Welcome to

Big Data and Data Science Technologies

Agenda

- → Who am i?
- → Who are you?
- → What is this course about?
- → Introducing perspectives
  - → Non-Functional Requirements
  - → Horizontal Scaling vs Vertical Scaling
  - → The practical history of storage solutions
  - → Modern large-scale deployments in the future

SDU

Who am i?

- About me: Jakob Hviid, from 1982, Gamer (627 games on just Steam), D&D, Professionel Nerd, love gadgets (everything is better with bluetooth or wifi).
- Currently: Assistant Professor (SDU) and Senior Architect (Energinet)
- Education: PhD in Software Engineering, M.Sc. in Software Engineering, B.Sc. Web Development, A.P Degree in Computer Science.
- Industry Experience
  - Previous career: Typically server and network architecture related work
  - Current career: Researcher, Teacher, Software Architect, Self Employed (SDU, Energinet, Hviid Development)
  - Other places: Esoft, SDU, Scannet, Unik System Design, Carlsberg / Tuborg / Coca Cola and more.
- Teaching:
  - Data Management (2 semester bachelor)
  - Big Data and Data Science Technologies (1 semester master)

SDU


WHO ARE YOU?

Course Introduction

B.Sc. != M.Sc.

- → At a B.Sc. The lecturer explains
  - → What the technologies are
  - → How they are used
  - → Why we are using them
- → At a M.Sc. The lecturer explains
  - → What technologies exist
  - → Why they are needed
  - → How they fit together
- → At a M.Sc. The lecturer does NOT explain
  - → How they are used
  - → How to write it
  - → Where to find all the information needed

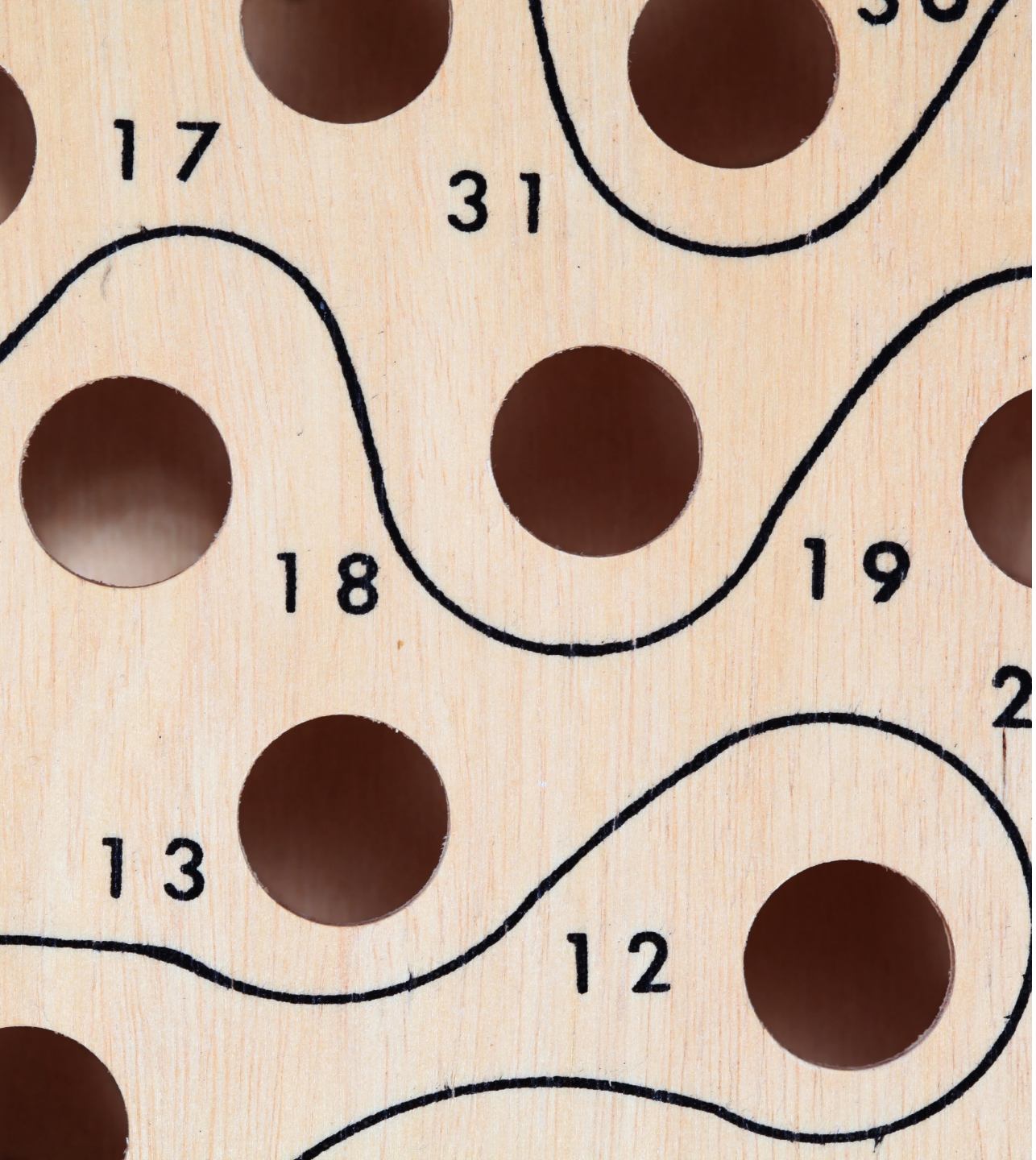
SDU


Beware!
Much of the knowledge is not in the books! Come to the lectures!

Skills

- → Analyze and implement a big data pipeline, including:
  - → Dataset selection
  - → Data ingestion
  - → Distributed data storage
  - → Distributed data streaming
  - → Distributed data processing

SDU



Competences

→ After successful completion of the course, the student is able to:

- → Design large scale storage for industrial, data-intensive applications
- → Design solutions for processing large data streams
- → Implement distributed data processing programs;
- → Implement complex analytical queries in big data query languages;
- → Configure, execute and debug programs over the Big data frameworks.

Course Structure

How would you solve this?

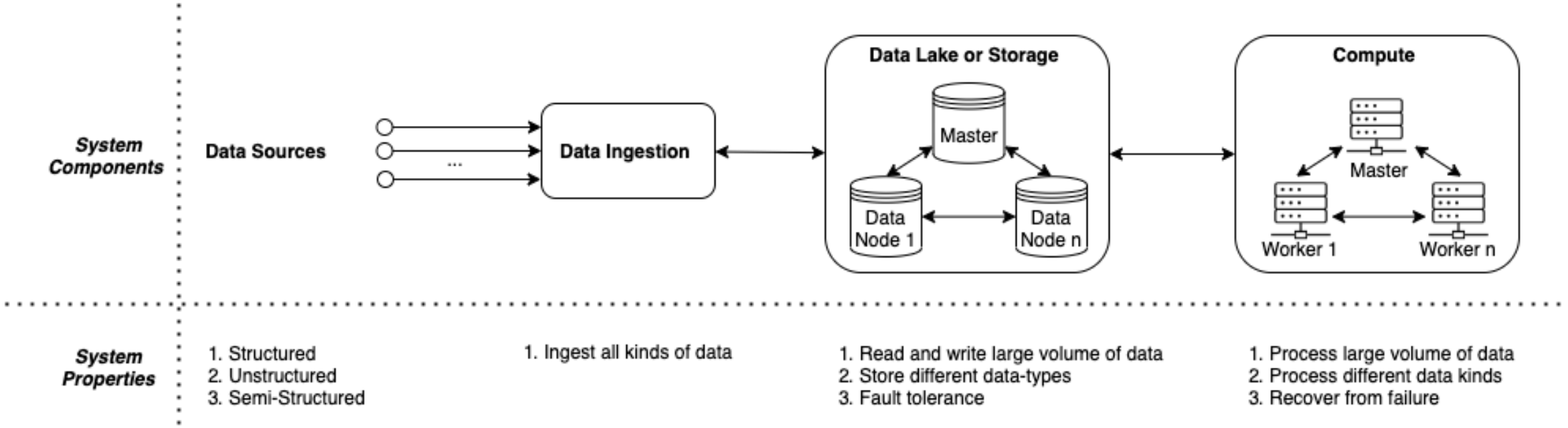
You have to receive 1 PB of data

Process it on the fly while you receive it

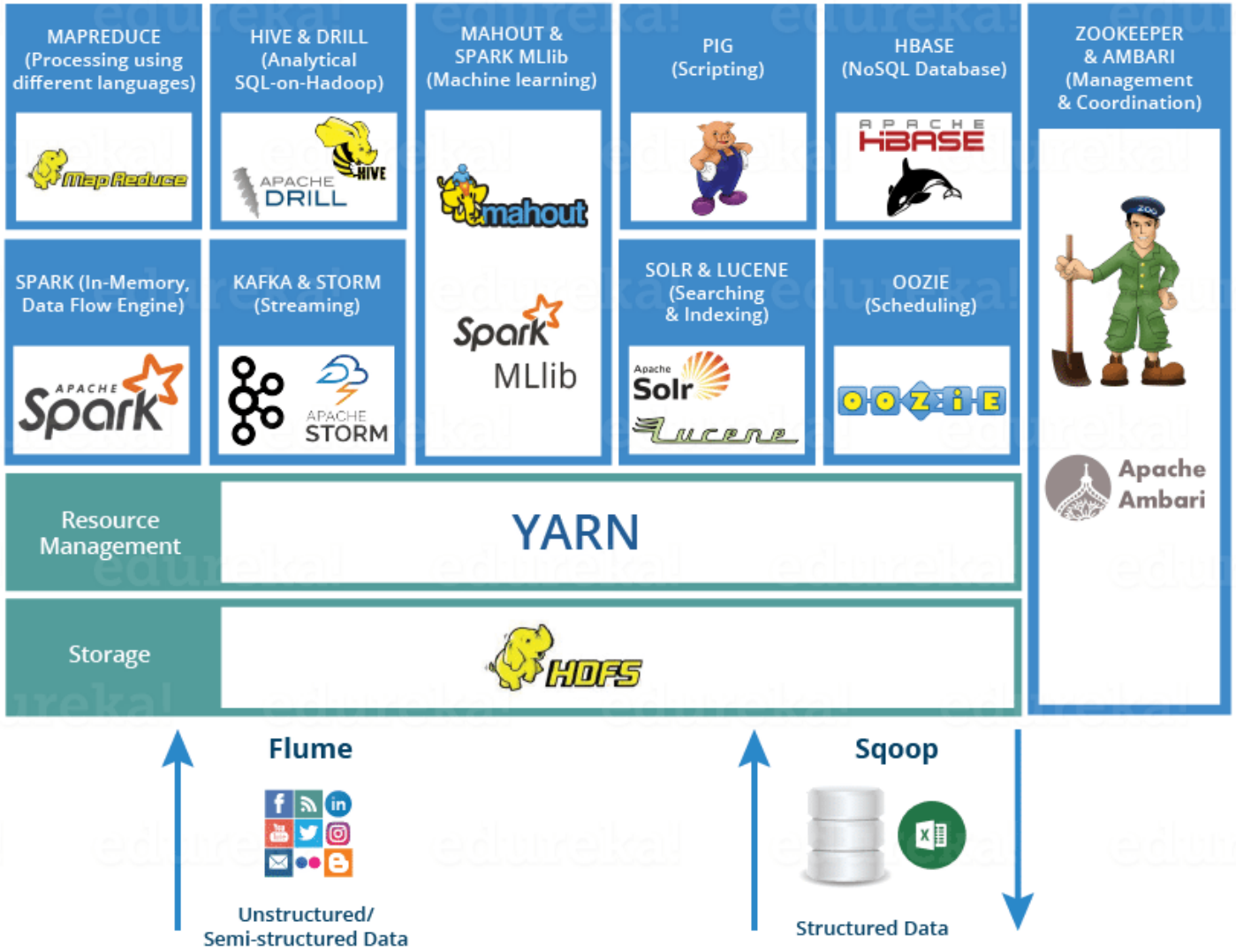
You have to store all of the data

You have to be able to run analytics on this data with a response time of maximum 3 min.

