Applying DDN to Machine Learning

Jean-Thomas Acquaviva
jacquaviva@ddn.com

Learning from What?

Image data
Facial recognition
Action recognition
Object detection and recognition
Handwriting and character recognition
Aerial images
...

Anomaly data
Time series

Biological data
Human
Animal
Plant
Microbe
Drug Discovery

Multivariate data
Financial
Weather
Census
Transit
Internet
Games
Other multivariate

Text data
Reviews
News articles
Messages
Twitter and tweets
Social network

Signal data
Electrical
Motion-tracking
Other signals

Sound data
Music
Speech data

Physical data
High-energy physics
Systems
Astronomy
Earth science
...

Type of data	Supported operations
discret quantitative data	Calculations, equality / difference, inferiority / superiority
Continuous quantitative data	Calculations, equality / difference, inferiority / superiority
nominal qualitative data	Equality / difference
ordinal qualitative data	Equality / difference, inferiority / superiority

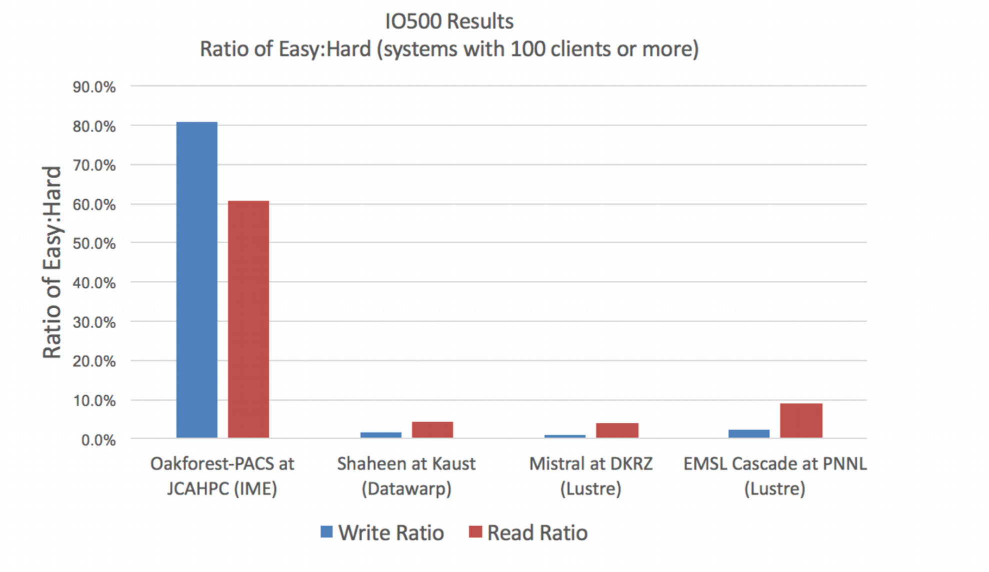
IO Characteristics: Read, Random, High Throughput per Client, File and IO Sizes between a few kb and a few MB

Training Sets typically larger than local caches

Diversity of Load: IO500

Detailed write

Rank	System	Institution	Filesystem	Client Nodes	Score	BW GiB/s	MD kIOP/s	Easy Write GiB/s	Hard Write GiB/s	Hard vs. Easy	Easy Read GiB/s	Hard Read GiB/s	Hard vs. Easy
1	Oakforest-PACS	JCAHPC	IME	2048	101.48	471.25	19.04	742.38	600.28	80.9%	427.41	258.93	60.6%
2	Shaheen	Kaust	DataWarp	300	70.9	151.53	33.17	969.45	15.55	1.6%	894.76	39.09	4.4%
3	Shaheen	Kaust	Lustre	1000	41	54.17	31.03	333.03	1.44	0.4%	220.62	81.38	36.9%

