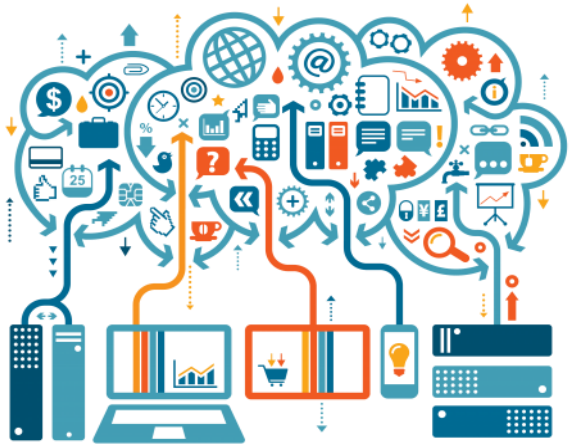
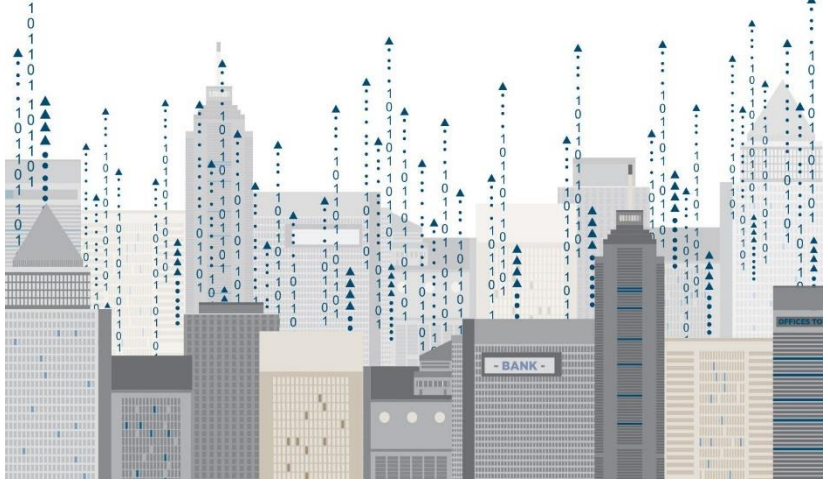


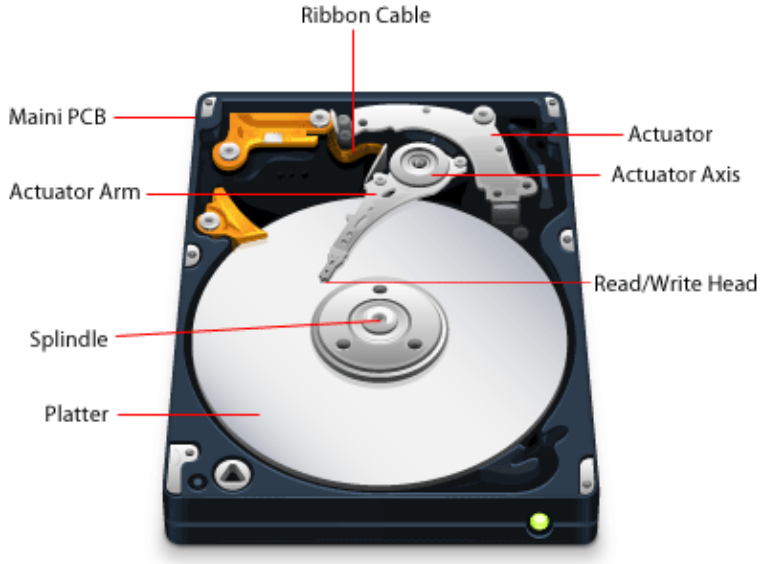


Big Data Analytics

Fall 2025

Lecture 5

HDD



Series of platters covered by a ferromagnetic coating. Direction of magnetization represents individual bits.

Data is written and read by a fast-moving head. Disk spins at 7200 RPMs - data read very quickly.

Slow but extremely cost efficient for long-term offline storage

Benefits of SSD over HDD only for day-to-day usage — eg rate at which they can load large files

SSD is non-volatile storage that uses flash memory to store– no moving parts, hence faster, durable, and energy-efficient.

Use NAND flash memory - data is retained even when the device is powered off.

No moving parts - Makes them more durable and less prone to physical failure from shock or vibration.

SSD

The following processes work faster with an SSD:

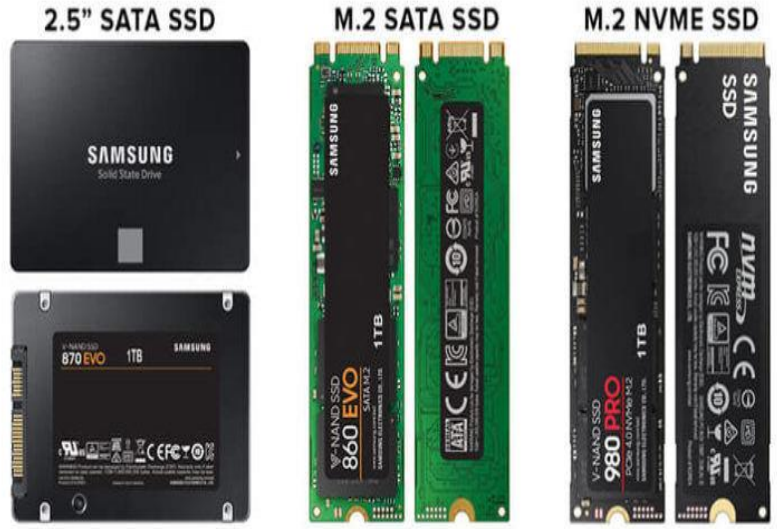
Booting up OS, Starting a program, Loading a new video game level, Opening a huge file in a resource-intensive program, Importing and exporting video files, Previewing video files in editing software

- SATA and NVM: Interfaces to connect storage to a system

- SATA: SATA (Serial Advanced Technology Attachment) connects both HDD and SSD to motherboard

- SATAIII supports 6 Gbps (600 MB/s)

- SATA uses a distinct 7-pin connector for data and a separate 15-pin connector for power.



- NVMe (Non-Volatile Memory Express): a high-performance, scalable interface specifically designed for SSDs to use the PCIe (Peripheral Component Interconnect Express) bus, providing much faster data transfer rates compared to SATA.

- Modern NVMe SSDs supporting speeds of up to 7,000 MB/s (M.2 slot or U.2 slot)

Use Case	How Optane Helps	Use Case
In-memory analytics (Spark, Flink, Presto)	Expand memory capacity (e.g., TBs per node) at lower cost than DRAM.	**In-memory analytics (Spark, Flink, Presto)**
Caching hot datasets	SSDs handle very high random reads/writes → faster queries in Hive LLAP, Presto, Druid.	**Caching hot datasets**
Hybrid storage tiers	Middle layer between DRAM and SSDs → latency-sensitive analytics.	**Hybrid storage tiers**
Write-heavy workloads (logs, streaming, ML checkpoints)	Extreme endurance makes it ideal for constant write-heavy environments.	**Write-heavy workloads (logs, streaming, ML checkpoints)**
Checkpointing / Recovery	Faster state recovery in distributed systems (Kafka, Flink, Hadoop YARN).	**Checkpointing / Recovery**

Aspect	HDD (Hard Disk Drive)	SSD (Solid State Drive)	Impact on BDA
Technology	Magnetic spinning disks with read/write head	Flash-based NAND memory (no moving parts)	SSDs remove mechanical latency → faster I/O in analytics workloads
Read/Write Speed	~80–160 MB/s (sequential)	~500 MB/s (SATA) to >5 GB/s (NVMe)	SSD accelerates data ingestion, ETL, and real-time queries
Latency	5–10 ms (seek time, rotational delay)	<0.1 ms (no mechanical delays)	Lower latency = faster query response, essential for streaming analytics
IOPS (Input/Output Ops/sec)	~100–200 IOPS	50,000–1,000,000+ IOPS	SSDs handle massive parallel reads/writes for distributed frameworks (e.g., Spark, Hadoop)
Durability (Shock Resistance)	Sensitive to physical shocks (mechanical parts)	Highly resistant (no moving parts)	SSD more reliable in clustered / distributed storage nodes

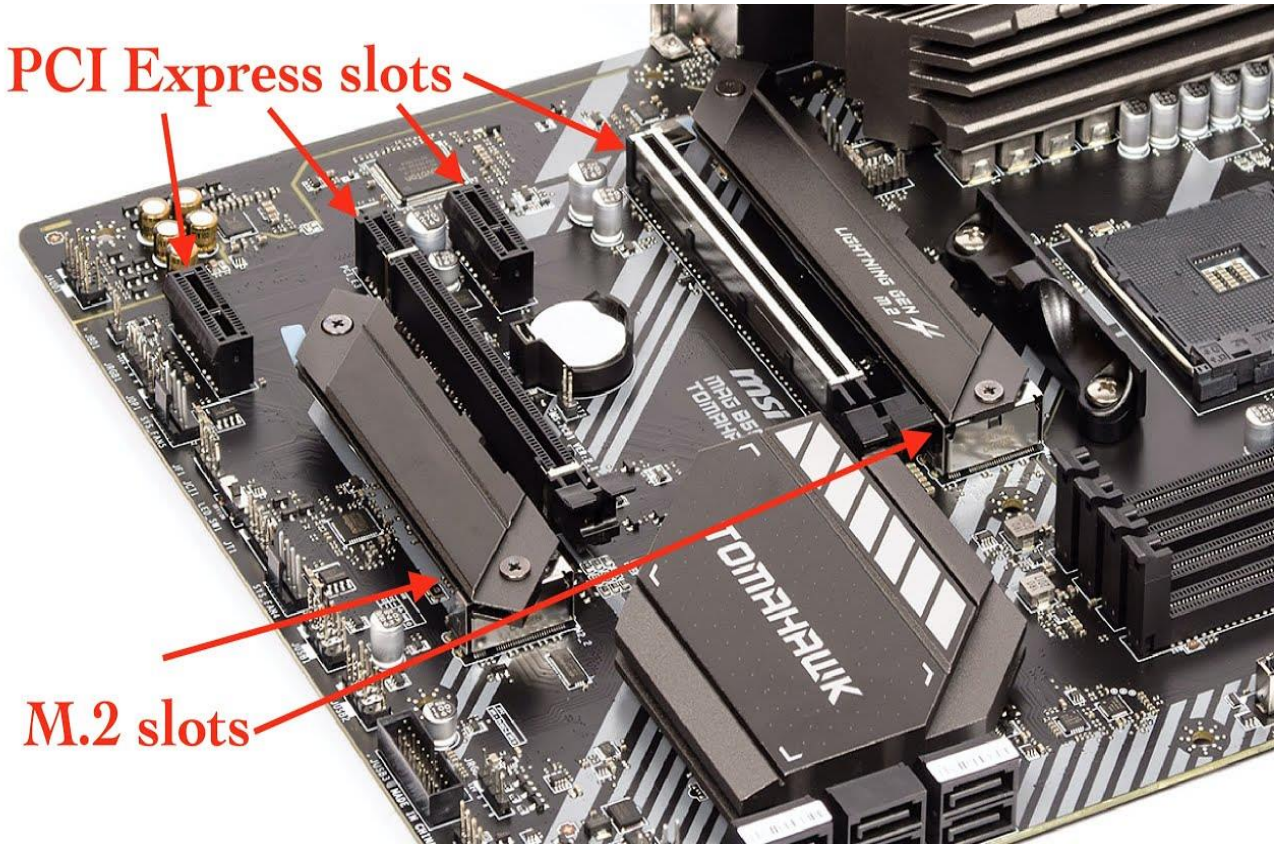
Aspect	HDD (Hard Disk Drive)	SSD (Solid State Drive)	Impact on BDA
Capacity	Larger capacities available (up to 20 TB+)	Lower typical capacity (commonly 1–8 TB, enterprise SSDs larger but costly)	HDDs still used in data lakes for raw archival storage
Cost per GB	Much cheaper (≈ $0.02–$0.03/GB)	More expensive (≈ $0.08–$0.12/GB)	HDD better for cold storage, SSD better for hot/active datasets
Power Consumption	Higher (6–10 W)	Lower (2–5 W)	SSDs save costs in large-scale BDA clusters (energy + cooling)
Lifespan (Write Cycles)	Not limited by write cycles, but wear due to mechanical failure	Limited write endurance (but modern enterprise SSDs are robust)	HDDs for long-term archival, SSDs for high-write workloads (real-time dashboards, logs)
Best Use Cases in BDA	Data lakes, cold storage, archival, batch analytics (HDFS raw layers)	Real-time analytics, machine learning pipelines, high-performance query engines (Presto, Spark, Hive LLAP)	Hybrid strategy often used: HDD for lake storage + SSD for compute nodes

Parameter	Definition	**HDD** (Spinning Disk)	**SSD (SATA)**	**SSD (NVMe/PCIe)**	**DRAM (Memory)**	**Persistent Memory (NVRAM/Optane)**	**BDA Relevance**
Transfer Speed	Sequential read/write speed	80–160 MB/s	500–600 MB/s	3–7 GB/s	10–25 GB/s	1–3 GB/s	Affects ETL, batch ingestion, and scan-heavy workloads
IOPS (Input/Output per second)	Number of random read/write ops per second	100–200	~50k–100k	500k–1M	Millions	300k–800k	Critical for real-time queries and OLTP-style workloads in BDA
Latency	Delay between request & first byte	5–10 ms	0.1–0.3 ms	~10–50 µs	<100 ns	~300 ns	Lower latency = faster streaming and real-time dashboards
Seek Time	Time to position head for data access	3–12 ms	N/A (no moving parts)	N/A	N/A	N/A	HDD bottleneck in random access, SSD/NVRAM superior

