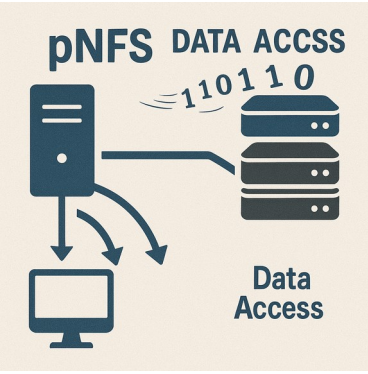


NFS Updates

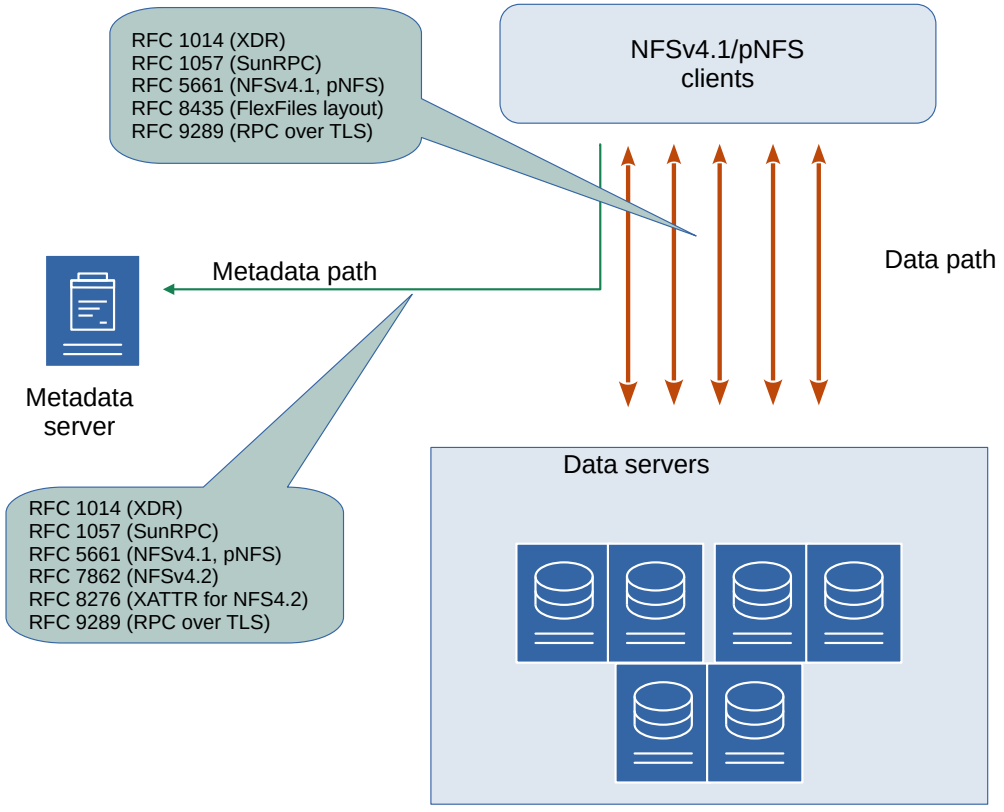
dCache user workshop, CC-IN2P3, Lyon

Tigran Mkrtchyan for the dCache collaboration

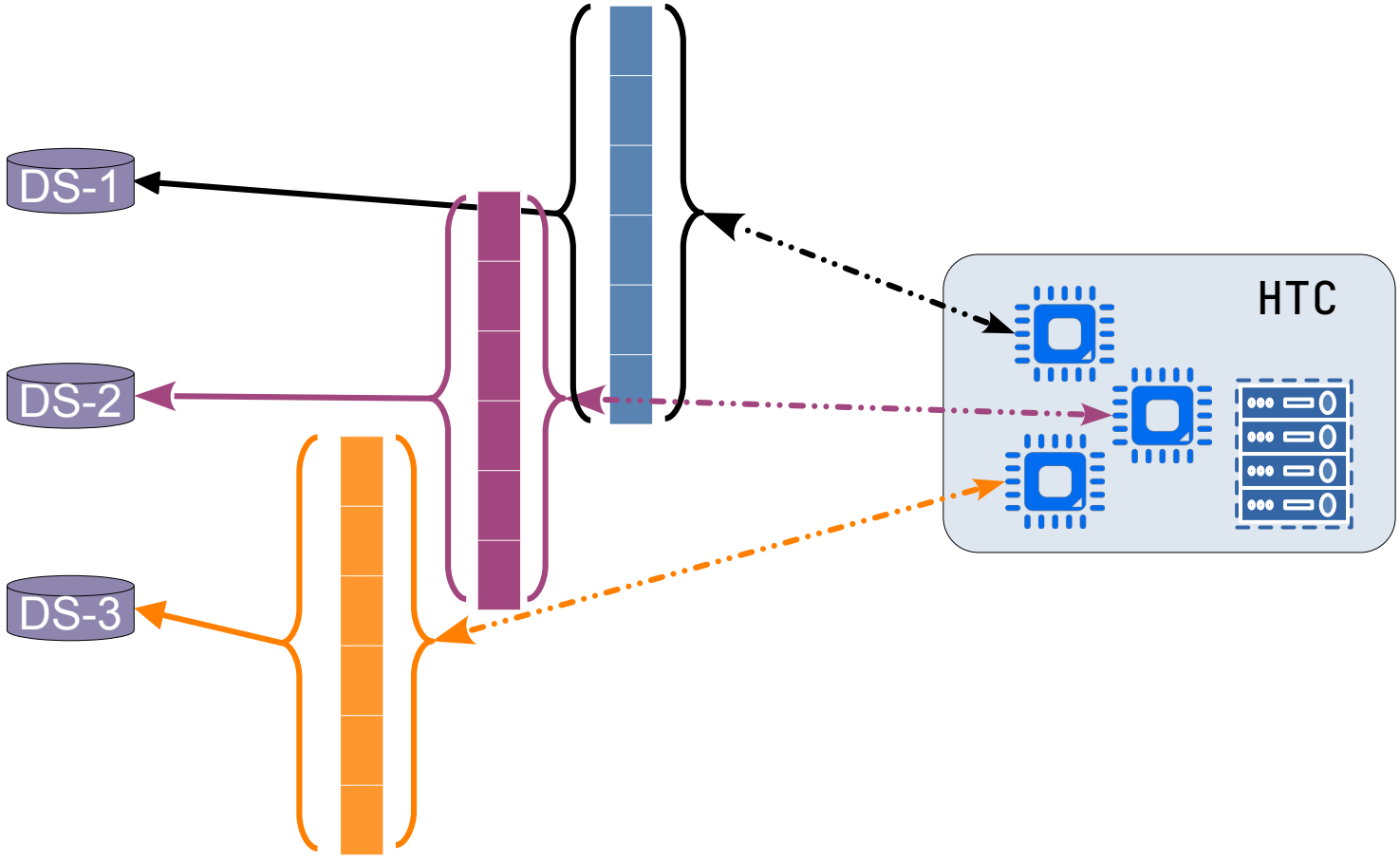
DESY. Fermilab neic NORDIC E-INFRASTRUCTURE COLLABORATION

