Distributed Asynchronous Object Storage (DAOS)

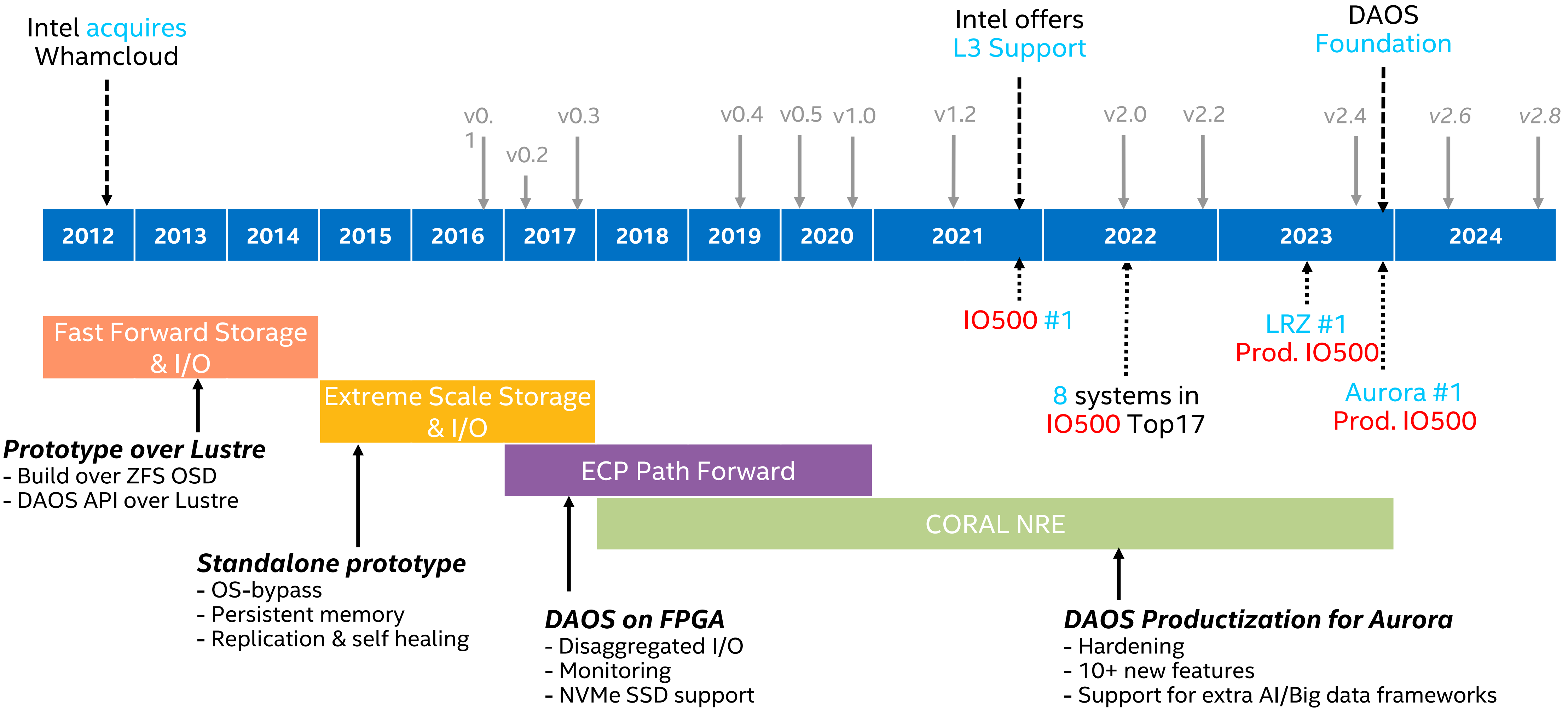
Using the DAOS Storage APIs with Weather and Climate Applications

Parallel IO NHR Workshop, DKRZ Hamburg, 08-May-2024

Michael Hennecke

intel.

DAOS Development History



DAOS Foundation – established 08-Nov-2023

Intel establishes open-source foundation

THE LINUX FOUNDATION

DAOS Foundation

Governance, Roadmap

Community Releases

Technical Working Groups

DAOS User Group

Hewlett Packard Enterprise
ENAKTA LABS
and other solution integrators

Platform integration
Feature development
Bug fixing
Testing
L1-L2 support services

Google Cloud
and other CSPs

Cloud integration
Feature development
Bug fixing
Testing

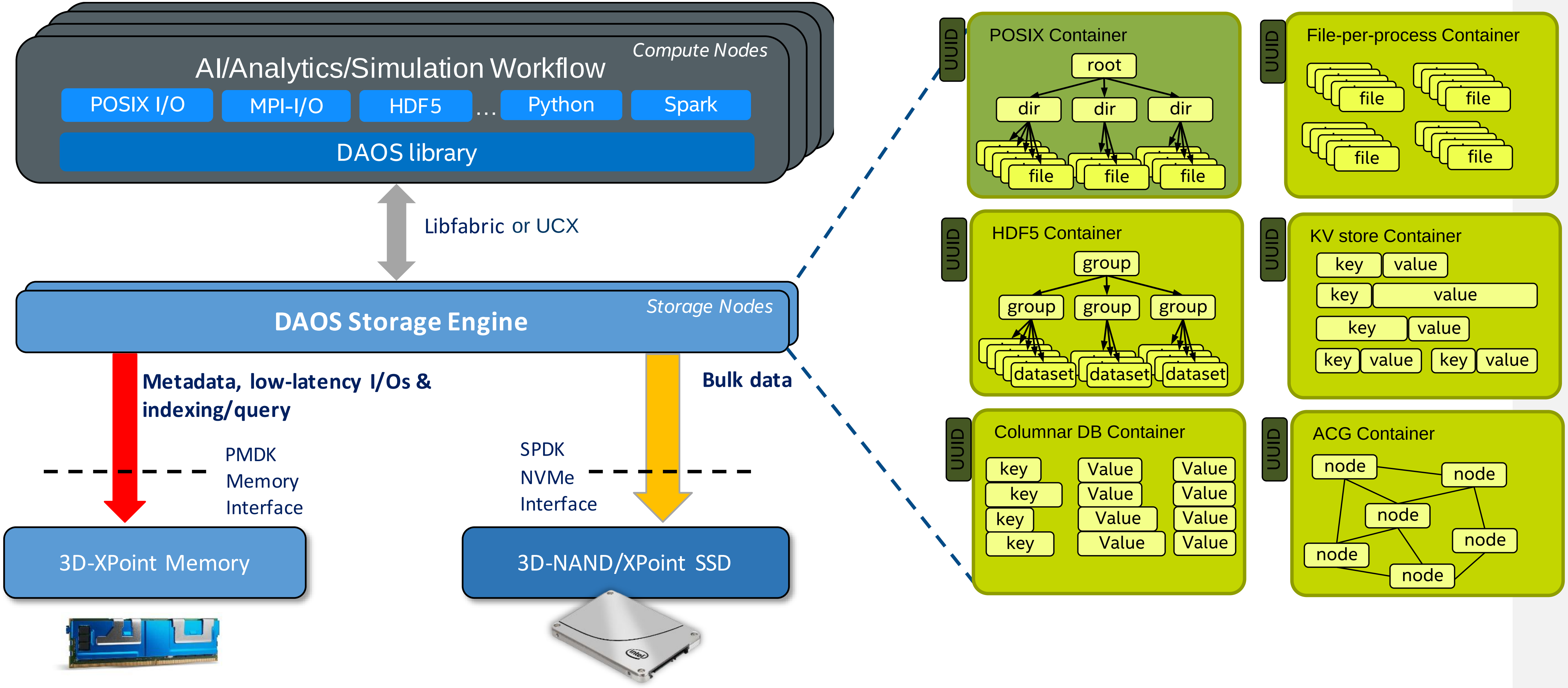
intel

Intel HW enablement & optimizations
Feature Development & Pathfinding
Reference design & system architecture
Bug fixing
Testing
NRE services
L3 support services