Parameter	Definition	**HDD** (Spinning Disk)	**SSD (SATA)**	**SSD (NVMe/PCIe)**	**DRAM (Memory)**	**Persistent Memory (NVRAM/Optane)**	**BDA Relevance**
Block Size	Smallest data unit read/written	512 B – 4 KB	4 KB	4 KB	64 B – 4 KB	256 B – 4 KB	Larger block = better throughput, smaller = better random I/O
Throughput	Effective sustained data transfer	100–200 MB/s	400–550 MB/s	3–6 GB/s	15–25 GB/s	1–3 GB/s	Defines how much data can be streamed for analytics jobs
Bandwidth (Bus Interface)	Data rate of connection bus	SATA-III: 6 Gb/s	SATA-III: 6 Gb/s	PCIe Gen4 x4: 64 Gb/s	DDR4/DDR5: 25–50 GB/s	PCIe/DDR-Tier	Determines scaling in distributed BDA clusters
Durability / Endurance	Resistance to wear/failure	Mechanical wear/failure	NAND wear (TBW limited)	NAND wear (better TBW than SATA)	High endurance	Limited write endurance	Impacts reliability in write-heavy analytics jobs
Cache / Buffer	Local cache for faster access	32–256 MB	512 MB–2 GB	1–4 GB + controller DRAM	Several MB per core	Large internal buffers	Helps smooth I/O spikes in big data pipelines
Power Consumption	Energy per active use	6–10 W	2–4 W	3–6 W	High (but efficient per GB/s)	Moderate	Relevant for cluster energy efficiency
Cost/GB	Storage cost efficiency	Very low ($0.02–0.03)	Moderate ($0.08–0.12)	Higher ($0.10–0.20)	Very high ($5–10)	High ($1–3)	Determines feasibility at PB scale in data lakes
Best Role in BDA	Where it fits	Data lakes, cold storage	Hot storage, query cache	Real-time analytics, ML pipelines	In-memory analytics (Spark, Flink)	Hybrid tier: between SSD & RAM	Balanced usage across pipeline layers

Storage Blues

- A typical HDD from 1990 could store 1,370 MB of data and had a transfer speed of 4.4 MB/s, read all the data from a full drive in around five minutes.
- Over 20 years later, 1-terabyte drives are the norm, but the transfer speed is around 100 MB/s - more than two and a half hours
- Nowadays with SSDs (~500MB/s), it would take around ~33 minutes
- Write speed max is 600MB/s – NVM SSDs are faster

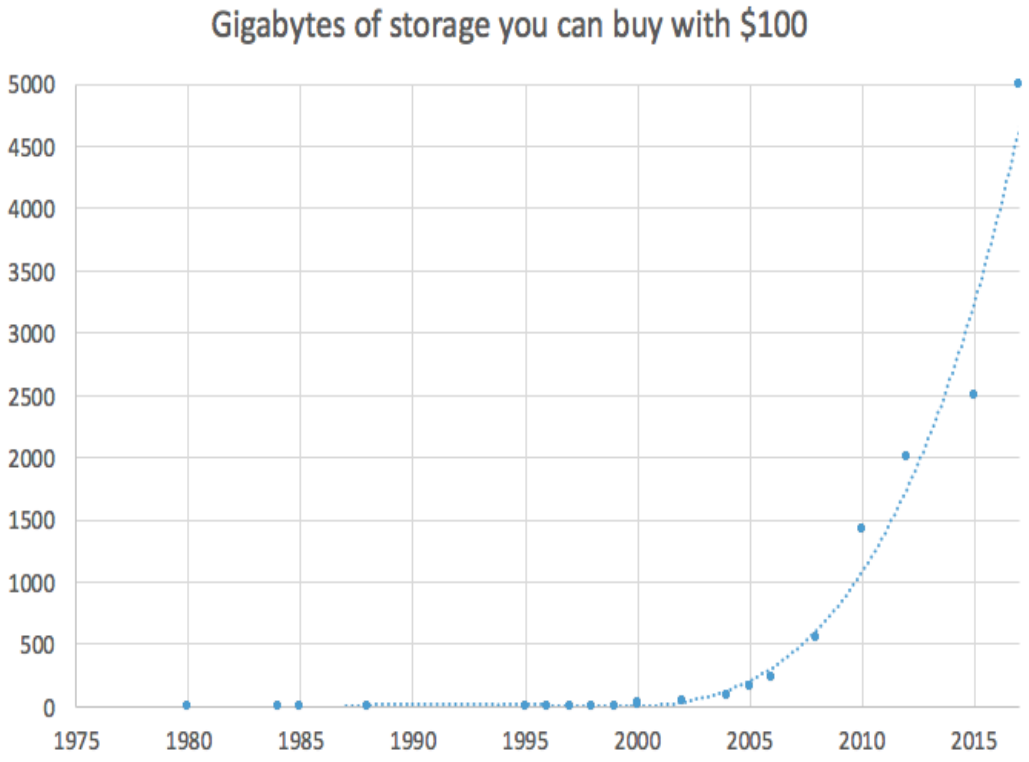

PCI Express slots
M.2 slots
MSI
MAG B5
TOMAHAWK
LIGHTNING GEN 4 M.2
TOMAHAWK

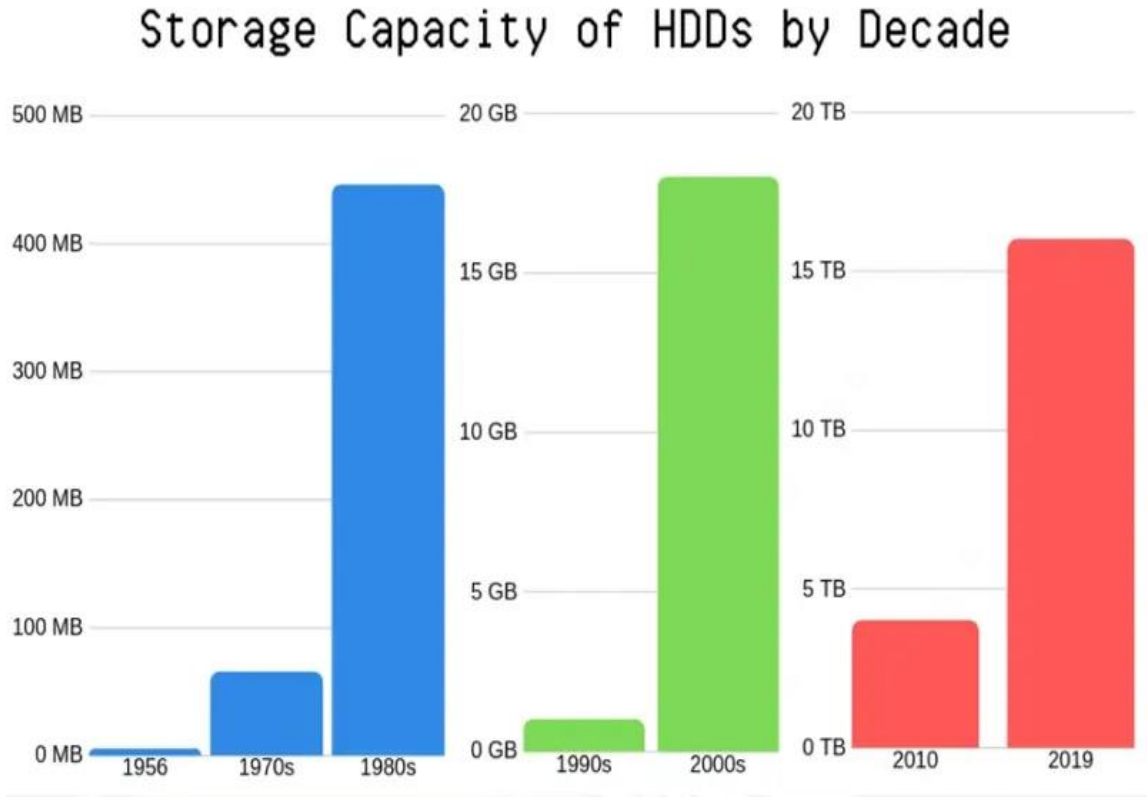

SAMSUNG
V-NAND SSD
980 PRO
PCIe 4.0 NVMe M.2
500GB
www.samsung.com/ssd
SAMSUNG ELECTRONICS CO., LTD.
XDA


Seagate

Storage Blues

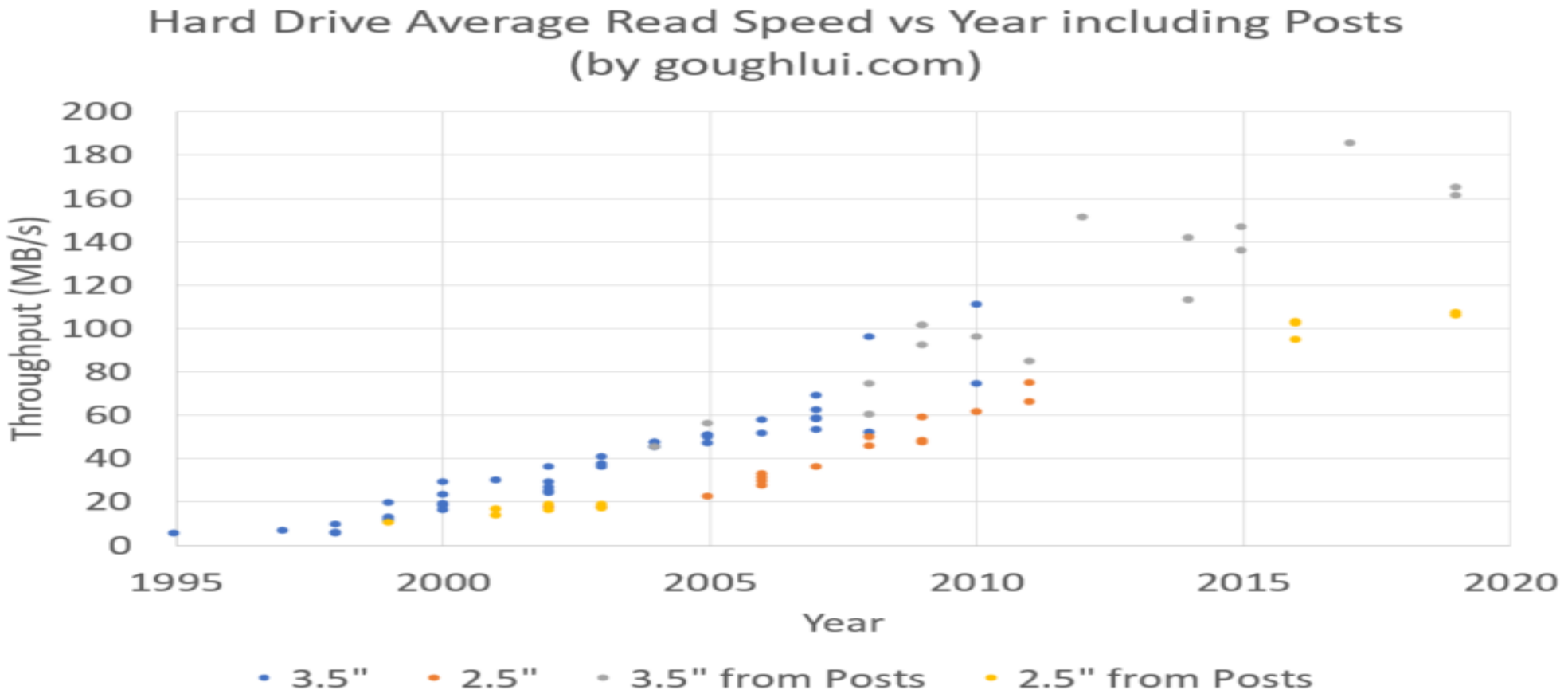
- Storage capacities of hard drives have increased massively over the years





