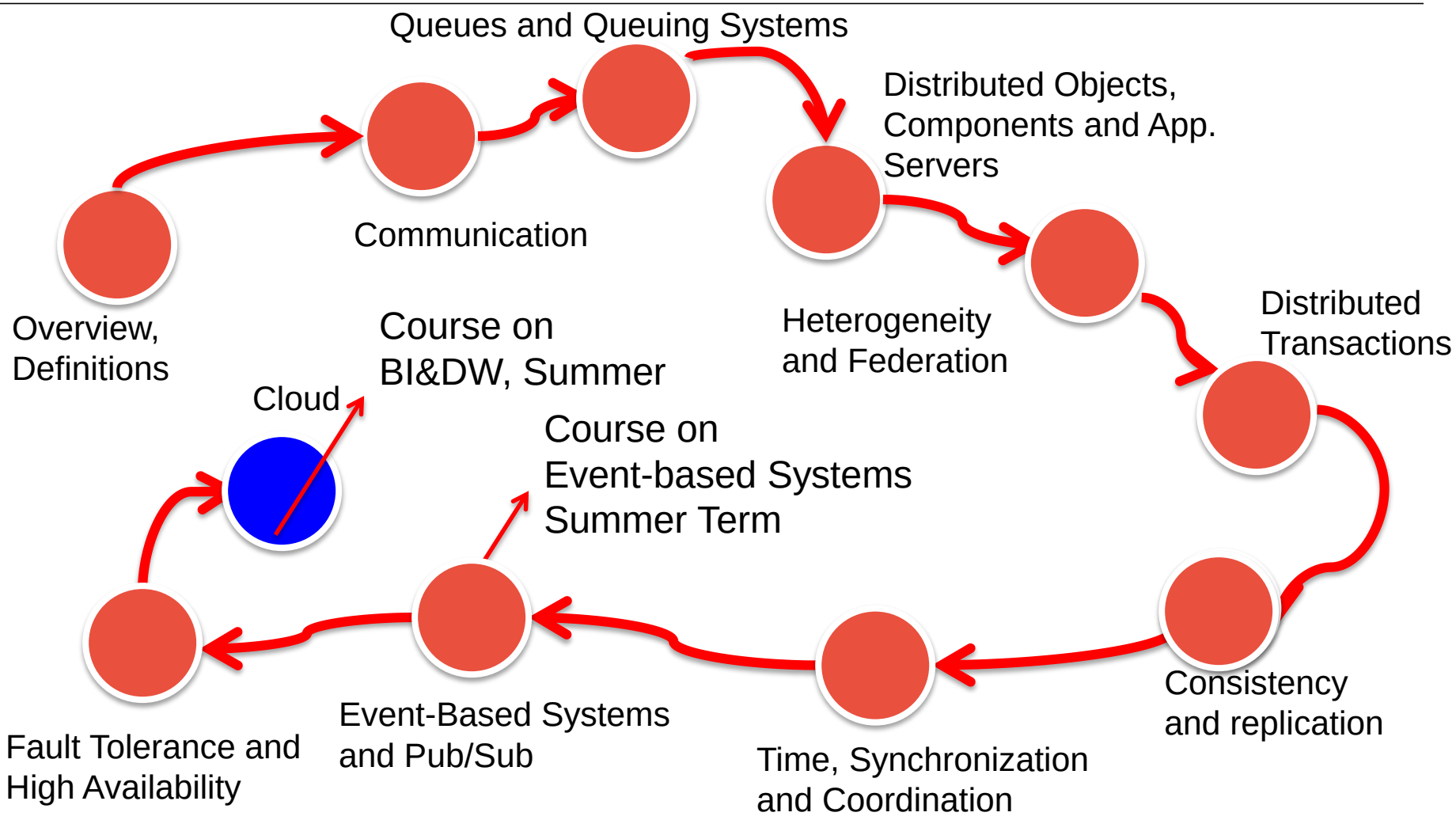
Middleware:
11. Aspects of Cloud Computing

TECHNISCHE UNIVERSITÄT DARMSTADT

A. Buchmann
Wintersemester 2013/2014

Topics


Queues and Queuing Systems
Distributed Objects, Components and App. Servers
Communication
Overview, Definitions
Course on BI&DW, Summer
Heterogeneity and Federation
Distributed Transactions
Cloud
Course on Event-based Systems Summer Term
Consistency and replication
Fault Tolerance and High Availability
Event-Based Systems and Pub/Sub
Time, Synchronization and Coordination

Topics

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Data Storage**
  - Large Files, Robust, Fault-tolerant, Availability
- **Data-Analysis** and Query Processing
  - Data models
  - Programming Languages and Programming Models
- **Massive distribution, Parallel Processing and Scalability**
- *Elasticity, Virtualization and Multi-Tenancy*

Reading for THIS Lecture

TECHNISCHE UNIVERSITÄT DARMSTADT

- The slides for the lecture are based on material from:
  - M. Tamer Özsu and Patrick Valduriez **Principles of Distributed Database Systems** (3nd Ed.). Prentice-Hall. 2011
    - Chapter 18.2
  - Cloud computing Materials. G. **Saake, V. Markl, K.U. Sattler**
  - Ghemawat, Gobioff, Leung. The Google File System. SOSP ’03
  - Chang, Dean et al. Bigtable: A Distributed Storage System for Structured Data. ACM Trans. Comput. Syst. 26, 2, June 2008
  - Dean, Ghemawat. MapReduce: simplified data processing on large clusters. CACM. ACM 51, 1 January 2008

Technical Aspects of Cloud Computing

TECHNISCHE UNIVERSITÄT DARMSTADT

- What is Cloud Computing?

