SCIENCE
PASSION
TECHNOLOGY

Architecture of DB Systems
10 Cloud DBMSs

Matthias Boehm

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

Last update: Jan 13, 2021

Announcements/Org

- **#1 Video Recording**
  - Link in TeachCenter & TUbe (lectures will be public)
  - Optional attendance (independent of COVID)
- **#2 COVID-19 Restrictions** (HS i5)
  - Corona Traffic Light: **RED**
  - Temporarily webex lectures until end of semester
- **#3 Projects**
  - Deadline **Jan 21, 11.59pm**
    (**soft deadline**, submissions accepted until Feb 28)
  - Performance target: ~ `T(SUT) < 4 * T(ref_impl)`
- **#4 Course Evaluation and Exam**
  - Evaluation period: **Dec 15 – Jan 31**
  - Exam date: **Feb 19** (virtual webex oral exams, 45min each)

Agenda

- **Cloud Computing Background**
- **PaaS: SQL on Hadoop**
- **SaaS: Cloud DBs and Cloud DWHs**
- **FaaS: Serverless Database Systems**

Cloud Computing Background

Motivation Cloud Computing

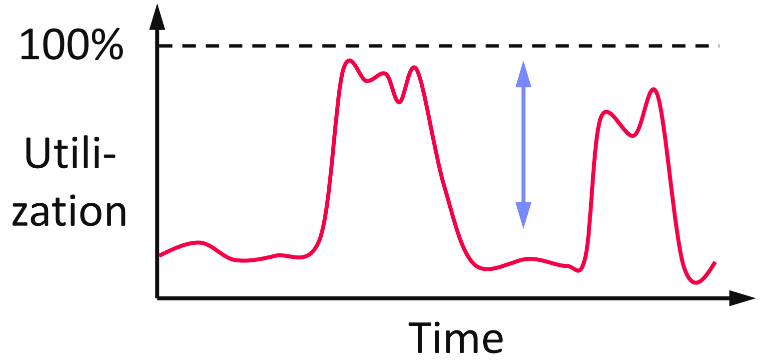
- **Definition Cloud Computing**
  - **On-demand, remote storage and compute resources, or services**
  - **User:** computing as a utility (similar to energy, water, internet services)
  - **Cloud provider:** computation in data centers / multi-tenancy
- **Service Models**
  - **IaaS: Infrastructure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

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]

Infrastructure as a Service (IaaS)

- **Overview**
  - Resources for **compute**, **storage**, **networking** as a service
    ➔ Virtualization as key enabler (simplicity and auto-scaling)
  - **Target user:** sys admin / developer

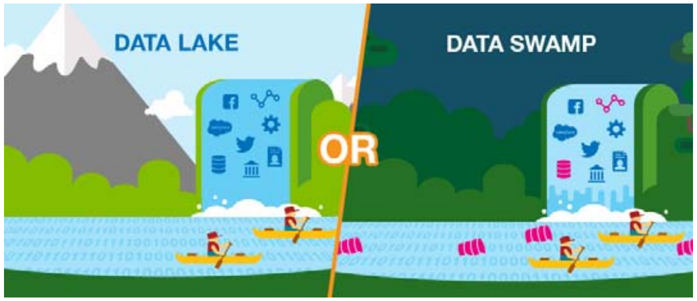
Computing as a utility

- **Storage**
  - Amazon AWS Simple Storage Service (S3)
  - OpenStack Object Storage (Swift)
  - IBM Cloud Object Storage
  - Microsoft Azure Blob Storage
- **Compute**
  - Amazon AWS Elastic Compute Cloud (EC2)
  - Microsoft Azure Virtual Machines (VM)
  - IBM Cloud Compute

PaaS: SQL on Hadoop

Data-parallel Computation

- **Concept "Data Lake"**
  - **Store massive amounts of structured and un/semi-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)



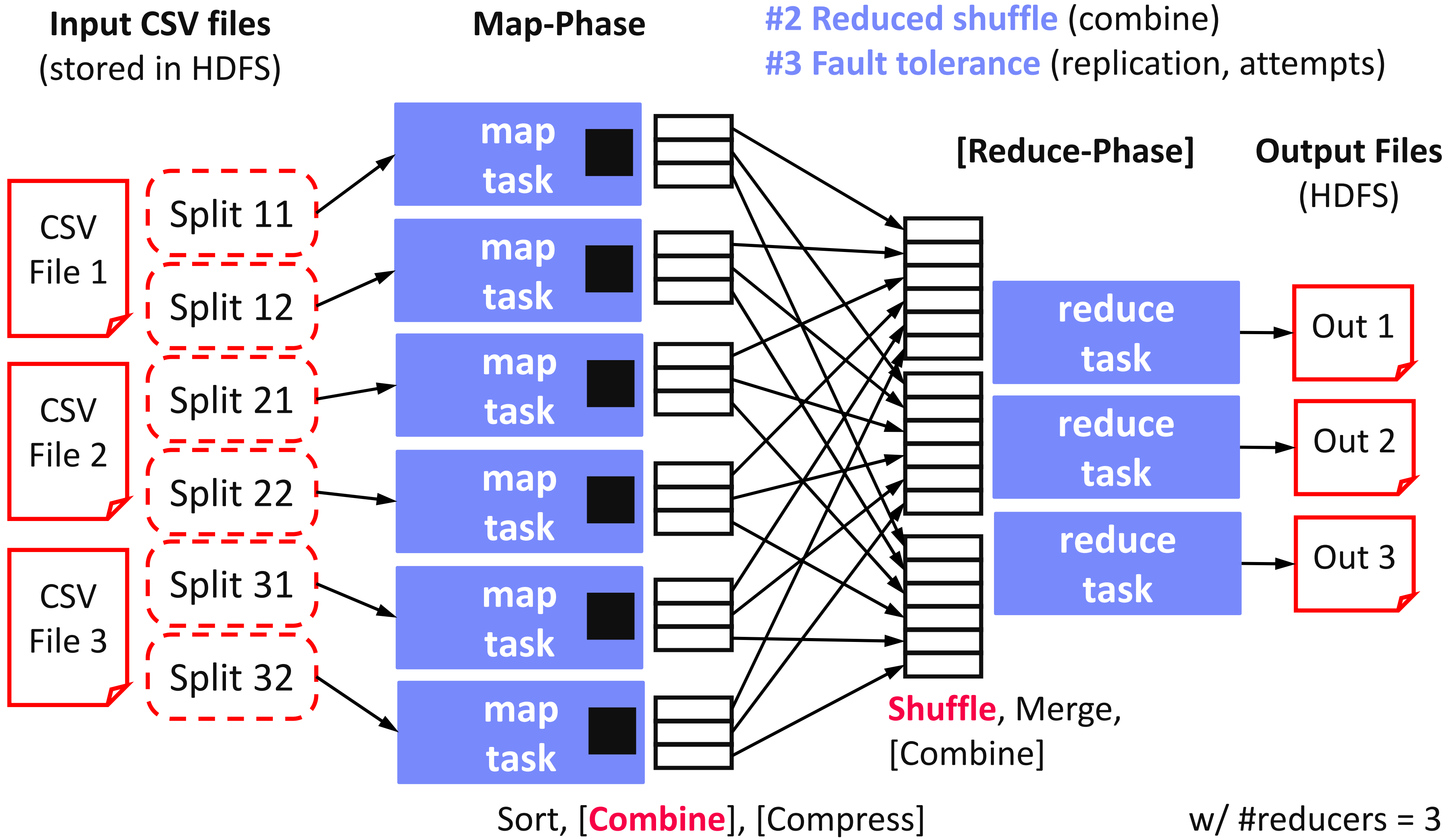
[**Credit:** www.collibra.com]