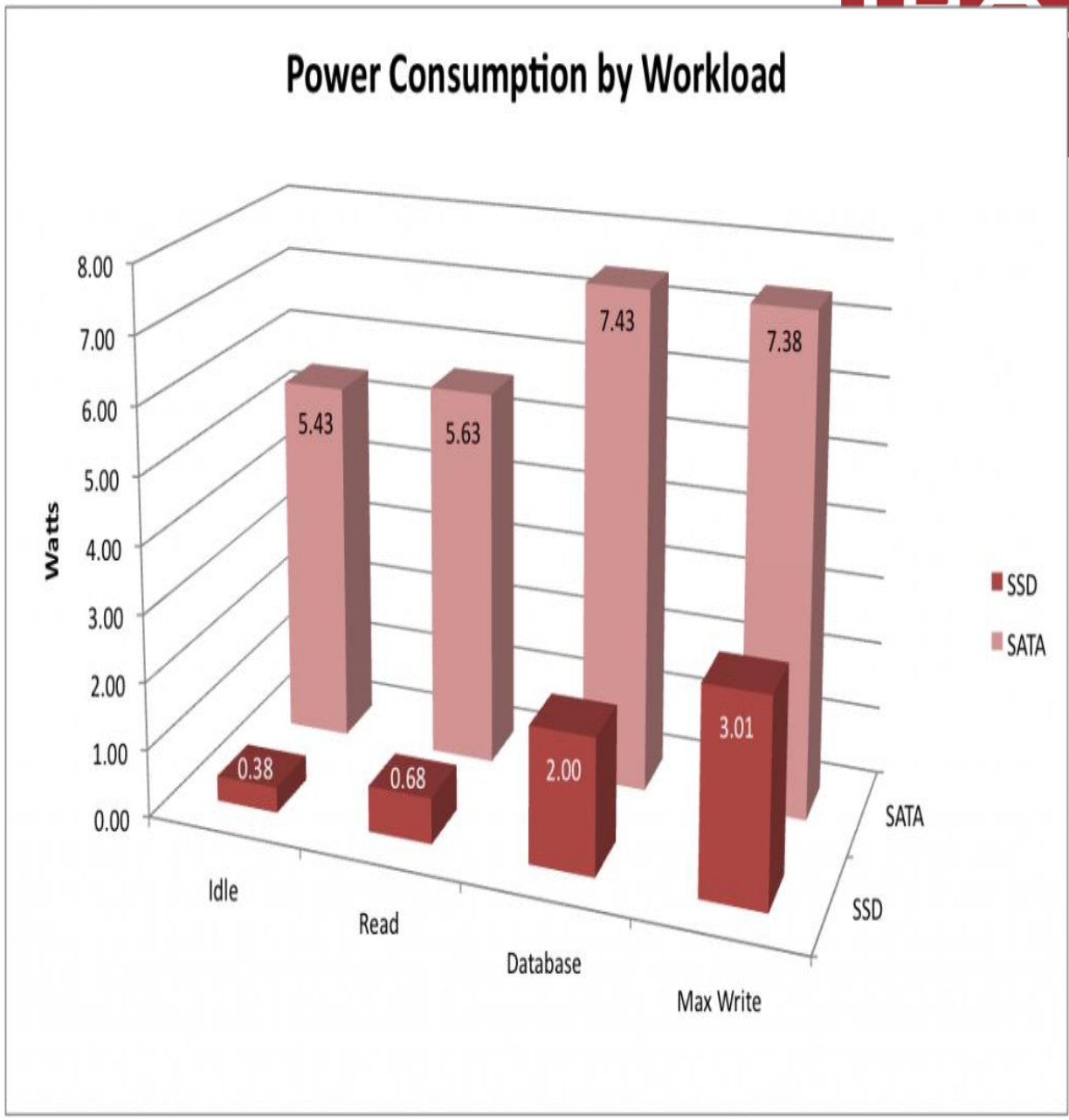
Storage Blues

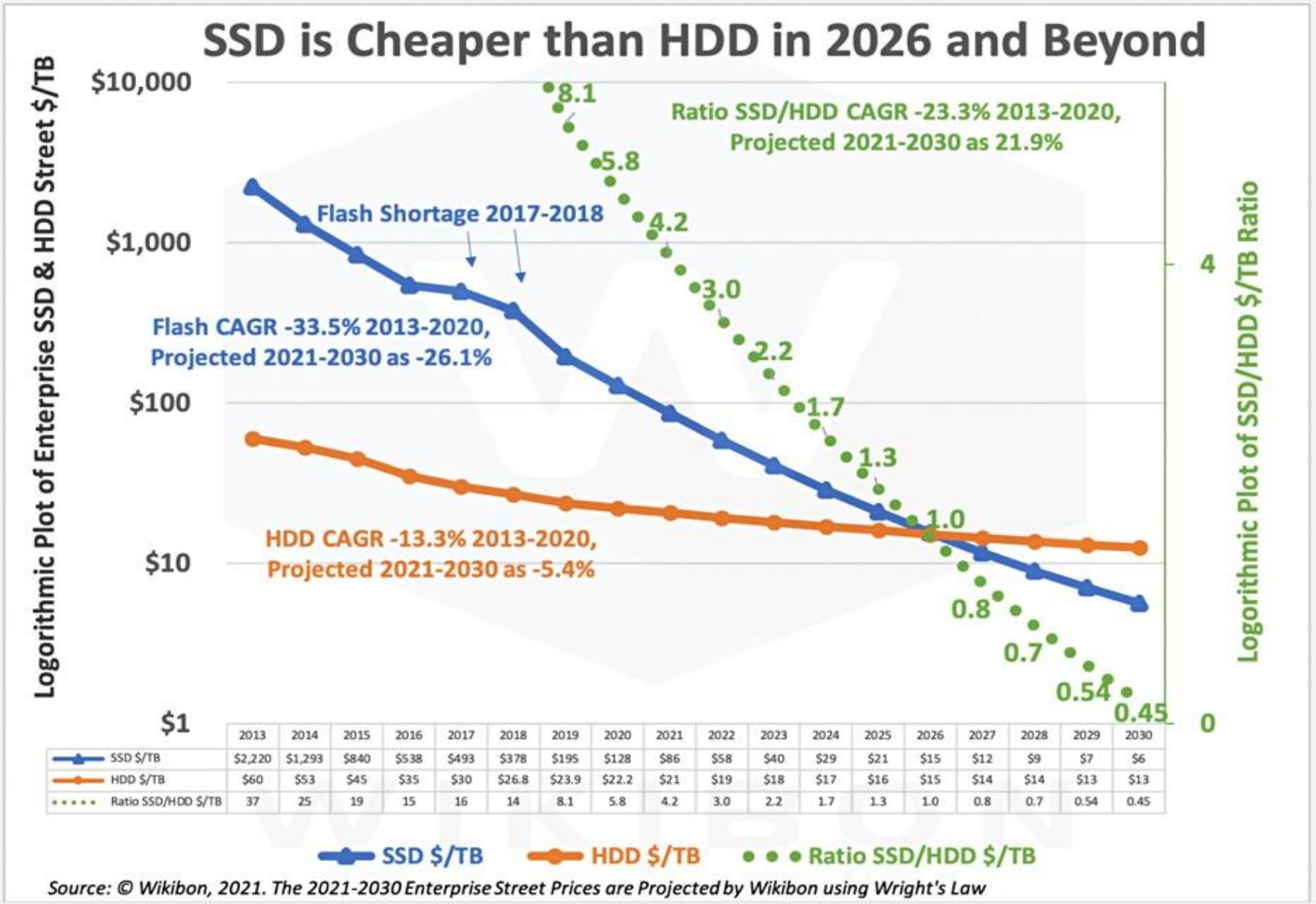
- But access speeds—the rate at which data can be read from drives—have not kept up



SSD vs. HDD power consumption

Power data per drive	SSD	HDD	HDD advantage
Idle (watts)	5	5.7	-15%
Active read (watts)	15	9.4	37%
Active write (watts)	20	6.4	68%
Read-intensive workload (avg. watts)	14.5	8.7	40%
Write-intensive workload (avg. watts)	18.0	6.6	63%
Power-density read-intensive (TB/watt)	2.1	2.5	19%
Power-density write-intensive (TB/watt)	1.7	3.3	94%



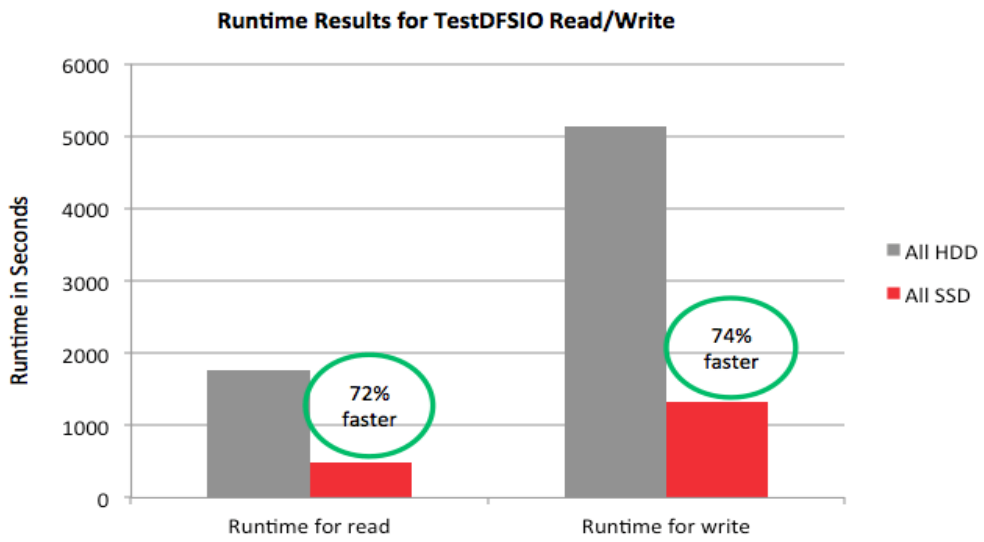


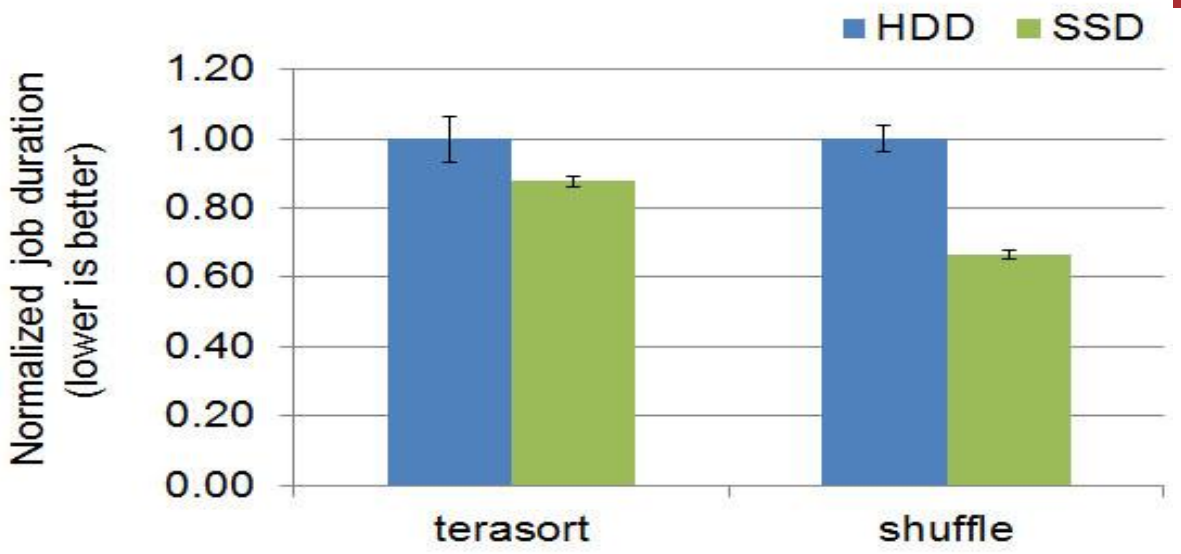
	2013	2014	2015	2016	2017	2018	2019	2020	2021	2022	2023	2024	2025	2026	2027	2028	2029	2030
SSD $/TB	$2,220	$1,293	$840	$538	$493	$378	$195	$128	$86	$58	$40	$29	$21	$15	$12	$9	$7	$6
HDD $/TB	$60	$53	$45	$35	$30	$26.8	$23.9	$22.2	$21	$19	$18	$17	$16	$15	$14	$14	$13	$13
Ratio SSD/HDD $/TB	37	25	19	15	16	14	8.1	5.8	4.2	3.0	2.2	1.7	1.3	1.0	0.8	0.7	0.54	0.45

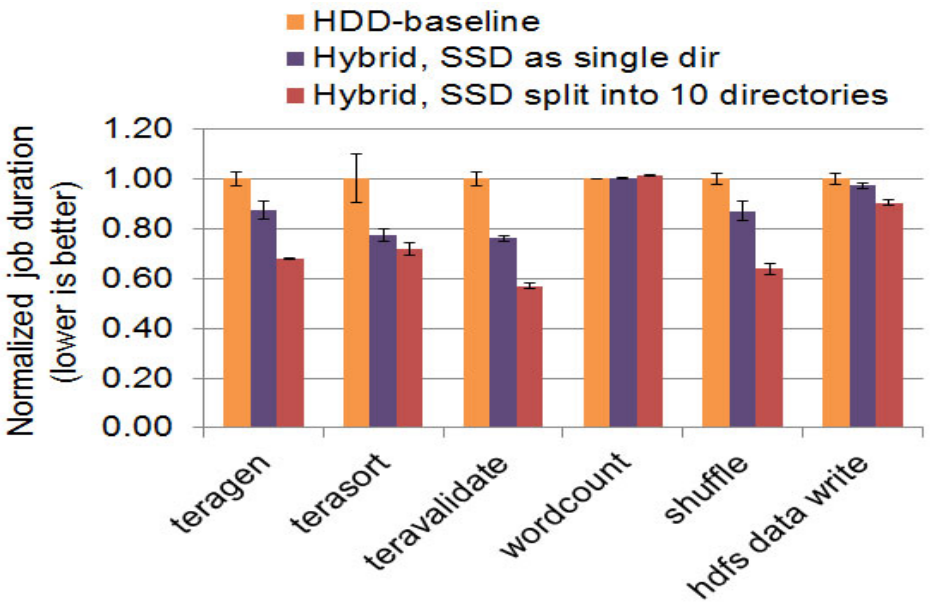
Figure 4 – SSD/HDD Pricing Ratio 2013 – 2030

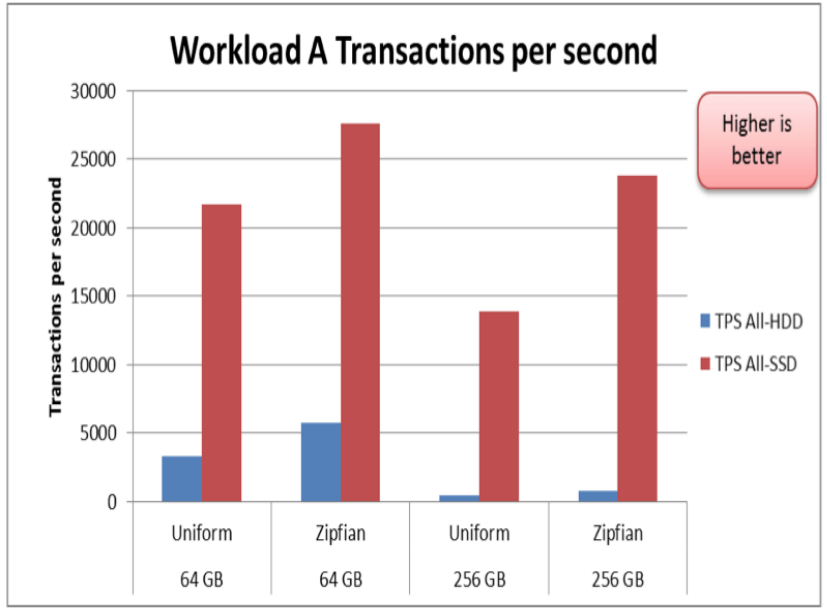
Source: © Wikibon, 2021.

