Lustre®

Lustre on Public Clouds

Opportunities, Challenges & Learnings

Michael Nishimoto, Chief Architect, michael@kmesh.io

#1 Lustre is complex!

Prepare for a Challenge

- NFS
- Panasas
- GPFS
- Lustre

Administrative Burden & Needed Expertise

(anecdotal)

http://www.linuxclustersinstitute.org/workshops/archive/21st/files/hpc_storage.pdf

Table 3. Overall ease of deployment

PVFS	Rating	Notes
Lustre		Lustre gets a one-and-a-half clouds rating for its comparatively complicated deployment—it required more nodes to be set up separately and specific installation packages. Our process was manual by design, because we wanted to see what it was like. An Azure Resource Manager template is also available to make deployment easier.
GlusterFS		We liked how straightforward GlusterFS was to deploy and gave it a three-cloud rating. There were fewer nodes to set up, and installation was simple.
BeeGFS		We found BeeGFS easy to deploy and gave it a three-cloud rating. We highly recommend using the Azure Resource Manager template.

https://azure.microsoft.com/mediahandler/files/resourcefiles/parallel-virtual-file-systems-on-microsoft-Azure/PVFS%20on%20Azure%20Guide.pdf

Lustre on Cloud is MORE Complex!

#2 ‘Infinite’ Cloud Resources

Multiple Clouds

Which cloud?

Azure

aws

Google Cloud Platform

Plethora of Instances

Which instance(s)?

Sizes for Linux virtual machines in Azure

11/13/2018 • 2 minutes to read • Contributors all

This article describes the available sizes and options for the Azure virtual machines you can use to run your Linux apps and workloads. It also provides deployment considerations to be aware of when you're planning to use these resources. This article is also available for Windows virtual machines.

Type	Sizes	Description
General purpose	B, Dsv3, Dv3, DSv2, Dv2, Av2, DC	Balanced CPU-to-memory ratio. Ideal for testing and development, small to medium databases, and low to medium traffic web servers.
Compute optimized	Fsv2, Fs, F	High CPU-to-memory ratio. Good for medium traffic web servers, network appliances, batch processes, and application servers.
Memory optimized	Esv3, Ev3, M, GS, G, DSv2, Dv2	High memory-to-CPU ratio. Great for relational database servers, medium to large caches, and in-memory analytics.
Storage optimized	Lsv2, Ls	High disk throughput and IO ideal for Big Data, SQL, NoSQL databases, data warehousing and large transactional databases.
GPU	NV, NVv2, NC, NCv2, NCv3, ND, NDv2 (Preview)	Specialized virtual machines targeted for heavy graphic rendering and video editing, as well as model training and inferencing (ND) with deep learning. Available with single or multiple GPUs.
High performance compute	H	Our fastest and most powerful CPU virtual machines with optional high-throughput network interfaces (RDMA).

Many Storage Options

Which storage(s)?

Block Blobs

Scalable object storage for documents, videos, pictures, and unstructured text or binary data. Choose from Hot, Cool, or Archive tiers.

Prices for locally redundant storage (LRS) Archive Block Blob start from:

$0.002/GB per month

See Pricing >

Azure Data Lake Storage

Combines the power of a Hadoop compatible file system with integrated hierarchical namespace with the massive scale and economy of Azure Blob Storage to help speed your transition from proof of concept to production.

Prices for LRS start from:

$0.001/GB per month

See Pricing >

Managed Disks

Persistent, secured disks that support simple and scalable virtual machine deployment. Designed for 99.999% availability. Choose Premium (SSD) Disks for low latency and high throughput.

Prices for Standard Managed Disks start from:

$1.54 per month

See Pricing >

Files

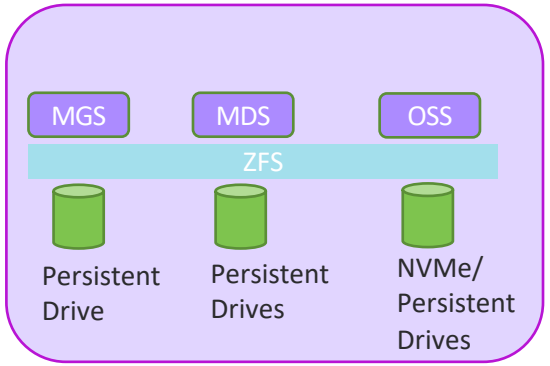
Fully managed file shares in the cloud, accessible via standard Server Message Block (SMB) protocol. Enables sharing files between applications using Windows APIs or REST API.

Prices for LRS File storage start from:

$0.060/GB per month

See Pricing >

Lustre Configuration



One Instance/VM

Lustre Version 2.11

ZFS version : 0.7.6

Persistent Drives: EBS(AWS)/Premium Disks (Azure)

NVMe Instances Comparison : IOR Tests

Specs are only guidance. Real performance matter!

Specification → Real Performance

Model	vCPU	Mem (GiB)	Networking Performance (Gbps)	Ephemeral	Dedicated EBS Bandwidth (Mbps)	$ Per Hour	$/GiB (Monthly)	RecordSize	Read MB/s	Write MB/s
i3.xlarge	4	30.5	Up to 10	1 x 0.95 NVMe SSD	850	$0.31	0.12	Lustre=128K ZFS=128K	**601**	**440**
i3.2xlarge	8	61	Up to 10	1 x 1.9 NVMe SSD	1,700	$0.62	0.24		**621**	**620**
c5d.4xlarge	16	32	Up to 10	1 x 400 NVMe SSD	3,500	$0.77	1.44		**837**	**387**
m5d.4xlarge	16	64	Up to 10	2 x 300 NVMe SSD	3,500	$0.90	1.13		**621**	**302**
i3.4xlarge	16	122	Up to 10	2 x 1.9 NVMe SSD	850	$1.25	0.24		**621**	**622**
i3.xlarge	4	30.5	Up to 10	1 x 0.95 NVMe SSD	850	$0.31	0.12	Lustre=1MBK ZFS=128K	**597**	**438**
i3.2xlarge	8	61	Up to 10	1 x 1.9 NVMe SSD	1,700	$0.62	0.24		**621**	**620**
c5d.4xlarge	16	32	Up to 10	1 x 400 NVMe SSD	3,500	$0.77	1.44		**840**	**387**
m5d.4xlarge	16	64	Up to 10	2 x 300 NVMe SSD	3,500	$0.90	1.13		**621**	**303**
i3.4xlarge	16	122	Up to 10	2 x 1.9 NVMe SSD	850	$1.25	0.24		**621**	**622**

# of slots = 8

- Instance cost != better IOR performance
- Write performance depends on the storage size
- Larger ZFS transfer size is better for IOR BW. Optimal ZFS transfer = 128k