- **Distributed Storage and Analysis**
  - Central abstraction: **distributed collection** Different physical representations
  - **Easy distribution** of pairs via horizontal partitioning (aka shards, partitions)
  - Frameworks: Hadoop MR, Spark, Flink
  - Deployment: on-prem and/or **cloud**

Key	Value
4	Delta
2	Bravo
1	Alfa
3	Charlie
5	Echo
6	Foxtrot
7	Golf
1	Alfa

Recap: MapReduce – Execution Model

#1 Data Locality (delay sched., write affinity)
#2 Reduced shuffle (combine)
#3 Fault tolerance (replication, attempts)



A History on "SQL on Hadoop"

[Daniel Abadi, Shivnath Babu, Fatma Ozcan, Ippokratis Pandis: Tutorial: SQL-on-Hadoop Systems. **PVLDB 8(12) 2015**]

(see DM Exercise 4)

- **Criticism MapReduce for Data Analytics**
  - Litter control of data flow, simplicity leads to inefficiencies
  - Fault tolerance not always necessary
  - Lack of integration into existing eco system of data analysis

[Andrew Pavlo et al.: A comparison of approaches to large-scale data analysis. **SIGMOD 2009**]

[Spyros Blanas et al.: A comparison of join algorithms for log processing in MapReduce. **SIGMOD 2010**]

- **SQL on Hadoop**
  - Query engines on distributed file systems and open storage formats (e.g., CSV, Sequence files, Avro, **Parquet**, **OCR**, **Arrow**)
  - Challenges w.r.t. metadata (schema/stats), and resource management
  - Non-relational data (e.g., JSON), and unclean, irregular, unreliable data

  ➔ **Specialized "SQL on Hadoop"** systems (with open / native storage formats)

A History on "SQL on Hadoop" – Systems

- **Hadoop Eco-system**
  - **HBase:** logical tables, CRUD, key-value storage on HDFS
  - **Hive:** SQL queries executed as MapReduce jobs (OLAP)
  - **Hive on Tez/Spark:** SQL queries executed as DAGs of operations

- **Proprietary Systems**
  - **MS SCOPE**
  - **HadoopDB**/Hadapt → Teradata (2014)
  - **Facebook Presto**
  - **Cloudera Impala**
  - **IBM BigSQL**

[Ronnie Chaiken et al.: SCOPE: easy and efficient parallel processing of massive data sets. **PVLDB 1(2) 2008**]

[Azza Abouzeid et al.: HadoopDB: An Architectural Hybrid of MapReduce and DBMS Technologies for Analytical Workloads. **PVLDB 2(1) 2009**]

[Marcel Kornacker et al.: Impala: A Modern, Open-Source SQL Engine for Hadoop. **CIDR 2015**]

[Scott C. Gray, Fatma Ozcan, Hebert Pereyra, Bert van der Linden and Adriana Zubiri: SQL-on-Hadoop without compromise, **IBM Whitepaper 2014**]