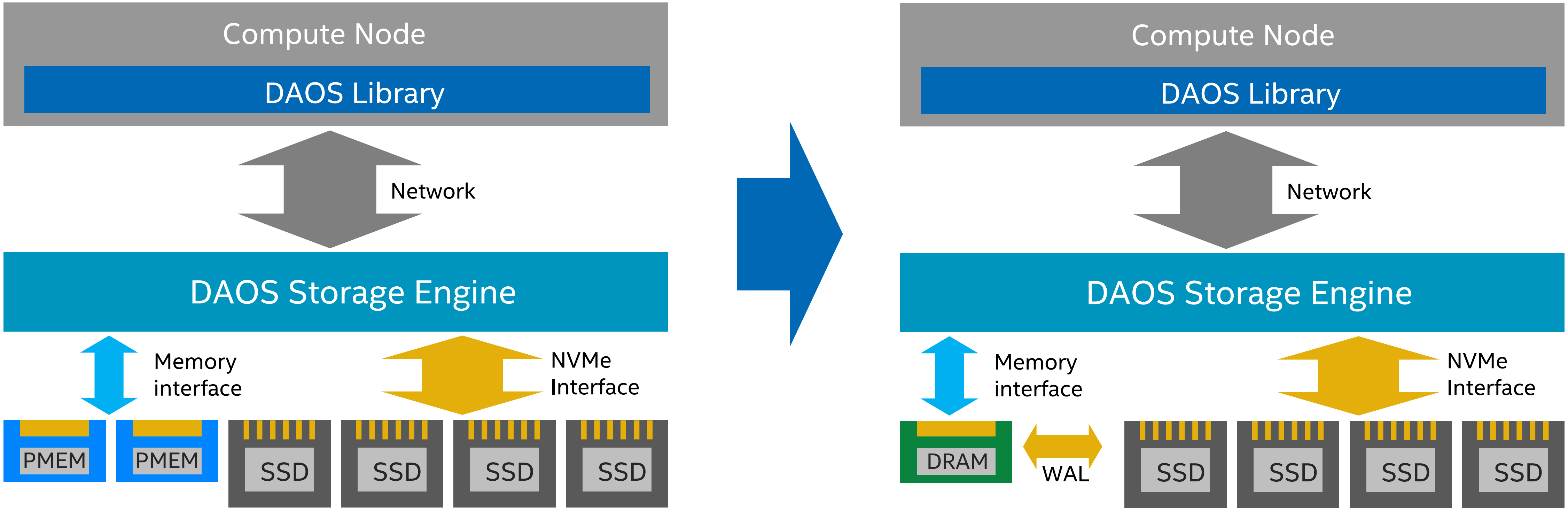
Argonne NATIONAL LABORATORY
and other customers

AI/HPC workflow integration
Feature development
NRE Investment

DAOS Exascale Storage Architecture (PMem based)



DAOS Architecture Evolution (with → without PMem)



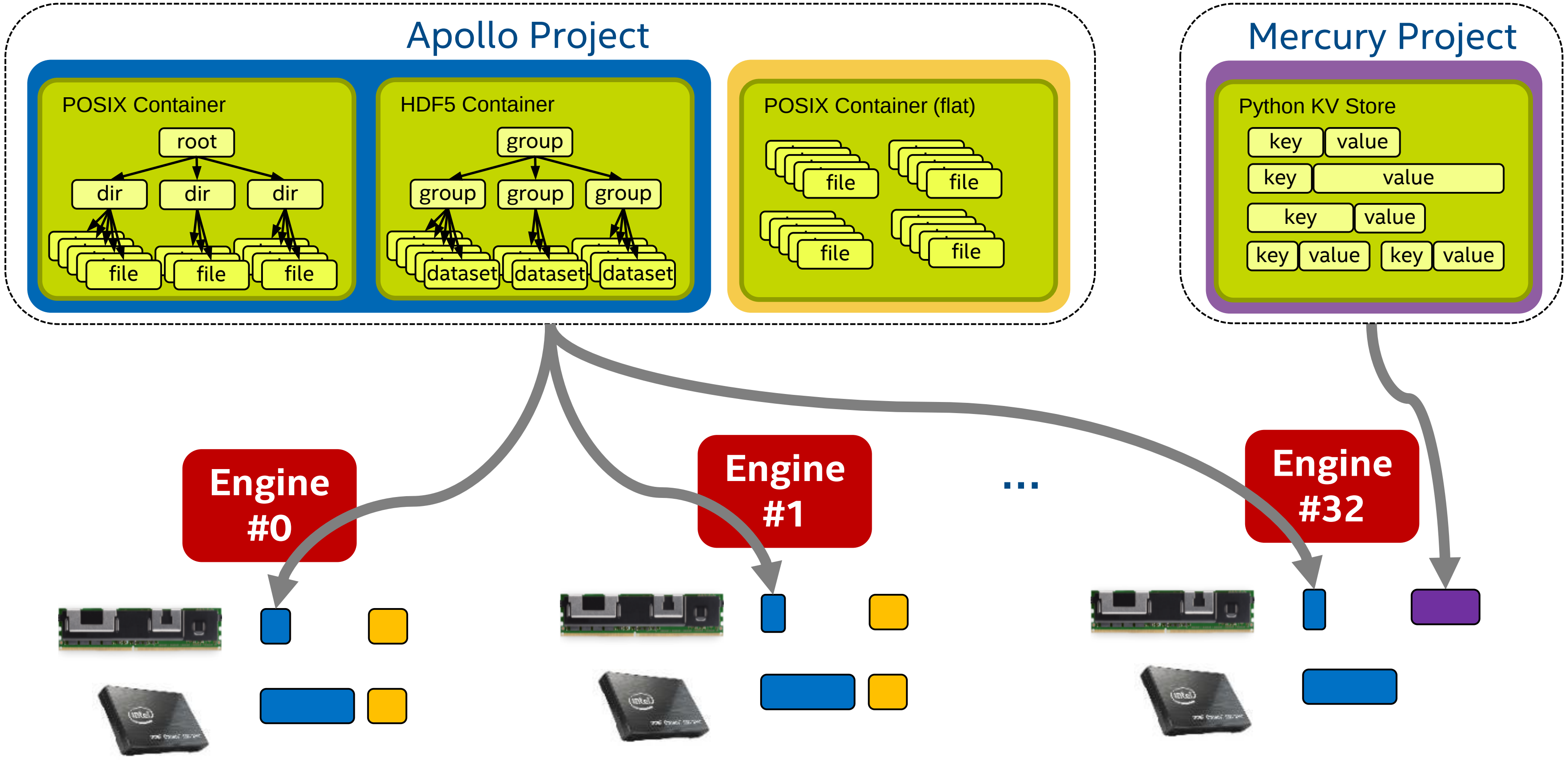
With Persistent Memory

Without Persistent Memory

DAOS Data Model: Pools

Applications
Compute Nodes (CN)

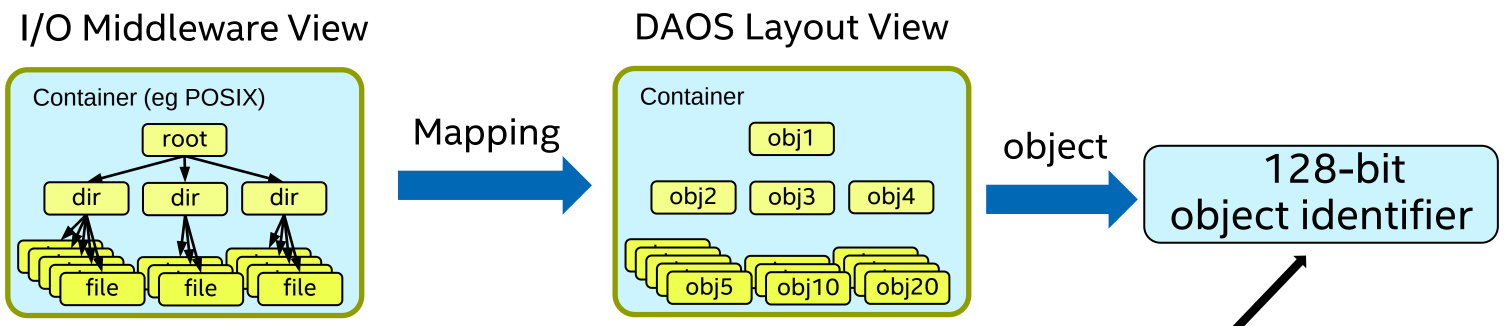
DAOS System
DAOS Nodes (DN)



