COPERNICUS MASTER IN DIGITAL EARTH

HPC FOR BIG DATA

3. The Hadoop Stack

Frédéric RAIMBAULT, Nicolas COURTY

University of South Brittany, France

IRISA laboratory, OBELIX team

Outlines

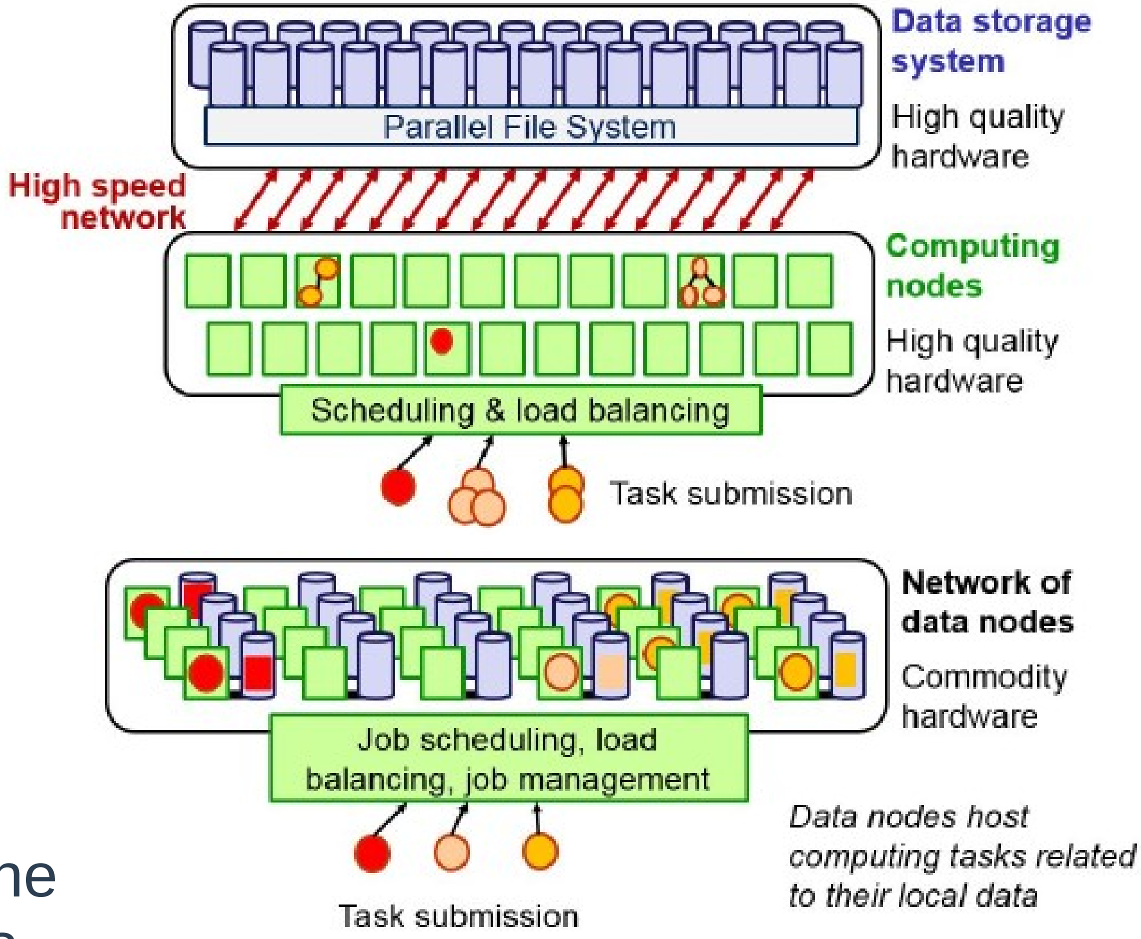
1. Hadoop Stack
2. HDFS
3. Amazon Services

1. The (Apache) Hadoop Stack

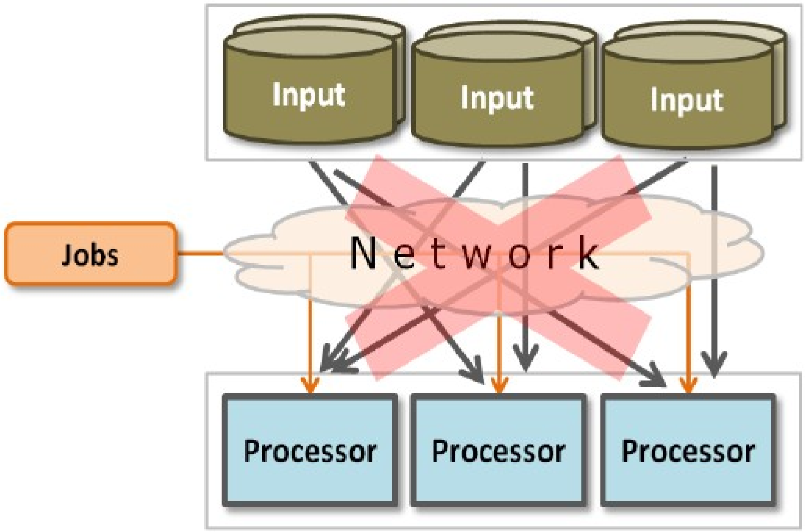
- One of the 350 open source projects of the Apache Software Foundation
- The Apache Hadoop project develops an open-source framework for reliable, scalable, distributed computing.
- Solves problems involving massive amounts of data and computation using a network of commodity servers.
- Its core is composed of a storage part (HDFS), and a processing part, a MapReduce programming tool.
  - Inspired by Google papers in 2003 on the Google File System and on the MapReduce programming model.

Google Big Data Approach

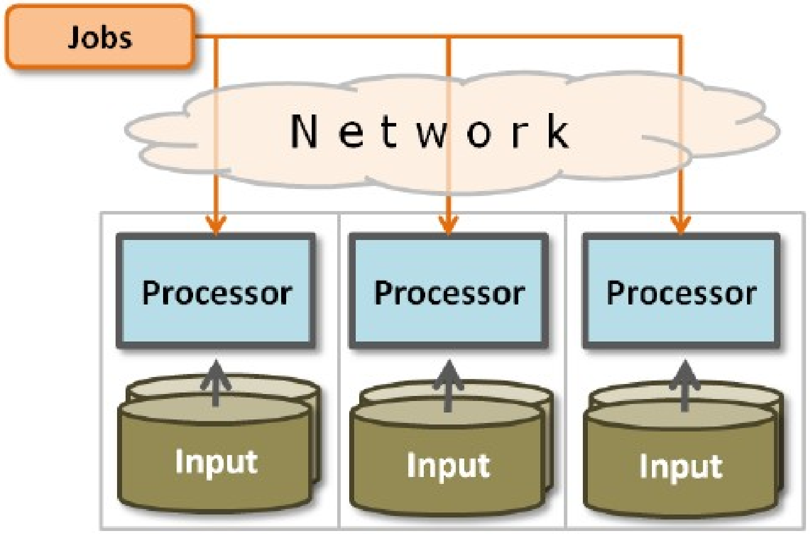
- **Conventional approach (DBMS and HPC)**
  - Data is located in a separate storage system
  - Computing nodes access remote data through a high speed network
- **Google Big Data approach (MapReduce)**
  - Data is partitioned across (data) nodes
  - Compute tasks are run to the data nodes owning the data



Remote vs Data Local Processing



Network I/O bottleneck: limits the scaling



Data local processing: distributes the program, not the data !

Data vs Compute Centric Paradigms

- **Co-located compute and storage resources**
  - Locality-oriented task scheduling
  - Minimized data movements
- **Takes advantage of commodity hardware, open software and the availability of bulk storage**

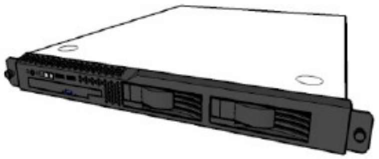
  => Reduced costs
  => Larger clusters
  => Easier scaling (horizontal scaling)
  => Increased throughput (vs responsiveness)

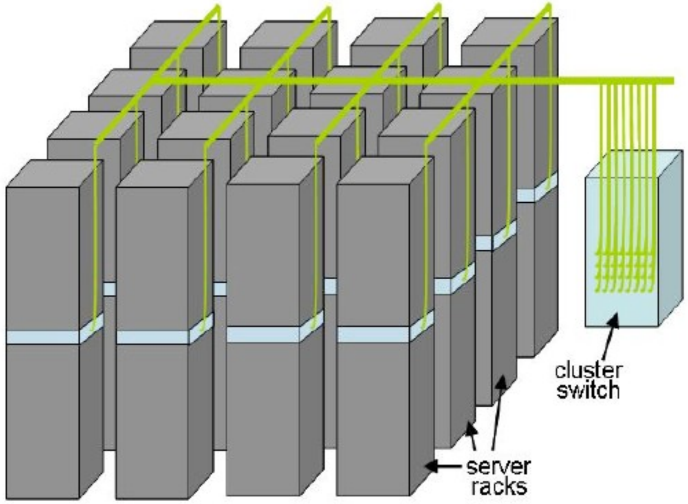
Some Facts about Google

- **Search engine among 4.47 billions URL (2017)**
  - Crawls, stores, ranks the Web pages and builds an inverted index
    - 4 MB per page → 20 PB → 10,000 HDD of 2TB
    - 40 MB/s HDD read throughput → 15h on 10,000 PC
- **Responds to 70,000 queries / s (2021)**

  → Geographically distributed data centers
- **Considering 2.5 millions servers (Gartner - 2016)**
    - MTBF of 1 server (1 HDD): 3 years
    - MTBF of 2.5M servers: 40s (2400 failures/day)

