Thema 2.3: Apache Spark

Skalierbare verteilte Datenanalyse

Lukas Wappler

12. Mai 2017

Inhaltsverzeichnis

"Statistical thinking will one day be as necessary for efficient citizenship as the ability to read and write."

H.G. WELLS (1866-1946)
Science-Fiction-Roman-Autor
Der Krieg der Welten

Einleitung

Heutige Probleme:

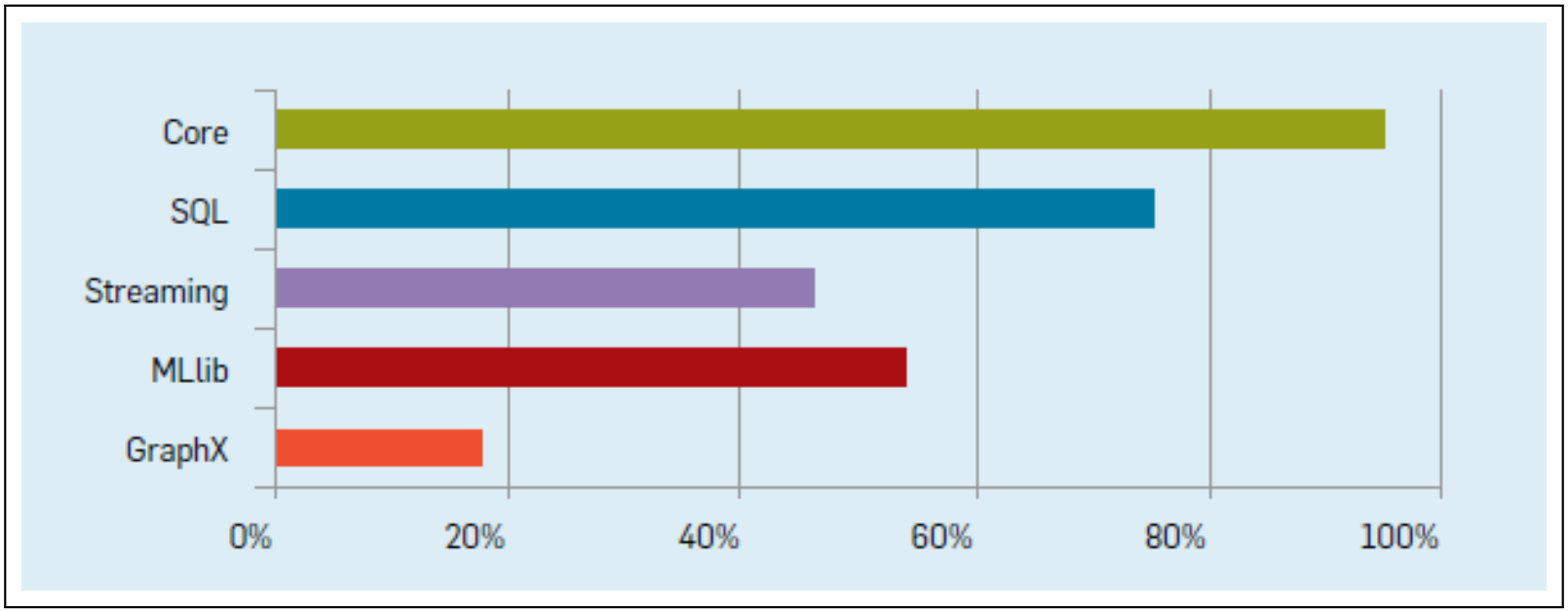
- Immer mehr Daten
- Datenanalysen müssen immer schneller werden
- Keine teuren Supercomputer nutzen

Ist Apache Spark die Lösung?

