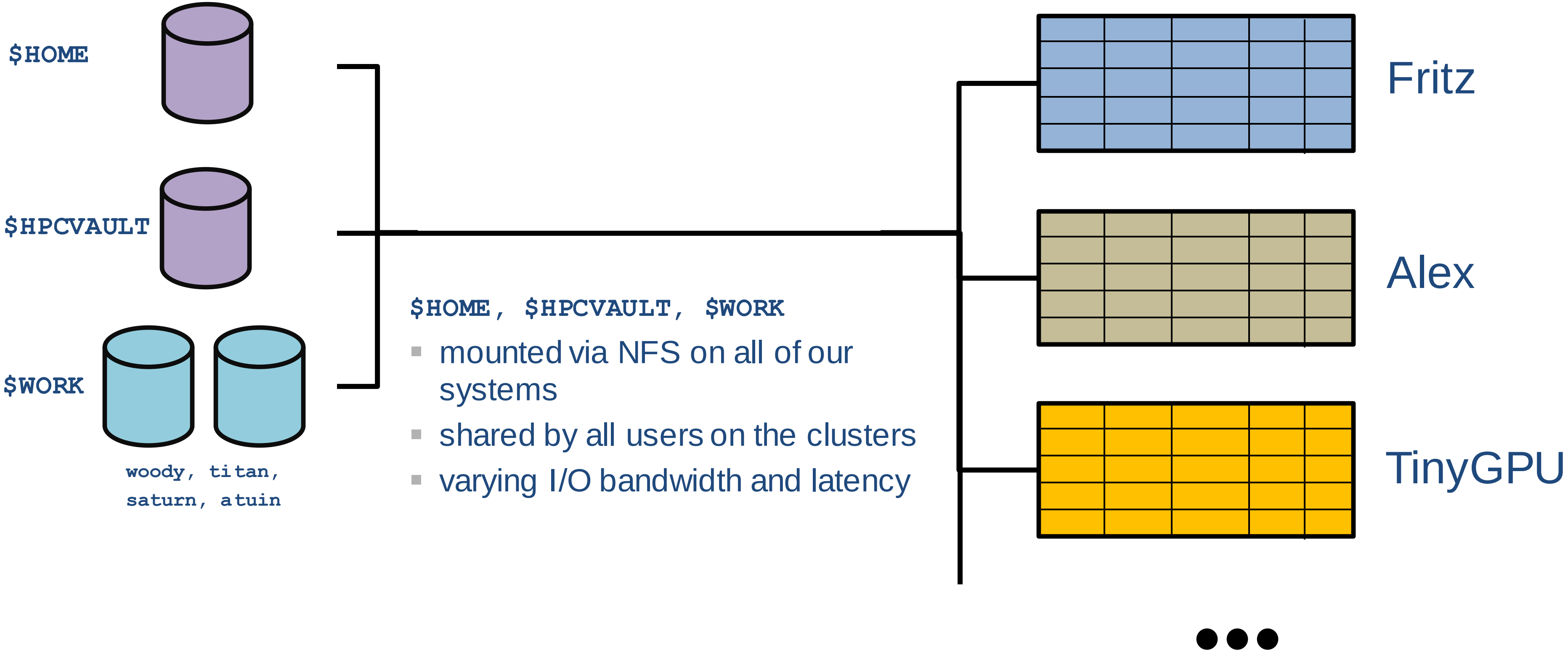
`/home/woody`
`/home/saturn`
`/home/titan`
	`$WORK`	Short-/mid-term storage, General-purpose	NFS	NO	NO	Account lifetime	500 GB

`/lxfs`	`$FASTTMP` (Fritz)	High performance parallel I/O	Lustre parallel FS via InfiniBand	NO	NO	High watermark	Only inodes
`/???`	`$TMPDIR`	Node-local, job-specific directory	SSD/ RAM disk	NO	NO	Job runtime	NO

`$TMPDIR:`

- SSDs vary in size across clusters, but generally > 1TB
- capacity of SSDs is shared with all other jobs on the same node

Storage at NRH@FAU



$HOME

$HPCVAULT

$WORK

woody, titan, saturn, atuin

$HOME, $HPCVAULT, $WORK

- mounted via NFS on all of our systems
- shared by all users on the clusters
- varying I/O bandwidth and latency