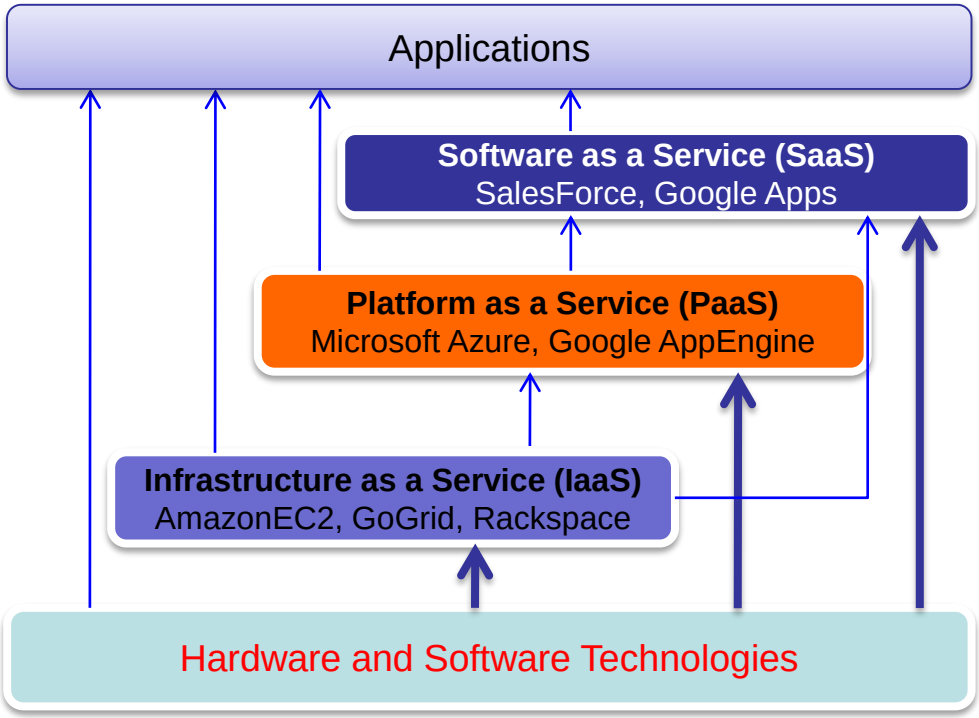
NIST:

Cloud computing is a **model** for enabling convenient, **on-demand** network access to a **shared pool of configurable computing resources** (e.g., networks, servers, storage, applications, and services) that can be **rapidly provisioned** and **released** with **minimal management effort** or **service provider** interaction.

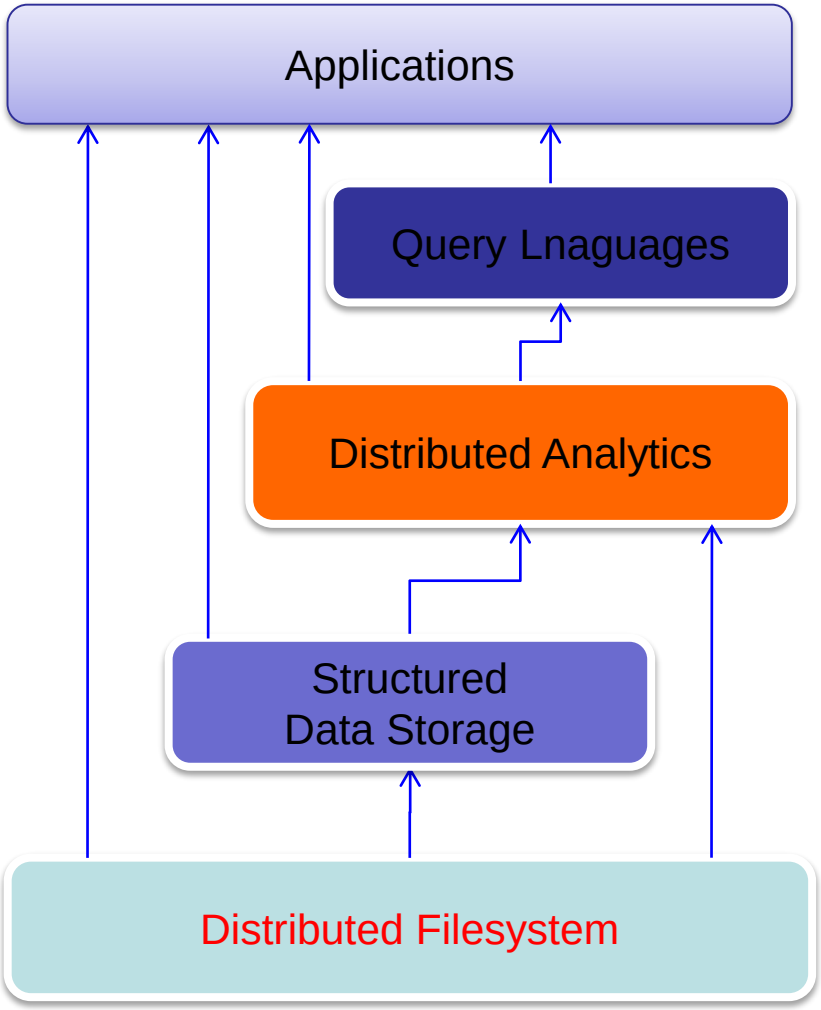
[http://www.nist.gov/itl/cloud/]