dmg pool create

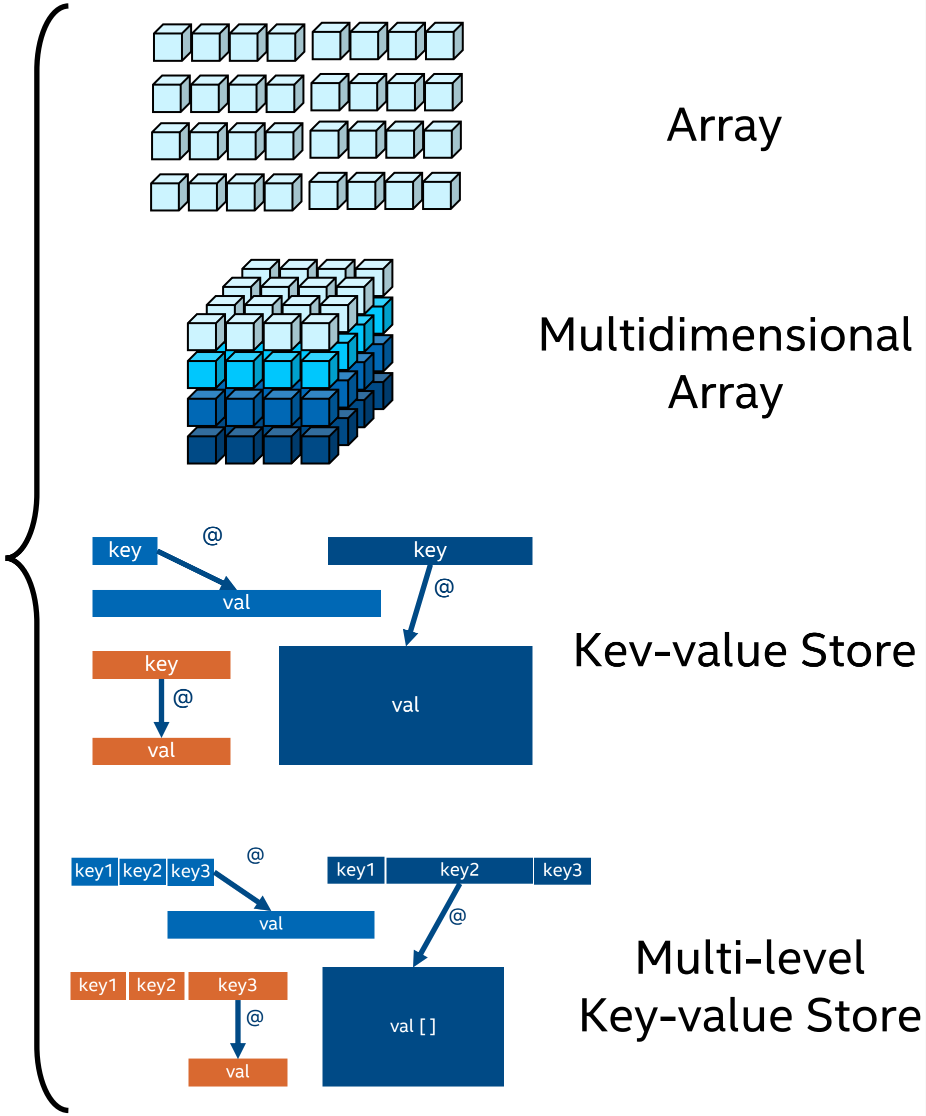
Pool_1 (blue)	--group apollo@	--size=10P # 6% SCM	# all ranks	can set/change pool properties, e.g.: -P space_rb:2,ec_cell_size:131072
Pool_2 (yellow)	--group apollo@	--size=1P -t 50,50	--ranks=0,1	
Pool_3 (purple)	--group mercury@	--size=2T -t 100,0	--ranks=32	

DAOS Object Models

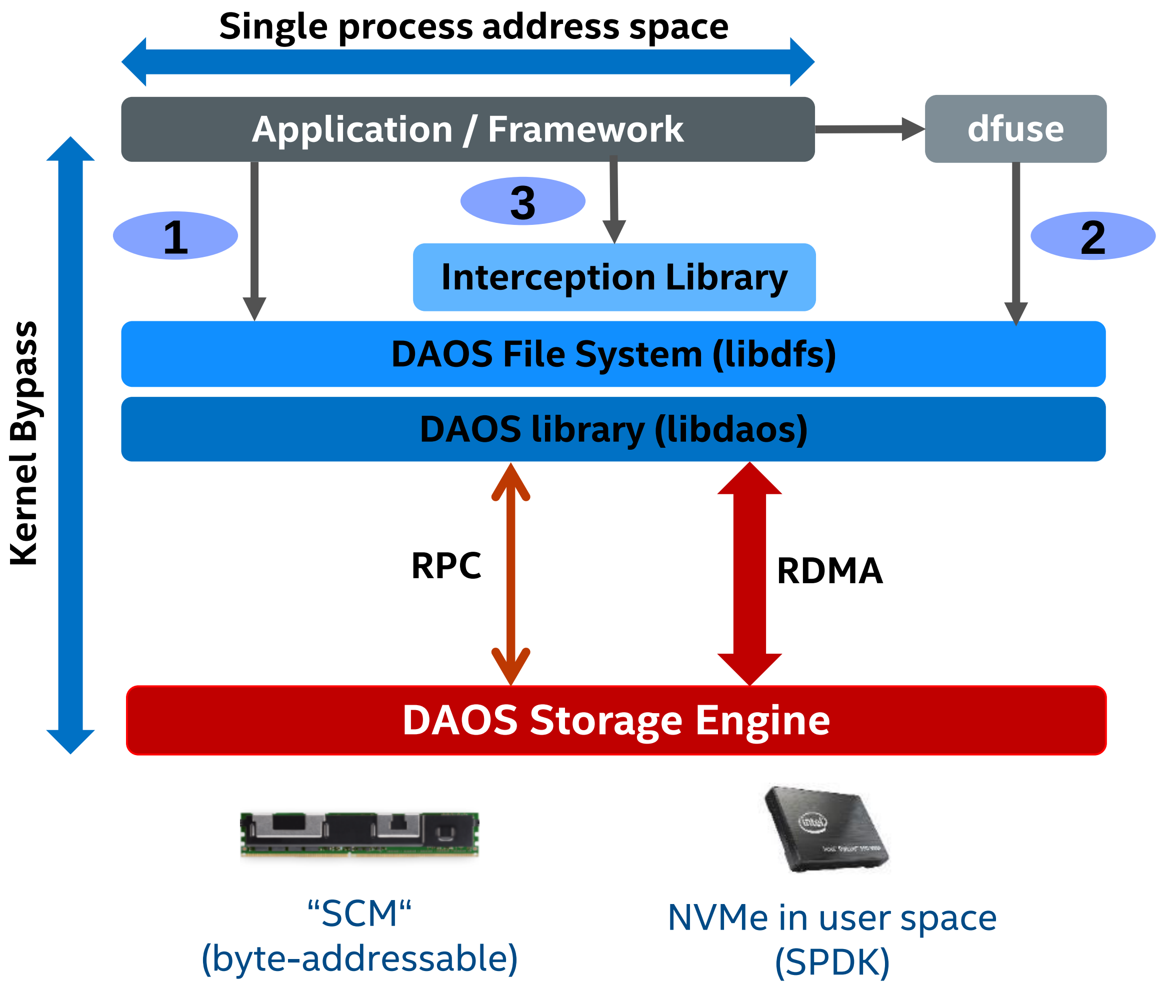


The 128-bit DAOS OID includes:

- Lower 96-bit: user's object ID
- Upper 32-bit used by DAOS to specify:
  - DAOS object type: KV, array, multi-level KV
  - DAOS object class: data distribution and protection



POSIX I/O Support



- DAOS File System (**libdfs**) 1
  - POSIX namespace in a container
  - Application/framework can link directly with libdfs.so
  - Full OS bypass, asynchronous I/O & list I/O support
- FUSE Daemon (**dfuse**) 2
  - Transparent access to DAOS through **dfuse** mount
  - Involves system calls
  - Caching & read-ahead option for AI APPs
- I/O interception library (**libioil, libpil4dfs**) 3
  - Combined with dfuse
  - OS bypass for read/write operations
  - LD_PRELOAD=/usr/lib64/lib{ioil,pil4dfs}.so
  - Supports static binaries