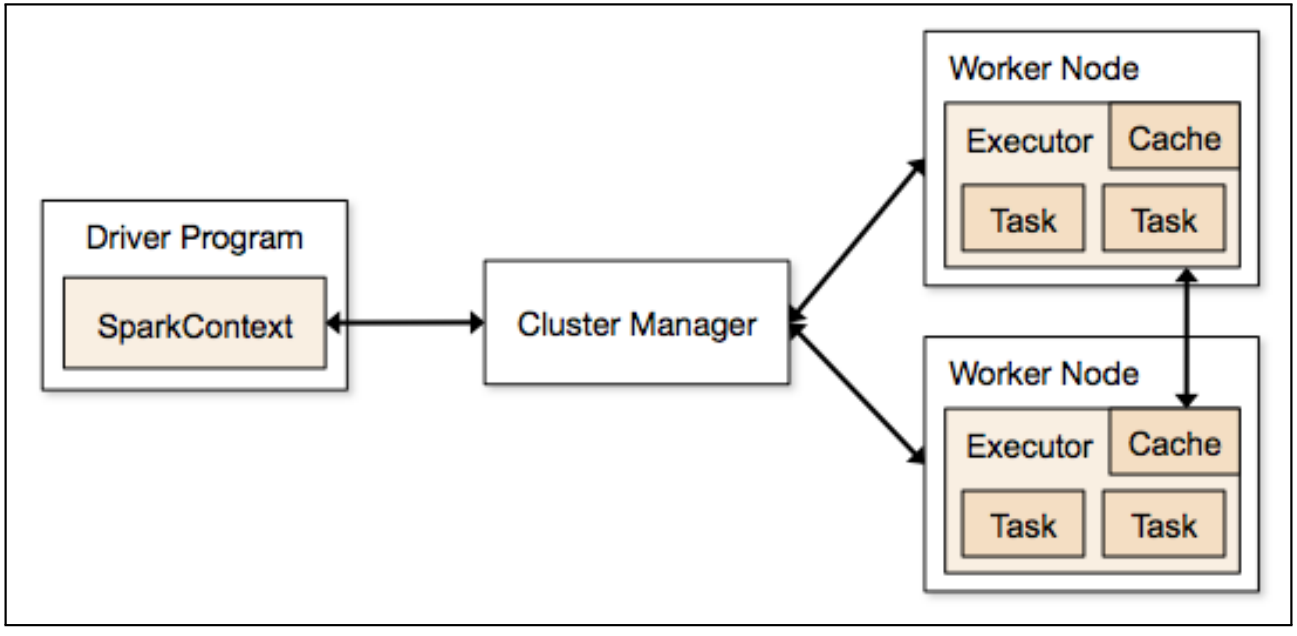
Apache Spark

- 2009 im AMPLab ins Leben gerufen
- 2013 von der Apache Software Foundation übernommen
- 2014 Top Level Projekt
- OpenSource

Kern-Bibliotheken / Komponenten



Spark-Core


Worker Node
Executor
Cache
Task
Task
Driver Program
SparkContext
Cluster Manager
Worker Node
Executor
Cache
Task
Task

