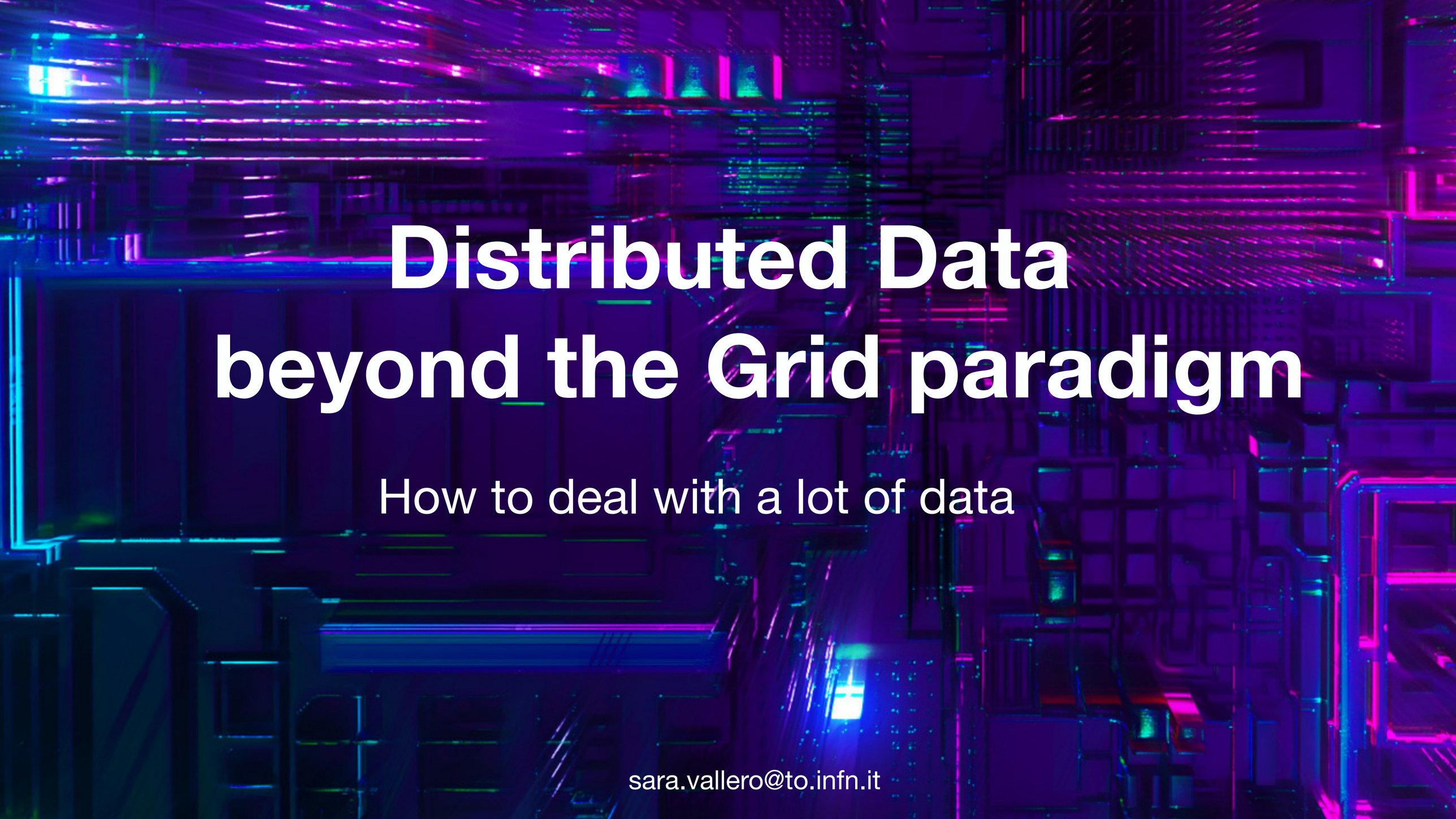


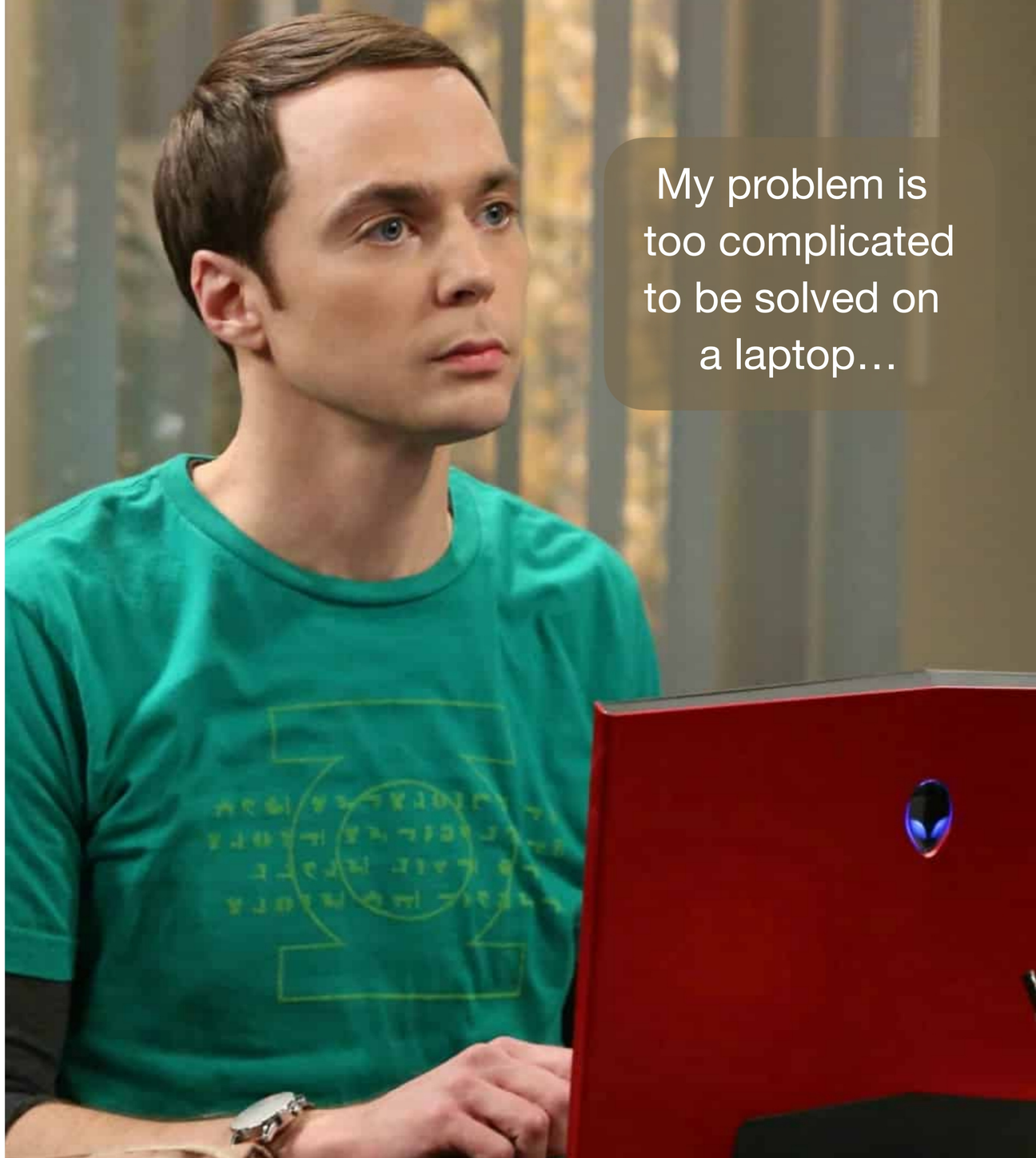
Distributed Data beyond the Grid paradigm

How to deal with a lot of data

sara.vallero@to.infn.it

Distributed computing

- The problem is not necessarily computing intensive
- But **a lot of data** needs to be processed
- A lot means: more than I can fit on a laptop or workstation



Distributed filesystems

- A Distributed Filesystem (DFS) is a physically distributed implementation of the classical time-sharing model of the traditional filesystem.
- It resides on different machines and/or sites.
- Offers a unified logical view of data, whether local or remote.

