Big and Fast Data - Verarbeitung von Streaming Data

AW2 Präsentation

Gerrit Thede

Fakultät Technik und Informatik
Department Informatik
HAW Hamburg

17. April 2014

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Big and fast Data

Skalierbare Big Data Systeme

- Skalierbar auf vielen Knoten
- Hadoop Filesystem
- Map Reduce
- Stapelverarbeitung von Offline Daten

Big and Fast Data Systeme

- Streaming Data
- Kontinuierliche Verarbeitung von Datenströmen
- Inkrementelle Verarbeitung
- geringe Latenzen
- Vermeidung von Neuberechnung der Datensätze

Vorstellung von drei Konzepten

- Incoop: MapReduce for Incremental Computations
- Discretized Streams: Fault-Tolerant Streaming Computation at Scale
- Muppet: MapReduce-Style Processing of Fast Data

Gemeinsamkeiten

- Einsatz des MapReduce Konzepts
- Zwischenergebnisse werden als Zustand gespeichert
- Wiederverwendbarkeit von nicht inkrementellen Algorithmen

Incoop: MapReduce for Incremental Computations

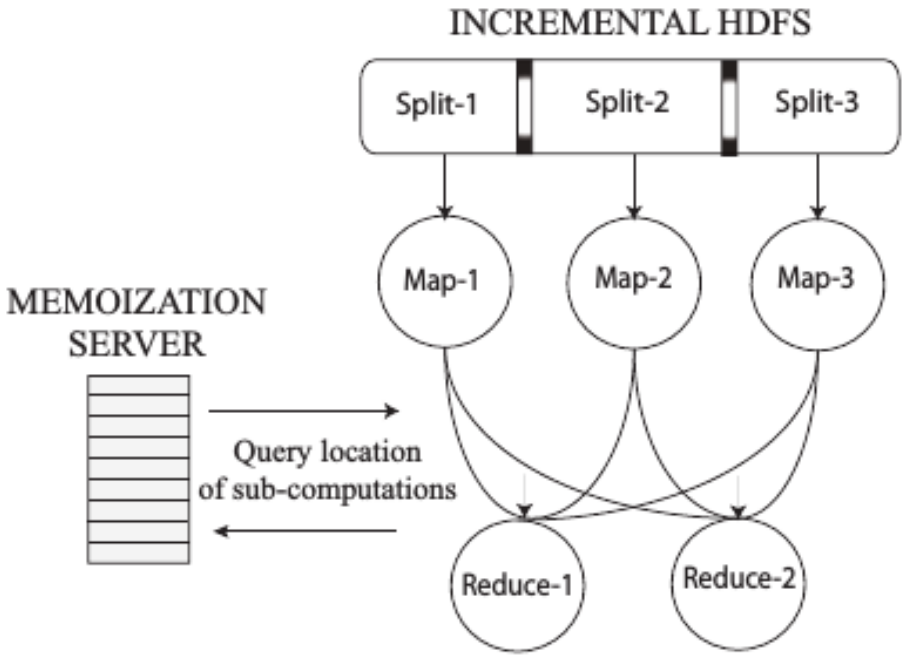
2011, Max Planck Institute for Software Systems et al. [BWR+11]

- Incoop: inkrementelles MapReduce Framework
- Erweiterung des Hadoop Frameworks
- Neuer Input erzeugt automatisch eine Neuberechnung des Ergebnis
- Effizient durch Wiederverwendung von Zwischenergebnissen
- Transparent für den Entwickler