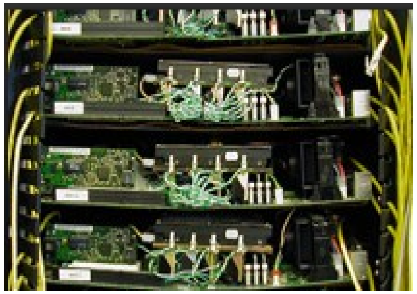
Google Hardware Infrastructure



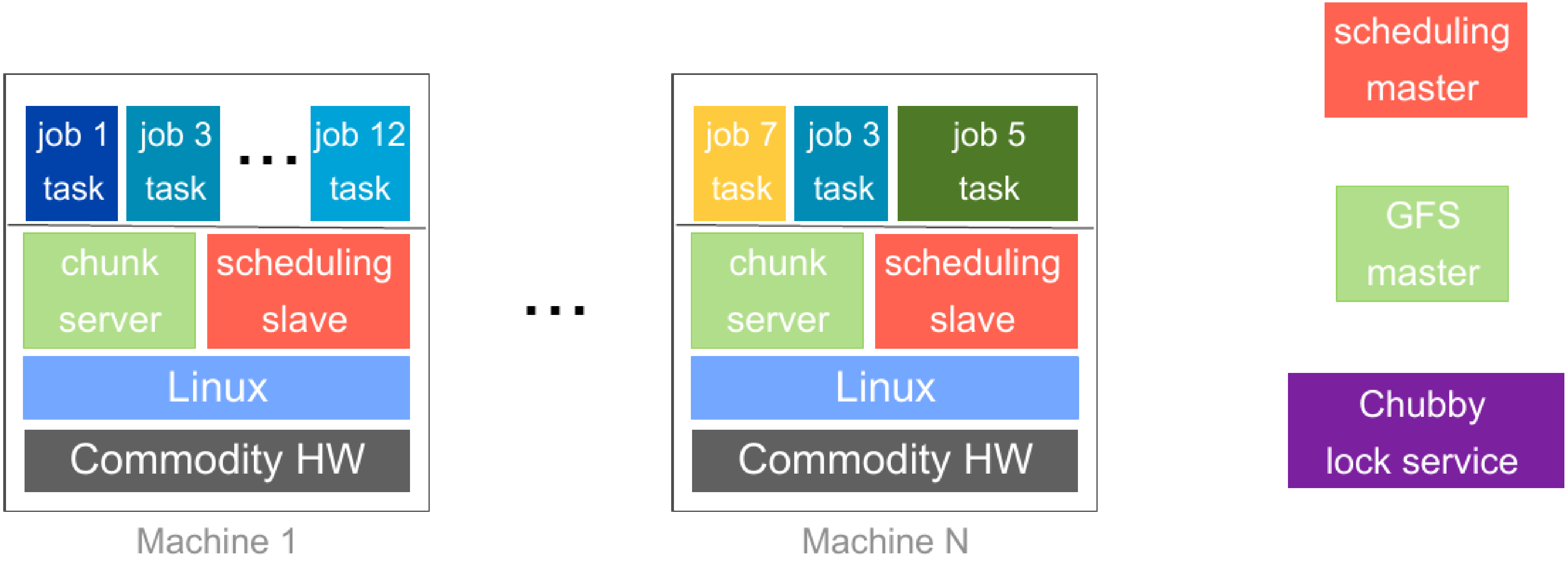


Servers
- CPUs
- DRAM
- Disks

Racks
- 40-80 servers
- Ethernet switch

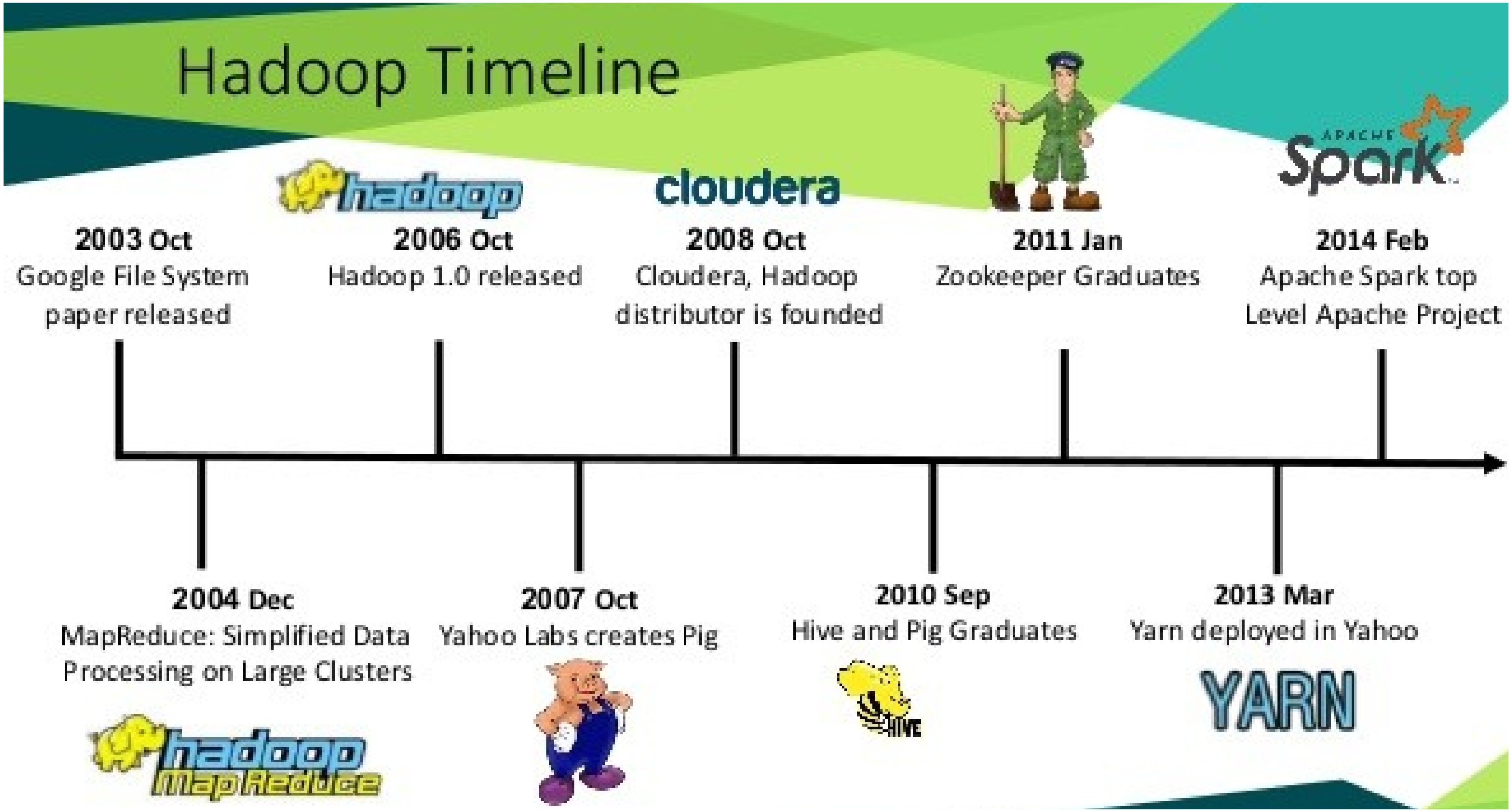


Google Software Infrastructure



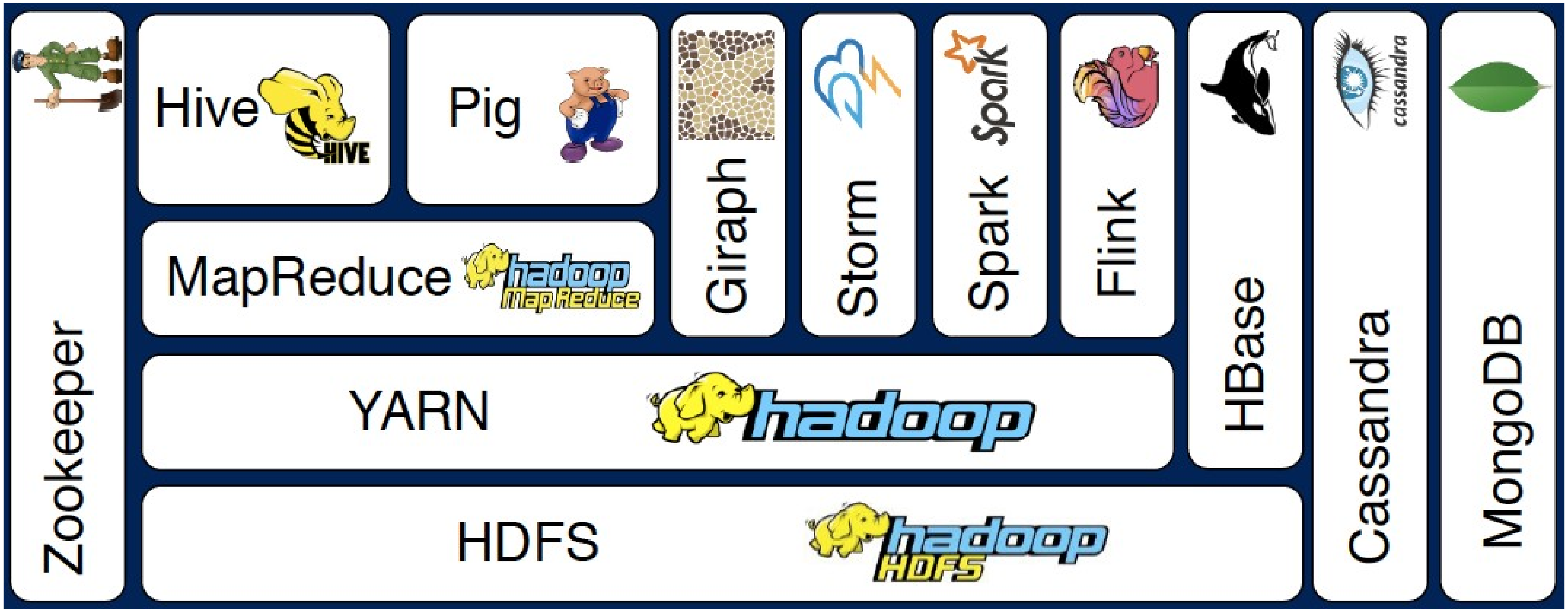
- **Stack of (own) distributed services over licence-free software**
  - Custom Redhat Linux distribution
  - Colossus (formerly GFS): distributed filesystem
  - Borg: cluster management system
  - Spanner (formerly BigTable) : distributed database
  - Chubby: synchronization and lock service
  - Cloud DataFlow (formerly MapReduce:) parallel programming system

Apache Hadoop



- **Hadoop website:** `http://hadoop.apache.org/`

Hadoop Software Stack



- **A galaxy of tools and frameworks**
- **Written entirely in Java**
  - But existing API for many languages

Powered by Hadoop

- **Hadoop is adapted, used, developed by most web giants: Yahoo!, Amazon, Facebook, Twitter, Spotify, eBay, Alibaba, BablaCar, LinkedIn, Last.fm,…**
- **Mimic at a smaller scale in industry: web-scale networking initiative**
  - Companies that built private, efficient and scalable cloud environments (based on Hadoop-like tools)

Outlines

1. Hadoop Stack
2. HDFS
3. Amazon Services

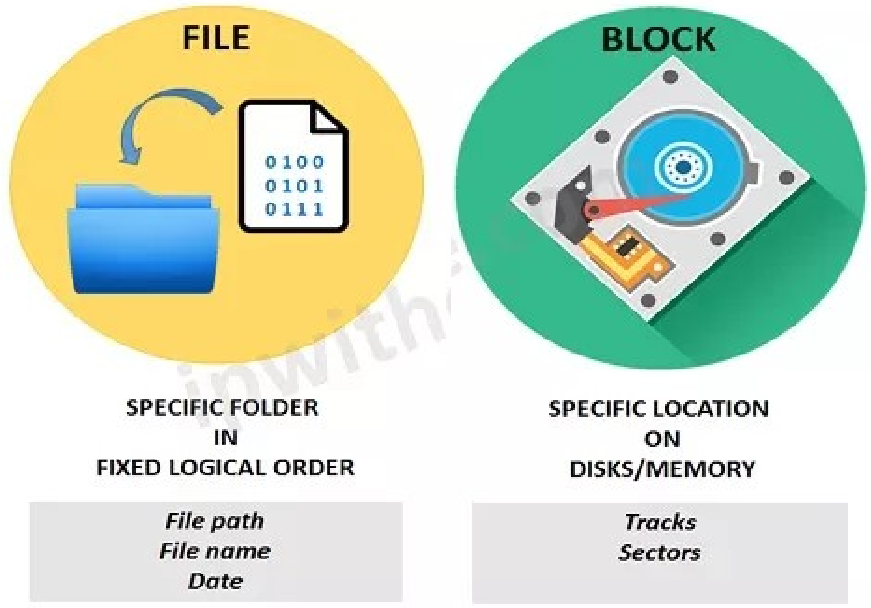
2. Hadoop Distributed File System

- **Assumptions**
  - Massive amount of data
  - Many concurrent access
    - Write Once Read Many
  - Commodity hardware
    - Failures on frequent basis
  - Mostly off-line batches
    - Streaming data access
    - High throughput vs. low latency

- **Features**
  - Scalability
  - Data Coherence
  - Reliability and fault tolerance
  - Hardware failure recovery
  - Portability
  - Move Computation to data

Key Concept: Filesystem

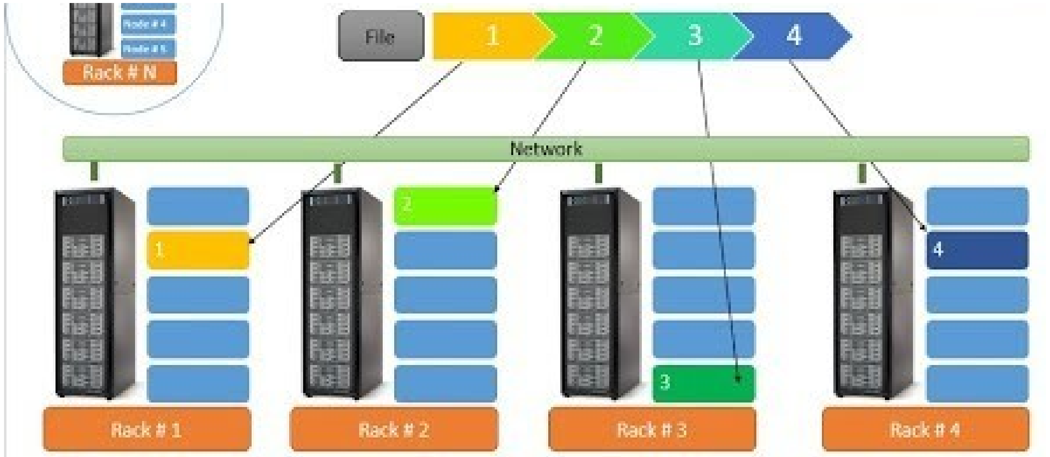
- **A software service, part of the Operating System**
- **Manages files stored on media HDD, SSD, CD, USB keys,…**
  - Storage
    - Allocation into devices
      - RAM, HDD, SSD, magnetic tapes, optical discs
      - Divided into chuncks (e.g. sector disk) that are linked
  - Retrieval (and updating)
    - Hierarchical (vs. flat) filesystem: directory
    - Metadata: filename, pathname, chunck list, ACL (owner, group)



- **User interface**
  - CLI (e.g. Unix shell commands `ls, cd, cat, mkdir, rm`, ...) or graphical file browsers (e.g. Dolphin, Explorer,...)
- **API for various programming languages**

