Distributed Data Management
Lecture Summary
Thorsten Papenbrock
F-2.04, Campus II
Hasso Plattner Institut
Overview

Topics DDM

1. Introduction
2. Foundations
3. Encoding & Communication
4. Akka Actor Programming
5. Data Models & Query Languages
6. Storage & Retrieval
7. Replication
8. Partitioning
9. Distributed Systems
10. Consistency & Consensus
11. Transactions
12. Batch Processing
13. Spark Batch Processing
14. Stream Processing
15. Distributed DBMS
16. Distributed Query Optimization
Some Important Topics

1 Introduction

Distributed Data Management

Lecture Summary

ThorstenPapenbrock

Slide 3

Even distributed parallelization cannot work around Amdal’s law!

\[
\text{Speedup}(s) = \frac{1}{(1 - p) + \frac{p}{s}}
\]
Some Important Topics

2 Foundations

Reliability

- = fault-tolerance:

| fault/defect | may cause | error | may not cause | failure |

ACID & CAP & BASE

- Availability
- Consistency
- Partition Tolerance

Task-Parallelism vs. Data-Parallelism

Multi-Threading vs. Distributed Computing

Distributed Data Management
Lecture Summary

Thorsten Papenbrock
Slide 4
Some Important Topics
3 Encoding & Communication

Java Serialization

```
class TestSerial implements Serializable {
    public byte version = 100;
    public byte count = 0;
}
```

AC ED 00 05 73 72 00 0A 53 65
72 69 61 6C 54 65 73 74 A0 0C
34 00 FE B1 DD F9 02 00 02 42
00 05 63 6F 75 6E 74 42 00 07
76 65 72 73 69 6F 6E 78 70 00
64

Dataflow Models

RPCs

Thorsten Papenbrock
Slide 5
Some Important Topics

4 Akka

Actor Model

```java
public class Worker extends AbstractActor {
    @Override
    public Receive createReceive() {
        return receiveBuilder()
            .match(String.class, s -> this.sender().tell("Hello!", this.self()))
            .match(Integer.class, i -> this.sender().tell(i * i, this.self()))
            .match(Double.class, d -> this.sender().tell(d > 0 ? d : 0, this.self()))
            .match(MyMessage.class, s -> this.sender().tell(new YourMessage(), this.self()))
            .matchAny(object -> System.out.println("Could not understand received message"))
            .build();
    }
}
```
Some Important Topics
5 Data Models & Query Languages

### Relational
- Row-Based
- Column-Based

### Non-Relational
- Key-Value
- Column-Family
- Document
- Graph

### Query Languages

**SPARQL**
```
SELECT ?locationName
WHERE {
    ?hpi :name "HPI gGmbH" .
    ?hpi :location ?locationName .
}
```

**MongoDB API**
```
db.people.find(
    {
        $or: [
            { status: "A" },
            { age: 50 }
        ]
    }
)
```

**SQL**
```
SELECT *
FROM PC PC1, PC PC2
WHERE PC1.speed = PC2.speed
AND PC1.ram = PC2.ram
AND PC1.model < PC2.model;
```

**Redis**
```
SET hello "hello world"
GET hello
→ "hello world"
```

**Cipher**
```
MATCH (me {name:"T. Papenbrock "})
MATCH (expert)-[:KNOWS]->(db:Database {name:"Neo4j"})
MATCH path = shortestPath( (me)-[:FRIEND*..5]-(expert) )
RETURN db, expert, path
```

**CQL**
```
SELECT *
FROM myTable
WHERE myField > 5000
AND myField < 100000
ALLOW FILTERING;
```
Some Important Topics
6 Storage & Retrieval

LSM-Trees with B-trees and SSTables

Segmentation

 Writes

Reads

Compacted

Current

B-Tree

SSTable
Some Important Topics

7 Replication

**Single-Leader Replication**

- Leader
- Follower

**Multi-Leader Replication**

- Leader
- Follower

**Leaderless Replication**

- Replica

---

**Quorum**

- quorum \((w,r)\)

**Quorum Consistency**

- \(w + r > n\)

---

**Gossip & Merkle Trees**

- \(H_{root} = h(H_{13} + H_{34})\)
- \(H_{13} = h(H_1 + H_2)\)
- \(H_{34} = h(H_3 + H_4)\)
- \(H_1 = h(T_1)\)
- \(H_2 = h(T_2)\)
- \(H_3 = h(T_3)\)
- \(H_4 = h(T_4)\)