Characteristics

TRANSPARENCY:

- Be able to perform the same operations as on a local filesystem
- Files must appear as they were put on a single location
- The complexity of the underlying system must be hidden to users

FAULT TOLERANCE:

- System availability despite network or server failures
- Data integrity and consistency when several users try to access them simultaneously

SCALABILITY:

- Efficiently leverage the dynamic addition of new servers
    - Centralised systems = single server = bottleneck (unless multithreading and/or caching)

Architecture

- **Client-server:** several servers store and share metadata and data between multiple clients (global namespace)
  - **Decentralised**: many servers. No specific server for metadata.
- **Cluster-based:** data and metadata are decoupled
  - **Centralised**: only 1 metadata server
  - Totally **distributed**: distributed metadata servers

Some common DFS



	HDFS	Ceph	GlusterFS	Lustre
Architecture	Centralized	Distributed	Decentralized (client-server)	Centralized
Interfaces				
System availability				
Data availability				
Load balancing				
Storage type				

Image from Gartner (March 2016)

Interfaces

- **Command Line Interface (CLI):** access files with traditional Unix commands (cp, rm, mv…)
- **Application Programming Interface (API):** implemented in different programming languages or REST(web-based)
- **Mount:** attach remote directories to the local filesystem (using unix *mount* or FUSE)

Some common DFS


	HDFS	Ceph	GlusterFS	Lustre
Architecture	Centralized	Distributed	Decentralized (client-server)	Centralized
Interfaces	CLI, FUSE, REST, API	FUSE, mount, REST	FUSE, mount	FUSE
System availability				
Data availability				
Load balancing				
Storage type				

FUSE: Filesystem in User SpacE

REST: REpresentational State Transfer (defining architectural constraints for web-based communications)

System availability

- **High availability:** metadata are replicated and distributed across several servers
- **Failover:** several servers in standby periodically save the metadata to be ready to take control
- **No failover:** this is a Single Point of Failure (SPOF). In HDFS a second server periodically saves the metadata, but a system stop is required for it to take control.

Some common DFS

	HDFS	Ceph	GlusterFS	Lustre
Architecture	Centralized	Distributed	Decentralized (client-server)	Centralized
Interfaces	CLI, FUSE, REST, API	FUSE, mount, REST	FUSE, mount	FUSE
System availability	No failover	High	High	Failover
Data availability				
Load balancing				
Storage type				

Image from Gartner (March 2016)

Data availability

- **Replication:** several copies of the data are made. This might raise consistency issues solved by synchronisation mechanisms and placement strategies.
- **RAID-like:** several copies of a whole storage device into other ones inside the same volume.
- **No replication:** rely on independent software.

Some common DFS



	HDFS	Ceph	GlusterFS	Lustre
Architecture	Centralized	Distributed	Decentralized (client-server)	Centralized
Interfaces	CLI, FUSE, REST, API	FUSE, mount, REST	FUSE, mount	FUSE
System availability	No failover	High	High	Failover
Data availability	Replication	Replication	RAID-like	No
Load balancing				
Storage type				

Load Balancing

- Overloaded servers can delay or abort request execution.
- These must be relieved and the data distributed on other servers in the cluster (possibly newly added empty ones).

Some common DFS


	HDFS	Ceph	GlusterFS	Lustre
Architecture	Centralized	Distributed	Decentralized (client-server)	Centralized
Interfaces	CLI, FUSE, REST, API	FUSE, mount, REST	FUSE, mount	FUSE
System availability	No failover	High	High	Failover
Data availability	Replication	Replication	RAID-like	No
Load balancing	Auto	Manual	Manual	No
Storage type				

Image from Gartner (March 2016)

Storage type

- **File store:** data is organised in directory trees, folders, and individual files (hierarchical). It functions similarly to a local hard drive.
  - **Parallel**: data is partitioned in blocks, then distributed simultaneously (striping)

- **Object store:** breaks files into pieces called objects. Each object receives a unique ID and also stores metadata about the file. Not hierarchical, scalable. Suited for unstructured data such as audio and video.

