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

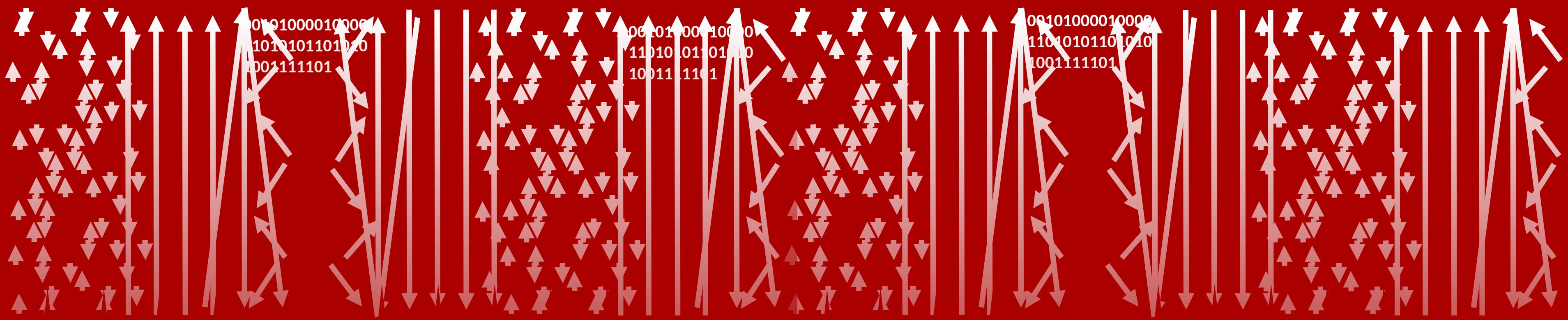
High-energy physics
Systems
Astronomy
Earth science
...

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

Machine Learning

Big Data

NoSQL Analytics



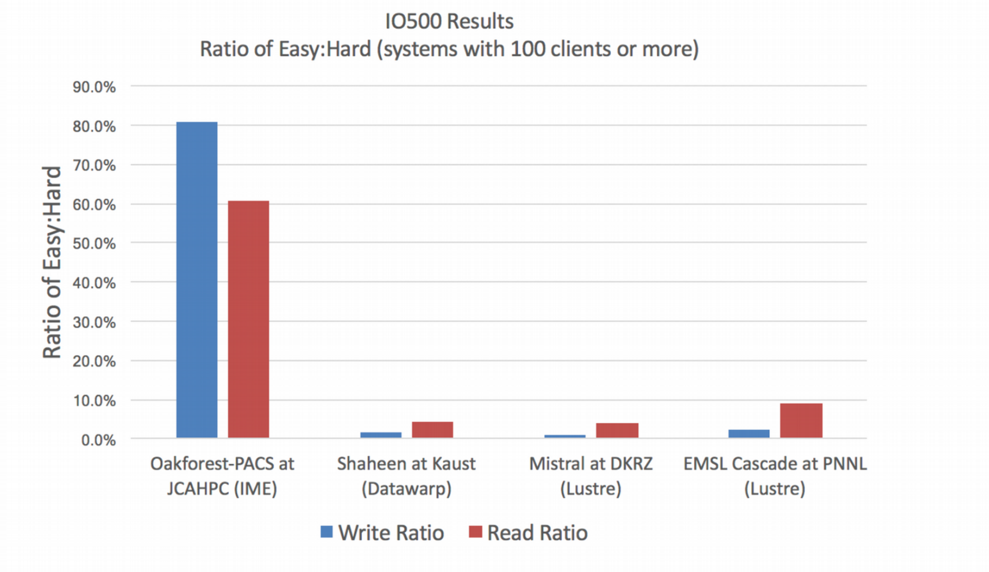
IO Characteristics: Read, Random, High Throughput per Client, File and IO Sizes between a few kb and a few MB

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

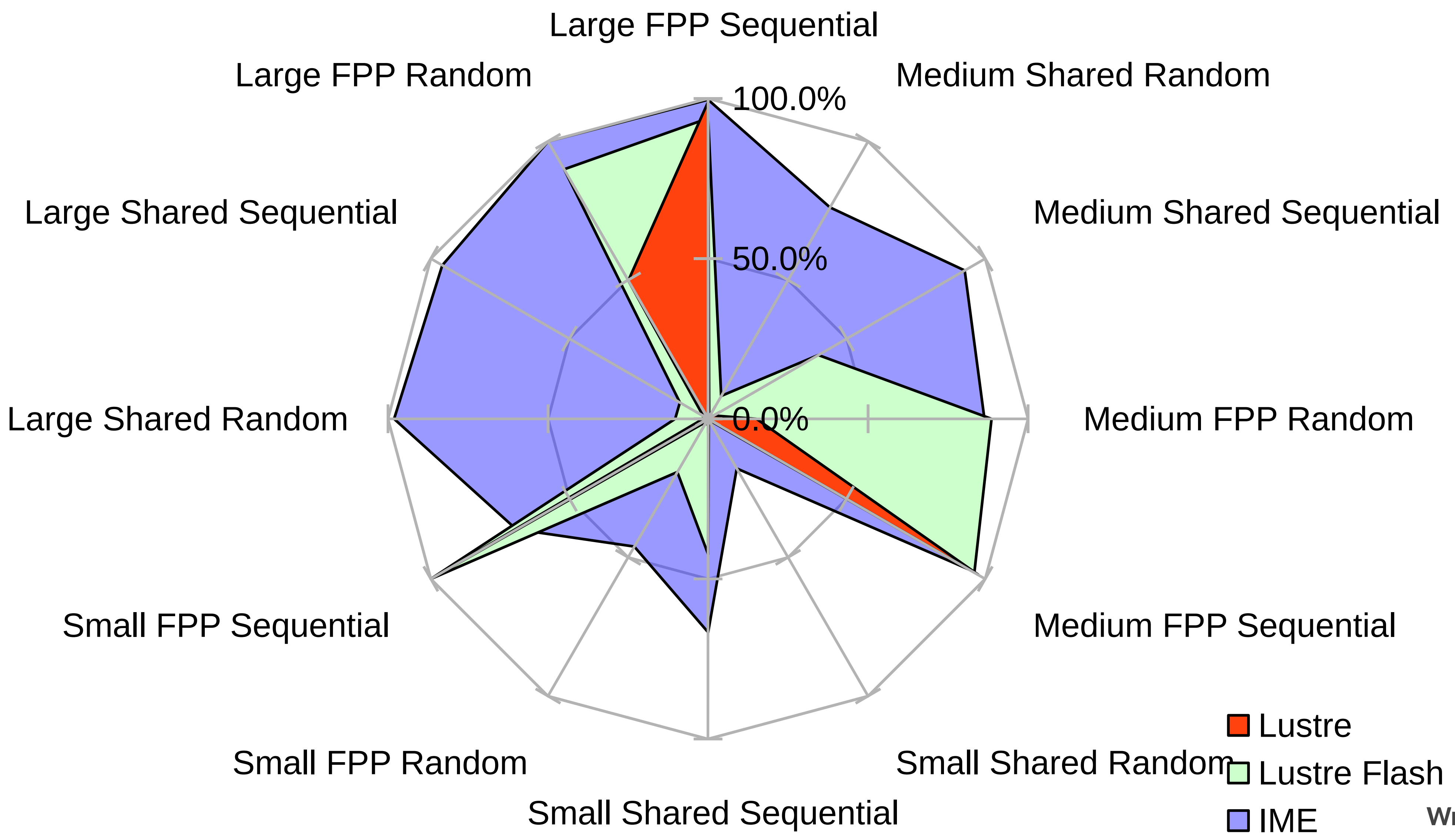
Acknowledging multi-criteria performance metrics

I/O Granularity	I/O control plane Pattern	I/O Data plane Pattern
Large (>= 1MB)	File Per Process (= share nothing)	Sequential
Large	File Per Process	Random
Large	Single Shared File	Sequential
Large	Single Shared File	Random
Medium (47008 Bytes)	File Per Process	Sequential
Medium	File Per Process	Random
Medium	Single Shared File	Sequential
Medium	Single Shared File	Random
Small (4KB)	File Per Process	Sequential
Small	File Per Process	Random
Small	Single Shared File	Sequential
Small	Single Shared File	Random

IO500 Easy !