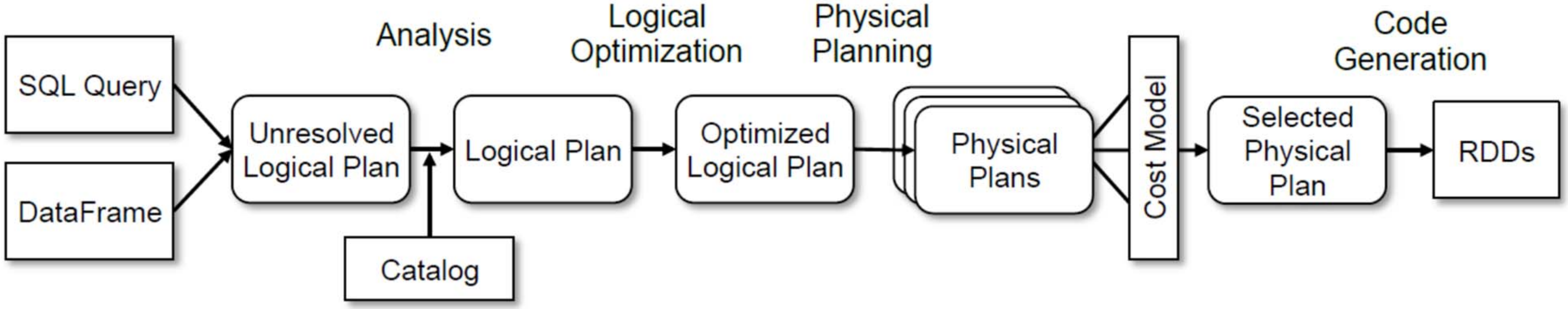
A History on “SQL on Hadoop” – SparkSQL

- **Overview** SparkSQL
  - New dataframe / dataset abstractions with various data source (+ pushdown)
  - SQL and programmatic APIs
  - Rewrite ruleset for query optimization
  - Off-heap data storage (`sun.misc.Unsafe`)
  - Whole-stage code generation

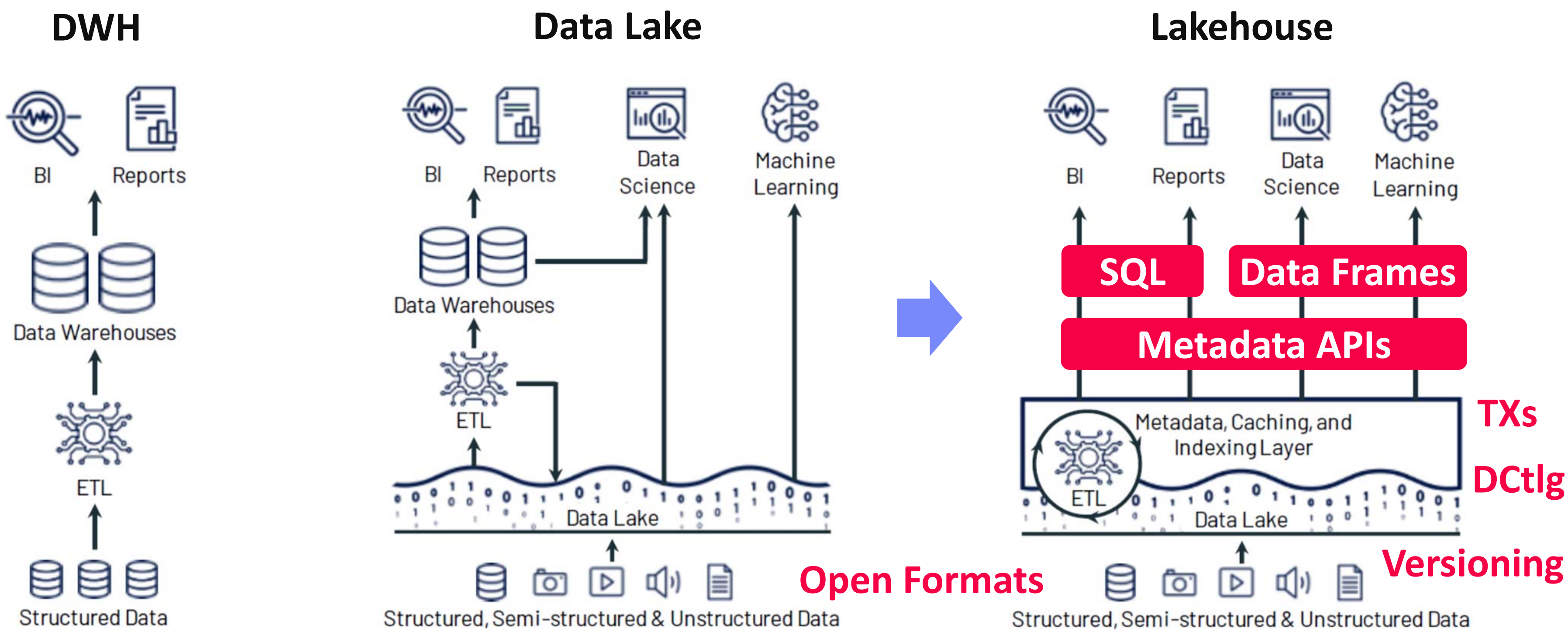
[Reynold S. Xin, Josh Rosen, Matei Zaharia, Michael J. Franklin, Scott Shenker, Ion Stoica: Shark: SQL and rich analytics at scale. **SIGMOD 2013**]

[Michael Armbrust et al.: Spark SQL: Relational Data Processing in Spark. **SIGMOD 2015**]

- **Query Planning**



Example Delta Lake (and Lakehouse Architecture)



[Michael Armbrust et al: **Delta Lake:** High-Performance ACID Table Storage over Cloud Object Stores. **PVLDB 13(12) 2020**]

[Michael Armbrust, Ali Ghodsi, Reynold Xin, Matei Zaharia: **Lakehouse:** A New Generation of Open Platforms that Unify Data Warehousing and Advanced Analytics, **CIDR 2021**]

SaaS: Cloud DBs and Cloud DWHs

Cloud Databases (DBaaS)

- **Motivation DBaaS**
  - Simplified setup, maintenance, tuning and auto scaling
  - Multi-tenant systems (scalability, learning opportunities)
  - Different types based on workload (OLTP vs OLAP, NoSQL)