---

UPDATE users SET picture_url='me-new.jpg'
WHERE user_id=1234
Some Important Topics
8 Partitioning

Range Partitioning by Hash of Key

Consistent Hashing

Rebalancing Partitions

Partition-Lookup

- the knowledge of which partition is assigned to which node
Some Important Topics
9 Distributed Systems

The $\varphi$ accrual failure detector

The network time protocol (NTP)

Leases

Distributed Data Management
Lecture Summary
Thorsten Papenbrock
Slide 11
Some Important Topics
10 Consistency & Consensus

Linearizability ↔ Total Order Broadcast ↔ Consensus

Leader Election

Ordering with Lamport timestamps

Blockchain

Thorsten Papenbrock Slide 12
Some Important Topics

11 Transactions

Causal Ordering

Two-Phase Commit (2PC)

- Obtain unique transaction ID
- Write data
- Prepare
- Commit
- If coordinator crashes: recover and continue sending commits/aborts
- If node crashes: recover (and query coordinator)
- Get ready to commit (append all writes to log on disk)
  - crashes, power failures, exhausted memory, … are no excuses later on

Snapshot Isolation via MVCC

- Transaction bid = 12
  - Account 1: created by = 3, deleted by = nil, id = 1, balance = 500
  - Account 2: created by = 5, deleted by = nil, id = 2, balance = 500
- Transaction bid = 13
  - Account 1: created by = 3, deleted by = nil, id = 1, balance = 500
  - Account 2: created by = 5, deleted by = nil, id = 2, balance = 500

Distributed Data Management

Lecture Summary

Thorsten Papenbrock

Slide 13
Some Important Topics

12 Batch Processing

HDFS

Transformation Pipelines

MapReduce

Distributed Data Management

Lecture Summary

Thorsten Papenbrock

Slide 14
Some Important Topics

13 Spark

---

**Spark Program (= Spark Job)**

(Java, Scala, Python, SQL, R)

---

**Transformation Pipeline**

- **Map**
- **Filter**
- **Distinct**
- **Reduce**
- **Filter**
- **Map**
- **Join**
- **Collect**

---

**val** sum = data.as[String]
  .filter(value => value == null)
  .flatMap(value => value.split("\s+"))
  .map(value => (value, 1))
  .reduceByKey(_ + _)
  .collect()

**val** result = flightData
  .groupBy("DESTINATION")
  .sum("FLIGHTS")
  .sort(desc("sum(FLIGHTS)"))
  .select(
    col("DESTINATION"),
    col("sum(FLIGHTS)").as("sum"))
  .collect()
Some Important Topics

14 Stream Processing

Data Streams

Transformation pipeline

Event Time vs. Processing Time

Windowing (Tumbling, Hopping, Sliding, Session)

CQL

SELECT count(*)
FROM Requests R
[PARTITION BY R.client_id
ROWS 10 PRECEDING
WHERE R.domain = 'stanford.edu']
WHERE R.url LIKE 'http://cs.stanford.edu/%%'

val env = StreamExecutionEnvironment.getExecutionEnvironment
val text = env.socketTextStream("localhost", 4242, 
\n")
val windowCounts = text
.flatMap { w => w.split("\s") }
.map { w => WordWithCount(w, 1) }
.keyBy("word")
.timeWindow(Time.seconds(5), Time.seconds(1))
.sum("count")
windowCounts.print().setParallelism(1)
env.execute("Socket Window WordCount")

case class WordWithCount(word: String, count: Long)
Some Important Topics
15 Distributed DBMS

Global as View

Local as View

Column Store Compression (see Parquet file format)