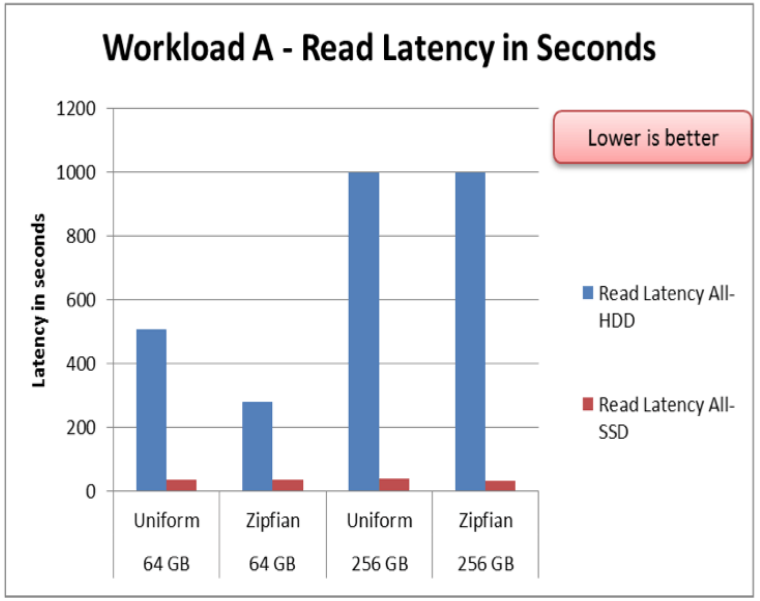
Results with Hadoop











Storage Blues

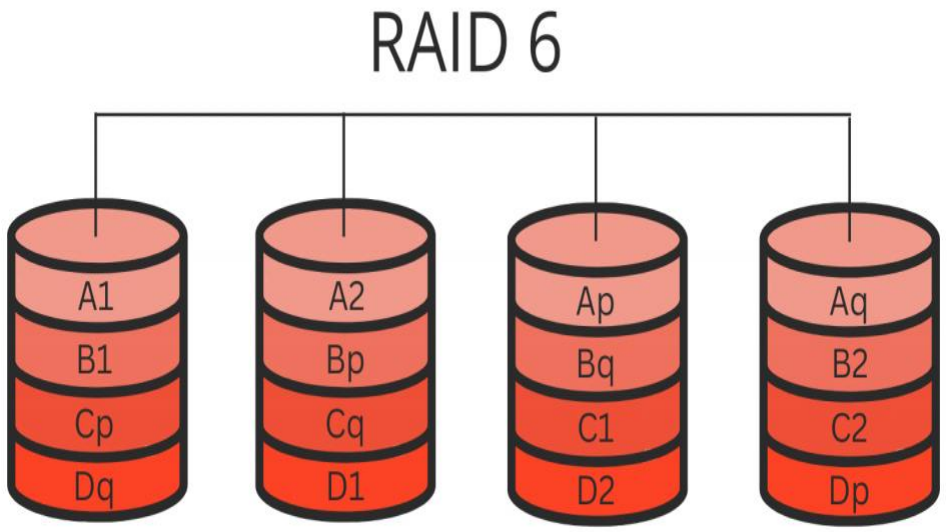
- Notwithstanding SSD, it takes a long time to read big data from a single drive
- If we had 100 drives, each holding one hundredth of the data. Working in parallel, we could read the data in under two minutes
- Using only one hundredth of a disk may seem wasteful - But we can store big data (100 datasets), each of which is 1 terabyte, and provide shared access to them.
- Users happy to share access in return for shorter analysis times – no interference

Storage Blues


002.006
002.005
002.004
NFSDOG1
IBM

Hardware Failure

- First problem: hardware failure: as soon as you start using many pieces of hardware, the chance that one will fail is high.
- Replication
- This is how RAID works (at the hardware level)- although Hadoop's Distributed Filesystem (HDFS), takes a slightly different approach (at the cluster level)



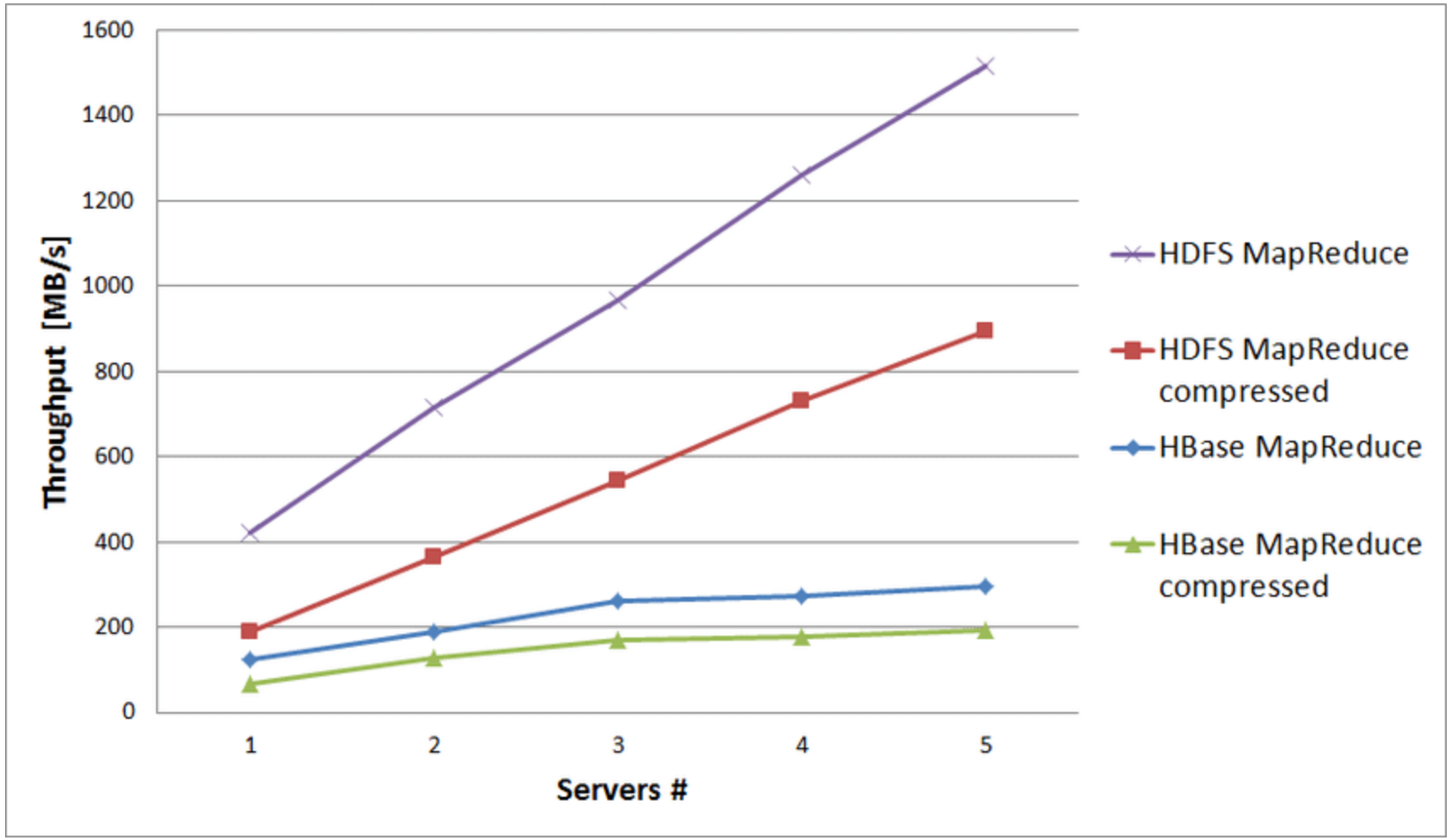
Aspect	RAID Replication (e.g., RAID 1/10)	HDFS Replication
Scope	Within a **single server**	Across a **cluster of servers/nodes**
Granularity	Replicates data blocks at **disk level**	Replicates HDFS blocks (default 128 MB) across nodes
Replication Factor	Fixed (usually 2 copies in RAID 1, mirrored sets in RAID 10)	Configurable (default 3 copies, can increase/decrease per dataset)
Location of Copies	All copies stored **inside one machine**	Copies distributed across **different machines/racks**
Performance Impact	- Writes: slower (must write to both disks)- Reads: faster (can read from any mirror)	- Writes: network overhead (sends block copies)- Reads: parallel across nodes (can pick nearest copy)
Recovery from Failure	Automatic switch to mirror disk	Automatic recovery by replicating missing block to healthy node
Use Case in BDA	Useful for **metadata servers** (NameNode, JournalNode)	Core mechanism for **data durability and fault tolerance**