Incoop

Erweiterungen des Hadoop Frameworks

- Incremental HDFS
- Memoization
- Incremental MapReduce

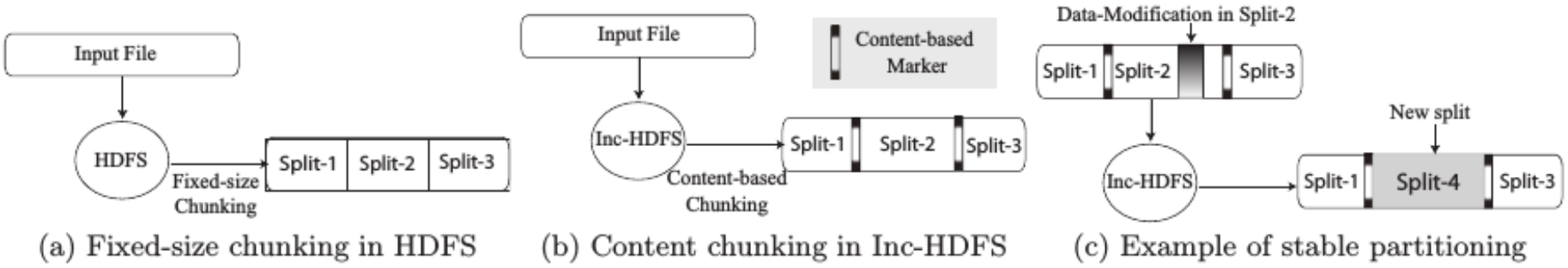


Incoop System Design [BWR+11]

Incoop

Incremental HDFS

- HDFS ist Append-only Dateisystem
- Daten Chunks basieren auf Ähnlichkeit, nicht fester Größe
- Ähnlicher Input wird an gleicher Stelle gespeichert (Fingerprint)
- Kompatibel zu HDFS API



File Chunking Strategie [BWR+11]

Incoop

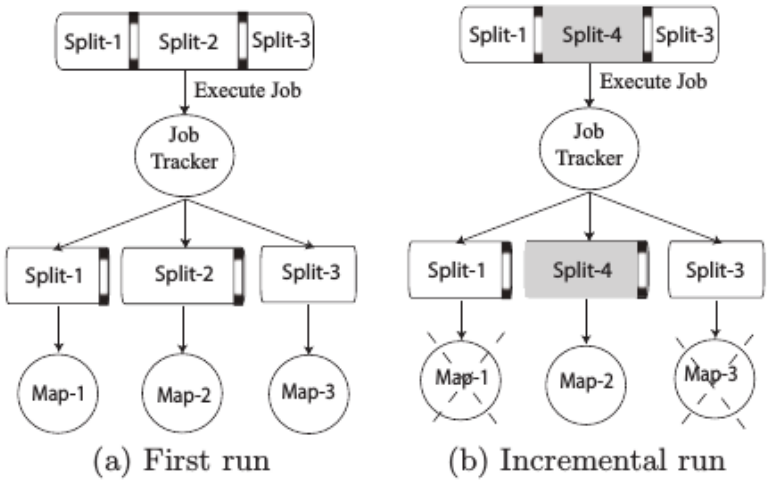
Memoization

- Cache von Ergebnissen eines Funktionsaufrufs
- deterministische Funktion liefert selbes Ergebnis bei gleichem Input
- wie Lookup Table
- erweiterter Hadoop Scheduler merkt sich, welche Node welches Ergebnis lieferte
- Lokalität der Tasks auf Nodes wird berücksichtigt für Effizienz

Incoop
Incremental MapReduce

Incremental Map

- Task-level Memoization
- Map Ergebnis wird persistent gespeichert
- Referenz zu Ergebnis speichert Memoization Server
- Folgedurchläufe werden effizienter

