# Data Access and Storage Systems at DESY

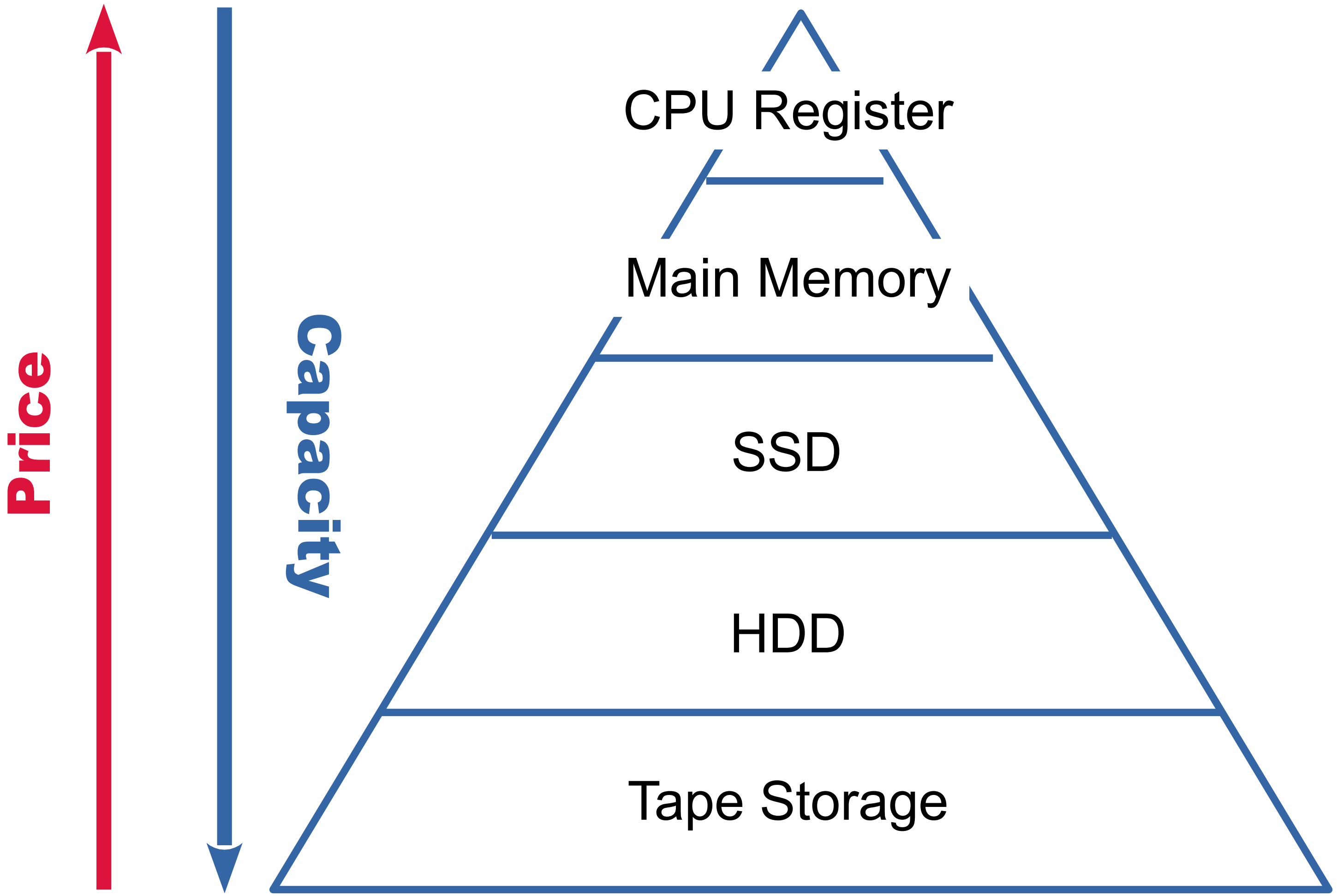
**FH SciComp Workshop 2025**

Tigran Mkrtchyan for DESY-IT
Hamburg, 03.07.2025

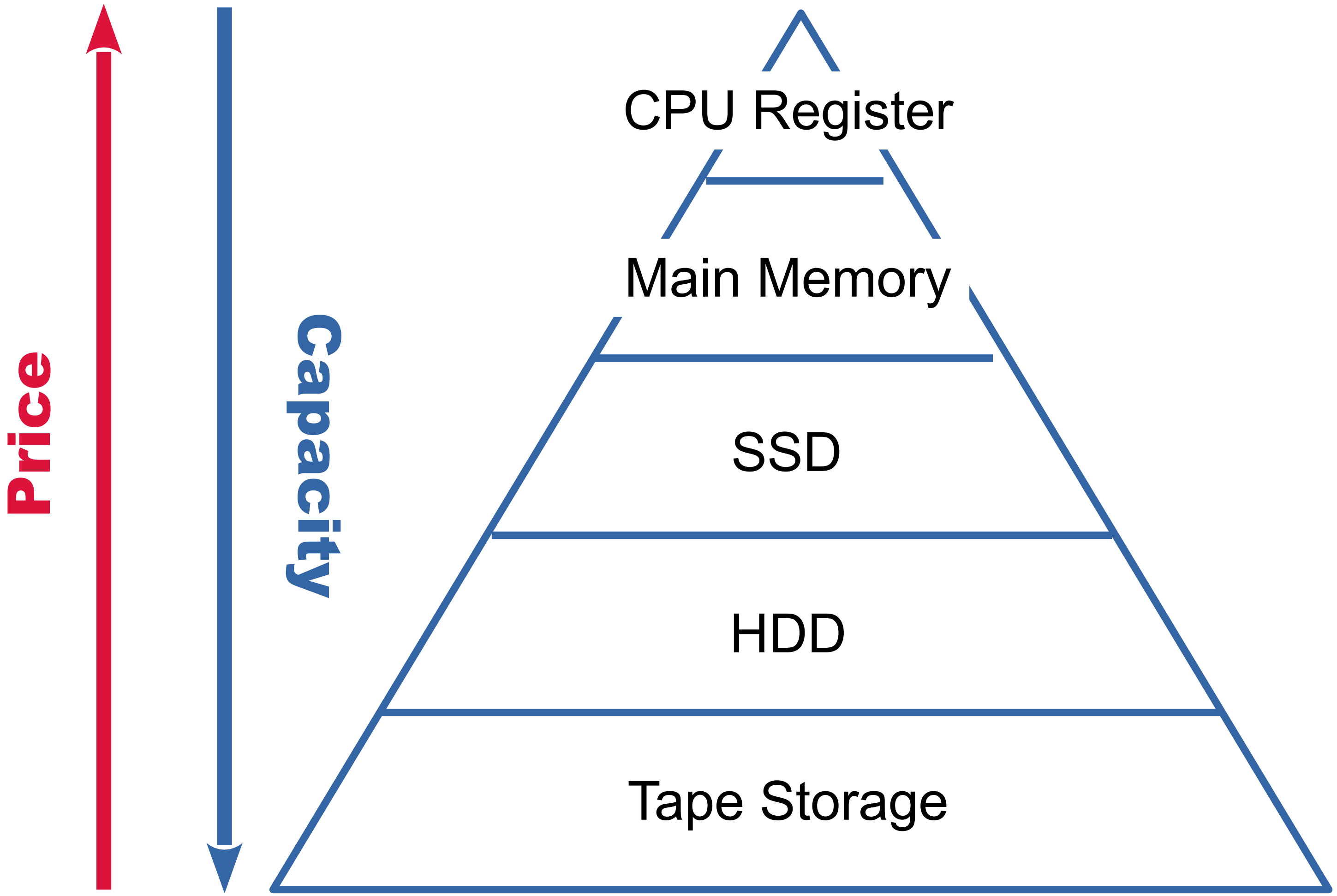
# Motivation

- What DESY-IT provides to store scientific data
- Give you a guideline to choose the right system for a  specific workload

