Database Systems
11 Distributed Storage

Matthias Boehm, Arnab Phani

Graz University of Technology, Austria
Computer Science and Biomedical Engineering
Institute of Interactive Systems and Data Science
BMVIT endowed chair for Data Management

SCIENCE
PASSION
TECHNOLOGY

Last update: Jan 10, 2020

Announcements/Org

- **#1 Video Recording**
  - Link in TeachCenter & Tube (lectures will be public)
- **#2 Exercises**
  - Exercise 1 graded, feedback in TC, office hours
  - Exercise 2 graded, feedback in TC, office hours
  - **Exercise 3 in progress of being graded**
- **#3 Exam Dates**
  - **Jan 30, 5.30pm, HS i13**
  - **Jan 31, 5.30pm, HS i13**
  - **Feb 3, 7.30pm, HS i13**
  - **Feb 6, 4.00pm, HS i13**
  - **Feb 7, 4.00pm, HS i13**
- **#4 Course Evaluation**
  - Evaluation period: **Jan 14 – Feb 14**
  - Please, participate w/ honest feedback (pos/neg)

56.4%

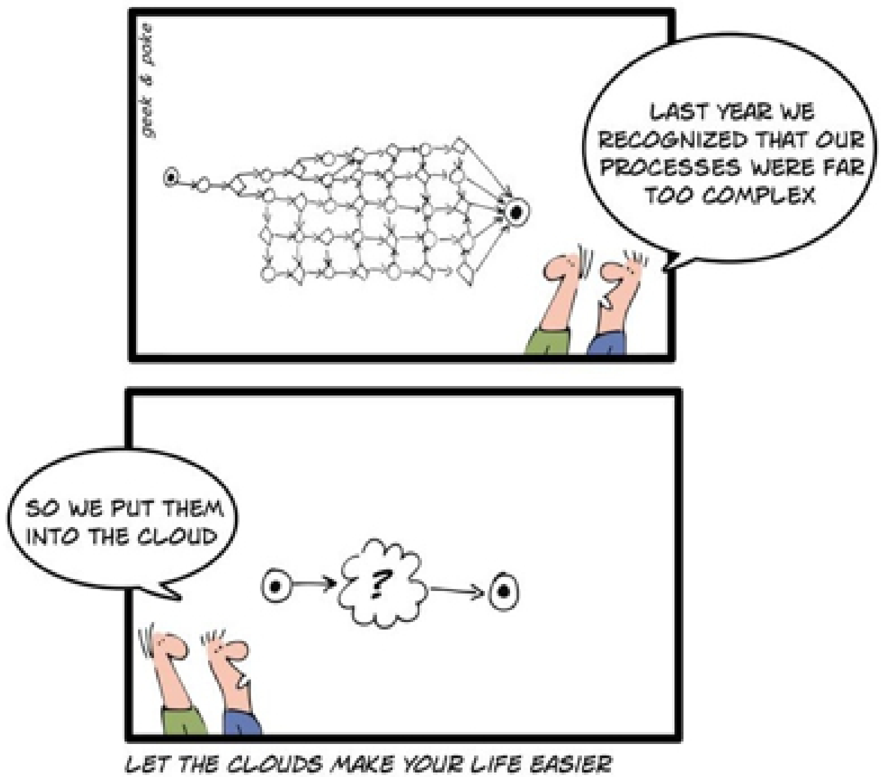
All draft modes accepted

Exam starts +10min, working time: 90min
(no lecture materials)

Agenda

- **Cloud Computing Overview**
- **Distributed Data Storage**
- **Distributed Data Analysis**
- **Exercise 4: Large-Scale Data Analysis**

Data Integration and Large-Scale Analysis (DIA)
(bachelor/master)



