Einführung in Spark

Was ist Spark?

Apache Spark

- ist eine der meistgenutzten Engines für Big Data workloads

- arbeitet massiv parallel
- optimiert und cachet intern, um extrem schnell zu arbeiten
- bietet APIs für diverse Sprachen: Java, Scala, Python, SQL
- bietet Erweiterungen für Machine Learning
- steht unter einer Open Source-Lizenz

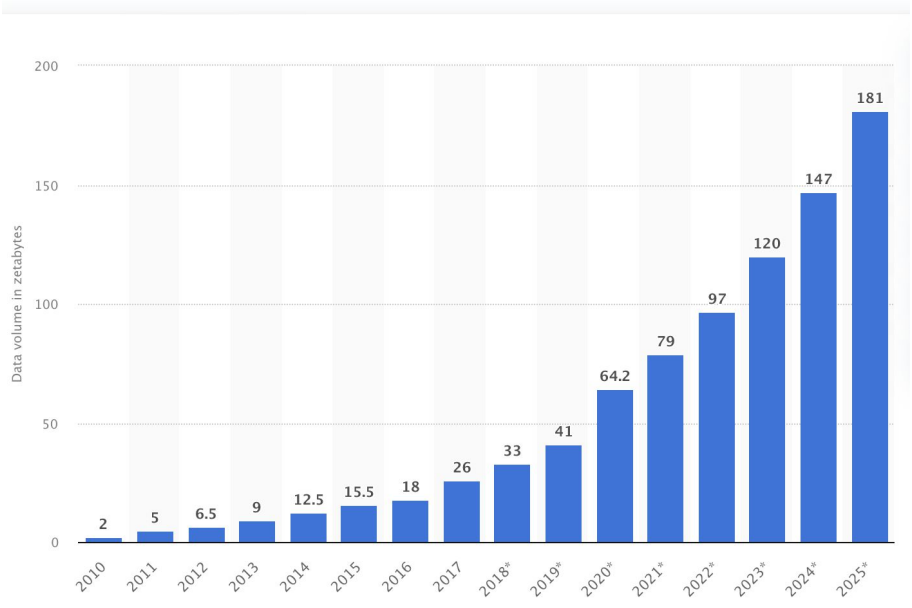
Motivation

Was ist Big Data?

Big Data

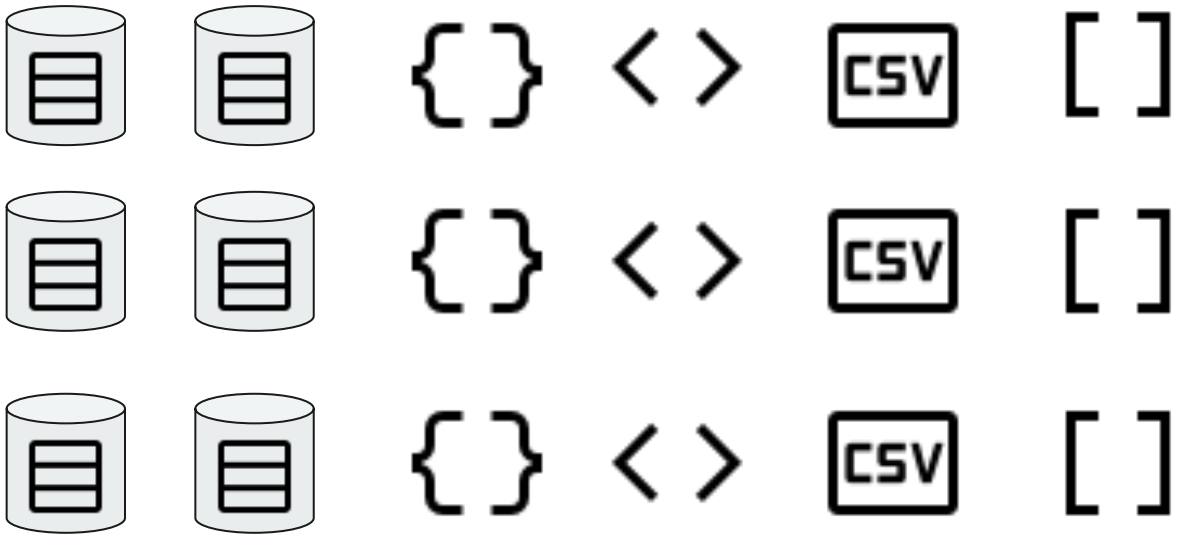
“Datenvolumina, die zu groß sind,
um von einem einzelnen Computer verarbeitet zu werden”