NVMe Instances Comparison : MDTEST

Model	vCPU	Memory (GiB)	$/Hour	Directory creation	Directory stat	Directory removal	File creation	File stat	File read	File removal	Tree creation	Tree removal	CPU Utilization
i3.xlarge	4	30.5	$ 0.31	4,827.42	17,709.11	6,363.59	6,660.80	9,780.99	9,267.85	9,737.73	2,276.53	2,601.46	77.10%
i3.2xlarge	8	61	$ 0.62	5,876.86	19,147.13	8,928.47	8,168.89	10,642.64	10,460.31	11,516.76	2,412.13	2,570.56	67.70%
c5d.4xlarge	**16**	**32**	**$ 0.77**	**4,034.69**	**30,153.35**	**14,768.59**	**13,628.94**	**16,969.22**	**17,973.16**	**18,410.46**	**4,152.59**	**4,449.19**	**19.60%**
m5d.4xlarge	16	64	$ 0.90	9,861.89	25,202.48	12,248.51	11,300.76	13,988.44	13,963.48	15,866.89	3,431.61	3,821.88	32.80%
i3.4xlarge	16	122	$ 1.25	6,990.27	19,614.65	8,810.93	8,400.30	10,692.49	10,716.49	11,489.88	2,530.25	2,685.22	20.40%

Command Used: mpiexec <host> /usr/bin/mdtest -I 10 -i 2 -z5 -b 5 -i 3 -d /mnt/kmesh/mdtest5/

Cross-cloud Comparison : Instances with NVMe

VDBench Comparison

		i3.2xlarge (AWS)			L16s_v2 (Azure)			%Diff		
		8 vCPU	61GiB	1x1900NVMe SSD	16vCPU	128Gib	2x1900GiB NVMe SSD			
		$.624/hr			$1.248/hr					
	Transfer Size (Bytes)	Total IOPS	Total MiB/s	Latency (MS)	Total IOPS	Total MiB/s	Latency (MS)	IOPS	MiB/s	Latency
100% Read	4K	49,024.80	191.50	1.32	36,650.50	238.30	2.59	-25%	24%	196%
0% Read	4K	11,129.80	43.47	5.79	8,483.60	54.04	10.20	-24%	24%	176%
100% Read	32K	34,292.60	1,071.64	1.89	18,147.50	958.76	5.68	-47%	-11%	300%
0% Read	32K	19,004.20	593.89	3.34	12,367.00	646.21	7.87	-35%	9%	235%
100% Read	128K	16,148.70	2,018.58	4.11	9,242.80	1,950.44	11.09	-43%	-3%	270%
0% Read	128K	5,862.70	732.84	10.77	3,745.60	770.21	24.02	-36%	5%	223%
100% Read	256K	9,263.70	2,315.92	6.94	5,607.70	2,333.38	16.98	-39%	1%	245%
0% Read	256K	4,383.10	1,095.79	14.30	2,152.30	874.68	39.62	-51%	-20%	277%

Cross-cloud Comparison : Instances with Persistent Drives

VDBench comparison

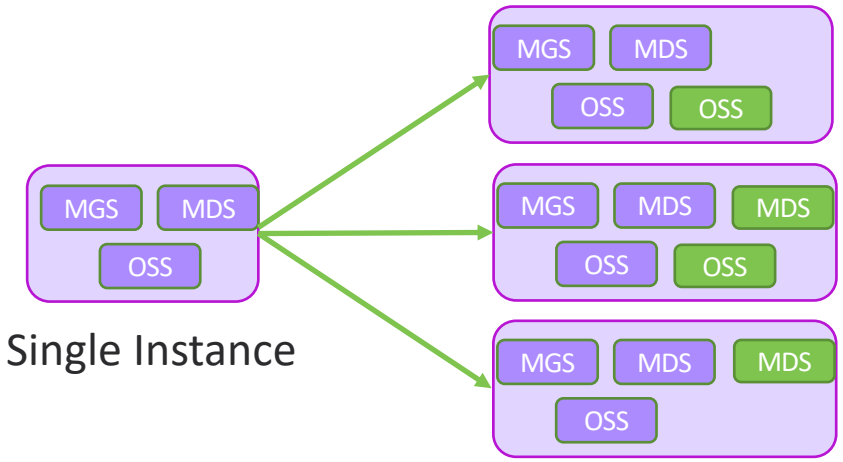
			EBS:c5.4xlarge (AWS) 16 vCPU 32 GiB Memory $0.680/hour			PD:Standard F16s(Azure) 16 vCPU 32 GiB Memory $0.796/hour			Diff %		
R/W Ratio	Seek %	Transfer Size (Bytes)	Total IOPS	Total MiB/s	Latency (MS)	Total IOPS	Total MiB/s	Latency (MS)	Total IOPS	Total MiB/s	Latency (MS)
100% Read	Random	4K	7385.4	28.85	4.327	3035.5	11.86	10.521	-59%	-59%	243%
100% Read	50	4K	5298.3	20.7	6.034	2158.1	8.43	14.812	-59%	-59%	245%
100% Read	Sequential	4K	83160.3	324.84	0.384	24244.1	94.7	1.323	-71%	-71%	345%
75% Read	Random	4K	6303.8	24.62	5.07	2728.1	10.66	11.709	-57%	-57%	231%
75% Read	50	4K	5666.8	22.14	5.64	2404.6	9.39	13.288	-58%	-58%	236%
75% Read	Sequential	4K	55336.6	216.16	0.577	36259	141.64	0.88	-34%	-34%	153%
50% Read	Random	4K	5642.1	22.04	5.664	2595.7	10.14	12.308	-54%	-54%	217%
50% Read	50	4K	6312.8	24.66	5.062	2735.6	10.69	11.678	-57%	-57%	231%
50% Read	Sequential	4K	51685.5	201.9	0.617	19642.7	76.73	1.622	-62%	-62%	263%
0% Read	Random	4K	3717.5	14.52	8.599	2162.4	8.45	14.776	-42%	-42%	172%
0% Read	50	4K	6757.7	26.4	4.727	3232.7	12.63	9.876	-52%	-52%	209%
0% Read	Sequential	4K	23989.9	93.71	1.329	9932.1	38.8	3.213	-59%	-59%	242%
100% Read	Random	32K	7186.6	224.58	4.447	2146.5	67.08	14.89	-70%	-70%	335%
100% Read	50	32K	3748.7	117.15	8.531	1374.6	42.96	23.262	-63%	-63%	273%
100% Read	Sequential	32K	12354.7	386.08	2.589	2900.2	90.63	11.018	-77%	-77%	426%
75% Read	Random	32K	7787.8	243.37	4.1	2805.5	87.67	11.385	-64%	-64%	278%
75% Read	50	32K	3602.3	112.57	8.873	1470	45.94	21.737	-59%	-59%	245%
75% Read	Sequential	32K	10752.1	336	2.972	6563.9	205.12	4.869	-39%	-39%	164%
50% Read	Random	32K	7074.3	221.07	4.512	3615.4	112.98	8.826	-49%	-49%	196%
50% Read	50	32K	3991.5	124.73	8.005	1677.3	52.41	19.052	-58%	-58%	238%
50% Read	Sequential	32K	10928.2	341.51	2.922	5197.5	162.42	6.149	-52%	-52%	210%
0% Read	Random	32K	5505.1	172.03	5.796	3138.7	98.09	10.164	-43%	-43%	175%
0% Read	50	32K	7130.7	222.84	4.471	3852.4	120.39	8.276	-46%	-46%	185%
0% Read	Sequential	32K	13303.6	415.74	2.393	7543.9	235.75	4.227	-43%	-43%	177%
100% Read	Random	128K	2570.1	321.26	12.445	1297	162.12	24.654	-50%	-50%	198%
100% Read	50	128K	1793.4	224.18	17.837	949.3	118.66	33.692	-47%	-47%	189%
100% Read	Sequential	128K	3300.5	412.56	9.693	1744.8	218.1	18.33	-47%	-47%	189%
75% Read	Random	128K	2801.3	350.16	11.405	1593.4	199.18	20.053	-43%	-43%	176%
75% Read	50	128K	2132.3	266.54	14.989	1124.9	140.62	28.418	-47%	-47%	190%
75% Read	Sequential	128K	3282.9	410.36	9.733	1915.4	239.42	16.687	-42%	-42%	171%
50% Read	Random	128K	2536.5	317.06	12.589	1484.9	185.62	21.512	-41%	-41%	171%
50% Read	50	128K	2187	273.38	14.603	1241.7	155.21	25.735	-43%	-43%	176%
50% Read	Sequential	128K	3287.8	410.98	9.711	1961.1	245.13	16.289	-40%	-40%	168%
0% Read	Random	128K	2251.4	281.42	14.173	1100.3	137.54	29.032	-51%	-51%	205%
0% Read	50	128K	2588.8	323.6	12.322	1224.9	153.11	26.074	-53%	-53%	212%
0% Read	Sequential	128K	3336.8	417.1	9.553	1591.5	198.94	20.061	-52%	-52%	210%