Grossmann, Gu 2009

Cloud provides on demand ressources and services over the Internet, at the scale and with the reliability of a data center



Typical Layers



High-level Data Processing Language
HiveQL, JaQL, Pig

Complex Data Analysis Operations
(Joins a la SQL, Grouping)
MapReduce

Simple and Flexible Data Model,
Basic Access Operations, Lookup
BigTable*, Hbase Cassandra SimpleDB*

Fast Data Access, Fault Tolerance,
Highly Available, Scalable
GoogleFS*, HDFS, S3, Dynamo*

Distributed File System

TECHNISCHE UNIVERSITÄT DARMSTADT

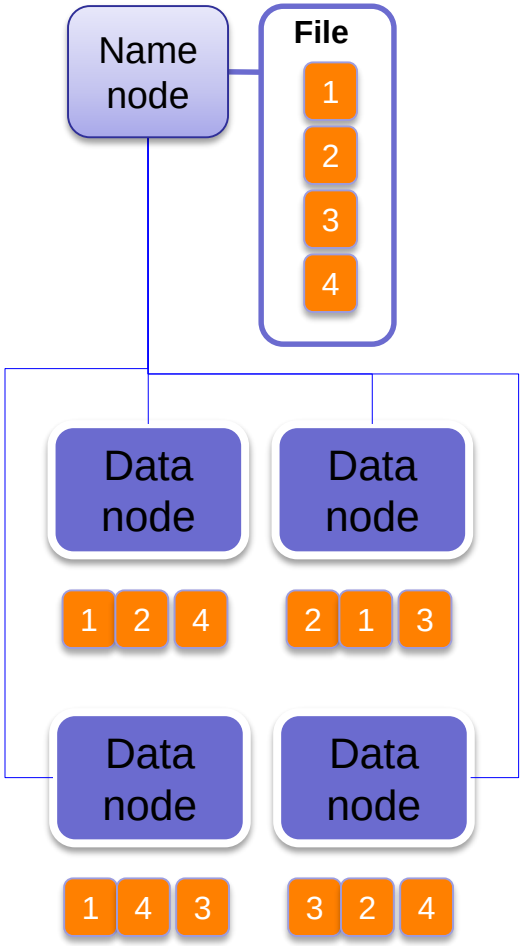
Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Distributed File System

TECHNISCHE UNIVERSITÄT DARMSTADT

- File System
  - Distributed over many nodes (Data Nodes)
  - Integrated Namespace for the whole cluster
  - Metadata managed centrally on a naming node
  - Access Model: Write-Once-Read-Many
- Files broken into Chunks
  - 128MB or more
  - Each block replicated on different nodes
- No modification operations except ‚Append‘
  - Insert, Delete, Update → Append
- Fault Tolerance and High Availability through Replication



Ghemawat, Gobioff, Leung. The Google File System. SOSP ‘03

GFS Priciples and Assumptions

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Fault Tolerance**
  - Failures are common (cheap standard components).
  - System must handle many simultaneously failing components
  - System must continuously monitor
- **Large Files**
  - Large (multi-GB) files are the norm (web documents w. many app. Objects)
  - Small files are supported but not optimized for
- **Workload**
  - Large Seq. Accesses (>1MB). Few Random Accesses.
  - Small Random accesses batched.
  - Large sequential writes, typically appends, virtually no random writes.
- **Append is the general case**
  - no mutable data, update only via append
- **Robust Sequential Accesses**
  - Bandwidth more important than Latency for the targeted applications
  - No Data Caches! Metadata cached.