PyDAOS Examples

```
from pydaos import (DCont, DDict, DObjNotFound)

daos_cont = DCont("mypool1", "mycont1", None)

# get or create a dictionary object
daos_dict = None
try:
        daos_dict = daos_cont.get("my_dict1")
except DObjNotFound:
        daos_dict = daos_cont.dict("my_dict1")

# insert a key/value pair
key   = "dog"
value = "perro"
daos_dict.put(key, value)

# read the value for a key:
try:
        value = str(daos_dict[key])
except KeyError:
        print("key not found")

# delete a key/value pair
daos_dict.pop(key)
```

```
# iterate the whole dictionary:
for key in daos_dict:
        print("key=" + key + "  value=" +
              str(daos_dict[key]))

# bulk insertion:
python_dict = {}
python_dict[key0] = value0
python_dict[key1] = value1
python_dict[key2] = value2
...
daos_dict.bput(python_dict)

# bulk read
python_dict = {}
python_dict[key0] = None
python_dict[key1] = None
python_dict[key2] = None
...
daos_dict.bget(python_dict)

# read the whole dictionary with dump()
python_dict = daos_dict.dump()

# get the total number of keys
print("dict has " + str(len(daos_dict)) + " keys")
```

DAOS and IO500

DAOS Deployments at ALCF (Aurora) and LRZ (SNG2)

Compute Nodes:
2x Intel SPR+**HBM**, **6x** Intel Xe "PVC" GPUs, **8x** HPE Slingshot

Compute Nodes:
2x Intel SPR, **4x** Intel Xe "PVC" GPUs, **2x** NVIDIA HDR

1024 DAOS Servers (Intel M50CYP):

- 2x Xeon 5320 26core 2.2GHz CPUs
- 16x 32GB DDR4 DRAM
- 16x **512GB** Intel Optane 200 PMem
- **16x** Samsung PM1733 **15.36TB** NVMe (gen4)
- 2x HPE Slingshot (200Gbps)

➔ 16k NVMe (250PB), 16k PMem (8PB), 2k engines

42 DAOS Servers (Lenovo SR630v2):

- 2x Xeon 8352Y 32core 2.2GHz CPUs
- 16x 32GB DDR4 DRAM
- 16x **128GB** Intel Optane 200 PMem
- **8x** Intel P5500 **3.84TB** NVMe (gen4)
- 2x NVIDIA HDR InfiniBand (200Gbps)

➔ 336 NVMe (1.3PB), 672 PMem (84TB), 84 engines

IO500-SC23 : Production List

Production SC23 List

Customize | Download

IO500

Production | 10 Node Production | Research | 10 Node Research | Full | Historical

Ranking of production system submissions. This is a subset of the Full List of submissions, showing only one highest-scoring result per storage system. Submitters who want a submission that is currently on the Research List to be on the Production List should contact the IO500 Steering Committee.

# ↑	BOF	INSTITUTION	SYSTEM	STORAGE VENDOR	FILE SYSTEM TYPE	CLIENT NODES	TOTAL CLIENT PROC.	SCORE ↑	BW (GIB/S)	MD (KIOP/S)	REPRO.	
1	SC23	Argonne National Laboratory	Aurora	Intel	DAOS	300	62,400	32,165.93	10,066.09	102,785.41	✓	642 servers
2	SC23	LRZ	SuperMUC-NG-Phase2-EC	Lenovo	DAOS	90	6,480	2,508.85	742.90	8,472.60	✓	42 servers
3	SC23	King Abdullah University of Science and Technology	Shaheen III	HPE	Lustre	2,080	16,640	797.04	709.52	895.35	✓	120 servers
4	ISC23	EuroHPC-CINECA	Leonardo	DDN	EXAScaler	2,000	16,000	648.96	807.12	521.79	✓	42 servers?
5	SC23	Memorial Sloan Kettering Cancer Center	IRIS	WekaIO	WekaIO	36	4,248	308.94	104.79	910.80	✓	22 servers
6	ISC22	China Telecom Research Institute	CTPAI	CTCLOUD	DAOS	10	200	187.84	25.29	1,395.01	-	
7	ISC23	Imperial College London	Imperial - hx cluster	Lenovo	Spectrum scale	32	512	119.56	44.63	320.31	✓	16 servers
8	SC23	Japan Agency for Marine-Earth Science and Technology	Earth Simulator 4	DDN	EXAScaler	10	320	101.88	48.19	215.38	✓	20 OSS, 4 MDS
9	SC23	Center for Research Informatics at University of Chicago	Randi	IBM	Spectrum Scale	10	160	60.88	31.05	119.36	✓	6 servers?
10	SC23	Poznan Supercomputing and Networking Center	Altair	Huawei/xFusion	Lustre	14	392	53.70	8.84	326.39	✓	32 servers

See the full IO500-SC23 Production list …

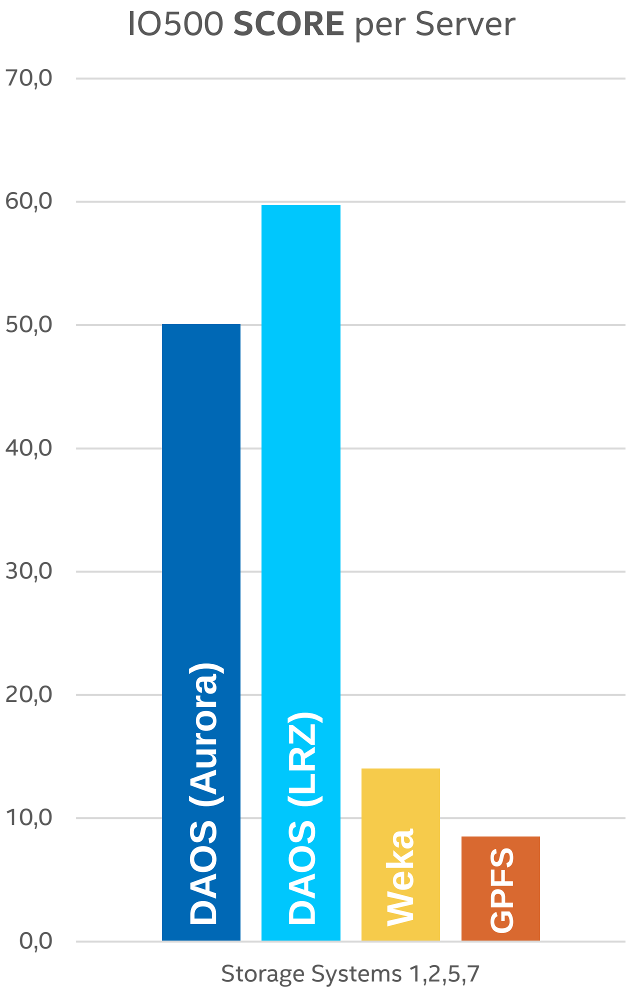
IO500-SC23 Performance per Server (Production List)