- **Block store:** breaks up data into blocks and then stores them as separate pieces, each with a unique ID.

Some common DFS

	HDFS	Ceph	GlusterFS	Lustre
Architecture	Centralized	Distributed	Decentralized (client-server)	Centralized
Interfaces	CLI, FUSE, REST, API	FUSE, mount, REST	FUSE, mount	FUSE
System availability	No failover	High	High	Failover
Data availability	Replication	Replication	RAID-like	No
Load balancing	Auto	Manual	Manual	No
Storage type	File	Object, Block, File	File	File (parallel)

Image from Gartner (March 2016)

Which one is best?

Small quantity of big files: HDFS, Lustre

Both small and big data: Ceph, GlusterFS

High throughput: Lustre

A closer look to HDFS

- manage pools of big data and supporting related BD analytics applications
- rapid transfer of data between compute nodes
- closely coupled with MapReduce and Spark
- breaks the information down into separate blocks and distributes them to different nodes in a cluster: highly efficient parallel processing
- highly fault-tolerant (data-wise). The file system replicates each piece of data multiple times and distributes the copies to individual nodes. In case of a node crash, processing can continue while data is recovered
- echoes POSIX design style in some aspects
- very large-scale implementations
- support for low-cost commodity hardware

HDF is commonly used in ML and BD analytics platforms, so it's worth knowing a little about it…

ADVANTAGES:

- achieve high throughput by co-locating data and computing on the same nodes (to overcome network limitations)

DISADVANTAGES:

- limited capacity per data node
- over-provisioning of compute resources (storage needs to grow faster than computing needs)

Other good solutions are available on the market, for instance MINIO (see later), that disaggregate storage and compute.

Hadoop | Overview | Datanodes | Datanode Volume Failures | Snapshot | Startup Progress | Utilities

Datanode Information

In operation

Node	Last contact	Admin State	Capacity	Used	Non DFS Used	Remaining	Blocks	Block pool used	Failed Volumes	Version
vdummy03.to.infn.it:50010 (192.168.2.182:50010)	0	In Service	9.52 GB	466.95 MB	2.35 GB	6.72 GB	163	466.95 MB (4.79%)	0	2.7.2
vdummy08.to.infn.it:50010 (192.168.2.187:50010)	1	In Service	9.52 GB	403.55 MB	2.5 GB	6.63 GB	162	403.55 MB (4.14%)	0	2.7.2
vdummy16.to.infn.it:50010 (192.168.2.195:50010)	2	In Service	9.52 GB	755.95 MB	2.32 GB	6.47 GB	115	755.95 MB (7.75%)	0	2.7.2
yoga-hdfs-namenode-0.yoga-hdfs-namenode.default.svc.cluster.local:50010 (192.168.2.39:50010)	1	In Service	525.61 GB	493.14 MB	30.87 GB	494.26 GB	153	493.14 MB (0.09%)	0	2.7.2
vdummy01.to.infn.it:50010 (192.168.2.180:50010)	0	In Service	9.52 GB	614.87 MB	2.76 GB	6.17 GB	149	614.87 MB (6.3%)	0	2.7.2
t2-mlwn-04.to.infn.it:50010 (192.168.2.84:50010)	0	In Service	424.09 GB	44.12 MB	34.05 GB	390 GB	12	44.12 MB (0.01%)	0	2.7.2
vdummy06.to.infn.it:50010 (192.168.2.185:50010)	0	In Service	9.52 GB	522.04 MB	2.36 GB	6.65 GB	127	522.04 MB (5.35%)	0	2.7.2