Cloud Computing Overview

Motivation Cloud Computing

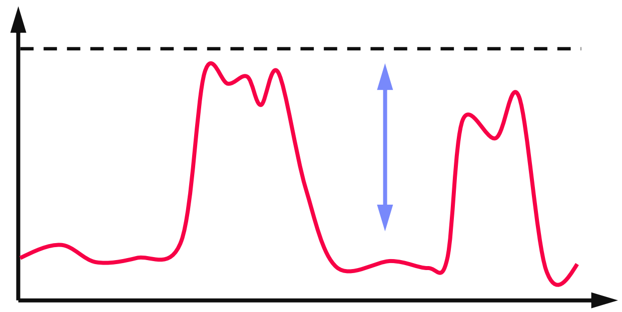
- **Definition Cloud Computing**
  - **On-demand, remote storage and compute resources, or services**
  - **User:** computing as a utility (similar to energy, water, internet services)
  - **Cloud provider:** computation in data centers / multi-tenancy

- **Service Models**
  - **IaaS: Infrastructure as a service** (e.g., storage/compute nodes)
  - **PaaS: Platform as a service** (e.g., distributed systems/frameworks)
  - **SaaS: Software as a Service** (e.g., email, databases, office, github)

→ Transforming IT Industry/Landscape
  - Since ~2010 increasing move from on-prem to cloud resources
  - System software licenses become increasingly irrelevant
  - Few cloud providers dominate IaaS/PaaS/SaaS markets (w/ 2018 revenue): **Microsoft Azure Cloud** ($ 32.2B), **Amazon AWS** ($ 25.7B), **Google Cloud** (N/A), **IBM Cloud** ($ 19.2B), **Oracle Cloud** ($ 5.3B), **Alibaba Cloud** ($ 2.1B)

Motivation Cloud Computing, cont.

- **Argument #1: Pay as you go**
  - No upfront cost for infrastructure
  - Variable utilization ➔ over-provisioning
  - **Pay per use or acquired resources**



- **Argument #2: Economies of Scale**
  - Purchasing and managing IT infrastructure at scale ➔ **lower cost** (applies to both HW resources and IT infrastructure/system experts)
  - Focus on **scale-out on commodity HW** over scale-up ➔ **lower cost**

- **Argument #3: Elasticity**
  - Assuming perfect scalability, work done in **constant time * resources**
  - Given virtually unlimited resources allows to reduce time as necessary

100 days @ 1 node

≈

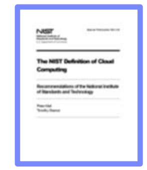
1 day @ 100 nodes

(but beware Amdahl's law: max speedup **sp = 1/s**)

Characteristics and Deployment Models

- **Extended Definition**
  - ANSI recommended definitions for service types, characteristics, deployment models

[Peter Mell and Timothy Grance: The NIST Definition of Cloud Computing, **NIST 2011**]