Entstehung von Daten: Volumen



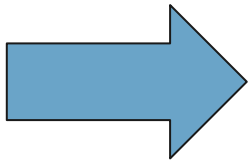
Entstehung von Daten: Beschaffenheit

Entstehung von Daten: Beschaffenheit



Ablage von dispositiven Daten

Data Warehouse



Data Lake/
Lakehouse

Auswertungsbedarf

- Kontinuierliche Reports
- BI/Data Science
- Machine Learning
- Datenprodukte

- Datengetriebene Organisationen
- Time to Market
- Skalierung

Einsatzgebiete von Apache Spark

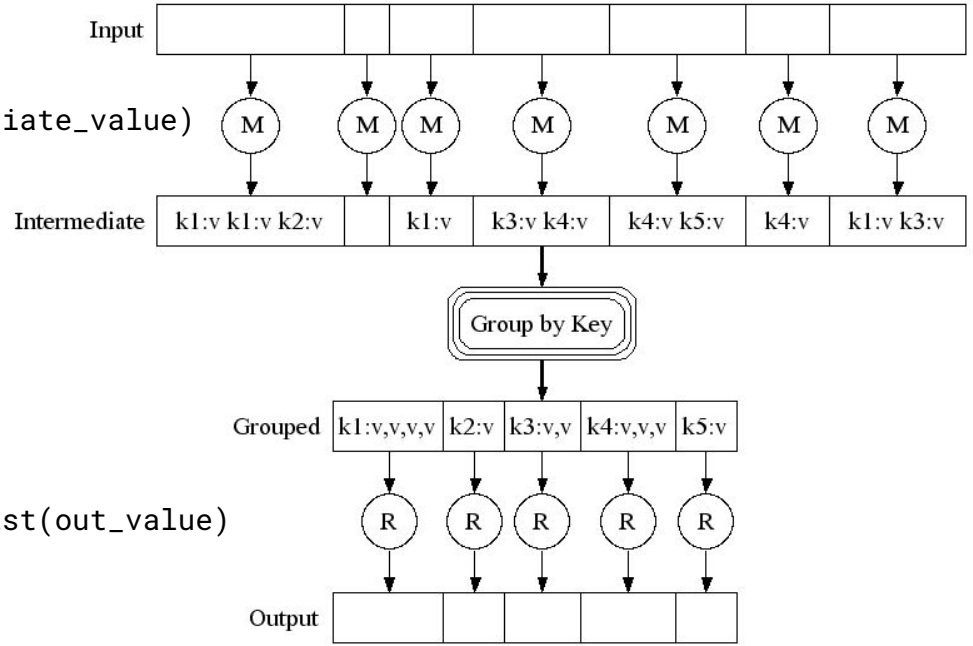
- kontinuierliche Reports
- Ad-Hoc-Analysen
  - geplant
  - explorativ
- Streaming
- Machine Learning
- Graphenanalyse

Entwicklung von Softwarelösungen für Big Data

Entwicklung

Map Reduce

```
map (in_key, in_value) -> list(out_key, intermediate_value)
```




```
reduce (out_key, list(intermediate_value)) -> list(out_value)
```