Data Cubes

<table>
<thead>
<tr>
<th>date_key</th>
<th>product_sk</th>
<th>store_sk</th>
<th>promotion_sk</th>
<th>customer_sk</th>
<th>quantity</th>
<th>net_price</th>
<th>discount_price</th>
</tr>
</thead>
<tbody>
<tr>
<td>140101</td>
<td>69</td>
<td>4</td>
<td>NULL</td>
<td>NULL</td>
<td>1</td>
<td>13.99</td>
<td>13.99</td>
</tr>
<tr>
<td>140102</td>
<td>69</td>
<td>5</td>
<td>19</td>
<td>NULL</td>
<td>3</td>
<td>14.99</td>
<td>9.99</td>
</tr>
<tr>
<td>140102</td>
<td>69</td>
<td>5</td>
<td>NULL</td>
<td>191</td>
<td>1</td>
<td>14.99</td>
<td>14.99</td>
</tr>
<tr>
<td>140102</td>
<td>74</td>
<td>3</td>
<td>23</td>
<td>202</td>
<td>5</td>
<td>0.99</td>
<td>0.89</td>
</tr>
<tr>
<td>140103</td>
<td>31</td>
<td>2</td>
<td>NULL</td>
<td>NULL</td>
<td>1</td>
<td>2.49</td>
<td>2.49</td>
</tr>
<tr>
<td>140103</td>
<td>31</td>
<td>3</td>
<td>NULL</td>
<td>NULL</td>
<td>3</td>
<td>14.99</td>
<td>9.99</td>
</tr>
<tr>
<td>140103</td>
<td>31</td>
<td>3</td>
<td>21</td>
<td>123</td>
<td>1</td>
<td>49.99</td>
<td>39.99</td>
</tr>
<tr>
<td>140103</td>
<td>31</td>
<td>8</td>
<td>NULL</td>
<td>NULL</td>
<td>1</td>
<td>0.99</td>
<td>0.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>date_key</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>140101</td>
<td>149.60</td>
</tr>
<tr>
<td>140102</td>
<td>132.18</td>
</tr>
<tr>
<td>140103</td>
<td>196.75</td>
</tr>
<tr>
<td>140104</td>
<td>178.36</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>total</td>
<td>14967.09</td>
</tr>
</tbody>
</table>

Bitmap Encoding
Run-length Encoding

Thorsten Papenbrock
Slide 17
Some Important Topics

16 Distributed Query Optimization

Distributed Joins

Distributed Query Execution

Distributed Join & Full Reducer

Distributed Data Management
Lecture Summary

ThorstenPapenbrock
Slide 18
What to remove from the exam menu?
Overview

Topics DDM++

17. Services and Containerization
18. Cloud-based Data Systems
19. Further Details
20. Distributed Algorithms
21. Mining Data Streams
Topics DDM++
17 Services and Containerization

**Akka Cluster** (Recap)

- Connects ActorSystem nodes in a cluster into one distributed system
- Has no control over ...
  - resource allocation
    ActorSystems use whatever JVM resources they are started with.
  - node scaling
    ActorSystems are automatically tied together but they are started from the outside world.
  - resource isolation
    ActorSystems on the same host may compete for resources; all actors in one ActorSystem share the same resources.
Batch & Stream Processing Frameworks (Recap)

- Connect nodes in a cluster into one distributed system
- Perform cluster-wide resource management
- Restrict the programming to...
  - non-interactive but data-driven applications
    Transformation pipelines do not wait for user input or have observable side effects for users.
  - non-branching data analytics or data transformation applications
    Transformation pipelines do not support complex, branching application logic.
  - non-dynamic step-by-step applications
    Transformation pipelines are static sequences of standard operations.
Kubernetes

- Connects nodes in a cluster into one distributed system
- Performs cluster-wide resource management
- Restricts the programming only slightly

“Kubernetes (k8s) is an open-source system for automating deployment, scaling, and management of containerized applications.”

https://kubernetes.io
Kubernetes

- Can be thought of as
  a) a container platform.
  b) a microservices platform.
  c) a portable cloud platform.
Container (Docker)

- share the infrastructure of their host
- are immutable: data is stored in outside volumes
- are created from container images like objects from classes
  - faster, smaller, and much more light-weight than VMs
Container
- an application written in any programming language
- implements and encapsulates some functionality
- brings its own dependencies
Kubernetes

Pod
- a group of containers tied to some pool of resources
- the smallest scheduling unit in Kubernetes
- isolated from other pods
Service

- a set of pods that work together to achieve a greater task
- i.e. the orchestration of some container functions into one service endpoint
- public elements that can be looked-up in the cluster

Kubernetes

- a set of pods that work together to achieve a greater task
- i.e. the orchestration of some container functions into one service endpoint
- public elements that can be looked-up in the cluster
**Volumes**
- objects describing persistent storage
- can be shared by the containers of one pod
17 Services and Containerization