Microsoft

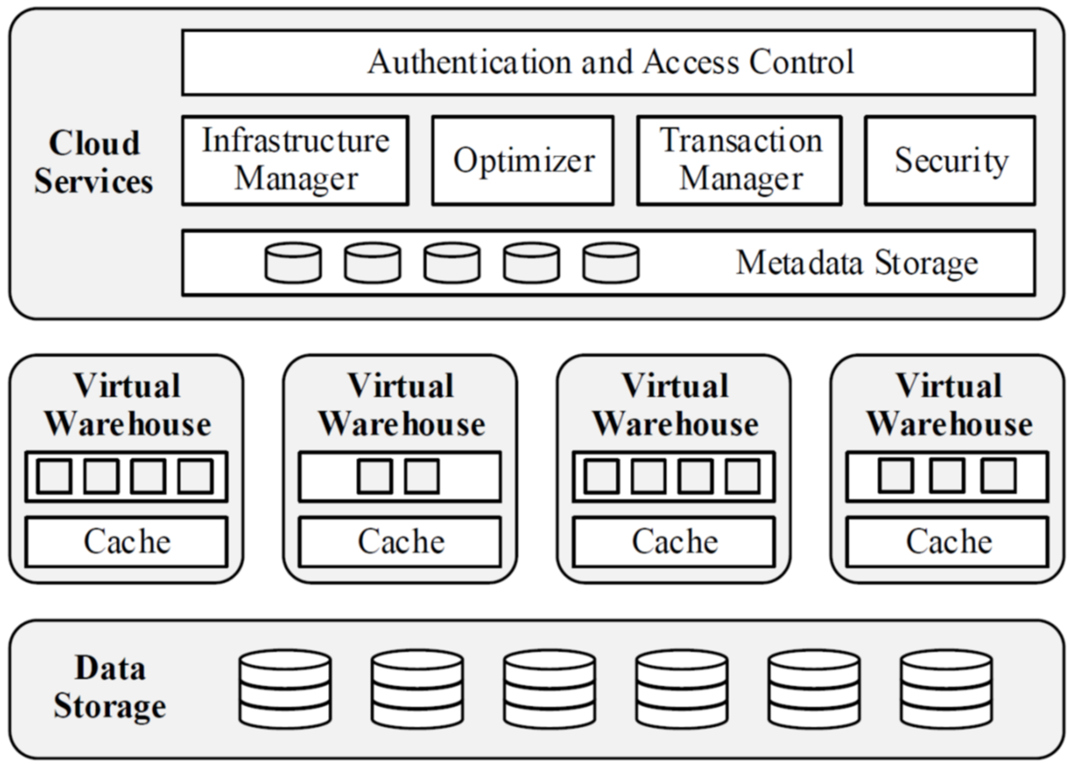
- **Elastic Data Warehouses**
  - Motivation: Intersection of data warehousing, cloud computing, distributed storage
  - Example Systems
    - **#1** Snowflake
    - **#2** Google BigQuery (Dremel)
    - **#3** Amazon Redshift
    - Azure SQL Data Warehouse / **#4** Azure SQL Database Hyperscale (Socrates)

Commonalities:
SQL, **column stores**,
data on **object store / DFS**,
elastic cloud scaling

Example Snowflake

[Benoît Dageville et al.: The Snowflake Elastic Data Warehouse. **SIGMOD 2016**]

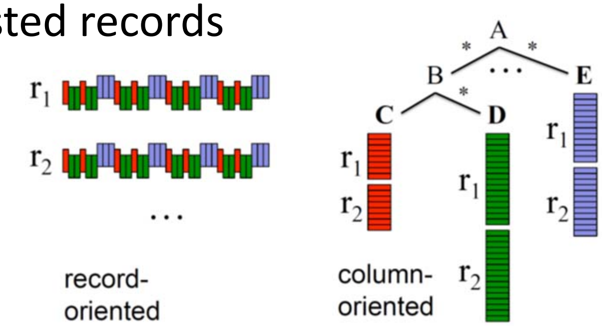
- **Motivation** (impl started late 2012)
  - **Enterprise-ready DWH solution for the cloud** (elasticity, semi-structured)
  - Pure SaaS experience, high availability, cost efficient
- **Cloud Services**
  - Manage virtual DHWs, TXs, and queries
  - Meta data and catalogs
- **Virtual Warehouses**
  - Query execution in EC2
  - Caching/intermediates
- **Data Storage**
  - **Storage in AWS S3**
  - PAX / hybrid **columnar**
  - **Min-max pruning**



Example Google BigQuery

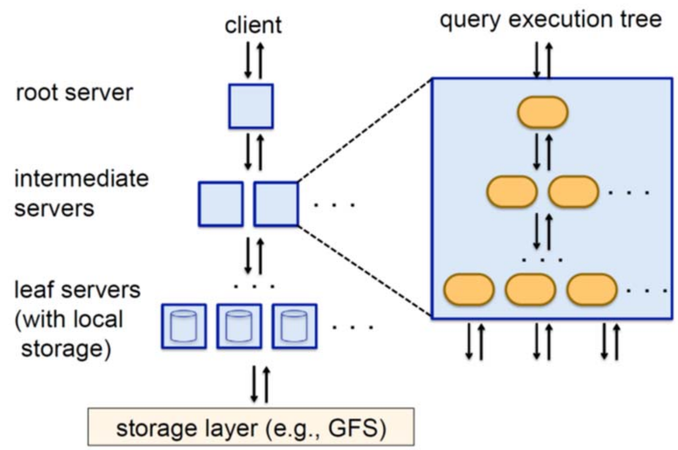
[Sergey Melnik et al.: Dremel: Interactive Analysis of Web-Scale Datasets. **PVLDB 3(1) 2010**]

- **Background Dremel**
  - Scalable and fast in-situ analysis of read-only nested data (DFS, BigTable)
  - **Data model:** protocol buffers - strongly-typed nested records
  - **Storage model:** columnar storage of nested data (efficient splitting and assembly records)
  - Query execution via multi-level serving tree



- **BigQuery System Architecture**
  - Public impl of internal Dremel system (2012)
  - SQL over structured, nested data (OLAP, BI)
  - **Extensions:** web Uis, REST APIs and ML
  - **Data storage:** Colossus (NextGen GFS)

[Kazunori Sato: An Inside Look at Google BigQuery, Google BigQuery White Paper 2012.]

