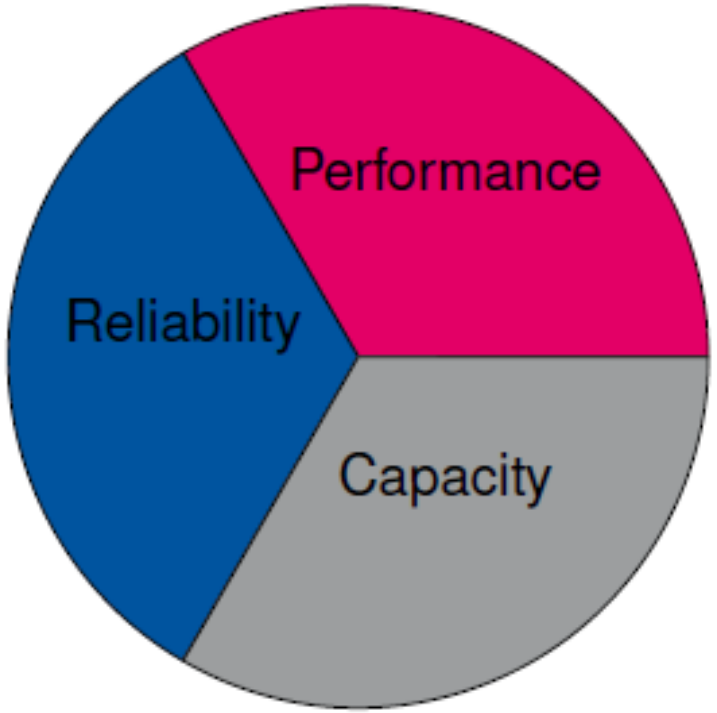
aiXcelerate 2024

Storage and Data Handling

NHR4CES NHR for Computational Engineering Science | HPC.NRW | EDIH Rheinland | WEST AI KI-Servicezentrum | IT Center | RWTH AACHEN UNIVERSITY

CLAIX – I/O Storage Strategy

- **Why do we need different filesystems?**
  - **Performance**
    - Bandwidth [GB/s]: How quickly can I move raw bytes?
    - Metadata [IOPS]: How quickly can I perform file operations?
    - Better performance means better and more expensive hardware
  - **Reliability**
    - Uptime: How often is the system unreachable?
    - Snapshots: Protection against accidental deletion
    - Backups: Protection against system failures
    - Better reliability means redundancies
  - **Capacity**
    - Total size in bytes
    - Total number of files
    - Higher capacities mean more hardware