**Kubernetes**

**API Server**
- REST interface for cluster configuration (workloads and containers)

**Controller Manager**
- creates/deletes Pods w.r.t. some target configuration

**Scheduler**
- dynamic Pod scheduling on the available cluster nodes based on resource-requirements and -availability

**etcd**
- service discovery and cluster management (see ZooKeeper)

**Kubelet**
- manages and monitors all Pods on one cluster node
Kubernetes vs. Akka – Similarities

- Both use many same programming patterns (scheduler, router, master-worker, proxies, singletons, ...)
- Both can implement batch- and stream-processing pipelines (map, reduce, join, filter ... transformations as actors/Pods)
- Both provide means for dynamic scaling (creating and deleting actors/Pods based on current load)
- Both support branching logic (actors/containers decide freely: if A do this; if B do that)
- Both provide isolation for state and computation (private data in actors/containers and private resources in ActorSystems/Pods)
Kubernetes vs. Akka – Differences

- Akka is more a **programming framework** while Kubernetes is an **orchestration framework** for programs (programming vs. configuration)

- Akka:
  - light-weight, bound to the JVM
  - difficult resource management
  - fully asynchronous messaging

- Kubernetes:
  - heavy-weight, code-agnostic due to containerization
  - powerful resource management
  - synchronous service calls
Topics DDM++
17 Services and Containerization

Akka in Kubernetes

Kubernetes Master

Container
ActorSystem

Pod

Kubernetes Node

Container
ActorSystem

Pod

Kubernetes Node

Container
ActorSystem

Pod

Kubernetes Node

Akka Cluster
Kubernetes further reading

- Official website and documentation
  https://kubernetes.io
- Wikipedia
  https://en.wikipedia.org/wiki/Kubernetes
- Book
  Designing Distributed Systems
Cloud-based Data Systems

- Physical storage servers
  - Partitioning: Each server persists some partitions of the data.
  - Replication: Partitions are replicated to several servers.
  - Dynamic: The number of storage servers may dynamically adjust to the amount of data.

- Virtual compute servers
  - Perform computations on the data (join, filter, sort, ...)
  - Created on-demand and possibly close to the data
  - Dynamic: The number of compute servers may dynamically adjust to the query load of the system.
Cloud-based Data Systems

- **Challenges**
  - Computation and data co-placement
  - Multi-tenancy data in one data system

- **Examples**
  - Amazon S3
  - Oracle Cloud Storage
  - Microsoft Azure Storage
  - Openstack Swift
  - EMC Atmos
  - EMC ECS
  - Hitachi Content Platform
19 Further Details on Distributed Systems

- Modelle verteilter Berechnungen
- Raum-Zeit Diagrammen
- Virtuelle Zeit; logische Uhren und Kausalität
- Wellenalgorithmen
- Verteilte und parallele Graphtraversierung
- Berechnung konsistenter Schnappschüsse
- Election und Symmetriebrechung
- Verteilte Terminierung
- Garbage-Collection in verteilten Systemen
- Beobachten verteilter Systeme
- Berechnung globaler Prädikate

https://vs.inf.ethz.ch/edu/WS0405/VA

https://www4.cs.fau.de/Lehre/WS03/V_VA/Skript
### 20 Distributed Algorithms

- **Sorting**
  (e.g. distributed merge sort)

- **Clustering**
  (e.g. distributed k-means)

- **Graph Traversal**
  (e.g. Bulk Synchronous Parallel model)

- **Machine Learning**
  (e.g. ML in Spark and Flink)

- **Data Mining**
  (e.g. distributed page rank)
Sampling
(e.g. representative sampling window)

Filtering
(e.g. Bloomfilter)

Counting
(e.g. HyperLogLog)

Aggregation
(e.g. windowing)

Popular elements search
(e.g. decaying windows)
Overview

Next Semester

Seminar:

**Sustainable Machine Learning on Edge Device Clusters**
- Data Preparation
- Data Cleaning
- Data Profiling
- Model Training

- On three clusters: PI & computer & server

Open positions:

**Student Assistant**
- DDM 2020 Tutor
- Project Metanome
- Project <?>
Welcome to the evaluation platform!

HPI users
Log in using your usual HPI credentials.

Username
Password
Login

External and D-School users
Here you can request a one-time login URL. We will send it to your email address.

Email address
Request login URL 
Help

https://evaluierung.hpi.uni-potsdam.de/