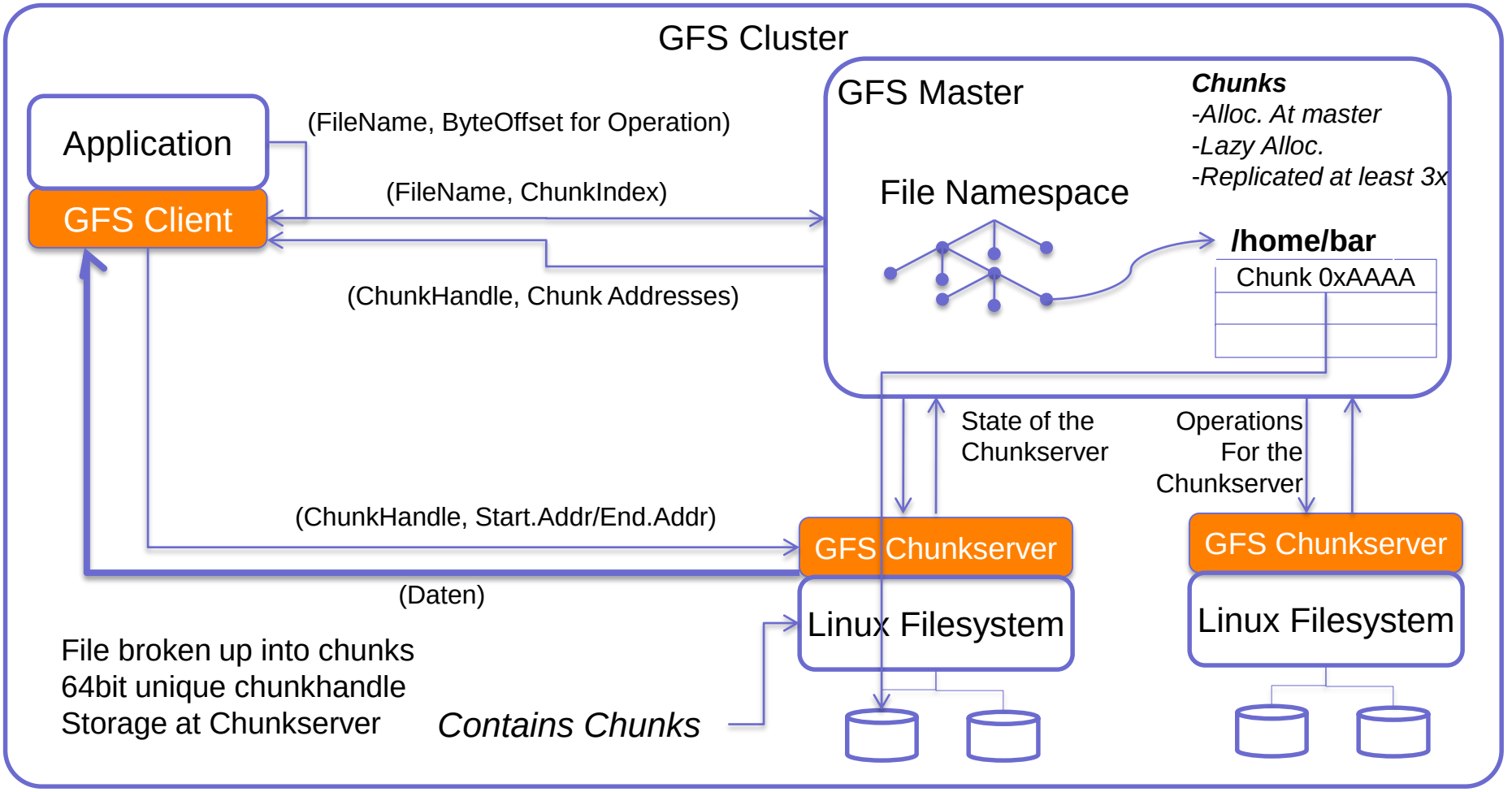
Ghemawat, Gobioff, Leung. The Google File System. SOSP '03

GFS Interface

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Operations**: *create, delete, open, close, read, write, append, snapshot*
  - Snapshot creates fast copy of (part of) a file
  - Append is atomic
  - Many clients can append simultaneously to the same file!

GFS Architecture

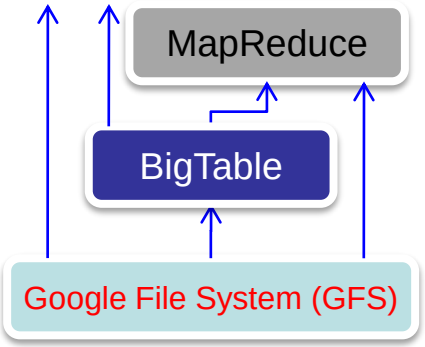


Ghemawat, Gobioff, Leung. The Google File System. SOSP '03

Read and Transfer Data

TECHNISCHE UNIVERSITÄT DARMSTADT

- Data Scan on remote nodes → expensive
  - 50MB/s remote versus 150MB/s locally
  - What is more efficient: ‚data-to-code' or ‚code-to-data'?
  - Individual nodes perform computations

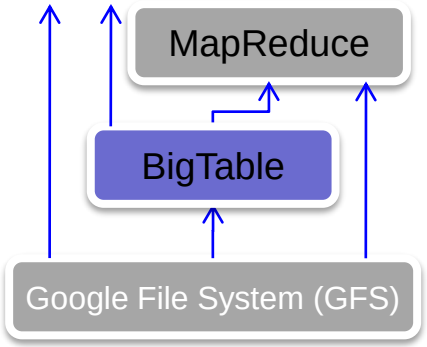


- Programs must be parallel scalable and fault tolerant
  - Development is cumbersome
  - Necessary: Suitable Programming model and frameworks

Big Table

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

BigTable



- Distributed Storage system for structured data
  - Shared Nothing Cluster
  - Similar to a parallel database
  - Fault tolerant and persistent
  - Scalable
    - 1000s of Servers
    - TB main memory data, PB on disk
    - Millions of IOPS, efficient Scan-Operations
  - No Transactions

- Simple data model
  - Aligned with the Relational model, lots of differences
  - No relational interface
  - Data are strings
  - Clients control the data locality through schemata
  - Data lie on disk or in main memory, stored as SSTables (a key-value store)

Chang, Dean et al. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst. 26, 2, June 2008*

BigTable - Datamodel

TECHNISCHE UNIVERSITÄT DARMSTADT

- BigTable **Cluster** manages several tables
- **Table**: sparse, distributed, persistent, multi-dimensional, sorted mapping table, comprising rows and columns
  - (**row**:string, **column**:string, **timestamp**:int64) → **value**:string
- **Rows**: Unlimited number, grouped together form a partition
  - Keys (up to 64KB, usually 10-100 Bytes): Lexicographically sorted → facilitates distribution!
  - Tablets: a group of rows with consecutive keys → Unit of distribution
    - Read operations on a small set of rows → very efficient!
  - Operations on a single row are atomic (no transactions across rows)

Chang, Dean et al. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst. 26, 2, June 2008*