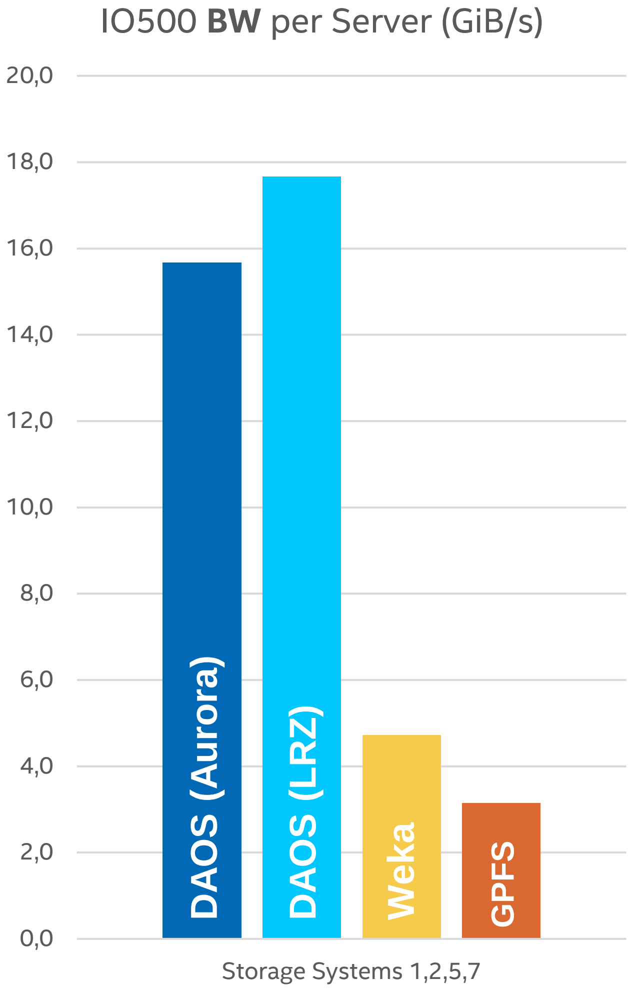
IO500-SC23 Production List

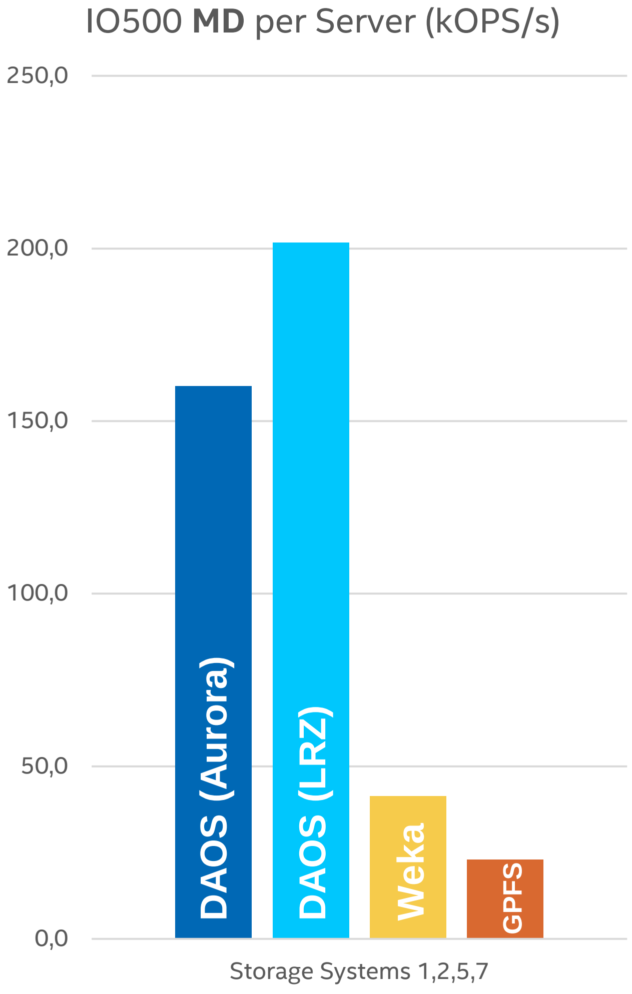
	#1 ANL (DAOS 2.5.0) 642 server @ 16 NVMe		#2 LRZ (DAOS 2.4.0) 42 server @ 8 NVMe		#5 MSKCC (WekaFS 4.2.4) 22 server @ 8 NVMe		#7 ICL (GPFS-ECE 5.1.7.1) 14 server @ 10 NVMe	
	absolute	**per-server**	absolute	**per-server**	absolute	**per-server**	absolute	**per-server**
MDTEST	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s
Easy Write	60985	**95,0**	6324	**150,6**	662	**30,1**	246	**17,6**
Easy Stat	225295	**350,9**	29403	**700,1**	9852	**447,8**	723	**51,6**
Easy Delete	57648	**89,8**	3442	**82,0**	882	**40,1**	177	**12,6**
Hard Write	33827	**52,7**	2644	**63,0**	120	**5,5**	52	**3,7**
Hard Read	141467	**220,4**	17023	**405,3**	3622	**164,6**	574	**41,0**
Hard Stat	230086	**358,4**	23242	**553,4**	8056	**366,2**	684	**48,9**
Hard Delete	62196	**96,9**	3112	**74,1**	89	**4,0**	45	**3,2**
IOR	GiB/s	GiB/s	GiB/s	GiB/s	GiB/s	GiB/s	GiB/s	GiB/s
Easy Write	20693	**32,2**	896	**21,3**	174	**7,9**	131	**9,4**
Easy Read	12122	**18,9**	1872	**44,6**	327	**14,9**	137	**9,8**
Hard Write	4216	**6,6**	252	**6,0**	44	**2,0**	7	**0,5**
Hard Read	9706	**15,1**	718	**17,1**	47	**2,1**	29	**2,1**
FIND	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s	kIOP/s
Find	229672	**357,7**	12733	**303,2**	262	**11,9**	3709	**264,9**
SCORE								
IO500 Score	32165	**50,1**	2508	**59,7**	308	**14,0**	119	**8,5**
IO500 BW	10066	**15,7**	742	**17,7**	104	**4,7**	44	**3,1**
IO500 MD	102785	**160,1**	8472	**201,7**	910	**41,4**	320	**22,9**

#3 is HPE-Lustre, #4 is DDN-Lustre -- hard to compare as those are HW-controllers, and hybrid (MDS=NVMe, OSS=NVMe+HDD)

IO500-SC23 Performance per Server (Production List)







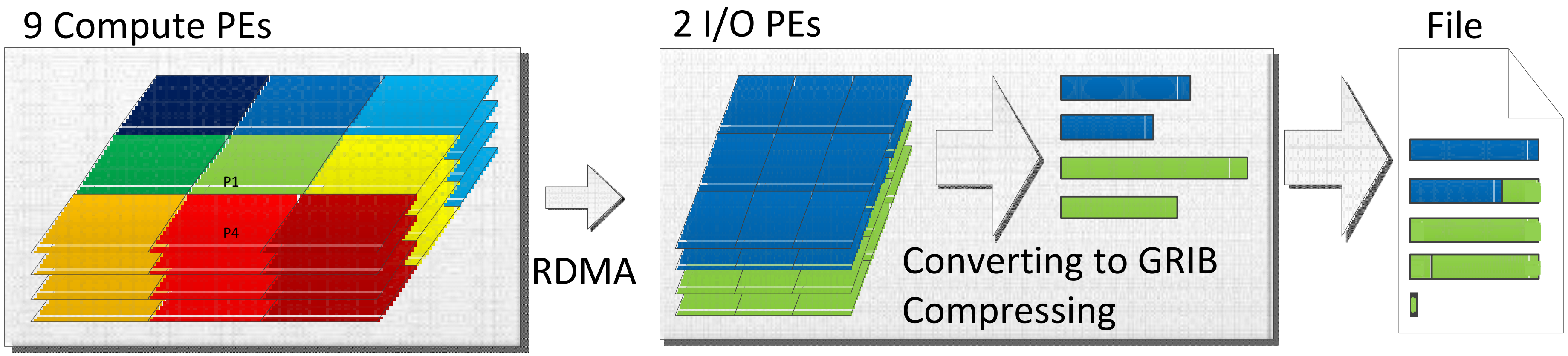
#3 is HPE-Lustre, #4 is DDN-Lustre -- hard to compare as those are HW-controllers, and hybrid (MDS=NVMe, OSS=NVMe+HDD)