# Storage Technology Hierarchy



# Storage Technology Hierarchy

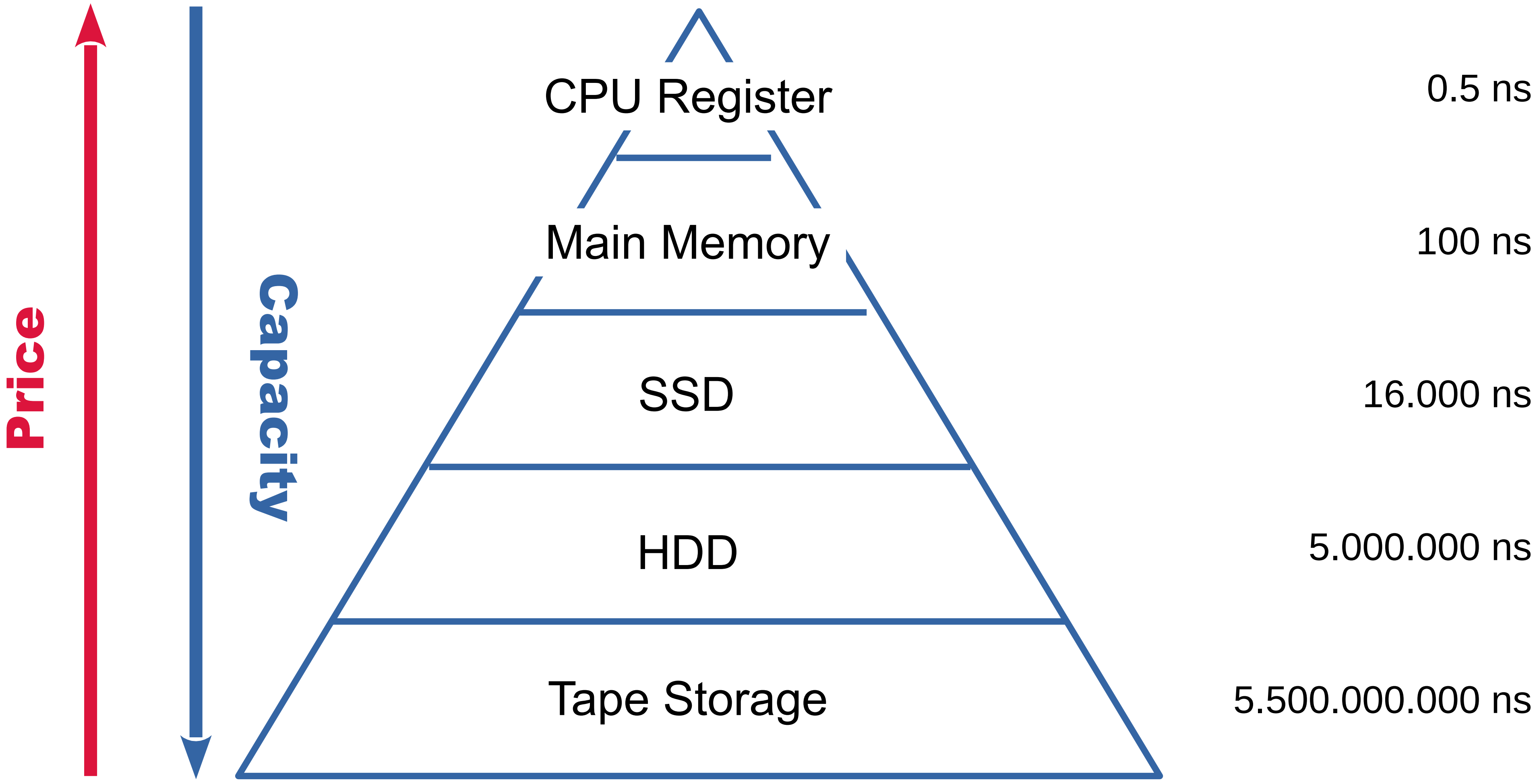


Price

Capacity

| | |
|---|---|
| CPU Register | ∞ € / TB |
| Main Memory | 5.000 € / TB |
| SSD | 100 € / TB |
| HDD | 15 € / TB |
| Tape Storage | 5 € / TB |

# Storage Technology Hierarchy

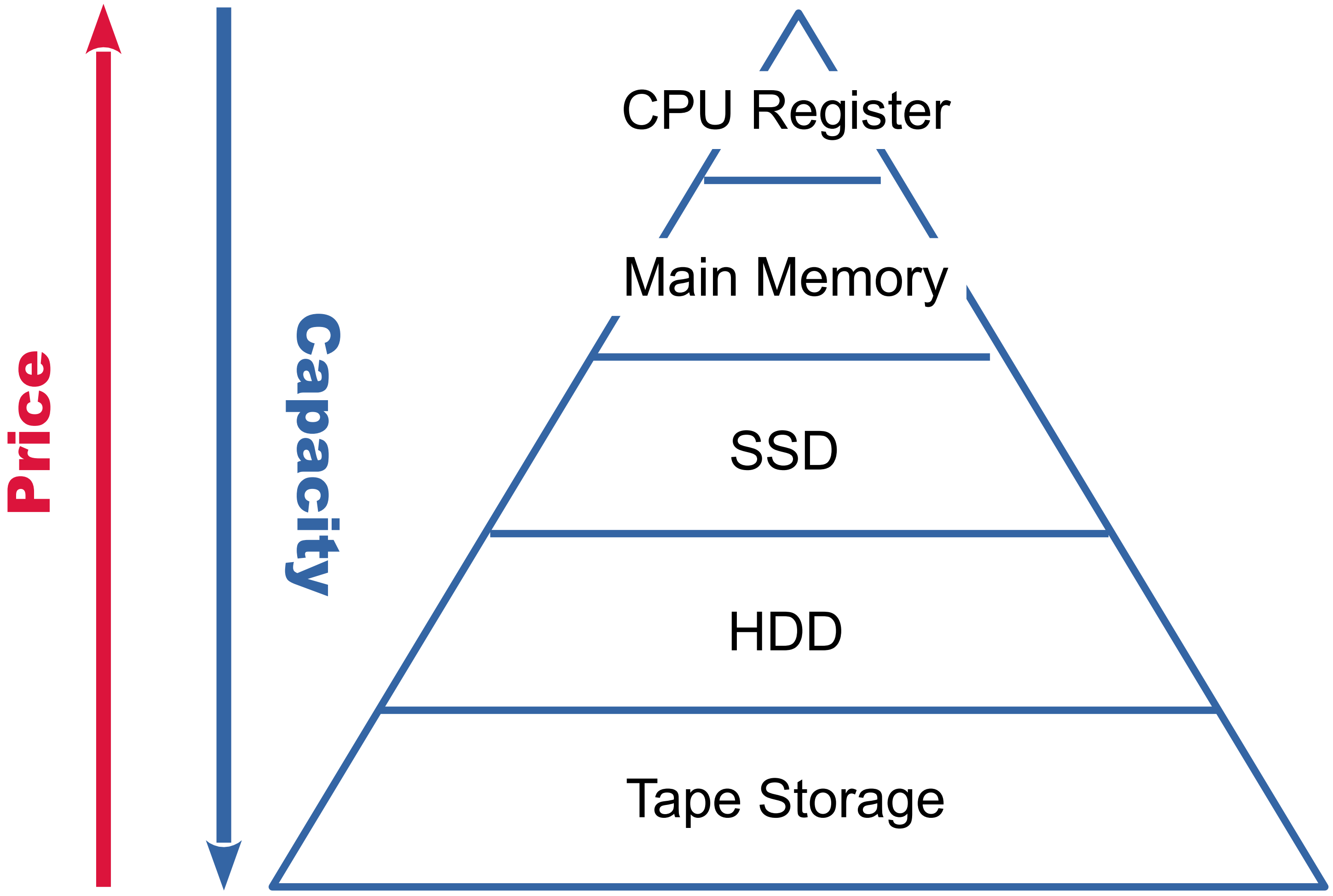


Price

Capacity

| | |
|---|---|
| CPU Register | 0.5 ns |
| Main Memory | 100 ns |
| SSD | 16.000 ns |
| HDD | 5.000.000 ns |
| Tape Storage | 5.500.000.000 ns |