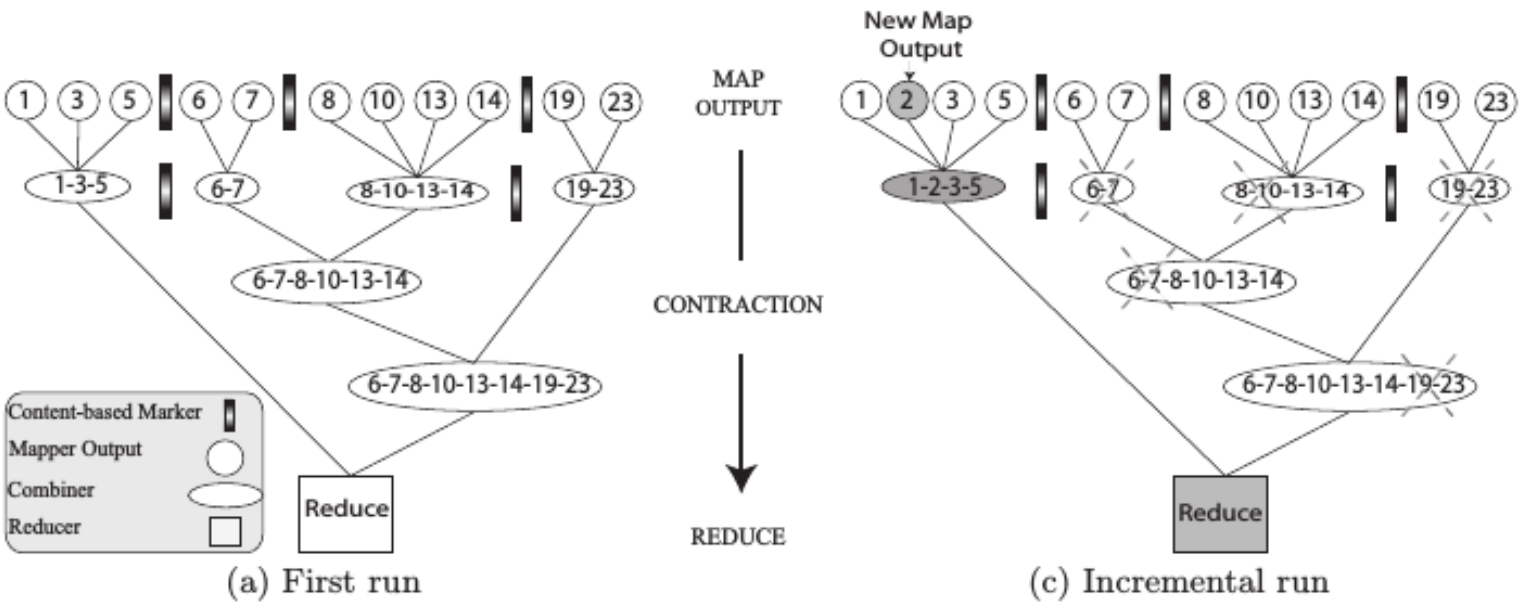


Incremental Map Tasks [BWR+11]

Incoop
Incremental MapReduce

Incremental Reduce

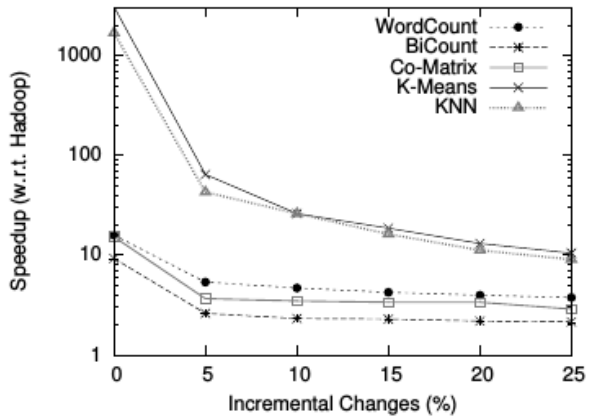
- Reduce Funktion sammelt Map Ergebnisse
- Erzeugung von Zwischenergebnissen in Contraction Phase
- Memoization aller Ergebnisse



(a) First run (c) Incremental run

Incremental Reduce mit Contraction Phase [BWR+11]

Incoop Ergebnisse



Beschleunigung bei verändertem Input gegenüber Hadoop [BWR+11]

- Performance Analyse mit Standard Algorithmen
- Performance Overhead durch Memoization
- Speicherplatz Overhead durch Zwischenergebnisse

Discretized Streams: Fault-Tolerant Streaming Computation at Scale

University of California, Berkeley [ZDL+13]

Spark Streaming, http://spark.apache.org/streaming/

- Latenzen unter einer Sekunde
- fehlertolerant, sehr schnelle Fehlerbehebung
- hohe Skalierbarkeit
- diskretisierte Streams
- Interaktive Queries
- MapReduce Stil
- Kombination von historischen Daten und Stream

Spark Streaming

Ansatz anderer Streaming Systeme

- Continuous Operator Model: Zustandsbehaftete Operatoren
- Probleme bei langsamen (Stragglers) und ausgefallenen Nodes
- Replikation notwendig: doppelte Hardware
- oder Backup: langsame Erholung bei Ausfall