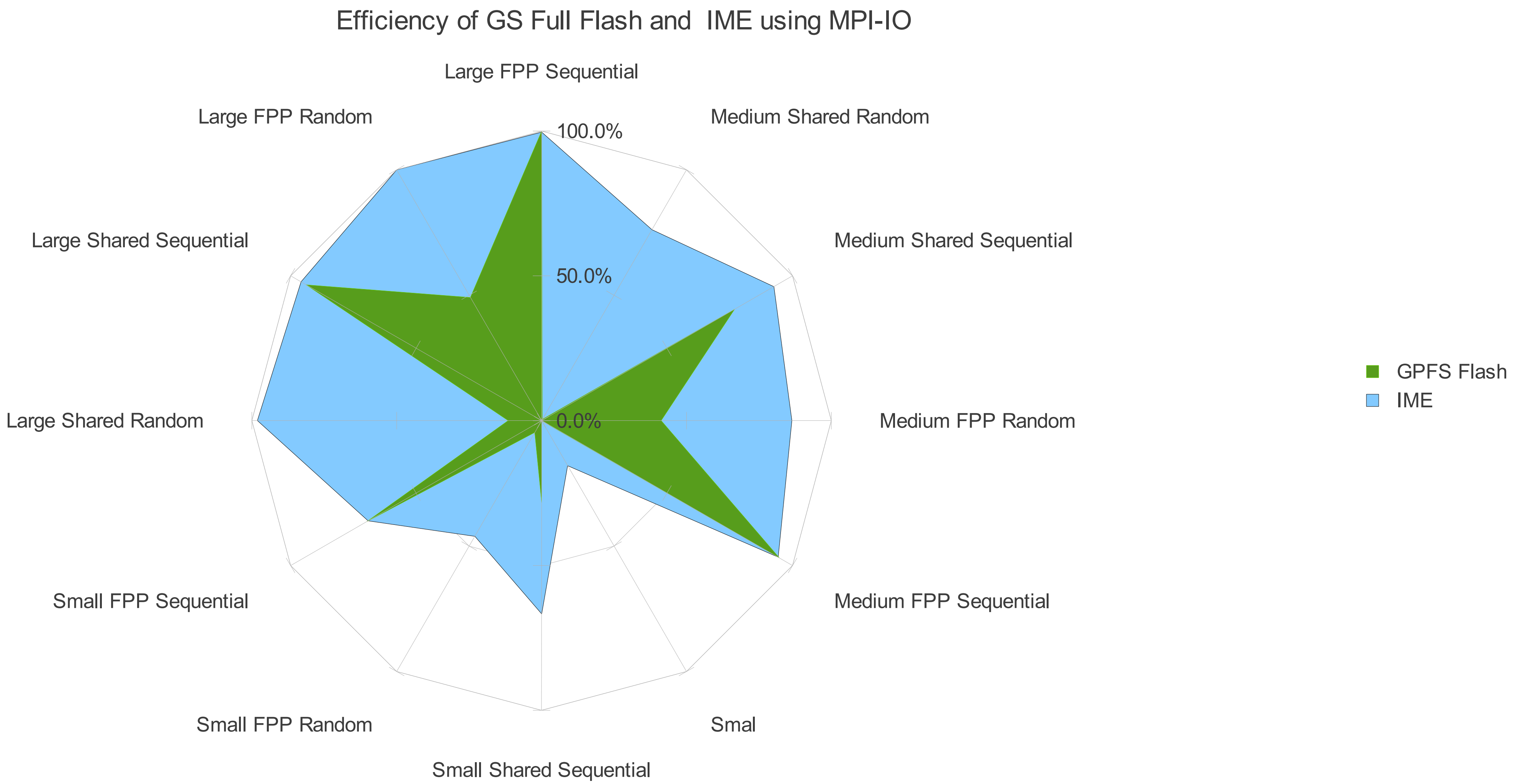
IO500 Hard !