# Storage Technology Hierarchy



Price

Capacity

| | | |
|---|---|---|
| CPU Register | 0.5 ns | 1 s |
| Main Memory | 100 ns | 3 min |
| SSD | 16.000 ns | 9 h |
| HDD | 5.000.000 ns | 116 days |
| Tape Storage | 5.500.000.000 ns | 349 years |

# Block-, File-, Object- Storage

### Blocks

- Data organized and accessed in fixed blocks of 512 – 4096 bytes
- Blocks are identified by Logical Block Address, which is a numerical value that specifies the location of the block within the storage device
- Additional system is required to keep track of user’s logical data and blocks
- Atomic update of a single block

### File(system)s

- Built on top of blocks storage
- Addresses data by bytes
- Organizes blocks into logical units - files
- Associates files with metadata, like name, size, access rights, etc...
- Provides interface (API) to access data without violating integrity
- Atomic file creation

### Objects

- Treats data as a single, non composite object (just like an elementary particle 😄)
- Object identified by unique identifiers
- Flat namespace
- Provides API for CRUD operations
- Atomic operation on a single object

# Block-, File-, Object- Storage

## Blocks

- Data organized and accessed in fixed blocks of 512 – 4096 bytes
- Blocks are identified by Logical Block Address, which is a numerical value that specifies the location of the block within the storage device
- Additional system is required to keep track of user’s logical data and blocks
- Atomic update of a single block

## File(system)s

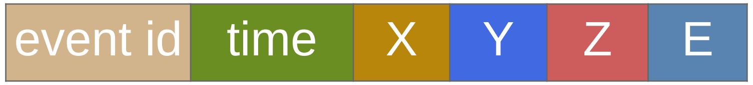
- Built on top of blocks storage
- Addresses data by bytes
- Organizes blocks into logical units - files
- Associates files with metadata, like name, size, access rights, etc...
- Provides interface (API) to access data without violating integrity
- Atomic file creation

## Objects

- Treats data as a single, non composite object (just like an elementary particle 😄)
- Object identified by unique identifiers
- Flat namespace
- Provides API for CRUD operations
- Atomic operation on a single object

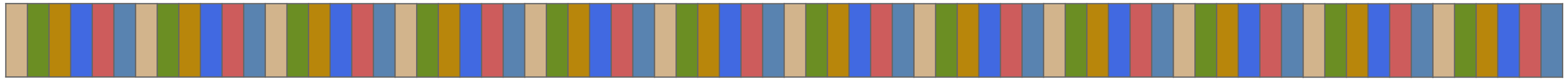
# Columnar- vs. Row- File Formats

User data

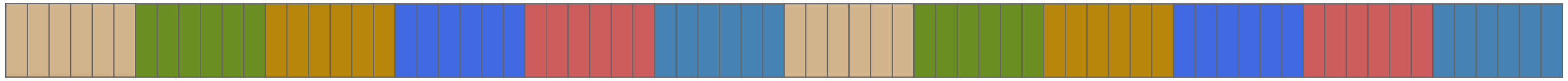


- Read only X values: 12 seeks
- Read full event #2: 1 seek

ROW oriented file format



- Read only X values: 1 seeks
- Read full event #2: 6 seek



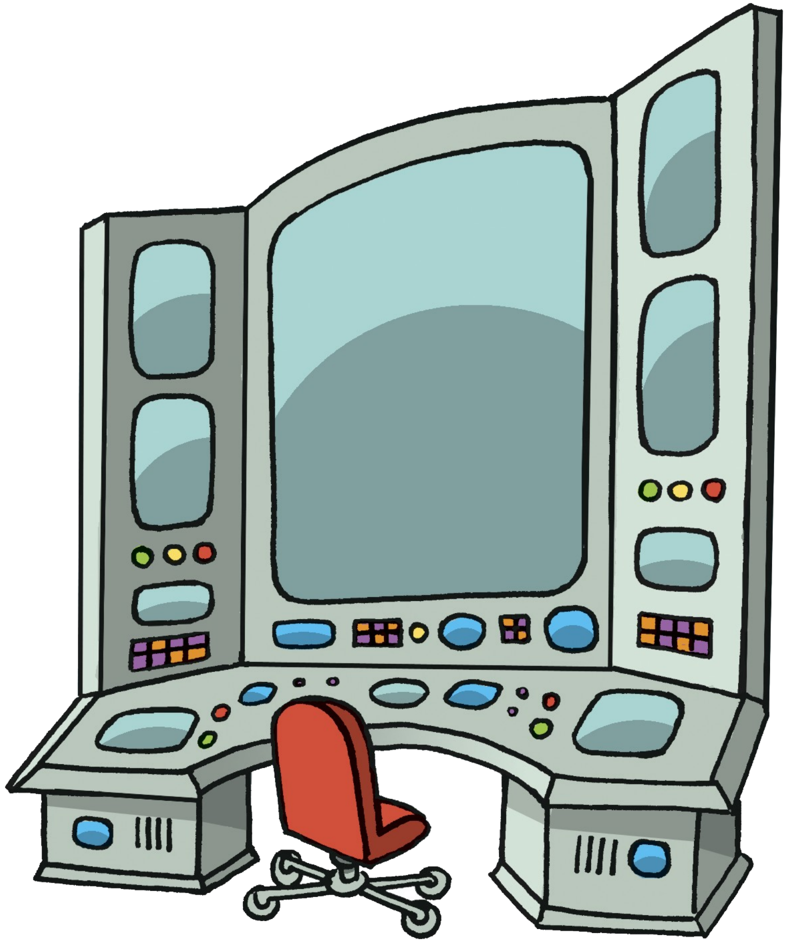
Column oriented file format

# Columnar- vs. Row- File Formats

IN FILESYSTEMS, AS IN THE QUANTUM WORLD, PERFORMANCE COMES AT A COST. YOU MAY HAVE A FILE OPTIMIZED FOR THE EVENT LOOP OR FOR COLUMNAR ANALYSIS, BUT NOT BOTH WITH CERTAINTY.

# Distributed Storage

- Share data between multiple client nodes
  - Aggregate storage from multiple nodes
  - Capacity and bandwidth grow with the number of storage nodes
- Location transparency
  - The name of a file doesn't depend on its physical location
- Location independence
  - The name of a file doesn't change when its physical location changes
- Network transparency
  - Remote data is accessed the same way as on a local filesystem, without needing application changes
- User mobility
  - Users can access the same file by the same file name from different hosts