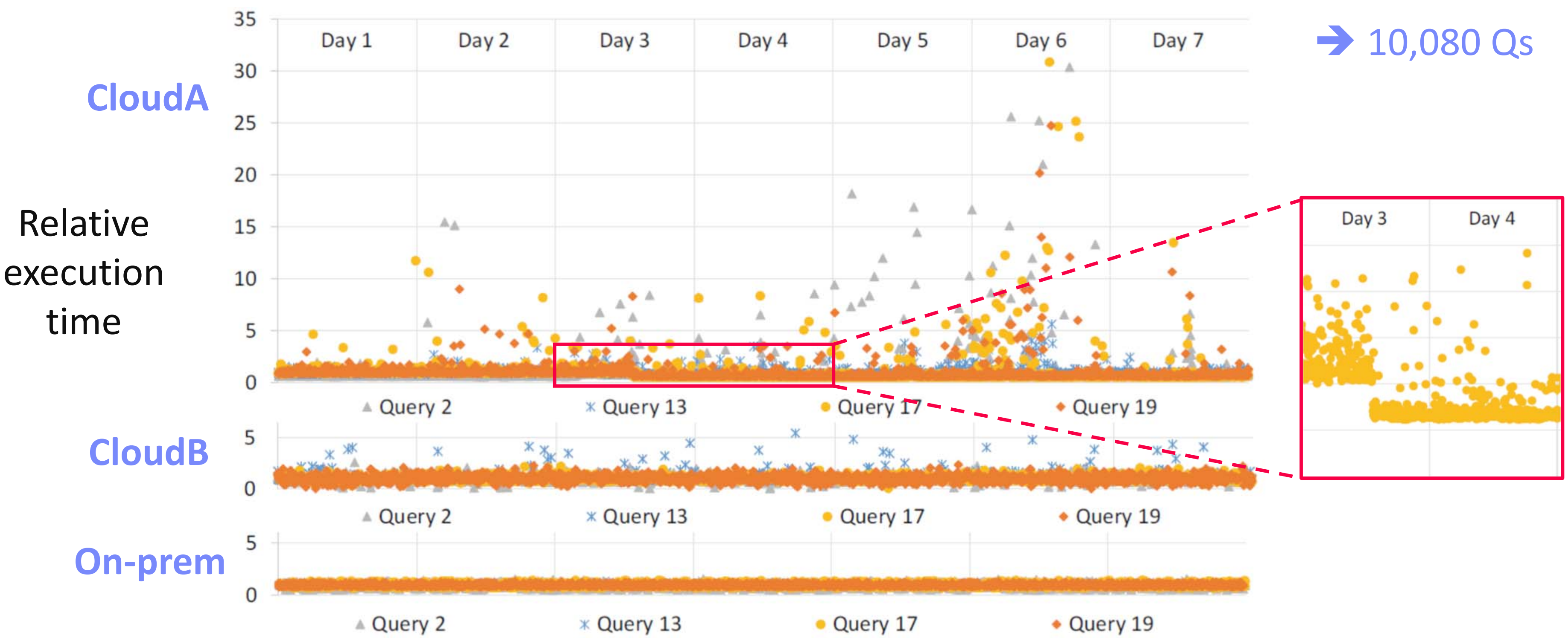


- **Characteristics**
  - **On-demand self service:** unilateral resource provision
  - **Broad network access:** network accessibility
  - **Resource pooling:** resource virtualization / multi-tenancy
  - **Rapid elasticity:** scale out/in on demand
  - **Measured service:** utilization monitoring/reporting
- **Deployment Models**
  - **Private cloud:** single org, on/off premises
  - **Community cloud:** single community (one or more orgs)
  - **Public cloud:** general public, on premise of cloud provider
  - **Hybrid cloud:** combination of two or more of the above

Excursus: 1 Query/Minute for 1 Week

- **Experimental Setup**
  - 1GB TPC-H database, 4 queries on 2 cloud DBs / 1 on-prem DB

[Tim Kiefer, Hendrik Schön, Dirk Habich, Wolfgang Lehner: **A Query, a Minute**: Evaluating Performance Isolation in Cloud Databases. TPCTC 2014]



Anatomy of a Data Center

Commodity CPU:
Xeon E5-2440: 6/12 cores
Xeon Gold 6148: 20/40 cores

Server:
Multiple sockets,
RAM, disks

Rack:
16-64 servers +
top-of-rack switch

Cluster:
Multiple racks + cluster switch

Data Center:
>100,000 servers

[Google Data Center, Eemshaven, Netherlands]

Fault Tolerance

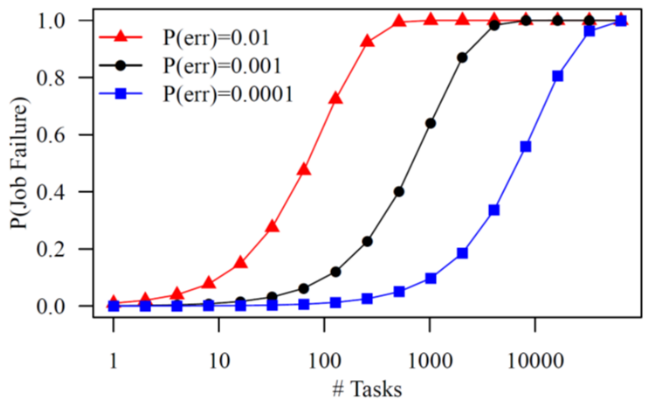
[Christos Kozyrakis and Matei Zaharia: CS349D: Cloud Computing Technology, lecture, **Stanford 2018**]

- **Yearly Data Center Failures**
  - ~0.5 overheating (power down most machines in <5 mins, ~1-2 days)
  - ~1 PDU failure (~500-1000 machines suddenly disappear, ~6 hrs)
  - ~1 rack-move (plenty of warning, ~500-1000 machines powered down, ~6 hrs)
  - ~1 network rewiring (rolling ~5% of machines down over 2-day span)
  - ~20 rack failures (40-80 machines instantly disappear, 1-6 hrs)
  - ~5 racks go wonky (40-80 machines see 50% packet loss)
  - ~8 network maintenances (~30-minute random connectivity losses)
  - ~12 router reloads (takes out DNS and external vIPs for a couple minutes)
  - ~3 router failures (immediately pull traffic for an hour)
  - ~dozens of minor 30-second blips for dns
  - ~1000 individual machine failures (2-4% failure rate, at least twice)
  - ~thousands of hard drive failures (1-5% of all disks will die)

Fault Tolerance, cont.