One way to solve this problem



The Apache Stack



SDU

Non-Functional Requirements

Just what is a Non Functional Requirement?

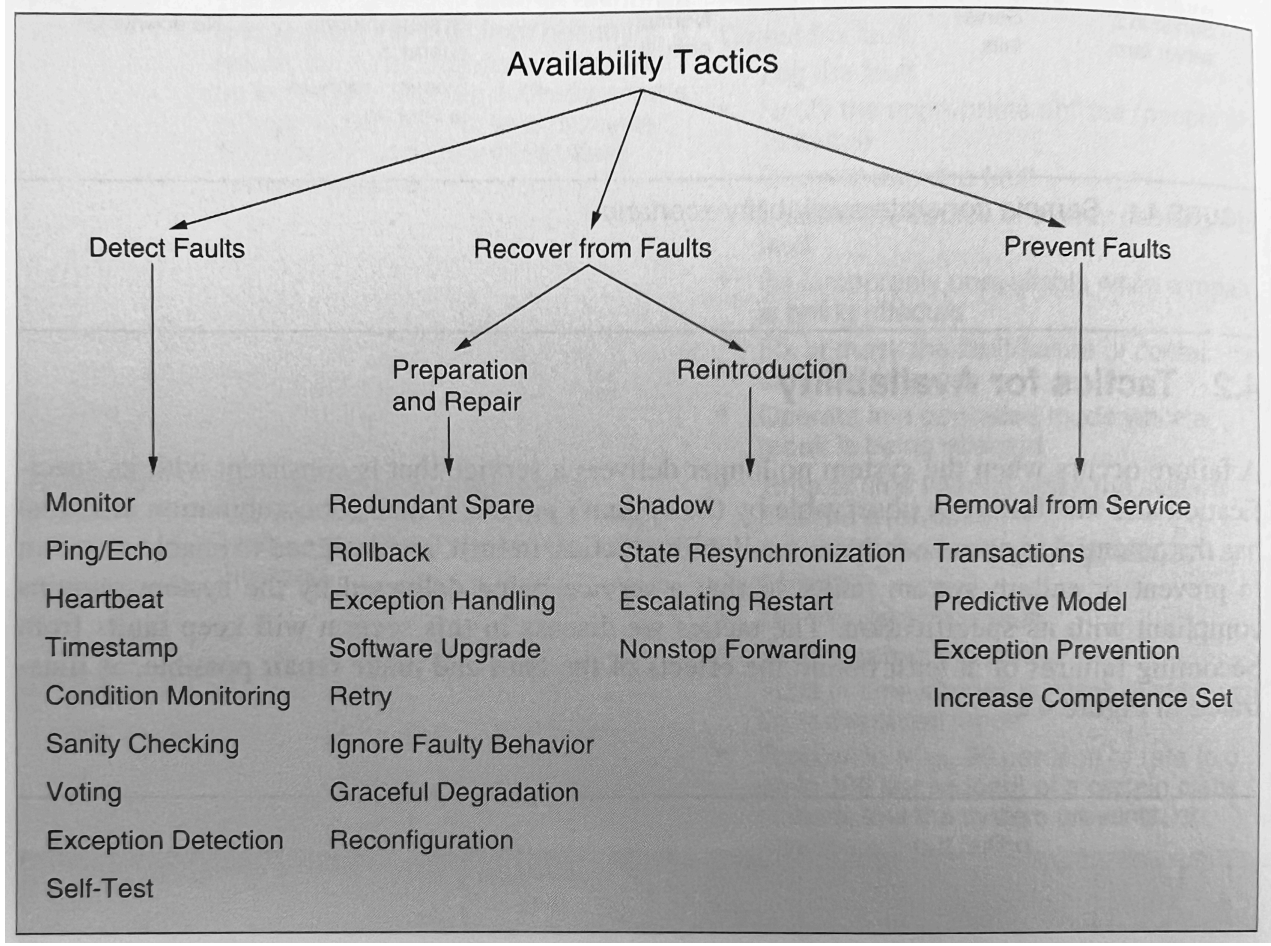
SDU

Non Functional Requirements

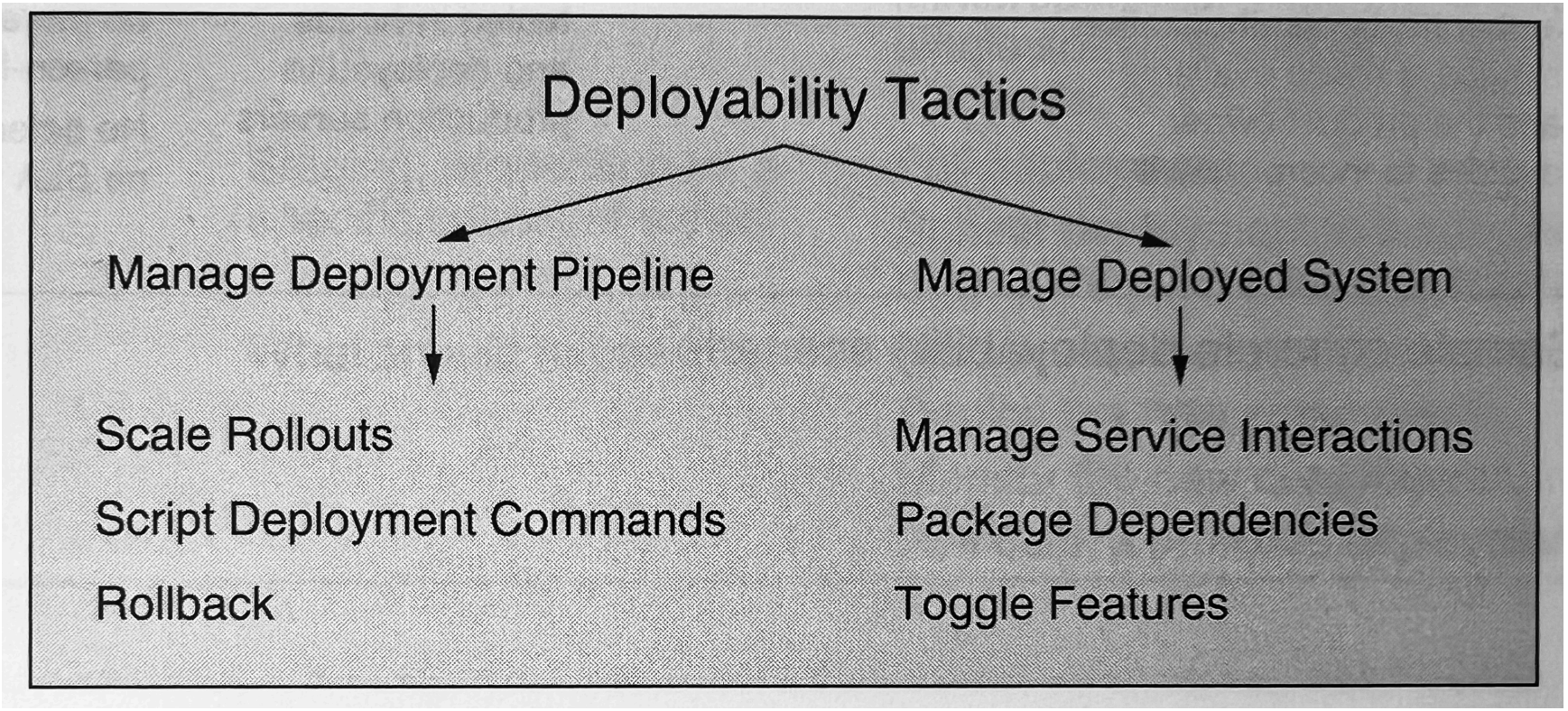
- → **Availability**
- → **Deployability**
- → Energy Efficiency
- → Integrability
- → Modifiability
- → **Performance**
- → Safety
- → Security
- → Traceability
- → Testability
- → Usability
- → Documentation