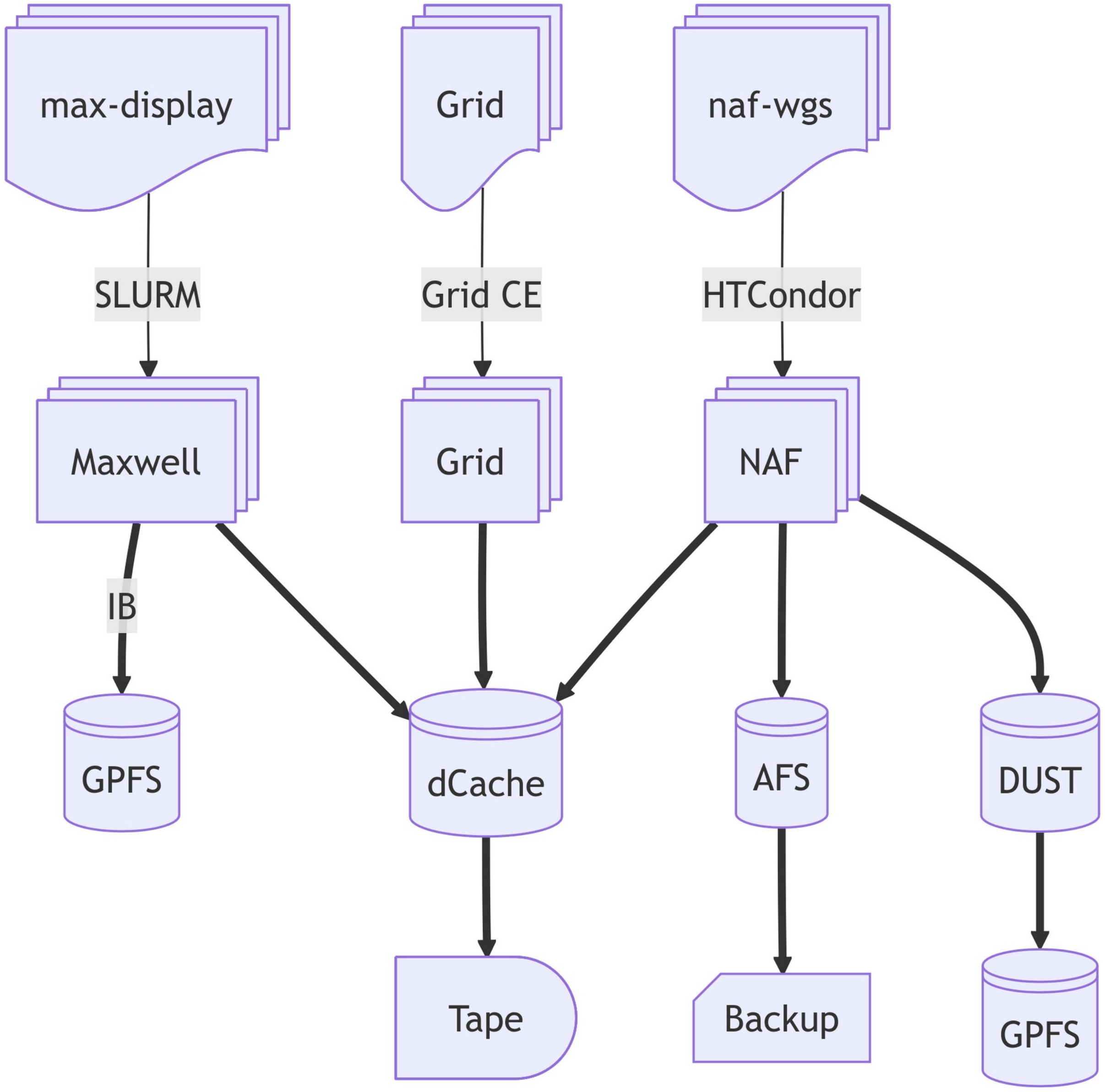


# Protocols vs Storage Systems

- Protocols
  - HTTP, WebDAV
  - NFS, SMB
  - XrootD*, DCAP
  - S3
- Storage systems / Servers
  - Apache, NGINX
  - NetApp
  - GPFS, AFS, DUST
  - dCache, XrootD, EOS
  - Amazon S3, MinIO
  - CEPH

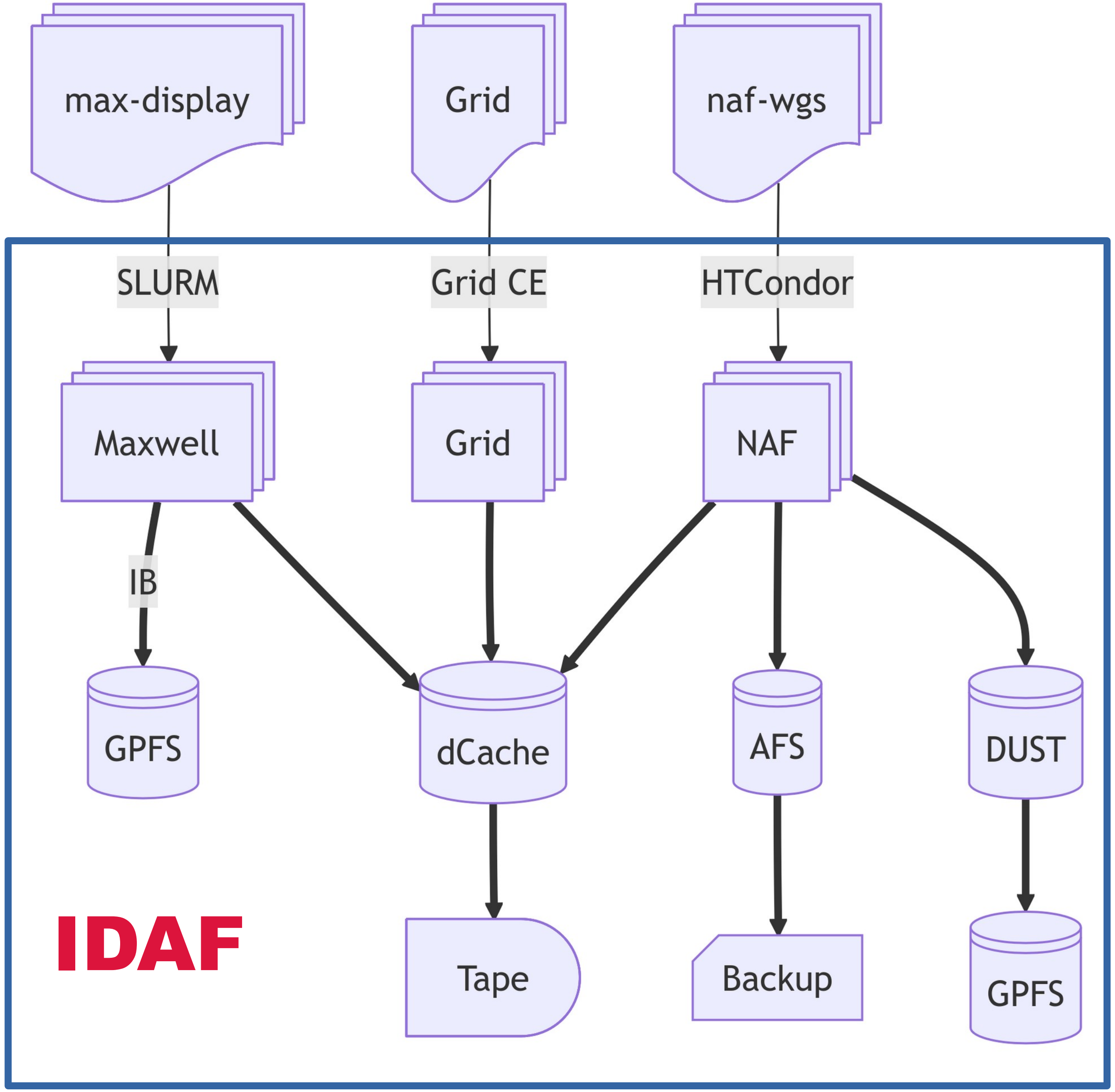
# Storage Systems at DESY

- GPFS
  - HPC filesystem, interconnected with low-latency network to Maxwell cluster.
- dCache
  - Large data store. Multi-protocol, multi-authentication support. Direct connection with tape library.
- AFS
  - Home directory on WGS and NAF nodes. Nightly incremental backups.
- DUST
  - Limited scratch space. Re-export of GPFS over NFS.