IO500 to a comprehensive picture: DDN Flash native vs Lustre



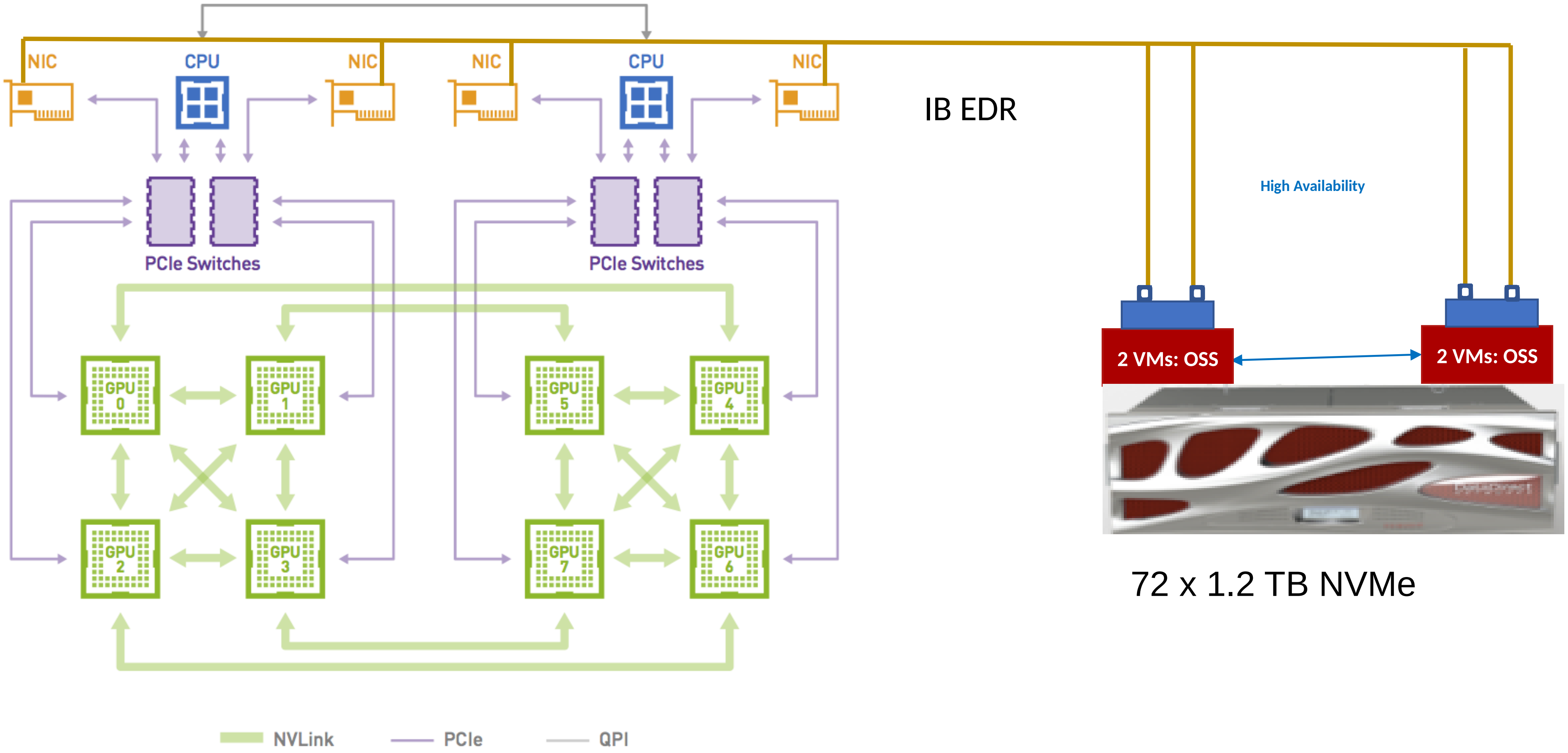
Write patterns

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



Write patterns

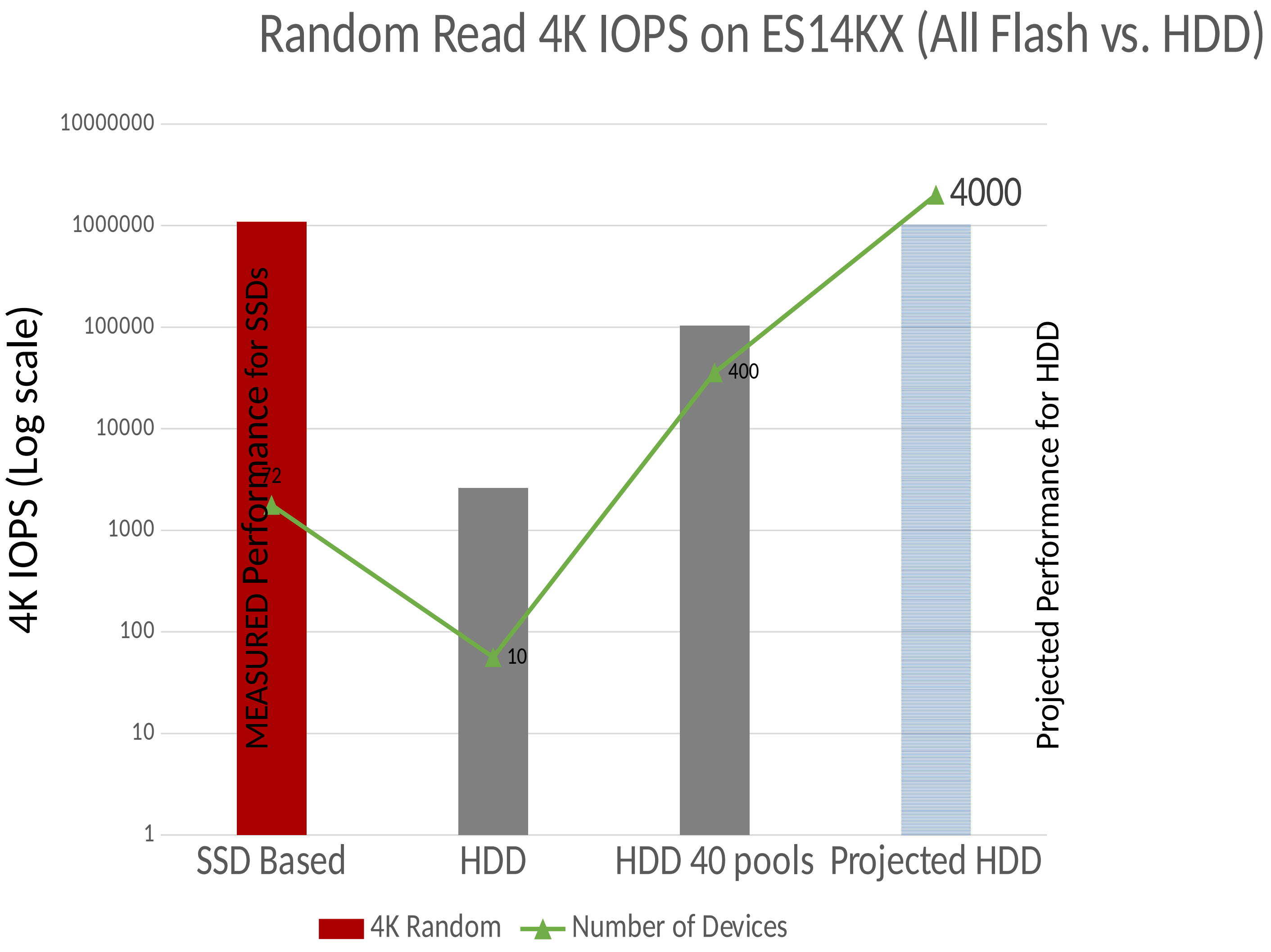
Example: EXAScaler DGX Solution (hardware view)



ES14KXE 72 SSDs

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

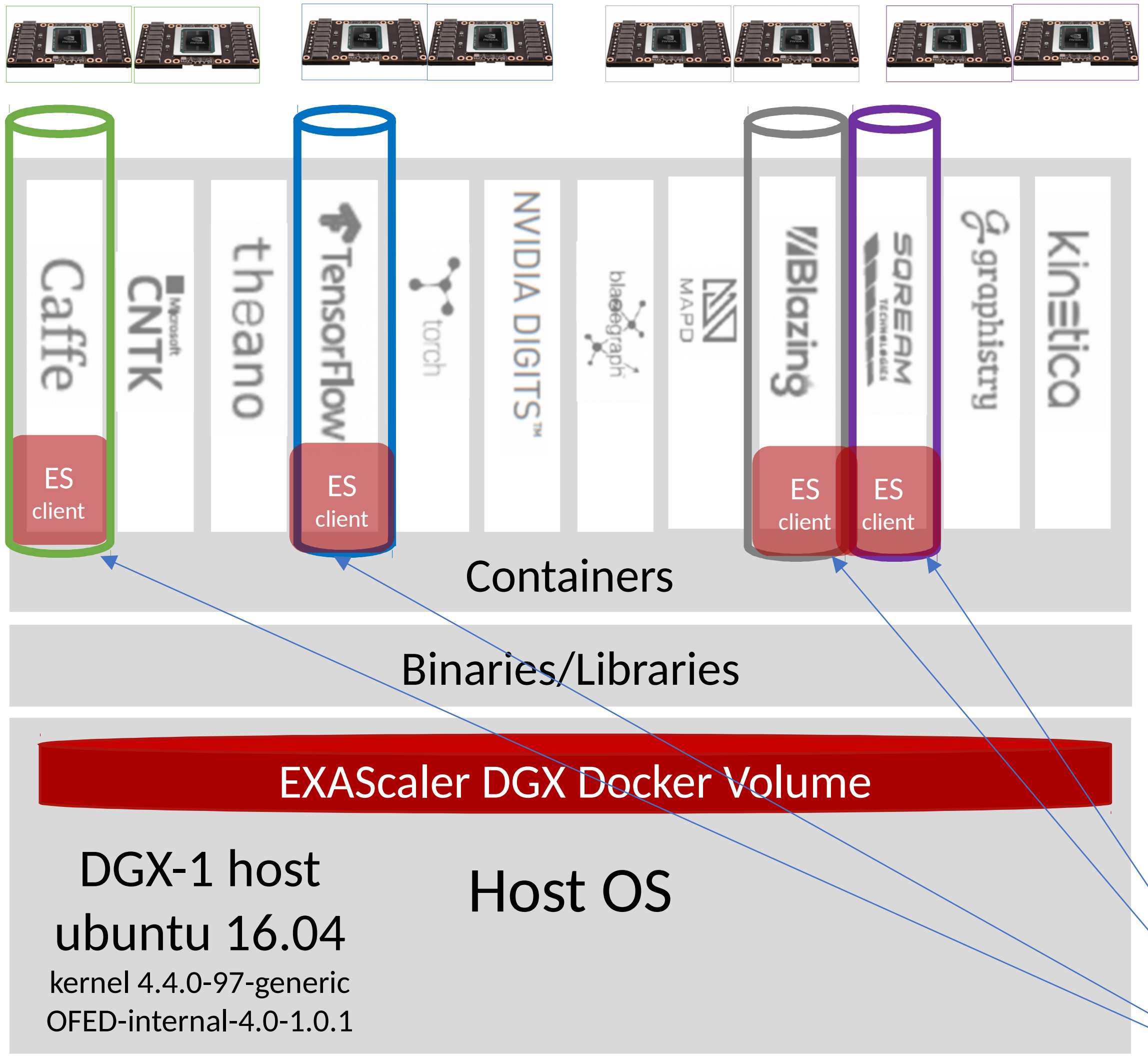
- ES14KX ALL Flash active-active controllers deliver 1M file IOPs – the equivalent of 4000 HDDs
- Scale IOPs further in the namespace with additional controllers
- Augment Flash with HDD at scale with up to 1680 HDDs per controller