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

/user Hierarchical directory structure Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	jovyan	jovyan	0 B	11/15/2019, 12:12:08 PM	0	0 B	aliber92
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:48:25 PM	0	0 B	bagnasco
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:40:00 PM	0	0 B	gabrielefronze
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:53:12 PM	0	0 B	giorgiobar
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:36:21 PM	0	0 B	leggerf
drwxr-xr-x	jovyan	jovyan	0 B	11/15/2019, 12:13:42 PM	0	0 B	marco-ph
drwxr-xr-x	jovyan	jovyan	0 B	11/15/2019, 12:15:11 PM	0	0 B	obertino
drwxr-xr-x	jovyan	jovyan	0 B	11/14/2019, 6:49:07 PM	0	0 B	slusso
drwxr-xr-x	jovyan	jovyan	0 B	9/30/2019, 12:53:34 PM	0	0 B	svallero
drwxr-xr-x	jovyan	jovyan	0 B	10/2/2019, 11:37:29 AM	0	0 B	testuser
drwxr-xr-x	jovyan	jovyan	0 B	10/7/2019, 7:16:50 PM	0	0 B	testuser2

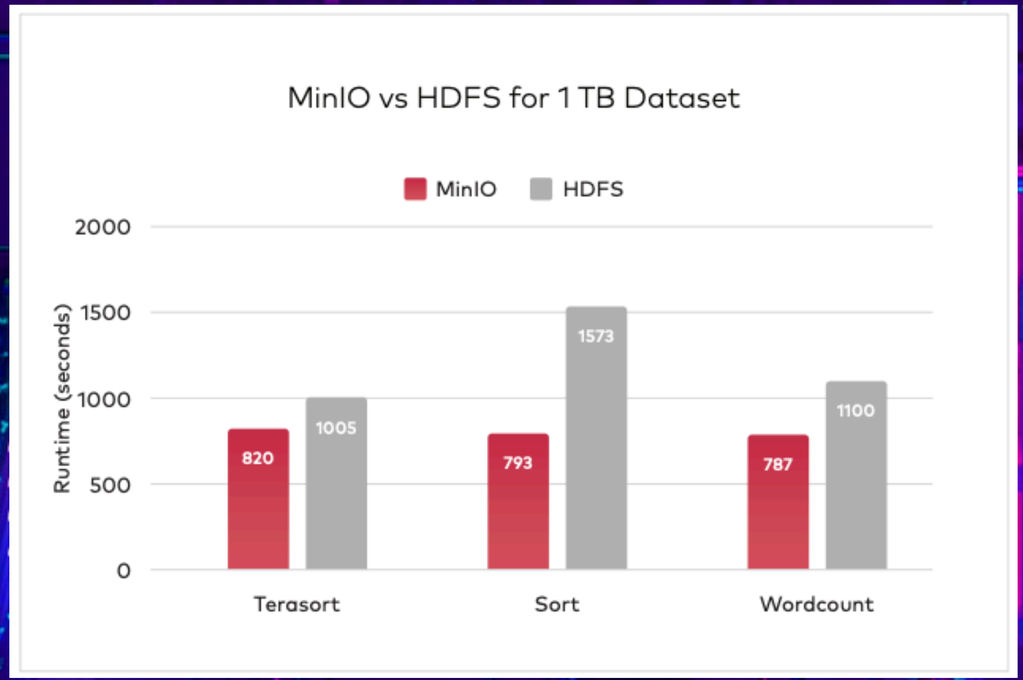
MINIO

- High performance object storage
- Designed for distributed architectures
- Compatible with the Amazon object storage solution (S3)
- Decouples storage and compute:
  - The two can be scaled independently
  - Natural fit for cloud environments where software stack and data pipelines are managed elastically
- Today, co-locating storage and compute might no longer be an advantage since:
  - Storage drives are denser
  - Networking is faster

MINIO is the storage solution adopted at the INFN-Cloud

MINO vs. HDFS:

- In the data-generation step (not performance critical) HDFS performs about 2 times better than MINIO
- In the benchmarking step, MINIO is up to 98% faster (results are shown below for a set of Hadoop benchmarks)



(https://min.io/resources/docs/MinIO-vs-HDFS-MapReduce-performance-comparison.pdf)