- **Other Common Issues**
  - **Configuration issues**, partial SW updates, SW bugs
  - **Transient errors:** no space left on device, memory corruption, stragglers

- **Recap: Error Rates at Scale**
  - Cost-effective commodity hardware
  - Error rate increases with increasing scale
  - Fault Tolerance for distributed/cloud storage and data analysis



➔ Cost-effective Fault Tolerance
  - **BASE** (basically **available**, soft state, **eventual consistency**)
  - Effective techniques
    - ECC (error correction codes), CRC (cyclic redundancy check) for detection
    - **Resilient storage:** replication/erasure coding, checkpointing, and lineage
    - **Resilient compute:** task re-execution / speculative execution

Containerization

- **Docker Containers**
  - **Shipping container analogy**
    - Arbitrary, self-contained goods, standardized units
    - Containers reduced loading times ➔ efficient international trade
  - **#1 Self-contained package** of necessary SW and data (read-only image)
  - **#2 Lightweight virtualization** w/ resource isolation via cgroups

- **Cluster Schedulers**
  - Container orchestration: scheduling, deployment, and management
  - Resource negotiation with clients
  - Typical resource bundles (CPU, memory, device)
  - Examples: **Kubernetes**, **Mesos**, (**YARN**), **Amazon ECS**, **Microsoft ACS**, **Docker Swarm**

[Brendan Burns, Brian Grant, David Oppenheimer, Eric Brewer, John Wilkes: Borg, Omega, and Kubernetes. **CACM 2016**]

➔ **from machine- to application-oriented scheduling**

Example Amazon Services – Pricing (current gen)

- **Amazon EC2 (Elastic Compute Cloud)**
  - IaaS offering of different node types and generations
  - **On-demand**, **reserved**, and **spot** instances

	vCores		Mem		
m4.large	2	6.5	8 GiB	EBS Only	$0.12 per Hour
m4.xlarge	4	13	16 GiB	EBS Only	$0.24 per Hour
m4.2xlarge	8	26	32 GiB	EBS Only	$0.48 per Hour
m4.4xlarge	16	53.5	64 GiB	EBS Only	$0.96 per Hour
m4.10xlarge	40	124.5	160 GiB	EBS Only	$2.40 per Hour
m4.16xlarge	64	188	256 GiB	EBS Only	$3.84 per Hour

- **Amazon ECS (Elastic Container Service)**
  - PaaS offering for Docker containers
  - Automatic setup of Docker environment

Pricing according to EC2
(in EC2 launch mode)

- **Amazon EMR (Elastic Map Reduce)**
  - PaaS offering for Hadoop workloads
  - Automatic setup of YARN, HDFS, and specialized frameworks like Spark
  - **Prices in addition to EC2 prices**

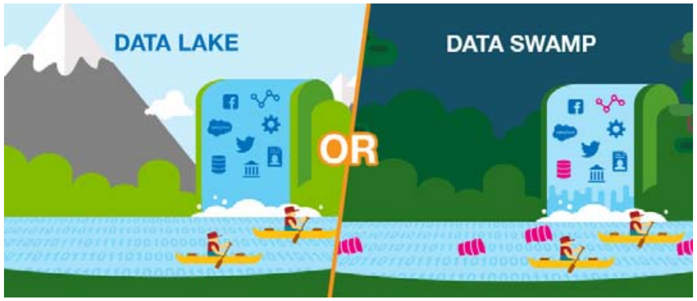
m4.large	$0.117 per Hour	$0.03 per Hour
m4.xlarge	$0.234 per Hour	$0.06 per Hour
m4.2xlarge	$0.468 per Hour	$0.12 per Hour
m4.4xlarge	$0.936 per Hour	$0.24 per Hour
m4.10xlarge	$2.34 per Hour	$0.27 per Hour
m4.16xlarge	$3.744 per Hour	$0.27 per Hour