RDD's

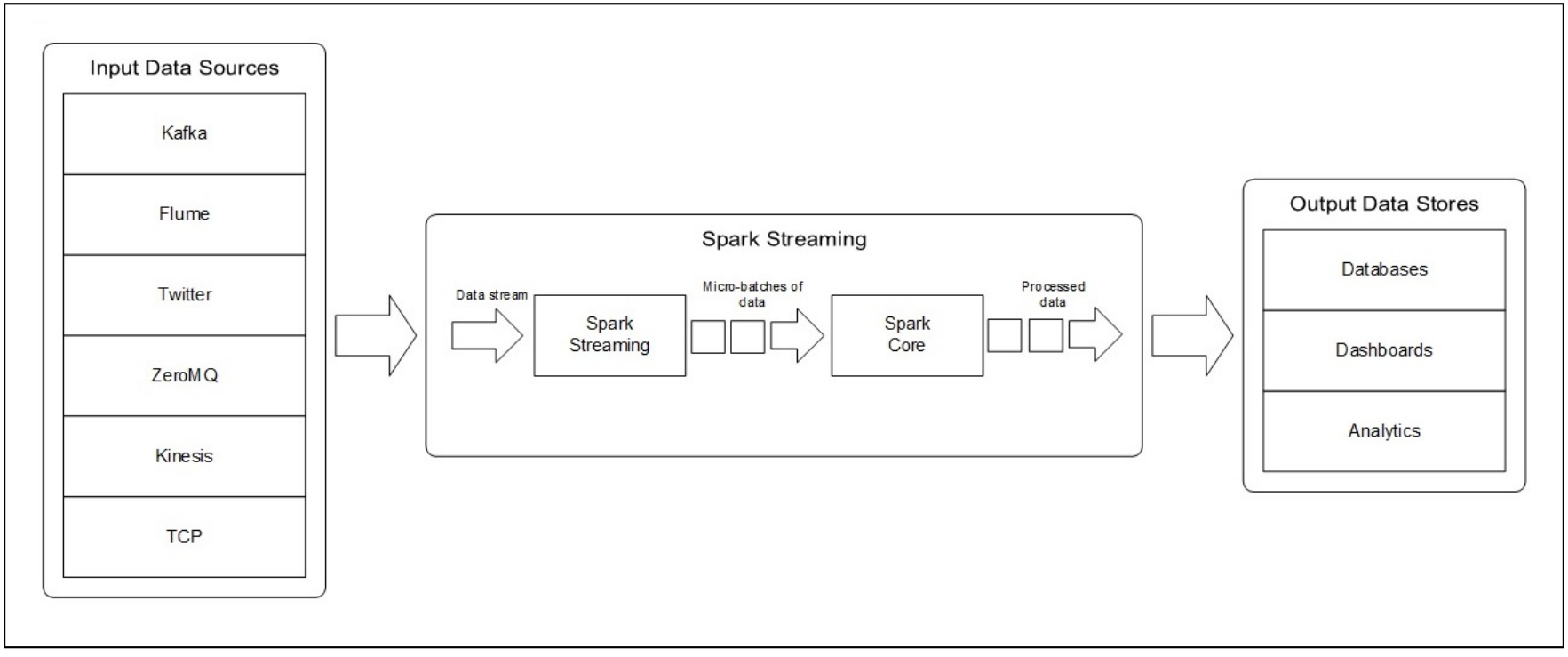
- belastbare (fehlertollerant / ausfallsicher)
- verteilt (Daten lassen sich aufteilen)
- nach Erstellung nur lesbar

Spark-SQL & Dataframes

- Eine Verbesserung von Shark
- kombiniert prozedurale Abfragen & relationale Datenbankabfragen
- Neben RDD's werden auch DataFrames verwendet.
- Anbindung unterschiedlichster Datenquellen