Spark Streaming
D-Stream Modell

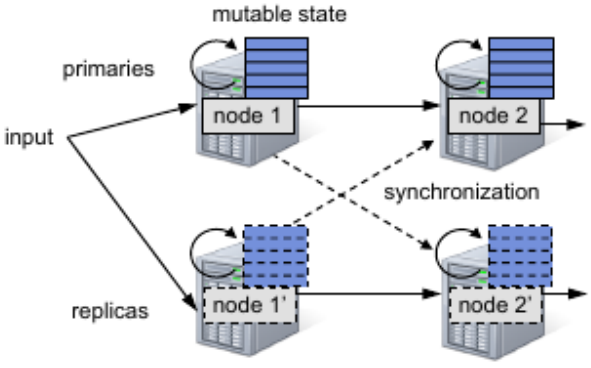
Ansatz mit diskretisierten Streams

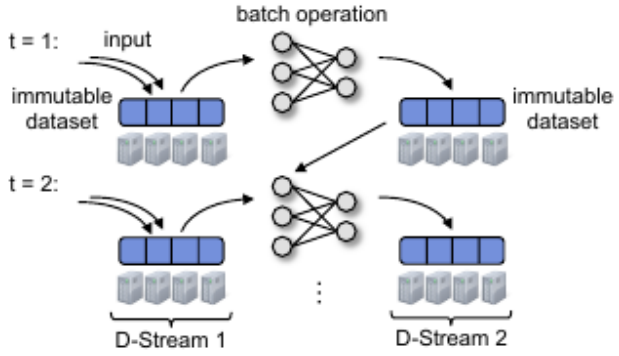
- Serie zustandsloser, deterministischer verarbeiteter Daten von kleinen Zeitintervallen
- keine Synchronisation des Zustands nötig
- Gruppierung der D-Streams nach eingehenden Timestamps
- Intervall z.B. 1Sek

Spark Streaming

Resilient Distributed Data Sets [ZCD+12]

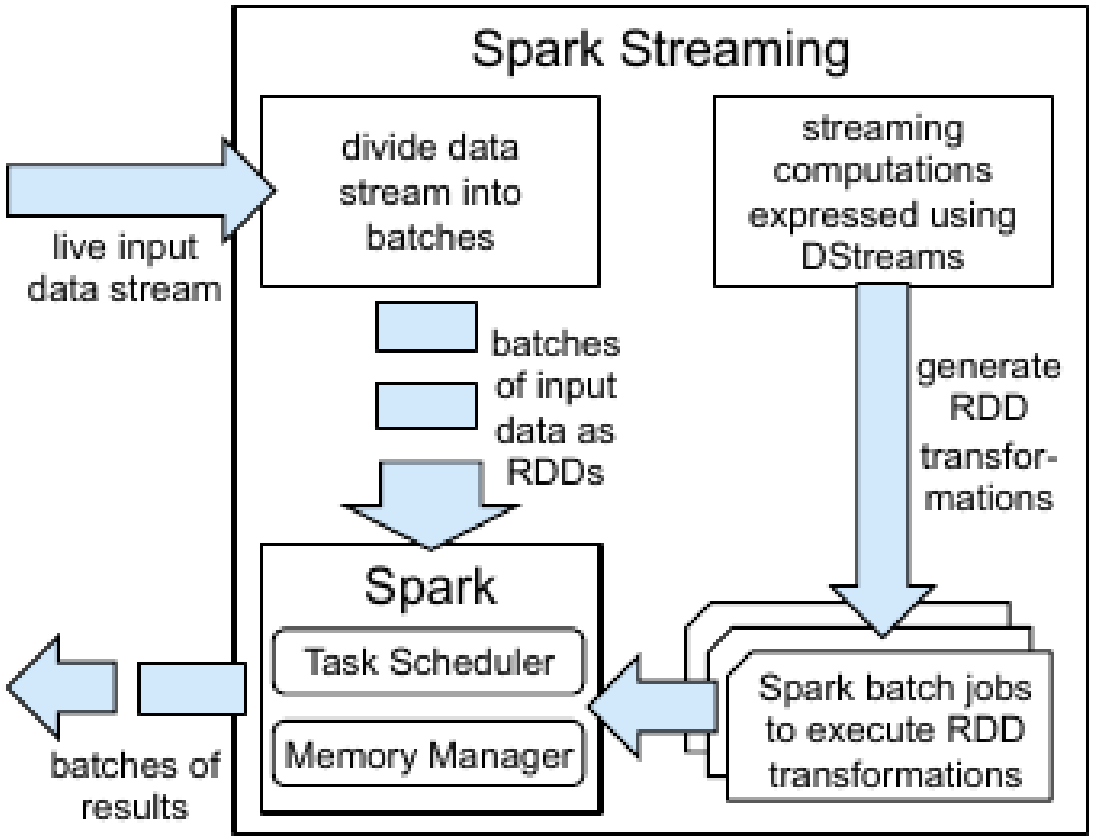
- read-only Sammlung von Datensätzen
- In-Memory Datenstruktur statt On-Disk
- Lineage Graph: Speichert Abstammung der Daten (mit welchen Operationen wurden sie erzeugt)
- Fällt ein Knoten aus, erzeugen alle Knoten des Clusters parallel Teile des verlorenen RDD