Distributed Data Storage

Data Lakes

- **Concept "Data Lake"**
  - **Store massive amounts of un/semi-structured, and structured data** (append only, no update in place)
  - **No need for architected schema** or upfront costs (unknown analysis)
  - Typically: file storage in open, raw formats (inputs and intermediates)
  - ➔ **Distributed storage and analytics** for scalability and agility

- **Criticism: Data Swamp**
  - Low data quality (lack of schema, integrity constraints, validation)
  - Missing meta data (context) and data catalog for search
  - ➔ **Requires proper data curation / tools** According to priorities (data governance)



[**Credit:** www.collibra.com]

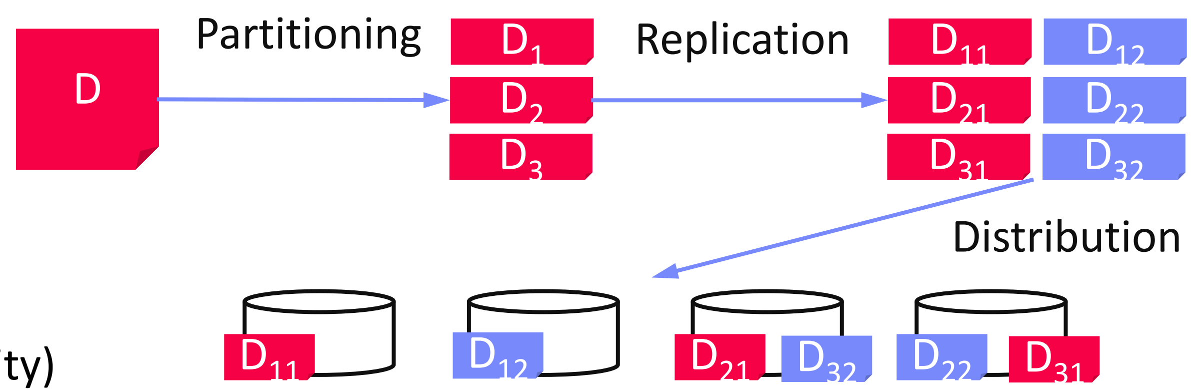
- **Excursus: Research Data Management**
  - FAIR data principles: findable, accessible, interoperable, re-usable

Object Storage

- **Recap: Key-Value Stores**
  - **Key**-value mapping, where values can be of a variety of data types
  - APIs for CRUD operations; scalability via sharding + **eventual consistency**

- **Object Store**
  - Similar to key-value stores, but: **optimized for large objects in GBs and TBs**
  - Object identifier (**key**), **meta data**, and object as binary large object (**BLOB**)
  - APIs: often REST APIs, SDKs, sometimes implementation of DFS APIs

- **Key Techniques**
  - Partitioning
  - Replication & Distribution
  - Erasure Coding (partitioning + parity)