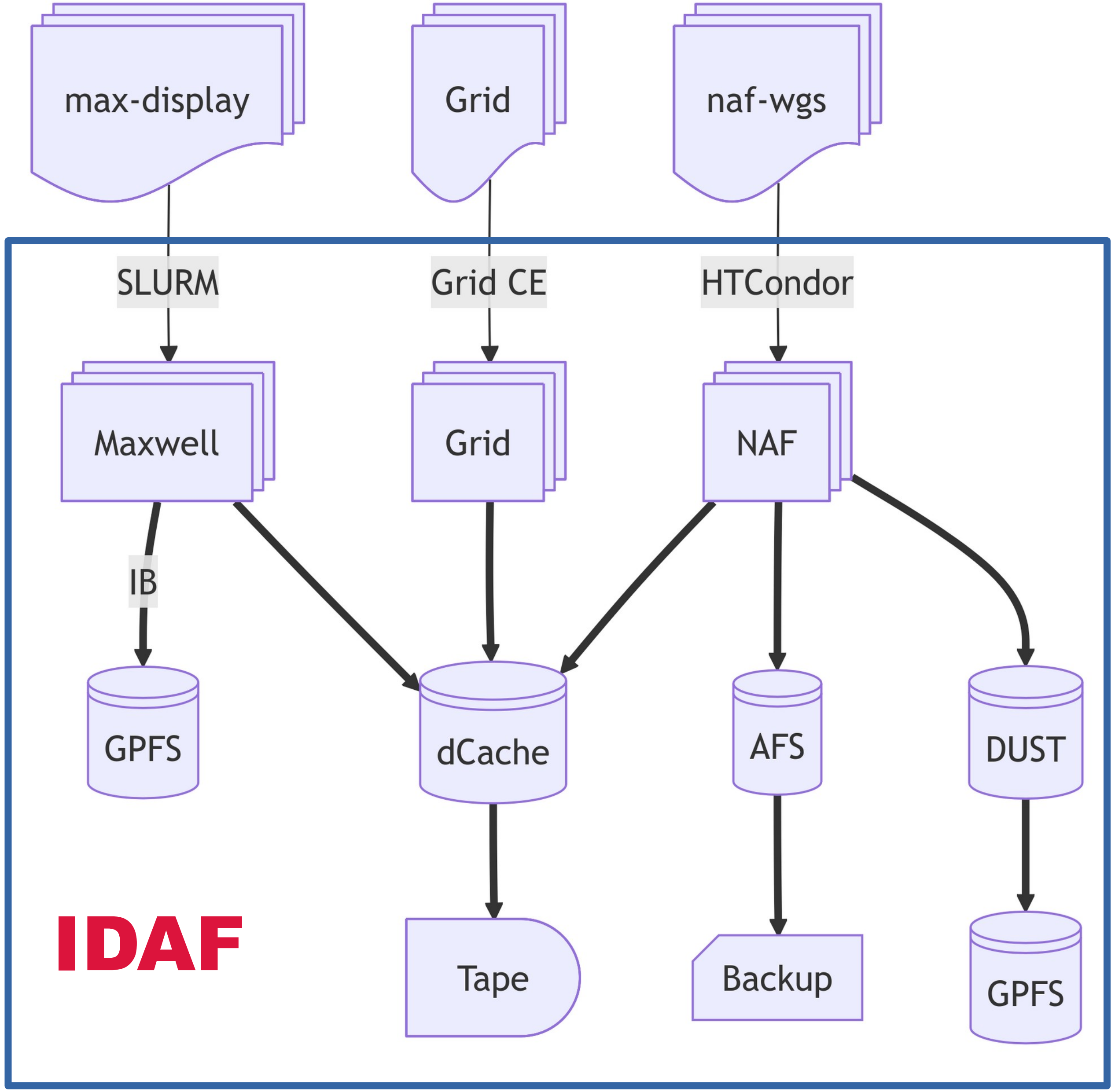
# Storage Systems at DESY

- GPFS
  - HPC filesystem, interconnected with low-latency network to Maxwell cluster.
- dCache
  - Large data store. Multi-protocol, multi-authentication support. Direct connection with tape library.
- AFS
  - Home directory on WGS and NAF nodes. Nightly incremental backups.
- DUST
  - Limited scratch space. Re-export of GPFS over NFS.



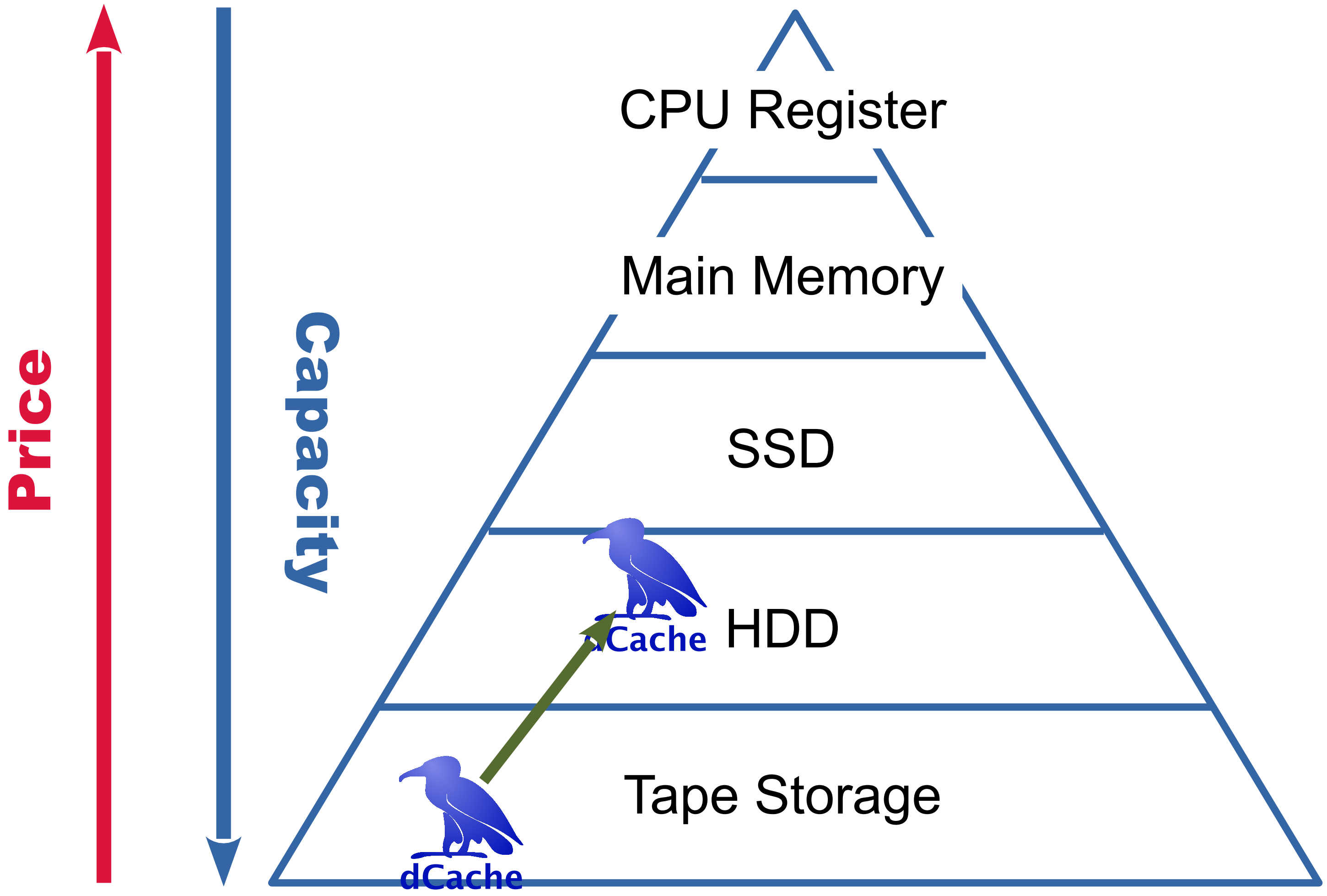
# Write Consistency

- GPFS
  - Multiple writer support. Updates visible to concurrent readers and writers.
- dCache
  - Single writer. A file is not accessible to readers as long as open-for-write. On-close immutable.
- AFS
  - Multiple writer support. Last writer wins.
- DUST
  - Multiple write support. Readers (and other writers) will see the updates after *close* or *fsync*.