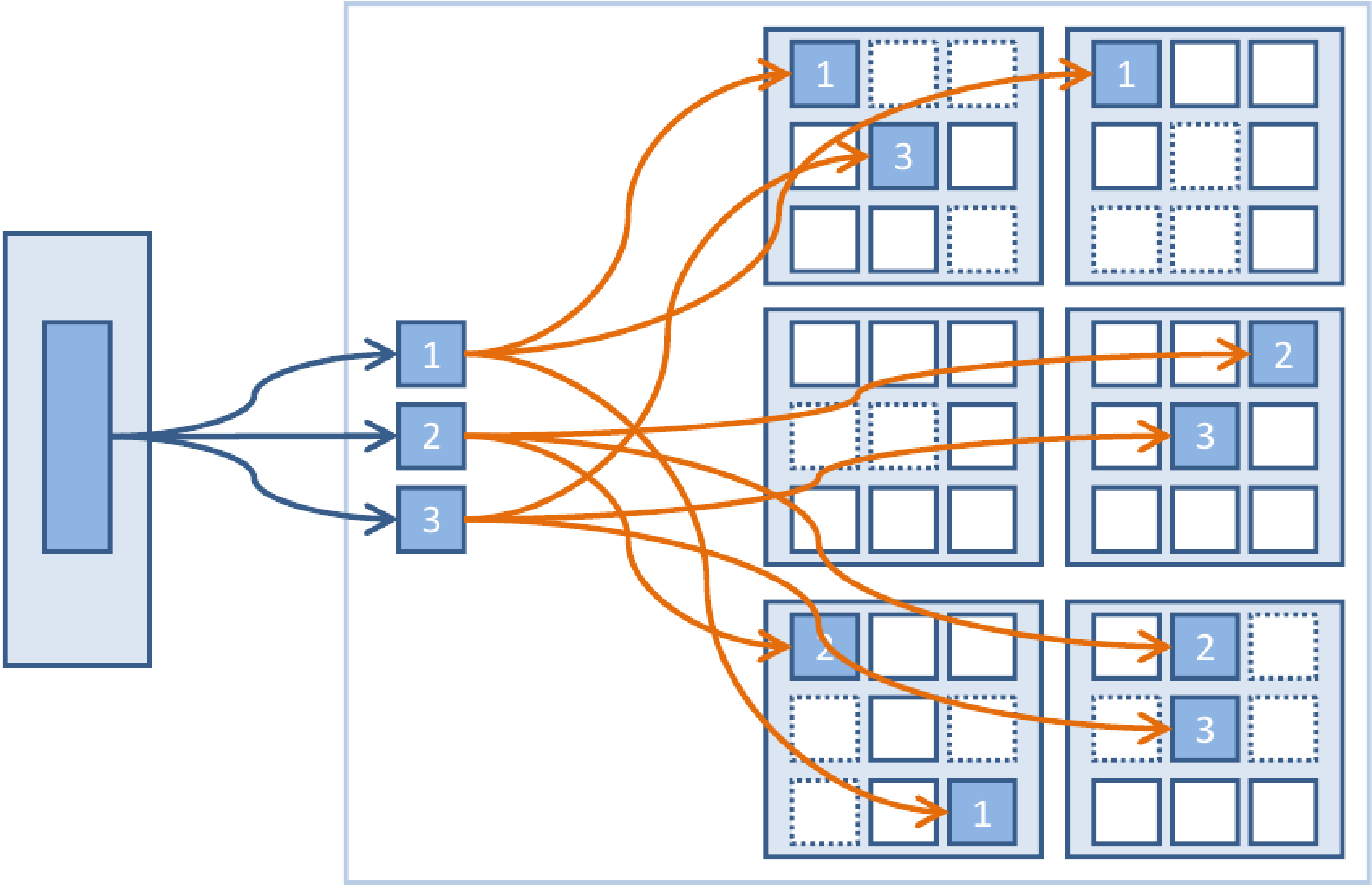
Key Concept: Distributed File-system

- **When a dataset outgrows the storage capacity of a single physical machine, we have to partition it across a number of separate machines.**
- **A <u>cluster</u> distributed file-system manages the storage across a network of machines**
- **Design goals (achieved or not):**
  - Access transparency
  - Location transparency
  - Concurrency transparency
  - Failure transparency
  - Replication and migration transparency
  - Scalability

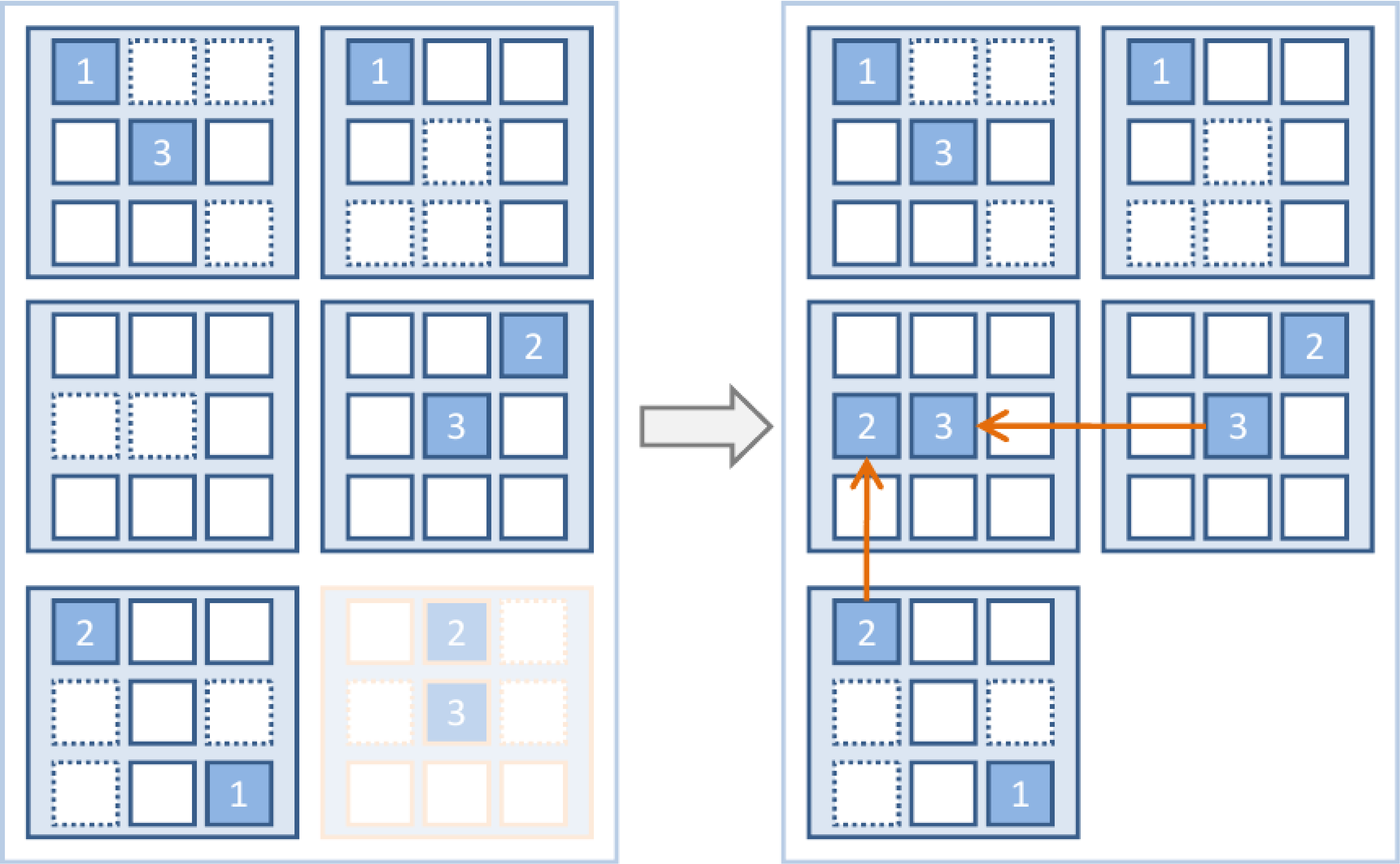


Cluster Distributed File Blocks

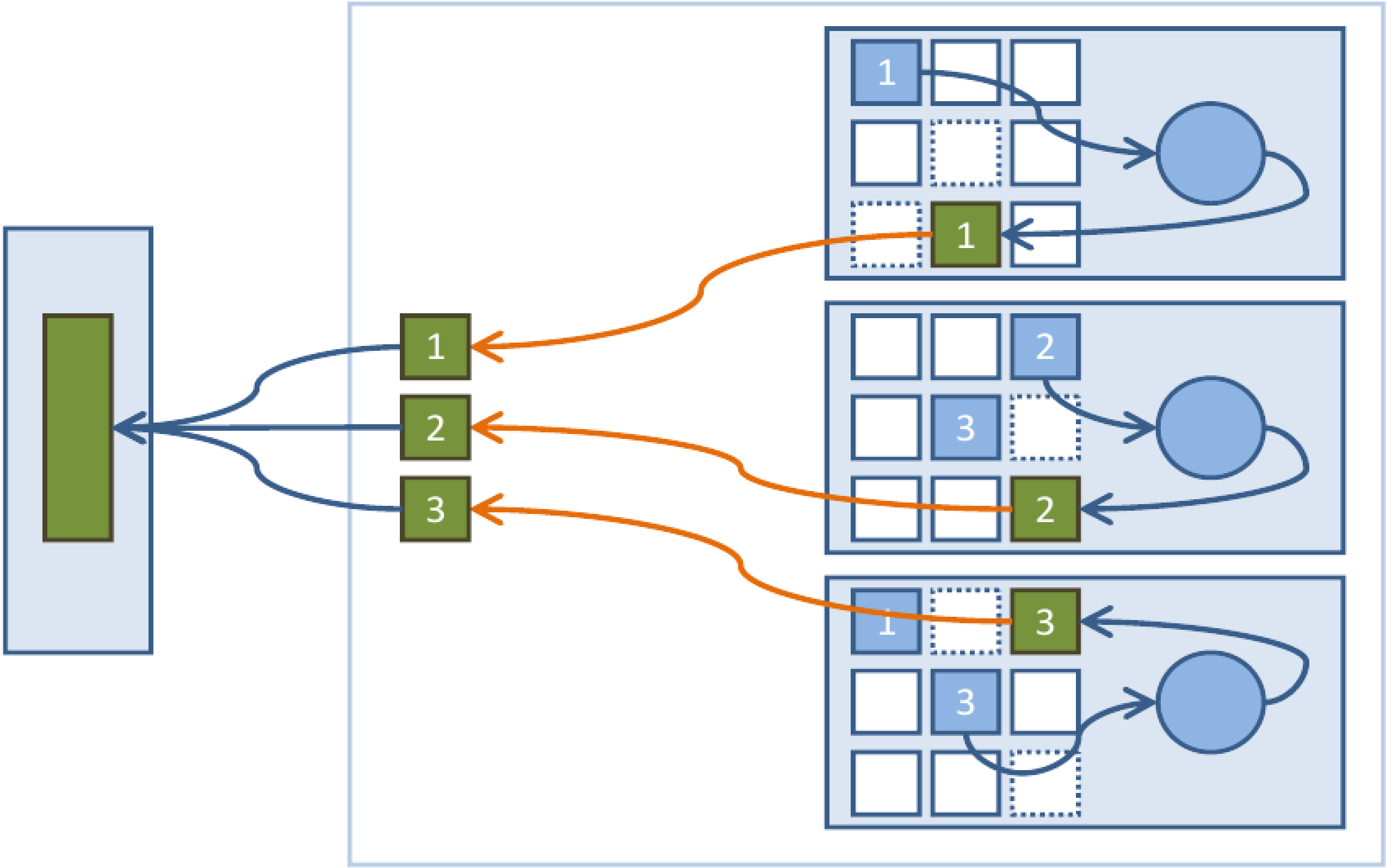
- **Split**
- **Distribute**
- **Replicate**