Map Reduce

Hello World Bye World	Hello Hadoop Goodbye Hadoop

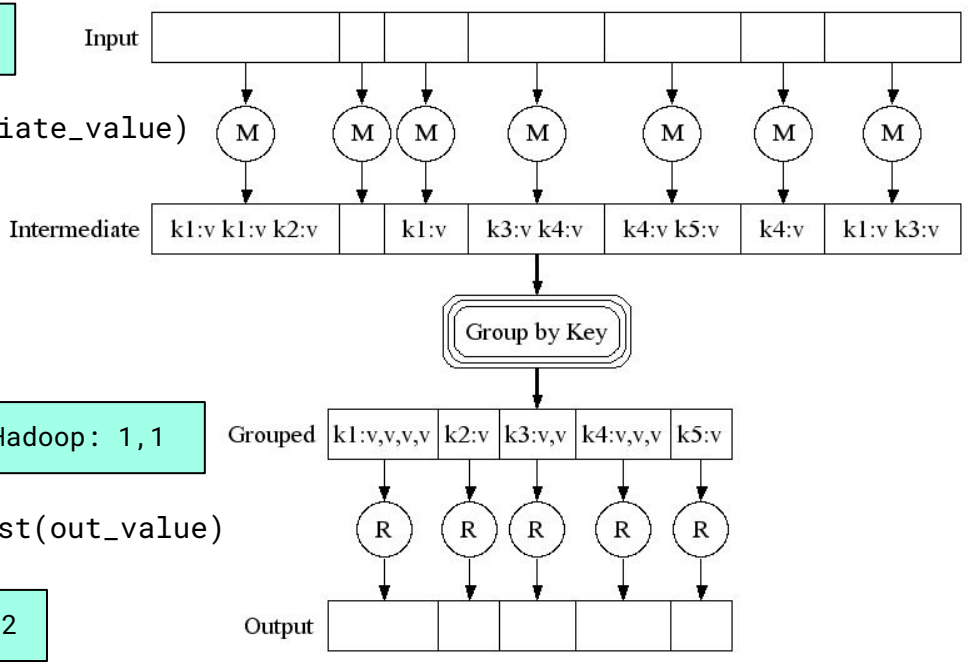
`map (in_key, in_value) -> list(out_key, intermediate_value)`

Hello: 1 World: 1 Bye: 1 World: 1	Hello: 1 Hadoop: 1 Goodbye: 1 Hadoop: 1

Hello: 1, 1	World: 1, 1	Bye: 1	Goodbye: 1	Hadoop: 1,1

`reduce (out_key, list(intermediate_value)) -> list(out_value)`

Hello: 2	World: 2	Bye: 1	Goodbye: 1	Hadoop: 2



Map Reduce

Hello World Bye World	Hello Hadoop Goodbye Hadoop

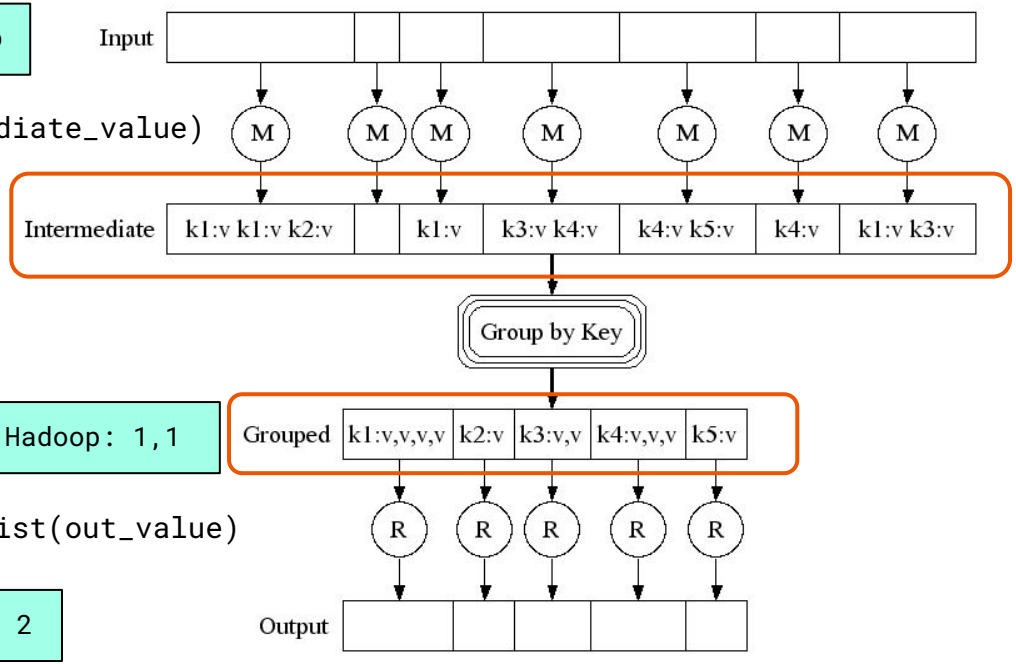
`map (in_key, in_value) -> list(out_key, intermediate_value)`