SDU

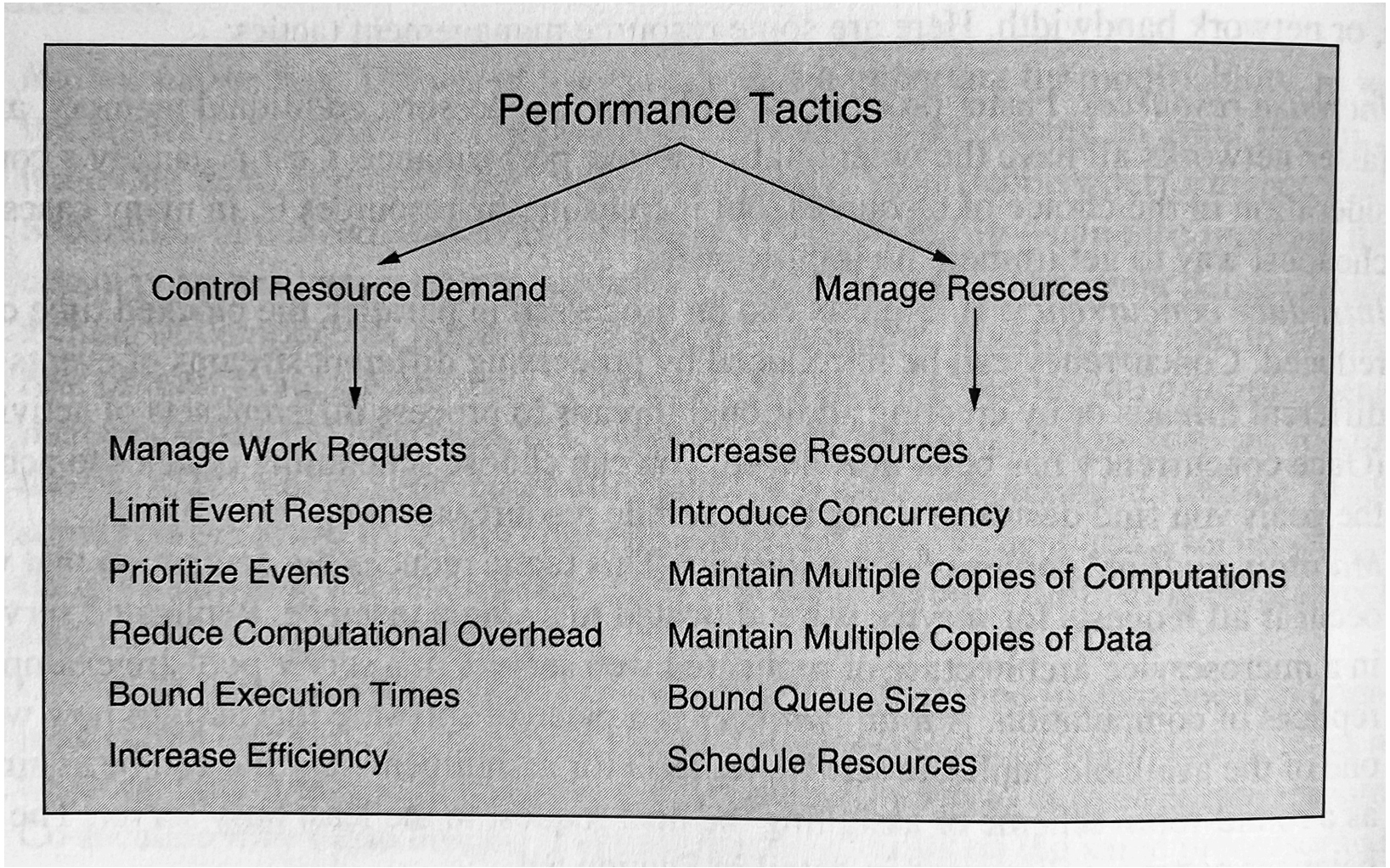
Availability Tactics



Deployability Tactics



Performance



RAID Levels

Hewlett Packard Enterprise

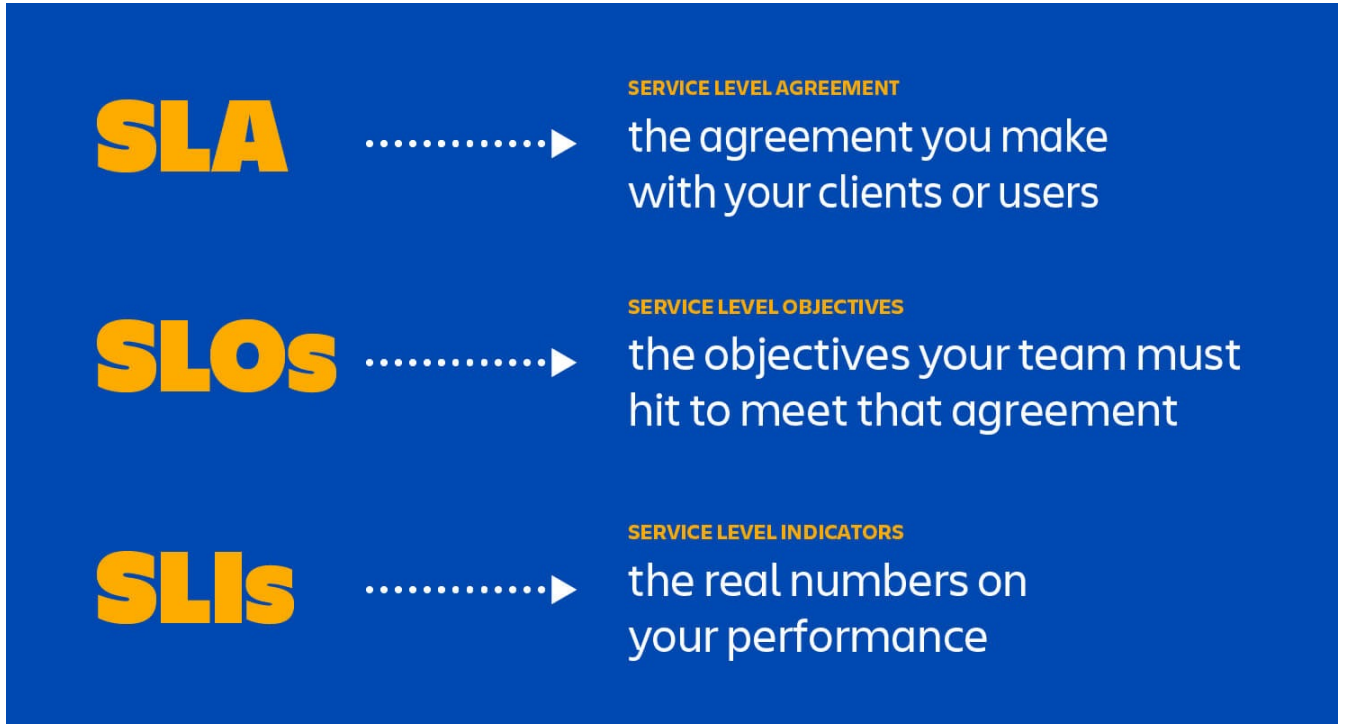
BREAKDOWN OF COMMON RAID LEVELS

RAID LEVEL	METHOD	HARDWARE / SOFTWARE	MINIMUM # OF DISKS	COMMON USAGE	PROS	CONS
JBOD	SPANNING		2	INCREASE CAPACITY	COST-EFFECTIVE STORAGE	NO PERFORMANCE OR SECURITY BENEFITS
0	STRIPING		2	HEAVY READ OPERATIONS	HIGH PERFORMANCE (SPEED)	DATA IS LOST IF ONE DISK FAILS
1	MIRRORING		2	STANDARD APP SERVERS	FAULT TOLERANCE, HIGH READ PERFORMANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/2)
5	STRIPING & PARITY		3	NORMAL FILE STORAGE & APP SERVERS	SPEED + FAULT TOLERANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/3)
6	STRIPING & DOUBLE PARITY		4	LARGE FILE STORAGE & APP SERVERS	EXTRA LEVEL OF REDUNDANCY, HIGH READ PERFORMANCE	LOW WRITE PERFORMANCE, REDUCED STORAGE (BY 2/5)
10 (1+0)	STRIPING & MIRRORING		4	HIGHLY UTILIZED DATABASE SERVERS	WRITE PERFORMANCE + STRONG FAULT TOLERANCE	REDUCED STORAGE (1/2), LIMITED SCALABILITY

SDU

Production Metrics and Concepts

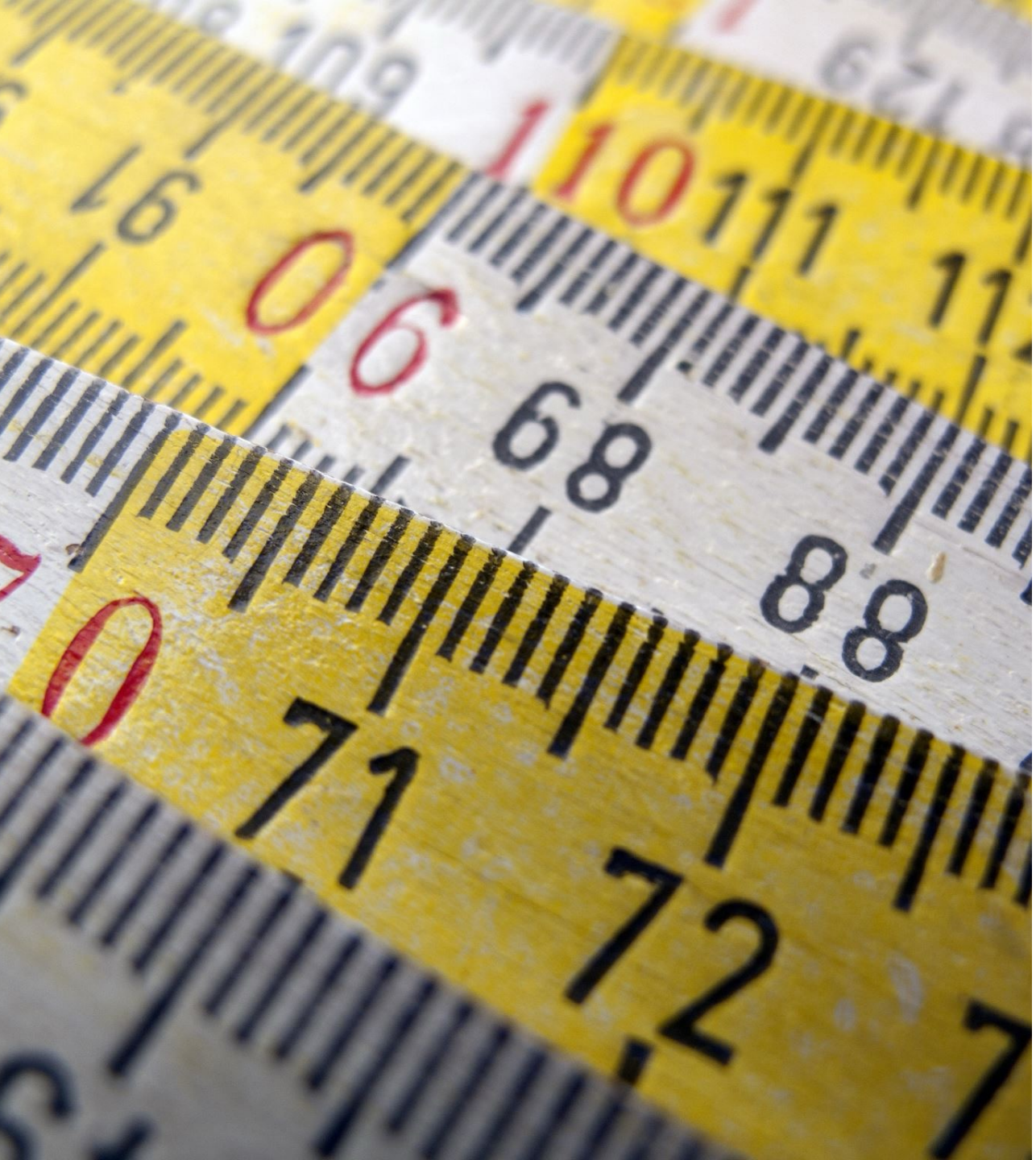
Severity Levels



What is this?

$$MTBF/(MTBF + MTTR)$$

SDU



Incident Metrics

- → MTBF: Mean Time Before Failure
- → MTTD: Mean Time to Detect
- → MTTR: Mean Time to Recovery / Repair / Response / Resolve
- → MTTA: Mean Time to Acknowledge

SDU

Error Budget

TABLE 4.1 System Availability Requirements

Availability	Downtime/90 Days	Downtime/Year
99.0%	21 hr, 36 min	3 days, 15.6 hr
99.9%	2 hr, 10 min	8 hr, 0 min, 46 sec
99.99%	12 min, 58 sec	52 min, 34 sec
99.999%	1 min, 18 sec	5 min, 15 sec
99.9999%	8 sec	32 sec

Severity Levels

Severity	Description	Examples
1	A critical incident with very high impact	• A customer-facing service, like Jira Cloud, is down for all customers • Confidentiality or privacy is breached • Customer data loss
2	A major incident with significant impact	• A customer-facing service is unavailable for a subset of customers • Core functionality (e.g. git push, issue create) is significantly impacted
3	A minor incident with low impact	• A minor inconvenience to customers, workaround available • Usable performance degradation

Big data;
What and why?