BigTable – Data Model

TECHNISCHE UNIVERSITÄT DARMSTADT

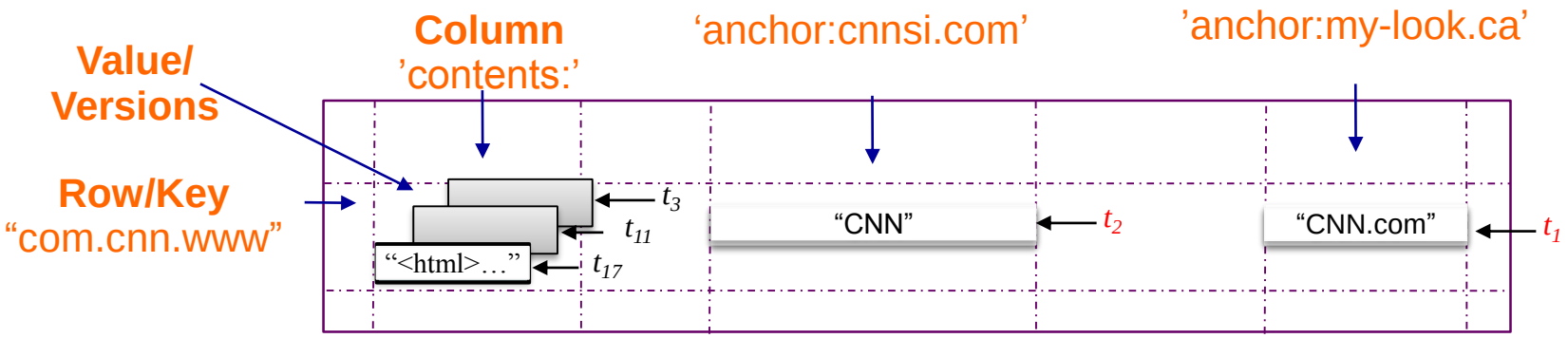
- **Columns**: Grouped in Column Families (also used for Access Control)
  - All data have usually the same data type
  - Column families explicitly produced → **family:qualifier** (e.g. language)
    - Unlimited number of columns (but expected to be max. in the hundreds)
    - Few Column Families→ Metada synchronisation
    - Granularity for: Access control, Placement (disk, main memory)

- **Tiemstamp**: cell content can be versioned
  - Old versions can be automatically garbage collected (garbage collection, N ts.)

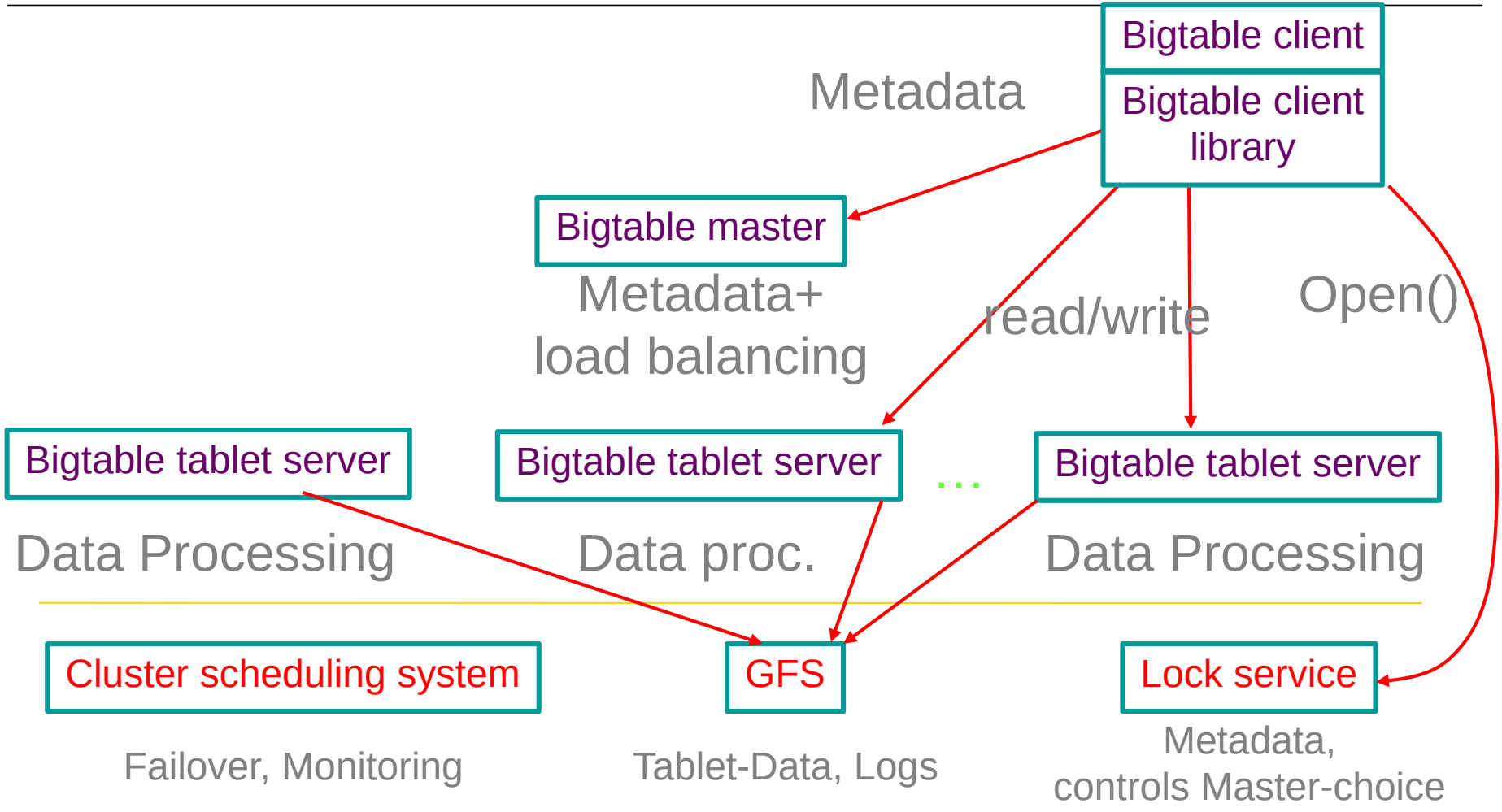
Chang, Dean et al. Bigtable: A Distributed Storage System for Structured Data. *ACM Trans. Comput. Syst. 26, 2, June 2008*