HELMHOLTZ RESEARCH FOR GRAND CHALLENGES

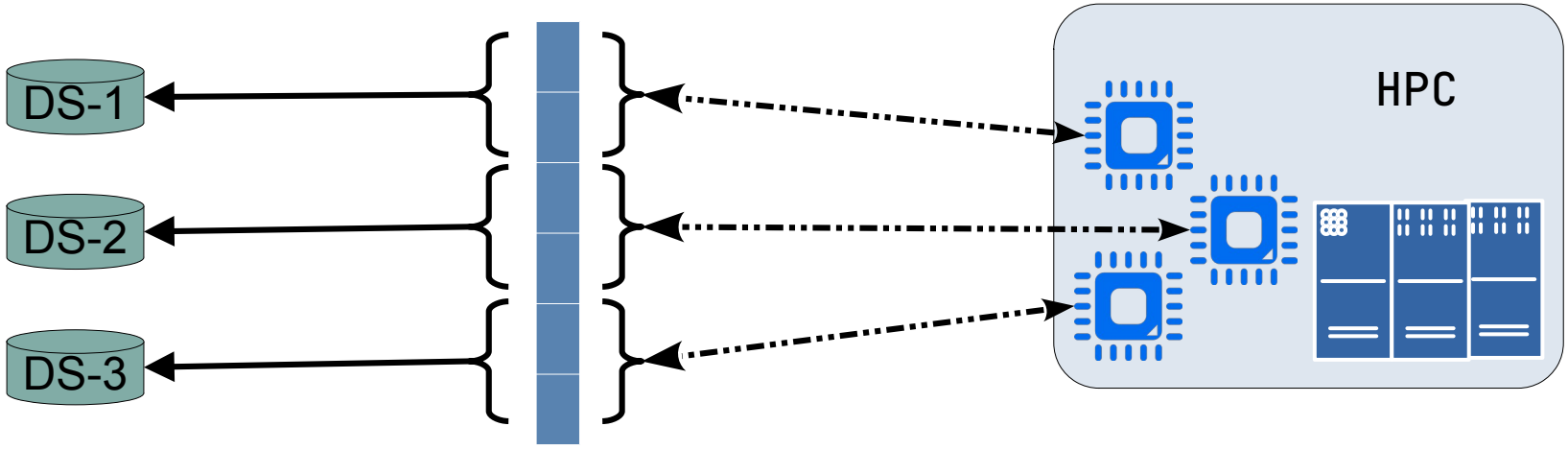
NFSv4.1/pNFS Anatomy



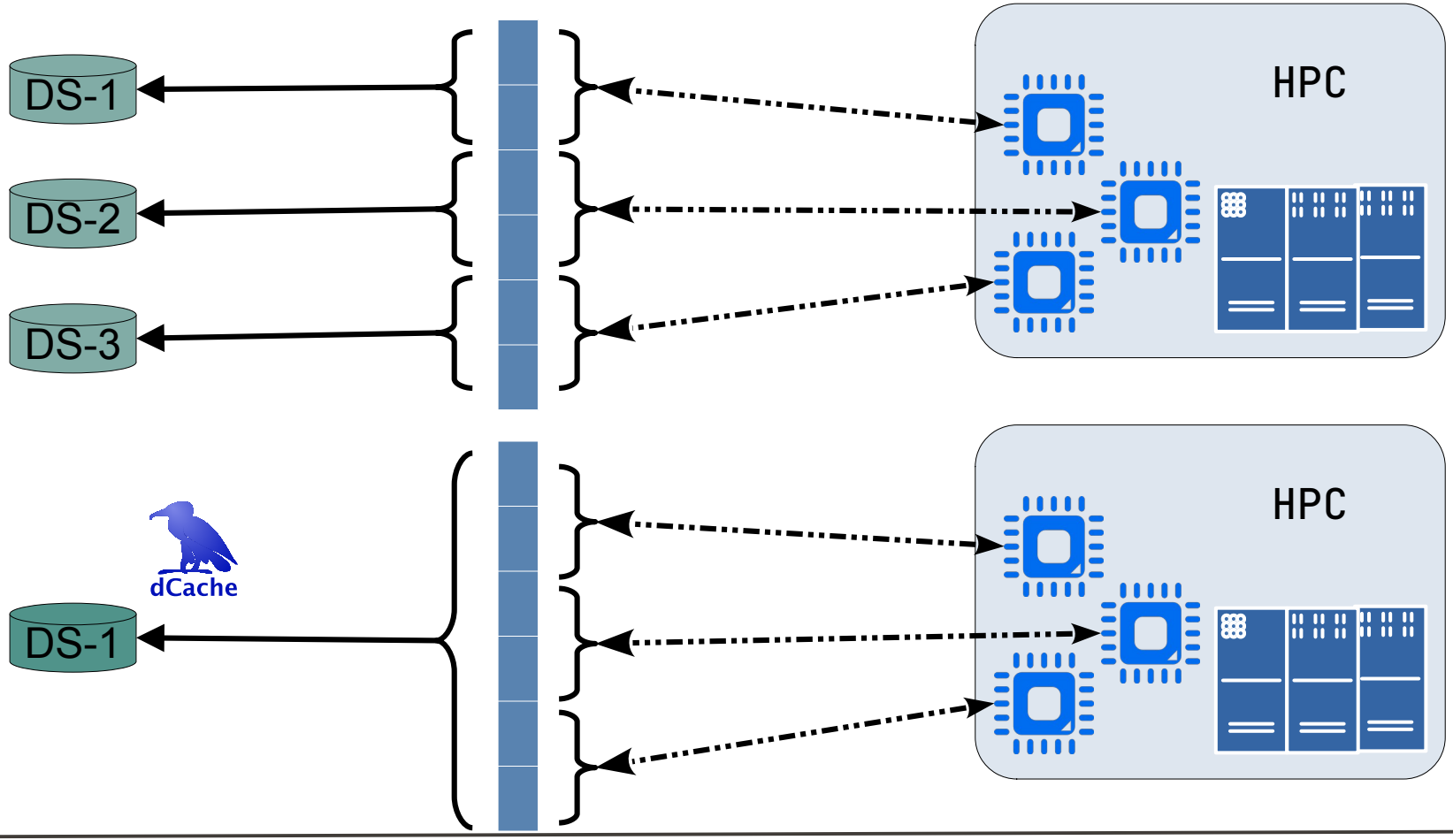