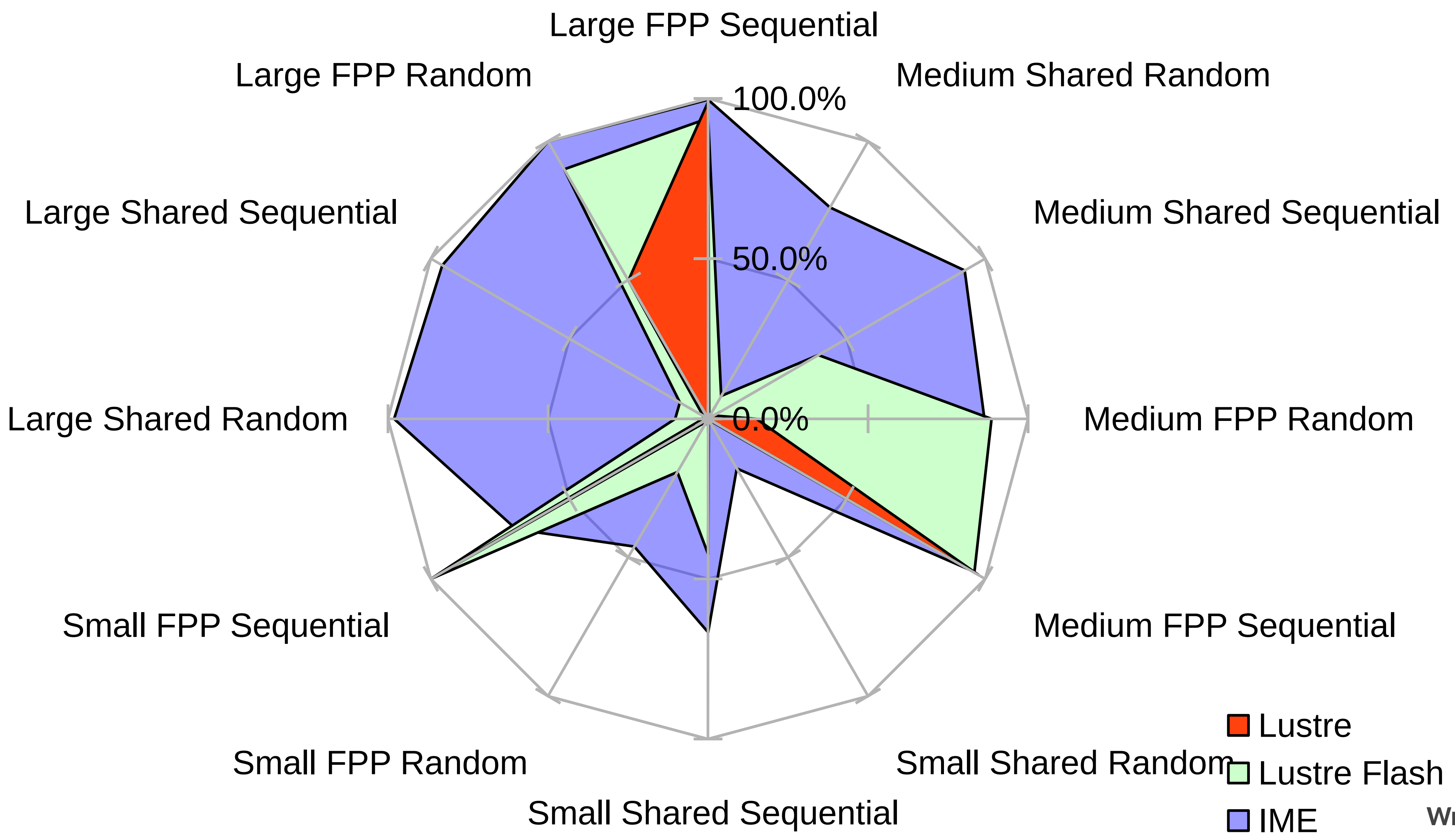


- KAUST BurstBuffer and Lustre at DKRZ show massive falls in IO performance
- Small DDN Lustre based on 12K at PNNL shows a similar pattern
- IME has a order of magnitude better ratio between easy and hard

Acknowledging multi-criteria performance metrics

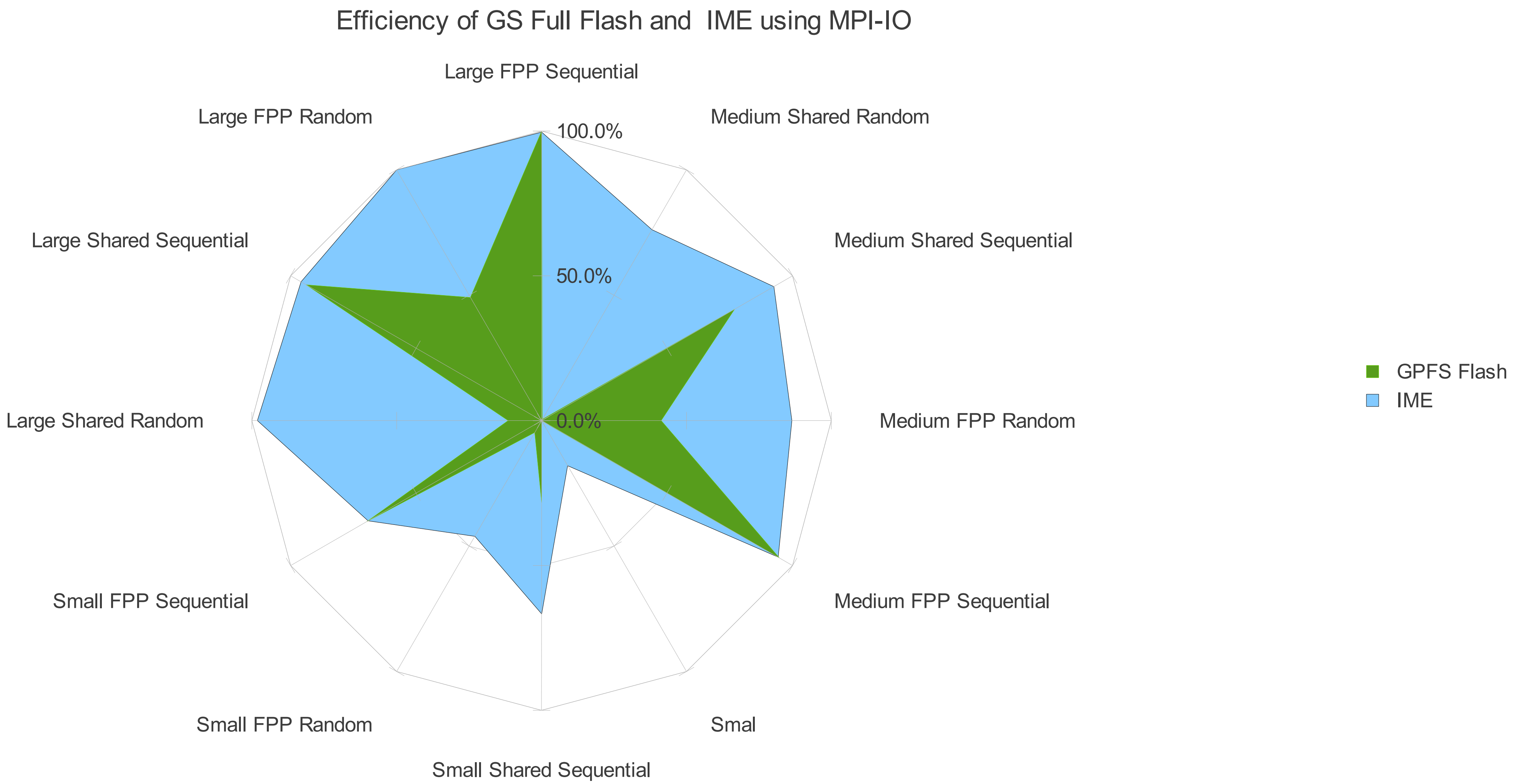
I/O Granularity	I/O control plane Pattern	I/O Data plane Pattern	
Large (>= 1MB)	File Per Process (= share nothing)	Sequential	IO500 Easy !
Large	File Per Process	Random	
Large	Single Shared File	Sequential	
Large	Single Shared File	Random	
Small	File Per Process	Sequential	
Small	File Per Process	Random	
Small (47008 Bytes)	Single Shared File	Sequential	IO500 Hard !
Small	Single Shared File	Random	