Big Data is defined in three Vs (yes there are more)

1. Volume: large amount of data.
2. Variety: the data comes in different forms (or sources) including traditional databases, images, documents and complex records.
3. Velocity: the content of the data is constantly changing through the absorption of complementary data collections, through the introduction of previously archived data or legacy collections and from streamed data arriving from multiple source.

Big Data Facts

Every 2 days we create as much data (5 Exabytes) as we did from the beginning of time until 2003.

Poor data across businesses and the government costs the U.S. economy $3.1 trillion dollars a year

Big data at Facebook

→ Based on company statistics [W1] (2012) **per day**:

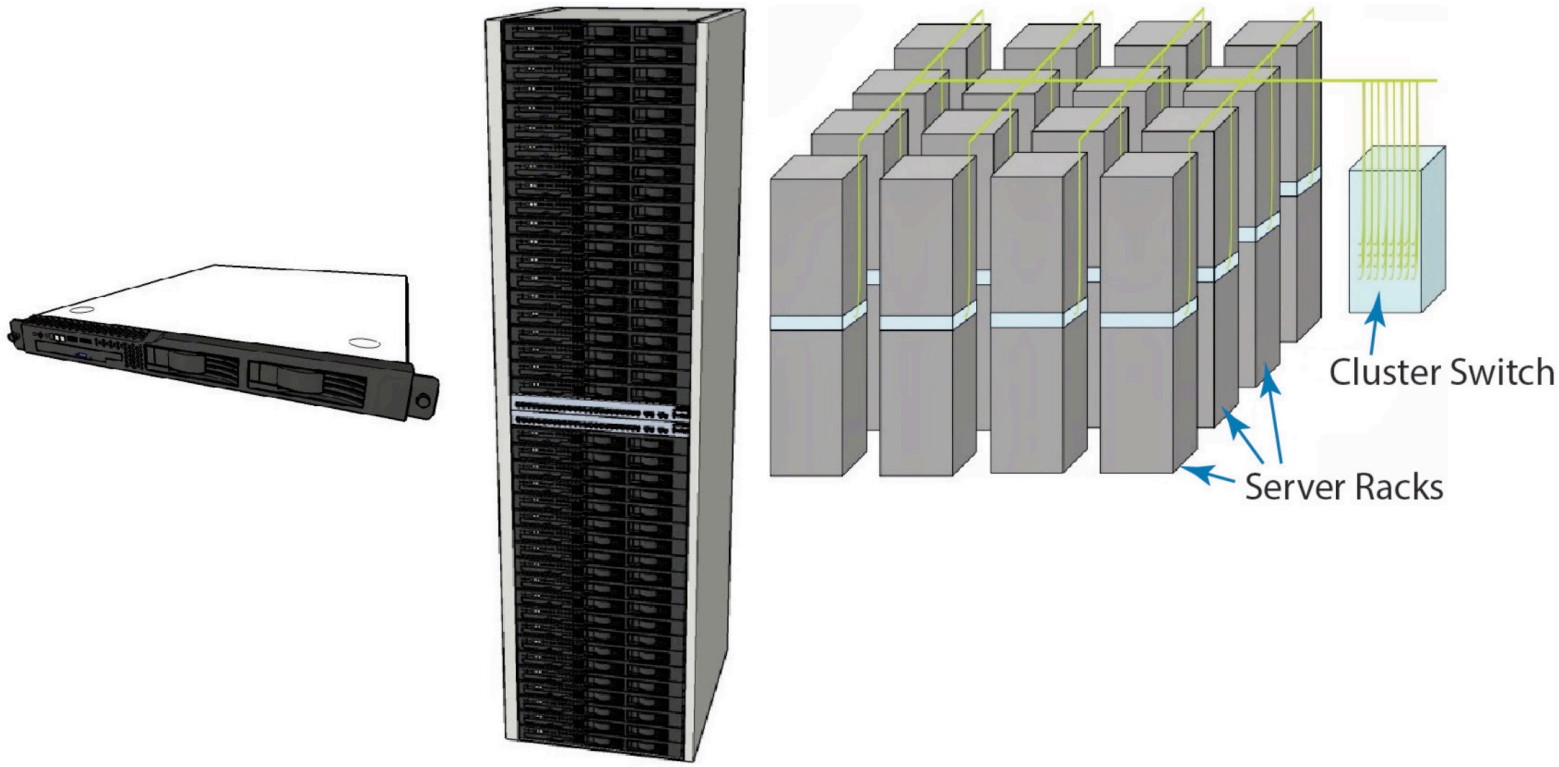
- 500+ TB of new data into the databases
- 2.5 billion content items shared
- This includes status updates, wall posts, photos, videos, comments
- 2.7 billion likes
- 300 million photos uploaded

Big Data Storage and Processing

To store and process this big-data, we need a new software stack:

- distributed, over **networked computing clusters**
- new **Distributed File System** (DFS)
- ... which is extremely fault-tolerant
- new **programming frameworks** over a DFS