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)



Tesla P100 NVLINK (170 Tflps)

- 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

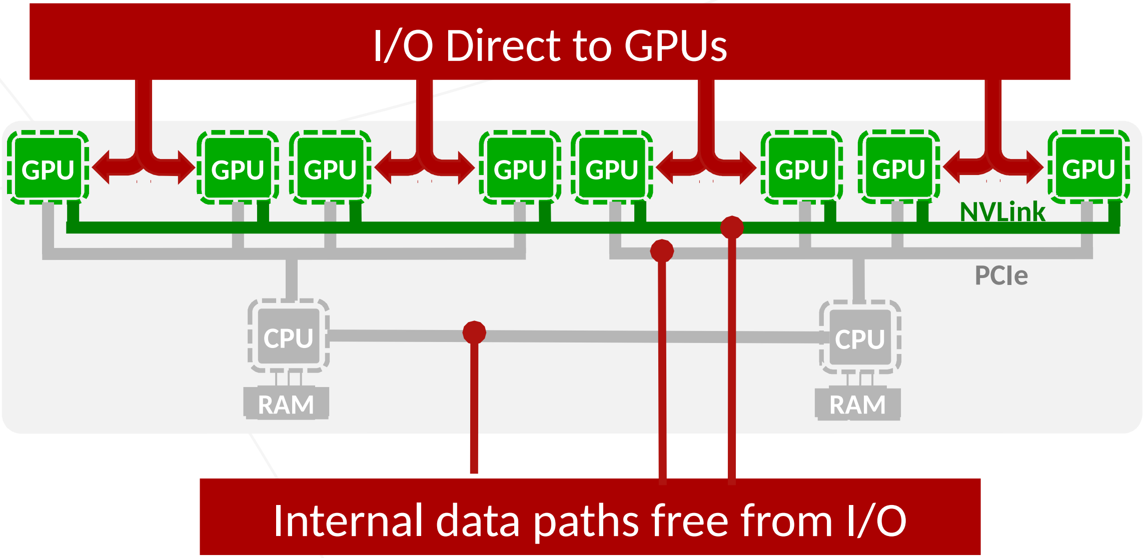
Resource isolation: Ios/GPUs/NIC/memory/namespaces
For the application/SW suite

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()



SIDE NOTE ON LMDB MMAP() 1/2

Lightning Memory-Mapped Database (LMDB)

- Ubiquitous in Deep learning framework
- At the core of file management for Caffé and Tensor flow
- Declare file as loaded in memory using mmap()

MMAP() declares as already in memory

- Similar to page swap

```
int fd = open(“my_file”, O_RDONLY, 0);
void* mmappedData = mmap(NULL, filesize, PROT_READ, MAP_PRIVATE, fd, 0);
MD5_Update (&mdContext, mmappedData, filesize);
```

- Memory pages are provisioned to potentially host the whole filesize
- First access to a page triggers a page fault.

→ Page fault mechanism will ask the kernel to emit the I/O requests

SIDE NOTE ON LMDB MMAP() 2/2

Page fault mechanism will ask the kernel to emit the I/O requests

- → No Posix read() or write() at the application level
- → I/O access are dealt internally by the kernel / page cache (readahead)