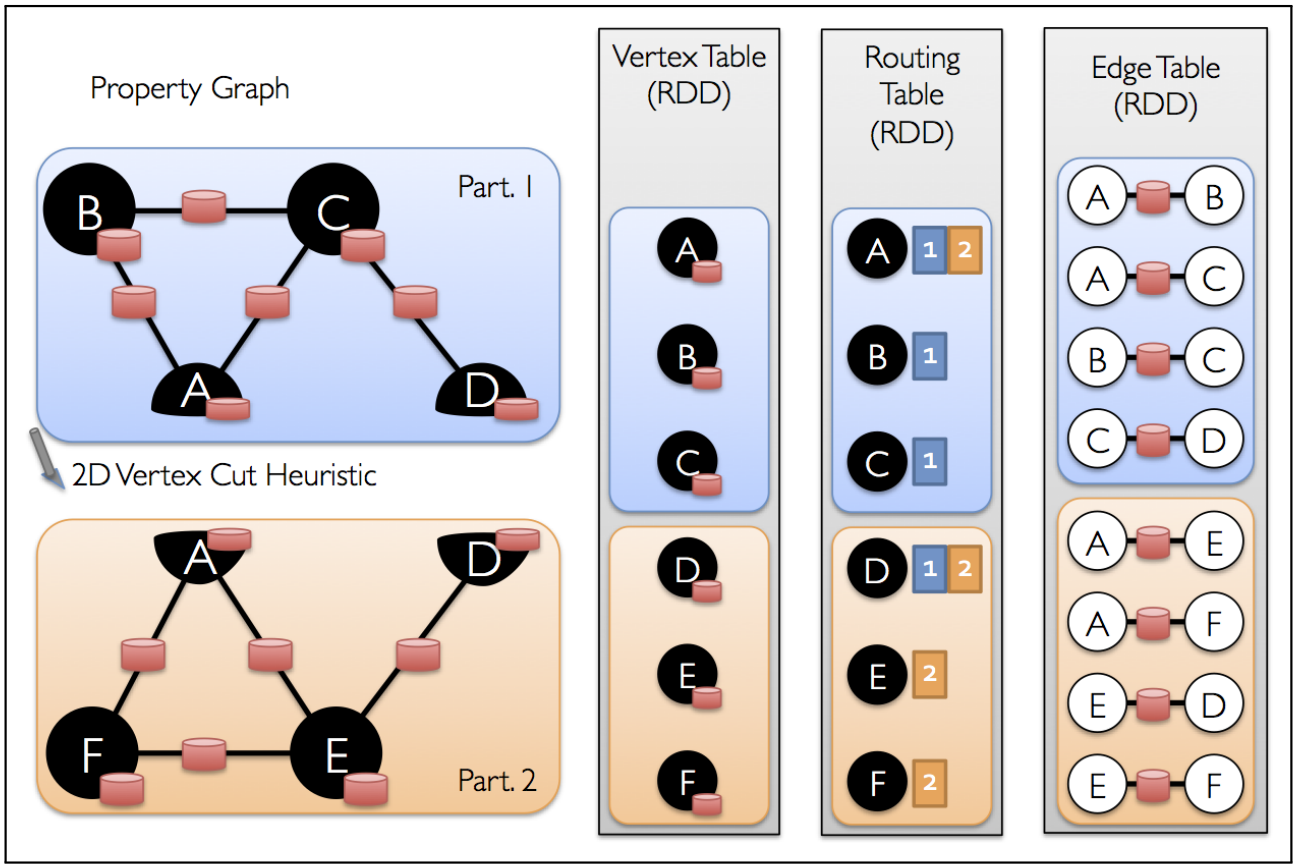
Verarbeitung von Datenströmen (Spark-Streaming)

- RDD's werden zu DStreams erweitert
- DStreams verwalten RDD's



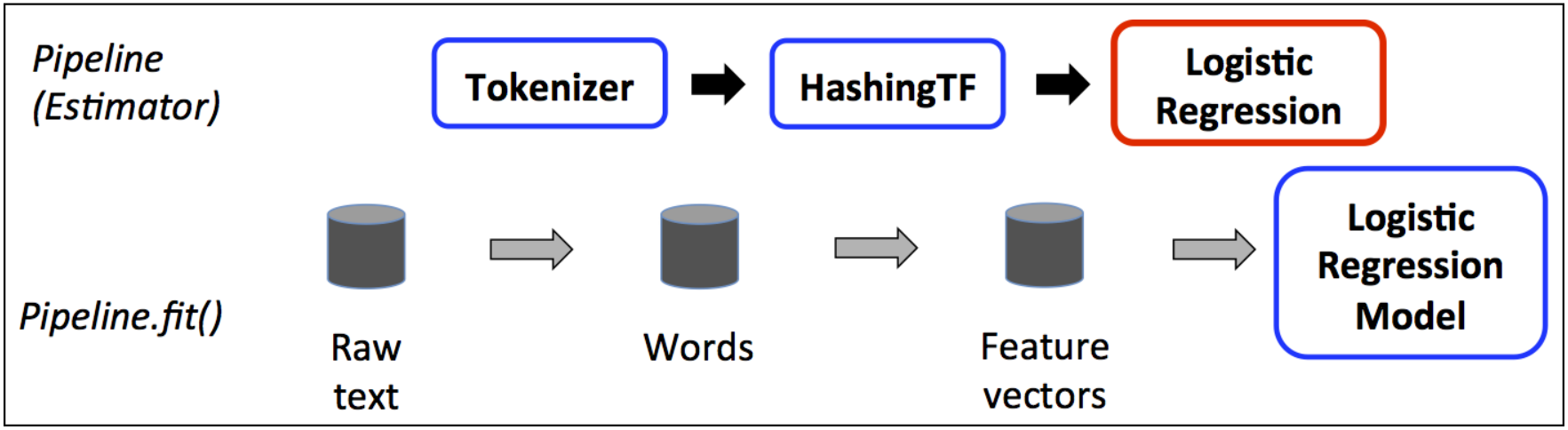
Berechnungen auf Graphen (GraphX)