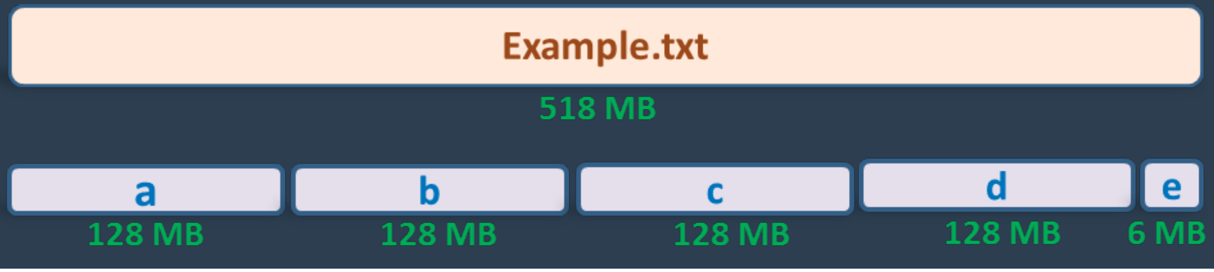
Replication for Automatic Repairing



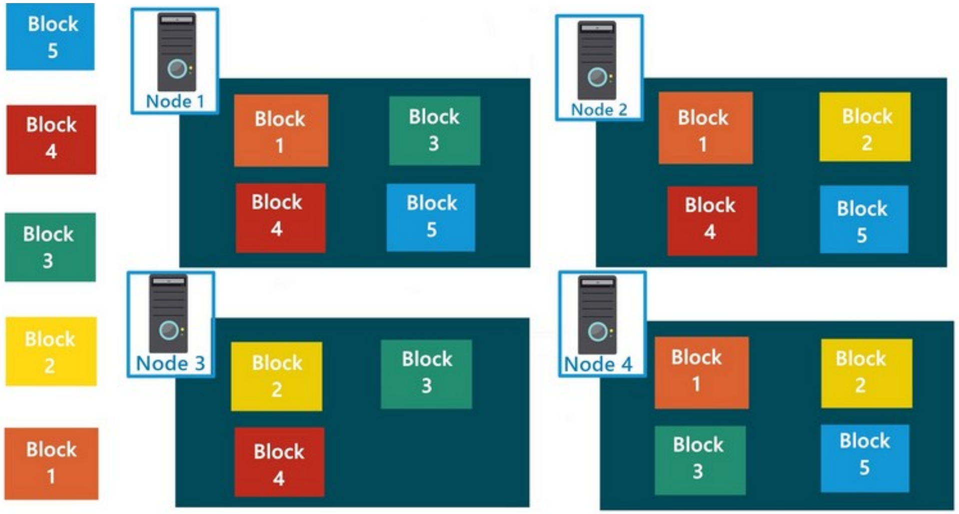
Replication for Data-Local Processing

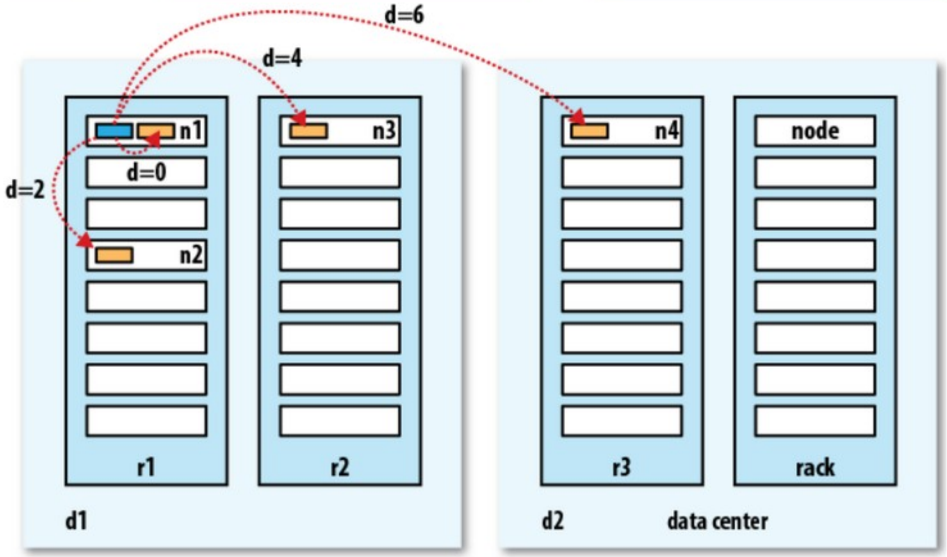


HDFS Blocks



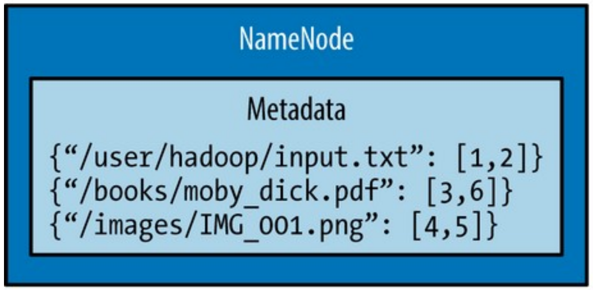
- **File are split into fixed size blocks**
  - Block size of 64 MB
    or 128 MB
    (last chunk may be smaller)
- **Data is distributed**
  and replicated on data nodes
  - Typical block replication of 4
  - Distance is taken into account for performance when choosing computation location



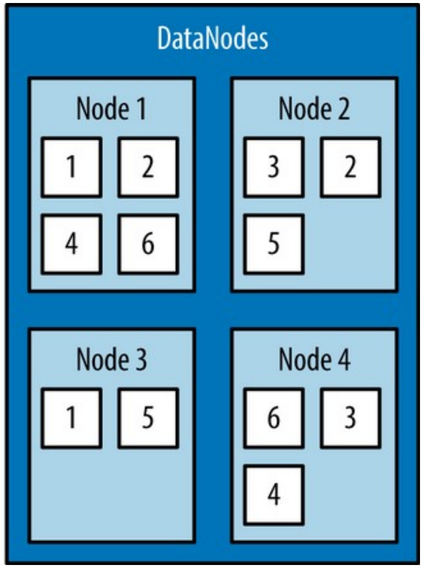


HDFS Metadata

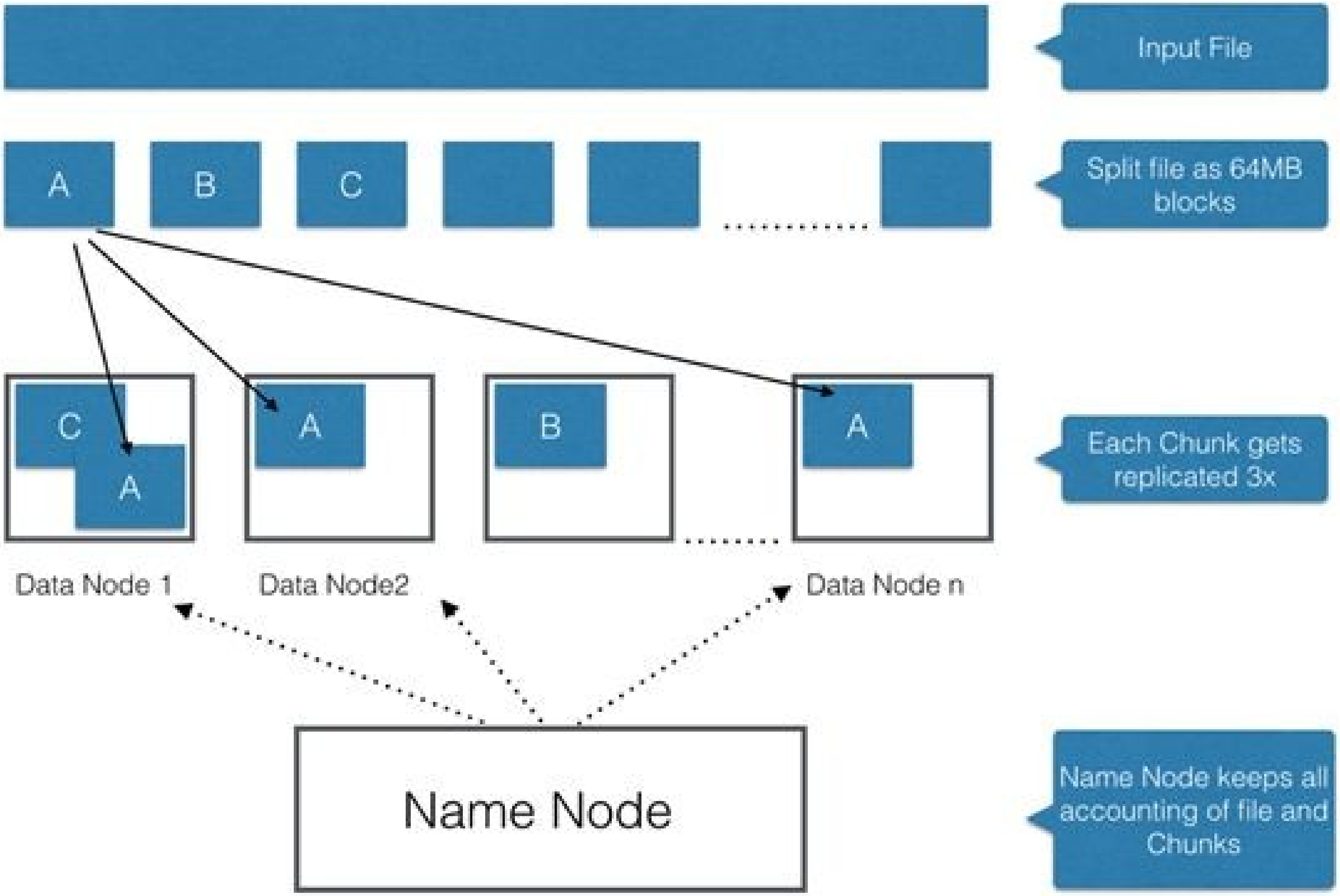
- **Hierarchical file system**
- **Pathname, filename, owner, groups, permissions**
  - Separate name space from the local FS
    - can’t use local file system commands
- **Blocks map**
  - Locations of blocks and locations of replicated blocks
- **Metadata are managed by the NameNode (one per cluster)**



```
{“/user/hadoop/input.txt”: [1,2]}
{“/books/moby_dick.pdf”: [3,6]}
{“/images/IMG_001.png”: [4,5]}
```



Metadata Centralization and Data Distribution


Input File
A
B
C
Split file as 64MB blocks
C
A
A
B
A
Each Chunk gets replicated 3x
Data Node 1
Data Node2
Data Node n
Name Node
Name Node keeps all accounting of file and Chunks

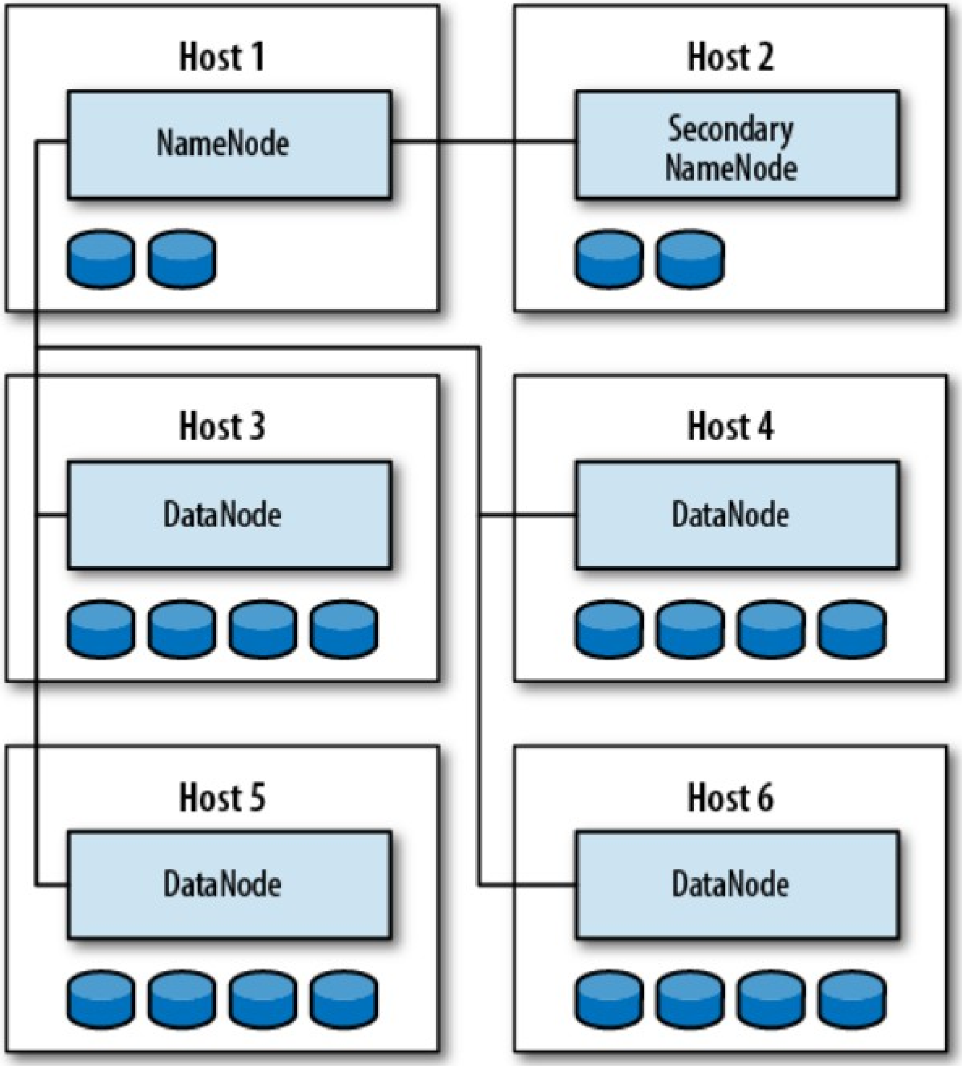
Data is divided into HDFS blocks

HDFS blocks are distributed and replicated into the DataNodes

Metadata are centralized into the NameNode

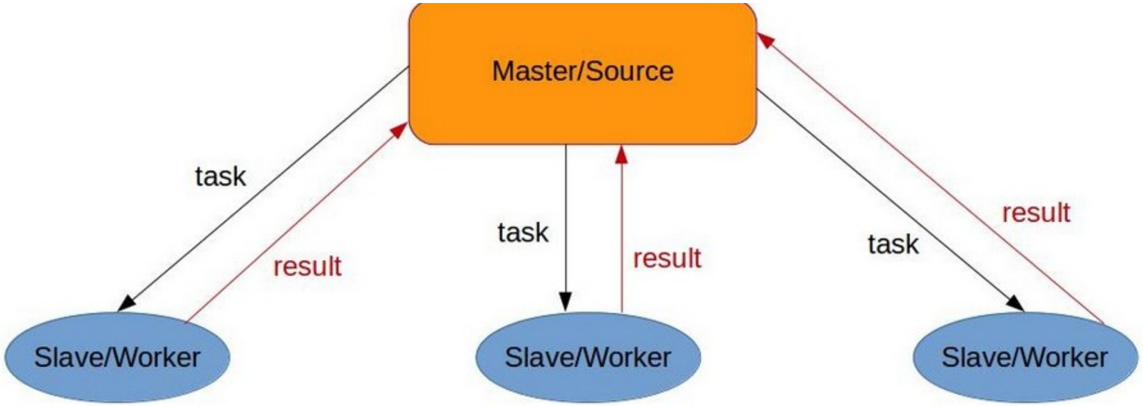
HDFS Architecture

- **Master-slave model**
  - NameNode: master
    - 1 per cluster
    - stores metadata and file-to-block map into memory
    - manages block replication
    - should run on a dedicated server
  - (Secondary NameNode)
    - 1 per cluster
    - generates snapshots of the primary NameNode's memory
  - DataNodes: slaves
    - 1 per node (server)
    - Store block data