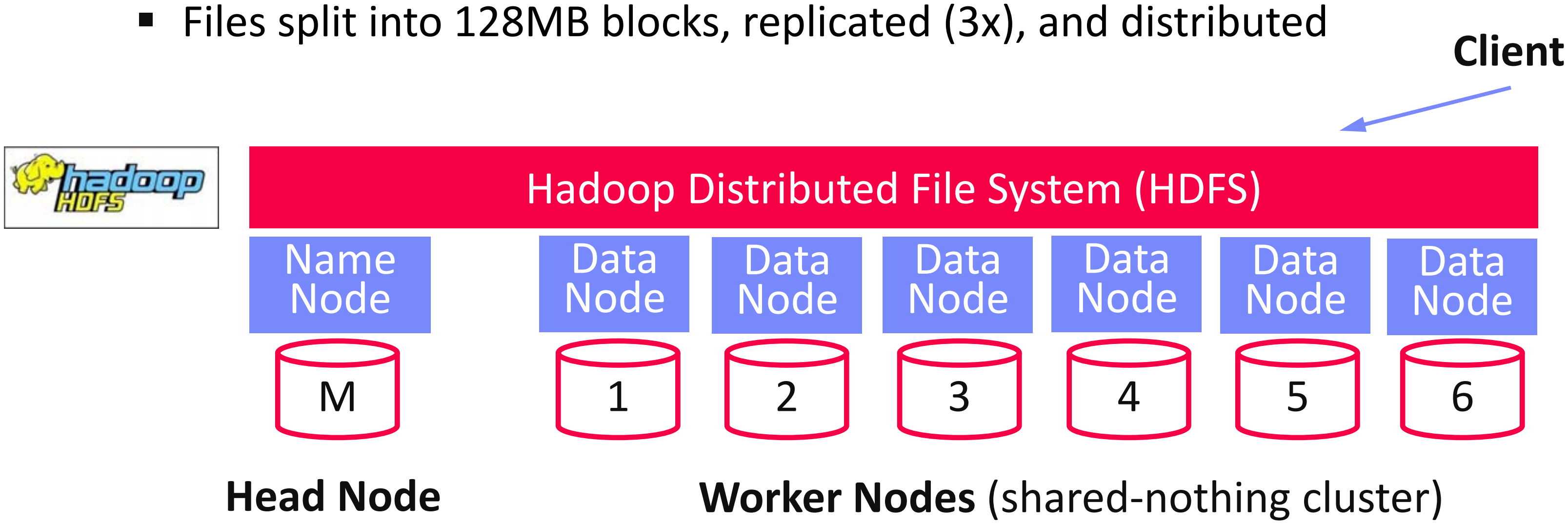
Object Storage, cont.

- **Example Object Stores / Protocols**
  - Amazon Simple Storage Service (S3)
  - OpenStack Object Storage (Swift)
  - IBM Object Storage
  - Microsoft Azure Blob Storage

- **Amazon S3**
  - Reliable object store for photos, videos, documents or any binary data
  - **Bucket:** Uniquely named, static data container
    `http://s3.aws-eu-central-1.amazonaws.com/mboehm-b1`
  - **Object:** key, version ID, value, metadata, access control
  - Single (5GB)/multi-part (5TB) upload and direct/BitTorrent download
  - **Storage classes:** STANDARD, STANDARD_IA, GLACIER, DEEP_ARCHIVE
  - **Operations:** GET/PUT/LIST/DEL, and SQL over CSV/JSON objects

Hadoop Distributed File System (HDFS)

- **Brief Hadoop History**
  - Google's GFS [SOSP'03] + MapReduce [ODSI'04] → Apache Hadoop (2006)
  - Apache Hive (SQL), Pig (ETL), Mahout (ML), Giraph (Graph)
- **HDFS Overview**
  - Hadoop's distributed file system, for large clusters and datasets
  - Implemented in Java, w/ native libraries for compression, I/O, CRC32
  - Files split into 128MB blocks, replicated (3x), and distributed



Hadoop Distributed File System, cont.

- **HDFS NameNode**
  - Master daemon that manages file system namespace and access by clients
  - Metadata for all files (e.g., replication, permissions, sizes, block ids, etc)
  - **FSImage:** checkpoint of FS namespace
  - **EditLog:** **write-ahead-log (WAL)** of file write operations (merged on startup)