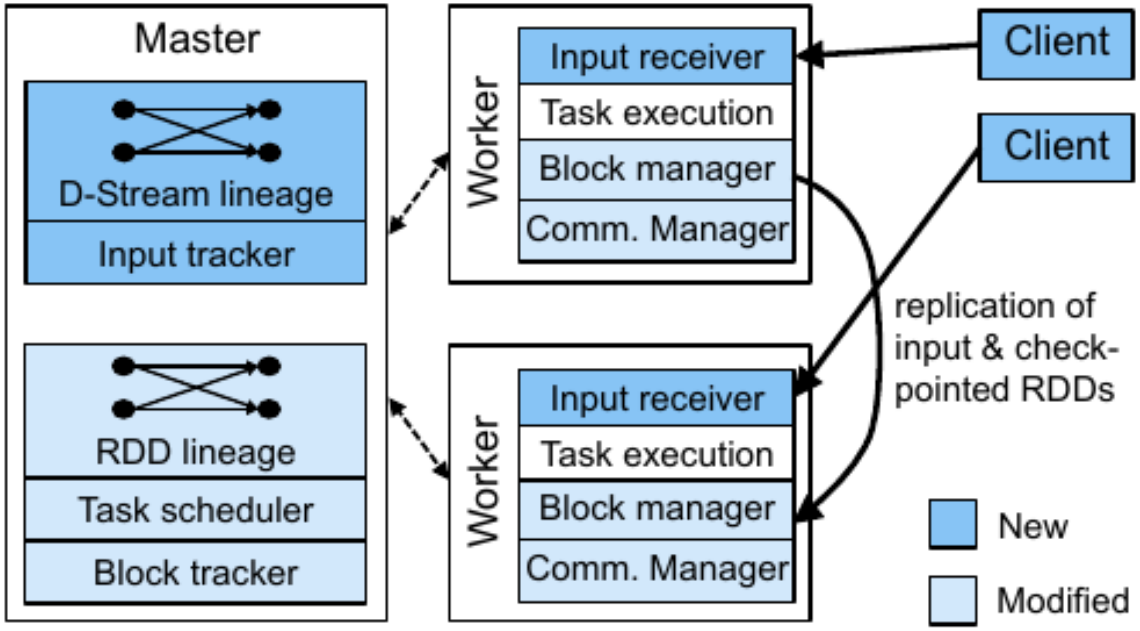
Replikation bei Continuous Operator Model und D-Stream Processing Model [ZDL+13]

Spark Streaming



Spark Streaming System Überblick [ZDL+13]

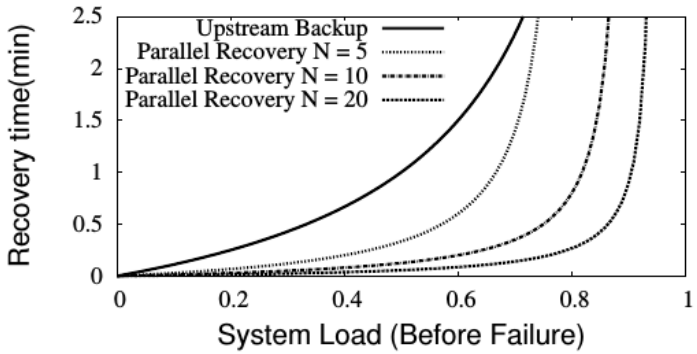
Spark Streaming



Spark Streaming Komponenten [ZDL+13]

- hohe Datenlokalität
- Block Store jedes Workers hat Least Recently Used Strategie
- Lineage Tracker verwirft alte Daten bei Checkpoints

Spark Streaming



Recovery Zeit bei einer Node mit Backup System gegenüber Spark [ZDL+13]

- Vorteil durch parallele Rekonstruktion der Daten eines ausgefallenen Knoten
- Nebenbei werden weiterhin Streams verarbeitet ohne Rückstau
- überlastete Nodes (Straggler) werden automatisch erkannt und Aufgaben neu verteilt vor Reduce