CPU utilization matters

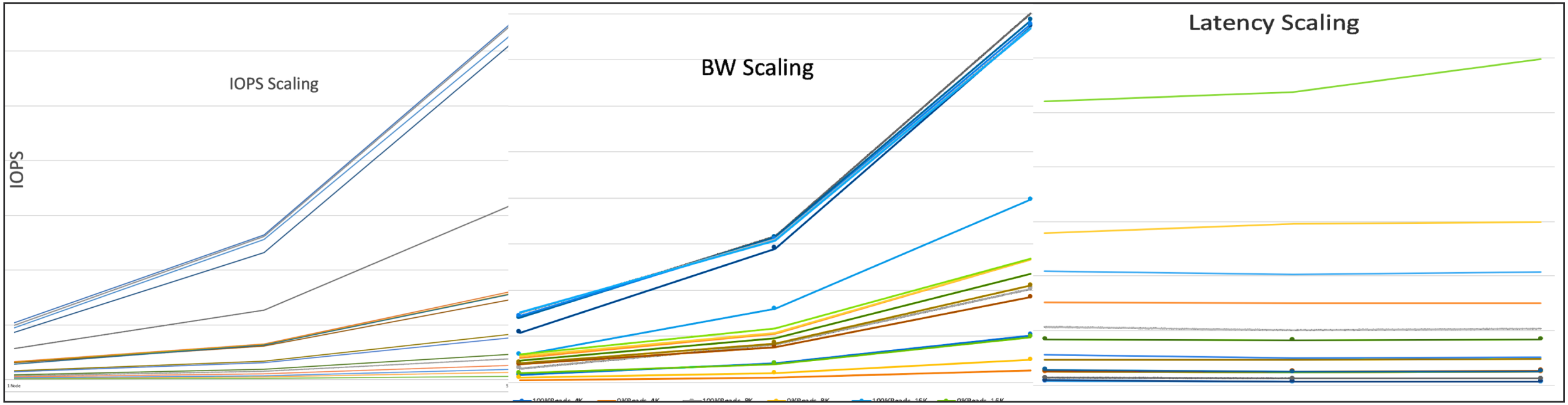
% CPU Idle c5.4xlarge (AWS)	% CPU Idle F16s (Azure)	CPU Idle Relative Diff
79.73	89.08	9.35
73.29	80.72	7.43
72.69	80.50	7.81
76.25	80.81	4.56
79.77	88.31	8.54
73.13	81.26	8.13
72.62	79.29	6.67
82.54	83.14	0.60
79.33	86.89	7.56
72.73	79.59	6.86
71.68	78.13	6.45
83.85	83.49	-0.36

Scaleout

Scaleout



- ✓ Single component per instance
  - Needs a balanced instance to host both OSS and MDS
- ✓ Multiple components per instance
  - Pick diff. instances for OSS and MDS



IO500 Benchmark

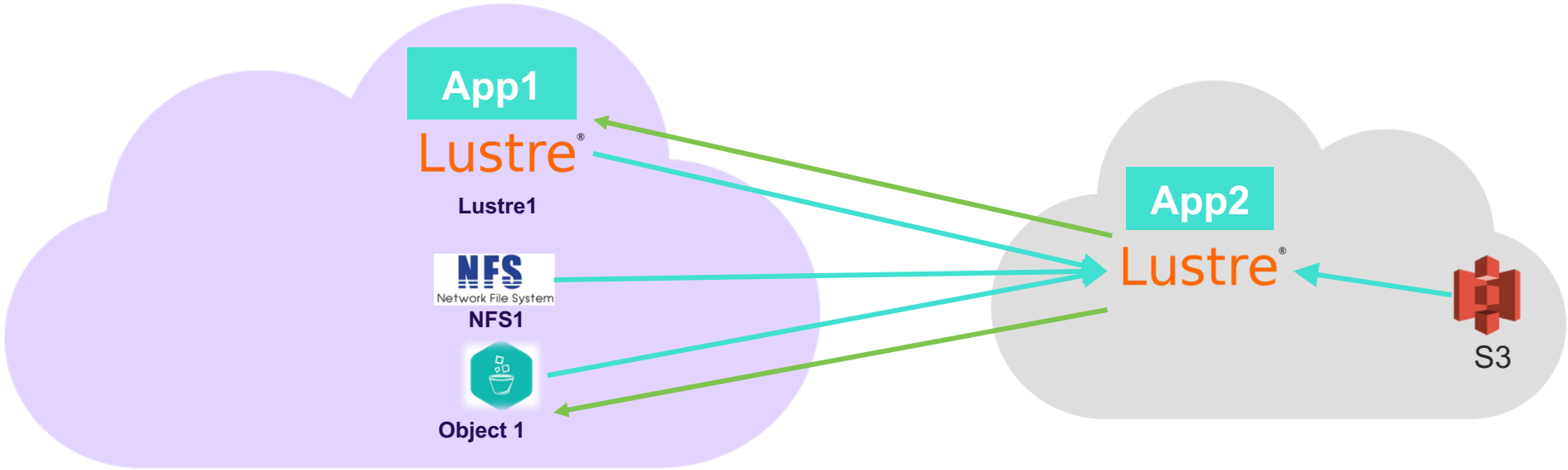
```
[Leaving] datafiles in /mnt/kmesh/io-500-new/io-500-dev/datafiles/io500.2019.05.14-20.08.15
[Summary] Results files in /mnt/kmesh/io-500-new/io-500-dev/results/2019.05.14-20.08.15
[Summary] Data files in /mnt/kmesh/io-500-new/io-500-dev/datafiles/io500.2019.05.14-20.08.15
[RESULT] BW   phase 1            ior_easy_write              2.318 GB/s : time 303.34 seconds
[RESULT] BW   phase 2            ior_hard_write              0.260 GB/s : time 339.07 seconds
[RESULT] BW   phase 3             ior_easy_read              2.323 GB/s : time 302.70 seconds
[RESULT] BW   phase 4             ior_hard_read              0.443 GB/s : time 198.91 seconds
[RESULT] IOPS phase 1         mdtest_easy_write              6.769 kiops : time 301.54 seconds
[RESULT] IOPS phase 2         mdtest_hard_write              1.795 kiops : time 300.90 seconds
[RESULT] IOPS phase 3                      find             76.890 kiops : time  33.54 seconds
[RESULT] IOPS phase 4          mdtest_easy_stat              8.328 kiops : time 245.16 seconds
[RESULT] IOPS phase 5          mdtest_hard_stat              3.218 kiops : time 168.01 seconds
[RESULT] IOPS phase 6        mdtest_easy_delete              5.536 kiops : time 368.95 seconds
[RESULT] IOPS phase 7          mdtest_hard_read              4.346 kiops : time 124.46 seconds
[RESULT] IOPS phase 8        mdtest_hard_delete              1.893 kiops : time 285.44 seconds
[SCORE] Bandwidth 0.887473 GB/s : IOPS 5.71656 kiops : TOTAL 2.2524
```