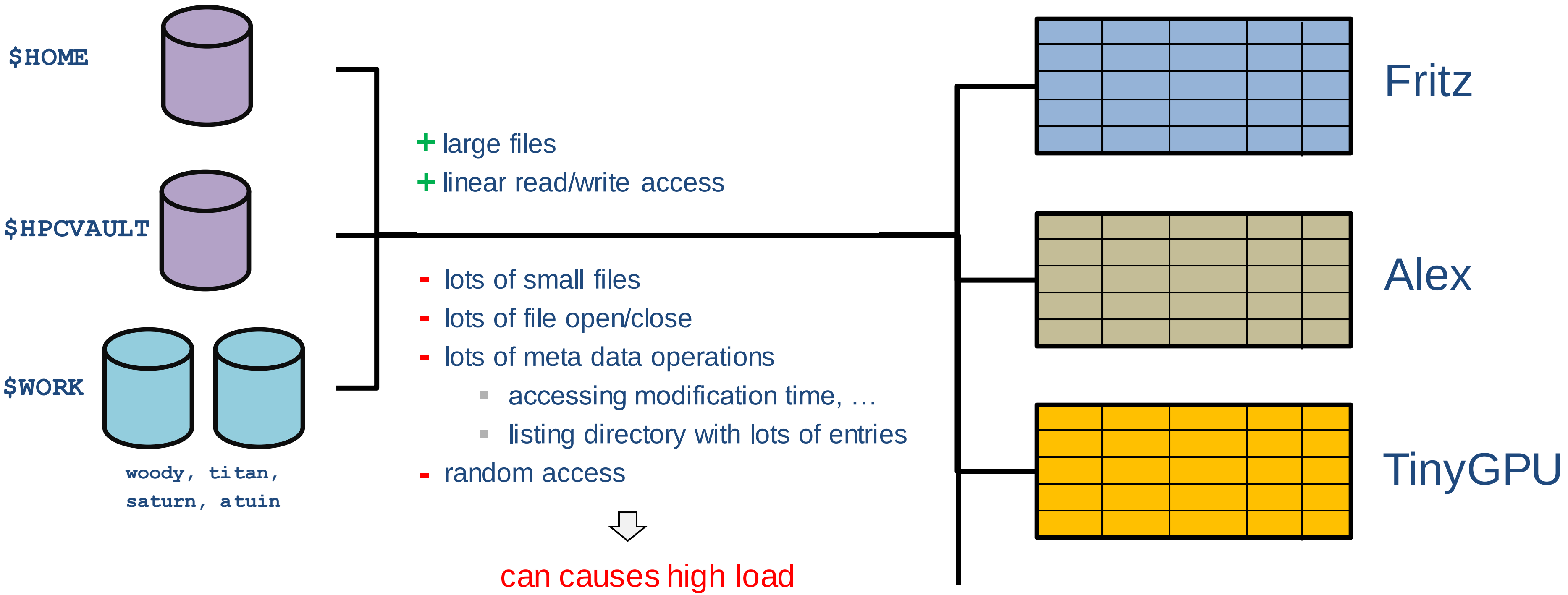
Fritz

Alex

TinyGPU

●●●

Storage at NRH@FAU



+ large files
+ linear read/write access

- lots of small files
- lots of file open/close
- lots of meta data operations
  - accessing modification time, …
  - listing directory with lots of entries
- random access