Collaboration with DKRZ on ICON

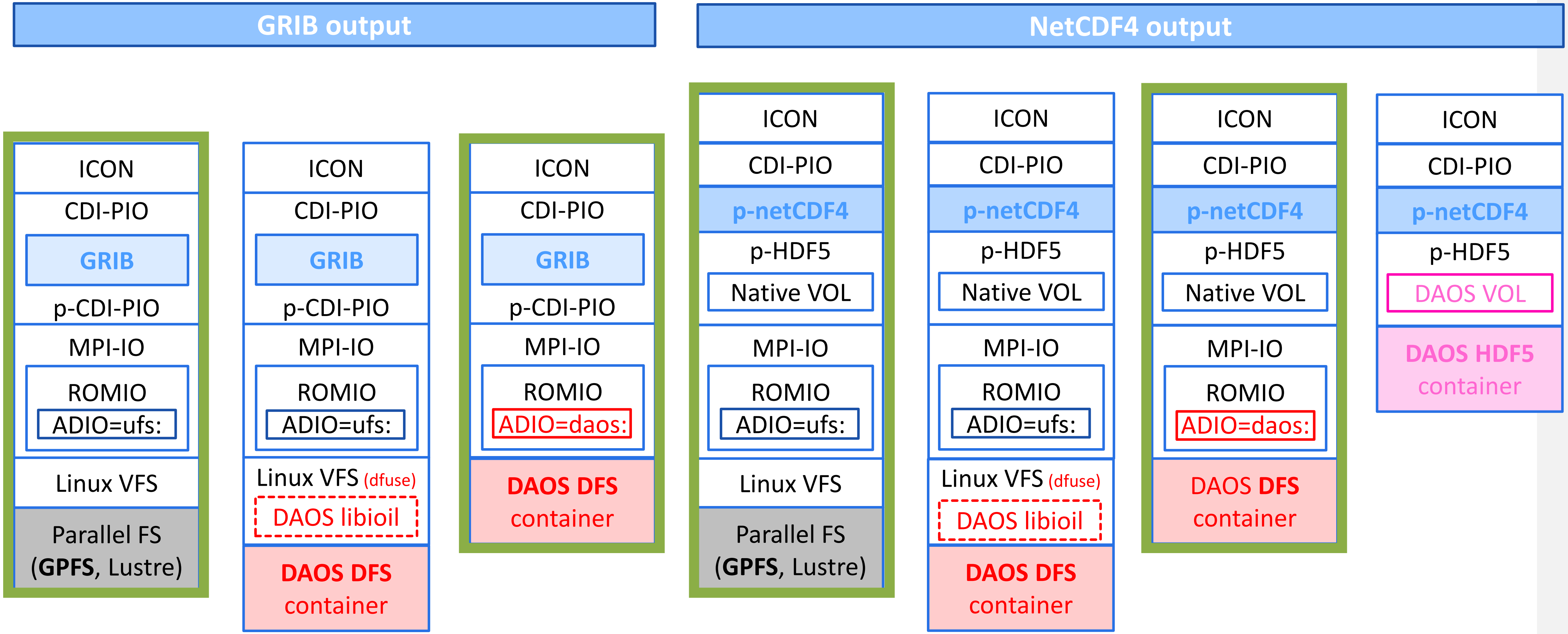
Panagiotis Adamidis, Xingran Wang, Thomas Jahns (DKRZ)
Michael Hennecke (Intel)
Christoph Pospiech (Lenovo)

Climate Simulation I/O – State of the Practice

- Writing timestep data to storage is offloaded to **separate "I/O PEs"** (MPI tasks)
- **Asynchronous data copy** to I/O tasks, to overlap I/O with computation
- I/O processes **transpose** the received datasets
  - Domain decomposition of Compute-PE memory does not match file layout…
- Some data formats (e.g., GRIB) also include **compression**



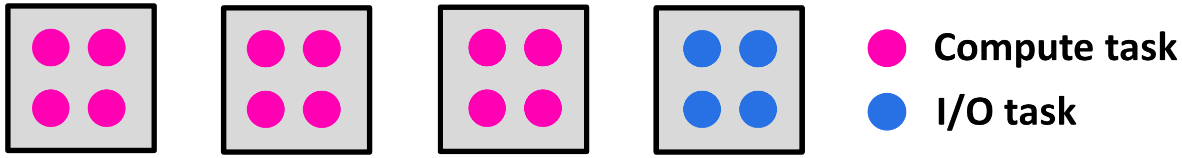
The Parallel I/O Software Stack for ICON



Mapping I/O Tasks to Nodes – LAST vs. BALANCED

CDI-PIO „**LAST**“ Mode:

I/O aggregator tasks are the last MPI ranks in the job, and get allocated on the last node(s):



Used in production, works well with task allocation of simulation codes like ICON.

PRO:

Simple task-to-node **mapping**.

Data **transposition** is fast (node-local).

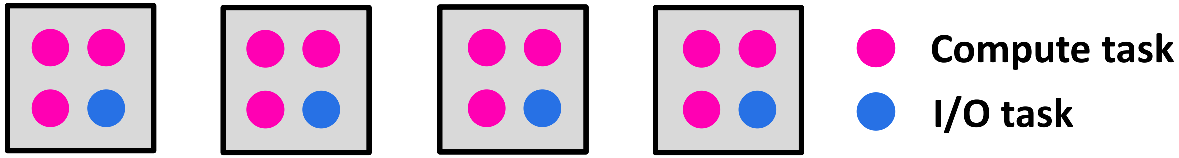
Minimal **file locking** contention.

CON:

I/O tasks' **storage bandwidth** and **memory capacity** limited to single(few) node(s).

CDI-PIO „**BALANCED**“ Mode:

One (or few) I/O aggregator task per node:



Not yet used in production, but promising...

PRO:

I/O tasks' **storage bandwidth** and **memory capacity** scales with # compute nodes.

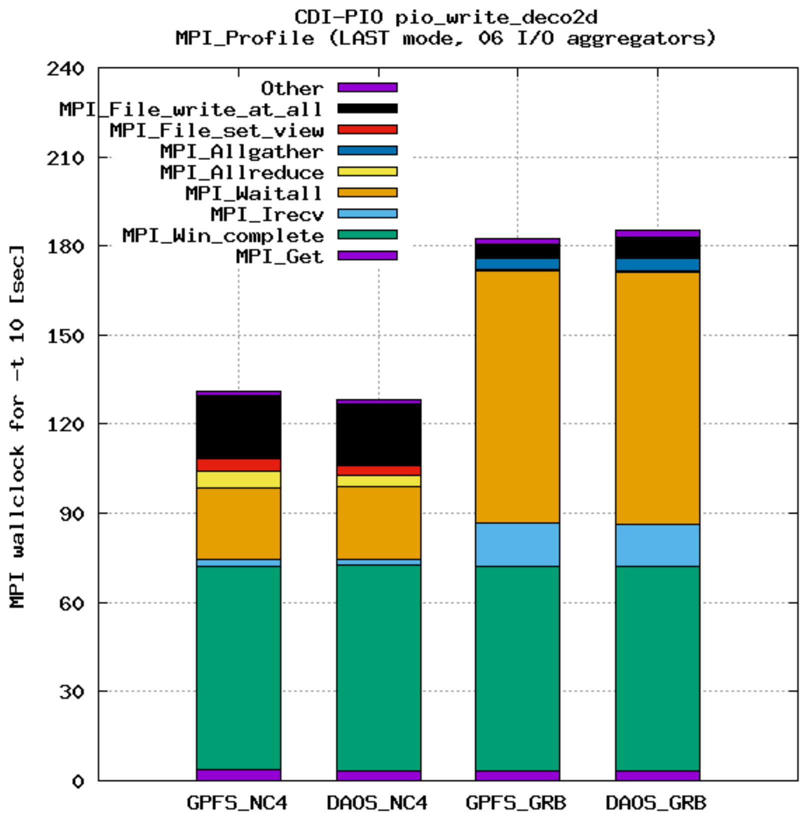
CON:

Task-to-node **mapping** is more complex.