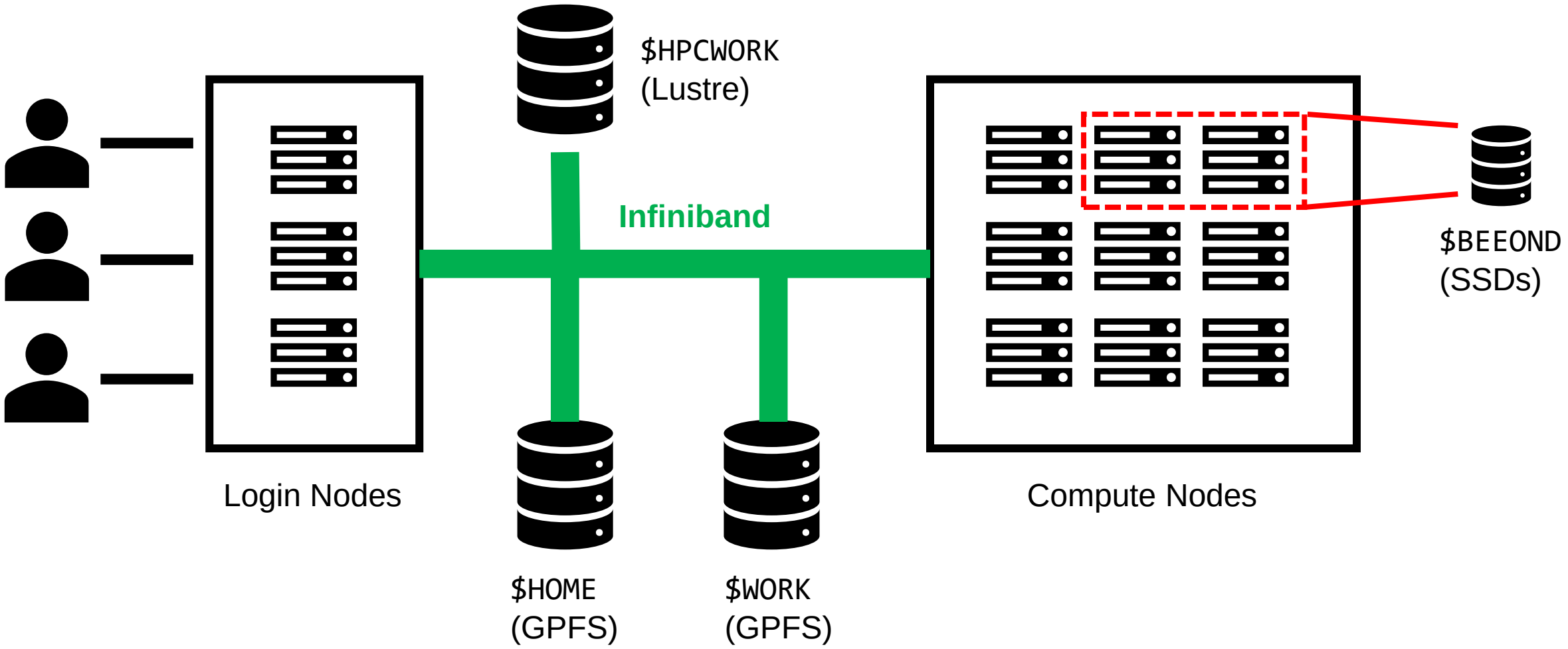


CLAIX – I/O Storage Overview

Access	Filesystem	Max. Quota	File Quota	Backup	Pros	Cons
`$HOME`	GPFS	150 GB*	~ 1 Mio.*	Tape (off-site)	Reliable Backup	Limited quota and BW
`$WORK`	GPFS	250 GB*	~ 1 Mio.*	Snapshots	Reliable	Limited quota and BW
`$HPCWORK`	Lustre	1000 GB*	~ 1 Mio.*	-	High quota and BW	Less reliable Not always good for metadata
`$BEEOND`	BeeGFS	~ 500 GB p. N.	-	-	High BW and metadata Combined storage	Temporary Only exclusive nodes Transfers required
`$TMP`	XFS	~ 500 GB p. N.	-	-	High BW and metadata	Temporary Transfers required

* Defaults for users and smaller project types

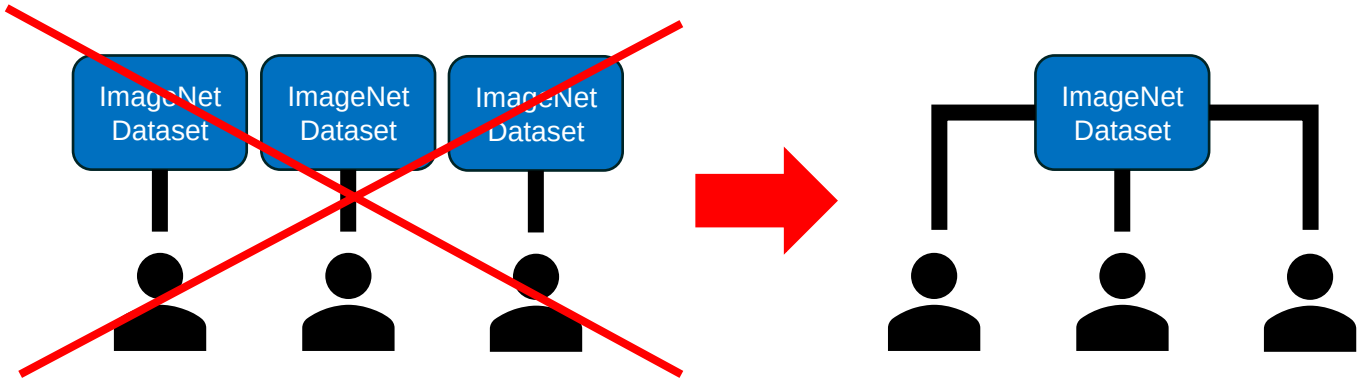
CLAIX – Connectivity



Current Challenges

- **Processing large amounts of data**
  - Large data volume
  - Large number of files
  - Example: 100 TB and 50 Mio. files