- Property-Graphen
- Abbildung in RDD-Tupeln
- 1. RDD enthält Ecken
- 2. RDD enthält Kanten



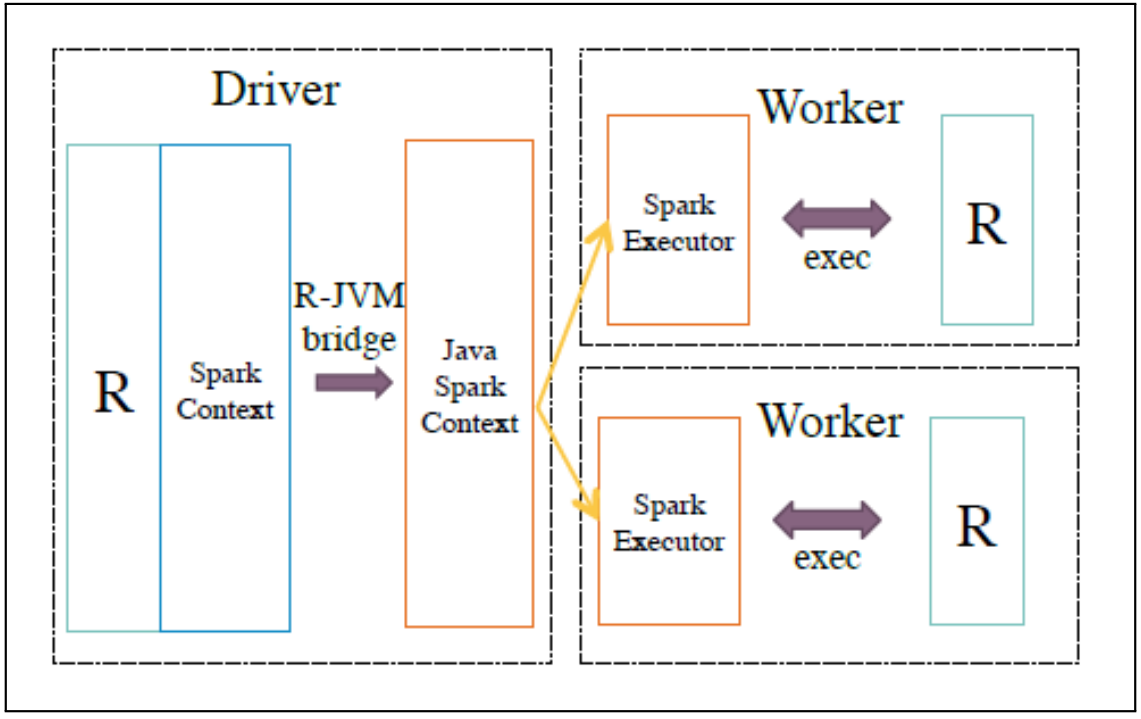
Maschinelles Lernen (MLlib)

- Nutzt DataFrames
- Transformator
- Estimator
- Pipeline



Skalierung von R Programmen (SparkR)

- R läuft nur in einem Thread
- R-JVM Brücke von R zu Java
- Kommunikation über Sockets



Mehrere Komponenten im Verbund

SparkCore, SparkSQL und MLlib im Verbund:

- 100.000 Mitarbeiter Datensätze
- JSON-einlesen
- Suchbegriffe trainieren
- Vorhersagen treffen
- Daten reduzieren
- Sortieren
- Ausgabe der Ergebnisse

Mehrere Komponenten im Verbund

Demo: Live oder Video.