MDT (1) : i3.4xlarge (NVMe)
OSTs(4) OST0, OST1, OST2, OST3 : i3.2xlarge(NVMe)
Client : c5n.9xlarge (1)
LFS block & stripe size: 1MB

Kmesh	AWS Cloud	Kmesh	Lustre	4	4	0.887	45.71	2.25

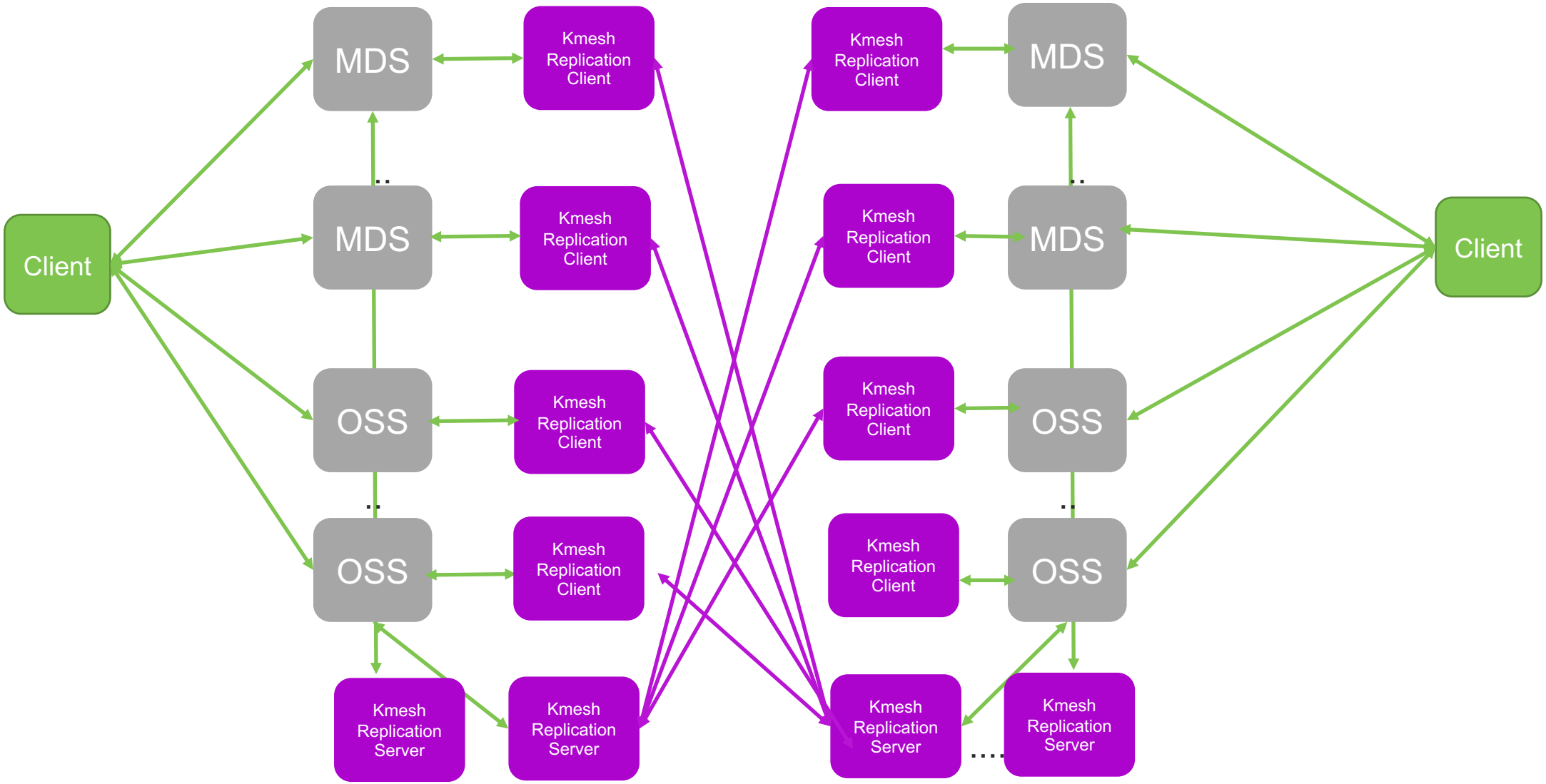
#	information							io500		
	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw	md
									GiB/s	kIOP/s
1	Oak Ridge National Laboratory	Summit	IBM	Spectrum Scale	504	1008	zip	330.56	88.20	1238.93
2	University of Cambridge	Data Accelerator	Dell EMC	Lustre	528	4224	zip	158.71	71.40	352.75
3	Korea Institute of Science and Technology Information (KISTI)	NURION	DDN	IME	2048	4096	zip	156.91	554.23	44.43
4	JCAHPC	Oakforest-PACS	DDN	IME	2048	16384	zip	137.78	560.10	33.89
5	WekaIO	WekaIO	WekaIO		17	935	zip	78.37	37.39	164.26
6	KAUST	ShaheenII	Cray	DataWarp	1024	8192	zip	77.37	496.81	12.05
7	University of Cambridge	Data Accelerator	Dell EMC	BeeGFS	184	5888	zip	74.58	58.81	94.57
8	Google	Exascaler on GCP	Google	Lustre	120	960	zip	47.23	23.06	96.74
9	JCAHPC	Oakforest-PACS	DDN	Lustre	256	8192	zip	42.18	20.04	88.78
10	KAUST	ShaheenII	Cray	Lustre	1000	16000		41.00*	54.17	31.03*
11	JSC	JURON	ThinkparQ	BeeGFS	8	64		35.77*	14.24	89.81*
12	DKRZ	Mistral	Seagate	Lustre	100	1000		32.15	22.77	45.39
13	DDN	Bancholab	DDN	Lustre	10	240	zip	31.50	6.33	156.69
14	IBM	Sonasad	IBM	Spectrum Scale	10	10	zip	24.24	4.57	128.61
15	Clemson University	ofsdev	Dell	BeeGFS	32	32	zip	22.02	8.55	56.68
16	Fraunhofer	Seislab	ThinkparQ	BeeGFS	24	24		16.96	5.13	56.14
17	Joint Institute for Nuclear Research	Govorun	RSC	Lustre	24	192	zip	12.08	3.34	43.65
18	PNNL	EMSL Cascade		Lustre	126	252		11.12	4.88	25.33
19	Queen Mary; University Of London	Apocrita	E8	GPFS	10	240	zip	9.65	4.32	21.55
20	Clemson University	Palmetto	Dell	BeeGFS	48	48	zip	7.64	2.93	19.88
#	information							io500		
	institution	system	storage vendor	filesystem type	client nodes	client total procs	data	score	bw	md
									GiB/s	kIOP/s
21	CERN	Bytecollider		CephFS	64	64	zip	7.56	2.83	20.16
22	SNL	Serrano	IBM	Spectrum Scale	16	160		4.25*	0.65	27.98*
23	STFC	Jasmin/Lotus	Purestorage	NFS	64	128	zip	2.33	0.26	20.93
24	Clemson University	Palmetto	Dell	OrangeFS	32	32	zip	2.31	1.93	2.77
25	Nemours	nas6	DDN	GPFS	1	2	zip	2.06	1.39	3.04
26	Nemours	nas6	DDN	GPFS	1	2	zip	1.52	0.74	3.11