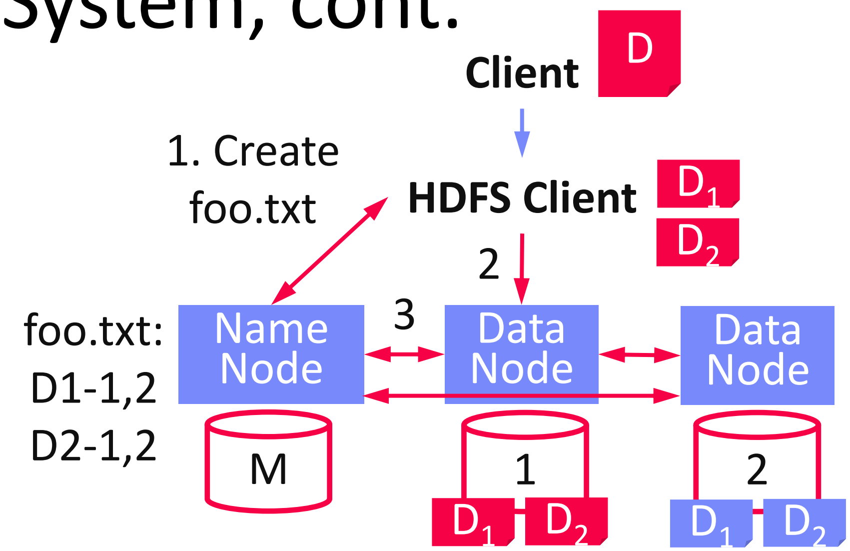
```
hadoop fs -ls ./data/mnist1m.bin
```

```
-rw-r--r--   3 mboehm hdfs  104510159 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00000
-rw-r--r--   3 mboehm hdfs  137887319 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00001
-rw-r--r--   3 mboehm hdfs  139012247 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00002
-rw-r--r--   3 mboehm hdfs  139123247 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00003
-rw-r--r--   3 mboehm hdfs  139053743 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00004
-rw-r--r--   3 mboehm hdfs  138928955 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00005
-rw-r--r--   3 mboehm hdfs  139016375 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00006
-rw-r--r--   3 mboehm hdfs  139047923 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00007
-rw-r--r--   3 mboehm hdfs  139042307 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00008
-rw-r--r--   3 mboehm hdfs  139068143 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00009
-rw-r--r--   3 mboehm hdfs  139029875 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00010
-rw-r--r--   3 mboehm hdfs  138901043 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00011
-rw-r--r--   3 mboehm hdfs  139042763 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00012
-rw-r--r--   3 mboehm hdfs  139030751 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00013
-rw-r--r--   3 mboehm hdfs  139172051 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00014
-rw-r--r--   3 mboehm hdfs  138962735 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00015
-rw-r--r--   3 mboehm hdfs  139079495 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00016
-rw-r--r--   3 mboehm hdfs   63417008 2018-10-20 22:59 /user/mboehm/data/mnist1m.bin/0-m-00017
```

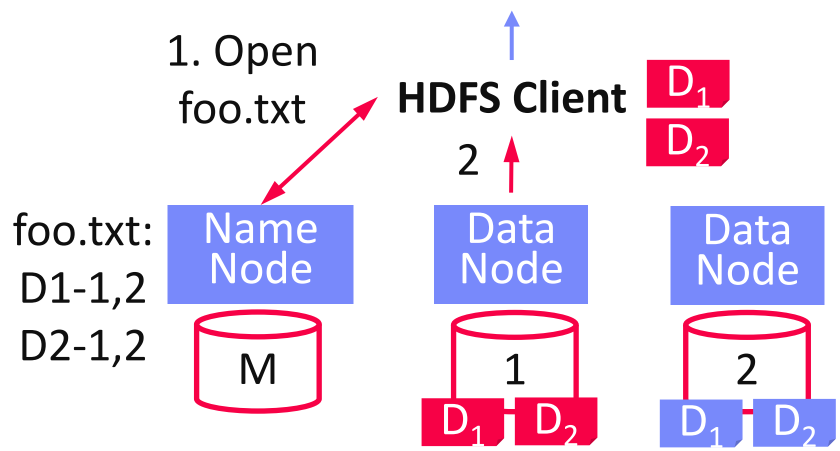
- **HDFS DataNode**
  - Worker daemon per cluster node that manages block storage (list of disks)
  - Block creation, deletion, replication as individual files in local FS
  - On startup: scan local blocks and send **block report** to name node
  - Serving block read and write requests
  - Send heartbeats to NameNode (capacity, current transfers) and receives replies (replication, removal of block replicas)

Hadoop Distributed File System, cont.