IO500 to a comprehensive picture: DDN Flash native vs Lustre



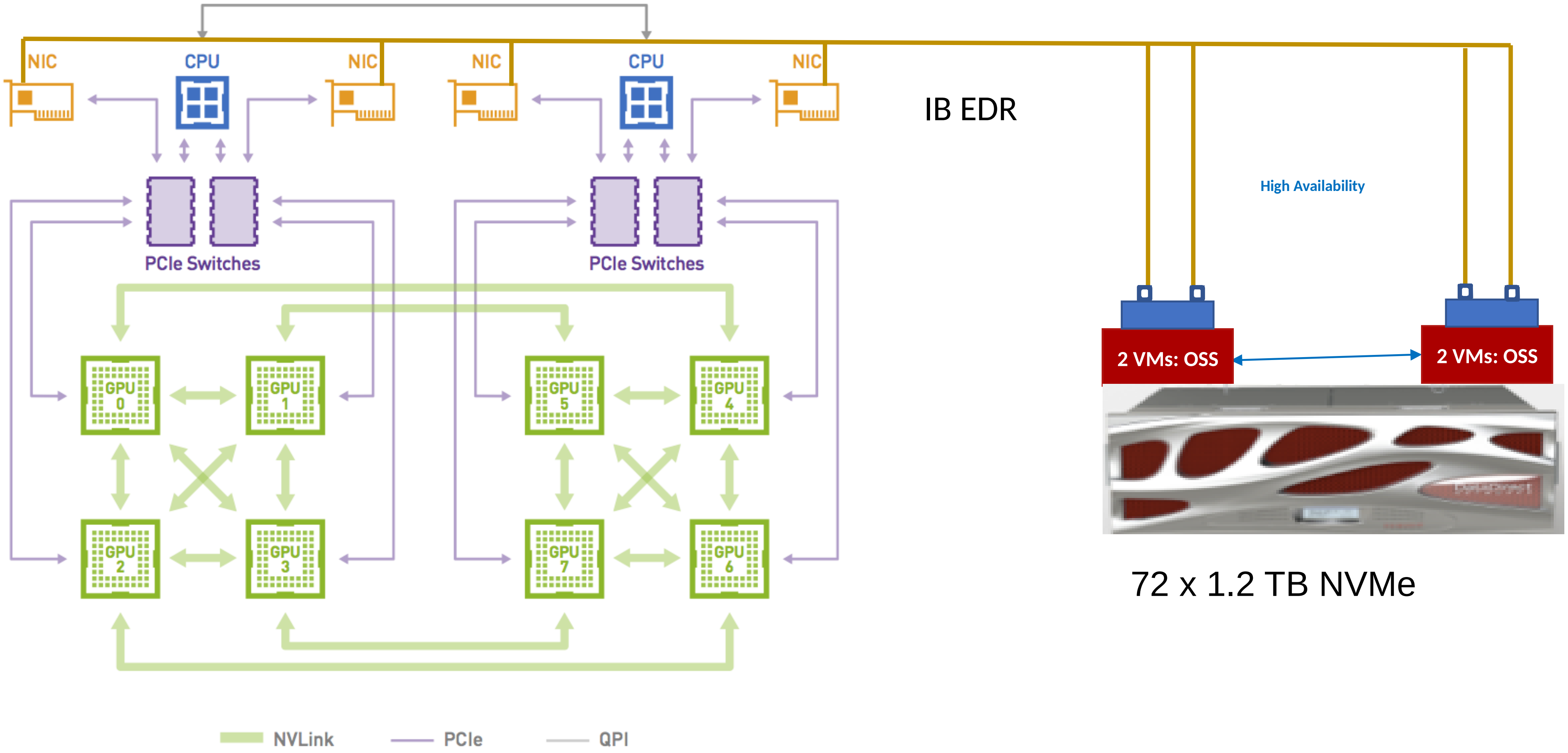
Write patterns

IO500 to a comprehensive picture: DDN Flash native E vs GridScaler



Write patterns

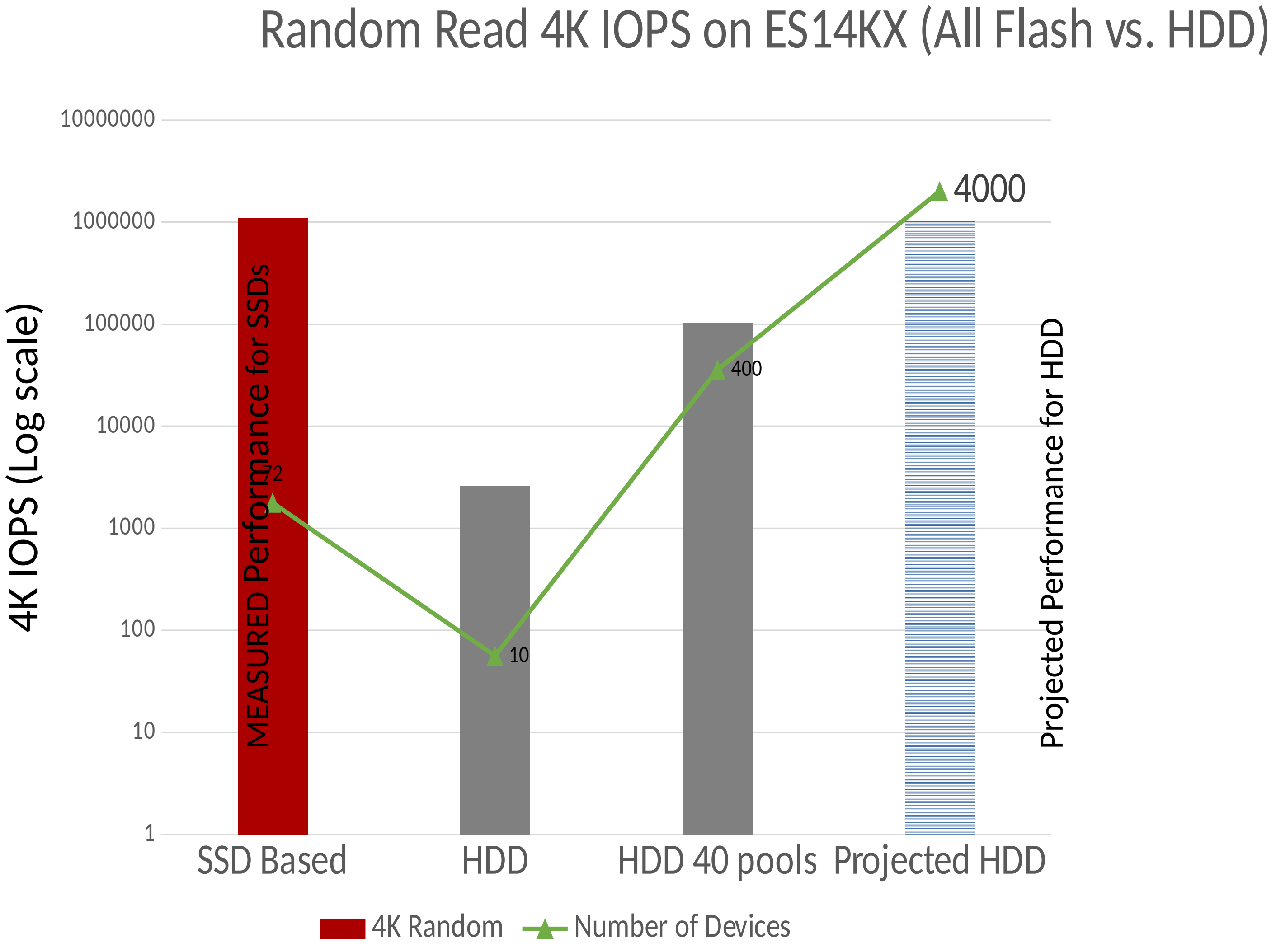
Example: EXAScaler DGX Solution (hardware view)