#3 Data Synchronization



- One-time copy
- Continuous synchronization
- Real-Time synchronization
- To cloud, from cloud
- Variety of sources

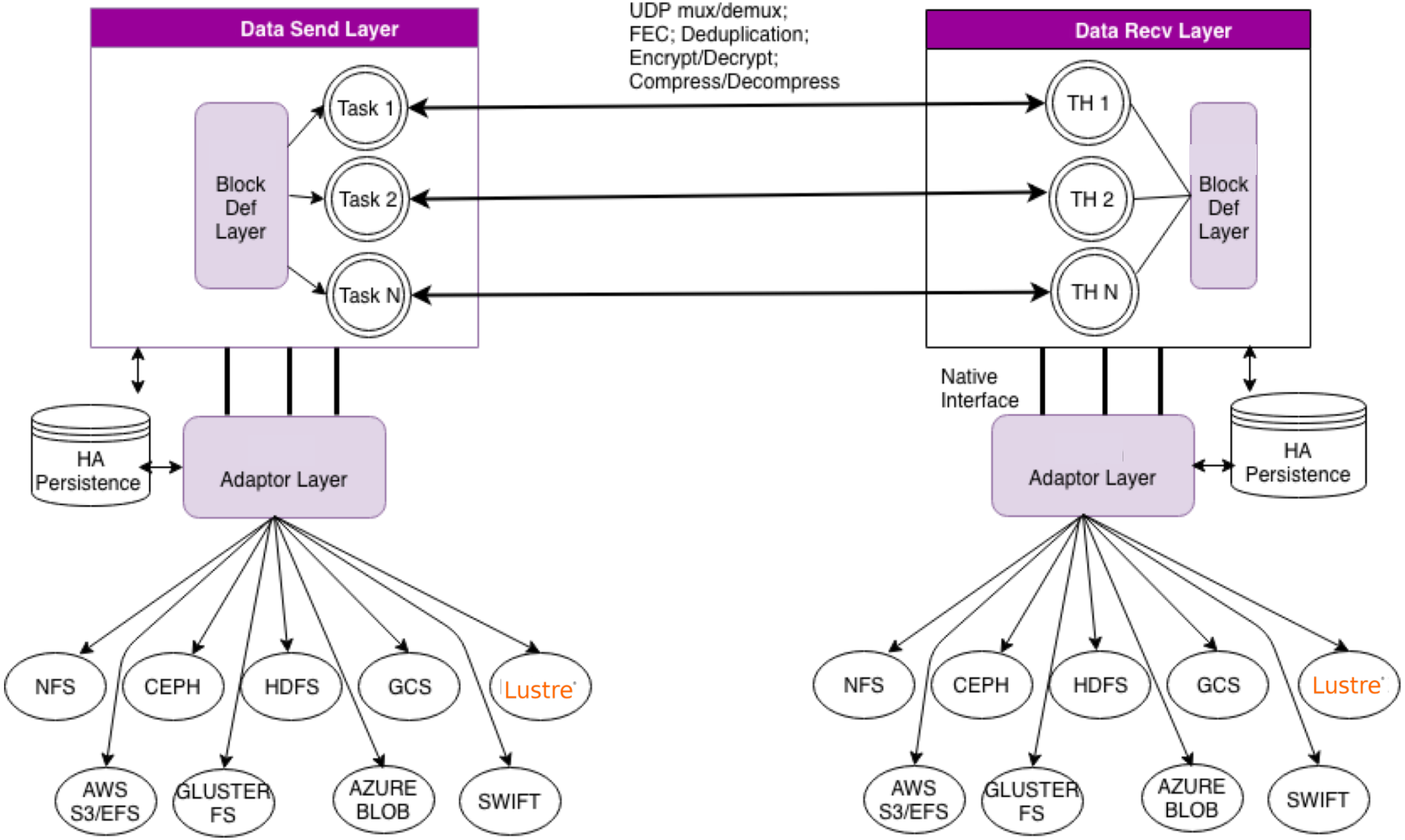
Lustre to Lustre Realtime Synchronization



Lustre to Lustre Synchronization

- Description
  - Lustre filesystem to Lustre filesystem data synchronization
  - File level metadata/data events are from one Lustre cluster to another Lustre cluster
- Mechanism
  - All the events are protected by LDLM lock when moved from one cluster to another cluster
  - Destination cluster does parallel read on the replicated data - application performs better as the destination replicated data is co-located to the application.
  - In asynchronous case, sequence generated from each of the cluster node, are ordered at the destination cluster and written to the destination cluster. Sequence generation is protected by LDLM lock
  - Replication manager on each cluster node to replicate data to different destination clusters
  - Replication manager performs replication across different clusters based on filtering and control policies applied
  - Replication manager, client, server are added as OBD devices layered in MDT/OST stacks

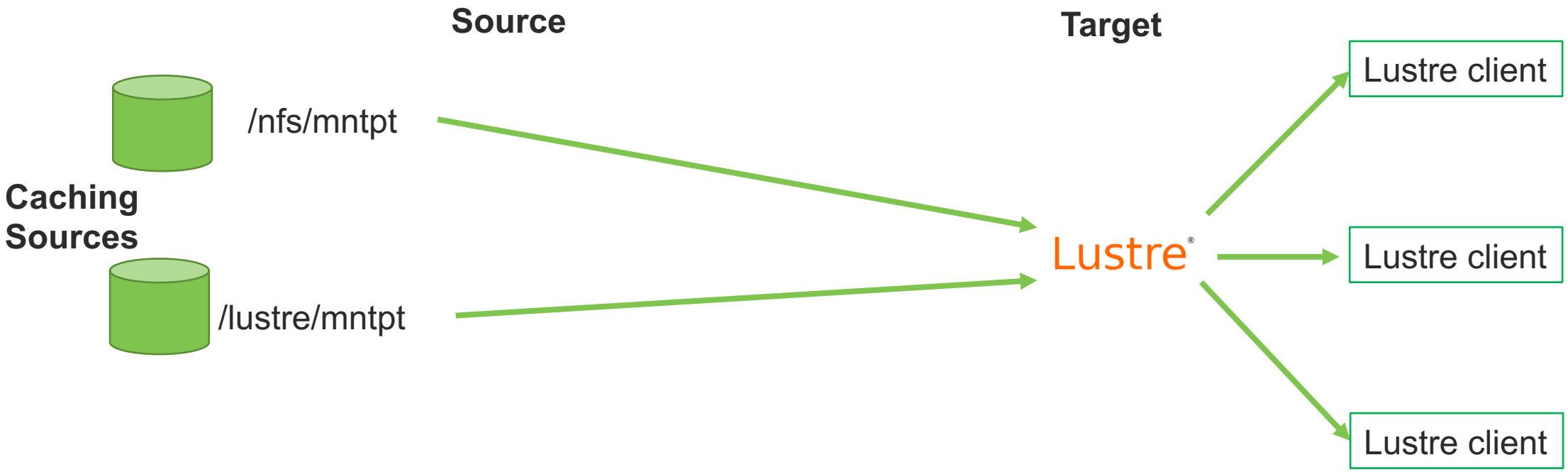
‘Any to Lustre Data Synchronization’ Architecture