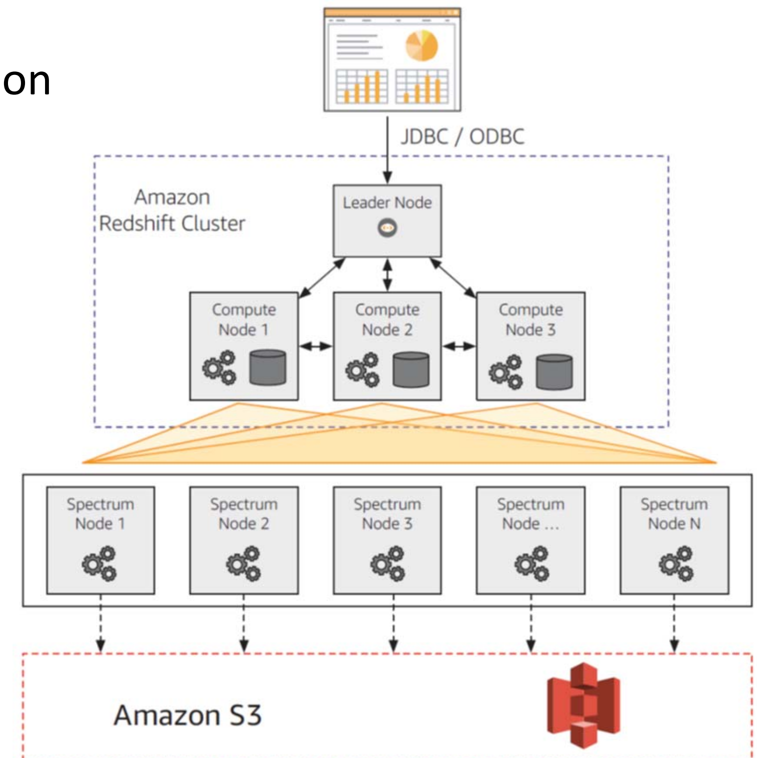


Example Amazon Redshift

[Anurag Gupta et al.: Amazon Redshift and the Case for Simpler Data Warehouses. **SIGMOD 2015**]

[Mengchu Cai et al.: Integrated Querying of SQL database data and S3 data in Amazon Redshift. **IEEE Data Eng. Bull. 41(2) 2018**]

- **Motivation** (release 02/2013)
  - **Simplicity and cost-effectiveness** (fully-managed DWH at petabyte scale)
- **System Architecture**
  - **Data plane:** data storage and SQL execution
  - **Control plane:** workflows for monitoring, and managing databases, AWS services
- **Data Plane**
  - Initial engine licensed from ParAccel
  - Leader node + sliced compute nodes in **EC2** (with **local storage**)
  - Replication across nodes + **S3 backup**
  - **Query compilation** in C++ code
  - Support for **flat and nested files**

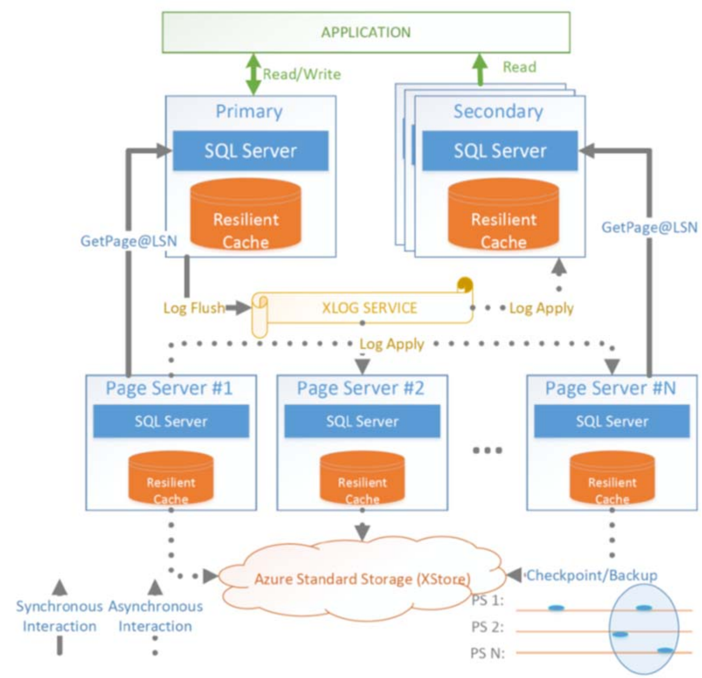


Example Microsoft Hyperscale (OLTP)

[Panagiotis Antonopoulos et al.: Socrates: The New SQL Server in the Cloud. **SIGMOD 2019**]

- **Overview**
  - Challenges of monolithic DBMSs in the cloud (cost-elasticity → scale-out/in data movement, availability/SW updates)
  - Socrates: new OLTP cloud database system **Azure DB Hyperscale**

- **Key Features**
  - **Separated Compute, Storage, Log**
  - **SQL Server** compute node w/ secondary and SSD-based caching
  - 128GB Page Servers → up to 100TB DB
  - Log server (landing zone, long-term log storage)
  - Azure storage layer
  - Period checkpointing