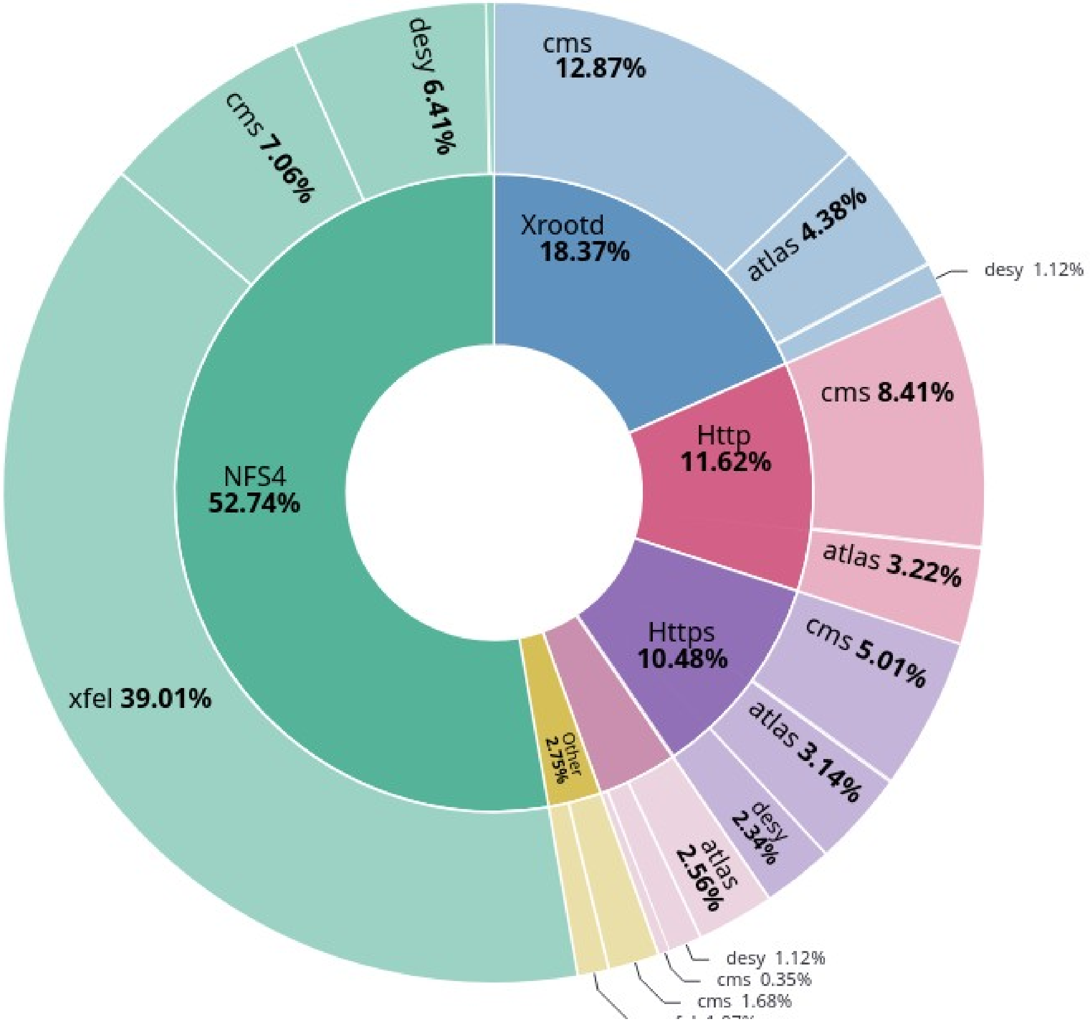
# dCache

- Highly scalable  system
  - Largest single instance at DESY 120PB on disk
- Multi-protocol storage system
  - Data access protocols
  - User authentication
- Designed for HTC workload
- Native tape integration



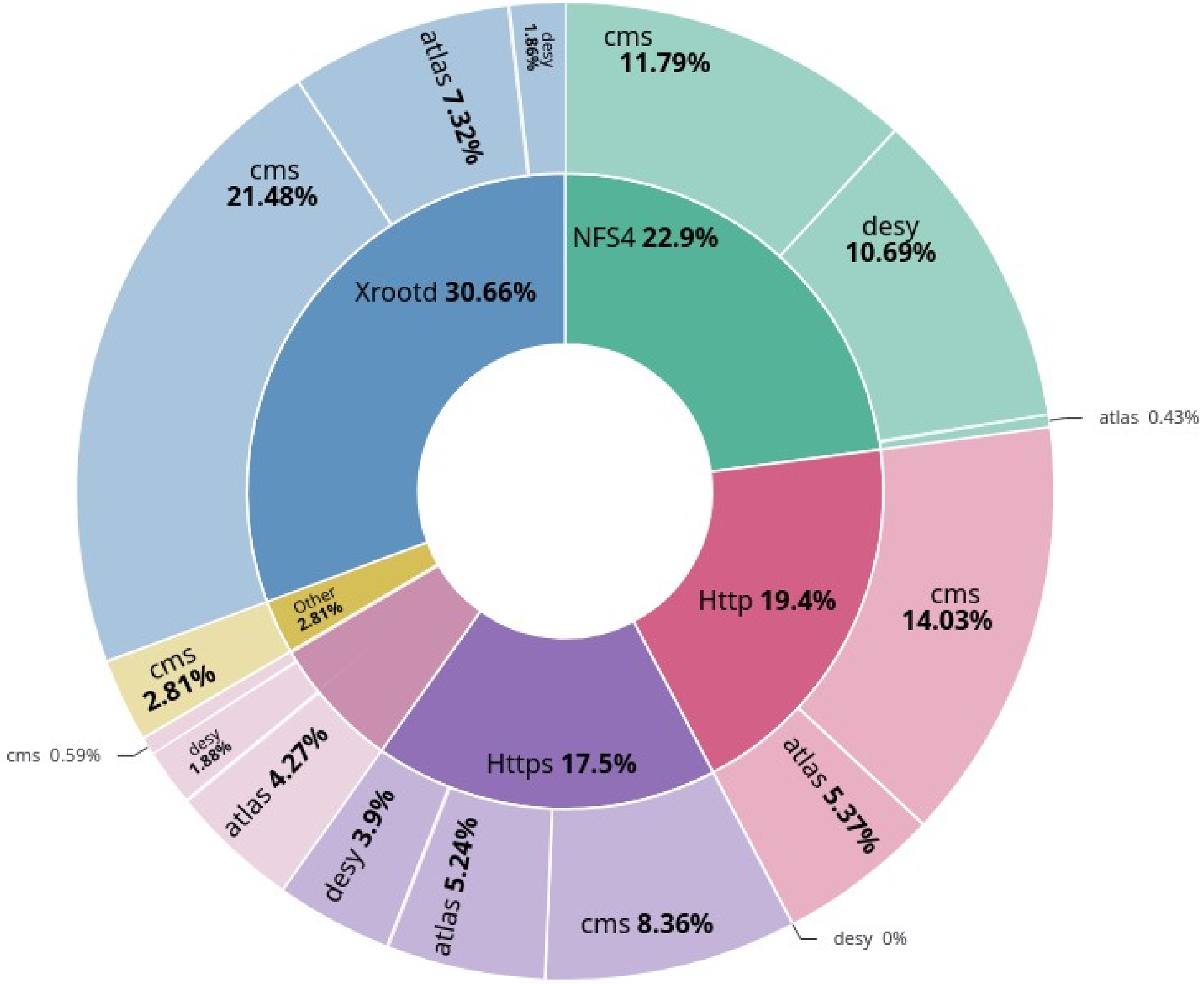
# dCache

- Highly scalable  system
  - Largest single instance at DESY 120PB on disk
- Multi-protocol storage system
  - Data access protocols
  - User authentication
- Designed for HTC workload
- Native tape integration