HTC: Many Cores, Many Files



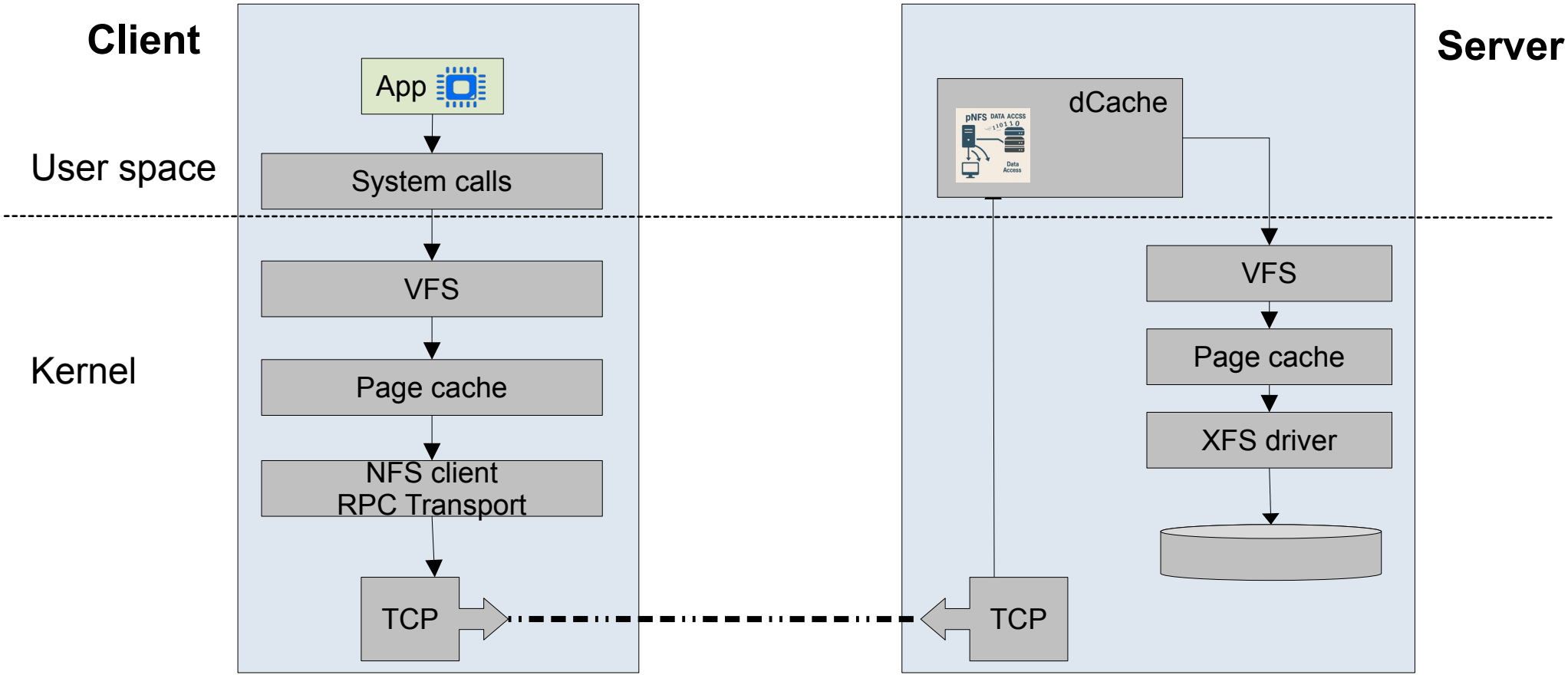
HPC: Many Cores, Single File



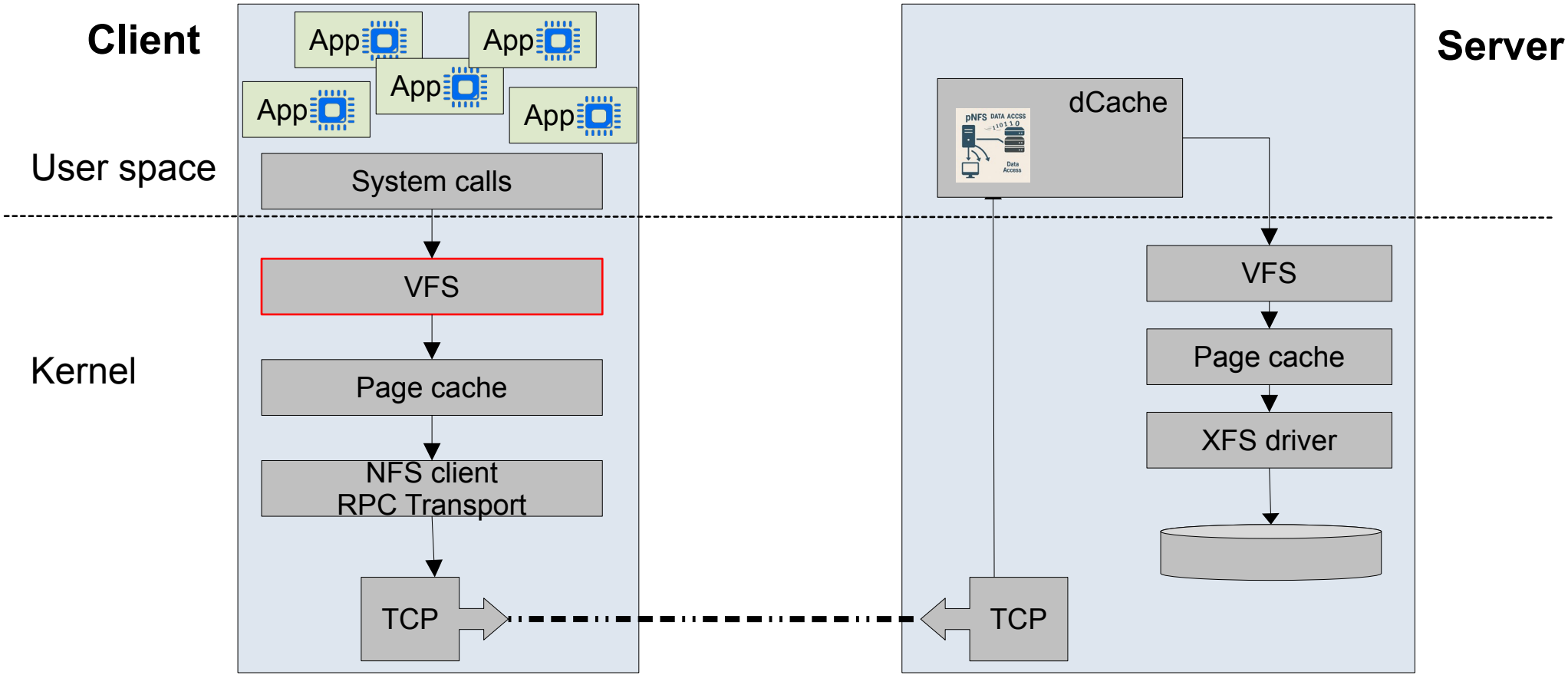
HPC: Many Cores, Single File