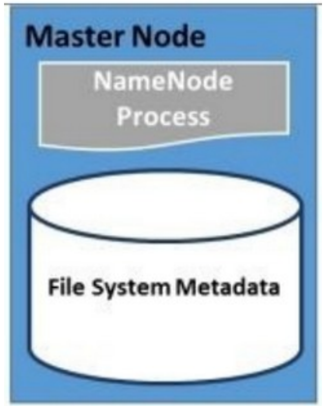
Key Concept: Master-Slave Model

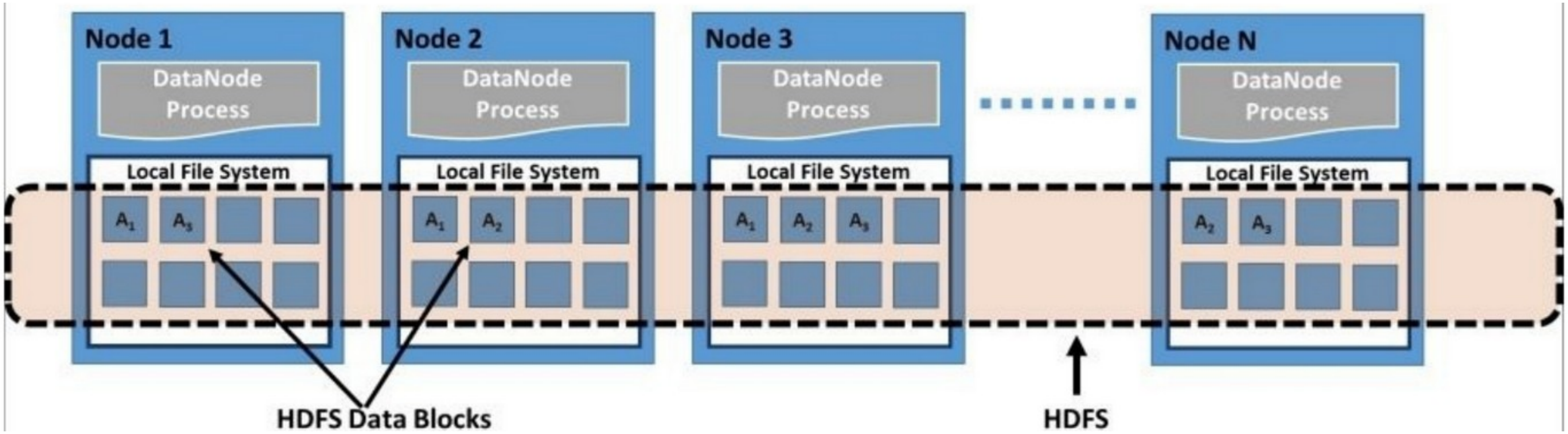
- **A parallel programming concept**
- **Divide the processing into smaller independent tasks**
  - One master distributes (scatters) the tasks (or the data)
    - And gather the partial results to produce the final result
  - The slaves nodes do the processing
    - Claim jobs until all is done
    - No communication between slaves
    - Return result to the master



HDFS Over Local FS

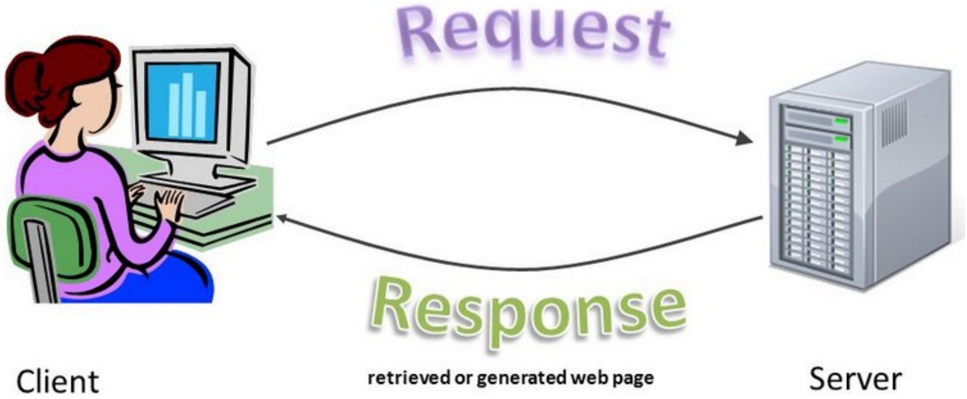
- **A (User space) distributed file system**
  - Not inside the OS: uses the FS of nodes
  - A network file system service
    - A **client-server** service
    - Implemented using the master-slave model

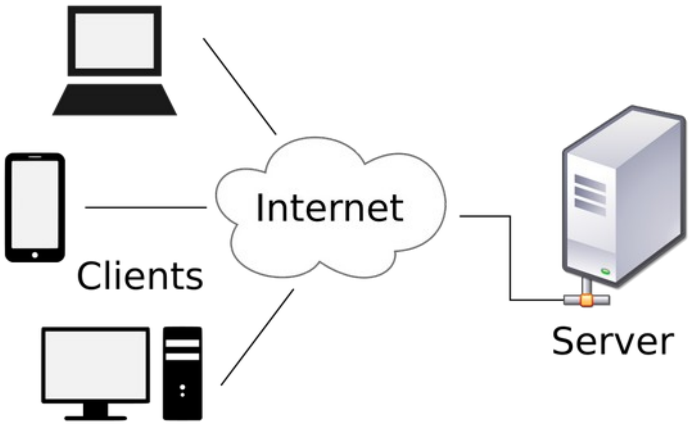




Key Concept: Client-Server Model (1)

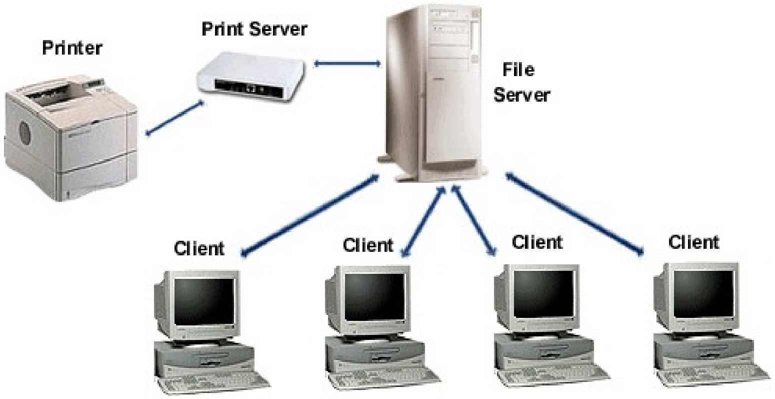
- **An application structure that partitions the roles between**
  - The server(s)
    - The provider of a resource or service
      - Name also given to the host computer
    - A process running continuously a program
      - e.g. Unix daemons
    - Accepts requests and replies
  - The clients
    - The service requesters
    - Invoke operations upon the server

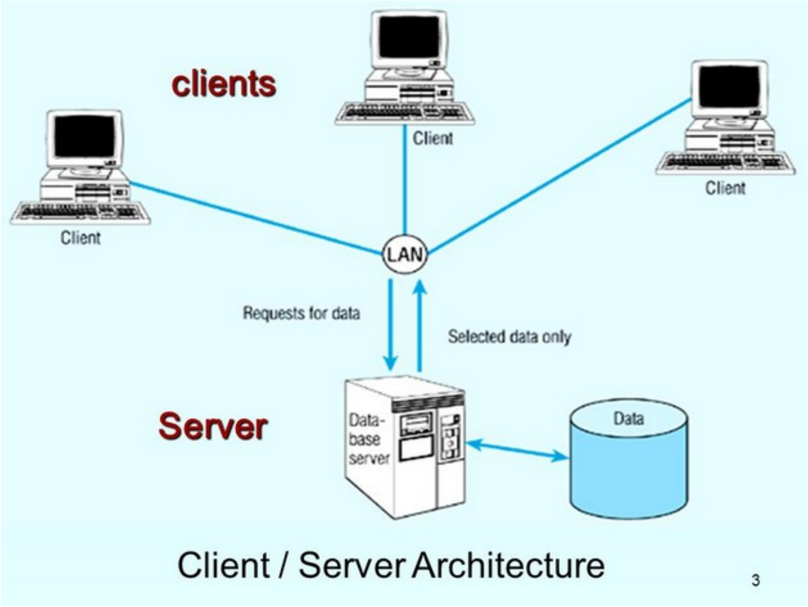


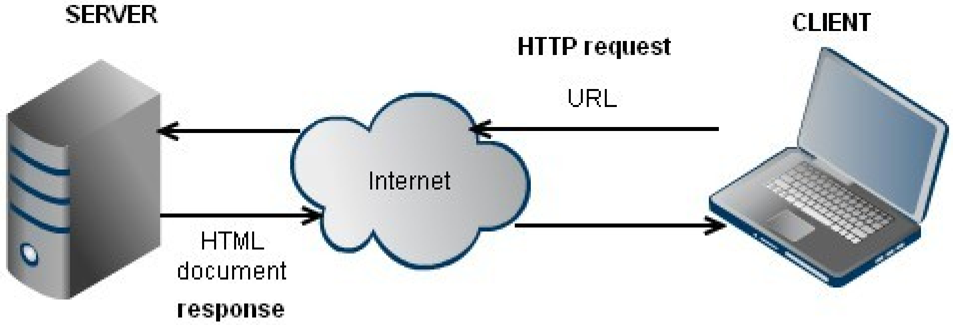


Key Concept: Client-Server Model (2)

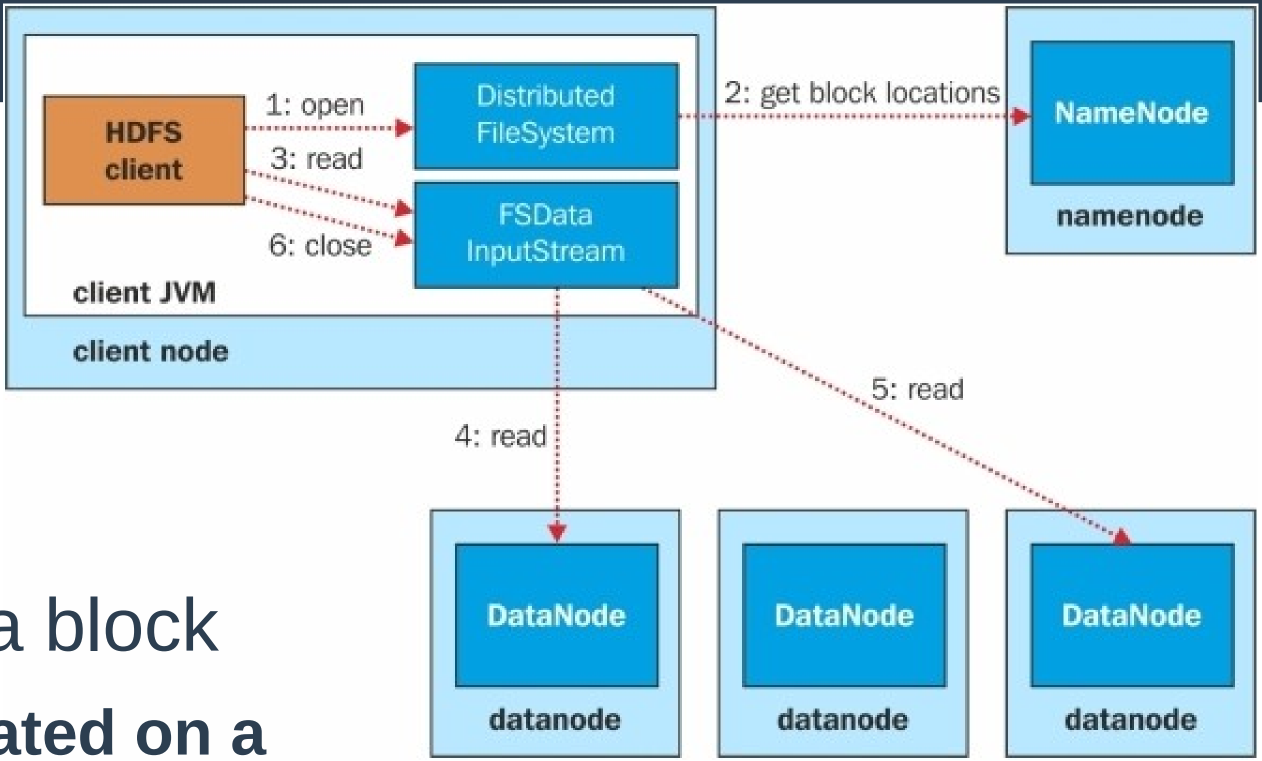
- **Clients and servers may reside on the same system**
  - e.g. file-system, printing, GUI
- **or on separate hardware**
  - communicate over a computer network
    - e.g. network file-system, network printing, email, DBMS, WWW