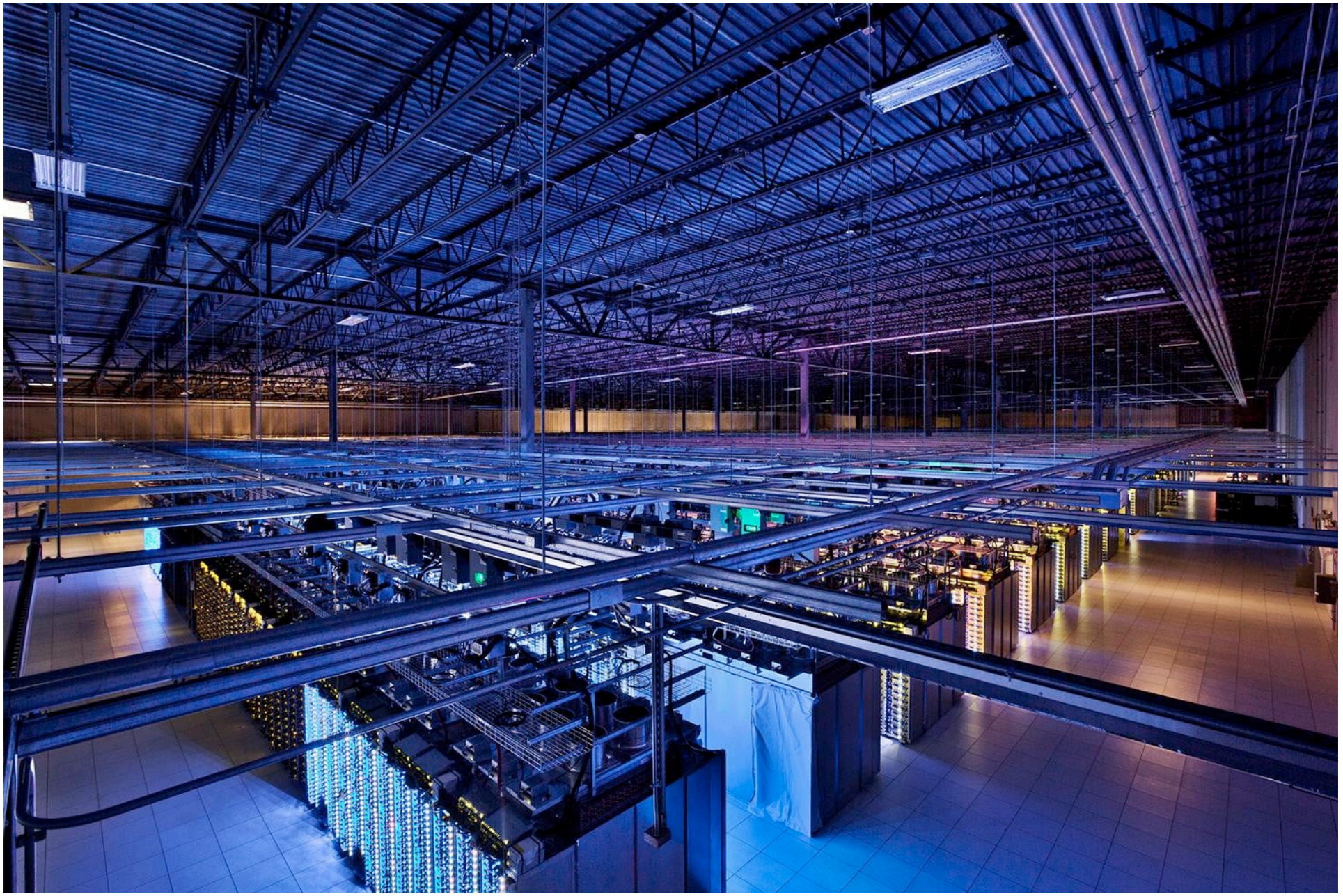
The cluster is the new computer



(left) commodity server

(middle) rack with Ethernet switch and 40-80 servers

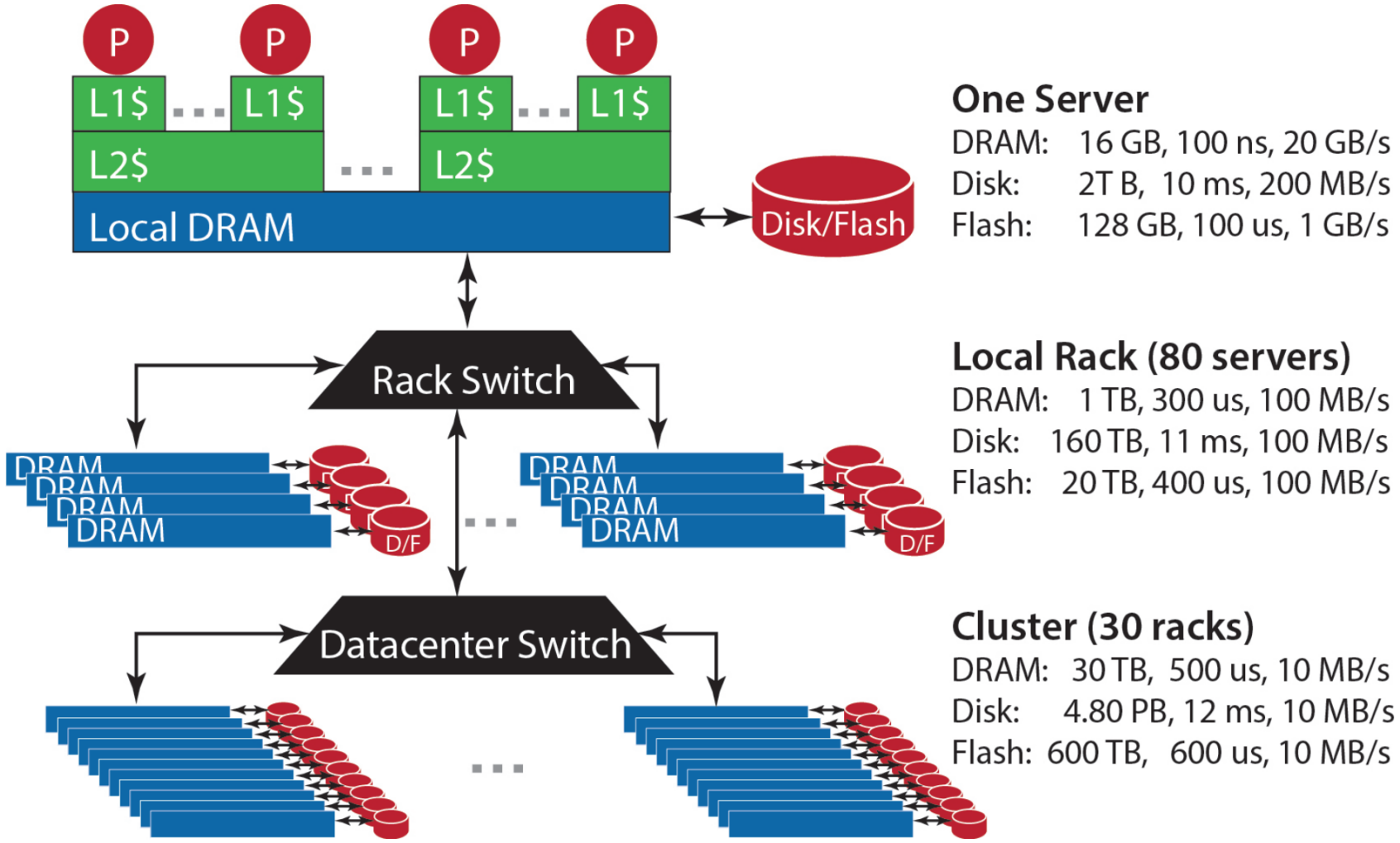
(right) cluster with a cluster-level Ethernet switch or router



SDU

(Google data center at Council Bluffs, Iowa [W2])

Cluster memory: size, latency and bandwidth



One Server
DRAM: 16 GB, 100 ns, 20 GB/s
Disk: 2T B, 10 ms, 200 MB/s
Flash: 128 GB, 100 us, 1 GB/s

Local Rack (80 servers)
DRAM: 1 TB, 300 us, 100 MB/s
Disk: 160 TB, 11 ms, 100 MB/s
Flash: 20 TB, 400 us, 100 MB/s

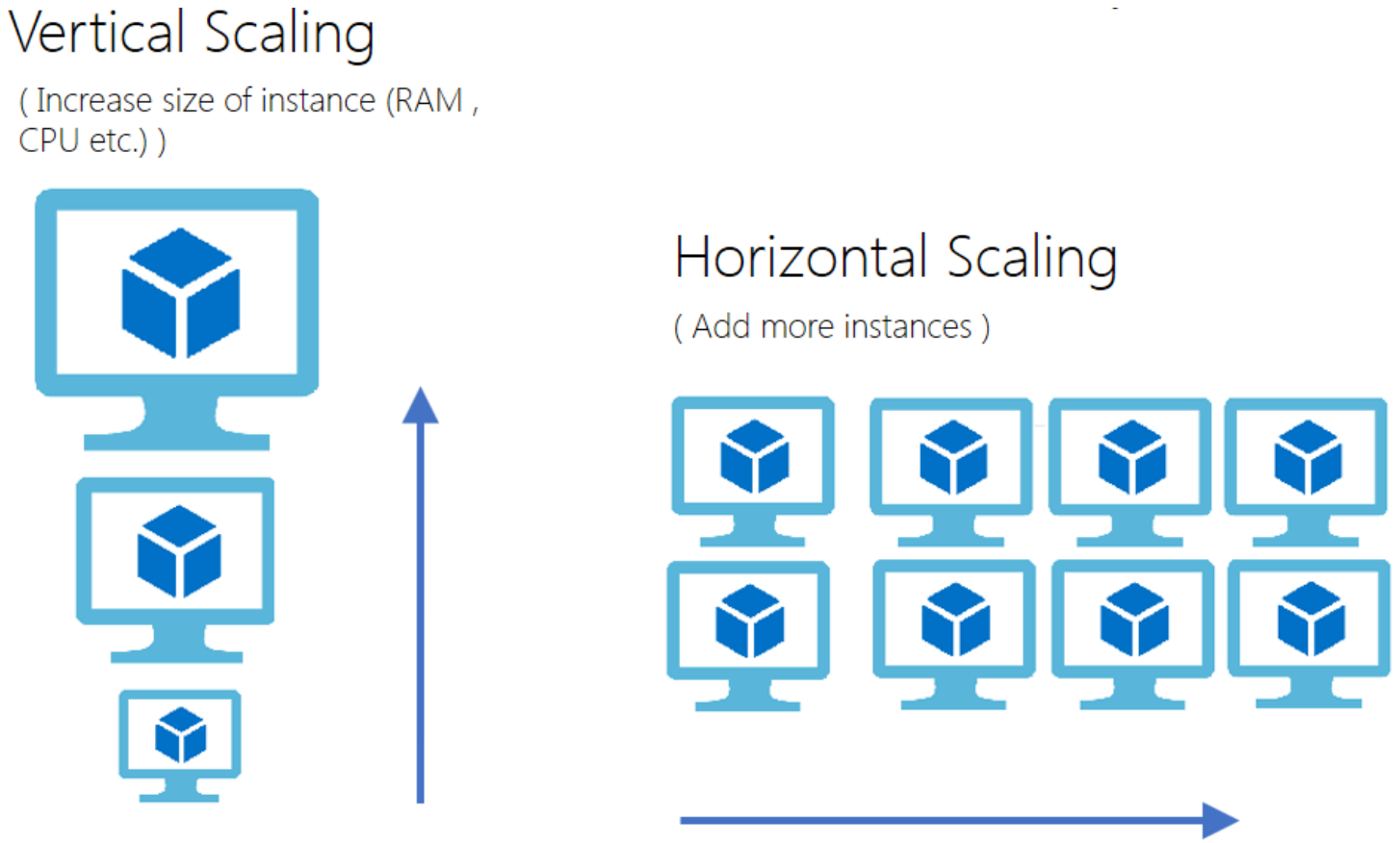
Cluster (30 racks)
DRAM: 30 TB, 500 us, 10 MB/s
Disk: 4.80 PB, 12 ms, 10 MB/s
Flash: 600 TB, 600 us, 10 MB/s

SDU

(Google numbers [W3]) – Example, numbers are outdated

Horizontal vs Vertical Scaling

Horizontal vs Vertical Scaling



- → Vertical
  - → Increasingly Expensive to get larger hardware
  - → Requires no code changes
- → Horizontal
  - → Less expensive hardware
  - → Allows for large-scale systems
  - → Need to change software for a distributed architecture
    - → Leads to more complex code
  - → Need for Load Balancing

→ Do you know examples of these?

Server Rooms, Servers, and Storage

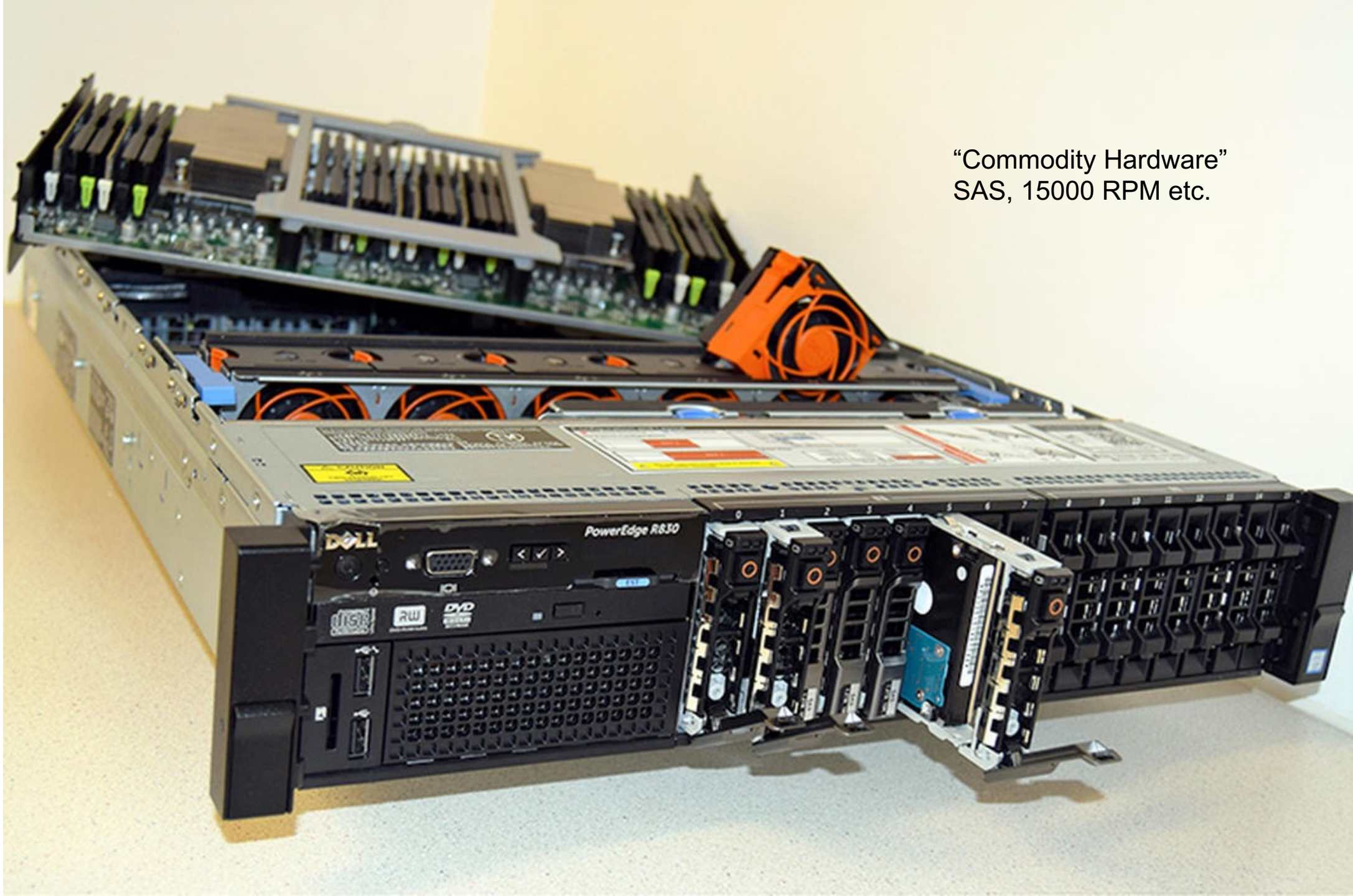
Google Server Room, one of many rows.