Hello: 1 World: 1 Bye: 1 World: 1	Hello: 1 Hadoop: 1 Goodbye: 1 Hadoop: 1

Hello: 1, 1	World: 1, 1	Bye: 1	Goodbye: 1	Hadoop: 1,1

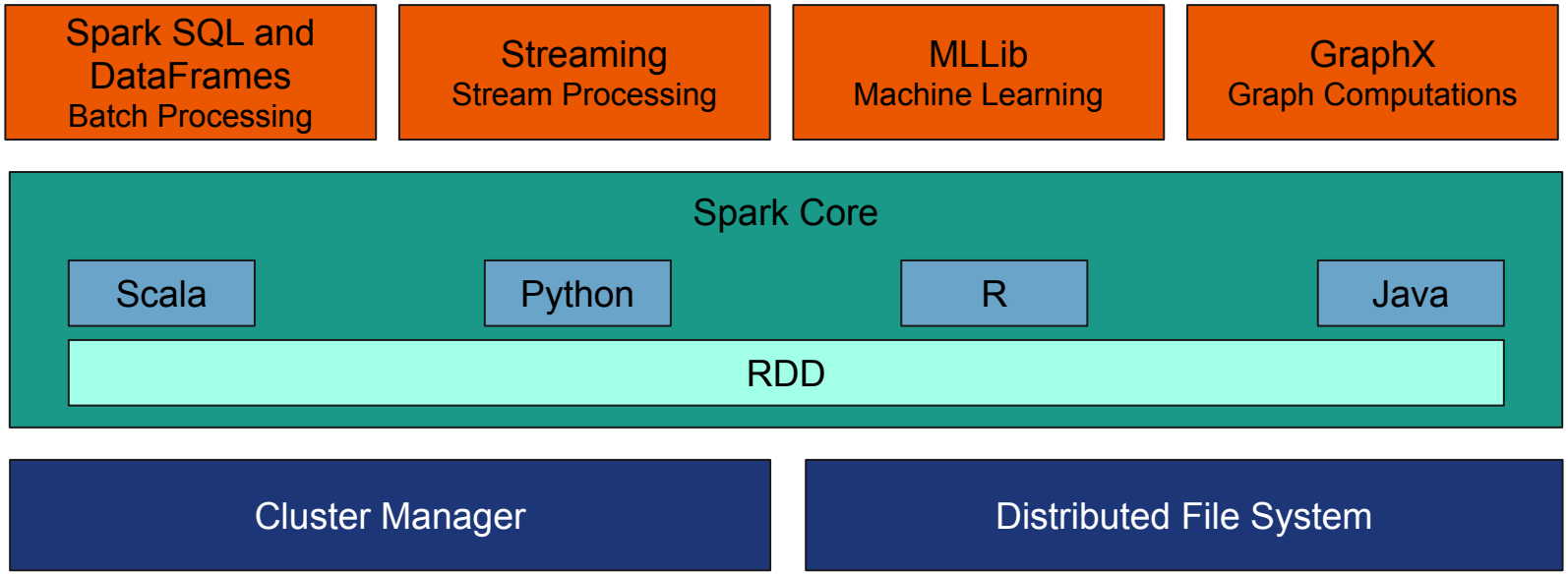
`reduce (out_key, list(intermediate_value)) -> list(out_value)`