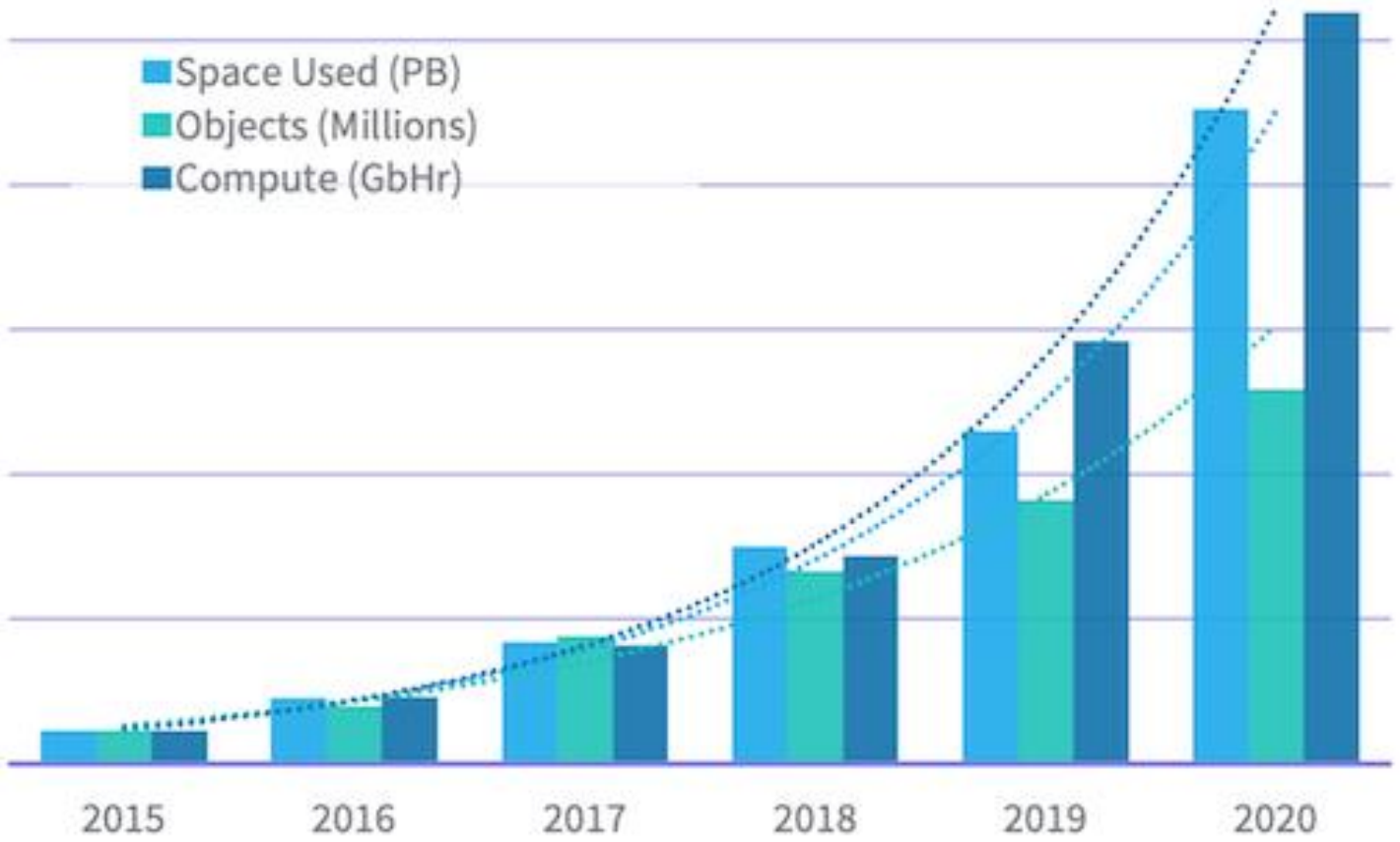
Combine distributed data for analysis

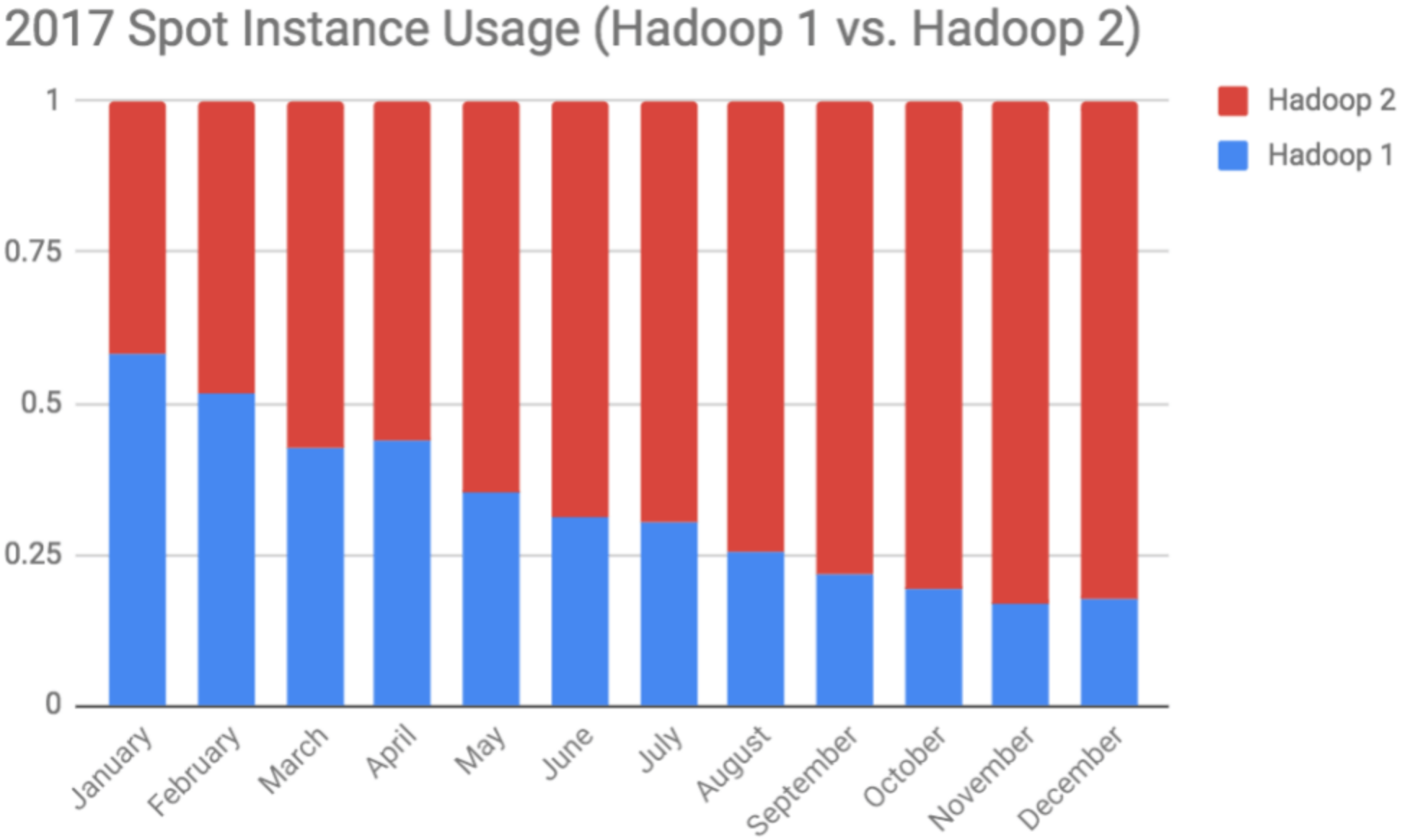
- Second problem: Most BDA tasks need to be able to combine the data in some way - data from one disk may need to be combined with data from any of the other 99 disks.
- Very difficult to do this correctly
- MapReduce: a programming model that:
  - Abstracts the problem from disk reads and writes
  - Transforms it into a computation over sets of keys and values.
- Two parts: map and the reduce—and it's the interface between the two where the "mixing" occurs.
- Like HDFS, MapReduce has built-in reliability (HDFS-based)

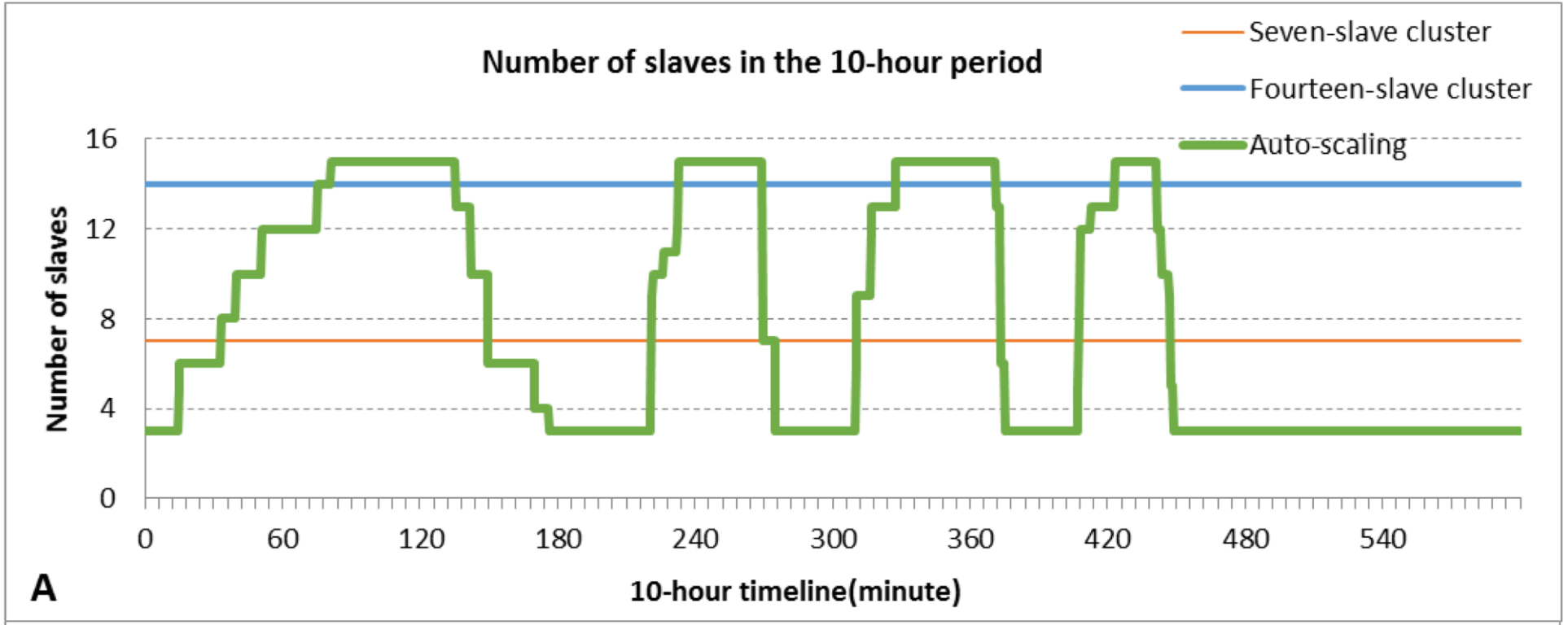
Hadoop's Power

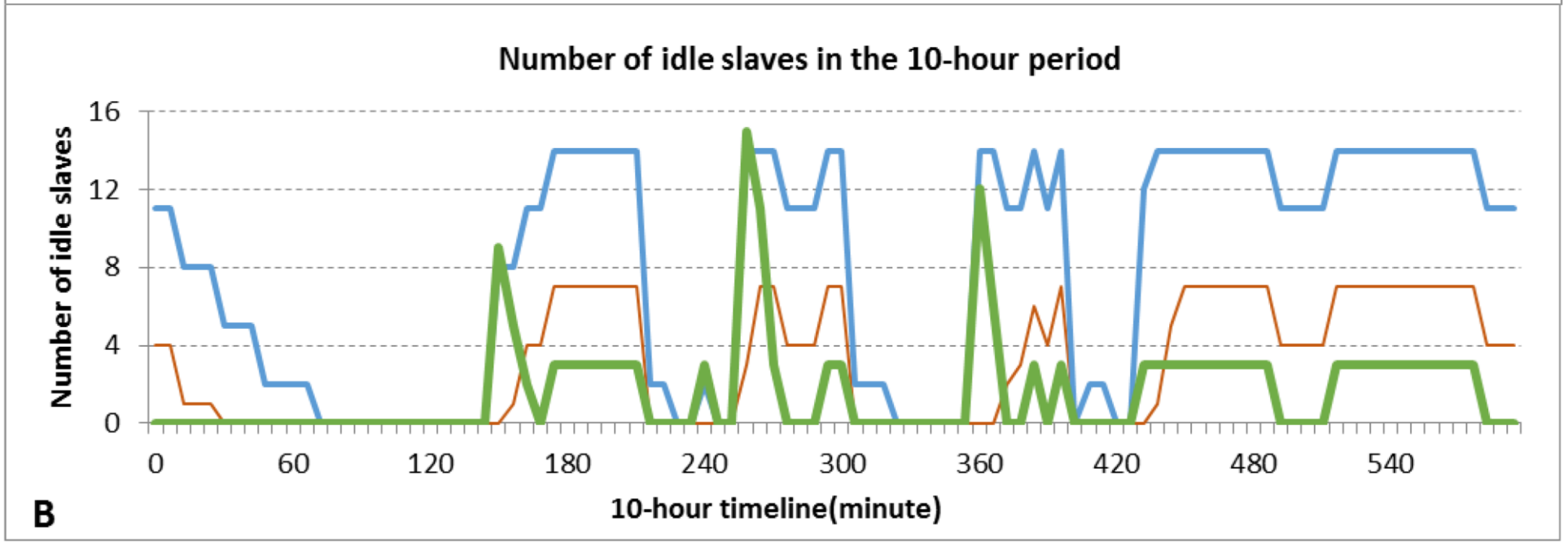
- A reliable, scalable platform for storage and analysis.
- Runs on commodity hardware and is open source - affordable.
- MapReduce - brutely scale as much as possible and combine forcibly)
- But: the premise is that the entire big dataset—or at least a good portion of it—can be processed for each query (and it is true)
- And this is the power: A batch query processor - ability to run an ad hoc query against your whole dataset and get the results in a reasonable time is innovative


1600
1400
1200
1000
800
600
400
200
0
Throughput [MB/s]
1
2
3
4
5
Servers #
HDFS MapReduce
HDFS MapReduce compressed
HBase MapReduce
HBase MapReduce compressed


Space Used (PB)
Objects (Millions)
Compute (GbHr)
2015
2016
2017
2018
2019
2020


2017 Spot Instance Usage (Hadoop 1 vs. Hadoop 2)
1
0.75
0.5
0.25
0
Hadoop 2
Hadoop 1
January
February
March
April
May
June
July
August
September
October
November
December


Number of slaves in the 10-hour period
Seven-slave cluster
Fourteen-slave cluster
Auto-scaling
Number of slaves
16
12
8
4
0
0
60
120
180
240
300
360
420
480
540
10-hour timeline(minute)
A