Some useful file format

Tabular data structures with headers

CSV

- Comma Separated Values
- Data is organised in rows
- Human readable (text file)
- Used i.e. to export data from Microsoft Excel

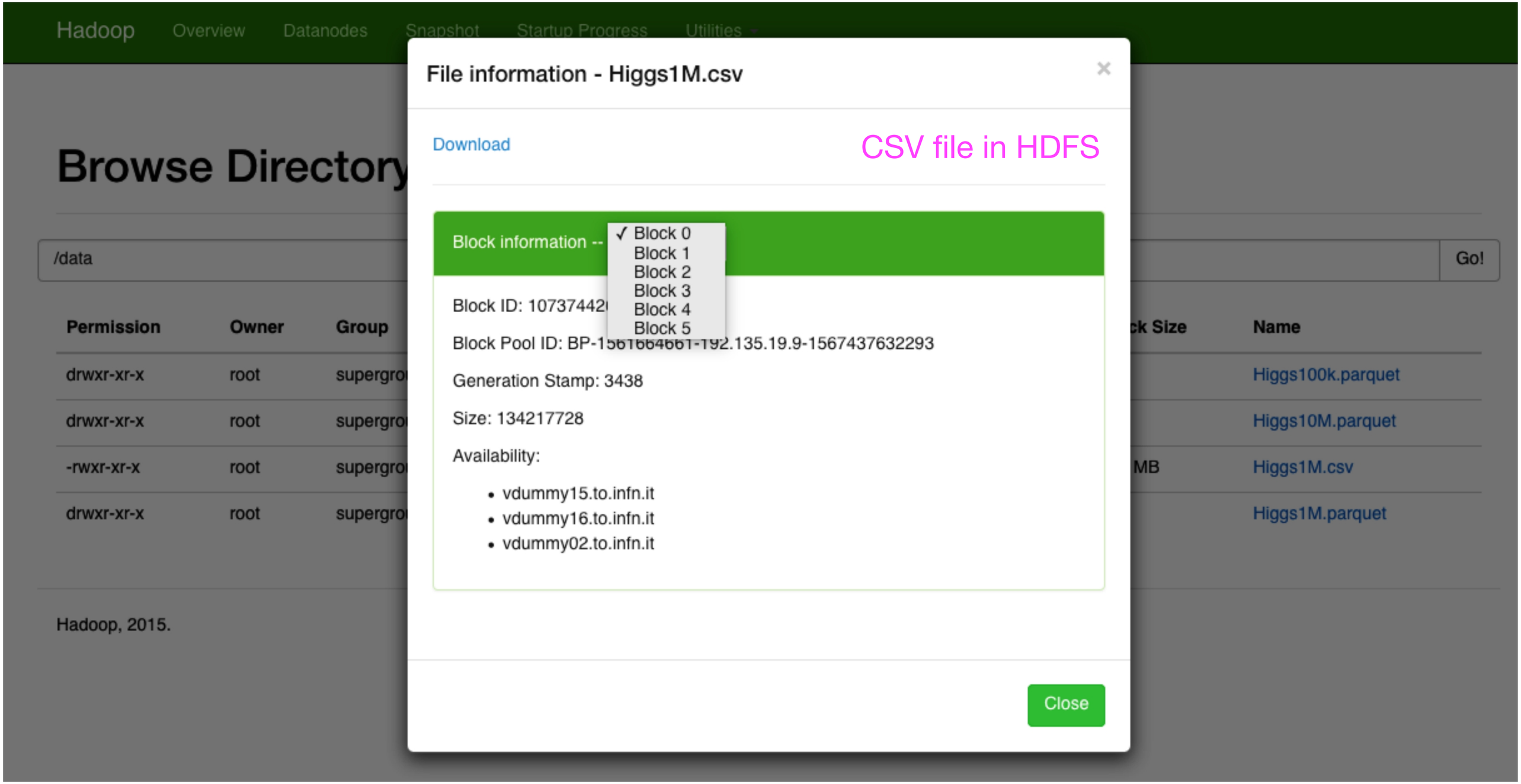
```
"Index", "Year", "Age", "Name", "Movie"
 1, 1928, 22, "Janet Gaynor", "Seventh Heaven, Street Angel and
Sunrise: A Song of Two Humans"
 2, 1929, 37, "Mary Pickford", "Coquette"
 3, 1930, 28, "Norma Shearer", "The Divorcee"
 4, 1931, 63, "Marie Dressler", "Min and Bill"
 5, 1932, 32, "Helen Hayes", "The Sin of Madelon Claudet"
 6, 1933, 26, "Katharine Hepburn", "Morning Glory"
 7, 1934, 31, "Claudette Colbert", "It Happened One Night"
 8, 1935, 27, "Bette Davis", "Dangerous"
 9, 1936, 27, "Luise Rainer", "The Great Ziegfeld"
10, 1937, 28, "Luise Rainer", "The Good Earth"
11, 1938, 30, "Bette Davis", "Jezebel"
12, 1939, 26, "Vivien Leigh", "Gone with the Wind"
13, 1940, 29, "Ginger Rogers", "Kitty Foyle"
14, 1941, 24, "Joan Fontaine", "Suspicion"
15, 1942, 38, "Greer Garson", "Mrs. Miniver"
16, 1943, 25, "Jennifer Jones", "The Song of Bernadette"
17, 1944, 29, "Ingrid Bergman", "Gaslight"
18, 1945, 40, "Joan Crawford", "Mildred Pierce"
```