  ➢ Might exceed quotas
  ➢ Lustre not always best when working with many small files



- **Common open-source datasets**
  - Example: ImageNet (ILSVRC2012, Winter21K)
  - Goals
    - Avoid redundant per user/project copies
    - Provide such datasets at central location (Note: read-only)
  - **But:** What if users want to process dataset differently?

- **Additional datasets possible:**
  - Open datasets
  - Interesting for larger user base
  - Pre-processed versions have to be commonly known/used
  - Contact: servicedesk@itc.rwth-aachen.de

Example Scenarios

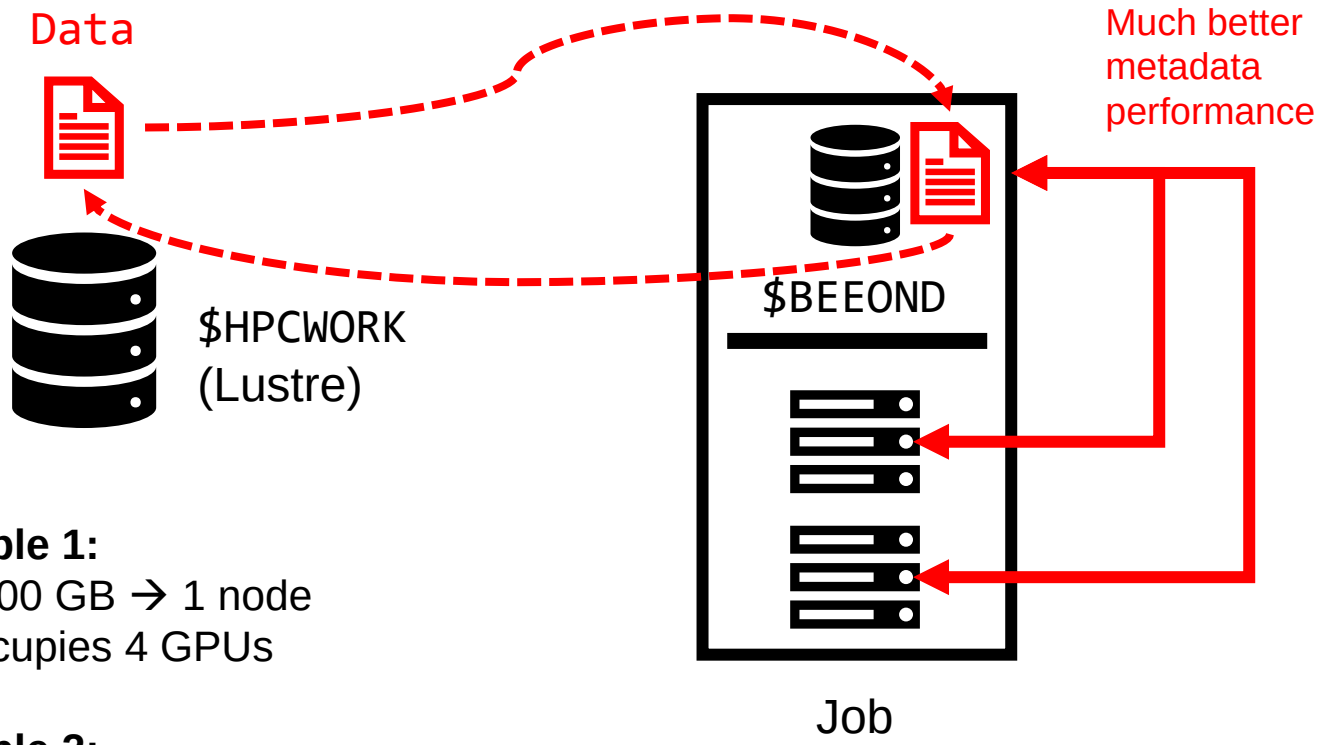
- **Question:** Where to store my data and how to access it?
  - **Scenario 1:**
    - Dataset size: small (e.g. < 250 GB)
    - Number of files do not exceed quota
  - **Scenario 2:**
    - Dataset size: medium – large
    - Number of files do not exceed quota
  - **Scenario 3:**
    - Dataset size: medium – large
    - Number of files exceed quota !!!
  - **Scenario 4:**
    - Multiple single GPU jobs
    - Each job uses the same dataset