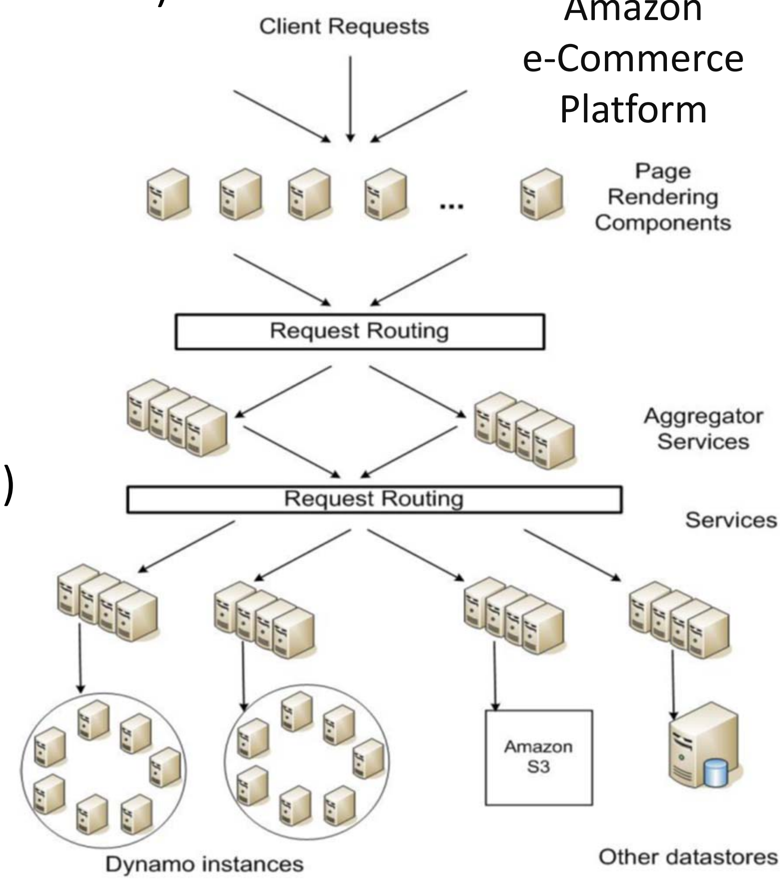
[Microsoft Mechanics: What is Azure Database Hyperscale?, https://www.youtube.com/watch?v=Z9AFnKI7sfI]

Example Dynamo (KV Store)

[Giuseppe DeCandia et al: Dynamo: amazon's highly available **key-value store**. **SOSP 2007**]

- **Motivation**
  - **Simple**, **highly-available** data storage for small objects in ~1MB range
  - Aim for **good load balance** (99.9th percentile SLAs)
- **#1 System Interface**
  - Simple get(k, ctx) and put(k, ctx) ops
- **#2 Partitioning**
  - **Consistent hashing** of nodes and keys on circular ring for **incremental scaling**
  - Nodes hold **multiple virtual nodes** for **load balance** (add/rm, heterogeneous)
- **#3 Replication**
  - Each data item **replicated N times** (at coord node and N-1 successors)
  - Eventual consistency with async update propagation based on **vector clocks**
  - Replica synchronization via **Merkle trees**

Amazon e-Commerce Platform

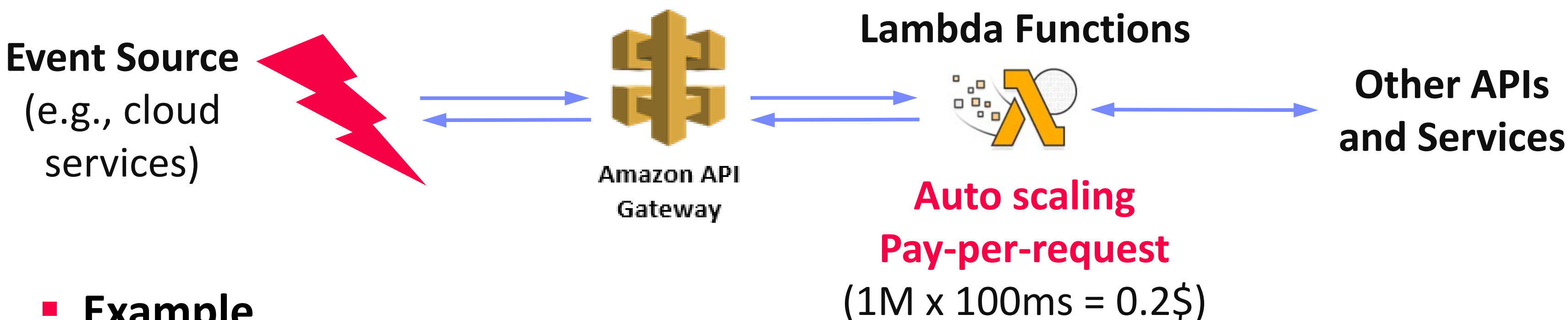


FaaS: Serverless Database Systems

Serverless Computing

[Joseph M. Hellerstein et al: Serverless Computing: One Step Forward, Two Steps Back. **CIDR 2019**]

- **Definition Serverless**
  - **FaaS:** functions-as-a-service (event-driven, stateless input-output mapping)
  - Infrastructure for deployment and auto-scaling of APIs/functions
  - Examples: **Amazon Lambda**, **Microsoft Azure Functions**, etc



- **Example**

```
import com.amazonaws.services.lambda.runtime.Context;
import com.amazonaws.services.lambda.runtime.RequestHandler;

public class MyHandler implements RequestHandler<Tuple, MyResponse> {
    @Override
    public MyResponse handleRequest(Tuple input, Context context) {
       return expensiveStatelessComputation(input);
    }
}
```

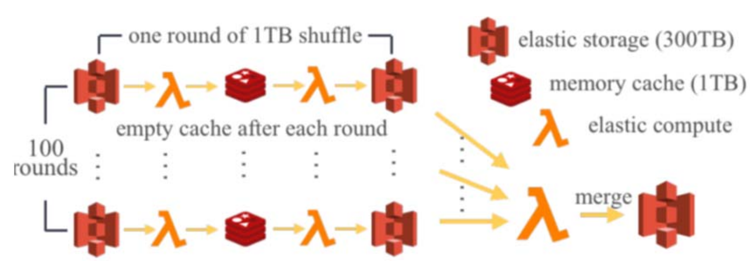
Applications

- **Embarrassingly-Parallel Use Cases**
  - Stateless image/video processing (thumbnails, encoding, rendering)
  - ML inference/scoring (e.g., object classification and detection)
  - Distributed compilation, unit testing

- **Data Analytics – CloudSort** (http://sortbenchmark.org/)