Performance

- schwer zu messen
- Fehler sind schwer zu lokalisieren
- Seiteneffekte bei verteilen Operationen

Besonderheiten bei der Speichernutzung

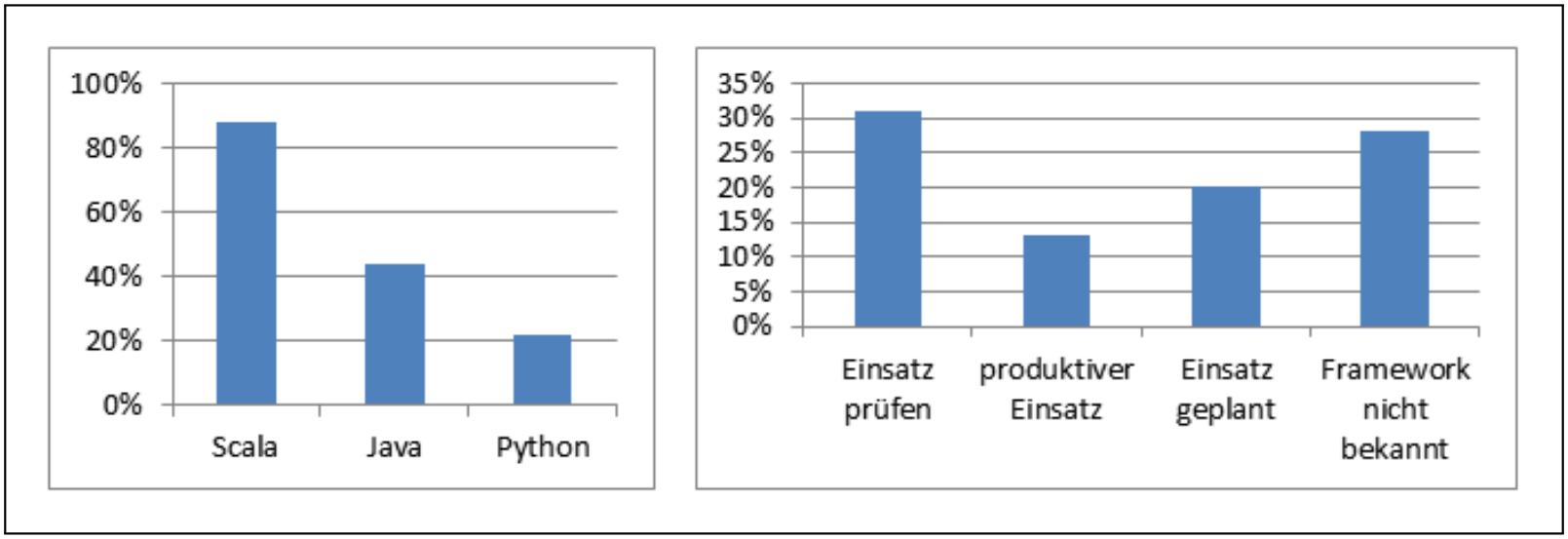
- Nutzung des Arbeitsspeichers
- speicherplatzeffiziente Datenstrukturen
- komprimierte Daten können Blockgrößen überschreiten

Netzwerk und I/O-Traffic

- Zero-copy I/O
- Off-heap network buffer management
- mehrere Verbindungen

Nutzung und Verbreitung

- Skala, Python, Java, (R)
- viele Datenquellen
- viele Dateiformate
- über 400 contributors
- aus über 100 Unternehmen
- Konferenzen (Spark Summit)
- 12,779 Sterne (Github)



- vielfältig einsetzbar
- kann mit Speziallösungen mithalten
- gute Dokumentation & Literatur
- Mischung verschiedener Datenstrukturen schwierig
- veraltete News, Blogs oder Foren
- kostengünstig
- flexibel

Ausblick und Weiterentwicklung

- ständige Weiterentwicklung
- Code ist über Github verfügbar
- 51 Releases (bzw. RC's)
- über 19 commits
- immer wieder Performancesteigerungen

Vielen Dank

Vielen Dank für Ihre Aufmerksamkeit!