Muppet: MapReduce-Style Processing of Fast Data

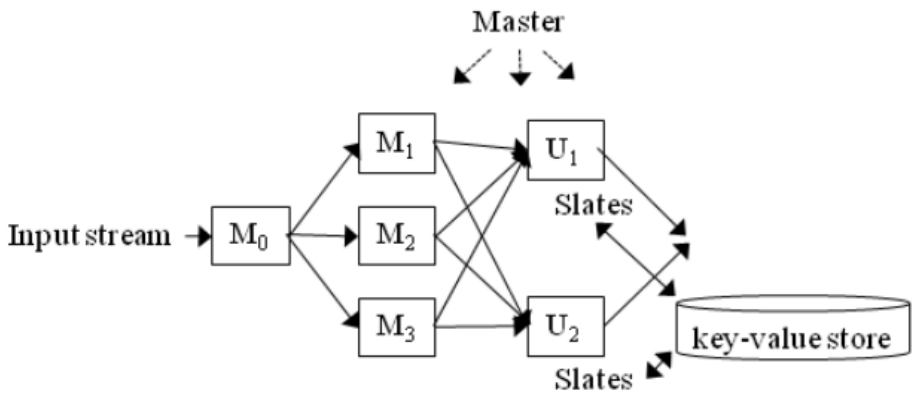
WalmartLabs, University of Wisconsin-Madison [LLP+12]

Muppet = MapUpdate Framework

- Framework wie MapReduce
- speziell entwickelt für Fast Data
- arbeitet auf Data Streams
- „Slates“ summieren Daten, die bisher verarbeitet wurden
- Stream „Events“ sind Tupel mit (StreamID, Timestamp, Key, Value)

MapUpdate Framework

- **Map Functions**: Input sind die Events von Streams in chronologischer Ordnung. Erzeugt Output Events mit neuem Timestamp.
- **Update Functions**: Input sind auch Stream Events. Für jeden Key wird eine Slate angelegt.
- **Slates**: speichert Zusammenfassung der Updates zu einem Key und wird kontinuierlich aktualisiert. Speicherung in-memory sowie in einem persistenten Key-Value Store.



Muppet System [LLP+12]

MapUpdate Framework

Fehlerbehandlung

- Worker werden in kurzen Intervallen auf Ausfall geprüft
- Bei Ausfall werden die verlorenen Events verworfen und nicht wiederhergestellt
- niedrige Latenz wird erreicht durch Verzicht auf Replikation
- Overflow: ist eine Worker-Queue voll, werden keine neuen Events angenommen und auf andere Nodes verteilt.

Zusammenfassung

- **Incoop**: inkrementelles Batchprocessing
- **Spark Streaming**: fehlertolerantes schnelles Streaming, Open Source
- **Muppet**: schnelles Streaming

Ende

Vielen Dank für Ihre Aufmerksamkeit!

Referenzen

[BWR+11] Bhatotia, Pramod, Alexander Wieder, Rodrigo Rodrigues, Umut A. Acar und Rafael Pasquin: *Incoop: MapReduce for Incremental Computations*. In: *Proceedings of the 2Nd ACM Symposium on Cloud Computing*, SOCC '11, Seiten 7:1–7:14, New York, NY, USA, 2011. ACM.

[LLP+12] Lam, Wang, Lu Liu, Sts Prasad, Anand Rajaraman, Zoheb Vacheri und AnHai Doan: *Muppet: MapReduce-style Processing of Fast Data*. Proc. VLDB Endow., 5(12):1814–1825, August 2012.

[ZCD+12] Zaharia, Matei, Mosharaf Chowdhury, Tathagata Das, Ankur Dave, Justin Ma, Murphy McCauley, Michael J. Franklin, Scott Shenker und Ion Stoica: *Resilient Distributed Datasets: A Fault-tolerant Abstraction for In-memory Cluster Computing*. In: *Proceedings of the 9th USENIX Conference on Networked Systems Design and Implementation*, NSDI'12, Seiten 2–2, Berkeley, CA, USA, 2012. USENIX Association.

[ZDL+13] Zaharia, Matei, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker und Ion Stoica: *Discretized Streams: Fault-tolerant Streaming Computation at Scale*. In: *Proceedings of the Twenty-Fourth ACM Symposium on Operating Systems Principles*, SOSP '13, Seiten 423–438, New York, NY, USA, 2013. ACM.