The Rack

→ Standard Rack is 42U (Rack Units)
→ 1U, which equates 44.45 mm, or one server on the picture
→ Servers may take up several U
→ This equates roughly 1.9m in height

SDU

Dell 2U Server



"Commodity Hardware"
SAS, 15000 RPM etc.

SDU

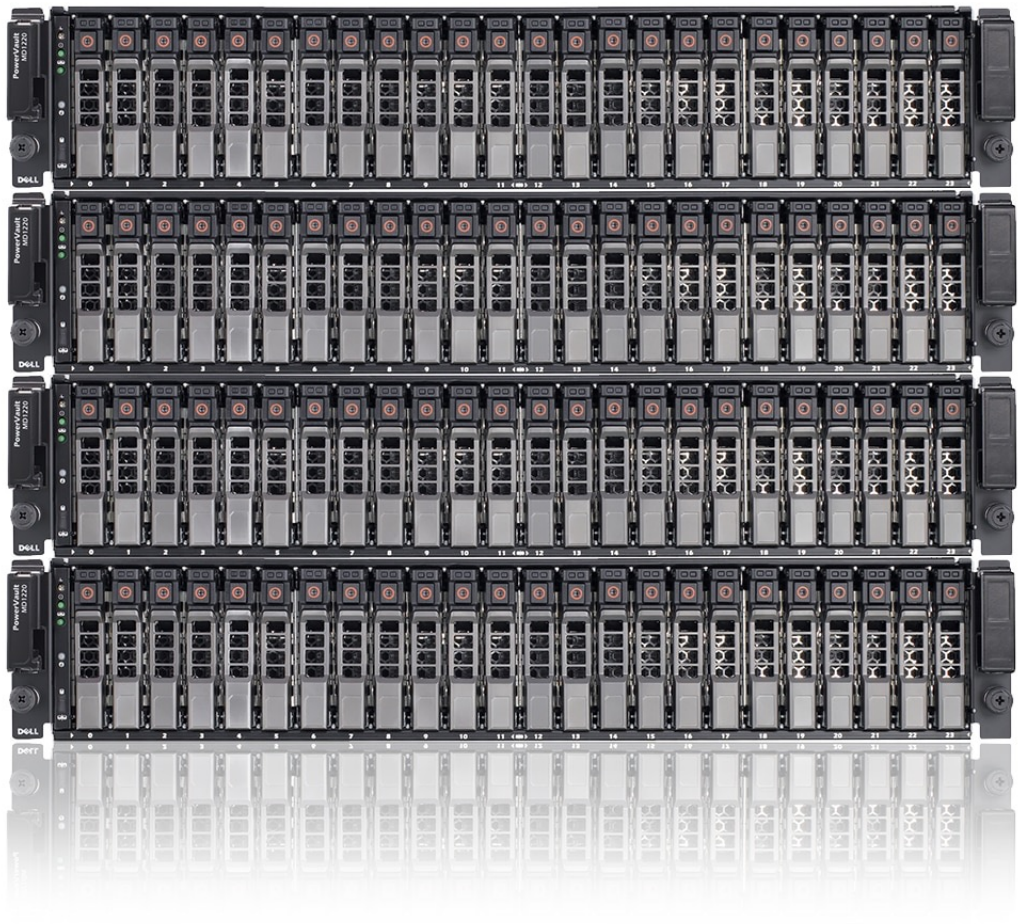
RAID Levels

Hewlett Packard Enterprise

BREAKDOWN OF COMMON RAID LEVELS

RAID LEVEL	METHOD	HARDWARE / SOFTWARE	MINIMUM # OF DISKS	COMMON USAGE	PROS	CONS
JBOD	SPANNING		2	INCREASE CAPACITY	COST-EFFECTIVE STORAGE	NO PERFORMANCE OR SECURITY BENEFITS
0	STRIPING		2	HEAVY READ OPERATIONS	HIGH PERFORMANCE (SPEED)	DATA IS LOST IF ONE DISK FAILS
1	MIRRORING		2	STANDARD APP SERVERS	FAULT TOLERANCE, HIGH READ PERFORMANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/2)
5	STRIPING & PARITY		3	NORMAL FILE STORAGE & APP SERVERS	SPEED + FAULT TOLERANCE	LAG FOR WRITE OPS, REDUCED STORAGE (BY 1/3)
6	STRIPING & DOUBLE PARITY		4	LARGE FILE STORAGE & APP SERVERS	EXTRA LEVEL OF REDUNDANCY, HIGH READ PERFORMANCE	LOW WRITE PERFORMANCE, REDUCED STORAGE (BY 2/5)
10 (1+0)	STRIPING & MIRRORING		4	HIGHLY UTILIZED DATABASE SERVERS	WRITE PERFORMANCE + STRONG FAULT TOLERANCE	REDUCED STORAGE (1/2), LIMITED SCALABILITY

SDU



Expanding with DAS

→ Adds storage to a single node server

→ Expensive

→ Does not add extra processing power

→ Controllers become bottlenecks

SDU

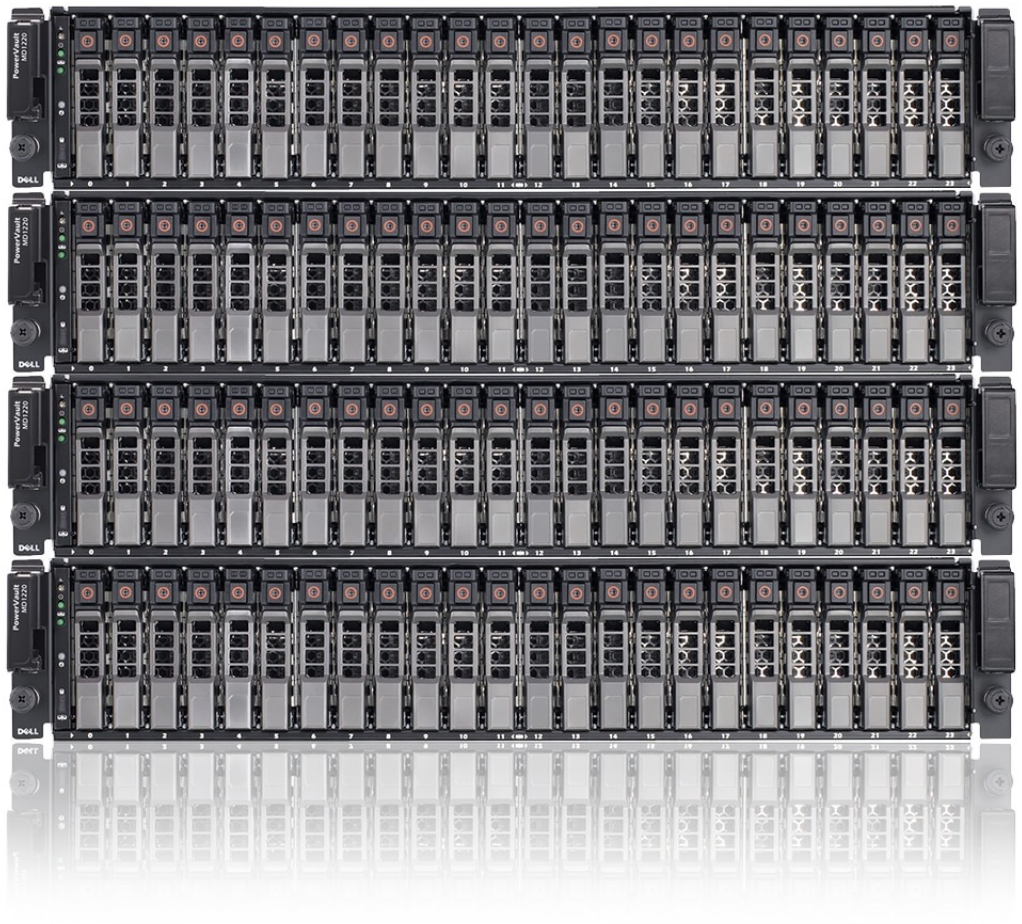
Considerations with traditional RAID

→ Disks can be protected at several levels

→ What if a node dies, not just a disk?

→ Replacing disks needs to be handled

→ Disks from same productions often fail together (Can be randomized)



Software Raid

- → A relatively new phenomenon
- → Some products:
  - → ZFS
  - → Microsoft Storage Spaces
- → Allows for versatility
  - → Does not require same disk sizes
  - → Does not require specific RAID controller
    - → Easy to move between setups
  - → Can often leverage RAM and SSD as read/write cache
- → Disadvantages
  - → Raid 5 calculated in CPU, not specialized hardware
  - → Raid 5 often not available

SAN

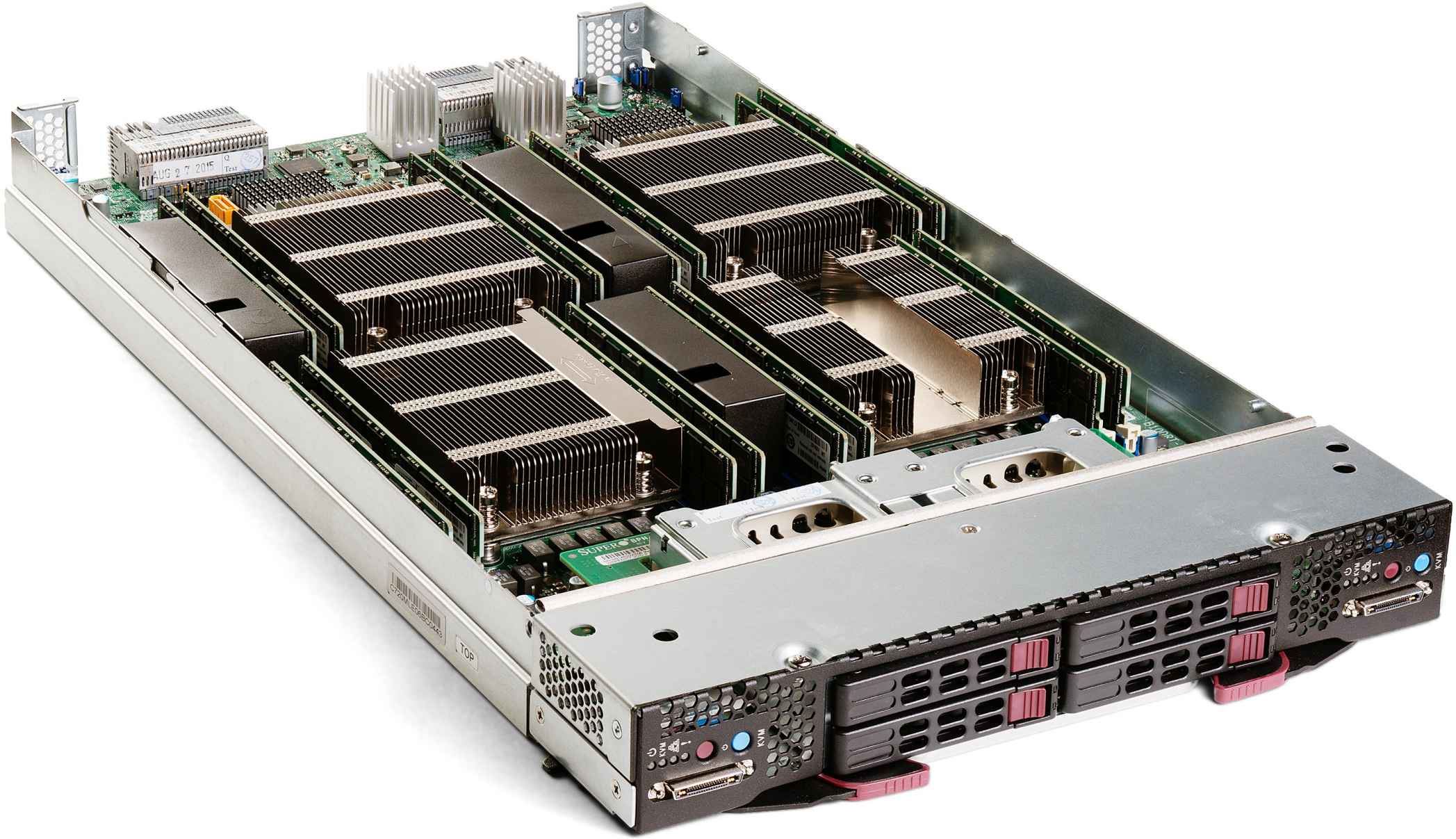
- → Virtualized storage
- → Managed
- → Critical operations
- → Extremely expensive (Management, Hardware, Maintenance)
- → High Performance (RAM and SSD read cache)
- → Limited by network speed
- → Allow for Bootable virtualized storage over a network.
  - → OS sees a directly attached disk, and the process is therefore transparent.
- → Easy to add disk space
- → Easy to resize disks of a servers SAN associated disk without reinstalling.
- → On some cases deduplication (especially with ZFS)
- → Storage easy to snapshot and backup.