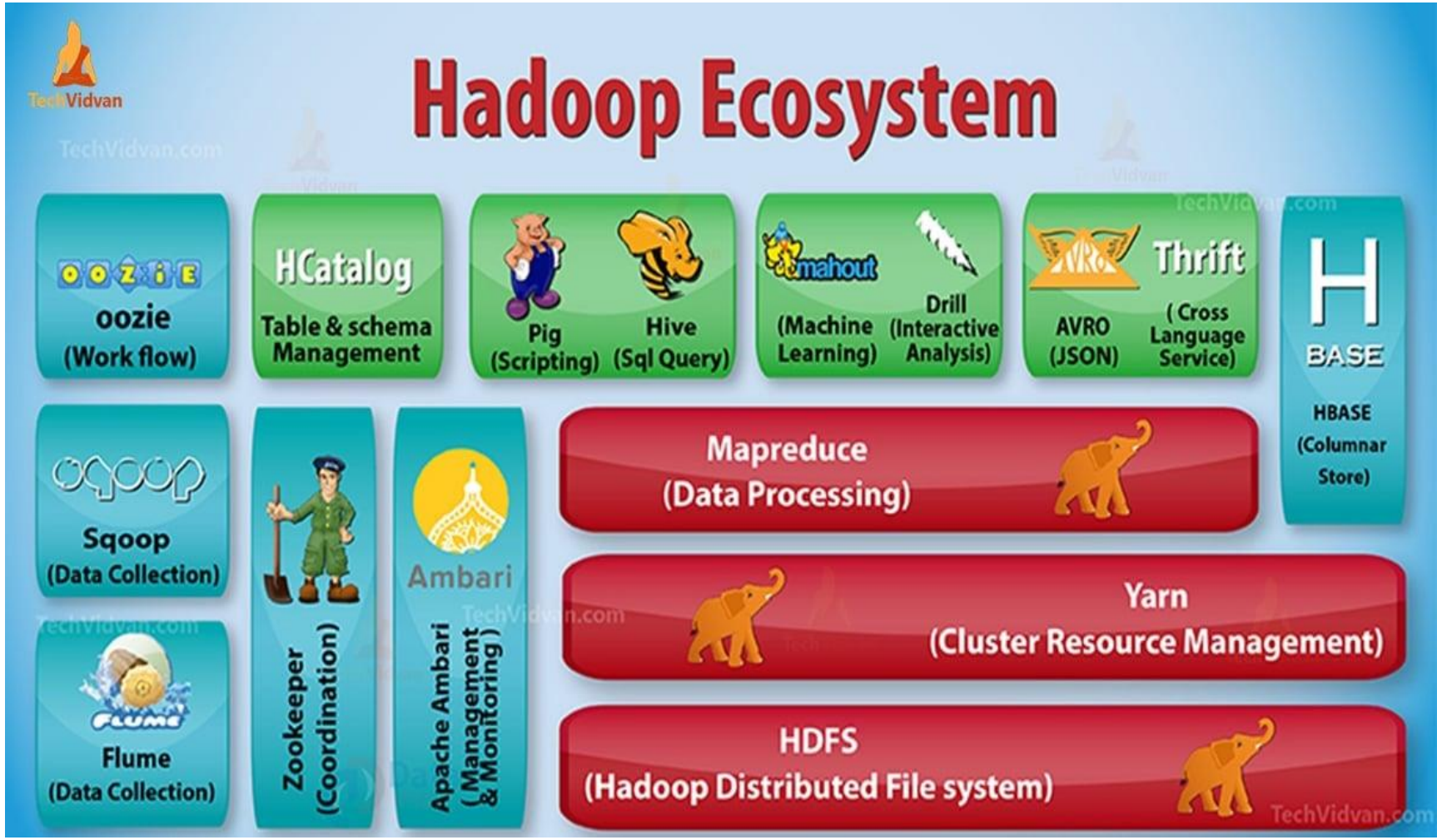

Number of idle slaves in the 10-hour period
Number of idle slaves
16
12
8
4
0
0
60
120
180
240
300
360
420
480
540
10-hour timeline(minute)
B

Hadoop's Power

- Mailtrust, Rackspace's mail division, used Hadoop for processing email logs - One ad hoc query they wrote was to find the geographic distribution of their users.
- In their words: *This data was so useful that we've scheduled the MapReduce job to run monthly, and we will be using this data to help us decide which Rackspace data centers to place new mail servers in as we grow*.
- But MapReduce is batch - not suitable for interactive analysis.
- You can't run a query and get results back in a few seconds or less. Queries typically take minutes or more, so it's best for offline use

Hadoop Eco

- However, since its original incarnation, Hadoop has evolved beyond batch processing
- “Hadoop”: refer to a larger ecosystem of projects, not just HDFS and MapReduce
- Many hosted by the Apache Software Foundation

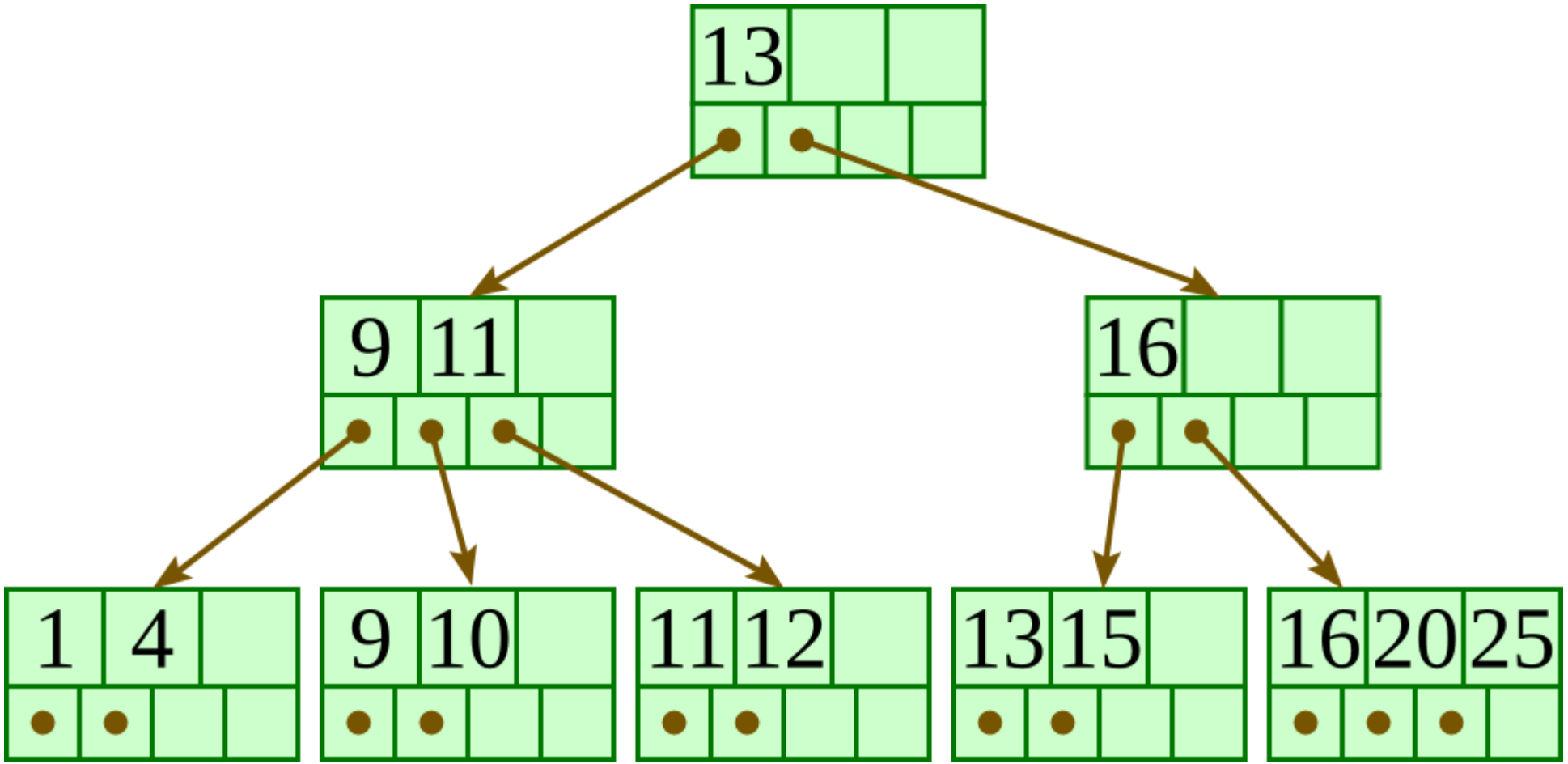


Hadoop Eco

- The first component to provide online access was Hbase: a key-value store that uses HDFS for its underlying storage.
- Provides both online read/write access of individual rows and batch operations for reading and writing data in bulk - good solution for building applications
- The real enabler for new processing models in Hadoop: YARN (which stands for Yet Another Resource Negotiator) in Hadoop 2.
- It is a cluster resource management system, which allows any distributed program (not just MapReduce) to run on data in a Hadoop cluster.

Hadoop services

- Interactive SQL:
  - A distributed query engine with dedicated “always on” daemons
  - Impala/Trino/Drill or Apache Hive on Tez
  - Low-latency responses for SQL on Hadoop with scalable big datasets
- Iterative processing:
  - ML algos: hold intermediate tasks in memory (compared to loading from disk)
  - Straightforward with Spark
- Stream processing:
  - Spark Streaming: Real-time, distributed computations - emit results to Hadoop
- Search:
  - Solr engine on Hadoop cluster, indexing documents as they are added to HDFS, and serving search queries from indexes stored in HDFS


13
9 11
16
1 4
9 10
11 12
13 15
16 20 25

More info

- Why can’t we use databases with lots of disks to do large-scale analysis? Why is Hadoop needed?
- Answer: Seek time is improving more slowly than transfer rate
  - Longer to R/W large portions of data than streaming it at transfer rate.
- For updating small portions of DB records, a traditional B-Tree (data structure of RDBMS that is limited by seek rate) works well.
- For updating majority of the DB, BTree is less efficient than MapReduce which uses Sort/Merge to rebuild the database

Storage Blues

Table 1-1. RDBMS compared to MapReduce

	Traditional RDBMS	MapReduce
Data size	Gigabytes	Petabytes
Access	Interactive and batch	Batch
Updates	Read and write many times	Write once, read many times
Transactions	ACID	None
Structure	Schema-on-write	Schema-on-read
Integrity	High	Low
Scaling	Nonlinear	Linear

Blues

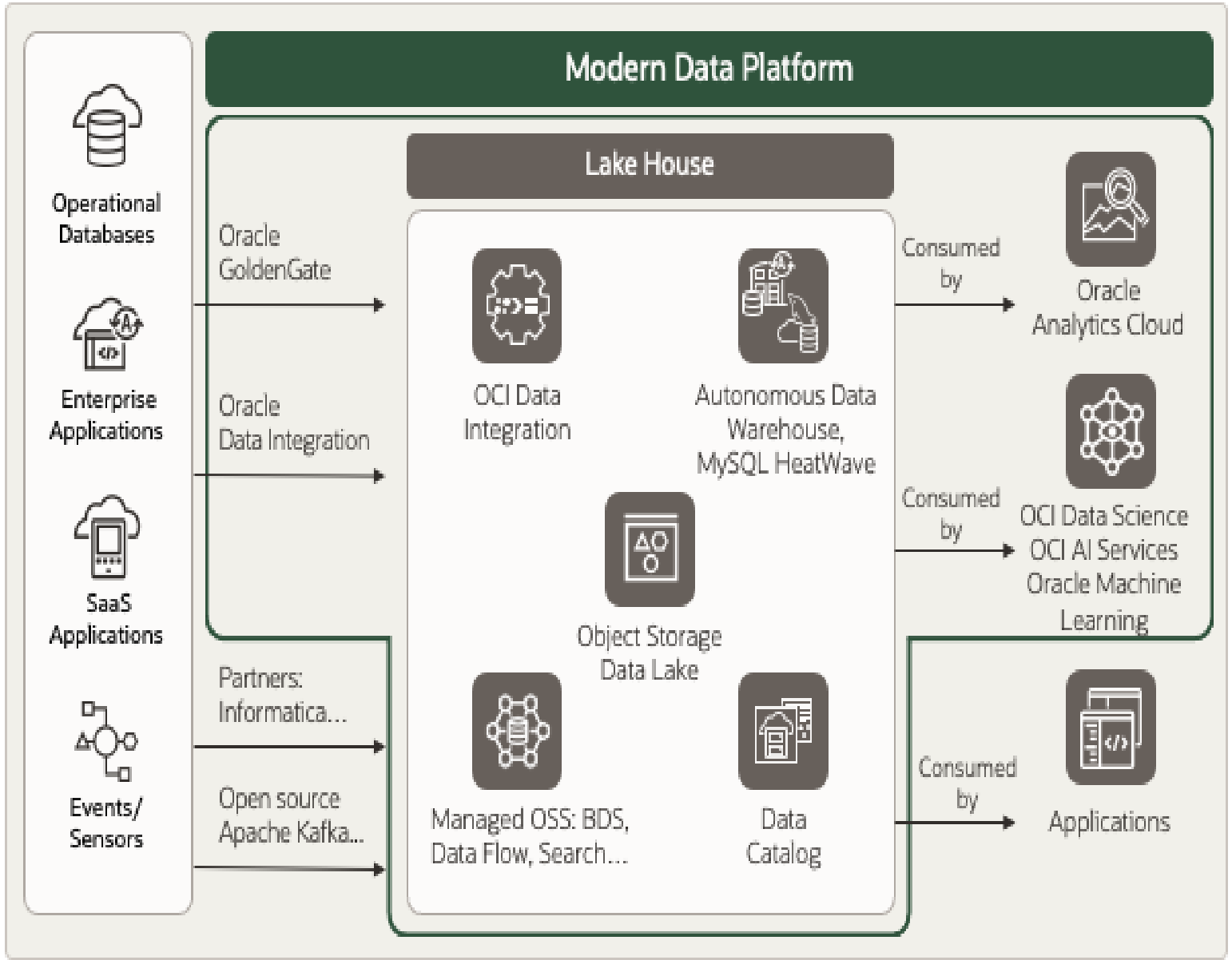
https://www.cs.utexas.edu/~rossbach/cs378/papers/dewitt08blog-mapreduce-backwards.pdf

- In January 2007: David J. DeWitt and Michael Stonebraker caused a stir by publishing “MapReduce: A major step backwards,”
- They criticized MapReduce for being a poor substitute for relational databases.
- Many commentators argued that it was a false comparison: “Databases are hammers; MapReduce is a screwdriver”
- DeWitt and Stonebraker followed up with “MapReduce II,” where they addressed the main topics brought up by others.