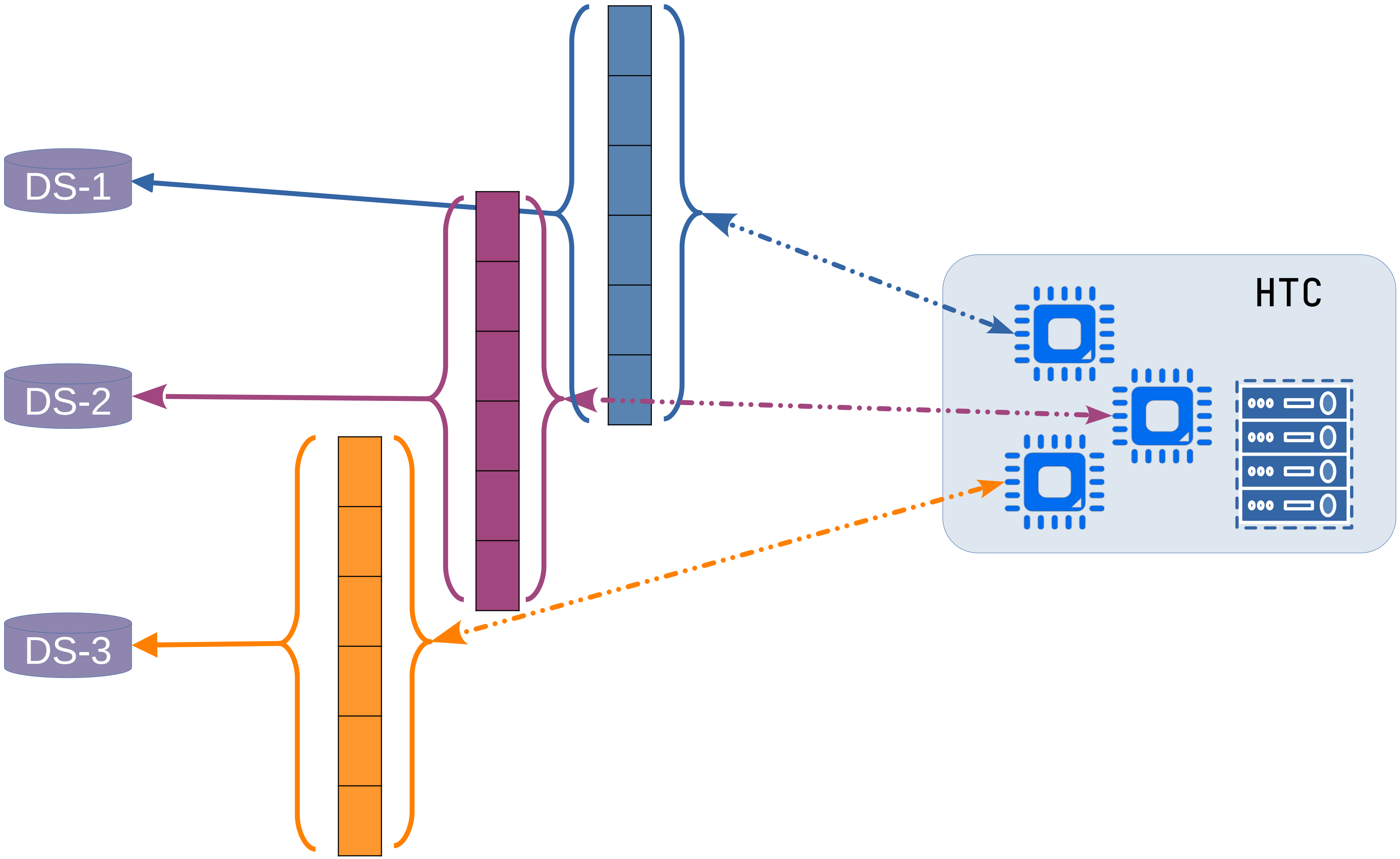


# dCache

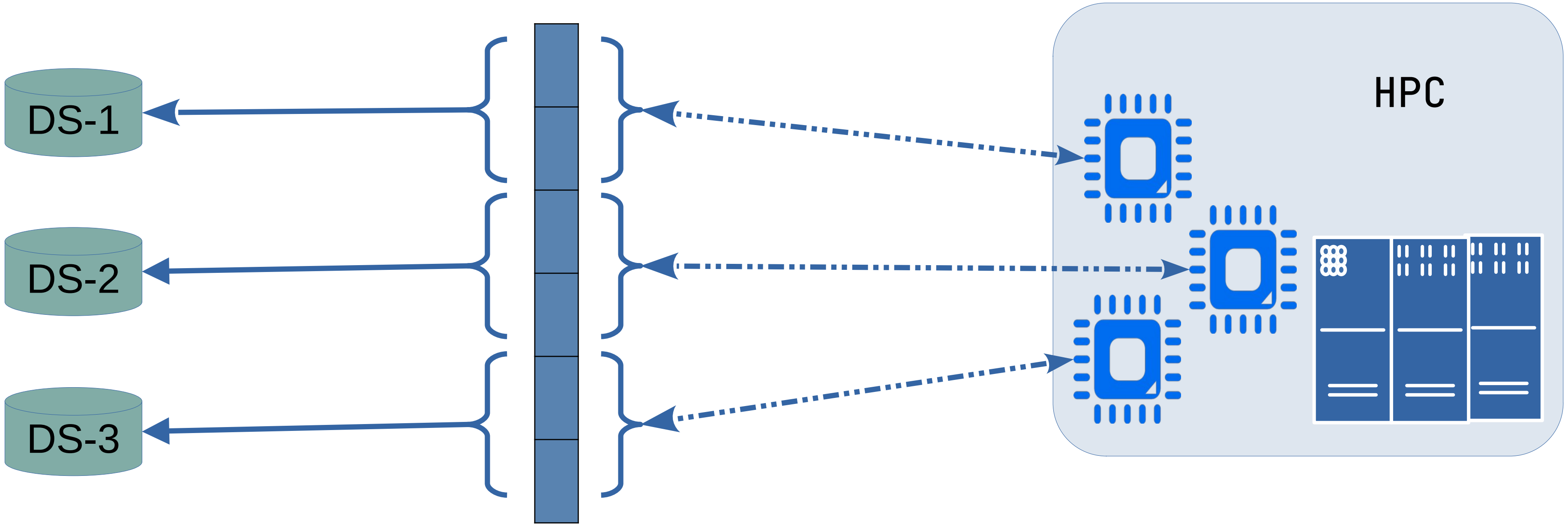
- Highly scalable system
  - Largest single instance at DESY 120PB on disk
- Multi-protocol storage system
  - Data access protocols
  - User authentication
- Designed for HTC workload
- Native tape integration