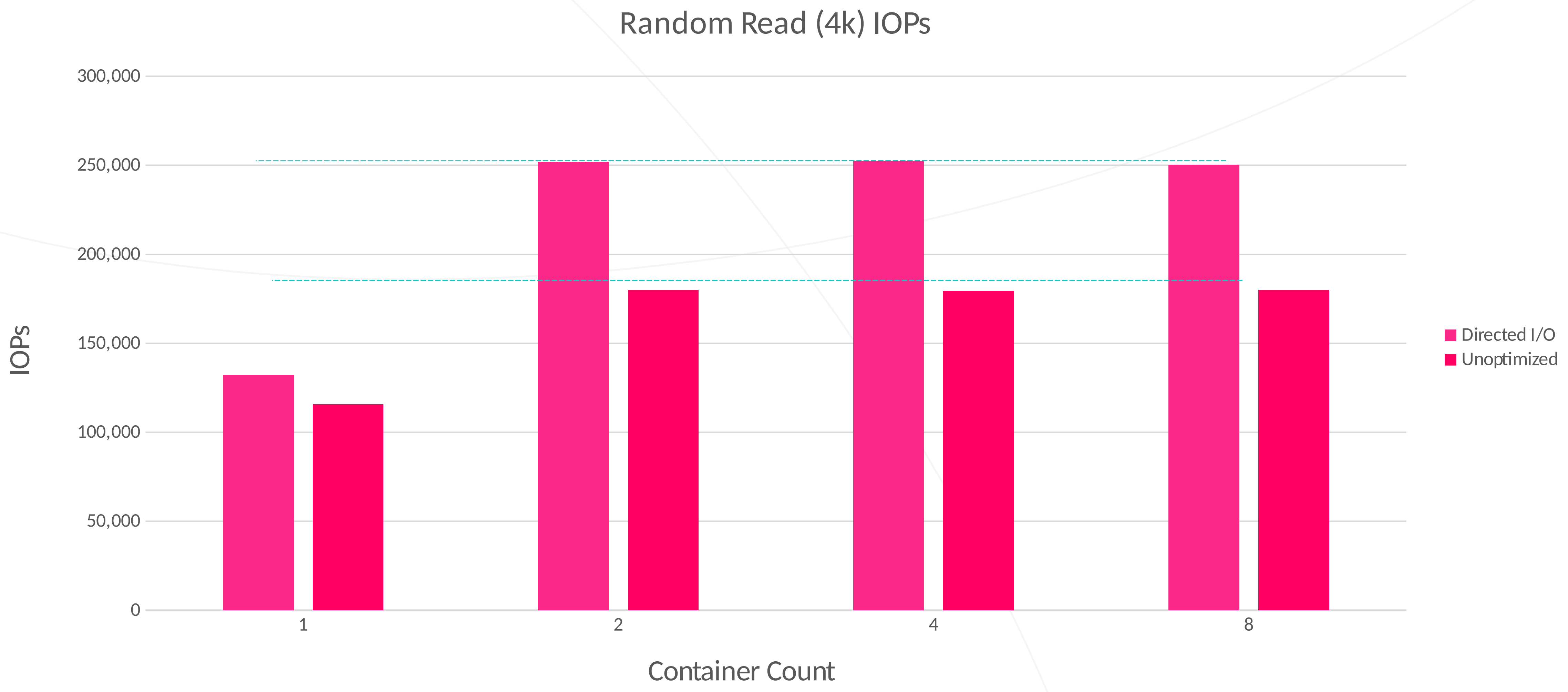
mmap() is nice for data scientists

- Abstraction of the storage layer
- Unified API

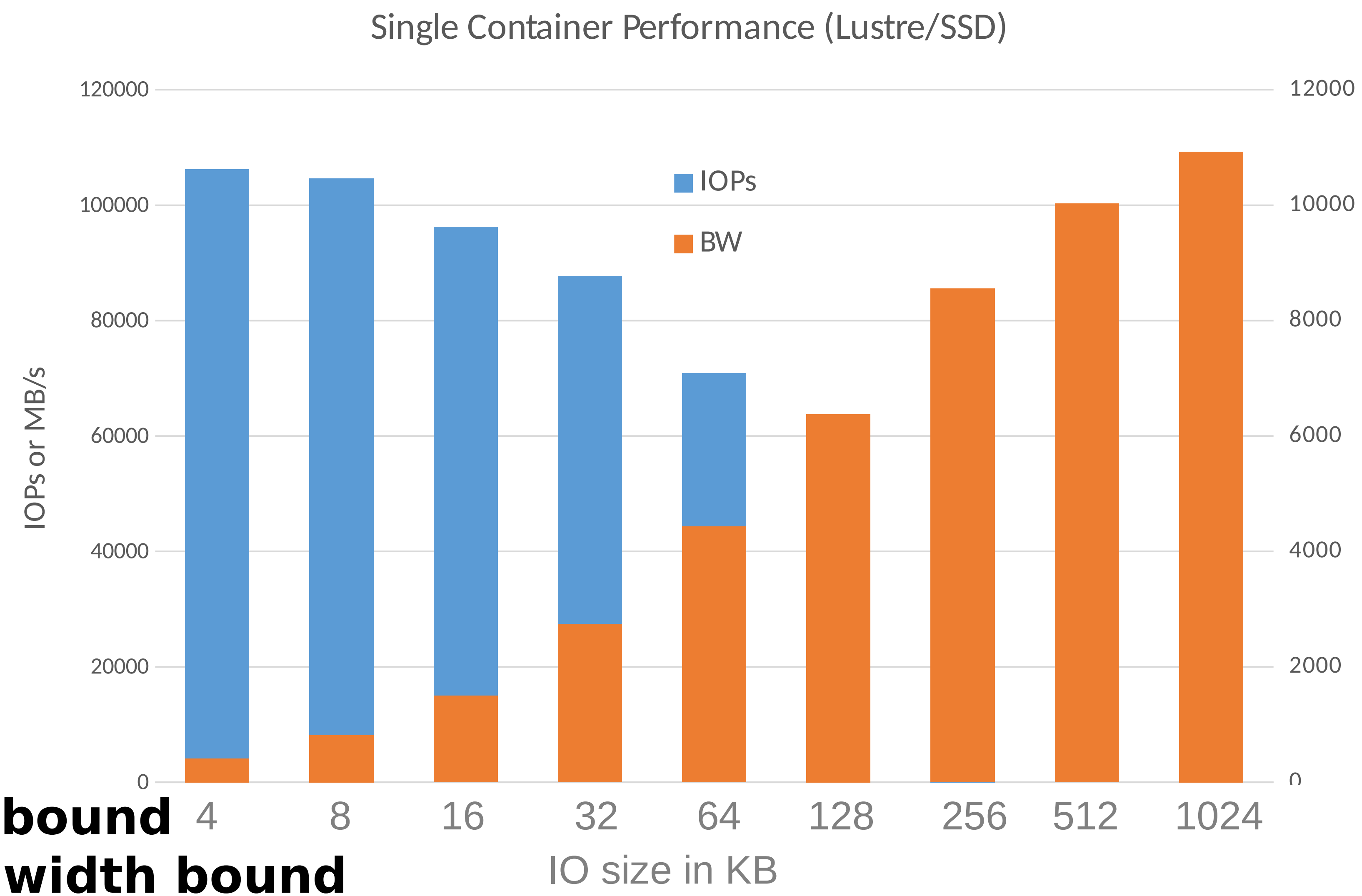
mmap() is tough for computer scientists

- Kernel activity is more complicated to monitor than application
- Tracing tools (ftrace) have significant overhead
- Work with Julian and Eugen on this topic