NFS Access Details

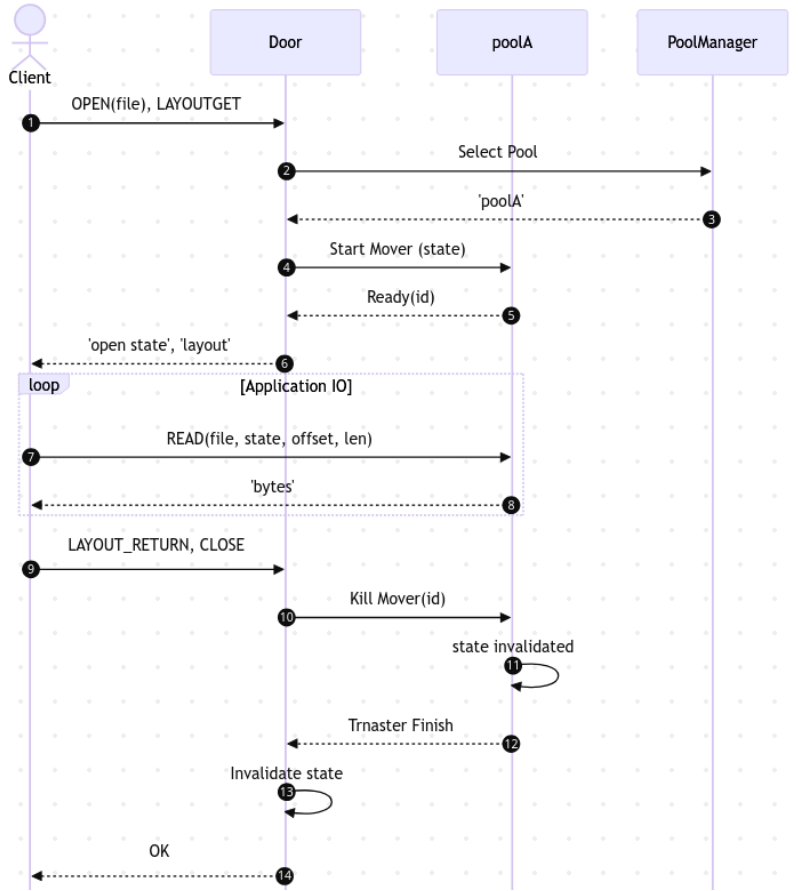


NFS Access Details



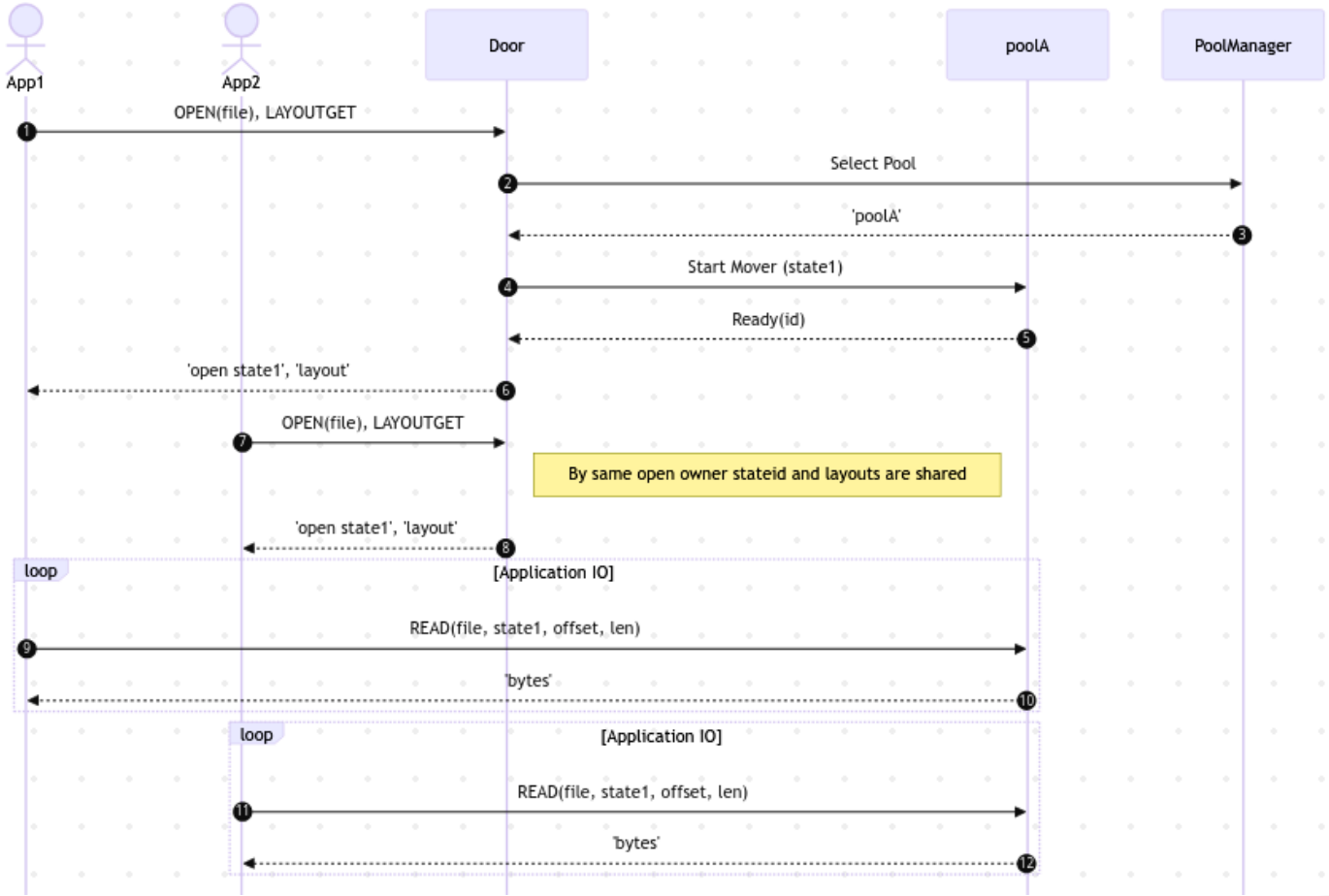
NFSv4.1/pNFS Door Flow

- Access validation on OPEN
- open-state used to identify mover
- Read/Write happens with associated state ID
- LAYOUTRETURN stops mover
- Close invalidates open-state.