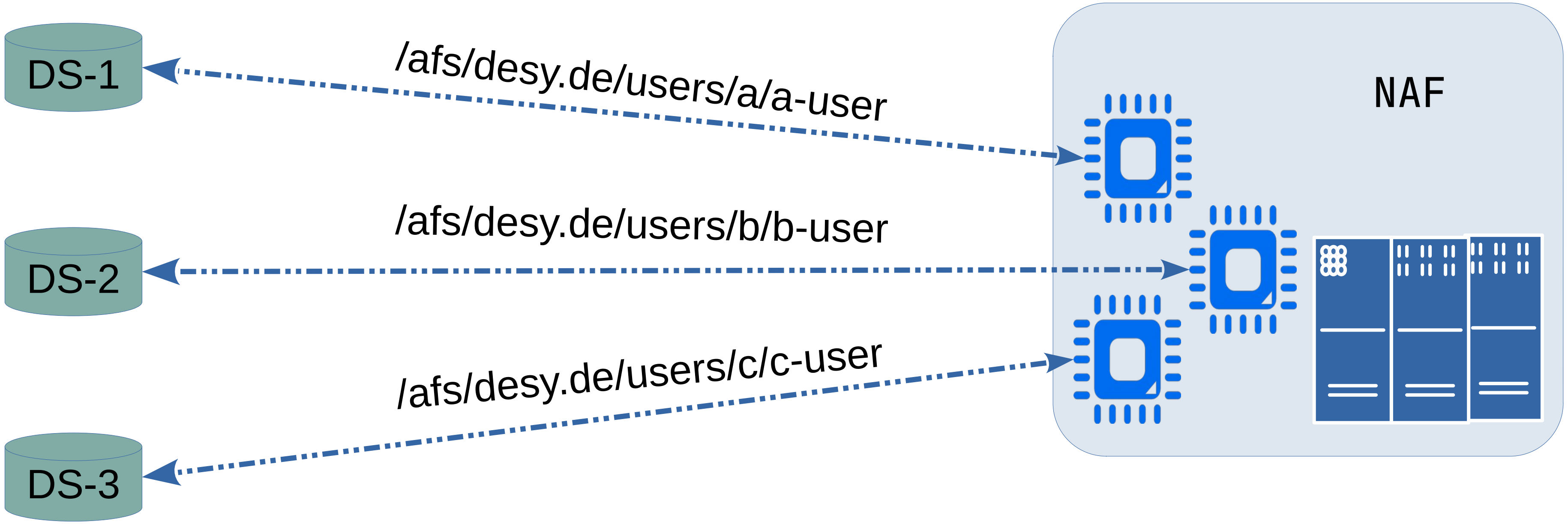
# HTC: Many Cores, Many Files



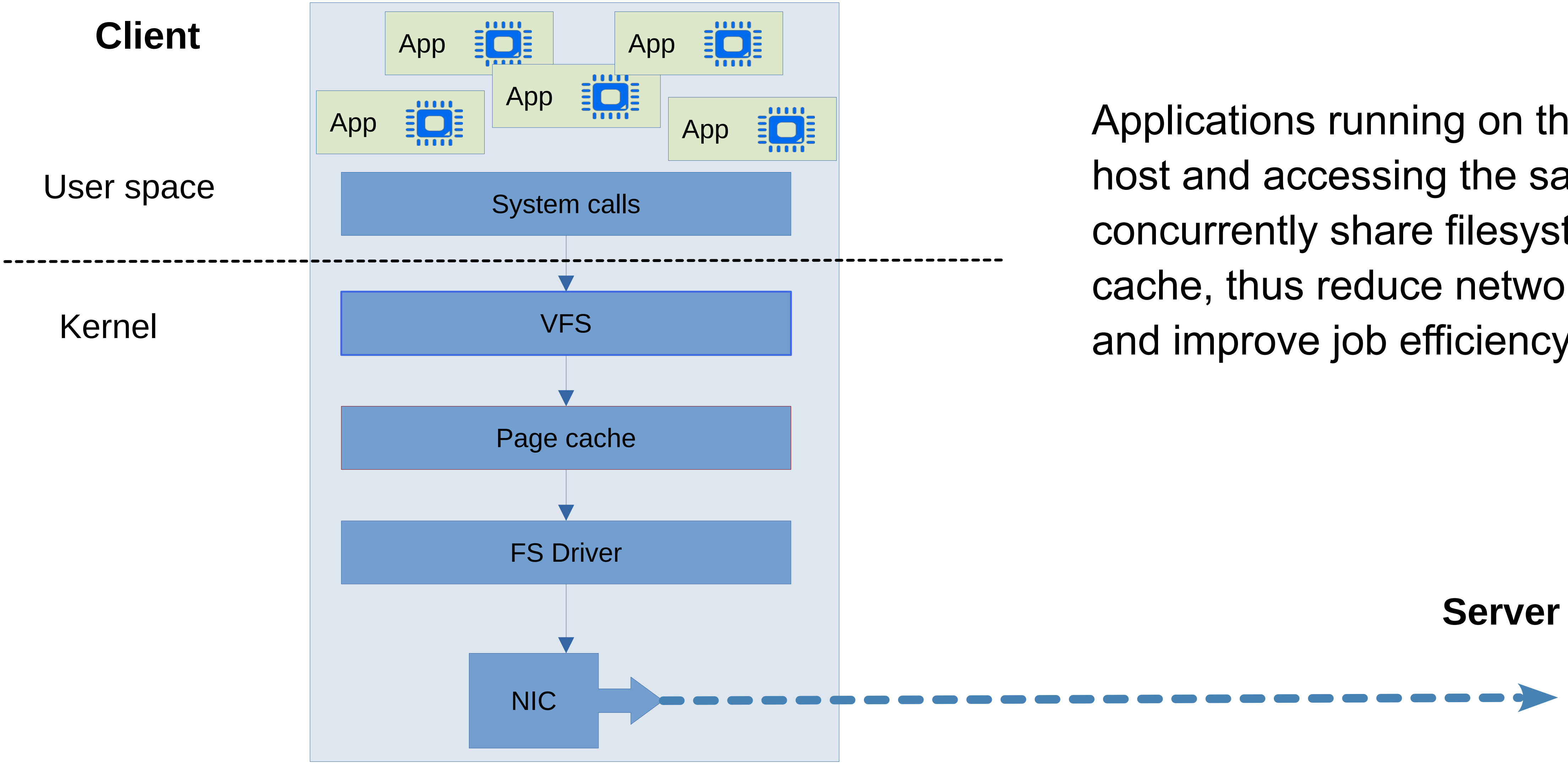
# HPC: Many Cores, Single File



# AFS: Single Volume, Single Server


DS-1
DS-2
DS-3
/afs/desy.de/users/a/a-user
/afs/desy.de/users/b/b-user
/afs/desy.de/users/c/c-user
NAF

# Client Side Caching



Applications running on the same host and accessing the same data concurrently share filesystem cache, thus reduce network traffic and improve job efficiency

# What I Should Use?

- GPFS
  - HPC workloads
    - Many processes accessing same or a small number of files for read or write.
  - Latency sensible analysis

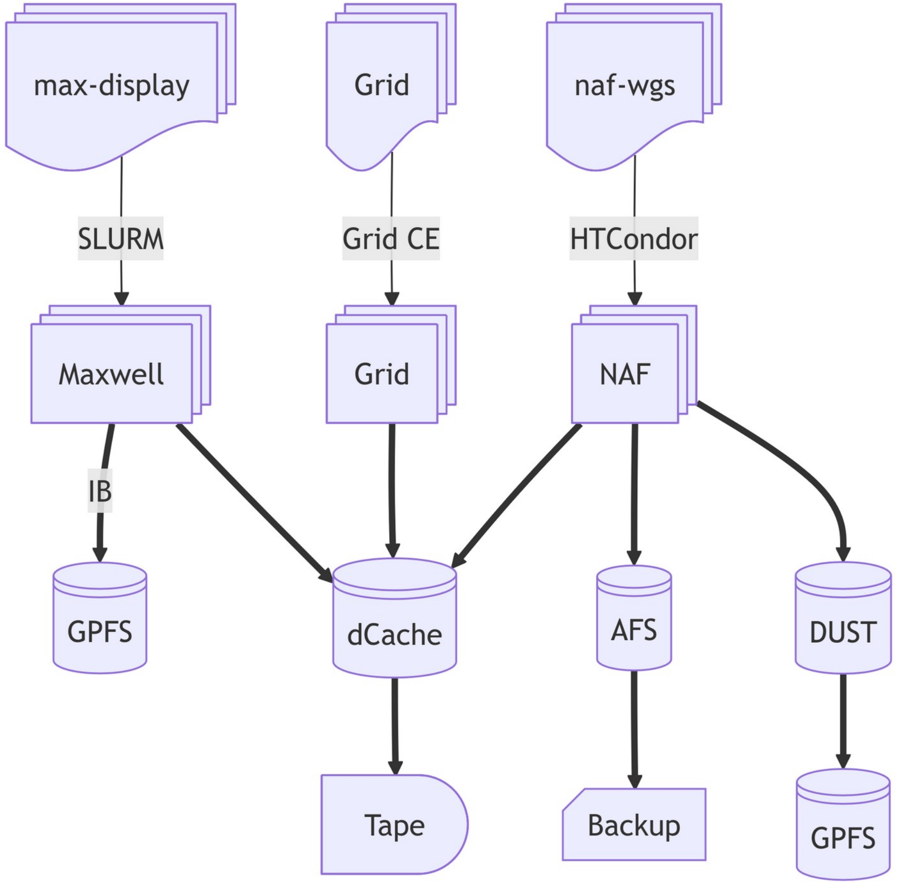
- AFS
  - Point access
    - Startup scripts
    - Jobs configurations

- dCache
  - Large volumes of immutable data.
  - HTC workloads
  - Data import/export with other sites
  - Multiple access protocols
  - Tape integration

- DUST
  - Small reproducible data sets
  - Job outputs
  - Concurrent writers
  - Local container images

# Summary

- DESY-IT provides a large set of storage system to cover various scientific use-cases
  - Some workloads require multiple solution in parallel
- There is no one size fits all
  - Contact us, if you are not sure which one is the right for you
- Share with us your data access strategies
  - Changes in experiment data access model might require cha in storage systems



# Questions