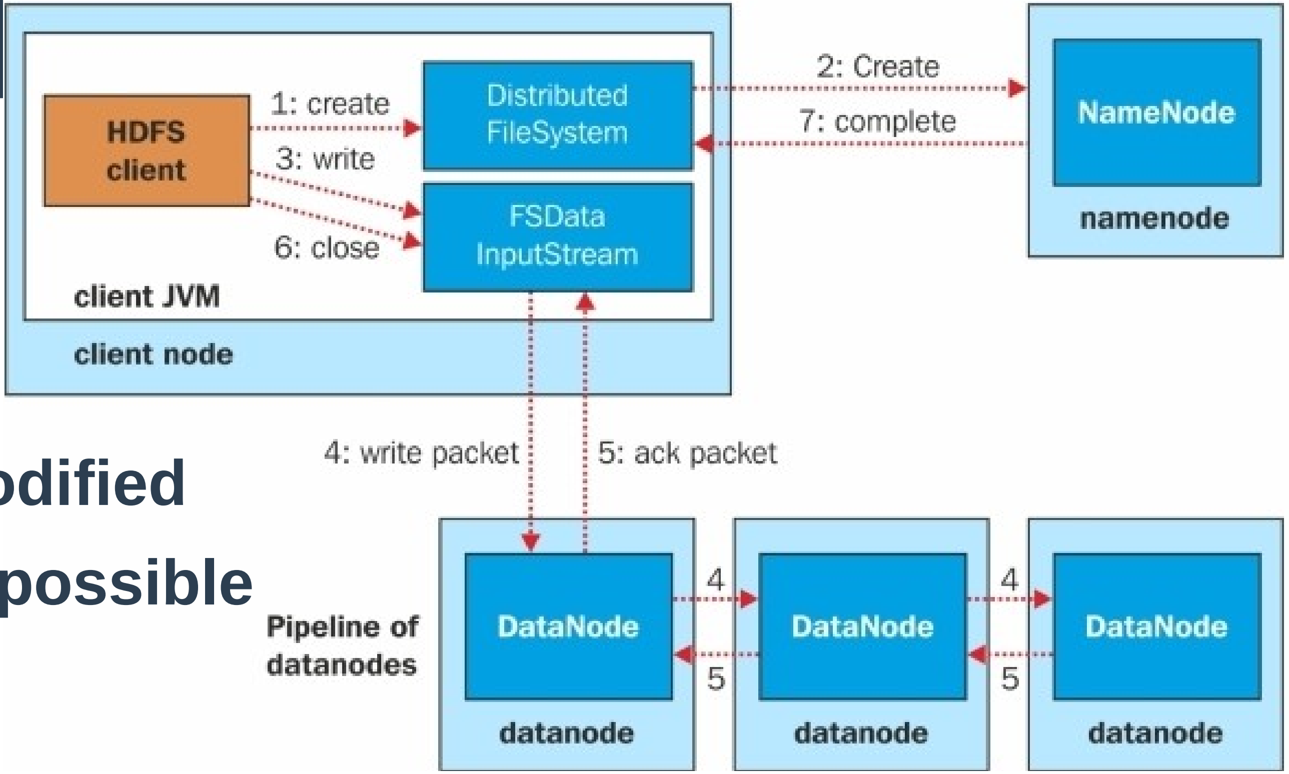


HDFS Read Process



- **Files are read by block**
  - The NameNode locates the nearest DataNode owning a block
- **Usually the client is located on a node machine (running a DataNode too)**
  - When distributing tasks to the nodes, the location of the HDFS client should be chosen according to the location of the block to be read to avoid moving data.
- **The client is not aware of this process!**

HDFS Write Process



- File creation
- File data cannot be modified
- But append access is possible
- Written locally on a datanode by block
- Blocks are replicated on others DataNodes on the fly
- On file closing, the NameNode commits it and the file becomes visible by others clients.

HDFS Clients

- **Many user and program interfaces**
  - HTTP server for browsing & supervising
  - Command Line Interface (shell commands)
  - Python API *Snakebite* or (*Boto* on AWS)
- **HDFS on “cluster-irisa”**
  - http://cluster-irisa.univ-ubs.fr/hadoop/hdfs/
  - Namenode: `hnn`
  - DataNodes: `bugs1`, `bugs2`, ... `bugs18`
- **HDFS on AWS**

`hdfs` CLI

- Usage: `$ hdfs COMMAND [-option <arg>]`
- Common file operations : `dfs` command
  - List directory contents: `-ls <rep>`
  - Creating a directory: `-mkdir <rep>`
  - Copy onto HDFS: `-put <local rep> <hdfs rep>`
  - Copy from HDFS: `-get <hdfs rep> <local rep>`
  - Read from HDFS: `-cat <hdfs rep>`
  - Remove from HDFS: `-rm <hdfs file>`
  - Move inside HDFS: `-mv <hdfs rep> <hdfs rep>`
- Administration operations
  - Check FS status: fsck (useful to get locations of blocks)
    - e.g.: `hdfs fsck /data/ais -files -blocks -locations`

Snakebite

- Python package created by Spotify
- Provides access to HDFS programmatically in Python

```
from snakebite.client import Client

client = Client("hnn",9000)

for y in client.ls(["/data/ais"]):

   print(y)

   p=y.get("path")

   …
```

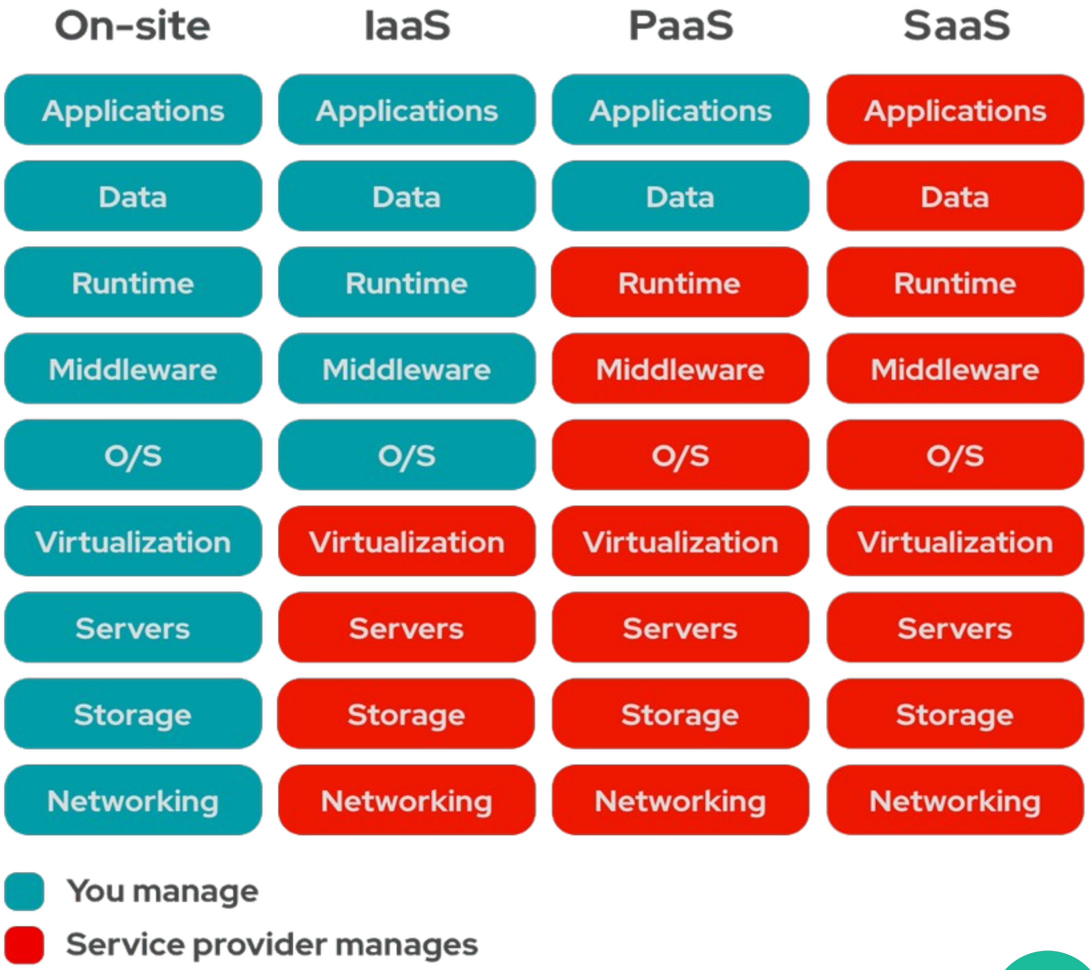
```
{'file_type': 'd', 'permission': 493, 'path': '/data/ais/2015', 'length': 0,
'owner': 'raimbaul', 'group': 'hadoop', 'block_replication': 0,
'modification_time': 1451726886727, 'access_time': 0, 'blocksize': 0}

{'file_type': 'd', 'permission': 493, 'path': '/data/ais/2016', 'length': 0,
'owner': 'raimbaul', 'group': 'hadoop', 'block_replication': 0,
'modification_time': 1546254021144, 'access_time': 0, 'blocksize': 0}
```

Outlines

1. Hadoop Stack
2. HDFS
3. Amazon Services

Clouds Service Models

- **Cloud computing: IT resources (computing,storage,network) provided as a service**
- **Services Models:**
  - Infrastructure as a Service (IaaS)
    - e.g. Amazon EC2
  - Platform as a Service (PaaS)
    - e.g. Amazon EMR
  - Software as a Service (SaaS)
    - e.g. Amazon S3

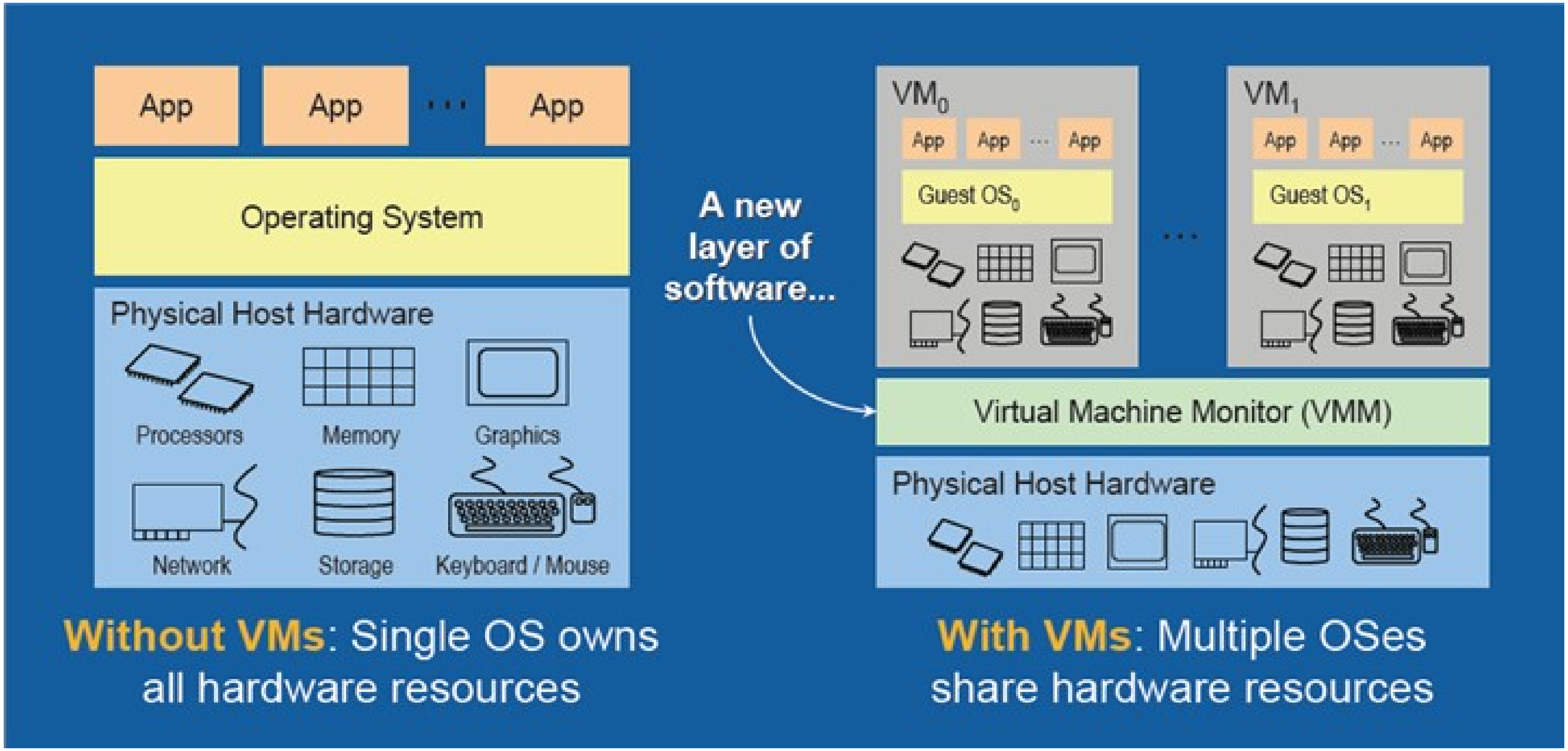


On-site	IaaS	PaaS	SaaS
🟦 Applications	🟦 Applications	🟦 Applications	🟥 Applications
🟦 Data	🟦 Data	🟦 Data	🟥 Data
🟦 Runtime	🟦 Runtime	🟥 Runtime	🟥 Runtime
🟦 Middleware	🟦 Middleware	🟥 Middleware	🟥 Middleware
🟦 O/S	🟦 O/S	🟥 O/S	🟥 O/S
🟦 Virtualization	🟥 Virtualization	🟥 Virtualization	🟥 Virtualization
🟦 Servers	🟥 Servers	🟥 Servers	🟥 Servers
🟦 Storage	🟥 Storage	🟥 Storage	🟥 Storage
🟦 Networking	🟥 Networking	🟥 Networking	🟥 Networking