Two and More Clients

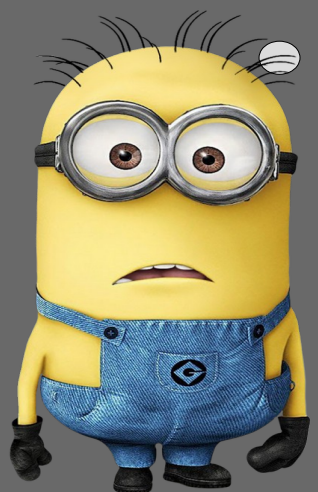
- Multiple opens on the same file are ‘merged’
- Single CLOSE closes the file

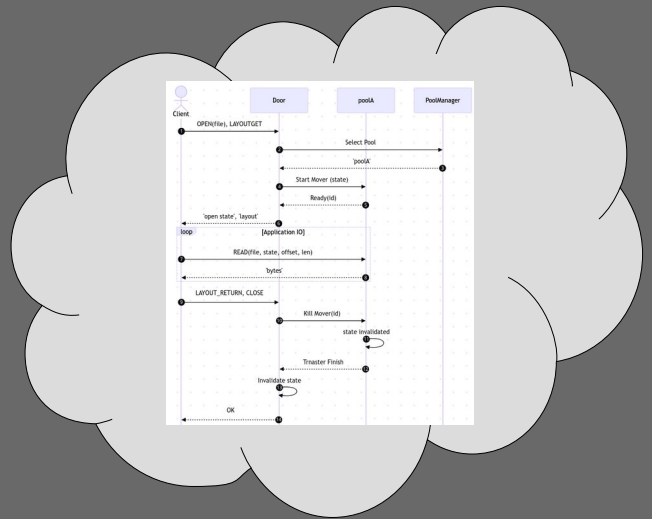


The Joy of Python World

```python
def read_frame_from_file(frame_id: int, data_file: str):
    """Read specified frame id from the specified file"""
    with h5py.File(data_file, 'r') as h5in:
        return h5in['/INSTRUMENT/.../image/data'][frame_id]


for i in range(events):
    f = read_frame_from_file(i, ‘file.hdf5’)
```




Client
Door
poolA
PoolManager
OPEN(file), LAYOUTGET
Select Pool
'poolA'
Start Mover (state)
Ready(id)
'open state', 'layout'
loop
[Application IO]
READ(file, state, offset, len)
'bytes'
LAYOUT_RETURN, CLOSE
Kill Mover(id)
state invalidated
Tmaster Finish
Invalidate state
OK