- **HDFS Write**
  - #1 Client RPC to NameNode to create file → lease/replica DNs
  - #2 Write blocks to DNs, pipelined replication to other DNs
  - #3 DNs report to NN via heartbeat



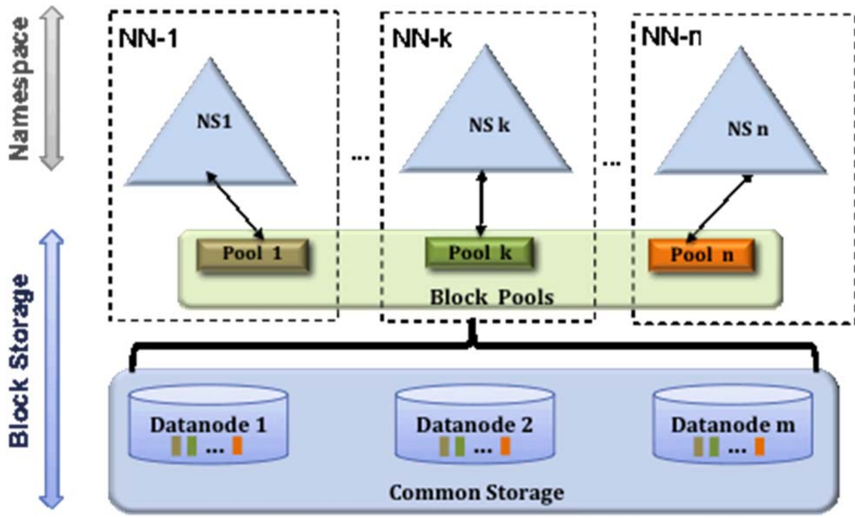
- **HDFS Read**
  - #1 Client RPC to NameNode to open file → DNs for blocks
  - #2 Read blocks sequentially from closest DN w/ block
  - InputFormats and RecordReaders as abstraction for multi-part files (incl. compression/encryption)



Hadoop Distributed File System, cont.

- **Data Locality**
  - HDFS is generally rack-aware (node-local, rack-local, other)
  - Schedule reads from closest data node
  - Replica placement (rep 3): local DN, remote-rack DN, same-remote-rack DN
  - MapReduce/Spark: locality-aware execution (**function vs data shipping**)

- **HDFS Federation**
  - Eliminate NameNode as namespace scalability bottleneck
  - Independent NameNodes, responsible for name spaces
  - DataNodes store blocks of all NameNodes
  - Client-side mount tables

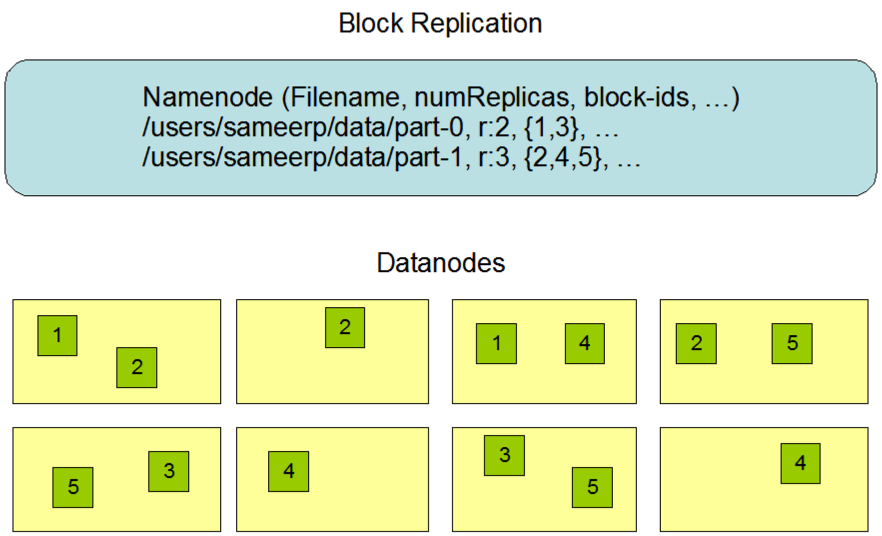


[**Credit:** https://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-hdfs/Federation.html]

BREAK (and Test Yourself)

- **Large-Scale Analysis** (6/100 points)
  - Describe means (i.e., techniques) of ensuring fault tolerance in distributed storage systems (like Hadoop Distributed File System) and distributed analysis frameworks (like Apache MapReduce or Apache Spark).

- **Solution of 1st part**
  - Data replication in HDFS [3]
  - Replica Placement Policy (node-local, rack-local, other)



[**Credit**: https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/HdfsDesign.html#Data_Replication]