CONTAINER PINNING OPTIMIZATION



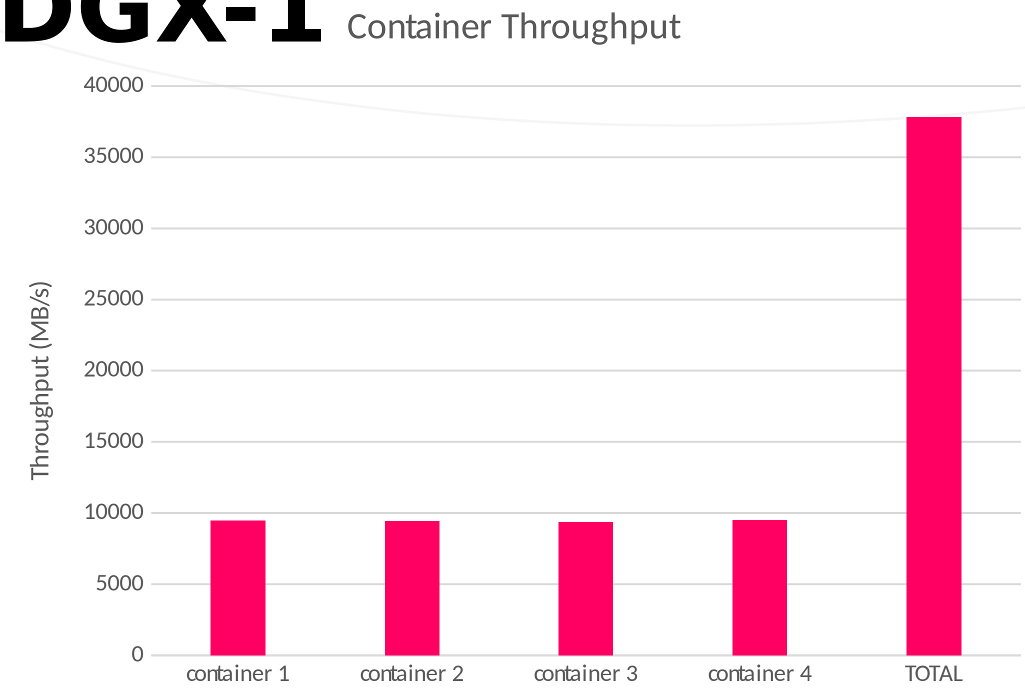
FROM IOPS TO BANDWIDTH

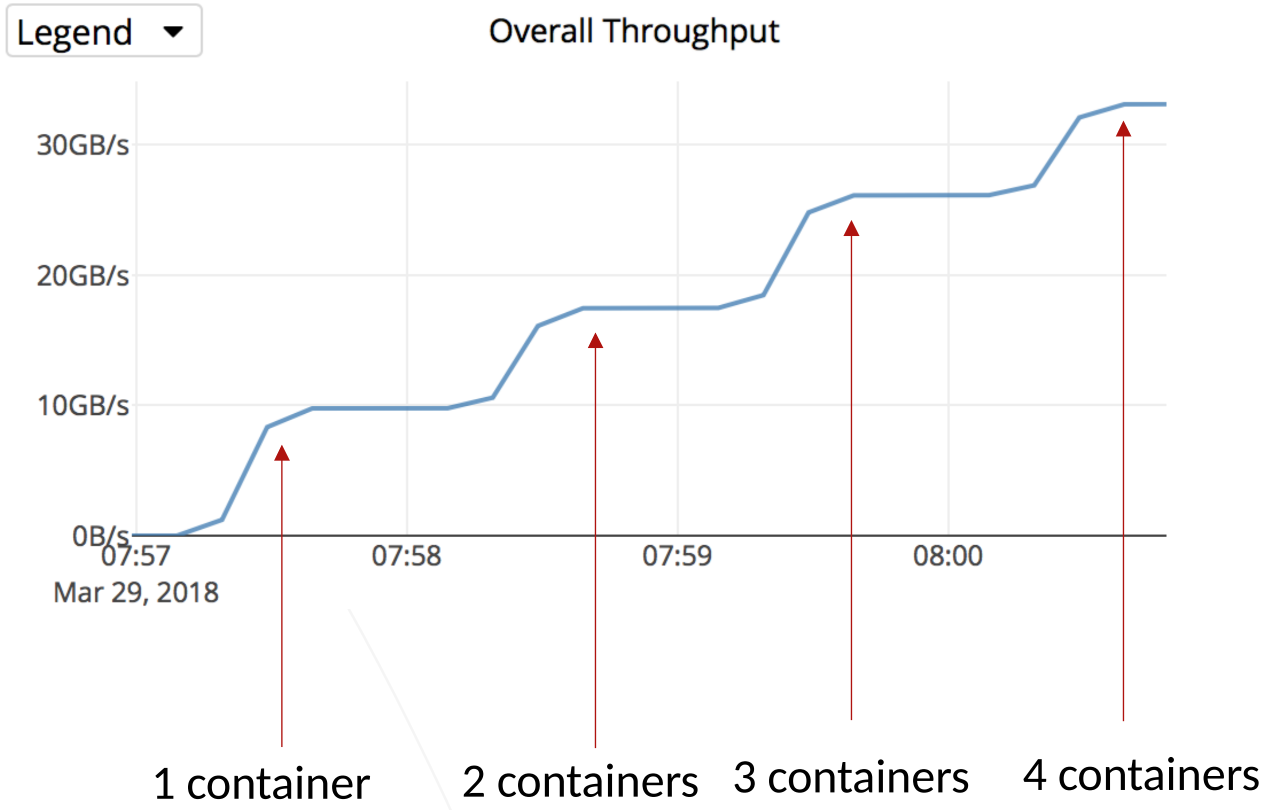


4KB random read is IOPS bound
1MB random read is Bandwidth bound

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

Single DGX-1

2018 DDN x12 Analytics

Bringing HPC technologies and know-how to analytics = x12

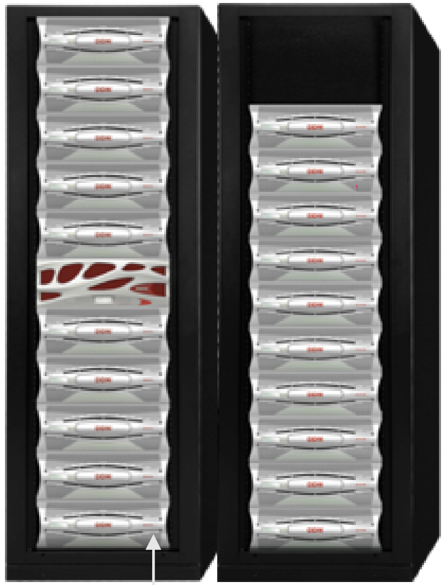
SCALE
SPEED

40GB/s in 4RU

ULTIMATE FLEXIBILITY

SCALE
VOLUME

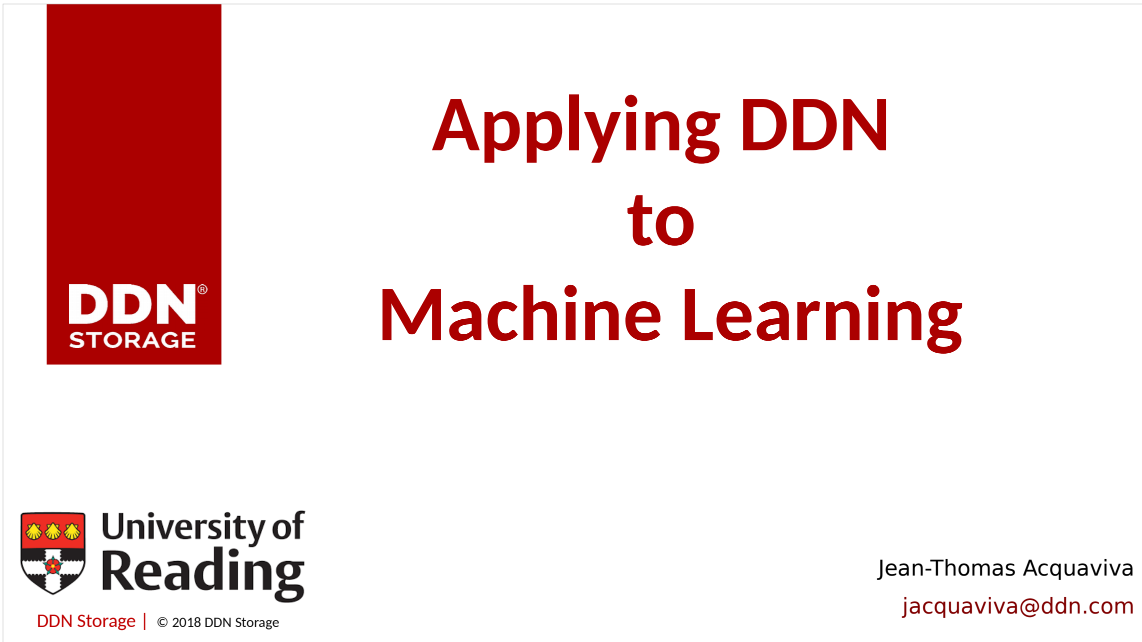
1PB in 4RU



1TB/s &
10M IOPS

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

Over 20PB

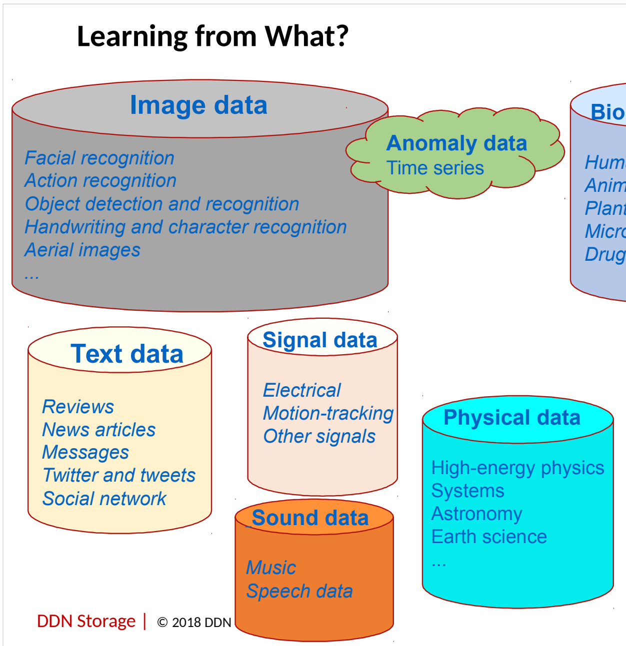


Systems that automatically learn without being programmed, it’s easy to understand, but complicated to put in place
getting value from large amount of data faced resolution of complex compute

Analytics allow you to understand the meaning behind your data
ML is to predict and act; train model that learn of taking decision

For this you need a powerfull compute and an efficient storage to feed your compute with data

The subject of this talk is to tell you how DDN



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

In our case intelligent car data are coming from, image, text (information on traffic), electrical vehicule sensors, weather and why not anomaly data (not an exhaustiv list)

How ML BD and NoSQL interfere with such flow of data

Machine learning use different kind of mathematical algoritm like (description of applied math methods)

Different sofware are available to manage that amount of data

Now on the market one kind find different frameworks too

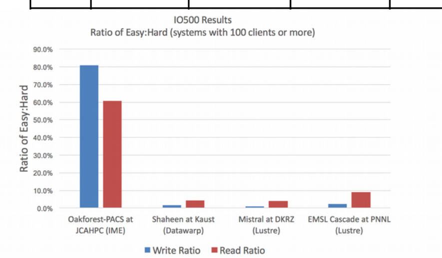
IO profile: In ML mainly we read, random HThroughput and file / IO size

Image-based deep learning for classification, object detection and segmentation benefit from high streaming bandwidth, random

Diversity of Load: IO500

Detailed write

Rank	System	Institution	Filesystem	Client Nodes	Score	BW	MD	Easy Write	Hard Write	Hard vs. Easy	Easy Read	Hard Read	Hard vs. Easy
						GiB/s	kIOP/s	GiB/s	GiB/s		GiB/s	GiB/s	
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