- **Option 1:** Store data on `$WORK`
- **Option 2:** Store data on `$HPCWORK` if data accessed in parallel

- **Option:** Store data on `$HPCWORK`
- High bandwidth, especially with parallel accesses
- Metadata performance might not be good for many small files
- You will create network traffic

Example Scenarios

- **Scenario 3:**
  - Dataset size: medium – large
  - Number of files exceed quota !!!
  - Solution / Approach:
    - Pack files into multiple tar balls
    - Copy data to local SSDs at start of your job
    - Unpack or use ratarmount?
  - Boundary conditions:
    - Volume limited by capacity of SSDs
  - Further considerations:
    - Copying data and unpacking might take some time

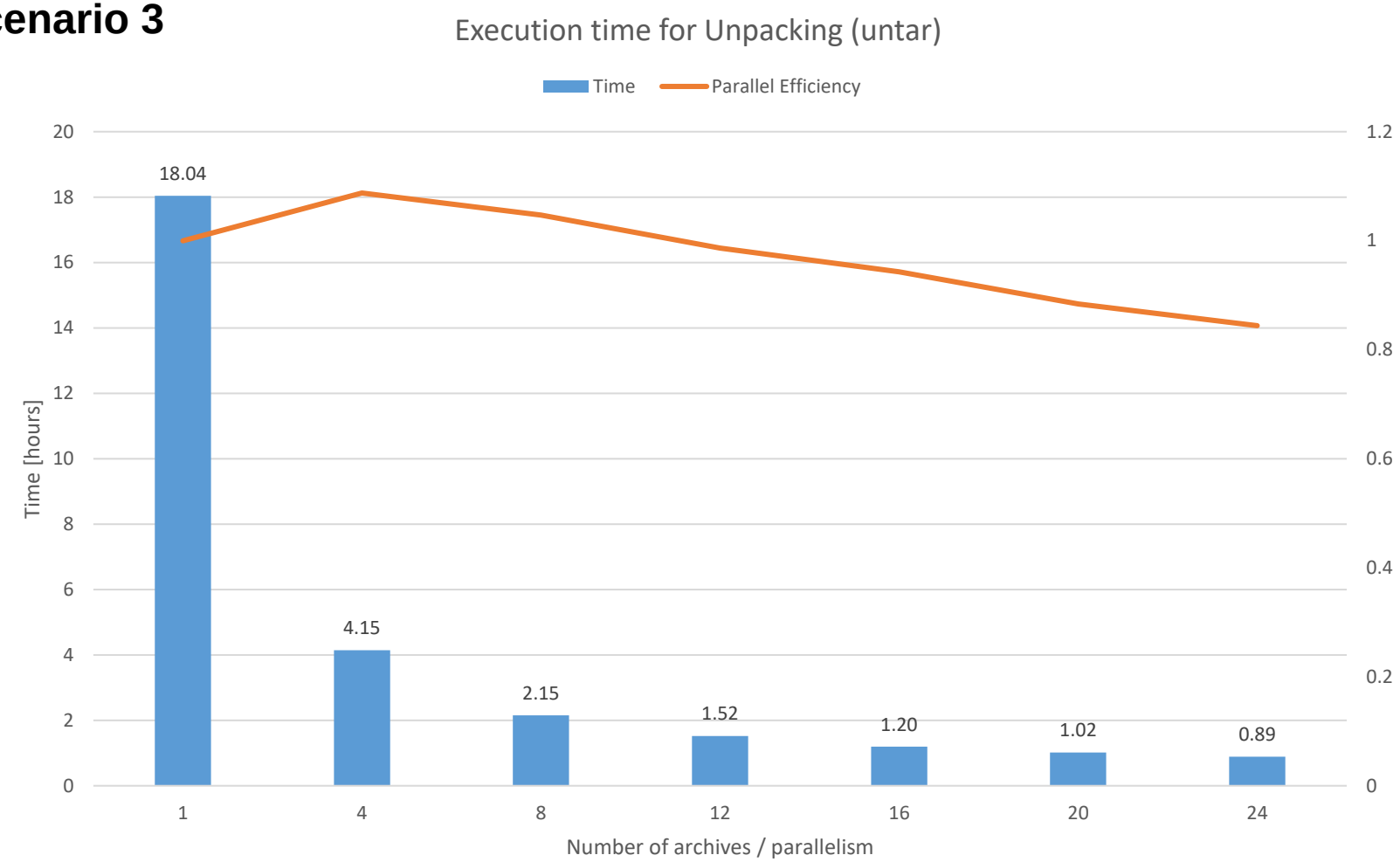


Example 1:
- < 500 GB → 1 node
- Occupies 4 GPUs

Example 2:
- 2 TB → 4 nodes
- Occupies 16 GPUs

Example Scenarios

- **Further Recommendations for Scenario 3**
  - Use multiple tar achives
  - Untar in parallel
  - **Example: Untar performance**
    - 2 TB of data
    - 1 Mio. files
    - `$BEEOND` of multiple nodes
    - Varying number of archives
  - ➢ Copying data to `$BEEOND`: 19 min



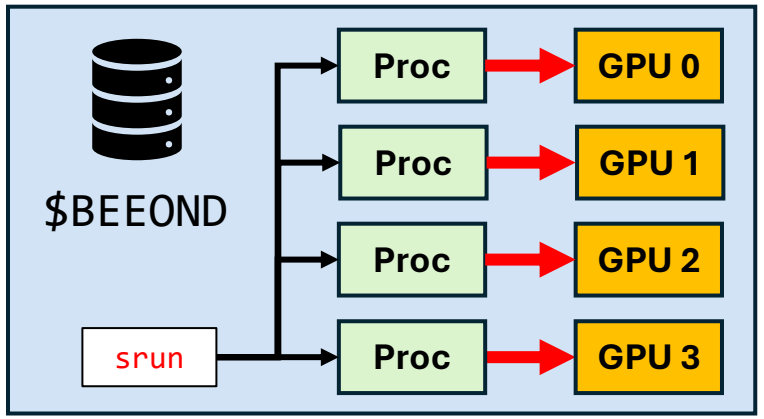
Example Scenarios

- **Scenario 4:**
  - Multiple single GPU jobs
  - Each job uses the same dataset (< 500 GB)

  - **Example:** Train models with different hyperparamters
    - No alterations on dataset (read-only)
    - No large runtime deviations

  - Naiive approach:
    - Submit single GPU jobs
    - Access dataset from `$WORK` or `$HPCWORK`

  - Optimization:
    - Group multiple jobs into one:
      e.g. 4 individual GPU instances in one 4-GPU job
    - Utilizes local `$BEEOND` storage on node
      - One time copy at start
      - All instances benefit faster local access




```
#!/usr/bin/zsh
#################################################################
### Slurm flags
#################################################################
#SBATCH --partition=c23g
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=4          → (MPI) processes
#SBATCH --cpus-per-task=24           → cores per process
#SBATCH --gres=gpu:4                 → GPUs per node
#SBATCH --beeond                     → Requests BeeOND, node is exclusive

#################################################################
### Execution
#################################################################
srun --wait=0 myscript.sh            → Each process executes script
                                       and determines its parameters
                                     → Jobs wait for each other at the end
```

Thank you!