ES14KXE 72 SSDs

Platform DDN ES14KXE Full Flash: 1M IOPS – 40 GB/s

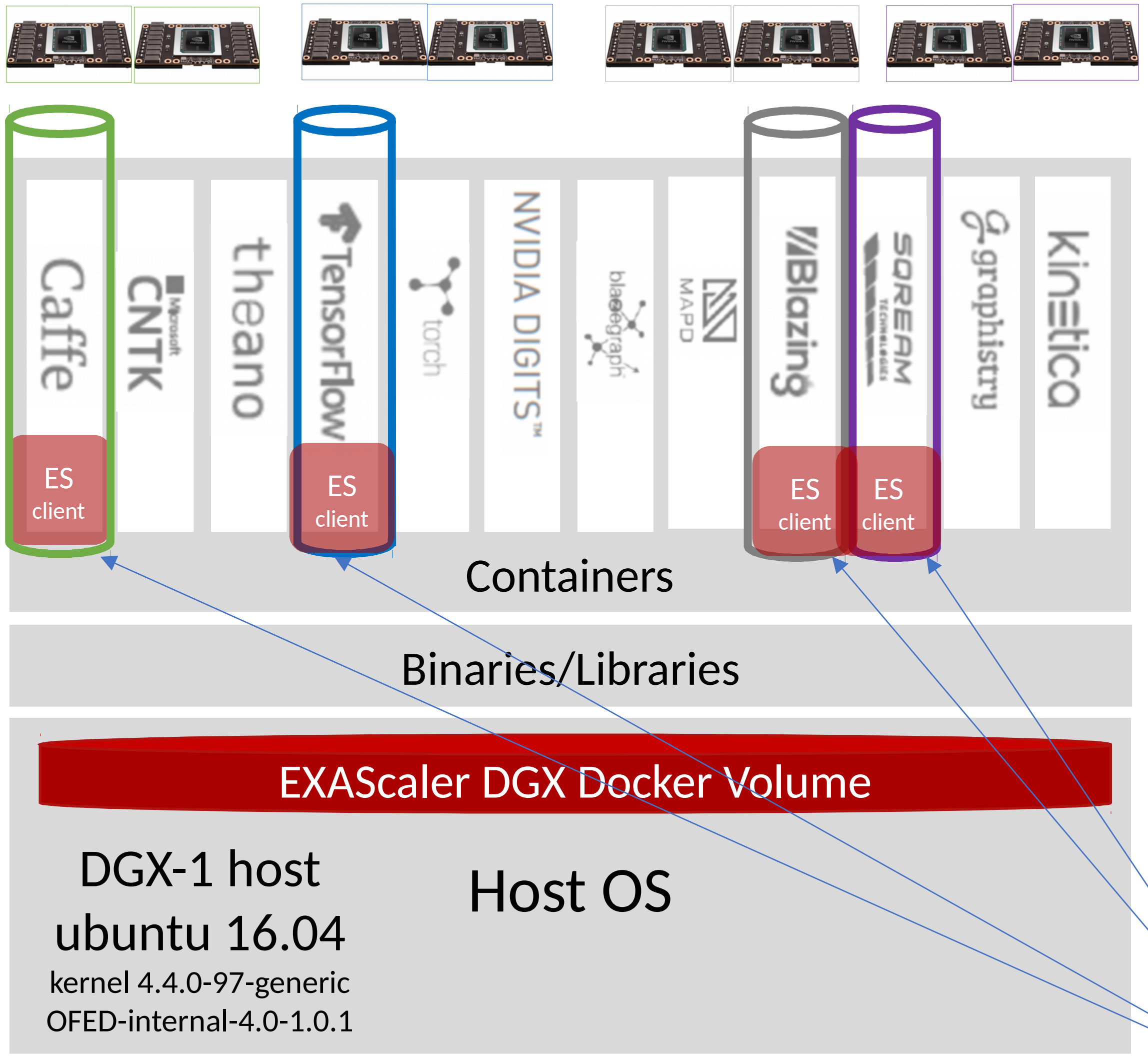
- ES14KX ALL Flash active-active controllers deliver 1M file IOPs – the equivalent of 4000 HDDs
- Scale IOPs further in the namespace with additional controllers
- Augment Flash with HDD at scale with up to 1680 HDDs per controller



WHAT PFS FOR AI APPLICATIONS?

Feature	Importance for AI	GPFS	Lustre
Shared Metadata Operations	**High** - training data are usually curated into a single directory	✘ Lower than 10K (minimal improvements with v5)	✓ Up to 200K
Support for high-performance mmap() I/O Calls	**High** - many AI applications use mmap() calls	✘Extremely poor	✓ Strong
Container Support	**High** - most AI applications are containerized	✘ Poor (network complexity & root issues)	✓ Available
Data Isolation for Containers	**Medium/High** – important for shared environments	✘ Not available today	✓ Available
Data-on-Metadata (small file support)	**Medium/High** – depends on data set	✘ DOM only for files smaller than 3.4k	✓ DOM is highly tunable
Unique Metadata Operations	**Medium** - depends on Installation Size and Application Workflow	✓Highly scalable	✓ Highly scalable with DNE 1/2

Example: EXAScaler DGX Solution (host part)



- Integrated Flash Parallel File System Access via TCP or IB
- Extreme Data Access Rates for concurrent DGX Containers

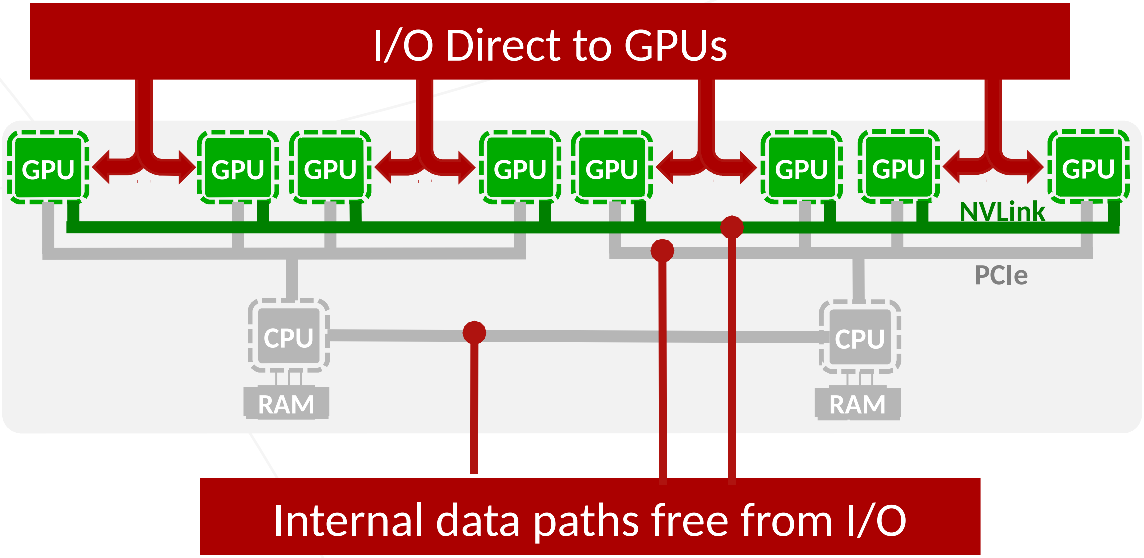
EXAScaler DGX Docker Volume

- Lustre ES3.2 kernel modules compiled for Ubuntu kernel and host's OFED
- Lustre userspace tools
- scripts for Lustre mount/umount