[Qifan Pu, Shivaram Venkataraman, Ion Stoica: Shuffling, Fast and Slow: Scalable Analytics on Serverless Infrastructure. **NSDI 2019**] (Locus)

  - Minimum cost for sorting 100TB
  - 500x slower on serverless compared to VMs → reason: **slow data shuffling**
  - Multi-round, hybrid shuffle (w/ same range partitioner)
    - Small, fast storage (e.g., Redis) for intermediates per round
    - Large, slow storage (S3) output
  - Final merge of runs into S3

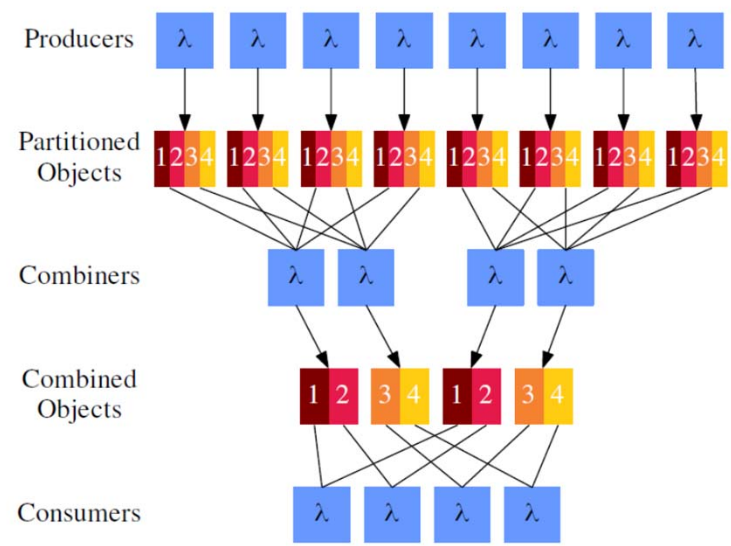


FaaS Query Processing – Starling (MIT)

- **Motivation**
  - Avoid pre-provisioning, and data loading
  - Pay per query w/ competitive performance
  - Tunable cost-performance per query

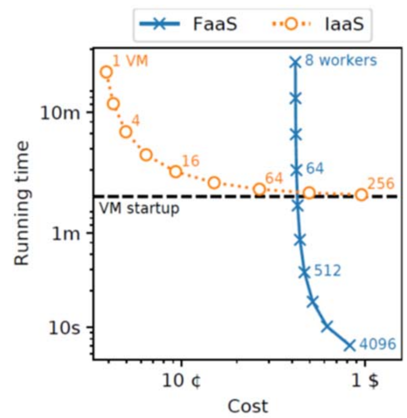
[Matthew Perron, Raul Castro Fernandez, David J. DeWitt, Samuel Madden: Starling: A Scalable Query Engine on Cloud Functions. **SIGMOD 2020**]

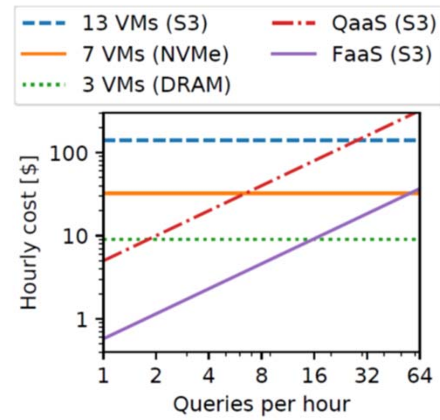
- **Starling Query Processing**
  - Coordinator compiles queries, and schedules tasks
  - Open input formats (CSV, ORC, Parquet)
  - Intermediates stored in S3
  - Shuffling: Mitigate many file problem by writing single file per task, read portions
  - Data centric query compilation
  - Task pipelining and straggler mitigation



FaaS Query Processing – Lambada (ETH)

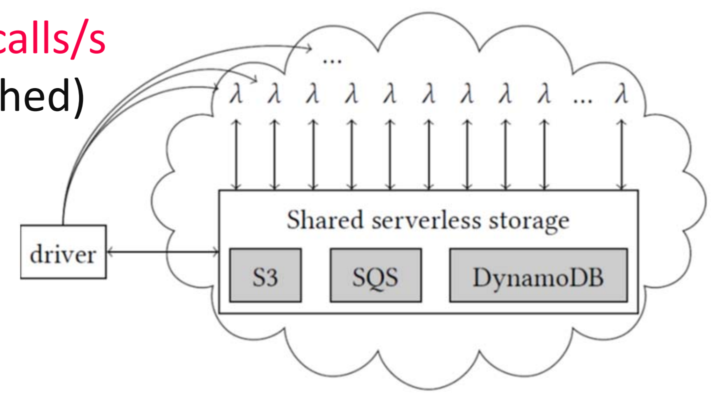
- **Potential Analysis**
  - Simulation of scan 1TB from S3 (**2min VM startup**, 4s fun startup)
  - Short startup + demand scaling
  - **Interactive analytics on cold data** (e.g., Hydrology, HE Physics)





- **Lambada Query Processing**
  - Driver on client machine w/ **batched, two-level invocation**
  - Data-parallel execution solely with serverless workers (lambda funs)
  - **Parquet scan operator** (sel/proj pushdown)
  - **Exchange operators** for join/sort/group-by (communication through shared storage)

~220 calls/s (batched)

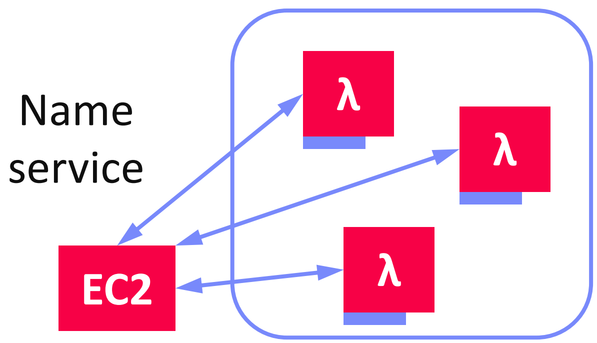


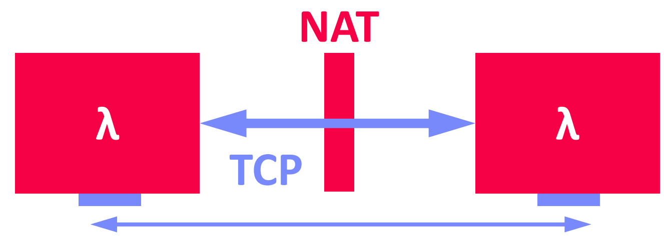
[Ingo Müller et al: Lambada: Interactive Data Analytics on Cold Data Using Serverless Cloud Infrastructure. **SIGMOD 2020**]

FaaS Query Processing – Lambada (ETH), cont.