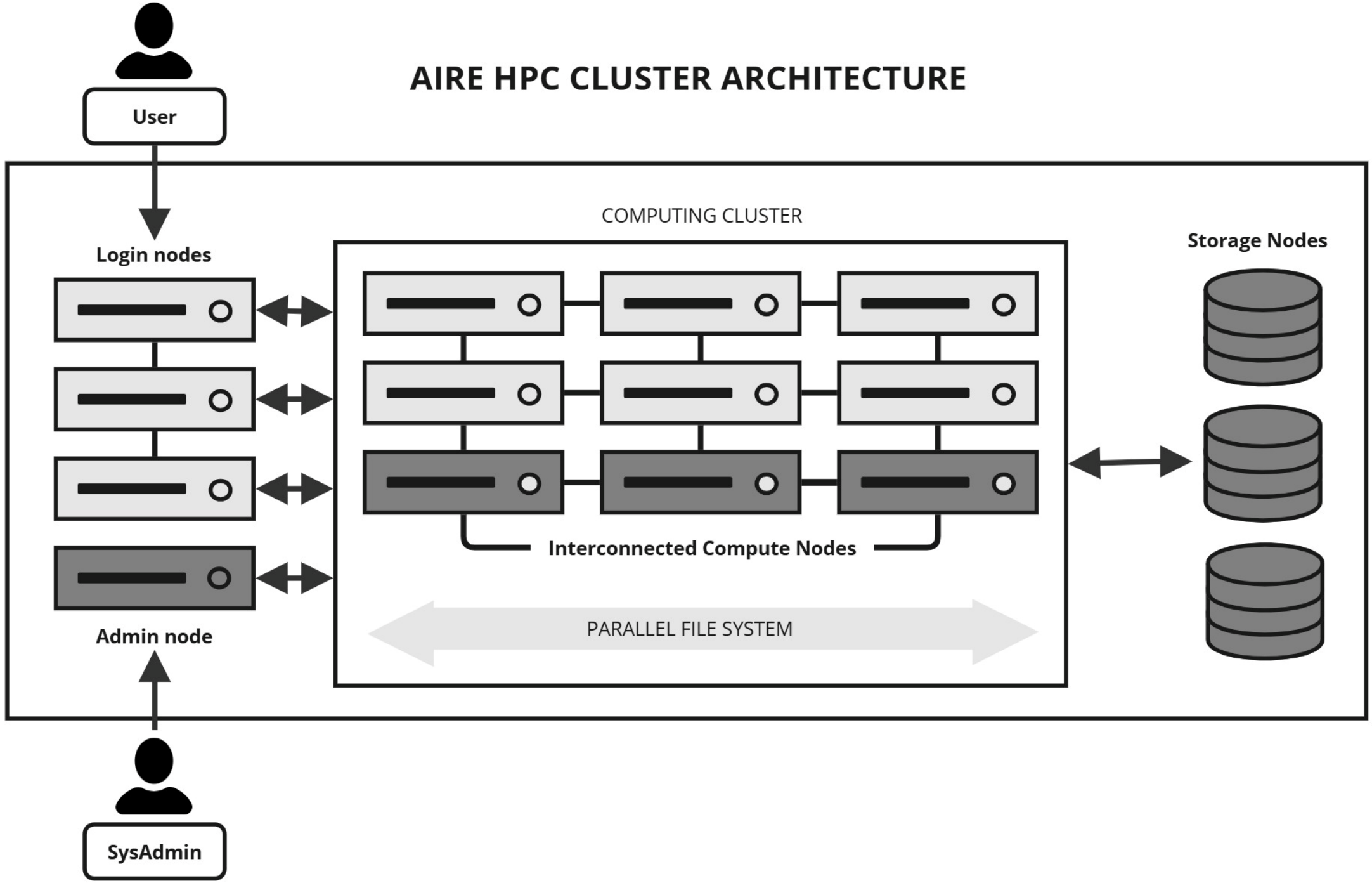
https://scienceblogs.com/goodmath/2008/01/22/databases-are-hammers-mapreduc

Lets change

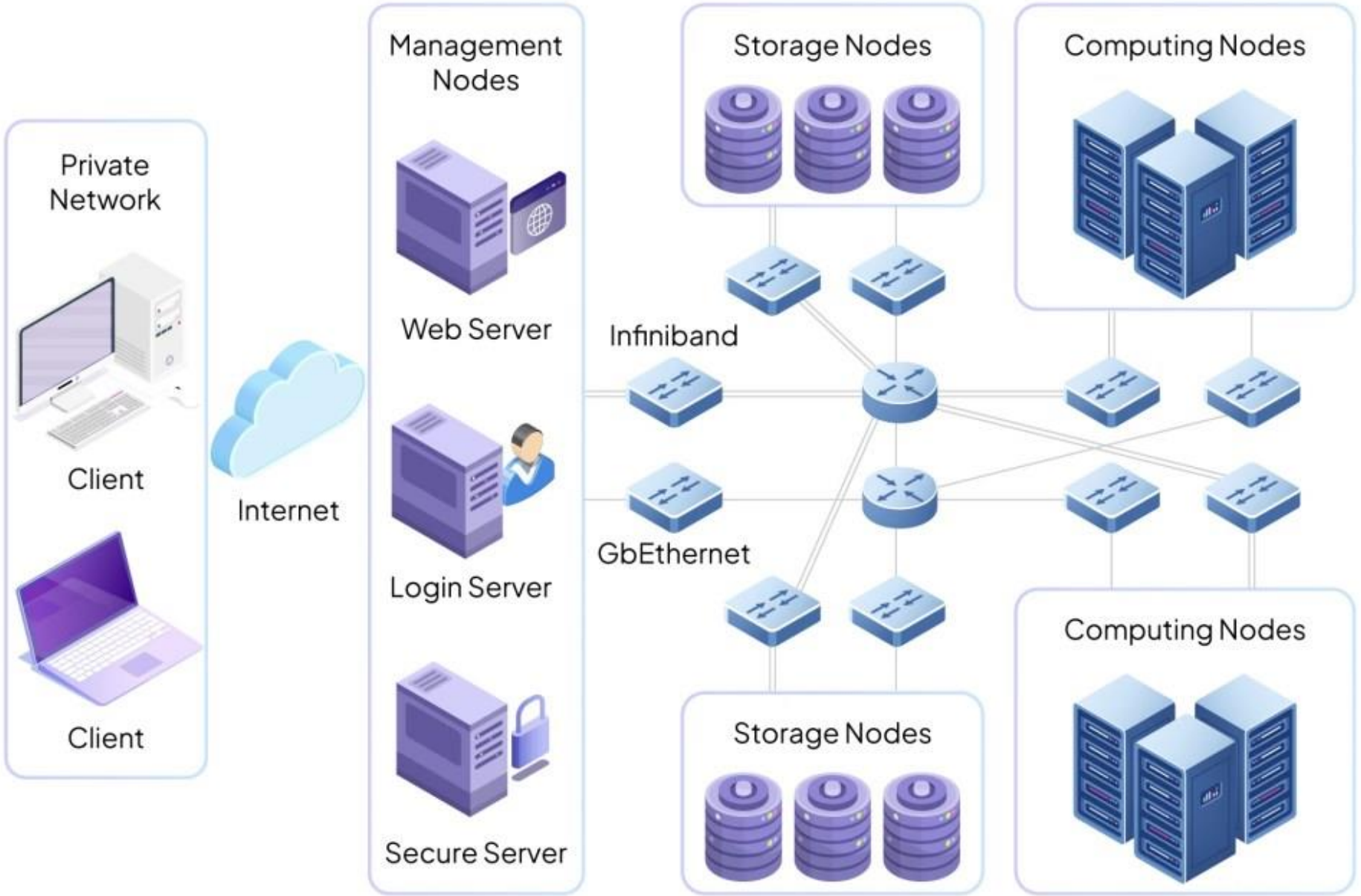
- RDBMs now incorporate some of ideas from Hadoop
- Hadoop/Hive are becoming more interactive (by moving away from MapReduce) and adding features like indexes and transactions that make them look more and more like traditional RDBMSs.

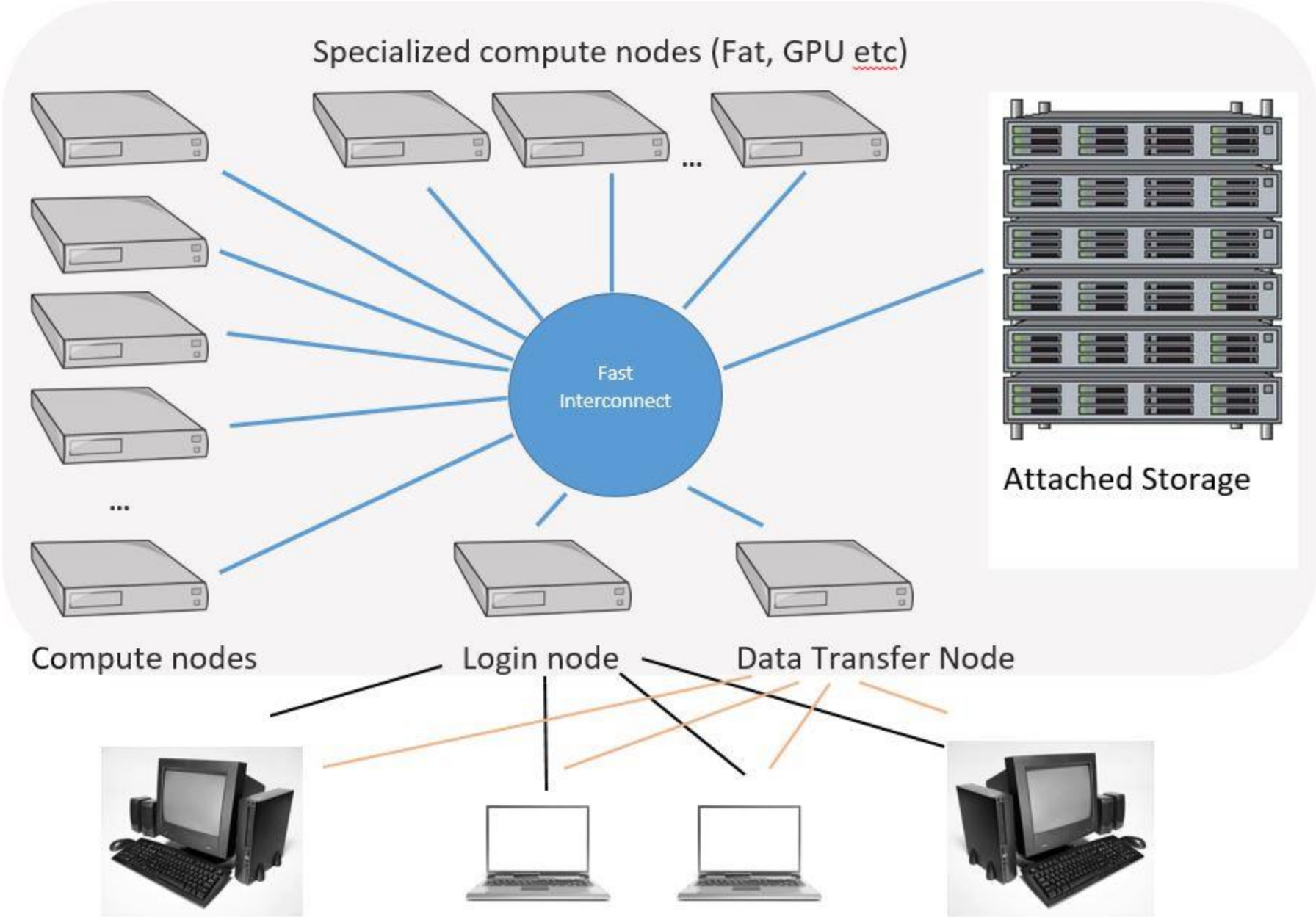


AIRE HPC CLUSTER ARCHITECTURE



HPC Infrastructure



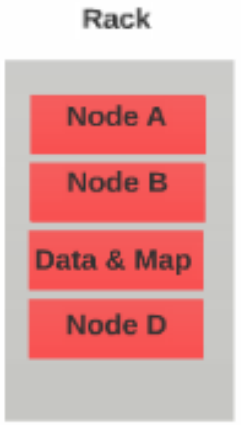

Specialized compute nodes (Fat, GPU etc)
...
Fast
Interconnect
...
Attached Storage
Compute nodes
Login node
Data Transfer Node

HPC

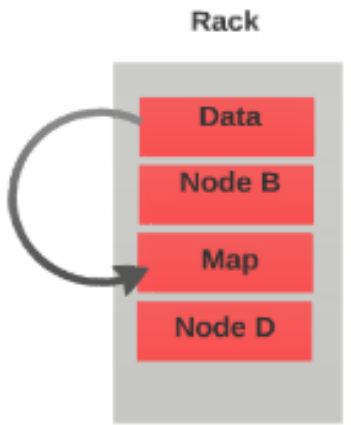
- HPC: does large-scale data processing: Message Passing Interface (MPI).
- HPC: distribute work across a cluster of machines, which access a shared filesystem, hosted by a storage area network (SAN).
- Works well for compute-intensive jobs
- Problem: nodes need to access larger data volumes (hundreds of gigabytes)
- Since the network bandwidth is the bottleneck and compute nodes become idle.