Apache Parquet

- Columnar storage format
- Self describing (schema is embedded within the data)
- Supports complex nested data structures
- Optimised for query performance
- Minimised I/O
- Compression and encoding (Snappy)

SNAPPY ENCODING: a fast data compression and decompression library in C++ by Google. It does not aim for maximum compression, instead for very high speeds and reasonable compression. Compression speed is 250 MB/s and decompression speed is 500 MB/s using a single core.

Image from Gartner (March 2016)


Hadoop
Overview
Datanodes
Snapshot
Startup Progress
Utilities
Browse Directory
/data
Go!
Permission
Owner
Group
Name
drwxr-xr-x
root
Higgs100k.parquet
drwxr-xr-x
root
Higgs10M.parquet
-rwxr-xr-x
root
MB
Higgs1M.csv
drwxr-xr-x
root
Higgs1M.parquet
Hadoop, 2015.
File information - Higgs1M.csv
Download
CSV file in HDFS
Block information --
Block 0
Block 1
Block 2
Block 3
Block 4
Block 5
Block ID: 10737442
Block Pool ID: BP-1561664661-192.135.19.9-1567437632293
Generation Stamp: 3438
Size: 134217728
Availability:
vdummy15.to.infn.it
vdummy16.to.infn.it
vdummy02.to.infn.it
Close

Hadoop Overview Datanodes Snapshot Startup Progress Utilities

Browse Directory

PARQUET file in HDFS

/data/Higgs1M.parquet Go!

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
-rwxr-xr-x	root	supergroup	0 B	10/2/2019, 5:00:03 PM	3	128 MB	_SUCCESS
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:57 PM	3	128 MB	part-00000-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:57 PM	3	128 MB	part-00001-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:57 PM	3	128 MB	part-00002-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00003-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.14 MB	10/2/2019, 4:59:57 PM	3	128 MB	part-00004-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:57 PM	3	128 MB	part-00005-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:57 PM	3	128 MB	part-00006-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:57 PM	3	128 MB	part-00007-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00008-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.15 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00009-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00010-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00011-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00012-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00013-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.15 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00014-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:58 PM	3	128 MB	part-00015-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:59 PM	3	128 MB	part-00016-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.09 MB	10/2/2019, 4:59:59 PM	3	128 MB	part-00017-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet
-rwxr-xr-x	root	supergroup	2.18 MB	10/2/2019, 4:59:59 PM	3	128 MB	part-00018-5febe6aa-bdd6-4f41-b199-9bdd2595d689-c000.snappy.parquet