[Michal Wawrzoniak et al: Boxer: Data Analytics on Network-enabled Serverless Platforms, **CIDR 2021**]

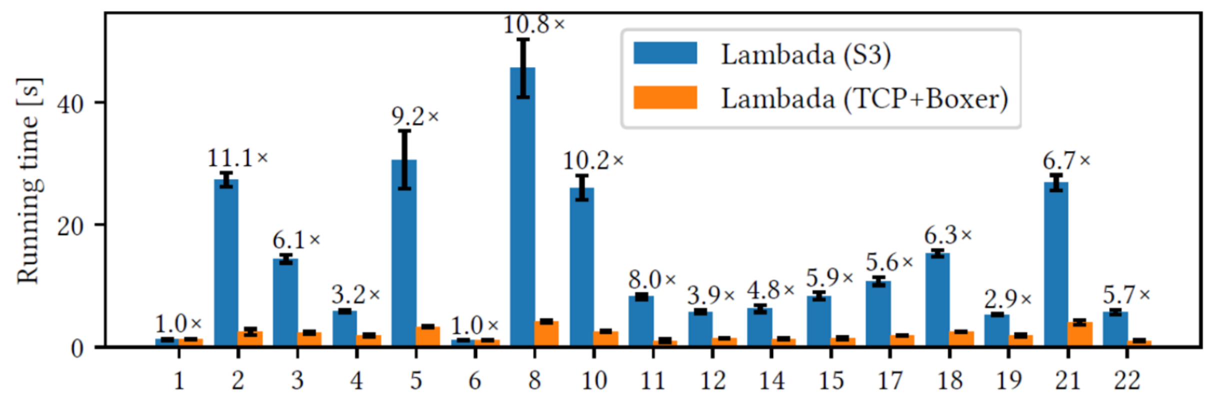
- **Function-to-function TCP Networking**
  - Problem: FaaS functions behind NAT
  - NAT Hole Punching (e.g., P2P research, exchange network addresses)
  - Setup and communication processes





Boxer lib intercepts connect(), exchanges IP info, and established normal TCP connection

- **TPC-H Performance**

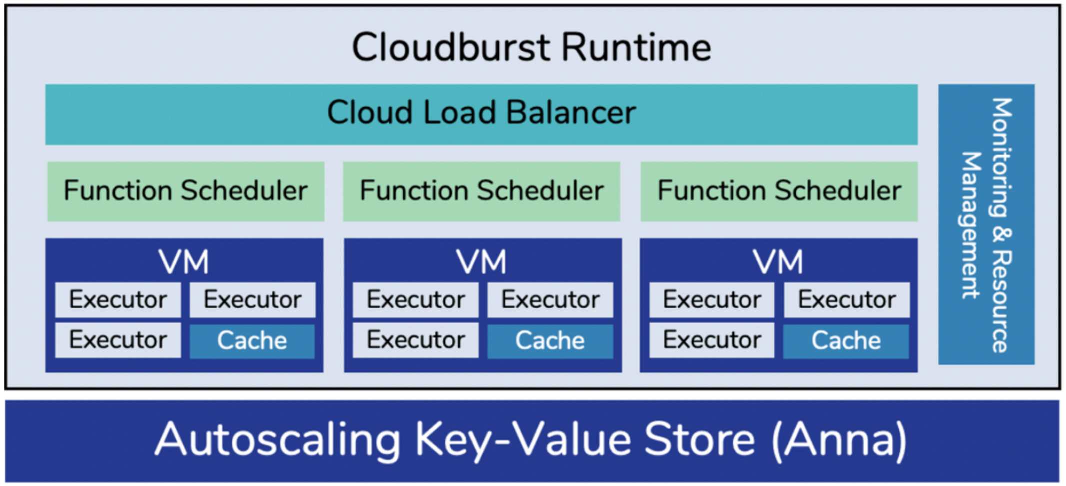


FaaS Query Processing – Cloudburst (UC Berkeley)

[Vikram Sreekanti et al: Cloudburst: Stateful Functions-as-a-Service. **PVLDB 13(11) 2020**]

- **Motivation**
  - Autoscaling serverless computing, with low-latency mutable state ➔ broader class of apps
  - State sharing and mutable caches co-located w/ functions (data locality)

- **Architecture**
  - VM orchestration via Kubernetes
  - Logical disaggregation with physical co-location
  - Functions interact w/ the cache not KV-Store
  - Anna periodically propagates key updates
  - Coordination-free consistency (via lattice data types: MapLattice)



Prototype not compatible w/ Public Cloud Lambda Functions

Summary and Q&A

- **Cloud Computing Background**
- **PaaS: SQL on Hadoop**
- **SaaS: Cloud DBs and Cloud DWHs**
- **FaaS: Serverless Database Systems**

Is FaaS/serverless the right underlying abstraction for query processing?
(general-purpose, startup time, price model, elasticity)

- **Next Lectures** (Part C)
  - **11** **Modern Concurrency Control** [Jan 20]
  - **12** **Modern Storage and HW Accelerators** [Jan 27]