Big Table example

- A slice of an example table that stores Web pages. The row name is a reversed URL.
- The contents column family contains the page contents, and the anchor column family contains the text of any anchors that reference the page.
- CNN's home page is referenced by both the Sports Illustrated and the MY-look home pages (so the row contains columns named anchor:cnnsi.com and anchor:my.look.ca.)
- Each anchor cell has one version; the contents column has three versions



System Structure



Chubby

TECHNISCHE UNIVERSITÄT DARMSTADT

- {lock/file/name} service
- Coarse-grained locks, can store small amount of data in a lock
- 5 replicas, need a majority vote to be active
- Based on leases, if lease expires all locks are lost
- Also an OSDI ’06 Paper

Map/Reduce

TECHNISCHE UNIVERSITÄT DARMSTADT

Created with wordle.net based on:
P. Bernstein. Middleware. CACM, Feb. 1996

DVS

Motivation for Map Reduce

TECHNISCHE UNIVERSITÄT DARMSTADT

"We realized that most of our computations involved applying a map operation to each logical record' in our input in order to compute a set of intermediate key/value pairs, and then applying a reduce operation to all the values that shared the same key in order to combine the derived data appropriately."

From: Dean, Ghemawat. MapReduce: simplified data processing on large clusters. CACM. ACM 51, 1 January 2008

"The issues of how to parallelize the computation, distribute the data, and handle failures conspire to obscure the original simple computation with large amounts of complex code to deal with these issues."

ibid

Map Reduce Example (Pseudocode)

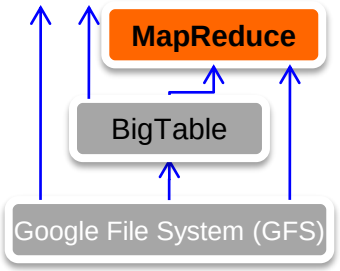
TECHNISCHE UNIVERSITÄT DARMSTADT

```
map(String key, String value):
  // key: document name
  // value: document contents
  for each word w in value:
        EmitIntermediate(w, “1”);

reduce(String key, Iterator values):
  // key: a word
  // values: a list of counts
  int result = 0;
  for each v in values:
        result += ParseInt(v);
  Emit(AsString(result));
```

What is Map/Reduce?

TECHNISCHE UNIVERSITÄT DARMSTADT



- Concepts from the field of functional programming
  - Good fit for parallelism and distribution
  - Offer good abstractions to application developers
  - Computations as expressed
  - Global data avoided: simple and safe
  - Output value of a function depends only on input parameters

- Map/Reduce – simple parallel programs for data analysis
  - Map and Reduce – higher-order functions
    - Take user defined functions as arguments
    - Return functions
  - The user implements the two functions Map/Reduce – realizes a unit of work

Dean, Ghemawat. MapReduce: simplified data processing on large clusters. CACM. ACM 51, 1 January 2008

Example MapReduce

TECHNISCHE UNIVERSITÄT DARMSTADT

- Problem: Count the number of words in a file
  - Parallel!
  - File1: abc, abc, dd ee ff abc
  - File2: das ee ff qwe
- output: abc (3), ee(2), ee(2), qwe(1), dd(1), das(1)