Data Locality

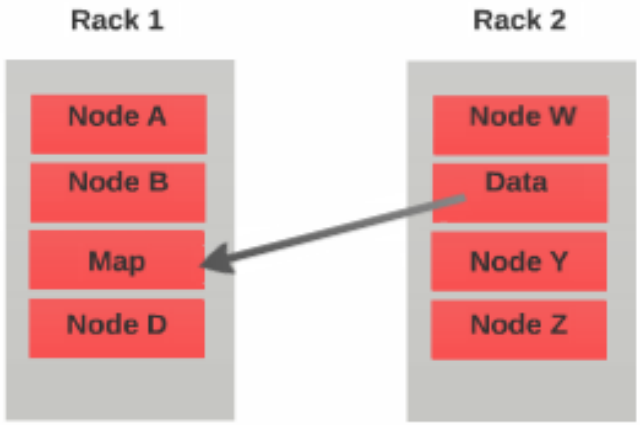
Data Local

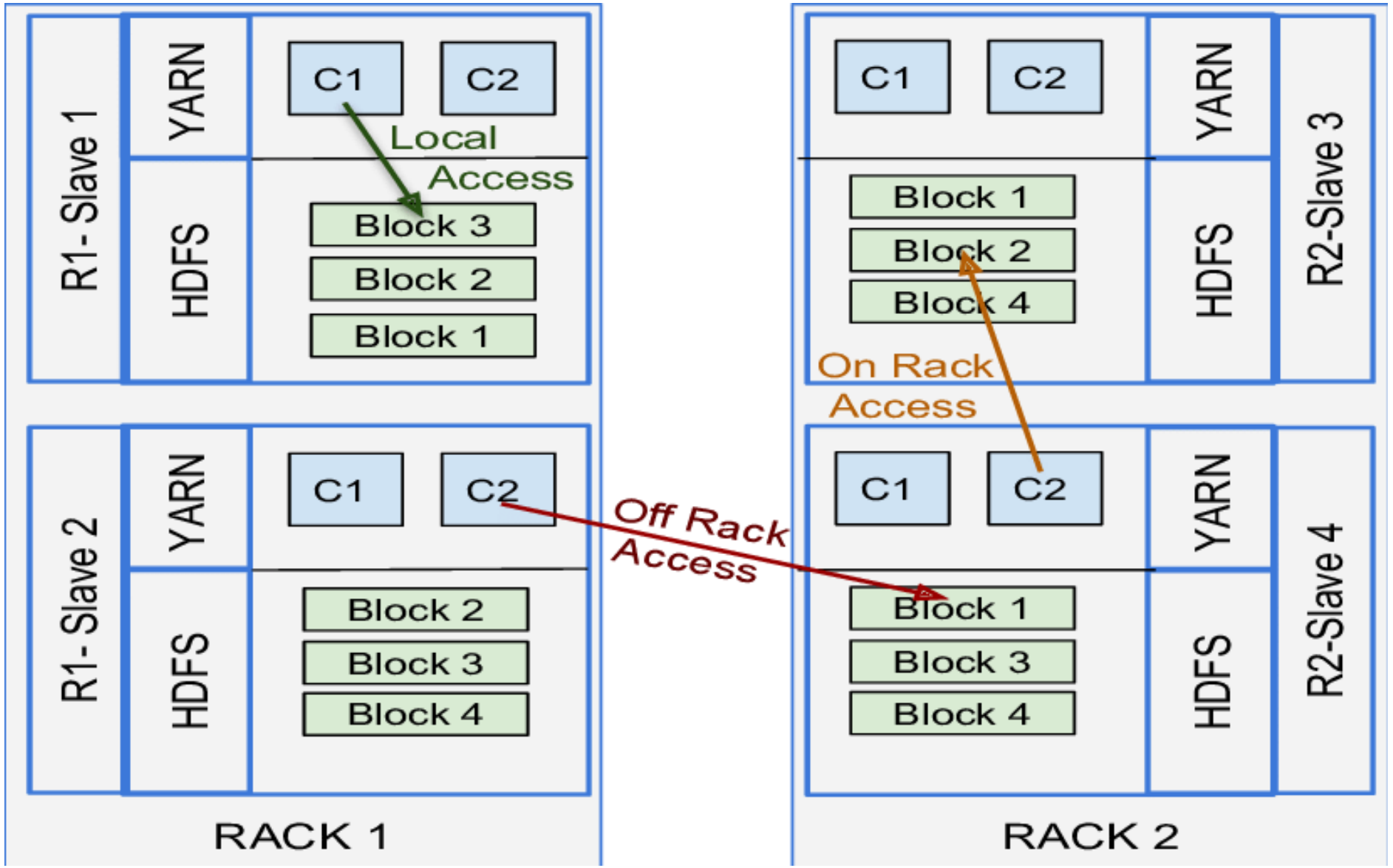


Rack Local



Different Rack




R1- Slave 1
YARN
HDFS
C1
C2
Local
Access
Block 3
Block 2
Block 1
R1- Slave 2
YARN
HDFS
C1
C2
Block 2
Block 3
Block 4
RACK 1
Off Rack
Access
C1
C2
YARN
HDFS
R2-Slave 3
Block 1
Block 2
Block 4
On Rack
Access
C1
C2
YARN
HDFS
R2-Slave 4
Block 1
Block 3
Block 4
RACK 2

HPC

- Hadoop co-locates data with computation: data access is fast because it is local.
- This is data locality: heart of data processing in Hadoop and is the reason for its good performance.
- Recognizes that network bandwidth is the most precious resource
- Because it is easy to saturate network links by copying data around
- Hadoop goes to great lengths to conserve it by explicitly modeling network topology.

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(&argc, &argv);
    // Get the number of processes
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);
    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);
    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);
    // Print a hello world message from each process
    printf("Hello world from processor %s, rank %d out of %d
processors\n",
            processor_name, world_rank, world_size);

    // Finalize the MPI environment
    MPI_Finalize();

    return 0;
}
```

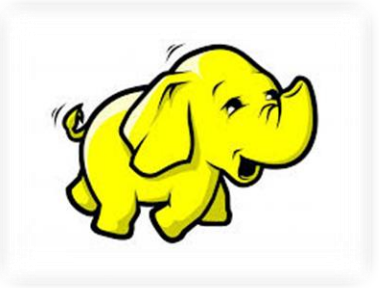
MPI Blues

- MPI gives great control to programmers
- But requires explicit handling of data flow, exposed via low-level C routines and sockets – also higher-level algos for analyses.
- Processing in Hadoop operates only at the higher level: the programmer thinks in terms of the data model (such as key-value pairs for MapReduce), while the data flow remains implicit.

```
#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize MPI
    MPI_Init(&argc, &argv);
    // Get the rank (ID) of the process
    int rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int data;
    if (rank == 0) {
        // Process 0 sends data
        data = 100;  // Some data to send
        printf("Process 0: Sending data %d to process 1\n", data);
        MPI_Send(&data, 1, MPI_INT, 1, 0, MPI_COMM_WORLD); // Send data to process 1
    }
    else if (rank == 1) {
        // Process 1 receives data
        MPI_Recv(&data, 1, MPI_INT, 0, 0, MPI_COMM_WORLD, MPI_STATUS_IGNORE); // Receive data from
process 0
        printf("Process 1: Received data %d from process 0\n", data);
    }
    MPI_Finalize();
    return 0;
}
```

History of Hadoop



- Created by Doug Cutting, creator of Apache Lucene (text search library)
- Origins in Apache Nutch, an open-source web search engine (part of Lucene)
- Hadoop: not an acronym:
  - “The name my kid gave a stuffed yellow elephant. Short, relatively easy to spell and pronounce, meaningless, and not used elsewhere: those are my naming criteria. Kids are good at generating such. Googol is a kid’s term” - Cutting
- Building a web-search engine from scratch was an ambitious goal: crawl and index websites complex to write - challenge to run without ops team
- It’s expensive: Mike Cafarella and Cutting estimated a system supporting a one-billion-page index would cost around $500,000 in hardware, with a monthly running cost of $30,000

Hadoop History

- Nutch started in 2002: working crawler and search system quickly emerged.
- However, architecture wouldn’t scale to the billions of pages on the Web.
- Help was at hand: Paper in 2003 - architecture of Google’s distributed filesystem, called GFS (used in production).
- GFS, or something like it, would solve their storage needs for the very large files generated as a part of the web crawl and indexing process.
- Would free up time being spent on admin tasks (managing storage nodes)
- In 2004, Nutch’s developers set about writing an open-source implementation, the Nutch Distributed Filesystem (NDFS).

https://static.googleusercontent.com/media/research.google.com/en//archive/gfs-sosp2003.pdf

https://static.googleusercontent.com/media/research.google.com/en//archive/mapreduce-osdi04.pdf

Hadoop History

- 2004: Google published paper that introduced MapReduce
- 2005: Nutch developers had a working MR in Nutch - all major Nutch algos had been ported to run using MapReduce and NDFS.
- 2006: form an independent subproject of Lucene called Hadoop.
- Cutting joined Yahoo!, which provided a dedicated team and the resources to turn Hadoop into a system that ran at web scale (see the following sidebar).
- February 2008: Yahoo! announced that its production search index was being generated by a 10,000-core Hadoop cluster.

Hadoop History

- Yahoo! Search consists of four primary components: the Crawler, which downloads pages from web servers; the WebMap, which builds a graph of the known Web; the Indexer, which builds a reverse index to the best pages; and the Runtime, which answers users' queries.
- The WebMap is a graph that consists of roughly 1 trillion (1012) edges, each representing a web link, and 100 billion (1011) nodes, each representing distinct URLs.
- Creating and analyzing such a large graph requires a large number of computers running for many days.
- In early 2005, the infrastructure for the WebMap, named Dreadnaught, needed to be redesigned to scale up to more nodes. Dreadnaught had successfully scaled from 20 to 600 nodes, but required a complete redesign to scale out further.
- Dreadnaught is similar to MapReduce

Hadoop History

- In January 2008: Hadoop was made its own top-level project at Apache
- Yahoo!, Last.fm, Facebook, and the New York Times.
- In one well-publicized feat, the New York Times used Amazon's EC2 compute cloud to crunch through 4 terabytes of scanned archives from the paper, converting them to PDFs for the Web.
- The processing took less than 24 hours to run using 100 machines
- In April 2008, Hadoop broke a world record to become the fastest system to sort an entire terabyte of data. Running on a 910-node cluster, Hadoop sorted 1 terabyte in 209 seconds (just under 3.5 minutes), beating the previous year's winner of 297 seconds.

Trends

- 2014:
  - Databricks were joint winners of the Gray Sort benchmark.
  - A 207-node Spark cluster to sort 100 terabytes of data in 1,406 seconds, a rate of 4.27 terabytes per minute.
- Today, Hadoop is widely used in mainstream enterprises.
- Hadoop's role as a general-purpose storage and analysis platform for big data has been recognized by the industry - reflected in the number of products that use or incorporate Hadoop in some way
- Commercial Hadoop support: EMC, IBM, Microsoft, and Oracle, as well as from specialist Hadoop companies such as Cloudera, Hortonworks, and MapR.

Tool / Framework	Strengths & Typical Use Cases	Weaknesses / Limitations	Relative Adoption / Market Position
Hadoop	• Batch processing of very large datasets • Strong data storage with HDFS / replicated storage • Mature ecosystem (Hive, Pig, HBase, etc.) • Used for archival, ETL of logs, data lakes, offline analytics	• High latency; many operations involve disk I/O • More operational overhead (managing clusters, storage, etc.) • Less suitable for real-time	• Still used heavily in enterprises with legacy/large-scale data workloads. • ~64% of **new** big data deployments are now cloud-based. • Hadoop distributions (Cloudera, AWS, Azure, GCP) still maintain a non-trivial share. e.g. Hadoop distribution market estimated share: Cloudera (18-22%), AWS (~14-18%), Google Dataproc (~12-16%), Azure HDInsight (~10-14%) etc.
Apache Spark	• Much faster for many workloads (in-memory computation; DAGs) • Good for both batch and streaming, machine learning • More interactive: Spark SQL, notebooks, etc.	• Needs more memory; for very large datasets memory management can be tricky. • Can be costlier if always needing high memory & compute • Cluster resource management still a challenge	• Spark has high “mindshare”. • PeerSpot data shows Spark with ~17-20% mindshare in BDA tools: Cloudera for Hadoop around ~22%. • Many cloud offerings (EMR, Databricks, etc.) are optimized for Spark workloads.
Apache Flink	• Strong real-time and stream processing capabilities • Low latency, good for event processing • Increasingly used in hybrid/cloud architectures	• Ecosystem not as mature as Hadoop + Spark • Learning curve	• In hybrid and cloud evaluations, Flink often outperforms Hadoop in time and resource efficiency for streaming or combined workloads.
Cloud (e.g. BigQuery, Redshift, Snowflake, Databricks Lakehouse)	• Fully managed; minimal overhead • Instant scaling, serverless, or managed clusters • Excellent for analytics, ad-hoc	• Less control over underlying cluster / storage (can mean cost surprises) • Sometimes vendor lock-in	• BigQuery / Snowflake /Databricks etc. are increasingly preferred for analytics workloads especially when teams want fast time-to-insight with managed services. • Many orgs reduce their Hadoop/MapReduce use and shift toward Spark + object storage or directly to these managed services.