Resource isolation: Ios/GPUs/NIC/memory/namespaces
For the application/SW suite

CONTAINER PINNING

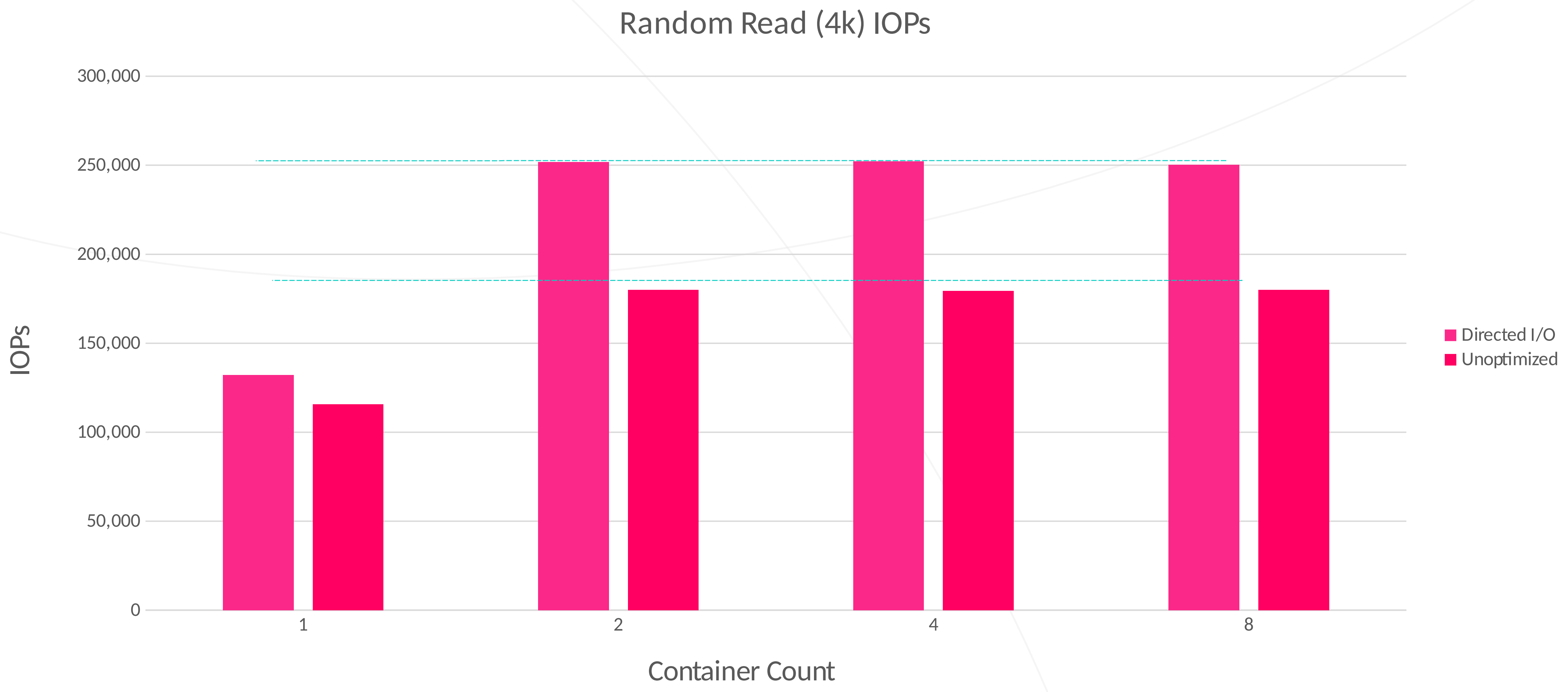
DDN's EXASCALER for DGX manages I/O-paths optimally through DGX-1 to maximize performance to your AI application and keep IO traffic from consuming internal data paths