Hello: 2	World: 2	Bye: 1	Goodbye: 1	Hadoop: 2



Aufbau und Funktionsweise von Apache Spark

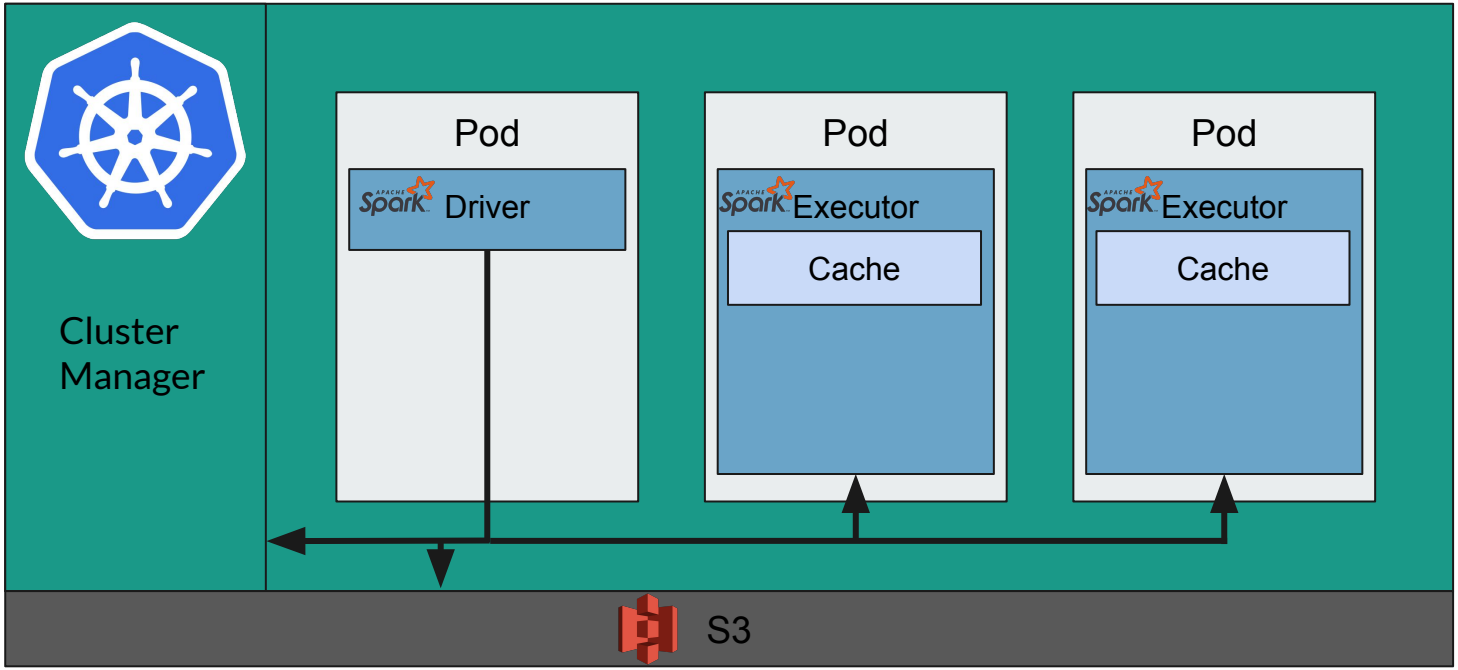
Bausteine


Spark SQL and DataFrames
Batch Processing
Streaming
Stream Processing
MLLib
Machine Learning
GraphX
Graph Computations
Spark Core
Scala
Python
R
Java
RDD
Cluster Manager
Distributed File System

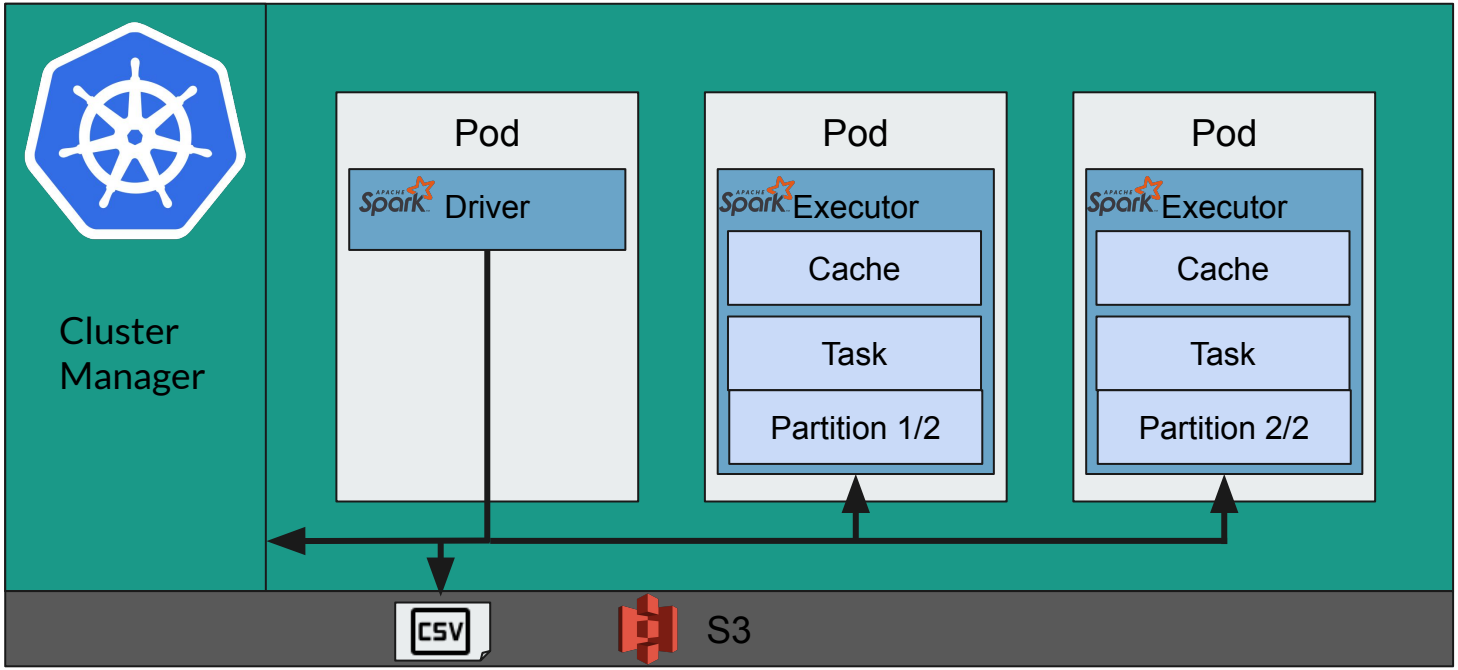
Wie kann ich Spark nutzen?