Commercial Clouds charge you by the amount of data stored or scanned per query

Dataset	Size on Amazon S3	Query Run time	Data Scanned	Cost
Data stored as CSV files	1 TB	236 seconds	1.15 TB	$5.75
Data stored in Apache Parquet format*	**130 GB**	**6.78 seconds**	**2.51 GB**	**$0.01**
Savings / Speedup	**87% less with Parquet**	**34x faster**	**99% less data scanned**	**99.7% savings**

Choosing the appropriate file format might not be just a matter of taste…

To take home

- We know what a distributed filesystem is, why it's useful…
- …and we can even name a few
- Co-locating data and compute on the same nodes might no longer be an advantage
- File format matters

What about the Grid?

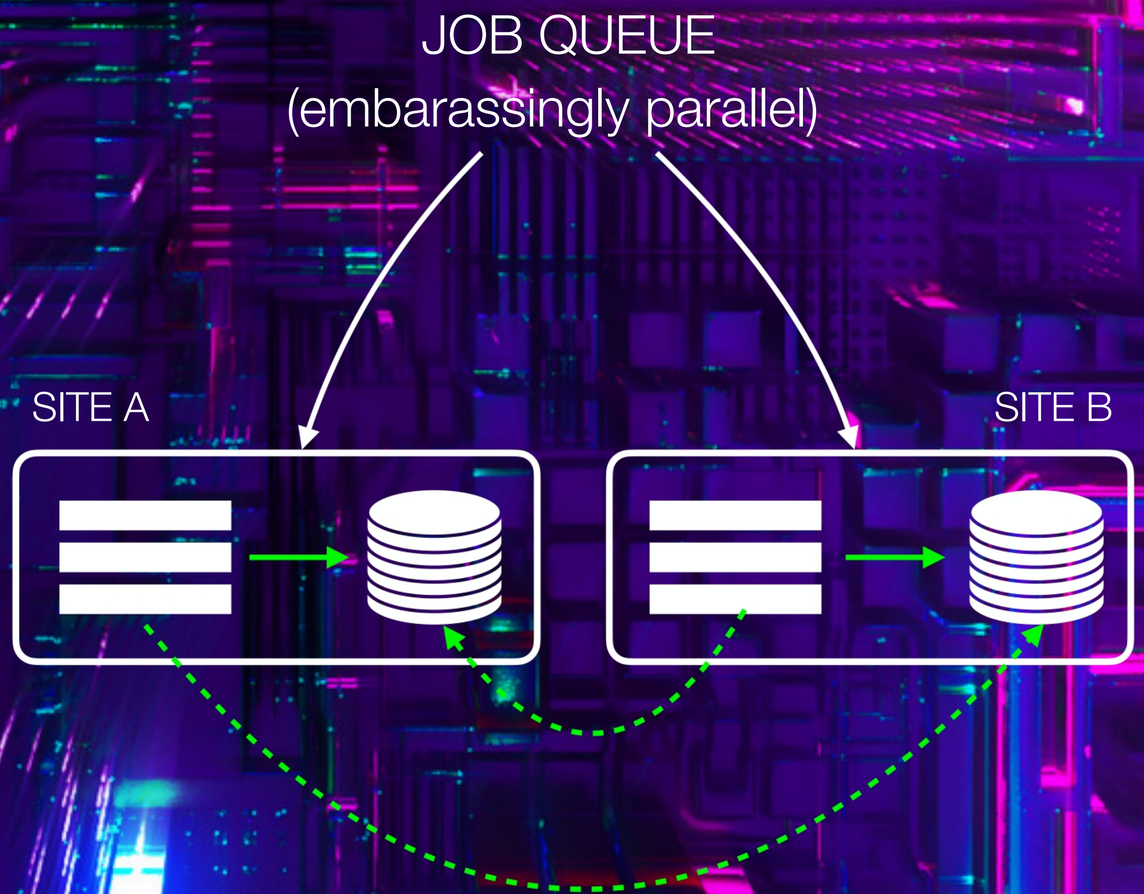


Image from Gartner (March 2016)