‘Any data source to Lustre’ Synchronization

- Description
  - Continuous data synchronization service that works between multiple data sources
  - Unifies various Data sources as single hierarchical file system.
  - Support for File (Lustre,NFS, Ceph, HDFS) and Object sources(S3, Azure Blob and Swift)
- Mechanism
  - Each data source is linked to the root of the file system and are identified by path names
  - Each file from a data source are chunked and forwarded to the destination based on chunk delta match.
  - Deduplication is done at chunk level. Each chunk checksum is computed and compared with the previous checksum.
  - Task manager schedules job in parallel and sends only the modified data to the destination.
  - Each sync task data is sent as multiple streams between the endpoints over UDP. Compression and FEC are done to improve network performance

Caching Data from Lustre FS or NFS



Use cases

→ Smaller target cloud storage capacity

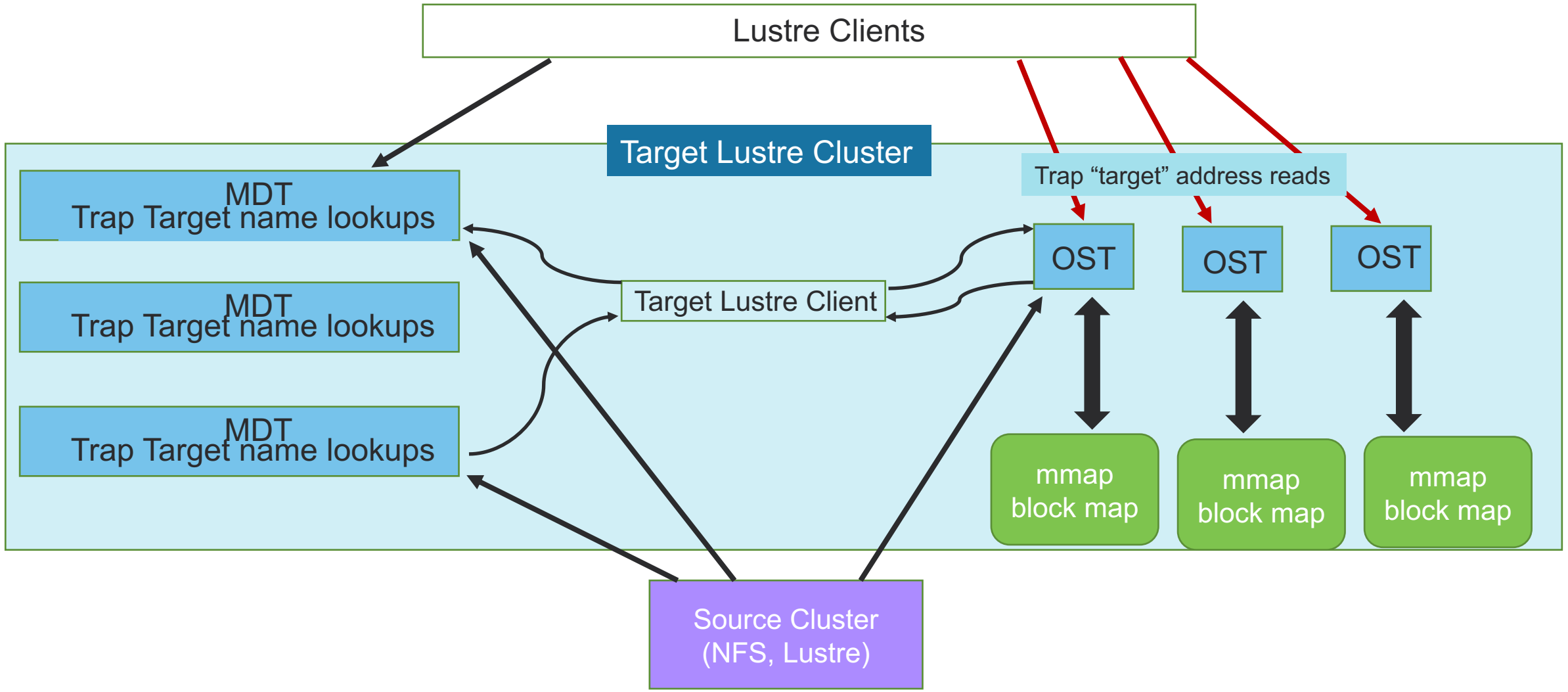
→ Faster App bring-up

→ Sensitive data (no permanent copy on cloud)

Features

- Manage object regions by hotness
- Perform intelligent pre-loading
- Interoperate with data synchronization

Caching Data from Lustre FS or NFS



Lustre Caching

- **Architecture**
  - Pull objects from source Lustre cluster or NFS server to target Lustre cluster
  - Data source is a mount point, so Lustre or NFS works
- **Metadata**
  - Trap file name miss lookup and pull in if the file exists on source
  - Trap directory name miss lookup, pull in if it exists on source, and pull in readdir contents
- **Data**
  - Trap address read range in “target”
  - Check address read ranges against new blockmap to determine if data must be read from source
  - Revalidate on access after per-object timeout by checking source time stamps
- **Caveats**
  - First release requires Lustre clients to mount the target Lustre FS as readonly
- **Future**
  - Eviction – needed if target cluster is smaller than source cluster
  - Intelligent pre-load

Other things to consider for cloud

- Optimize cost
  - ZFS Deduplication
  - ZFS Compression
  - HSM
- Data Management
  - Snapshots/Space efficient-clones