- Command Line/Spark Shell
- Notebooks wie Jupyter und Zeppelin
- IDEs wie PyCharm, IntelliJ, …
- Cloud Services wie Databricks, GCP BigQuery

Verteilung


Cluster
Manager
Pod
Spark Driver
Pod
Spark Executor
Cache
Pod
Spark Executor
Cache
S3

Ausführung

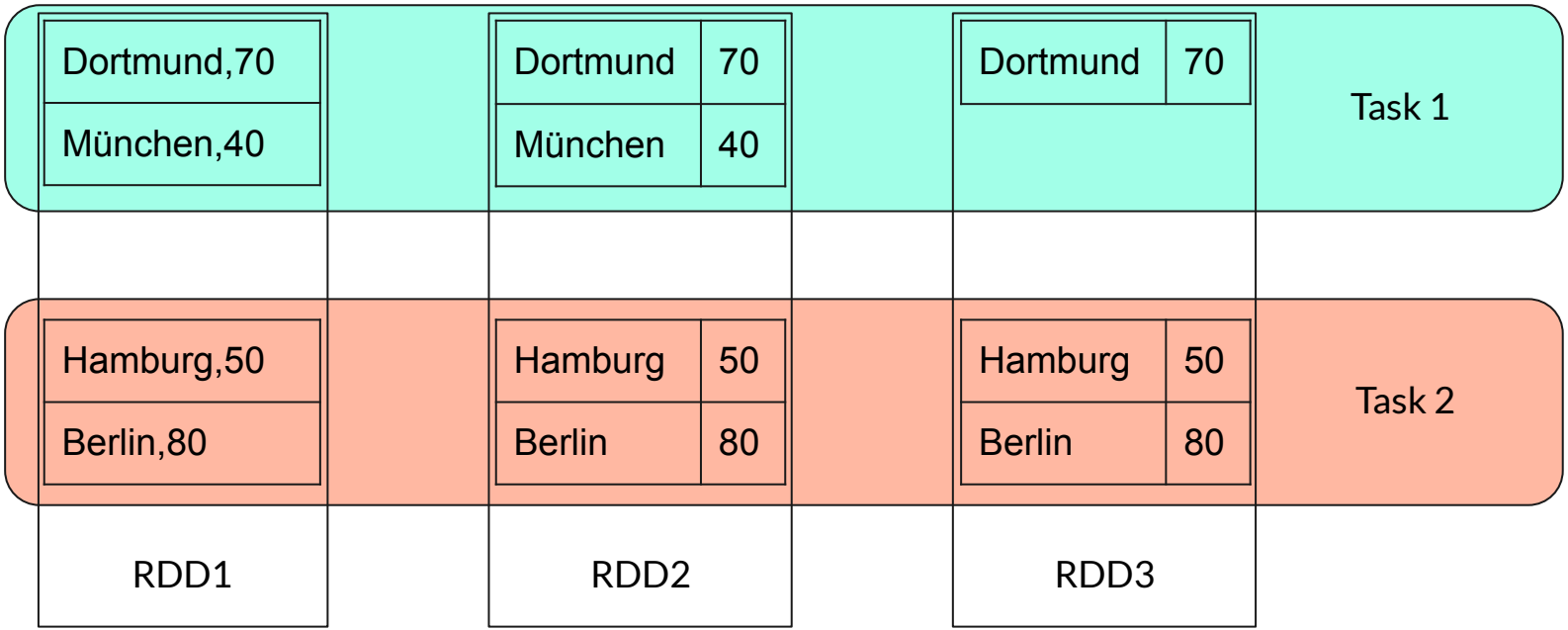

Cluster Manager
Pod
Spark Driver
Pod
Spark Executor
Cache
Task
Partition 1/2
Pod
Spark Executor
Cache
Task
Partition 2/2
CSV
S3