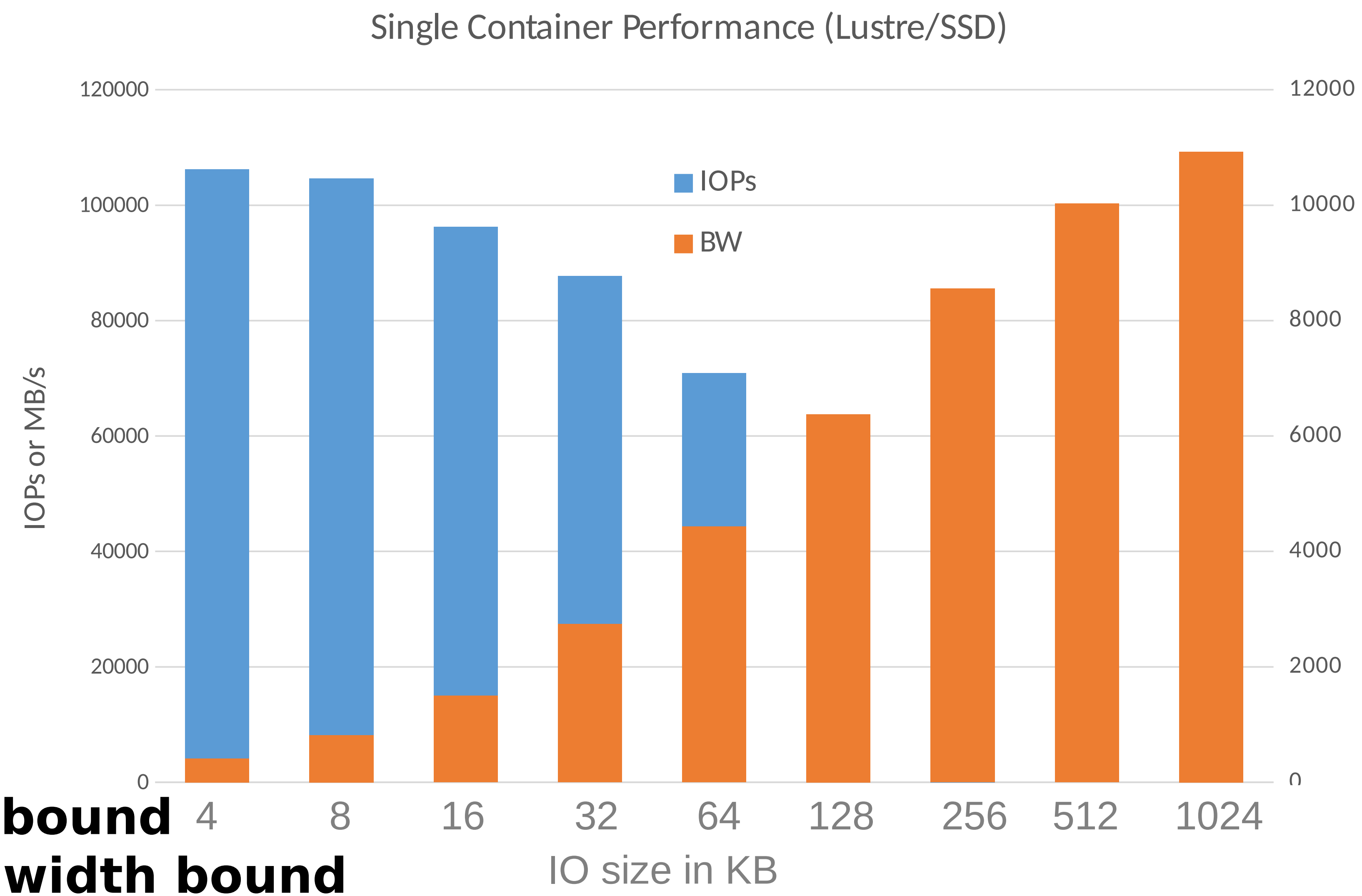
mmap() support

LMDB is key to manage file in several framework (caffé). LMDB relies on mmap() hence page fault to trigger kernel internal I/O call (read_ahead)