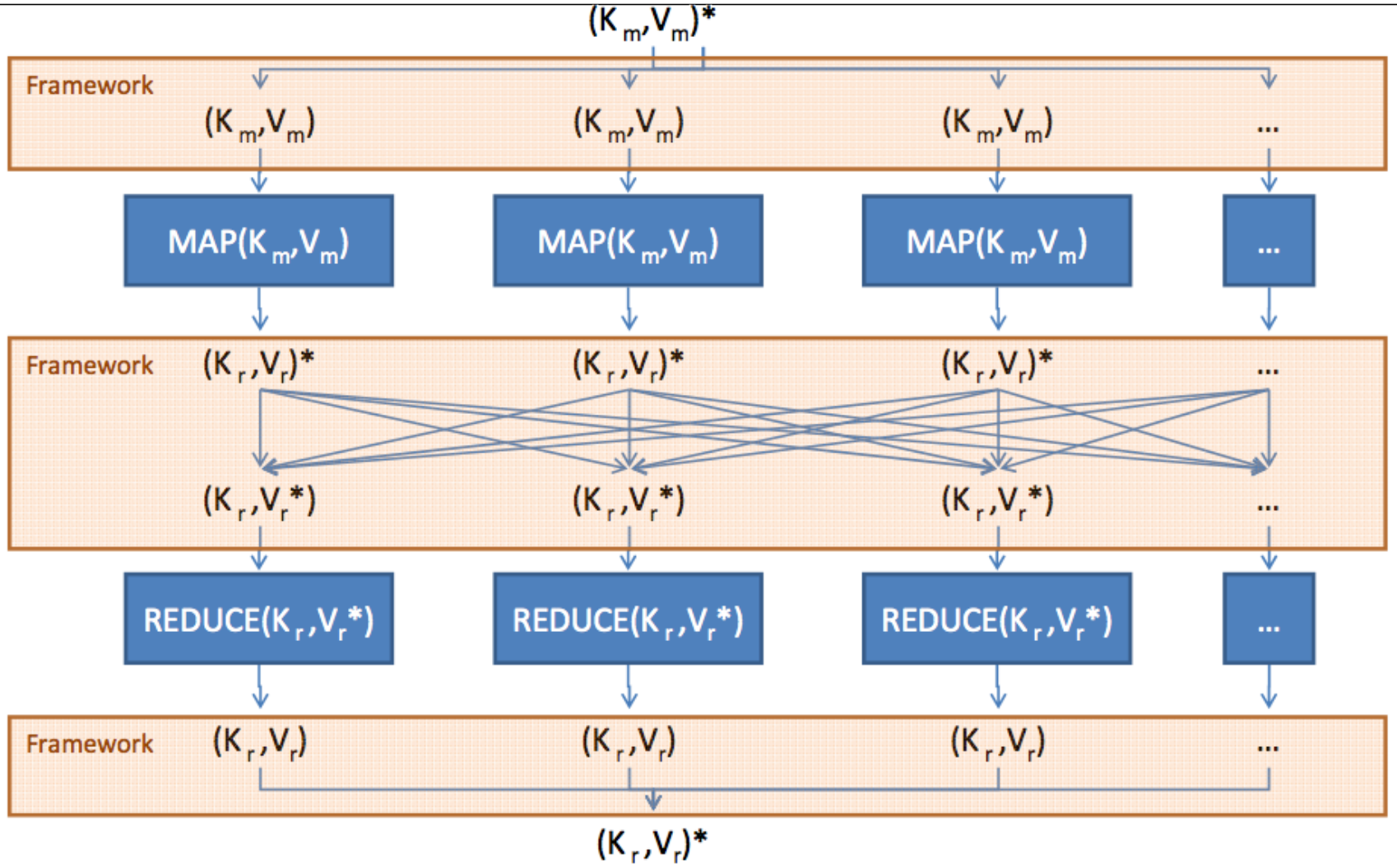
- Solution: Map-Reduce Unit of Work

```
map( filename, line ) {
        foreach ( word in line )
                emit( word, 1 );
}
```

```
reduce( word, numbers ) {
        int sum = 0;
        foreach ( value in numbers ) {
                sum += value;
        }
        emit( word, sum );
}
```