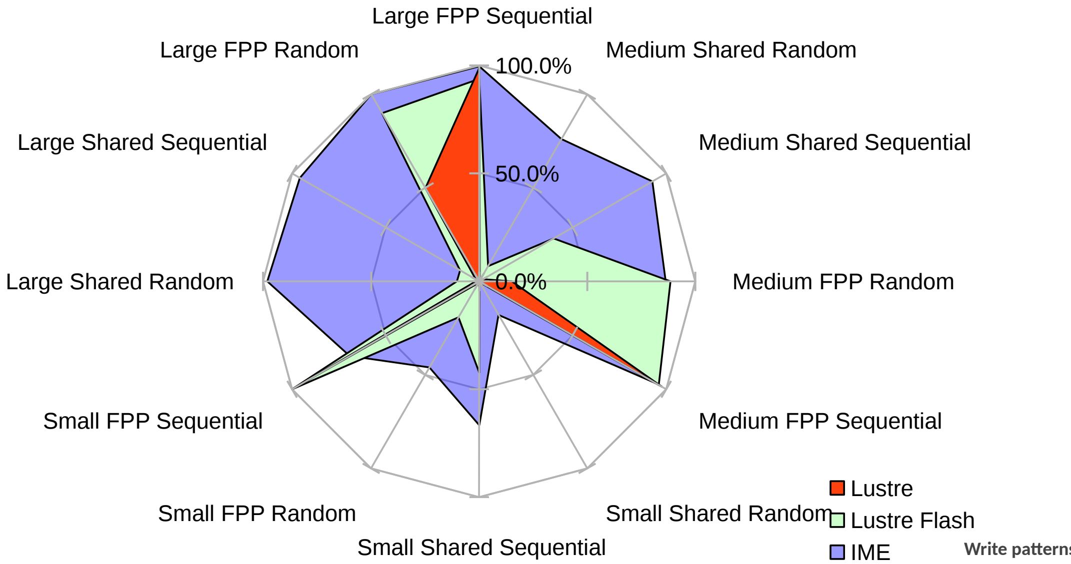
Acknowledging multi-criteria performance metrics

I/O Granularity	I/O control plane Pattern	I/O Data plane Pattern
Large (>= 1MB)	File Per Process (= share nothing)	Sequential
Large	File Per Process	Random
Large	Single Shared File	Sequential
Large	Single Shared File	Random
Medium (47008 Bytes)	File Per Process	Sequential
Medium	File Per Process	Random
Medium	Single Shared File	Sequential
Medium	Single Shared File	Random
Small (4KB)	File Per Process	Sequential
Small	File Per Process	Random
Small	Single Shared File	Sequential
Small	Single Shared File	Random

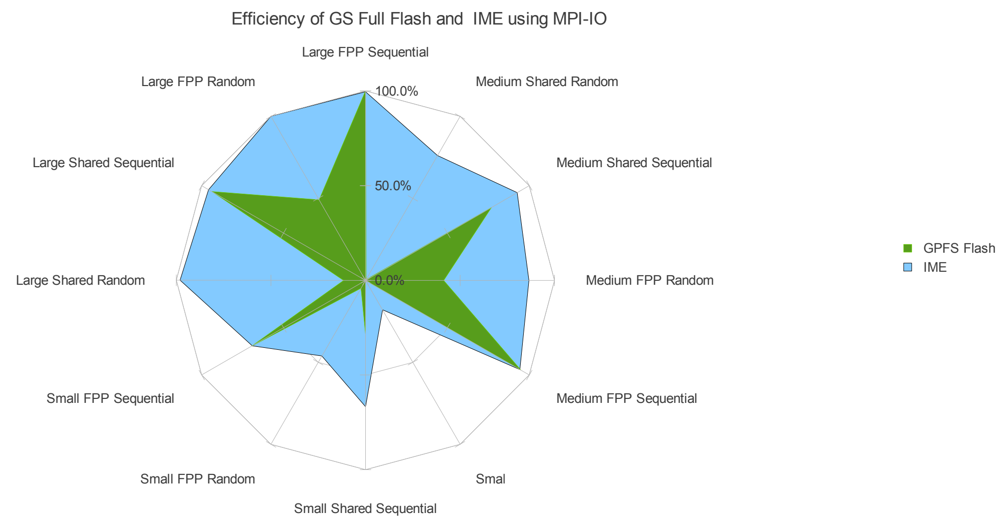
IO500 Easy !

IO500 Hard !

IO500 to a comprehensive picture: DDN Flash native vs Lustre

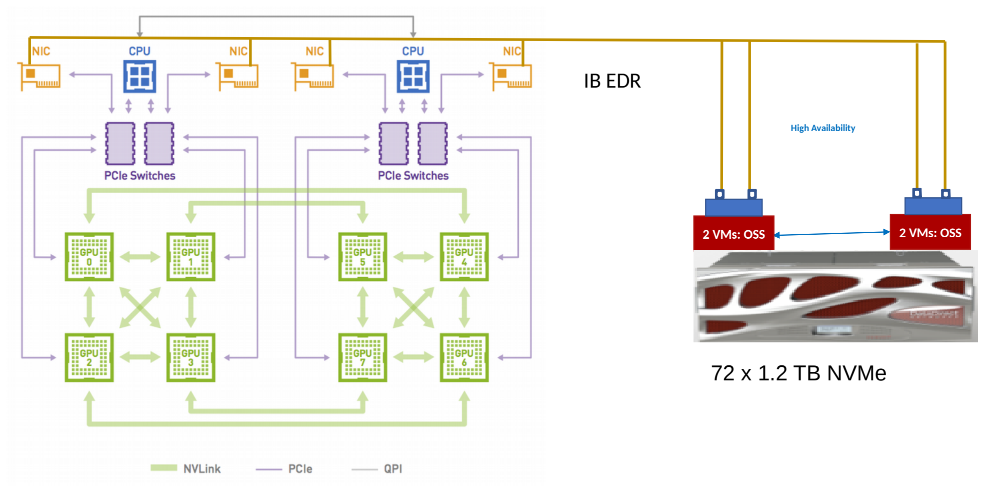


IO500 to a comprehensive picture: DDN Flash native E vs GridScaler



Write patterns

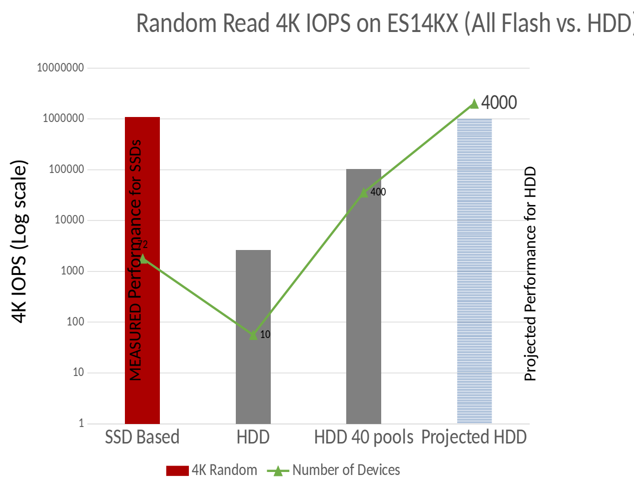
Example: EXAScaler DGX Solution (hardware view)



ES14KXE 72 SSDs

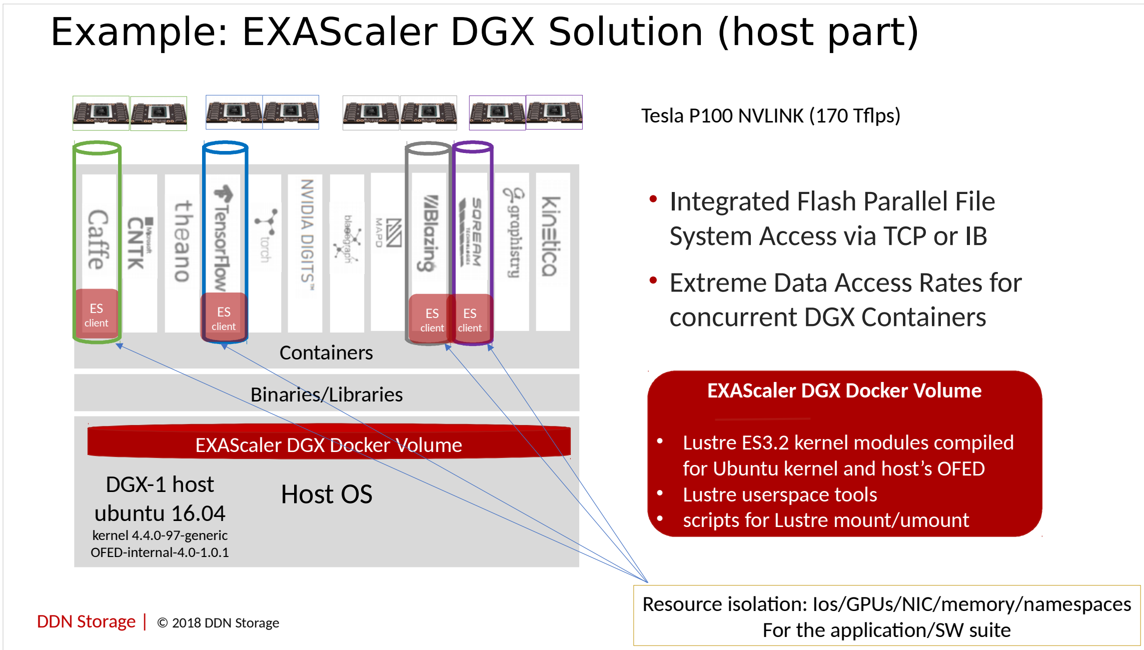
Platform DDN ES14KXE Full Flash: 1M IOPS – 40 GB/s