SDU

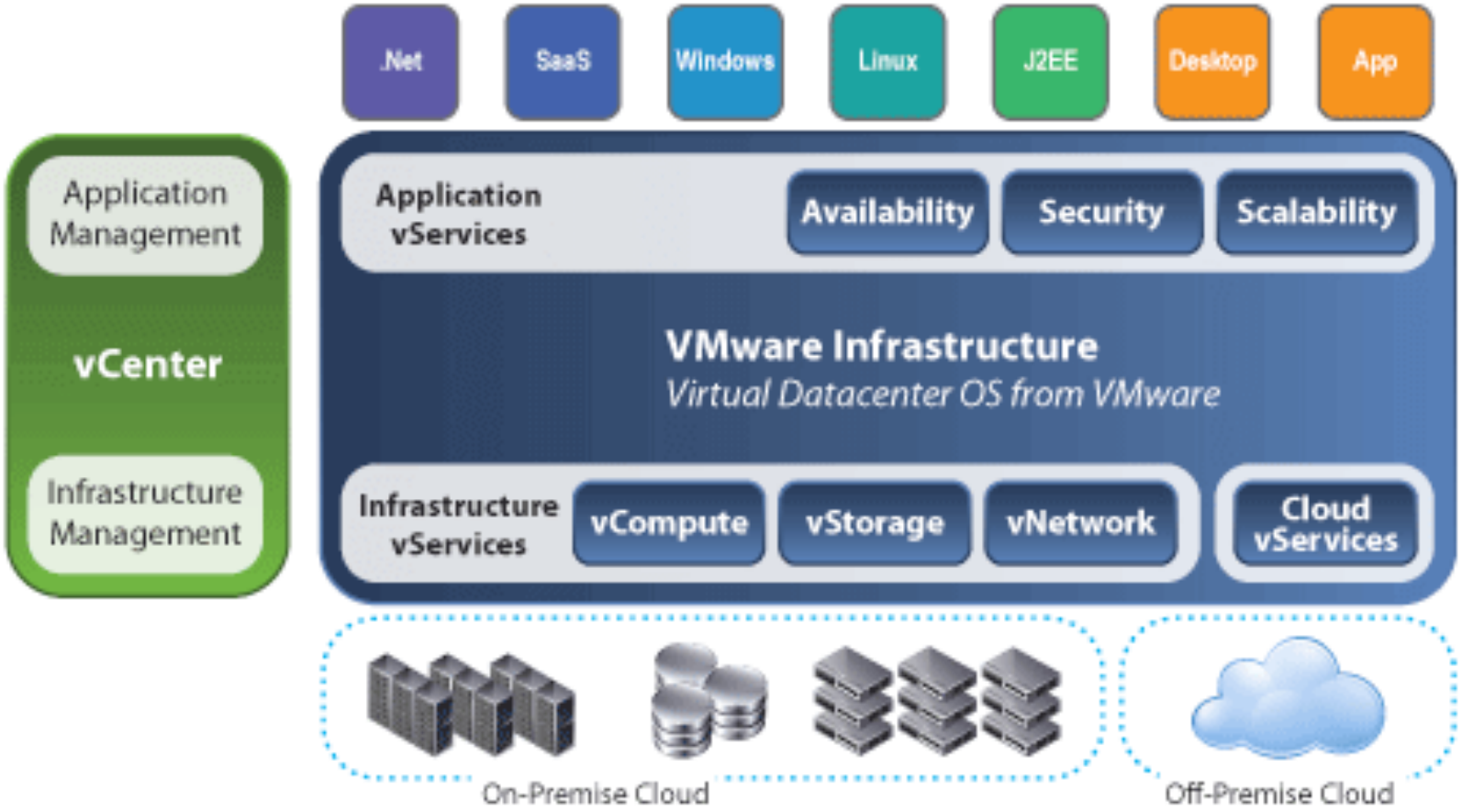
SAN helps separate processing and storage

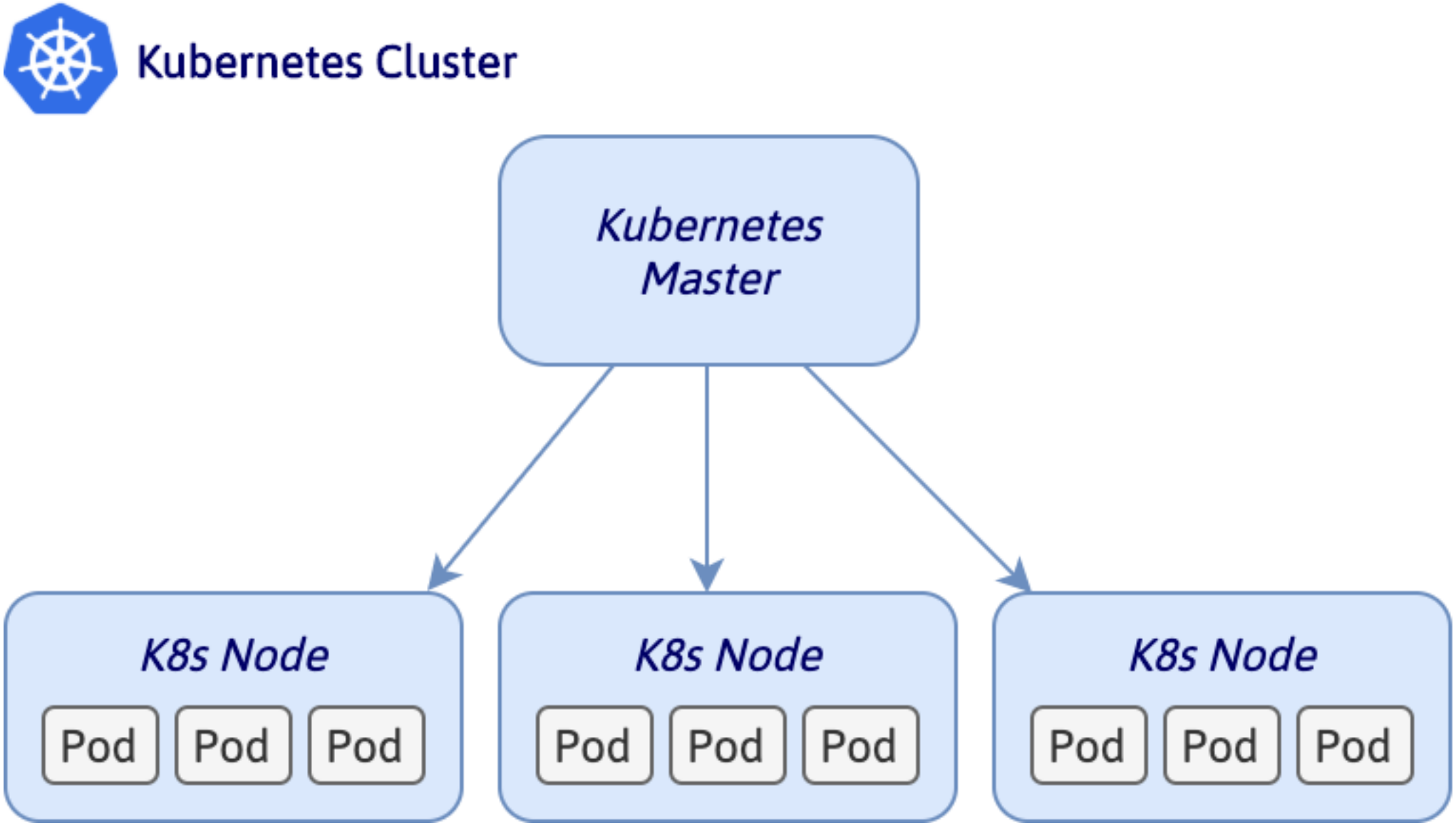
→ Blade chassis for high density computing tasks
→ Typically connected to a SAN, but this is not required
→ Exists in half the size visualized to the left