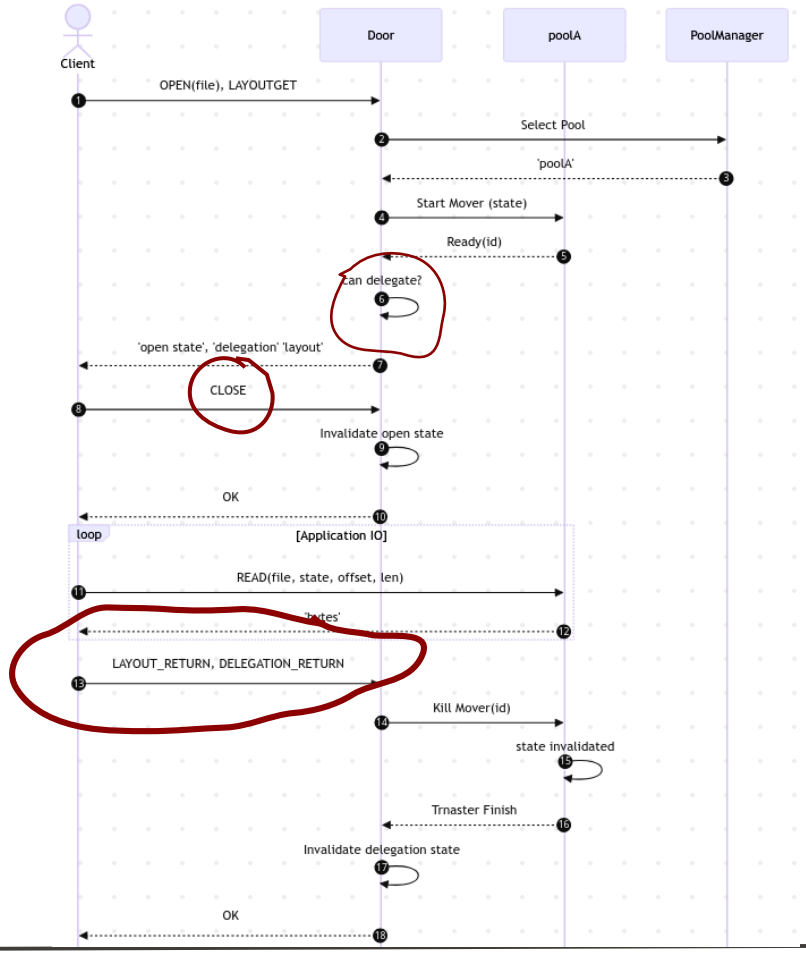

“All CPU issues eventually turns into I/O problems.”
Derived from Amdahl's Law

NFS Delegations

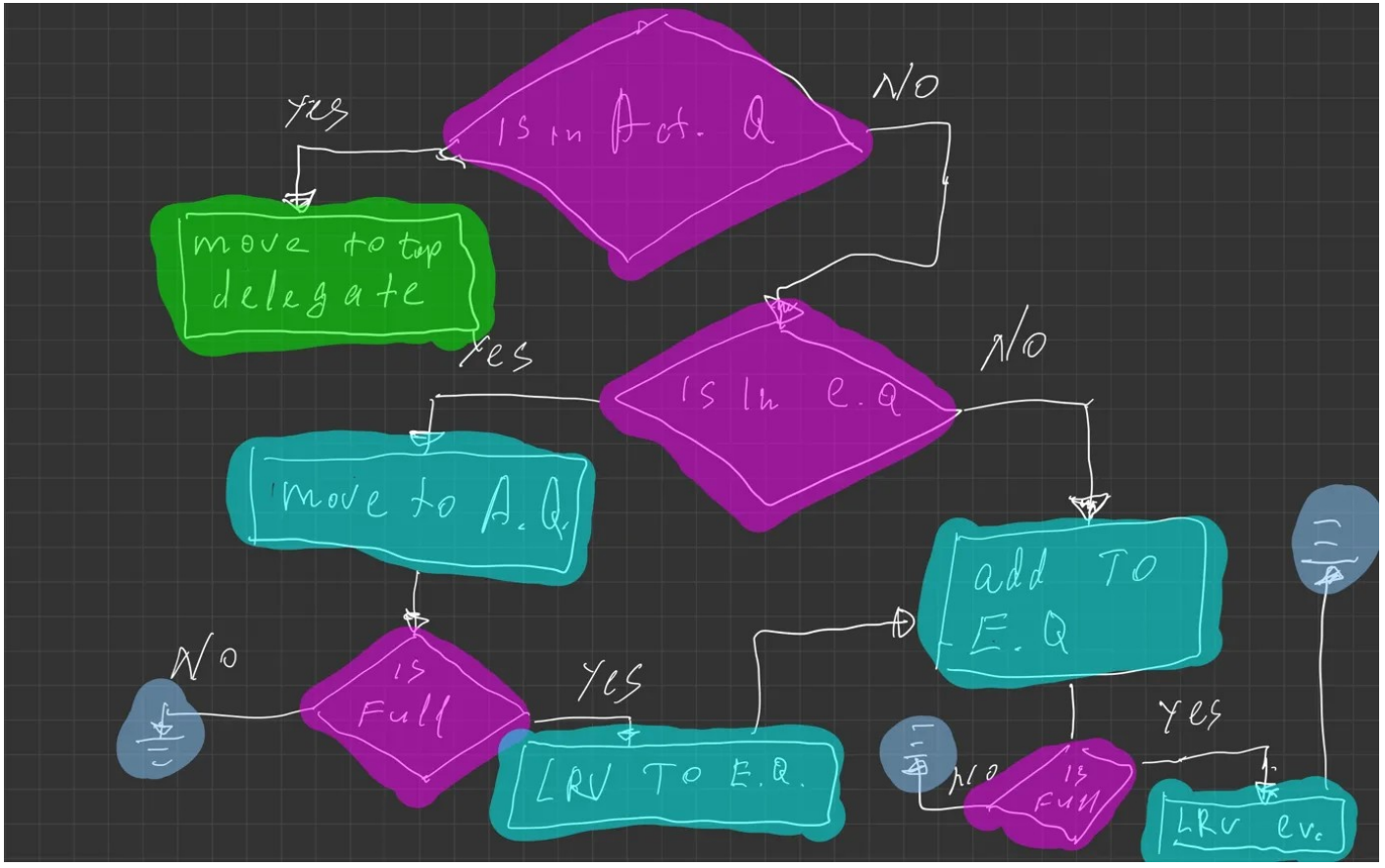
- “A delegation is passed from the server to the client, specifying the object of the delegation and the type of delegation”
- “When a file is being OPENed, the server may delegate further handling of opens and closes for that file to the opening client”
- “When a client has an OPEN delegation, it does not need to send OPENs or CLOSEs to the server”

NFS Delegations

- Repetitive OPENs trigger delegation
- Mover is independent of open-state
- Read/Write happens with layout state ID
- DELEGATION return independent of CLOSE
- Client decided when to return DELEGATION (or re-call)