Deduplication performance

			No Dedupe			Deduplication 2:1			% Impact		
	IO Distibution	Transfer Size (Bytes)	Total IOPS	Total MiB/s	Latency (MS)	Total IOPS	Total MiB/s	Latency (MS)	IOPS	MiB/s	Latency
100% Read	Random	4K	16800	66	1.9	10,687	42	3.0	-36%	-36%	57%
100% Read	50%	4K	11555	45	2.8	5,594	22	5.7	-52%	-52%	107%
100% Read	Sequential	4K	285694	1116	0.1	150,908	589	0.2	-47%	-47%	91%
75% Read	Random	4K	8852	35	3.6	5,516	22	5.8	-38%	-38%	60%
75% Read	50%	4K	8688	34	3.7	4,968	19	6.4	-43%	-43%	75%
75% Read	Sequential	4K	99145	387	0.3	31,613	123	1.0	-68%	-68%	213%
50% Read	Random	4K	5657	22	5.7	4,030	16	7.9	-29%	-29%	40%
50% Read	50%	4K	7308	29	4.4	4,918	19	6.5	-33%	-33%	48%
50% Read	Sequential	4K	70339	275	0.5	30,572	119	1.0	-57%	-57%	128%
0% Read	Random	4K	3122	12	10.2	2,230	11	12.9	-29%	-13%	26%
0% Read	50%	4K	5163	20	6.2	4,268	18	8.0	-17%	-13%	30%
0% Read	Sequential	4K	27473	107	1.2	23,016	90	1.4	-16%	-16%	17%
100% Read	Random	32K	13833	432	2.3	9,537	298	3.4	-31%	-31%	45%
100% Read	50%	32K	8974	280	3.6	4,569	143	7.0	-49%	-49%	97%
100% Read	Sequential	32K	31196	975	1.0	16,338	511	2.0	-48%	-48%	91%
75% Read	Random	32K	7303	228	4.4	5,555	174	5.7	-24%	-24%	31%
75% Read	50%	32K	6237	195	5.1	3,742	117	8.5	-40%	-40%	67%
75% Read	Sequential	32K	20405	638	1.6	10,819	338	2.9	-47%	-47%	88%
50% Read	Random	32K	5527	173	5.8	5,012	157	6.4	-9%	-9%	10%
50% Read	50%	32K	5364	168	6.0	3,465	108	9.2	-35%	-35%	55%
50% Read	Sequential	32K	15179	474	2.1	9,283	290	3.4	-39%	-39%	63%
0% Read	Random	32K	7132	223	4.5	4,677	146	6.8	-34%	-34%	52%
0% Read	50%	32K	7777	243	4.1	4,988	156	6.4	-36%	-36%	56%
0% Read	Sequential	32K	14969	468	2.1	8,897	278	3.6	-41%	-41%	68%
100% Read	Random	128K	7252	907	4.4	3,727	466	8.6	-49%	-49%	95%
100% Read	50%	128K	5443	680	5.9	2,565	321	12.5	-53%	-53%	112%
100% Read	Sequential	128K	8008	1001	4.0	4,197	525	7.6	-48%	-48%	91%
75% Read	Random	128K	4248	531	7.5	2,684	336	11.9	-37%	-37%	58%
75% Read	50%	128K	3777	472	8.5	2,070	259	15.4	-45%	-45%	83%
75% Read	Sequential	128K	5805	726	5.5	3,470	434	9.2	-40%	-40%	67%
50% Read	Random	128K	2871	359	11.1	2,309	289	13.8	-20%	-20%	24%
50% Read	50%	128K	2753	344	11.6	2,078	260	15.4	-25%	-25%	32%
50% Read	Sequential	128K	5334	667	6.0	3,210	401	9.9	-40%	-40%	66%
0% Read	Random	128K	2859	357	11.1	1,484	186	21.5	-48%	-48%	93%
0% Read	50%	128K	2774	347	11.5	1,628	204	19.6	-41%	-41%	71%
0% Read	Sequential	128K	4480	560	7.1	2,492	311	12.8	-44%	-44%	80%

IOPS, BW : 9-50% drop
Latency: 24-213% increase

VDBench Testing

Instance:i3.2xlarge
Used Space: 512GiB

Compression performance

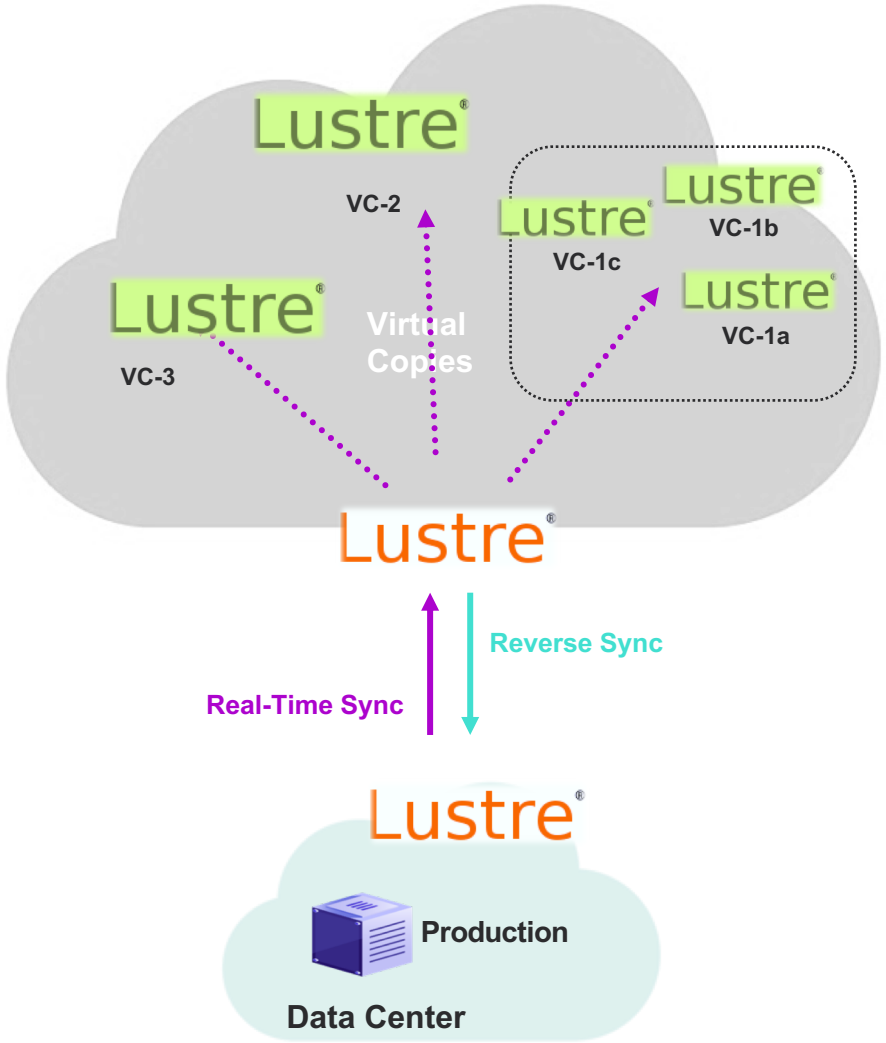
			No Compression			Compression (~50%)			% Impact		
	IO Distibution	Transfer Size (Bytes)	Total IOPS	Total MiB/s	Latency (MS)	Total IOPS	Total MiB/s	Latency (MS)	IOPS	MiB/s	Latency
100% Read	Random	4K	16800	66	1.9	16243	63	2.0	-3%	-3%	3%
100% Read	50%	4K	11555	45	2.8	11746	46	2.7	2%	2%	-2%
100% Read	Sequential	4K	285694	1116	0.1	262296	1025	0.1	-8%	-8%	9%
75% Read	Random	4K	8852	35	3.6	10526	41	3.0	19%	19%	-16%
75% Read	50%	4K	8688	34	3.7	10247	40	3.1	18%	18%	-15%
75% Read	Sequential	4K	99145	387	0.3	111401	435	0.3	12%	12%	-11%
50% Read	Random	4K	5657	22	5.7	7388	29	4.3	31%	31%	-23%
50% Read	50%	4K	7308	29	4.4	9053	35	3.5	24%	24%	-19%
50% Read	Sequential	4K	70339	275	0.5	78557	307	0.4	12%	12%	-11%
0% Read	Random	4K	3122	12	10.2	4708	18	6.8	51%	51%	-34%
0% Read	50%	4K	5163	20	6.2	7684	30	4.2	49%	49%	-33%
0% Read	Sequential	4K	27473	107	1.2	30321	118	1.1	10%	10%	-9%
100% Read	Random	32K	13833	432	2.3	13682	428	2.3	-1%	-1%	1%
100% Read	50%	32K	8974	280	3.6	8971	280	3.6	0%	0%	0%
100% Read	Sequential	32K	31196	975	1.0	29148	911	1.1	-7%	-7%	7%
75% Read	Random	32K	7303	228	4.4	9093	284	3.5	25%	25%	-20%
75% Read	50%	32K	6237	195	5.1	7195	225	4.4	15%	15%	-13%
75% Read	Sequential	32K	20405	638	1.6	22285	696	1.4	9%	9%	-8%
50% Read	Random	32K	5527	173	5.8	6814	213	4.7	23%	23%	-19%
50% Read	50%	32K	5364	168	6.0	6305	197	5.1	18%	18%	-15%
50% Read	Sequential	32K	15179	474	2.1	18345	573	1.7	21%	21%	-17%
0% Read	Random	32K	7132	223	4.5	5901	184	5.4	-17%	-17%	21%
0% Read	50%	32K	7777	243	4.1	7585	237	4.2	-2%	-2%	3%
0% Read	Sequential	32K	14969	468	2.1	15092	472	2.1	1%	1%	-1%
100% Read	Random	128K	7252	907	4.4	7008	876	4.6	-3%	-3%	3%
100% Read	50%	128K	5443	680	5.9	5394	674	5.9	-1%	-1%	1%
100% Read	Sequential	128K	8008	1001	4.0	7699	962	4.2	-4%	-4%	4%
75% Read	Random	128K	4248	531	7.5	5075	634	6.3	19%	19%	-16%
75% Read	50%	128K	3777	472	8.5	4364	546	7.3	16%	16%	-13%
75% Read	Sequential	128K	5805	726	5.5	6620	828	4.8	14%	14%	-12%
50% Read	Random	128K	2871	359	11.1	3374	422	9.5	18%	18%	-15%
50% Read	50%	128K	2753	344	11.6	3297	412	9.7	20%	20%	-17%
50% Read	Sequential	128K	5334	667	6.0	5533	692	5.8	4%	4%	-4%
0% Read	Random	128K	2859	357	11.1	3366	421	9.5	18%	18%	-15%
0% Read	50%	128K	2774	347	11.5	4218	527	7.5	52%	52%	-34%
0% Read	Sequential	128K	4480	560	7.1	5300	662	6.0	18%	18%	-15%