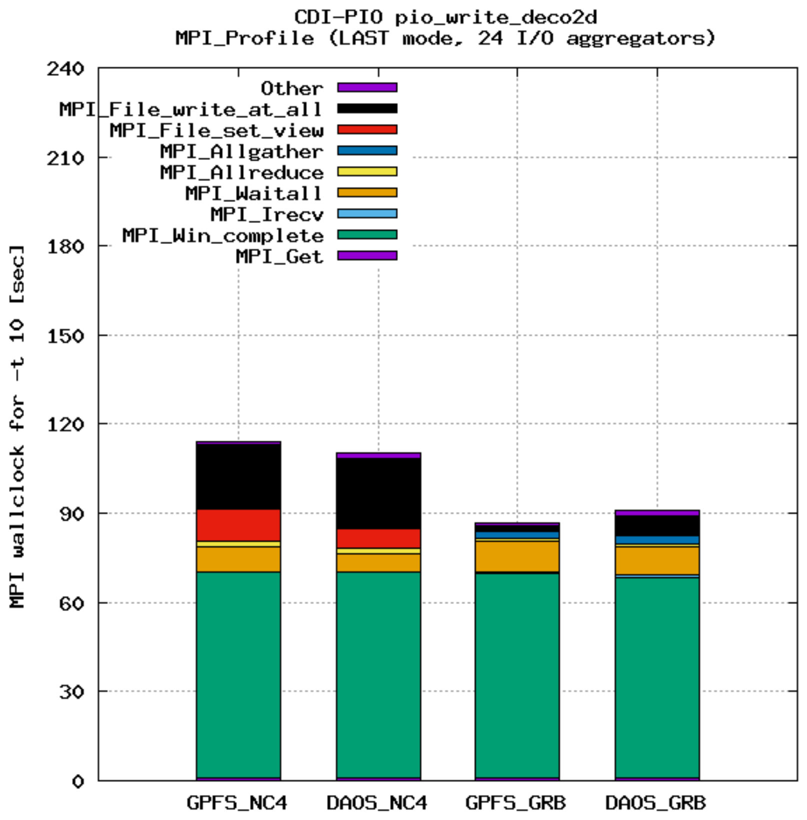
Data **transposition** is slower (inter-node).

Increased **file locking** contention.

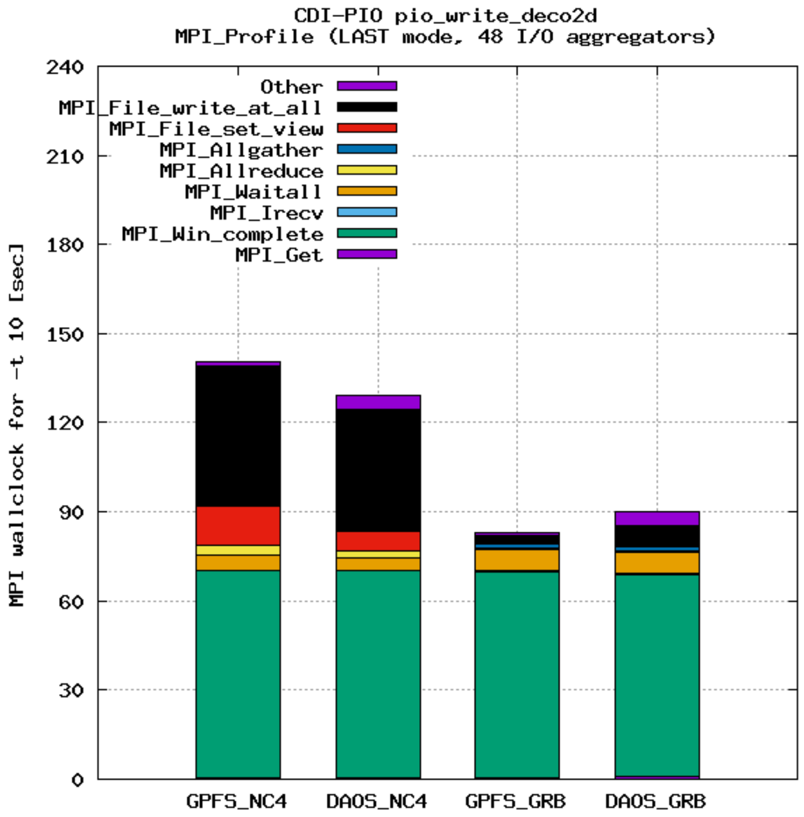
CDI-PIO `pio_write_deco2d` MPI Profiles („LAST“ Mode)



6 I/O aggregators



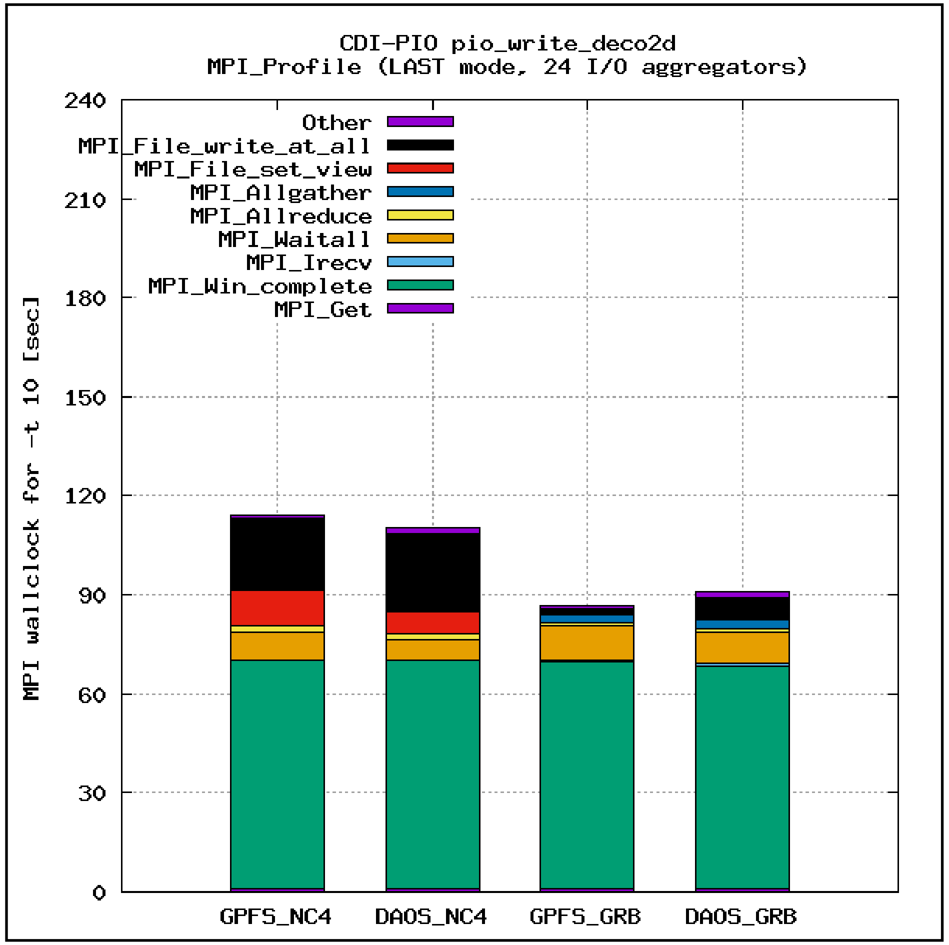
24 I/O aggregators



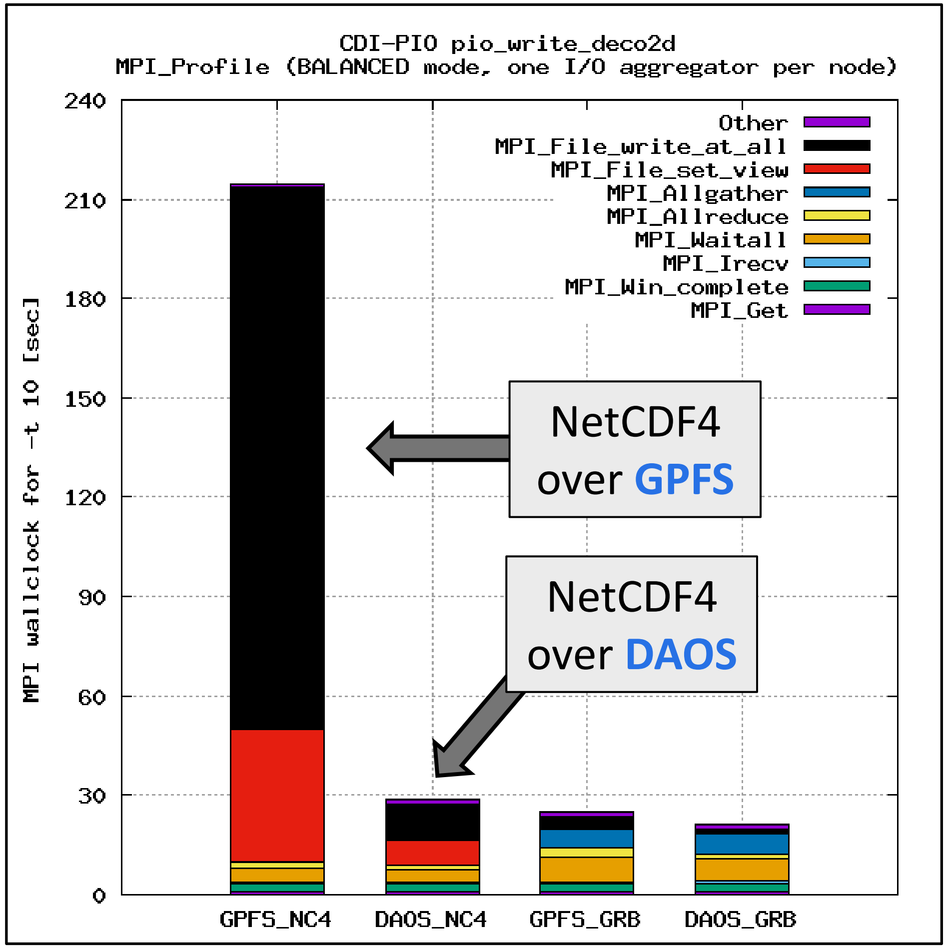
48 I/O aggregators

CDI-PIO `pio_write_deco2d` MPI Profiles

„LAST“ mode (24 I/O tasks)