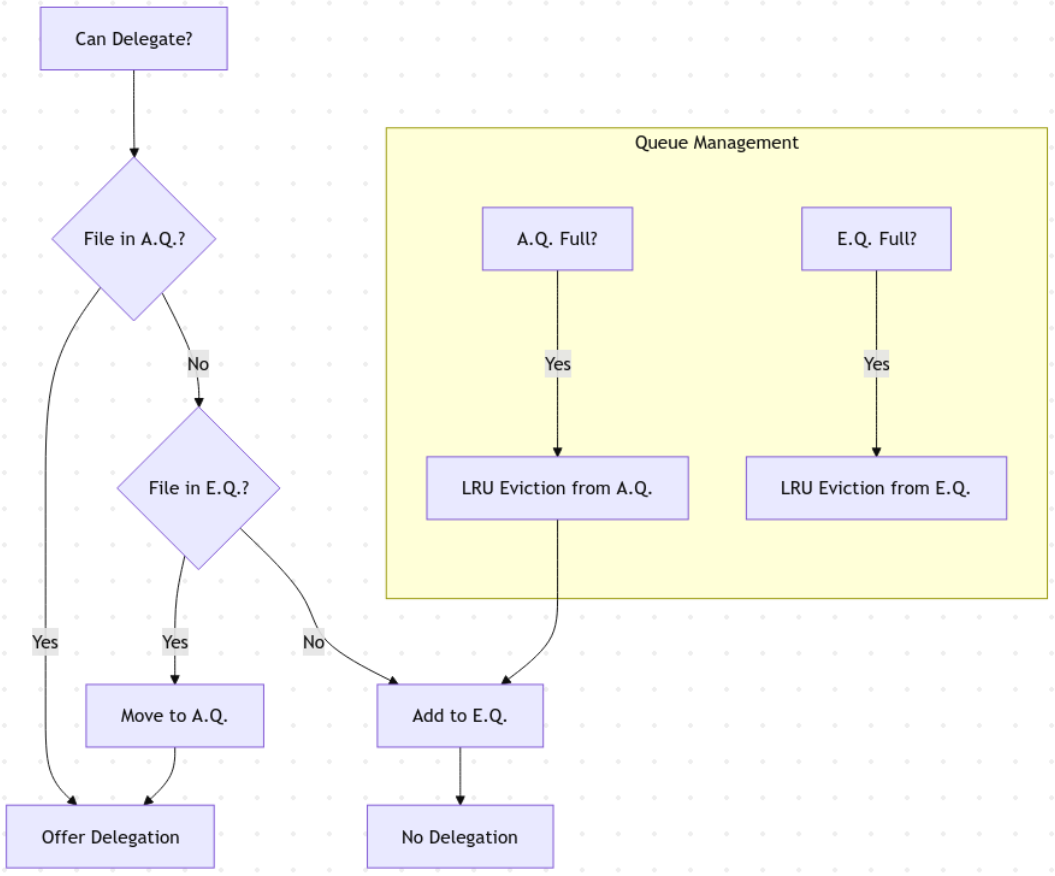
Adaptive Delegation Heuristic (current)



Adaptive Delegation Heuristic

Per NFS Client

1. When a file is accessed for the first time, it's added to the Eviction Queue
2. If a file in the Eviction Queue is accessed again, it is moved to the Active Queue
3. If a file in the Active Queue is accessed again, the system recommends delegation
4. If a file in the Active Queue is not accessed for a specified idle time, it is moved to the Eviction Queue
5. When active queue reaches capacity, the least recently used (LRU) file is pushed into eviction queue
6. When evicted from active queue entry is not accessed for a specified idle time, entry evicted
7. When eviction queue reaches capacity, the least recently used (LRU) file is evicted

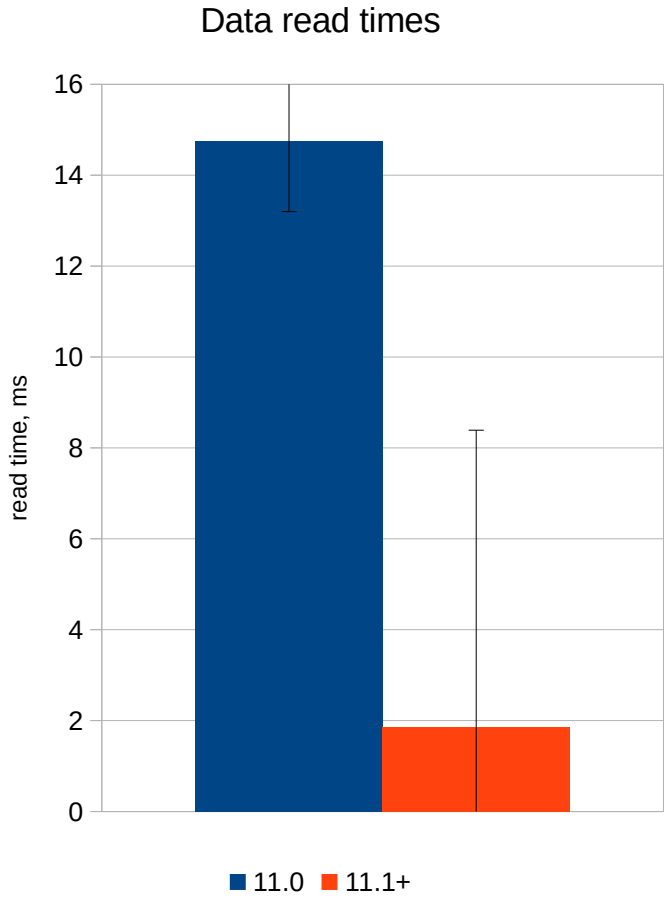


- **dCache 11.0.x**

```
$ ~/xfel_loop_test.py
    Processing time: 147.66 s,
            per frame time: 14.75 ±1.55 ms.
```

- **dCache 11.1+.x**

```
 $ ~/xfel_loop_test.py
    Processing time: 18.65 s,
            per frame time: 1.86 ±6.53 ms.
```



Summary

- dCache team keep improving NFS access to provide best possible POSIX experience
- dCache works best for HTC workload
  - Many cores accessing different files
- If many jobs access same file, group them on a single (handful) node(s)
- Two (and more) concurrent transfers share the mover (and the billing record)
- NFS Delegations: server controlled client cache
  - Movers stay alive after transfer
  - Billings **transferTime** and **transferSize** doesn’t report single transfer

Thank you!

Questions?

Amdahl's Law

“the overall performance improvement gained by optimizing a single part of a system is limited by the fraction of time that the improved part is actually used”