SDU

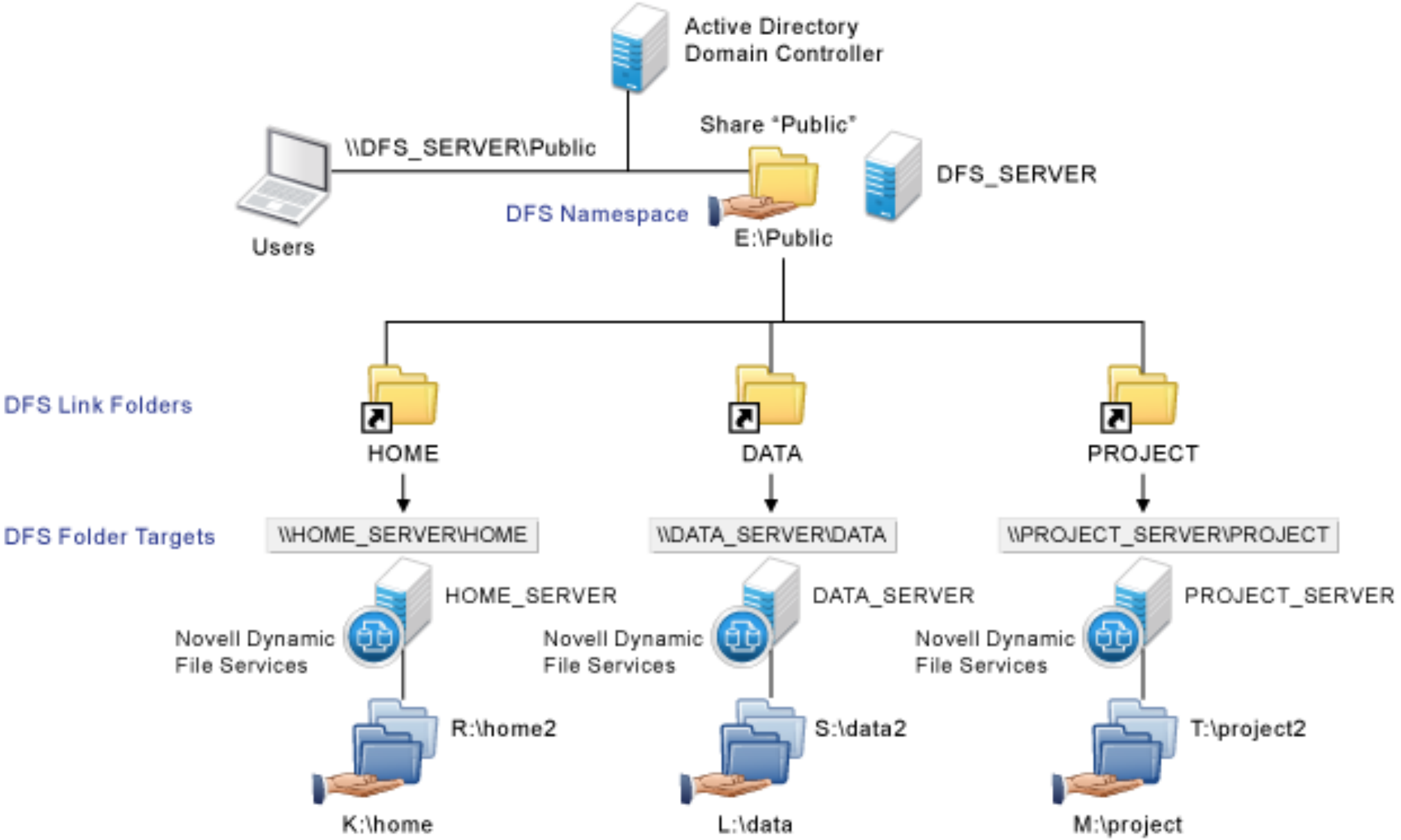


SDU


.Net
SaaS
Windows
Linux
J2EE
Desktop
App
Application Management
vCenter
Infrastructure Management
Application vServices
Availability
Security
Scalability
VMware Infrastructure
Virtual Datacenter OS from VMware
Infrastructure vServices
vCompute
vStorage
vNetwork
Cloud vServices
On-Premise Cloud
Off-Premise Cloud

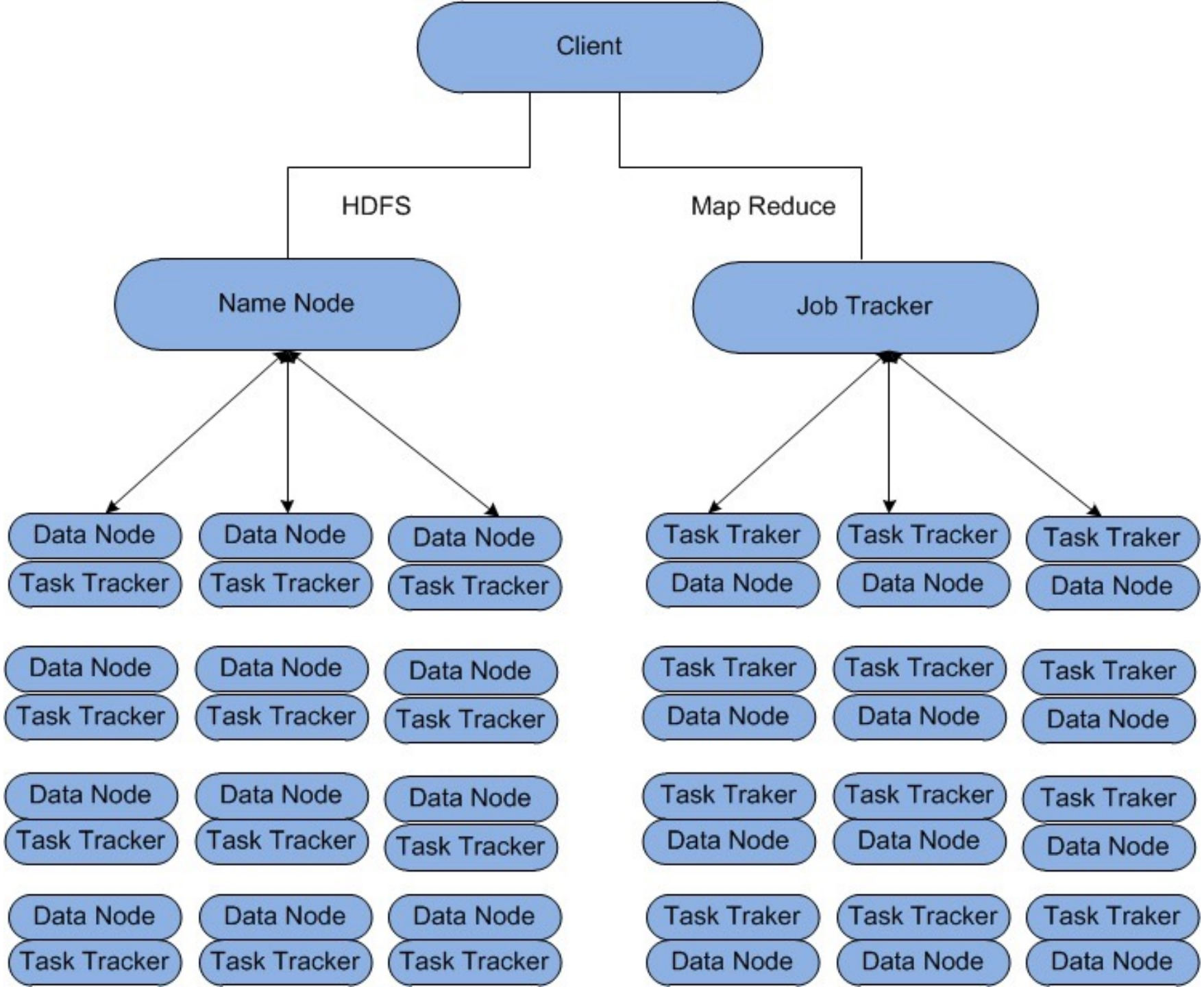

Kubernetes Cluster
Kubernetes Master
K8s Node
Pod
Pod
Pod
K8s Node
Pod
Pod
Pod
K8s Node
Pod
Pod
Pod

DFS Microsoft

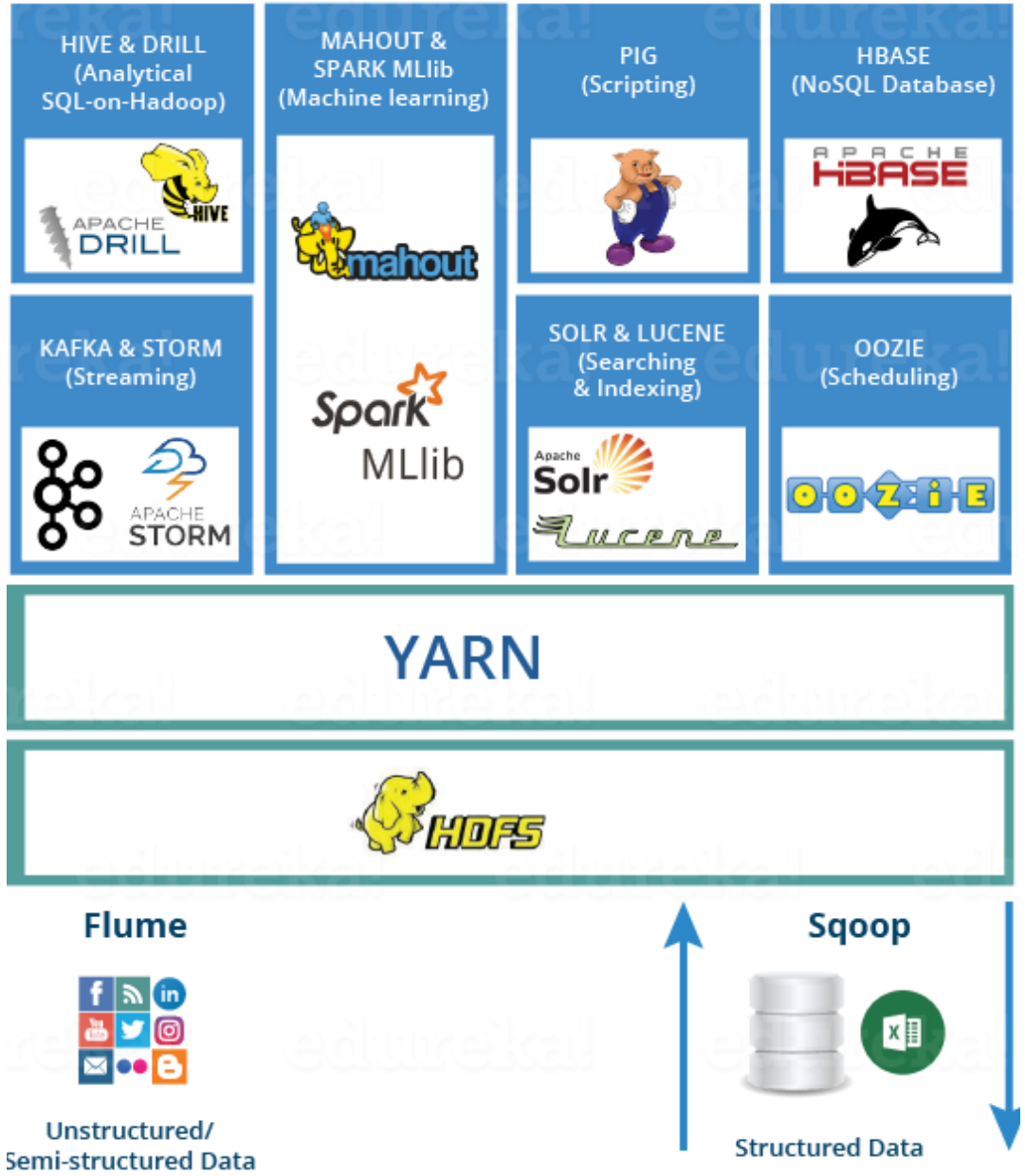


SDU

Hadoop and Map Reduse



SDU



DFS - Hadoop

- → Google published GFS and MapReduce papers in 2003-2004.
- → Yahoo! Was building “Nutch”, an open source web search engine at the same time.
- → Yahoo! Adopted the technologies pioneered by Google.
- → Hadoop was primarily driven by Dough Cutting and Tom White in 2006.
- → Its been evolving ever since…

Exercises!!!! – oh wait - Just get Docker Desktop to work (With K8S)