CONTAINER PINNING OPTIMIZATION



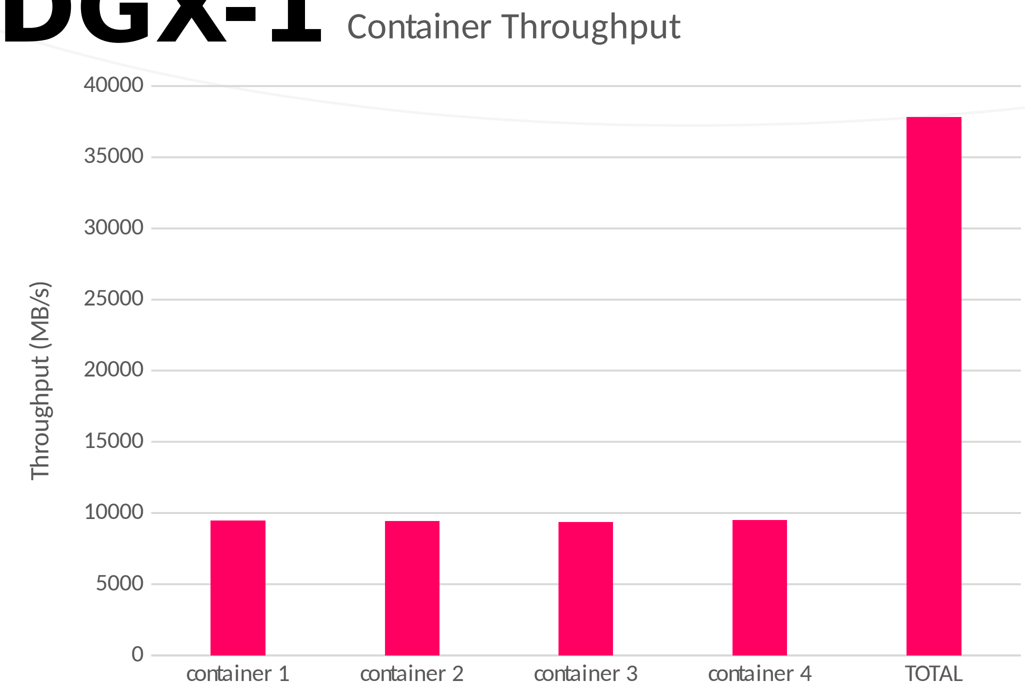
FROM IOPS TO BANDWIDTH

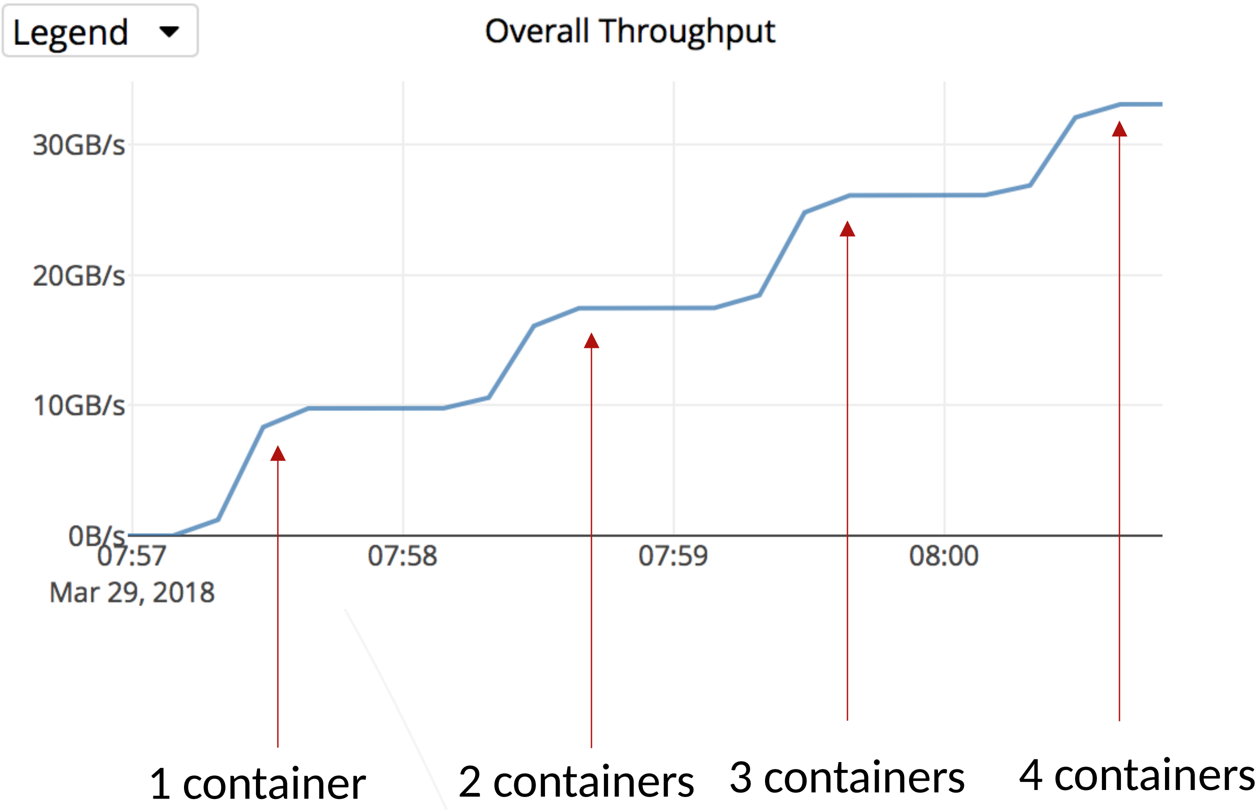


4KB random read is IOPS bound
1MB random read is Bandwidth bound

DGX CONTAINER THROUGHPUT

SCALING UP WORKLOADS IN DGX-1





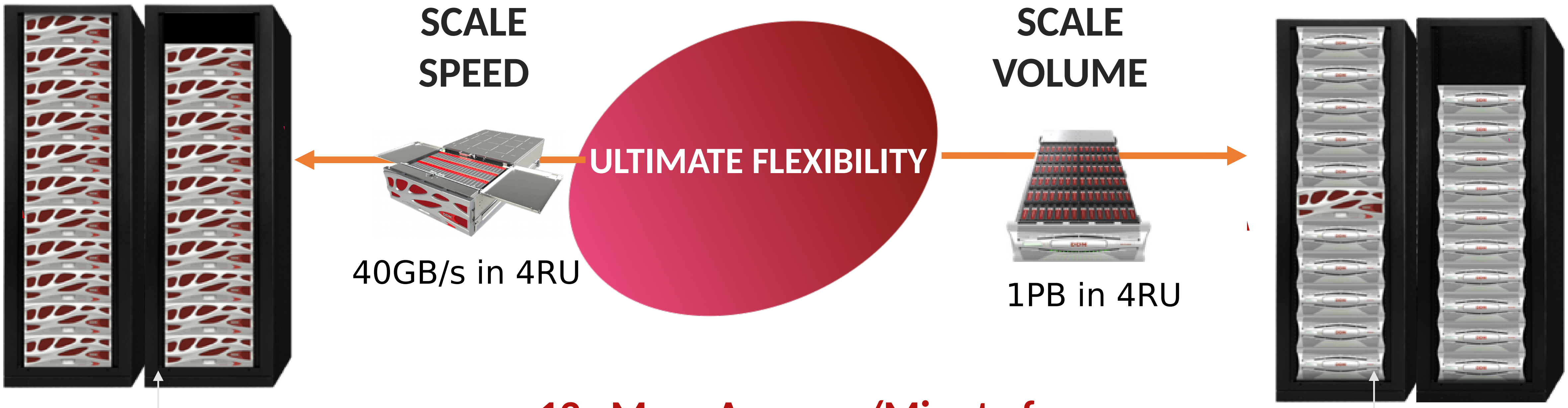
EXASCALER ALL FLASH

Remove I/O burden from data scientist shoulders

→ I/O no longer the limiting factor
→ Saturation of the network
→ 250 KIOPS on a DGX-1
→ 1 Millions IOPS with 4 DGX-1

2018 DDN x12 Analytics

Bringing HPC technologies and know-how to analytics = x12



1TB/s &
10M IOPS

12x More Answers/Minute for
Your AI at Any Scale!

Over 20PB