-> Negligible performance impact

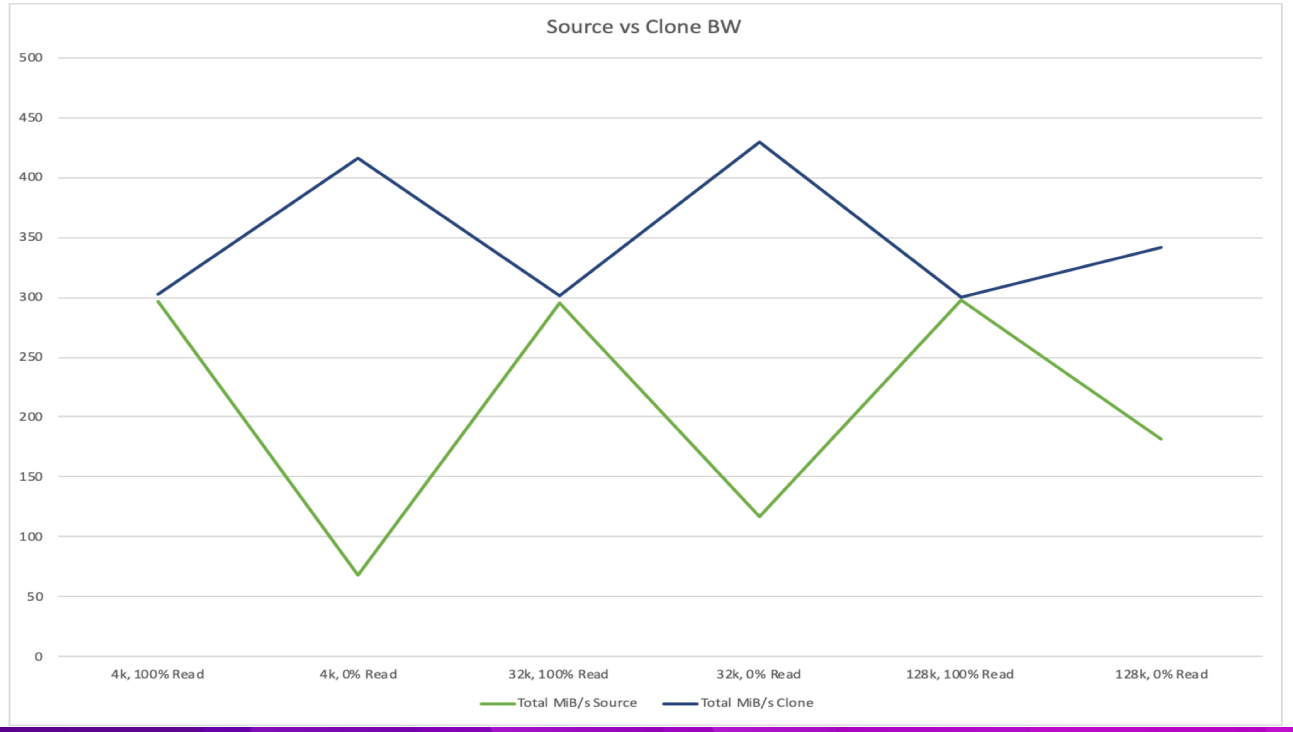
VDBrench Testing

Instance:i3.2xlarge
Used Space: 512GiB
Compression Algorithm: ZFS LZ4

Space Efficient Clones



1. Create snapshot
2. Create zfs clone from the above created snapshot
3. Set the right attributes
4. Rename the cloned fs to new name
5. Mount the mdt/ost clones created



Summary

1. VM/Instance specs are a starting point. Do your validation.
2. Use price/performance metric to decide instance, storage and cloud
3. ‘Cloud benchmarking’ is needed. IO500 board has been engaged.
4. Selecting a data orchestration strategy is key for hybrid HPC
5. Use Lustre features when needed
   - ZFS LZ4 compression can be enabled default
   - Dedupe can work in some scenarios
   - HSM is a great way to reduce the cost
   - Cloning performance is great for read-heavy workloads