🟦 You manage

🟥 Service provider manages

Key Concept: Hardware Virtualization



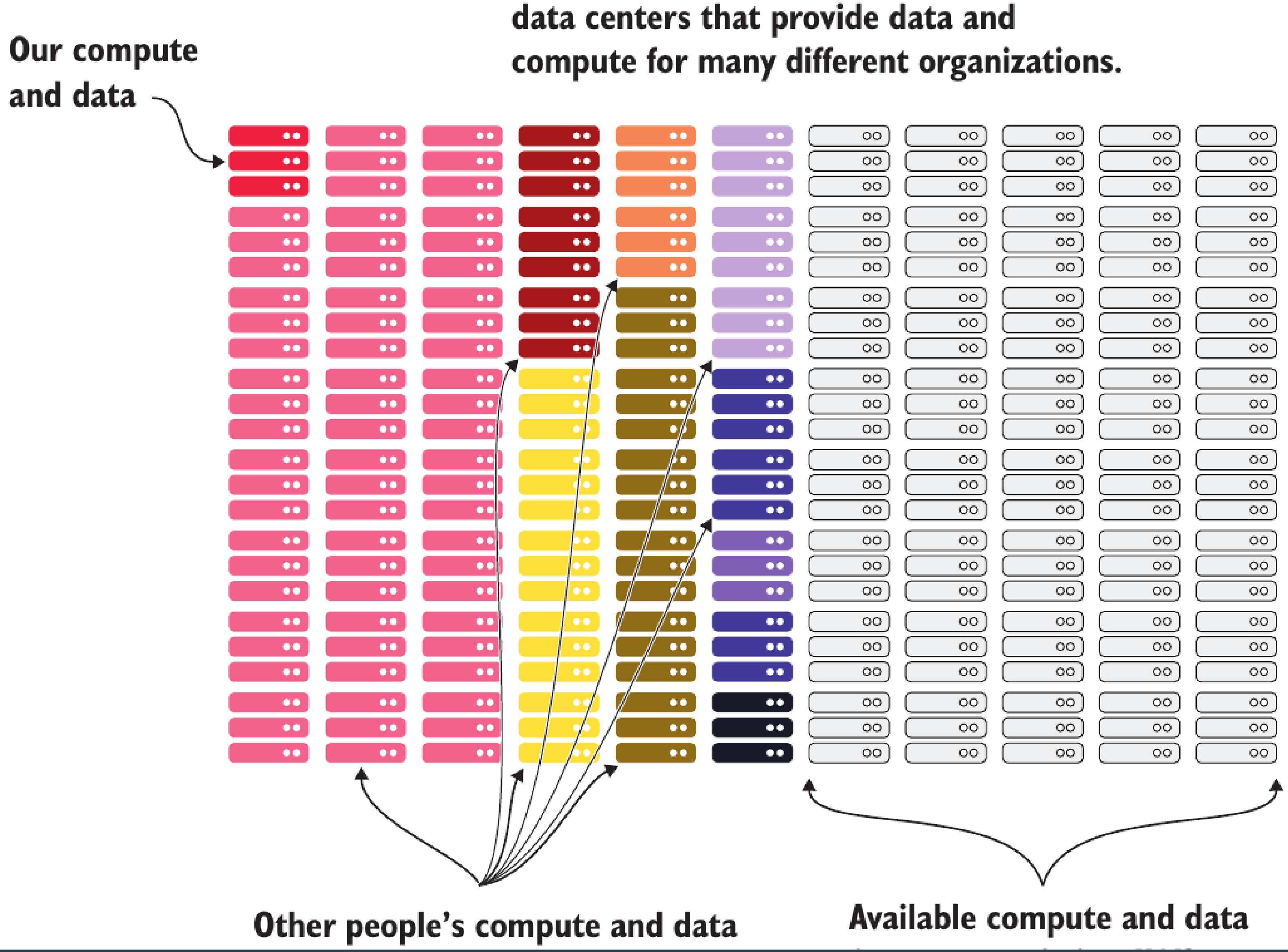
Without VMs: Single OS owns all hardware resources

With VMs: Multiple OSes share hardware resources

- **Share hardware resources at the server level**
  - Consolidation allows more efficient use of resources

- **Virtual Machine (VM)**
  - Easier to provision and deploy
  - Can be replicated, stopped, restarted, moved...

Limitless Storage and Computation



data centers that provide data and compute for many different organizations.

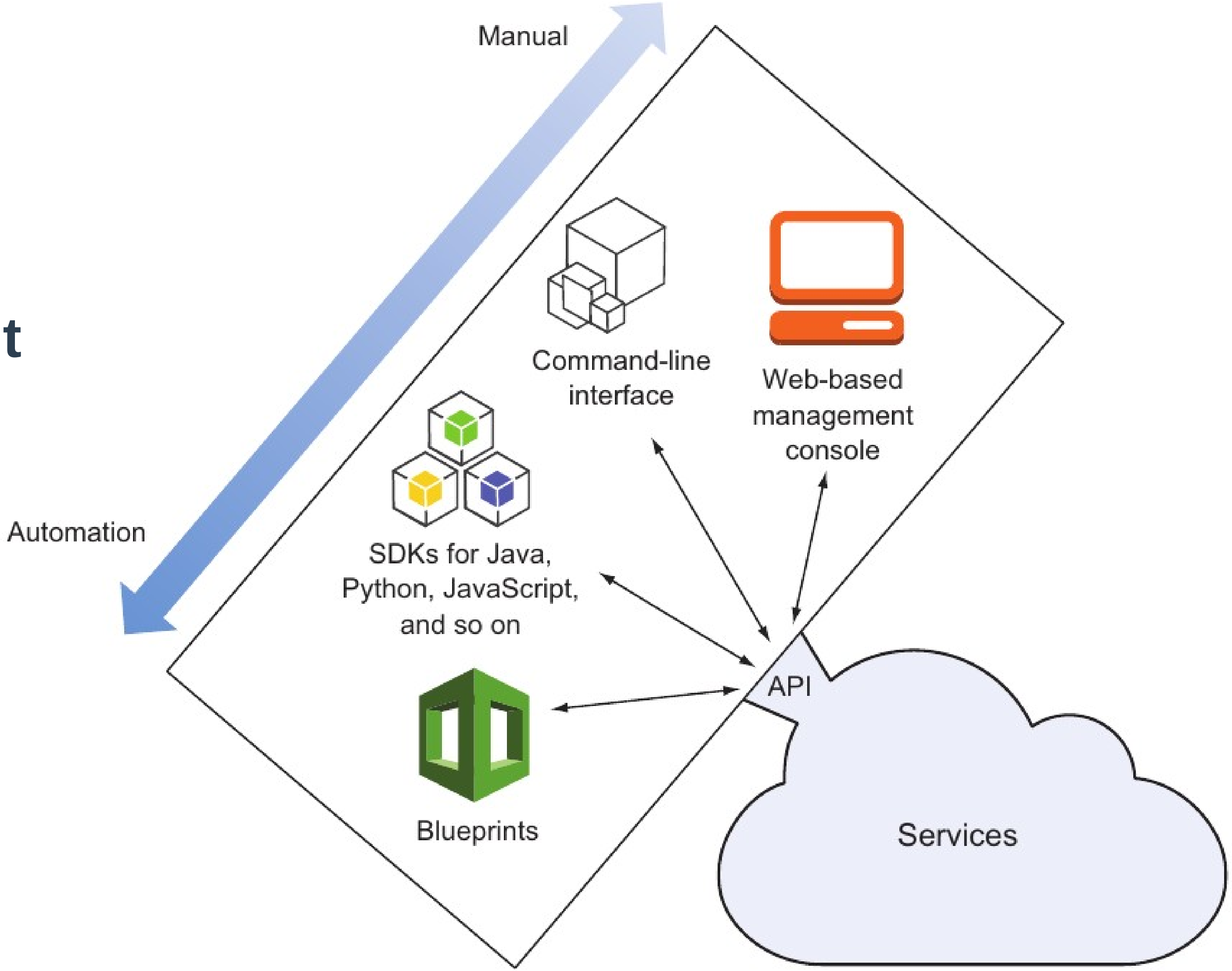
Our compute and data

Other people's compute and data

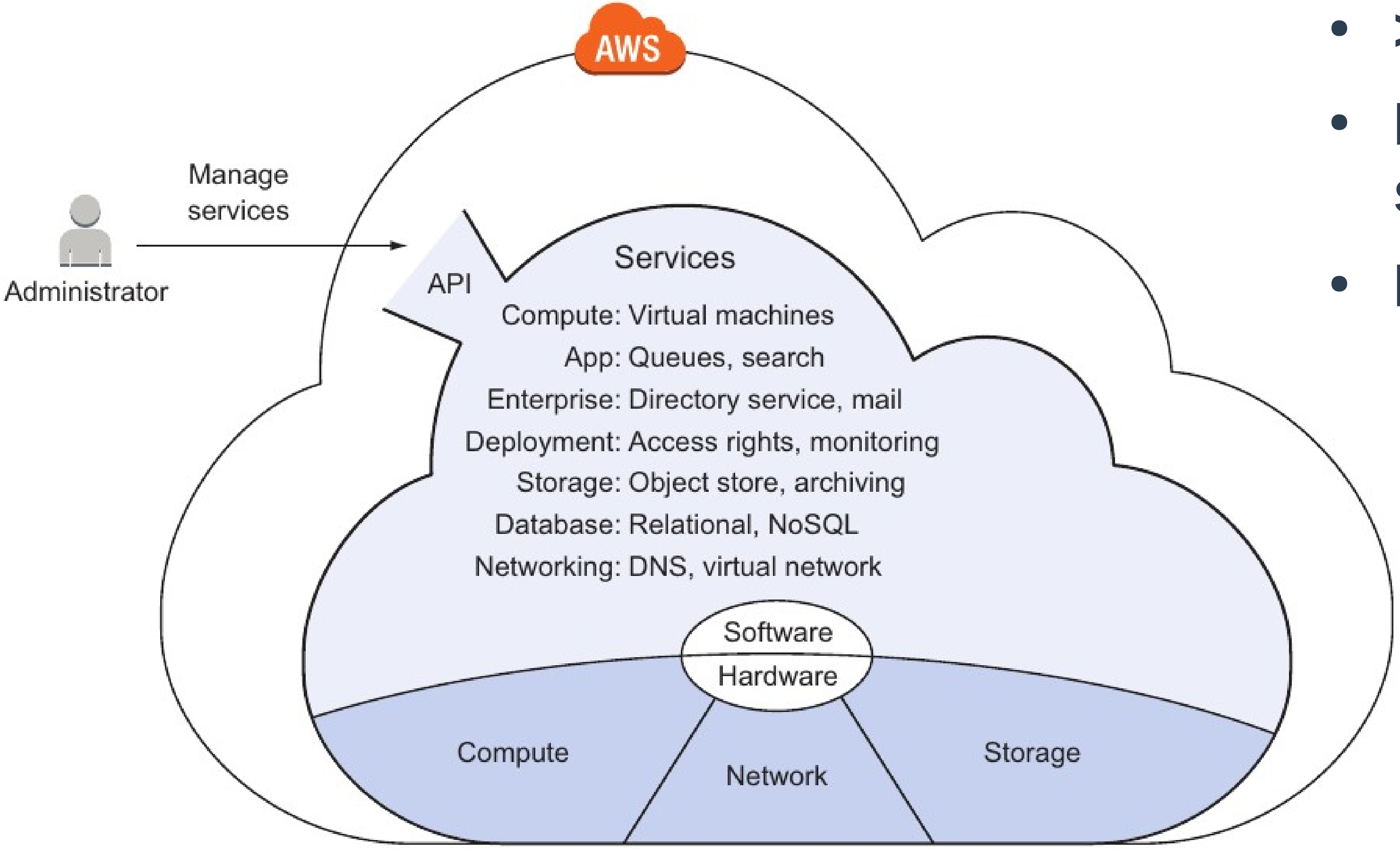
Available compute and data

Interacting with Cloud Services

- GUI
- CLI
- API
- Blueprint


Manual
Automation
Command-line interface
Web-based management console
SDKs for Java, Python, JavaScript, and so on
Blueprints
API
Services