RDD: Resilient Distributed Dataset

- grundlegende Datenstruktur in Spark

- Verteilt auf mehrere Partitionen
- Unveränderlich
- Fehlertolerant

RDDs, Partitionen und Tasks



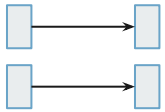
Arten von Operationen

- Transformation: Operationen RDDs → RDD
  - z.B. filter, map, distinct, reduceByKey
  - Lazy evaluation

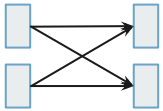
- Aktionen: Erzwingen die Rückgabe eines Ergebnisses an das Driver-Modul
  - z.B. count, reduce, Ergebnisse schreiben
  - Eager evaluation

Arten von Transformationen

- Narrow: Können ohne Shuffling ausgeführt werden
  - Input-Partition wird nur einmal verwendet
  - z.B. map, filter
  - auf vielen Partitionen parallel
  - Typisch: Eingabe-Partitionen=Ausgabe-Partitionen



- Wide: Erfordern Shuffling
  - Input-Partition wird mehrfach verwendet
  - z.B. groupByKey, reduceByKey, sortBy



Beispieldatensatz: kennzahlen.csv

```
Dortmund,70
München,40
Hamburg,50
Dortmund,20
Berlin,80
Hamburg,30
```

Beispieldatensatz

Dortmund,70
München,40
Hamburg,50
Dortmund,20
Berlin,80
Hamburg,30

Einlesen

Beispieldatensatz

Dortmund	70
München	40
Hamburg	50
Dortmund	20
Berlin	80
Hamburg	30

1) **Aufspalten am Komma**
2) Einträge zählen
3) Gruppieren und Summieren
4) Filtern > 50
5) Ausgabe schreiben

Beispieldatensatz

Dortmund	70
München	40
Hamburg	50
Dortmund	20
Berlin	80
Hamburg	30

=> 6

1) Aufspalten am Komma
2) **Einträge zählen**
3) Gruppieren und Summieren
4) Filtern > 50
5) Ausgabe schreiben

Beispieldatensatz

Dortmund	90
München	40
Hamburg	80
Berlin	80

1) Aufspalten am Komma
2) Einträge zählen
3) **Gruppieren und Summieren**
4) Filtern > 50
5) Ausgabe schreiben

Beispieldatensatz

Dortmund	90
Hamburg	80
Berlin	80

1) Aufspalten am Komma
2) Einträge zählen
3) Gruppieren und Summieren
4) **Filtern > 50**
5) Ausgabe schreiben

Beispieldatensatz

Dortmund	90
Hamburg	80
Berlin	80

1) Aufspalten am Komma
2) Einträge zählen
3) Gruppieren und Summieren
4) Filtern > 50
5) **Ausgabe schreiben**

Beispieldatensatz

Dortmund	90
Hamburg	80
Berlin	80

1) Aufspalten am Komma
2) Einträge zählen
3) Gruppieren und Summieren
4) Filtern > 50
5) Ausgabe schreiben

Aktionen

Beispieldatensatz

Dortmund	90
Hamburg	80
Berlin	80

Job

1) Aufspalten am Komma
2) Einträge zählen

3) Gruppieren und Summieren
4) Filtern > 50
5) Ausgabe schreiben

Beispieldatensatz

Dortmund	90
Hamburg	80
Berlin	80

1) Aufspalten am Komma
2) Einträge zählen
3) Gruppieren und Summieren
4) Filtern > 50
5) Ausgabe schreiben

Transformationen

- Narrow
- Wide

Beispieldatensatz

Dortmund	90
Hamburg	80
Berlin	80

Stage, Task

1) Aufspalten am Komma
2) Einträge zählen
3) Gruppieren und Summieren
   Shuffle
   - schreiben

   - lesen
4) Filtern > 50
5) Ausgabe schreiben