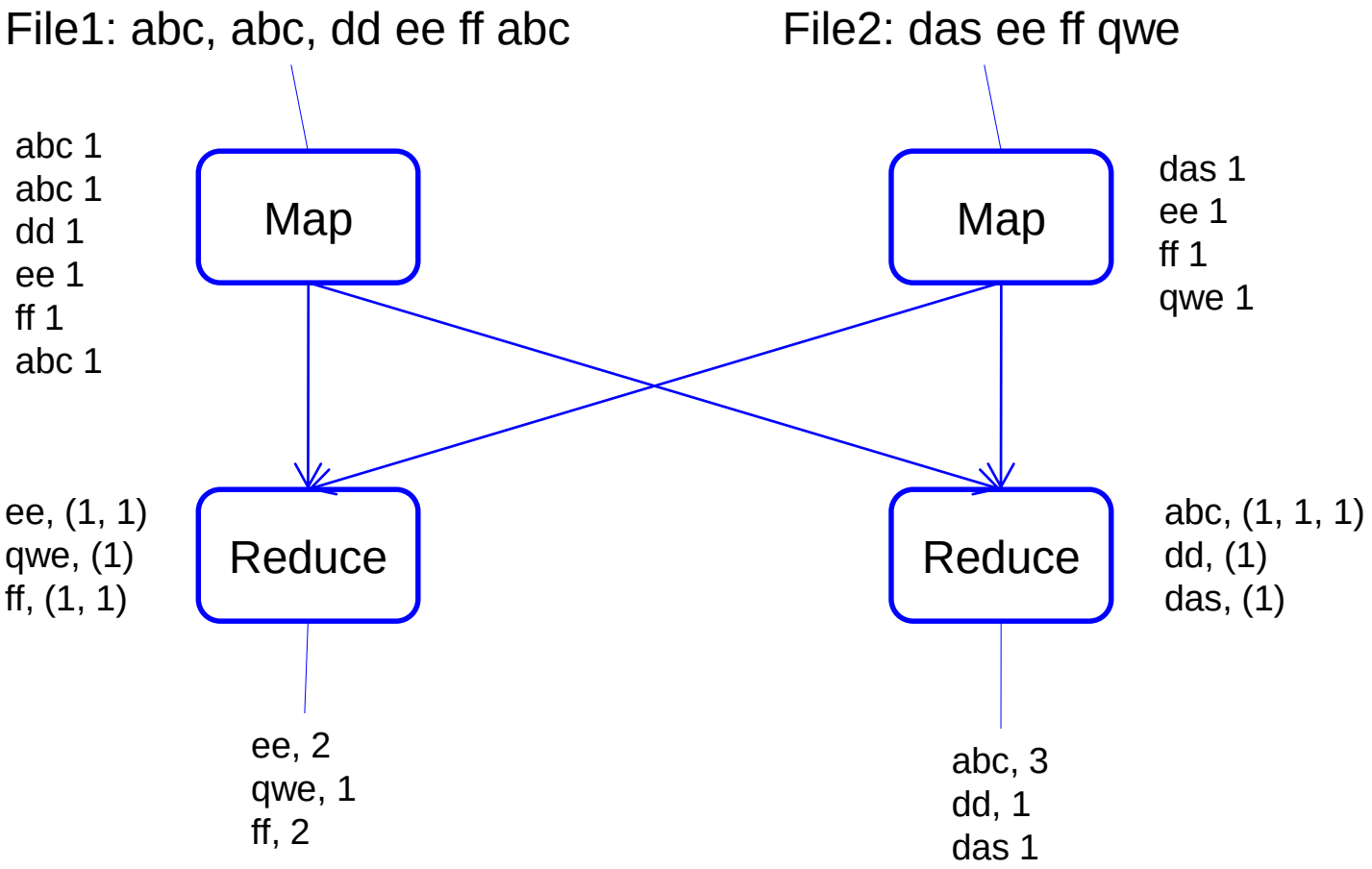

Dean, Ghemawat. MapReduce: simplified data processing on large clusters. CACM. ACM 51, 1 January 2008

Data Flow



Dean, Ghemawat. MapReduce: simplified data processing on large clusters. CACM. ACM 51, 1 January 2008

Example MapReduce

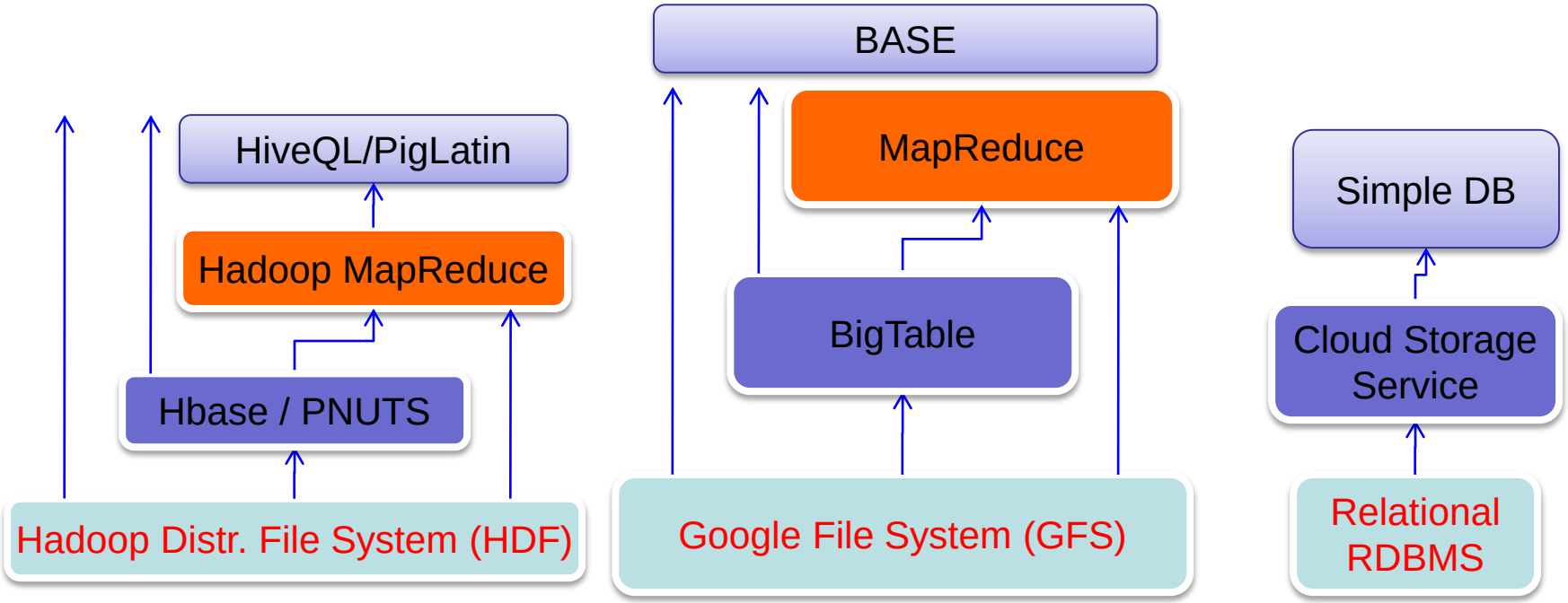


Dean, Ghemawat. MapReduce: simplified data processing on large clusters. CACM. ACM 51, 1 January 2008

Some Frameworks/Stacks

TECHNISCHE UNIVERSITÄT DARMSTADT

- Many requirements, Many Alternatives →The Same goal!
  - Yahoo and Google have simlimar stacks however with significant differences Amazon, Microsoft, Salesforce, Facebook ... Significantly different



DVS

Summary

TECHNISCHE UNIVERSITÄT DARMSTADT

- **Data Storage**
  - Large Files, Robust, Fault-tolerant, Availability
- **Data-Analysis** and Query Processing
  - Data models
  - Programming Languages and Programming Models
- **Massive distribution, Parallel Processing and Scalability**
- *Elasticity, Virtualization and Multi-Tenancy*

Middleware:
Course Summary

A. Buchmann
Wintersemester 2013/2014

TECHNISCHE UNIVERSITÄT DARMSTADT

DVS

Thank You!

TECHNISCHE UNIVERSITÄT DARMSTADT

Questions?

DVS