„BALANCED“ mode (24 I/O tasks)



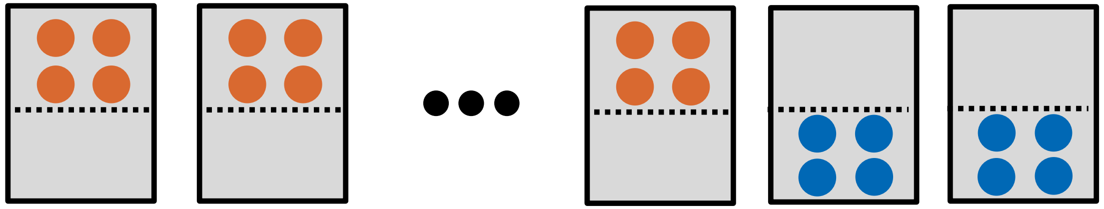
R2B9 ICON Simulation on **GPU Nodes** (not using DAOS)

High resolution **R2B9** ICON Monsoon Experiment (30 day simulation) with CDI-PIO.

Runs on **77 nodes** of FZJ‘s JUWELS Booster (heterogeneous architecture):
Compute tasks allocated on **GPUs**, and **20 I/O aggregator tasks** on **CPUs**.

CDI-PIO „**LAST**“ Mode:

I/O aggregator tasks are the last MPI ranks in the job, on dedicated nodes:



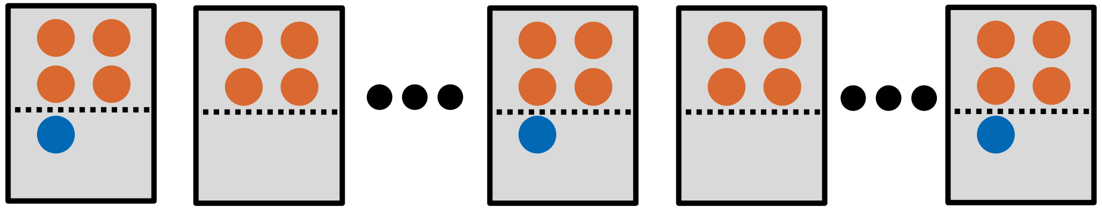
- **288 compute task on GPUs (72 nodes)**
- **20 I/O task on CPUs (on 5 nodes)**

Runtime:
20893 sec

CDI-PIO „**BALANCED**“ Mode:

Adjacent I/O aggregator tasks have the same distance

19687sec/20893sec = 94.2%
Best case: 288/308 = 93.5%

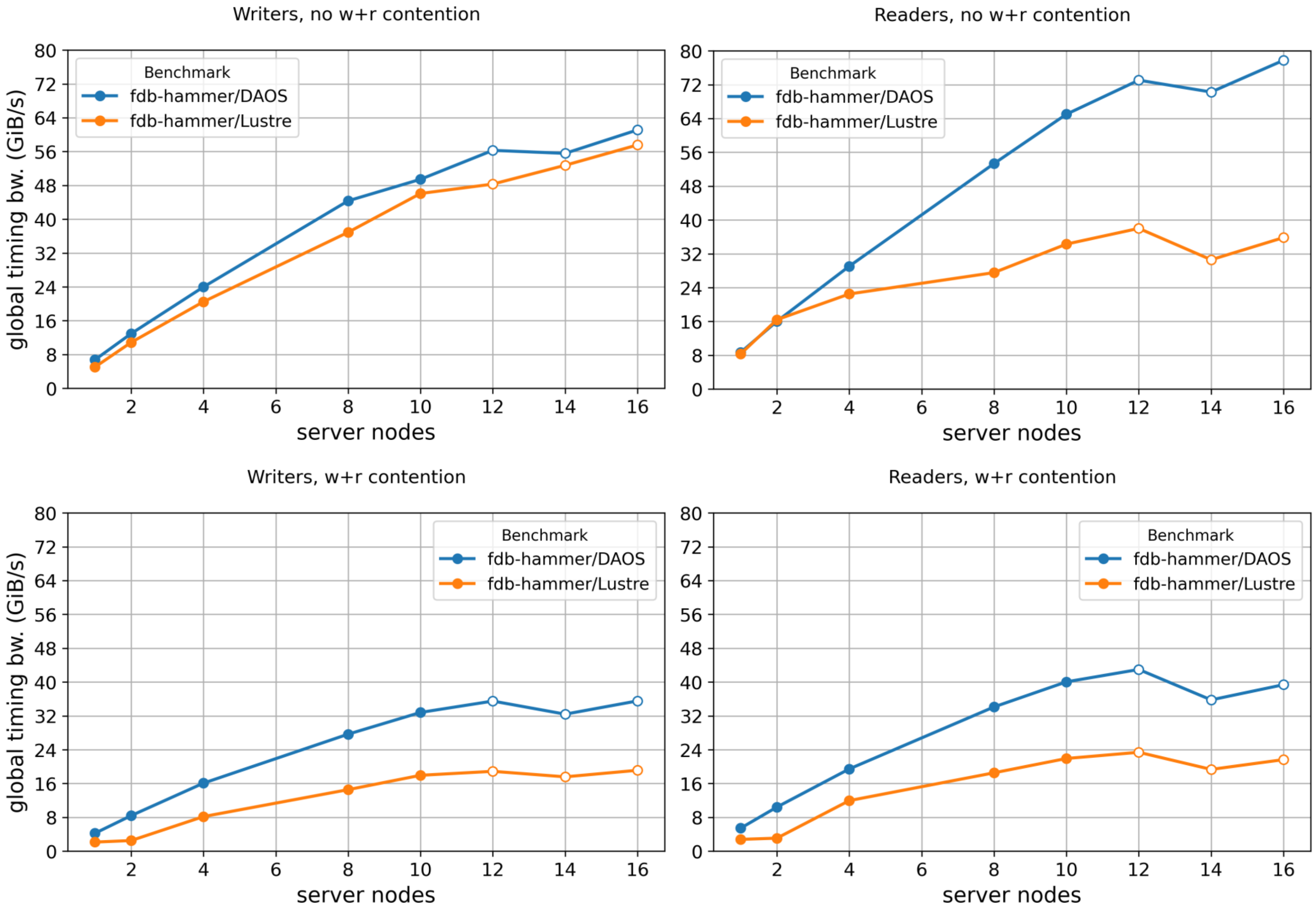


- **308 compute task on GPUs (77 nodes)**
- **20 I/O task on CPUs (on 20 of 77 nodes)**

Runtime:
19687 sec

Collaboration with ECMWF on FDB

Nicolau Manubens, Simon Smart, Emanuele Danovaro, Tiago Quintino (ECMWF)
Adrian Jackson (EPCC)
Mohamad Chaarawi, Michael Hennecke (Intel)



To be presented at PASC'24, 3-5 June 2024.
Paper available at http://www.arxiv.org/abs/2404.03107

ECMWF | epcc |

daos

THE LINUX FOUNDATION

For more information on the DAOS Foundation and the DAOS Project, please visit

https://daos.io/