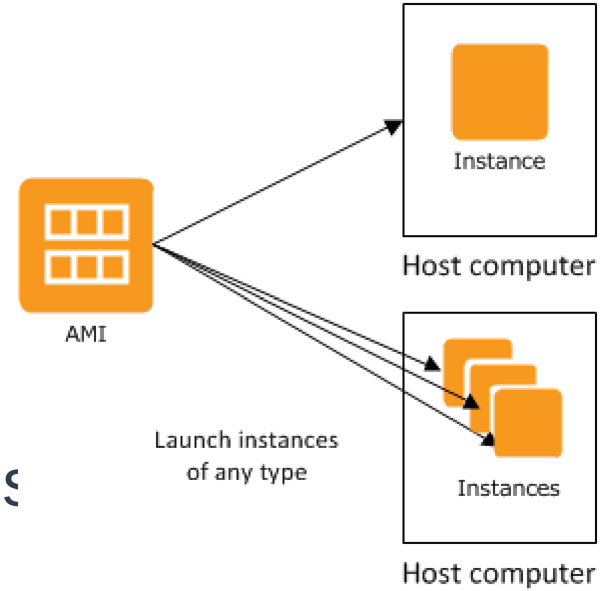
AWS: Amazon Web Services


AWS
Manage services
Administrator
API
Services
Compute: Virtual machines
App: Queues, search
Enterprise: Directory service, mail
Deployment: Access rights, monitoring
Storage: Object store, archiving
Database: Relational, NoSQL
Networking: DNS, virtual network
Software
Hardware
Compute
Network
Storage

- **> 100 !**
- **Hardware and software service**
- **Based on VM**
  - Which are exposed as a service
  - And accessed with *ssh*

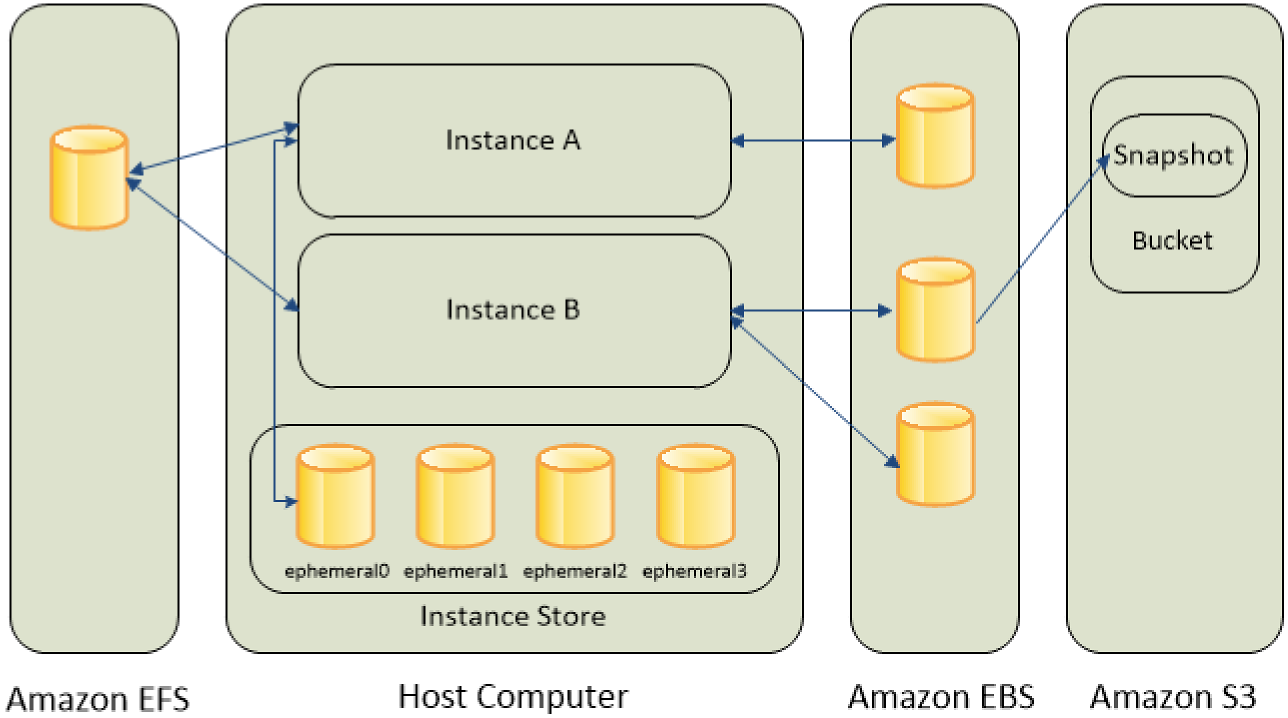
AWS Compute Services

- **Elastic Cloud Compute (EC2)**
  - Virtual Server Instance
  - Copy of an AMI (Amazon Machine Image)
    - Software configuration (OS,applications
  - Instance type
    - Hardware configuration (CPU/GPU cores, memory, storage)
  - On-demand or reserved instance
- **Elastic Map Reduce (EMR)**
- **...**

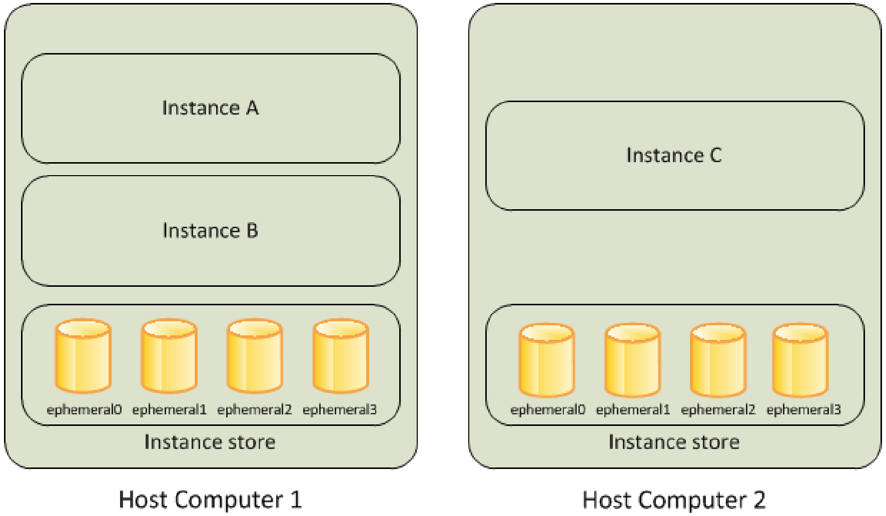


AWS Data Storage Services

- **Used by an EC2 instance**
- **EC2 Ephemeral Storage**
- **EBS: Elastic Block Store**
- **EFS: Elastic File System**
- **S3: Simple Storage Service**

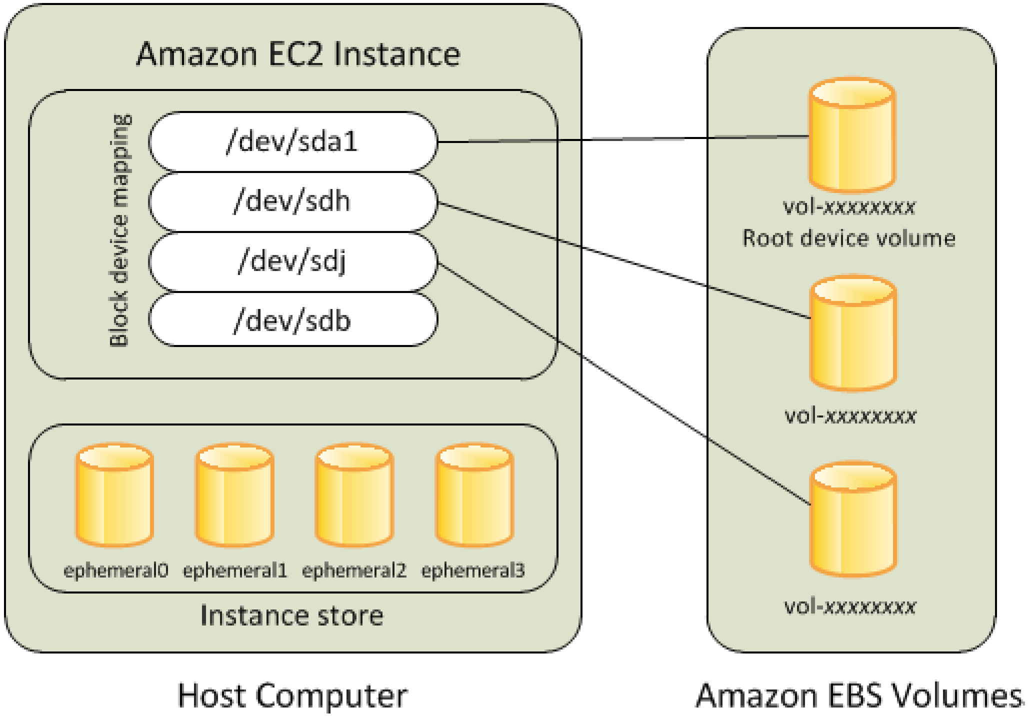


Amazon EC2 Instance Store



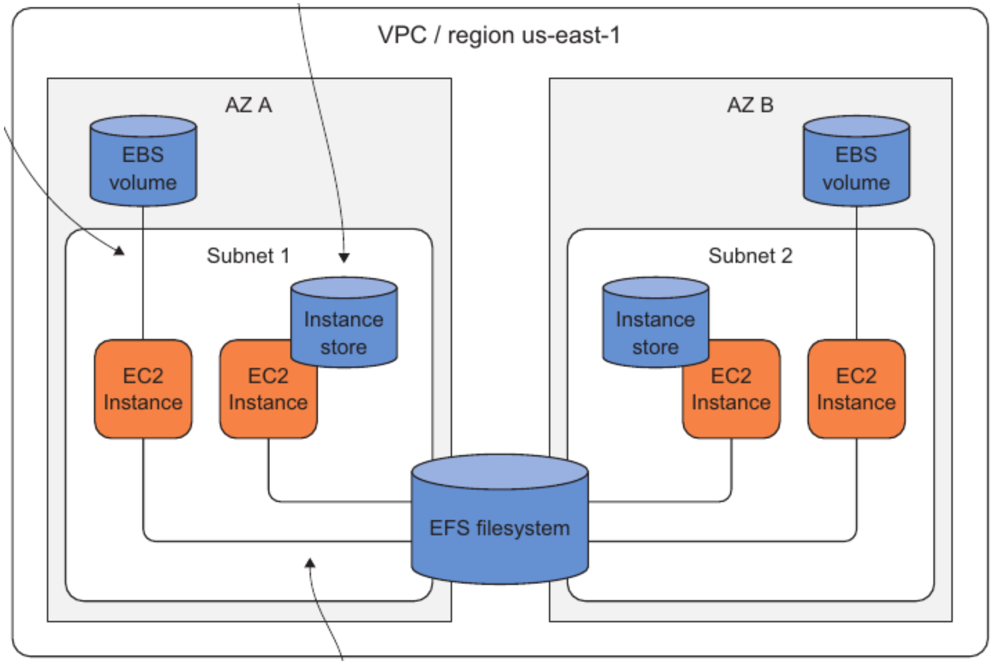
- **Temporary block level storage for each EC2 instance**
- **Storage located on disk (HDD or SSD) attached to the host computer**
- **Content disappears when the instance terminates**

Amazon Elastic Block Store (EBS)



- **Low latency**
- **Low bandwidth**
- **Persistent**
- **Scalable**
- **Attached to one instance**
- **Use cases**
  - databases,
  - I/O intensive applications

Amazon Elastic File System (EFS)



- **A network file-system (it uses NFSv4)**
- **Scalable**
- **Persistent**
- **Highly available**
- **Shares data volumes between multiple instances**
- **Use cases:**
  - Home directories on a cluster

Amazon Simple Storage Service (S3)

- **A web service**
- **Object storage: file + metadata**
- **Objects are stored into containers called buckets**
  - Named as `s3://ubs-ais-2020/20200101.csv.bz2`
- **May be accessed and managed by:**
  - The Amazon console UI
  - A web browser
  - AWS CLI
  - An API (programmatically with e.g. Boto)

- **Supports versioning (multiple variants of an object)**
- **High latency, low bandwidth, low cost**
- **Use cases:**
  - Private and public big data sets
    - Registry of Open Data:

https://registry.opendata.aws

  - Backups

Amazon EMR File System (EMRFS)

- **part of Elastic Map Reduce**
- **EMR: Amazon implementation of Hadoop tools as web services**
  - Clusters of EC2 instances
  - Life span of HDFS limited to those of the EC2 instances
- **EMRFS: add-on of HDFS used by EMR clusters**
  - Provides the convenience of accessing and storing persistent data in S3