⇩

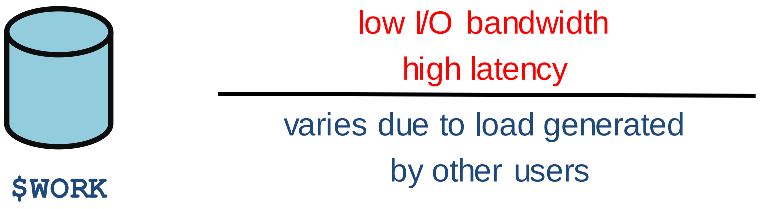
can causes high load
on NFS servers

●●●

store data in archive/container format on **`$WORK`**

unpack/copy to node local storage **`$TMPDIR`**

Staging data in an out during a job

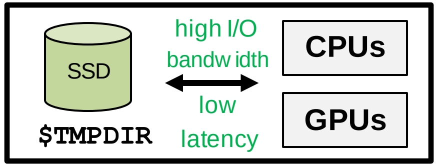


$WORK

low I/O bandwidth

high latency

varies due to load generated by other users



SSD

$TMPDIR

high I/O bandwidth

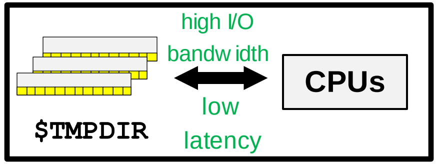
low latency

CPUs

GPUs

Alex, TinyGPU

w/o GPUs: TinyFat, Woody



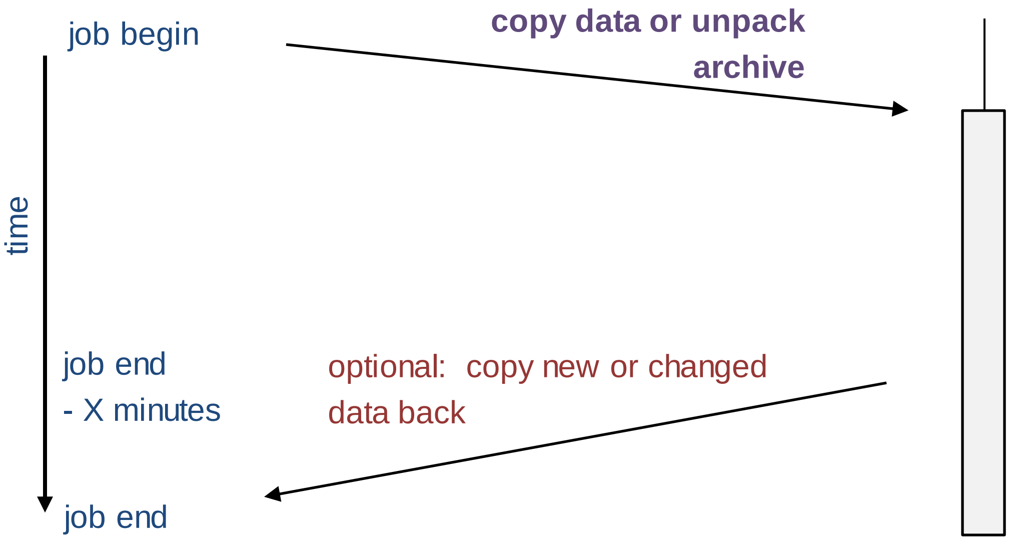
$TMPDIR

high I/O bandwidth

low latency

CPUs

Fritz, Meggie



job begin

copy data or unpack archive

time

job end - X minutes

optional: copy new or changed data back

job end

```
#!/bin/bash -l
#SBATCH --time=<TIME>
# ...

tar xf "$WORK/data.tar" -C "$TMPDIR"

python3 train.py --workdir "$TMPDIR" …

# OPTIONAL
cp -r "$TMPDIR/results" "$WORK"
```

at job end **$TMPDIR** gets automatically deleted

Archives…

- Typically: tar, zip, …
- If you only want to unpack selected files from an archive:
    - zip or any other format that has an index
- Compression:
    - depends on
        - your data
        - performance of decompression
    - benchmark yourself

Example use cases

Many files, frequent accesses

- Data set with many separate files on **`$WORK`**
- Many accesses per second to the data set

Remedy:

- Store as an archive/container format on **`$WORK`**
- Usage options:
  - Unpack archive to **`$TMPDIR`** and use data from there or
  - Load into RAM (if size permits it)

Share data among jobs on the same node

- Copying archive/dataset to **`$TMPDIR`** takes very long

Remedy:

- Share data with your concurrently running jobs on the same node
- Details: https://doc.nhr.fau.de/data/staging/#share-staged-data-with-concurrently-running-jobs-on-the-same-node

Frequent checkpoints

- High frequent checkpointing to `$WORK`

Remedy:

- Reduce frequency
- Use the lowest frequency that makes sense for your case

Frequent log file writing

- Continuously writing to logfile on `$WORK`

Remedy:

- Write logfile to `$TMPDIR`
- Before job ends copy logfile from `$TMPDIR` to `$WORK`

Questions? Suggestions?

Contact hpc-support@fau.de