- 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



Docker est un outil qui peut empaqueter une application et ses dépendances dans un conteneur isolé, qui pourra être exécuté sur n'importe quel serveur ». Ceci permet d'étendre la flexibilité et la portabilité d'exécution d'une application, que ce soit sur la machine locale, un cloud privé ou public, une machine nue, etc

Il s'appuie sur les fonctionnalités du noyau et utilise l'isolation de ressources (comme le processeur, la mémoire, les entrées et sorties et les connexions réseau) ainsi que des espaces de noms séparés pour isoler le système d'exploitation tel que vu par l'application.

DGX est sur ubuntu dont le kernel change très vite (pas comme rh ou centos)

DDN a développé des scripts pour compiler/installer /utiliser un client lustre à la volée par container

Un container par appli/suite ML on peut

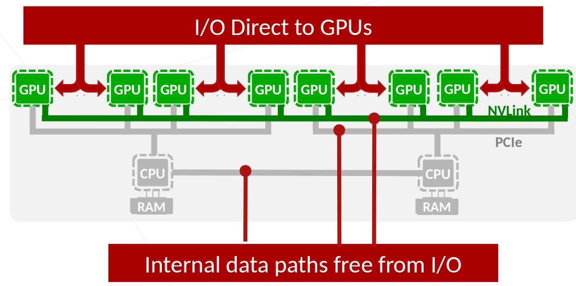
EXASCALER + DGX-1

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()



SIDE NOTE ON LMDB MMAP() 1/2

Lightning Memory-Mapped Database (LMDB)

- Ubiquitous in Deep learning framework
- At the core of file management for Caffé and Tensor flow
- Declare file as loaded in memory using mmap()

MMAP() declares as already in memory

- Similar to page swap

```
int fd = open(“my_file”, O_RDONLY, 0);
void* mmappedData = mmap(NULL, filesize, PROT_READ, MAP_PRIVATE, fd, 0);
MD5_Update (&mdContext, mmappedData, filesize);
```

- Memory pages are provisioned to potentially host the whole filesize
- First access to a page triggers a page fault.

→ Page fault mechanism will ask the kernel to emit the I/O requests

SIDE NOTE ON LMDB MMAP() 2/2

Page fault mechanism will ask the kernel to emit the I/O requests

- → No Posix read() or write() at the application level
- → I/O access are dealt internally by the kernel / page cache (readahead)

mmap() is nice for data scientists

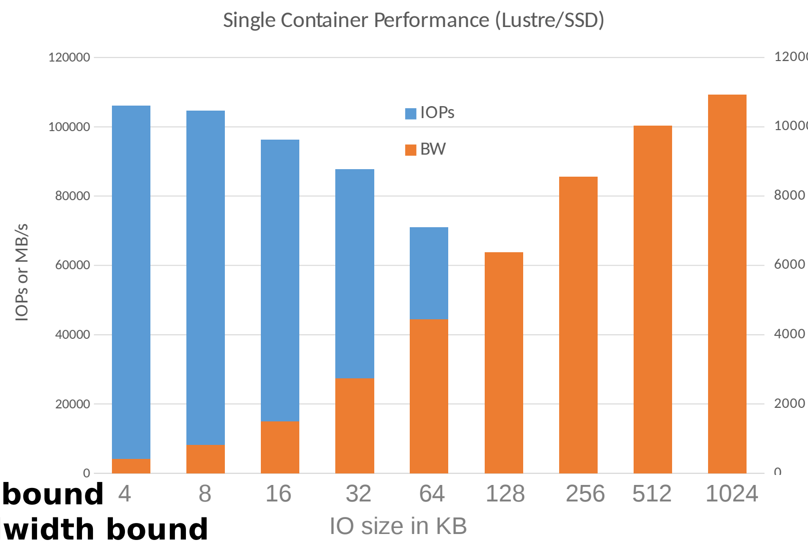
- Abstraction of the storage layer
- Unified API

mmap() is tough for computer scientists

- Kernel activity is more complicated to monitor than application
- Tracing tools (ftrace) have significant overhead
- Work with Julian and Eugen on this topic

CONTAINER PINNING OPTIMIZATION

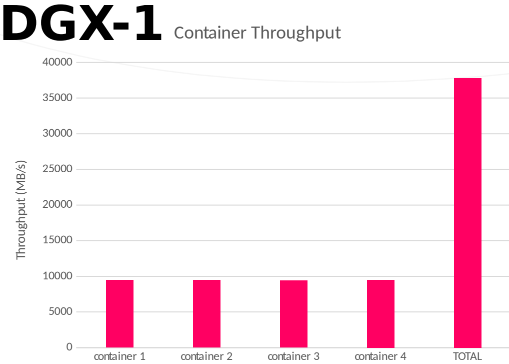
FROM IOPS TO BANDWIDTH

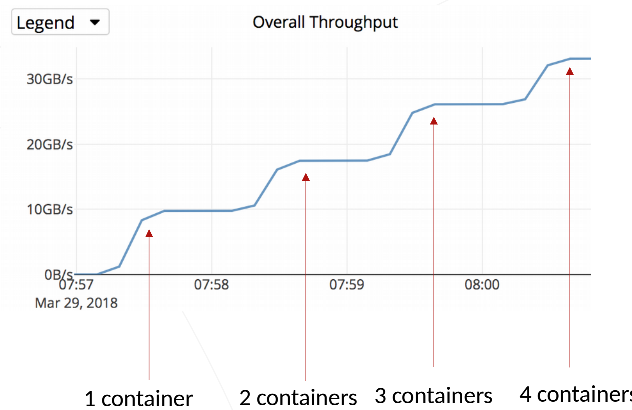


4KB random read is IOPS bound
1MB random read is Bandwidth bound

DGX CONTAINER THROUGHPUT

SCALING UP WORKLOADS IN DGX-1





EXASCALER ALL FLASH

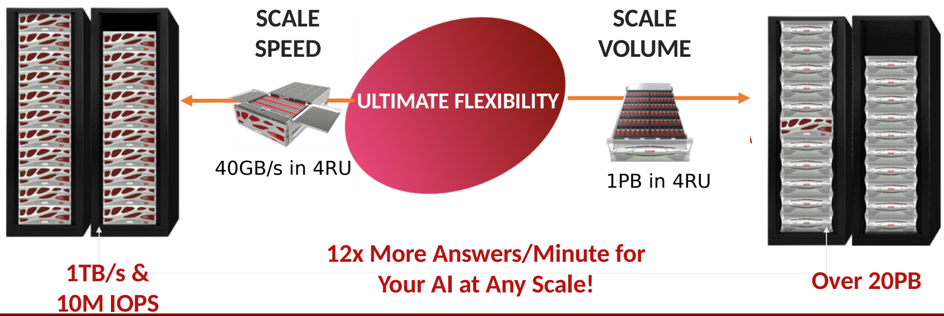
Remove I/O burden from data scientist shoulders

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

19

2018 DDN x12 Analytics

Bringing HPC technologies and know-how to analytics = x12



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