Big Data Research in Berlin
BBDC and Apache Flink

Tilmann Rabl
rabl@tu-berlin.de
dima.tu-berlin.de | bbdc.berlin
Agenda

• About Data Management, Big Data, and Data Scientists
• About the Berlin Big Data Center
• Apache Flink – the Next Generation of Big Data Analytics Technologies

a technology-driven perspective
Data & Analysis: Increasingly Complex!

- **Data Volume** too large
- **Velocity** too fast
- **Variability** too heterogeneous
- **Veracity** too uncertain

**Data**

- Reporting
- Ad-Hoc Queries
- ETL/ELT

**Analysis**

- aggregation, selection
- SQL, XQuery
- MapReduce

- Data Mining
- Predictive/Prescriptive
- MATLAB, R, Python

**Tools & Algorithms**

- **MATLAB, R, Python**
- **MapReduce**
- **ETL/ELT**

**Scalability**
Deep Analysis of Big Data

Small Data

Big Data (3V)

Simple Analysis

Deep Analytics

MATLAB

R

hadoop
Databases ➤ “Big Data”

• Tables ➤ Tables and unstructured files
  – Schema on read

• Parallel ➤ More parallel, commodity, shared clusters
  – Mid-query fault tolerance, resource allocation

• SQL ➤ SQL and Java, Scala, Python, you name it
  – General object manipulation

• Data Warehousing ➤ Logs, ML, Graphs, also DW
  – Iterative processing, user-defined functions
“Data Scientist” – “Jack of All Trades!”

Domain Expertise (e.g., Industry 4.0, Medicine, Physics, Engineering, Energy, Logistics)
Mathematical Programming
Linear Algebra
Stochastic Gradient Descent
Error Estimation
Active Sampling
Regression
Monte Carlo
Statistics
Sketches
Hashing
Convergence
Decoupling
Iterative Algorithms
Curse of Dimensionality

Application

ML

Data Science

DM

Scalable Data Management

Machine Learning

Statistics, Data Analysis

Data Warehouse/OLAP
NF²/XQuery
Resource Management
Hardware Adaptation
Fault Tolerance
Memory Management
Parallelization
Scalability
Memory Hierarchy
Data Analysis Language
Compiler
Query Optimization
Indexing
Control Flow
Data Flow
Real-Time

Real-Time
Big Data Analytics Requires Systems Programming

“Big Data’s Big Problem: Little Talent“
Wall Street Journal

Neelie Kroes (ICT 2013, Nov. 7, Vilnius)

R/Matlab: 3 million users

Hadoop: 100,000 users

Data Analysis
Statistics
Algebra
Optimization
Machine Learning
NLP
Signal Processing
Image Analysis
Audio, Video Analysis
Information Integration
Information Extraction
Data Value Chain
Data Analysis Process
Predictive Analytics

Big Data is now where database systems were in the 70s (prior to relational algebra, query optimization and a SQL-standard)!

People with Big Data Analytics Skills
- Indexing
- Parallelization
- Communication
- Memory Management
- Query Optimization
- Efficient Algorithms
- Resource Management
- Fault Tolerance
- Numerical Stability

Declarative languages to the rescue!
Machine Learning + Data Management = X

Think ML-algorithms in a scalable way

Declarative Languages
Automatic Adaption
Scalable processing

Think ML-algorithms in a scalable way

Goal: Data Analysis without System Programming!

Declarative Languages
automatic adaption
Scalable processing

Parallelization
Compiler
Memory Management
Memory Hierarchy
Data Analysis Language
Query Optimization
Dataflow
Indexing

Feature Engineering
Representation Algorithms (SVM, GPs, etc.)

Iteration
Process iterative algorithms in a scalable way

Statistic
Sketches
Hashing
Isolation
Convergence
Curse of Dimensionality
Iterative Algorithms
Control flow

Mathematical Programming
Linear Algebra
Error Estimation
Active Sampling
Regression Monte Carlo

Relational Algebra/SQL
Data Warehouse/OLAP
NF²/XQuery
Scalability
Hardware adaption
Fault Tolerance
Resource Management

Technology X

Control flow

Machine Learning + Data Management = X
X = Big Data Analytics – System Programming! („What“, not „How“)

Description of „What“?
(declarative specification)
Technology X

Think ML-algorithms in a scalable way
Analysis of "data in motion"
Multimodal analysis
Numerical stability

Declarative specification
Automatic optimization, parallelization and hardware adaption of dataflow and control flow with user-defined functions, iterations and distributed state
Scalable algorithms and debugging

Algorithmic fault tolerance
Consistent intermediate results
Software-defined networking

process
Iterative algorithms in a scalable way

Larger human base of „data scientists“
Reduction of „human“ latencies
Cost reduction

Description of „How“?
(State of the art in scalable data analysis)
Hadoop, MPI

Machine

Data Analyst
Application Examples: Technology Drivers and Validation

**Technology X**

- **ML** (Machine Learning)
  - Integration: video, images, text
  - Multimodal data
  - Text data flows
- **DM** (Data Mining)
  - Hierarchical numerical simulation data
  - Numerical stability

---

**Application Example:**
- **Marketplace for information**
  - Economics-based
- **Health**
  - Society-based
- **Material science**
  - Science-based

*Think ML-algorithms in a scalable way*
*Declarative*
*Process iterative algorithms in a scalable way*
Introducing Apache Flink – A Success Story from Berlin, Germany, and Europe

- Project started under the name “Stratosphere” late 2008 as a DFG funded research unit, comprised of TU Berlin, HU Berlin, and the Hasso Plattner Institute Potsdam.

- Apache Open Source Incubation since April 2014, Apache Top Level Project since December 2014

- Fast growing community of open source users and developers in Europe and worldwide, in academia (e.g., SICS/KTH, INRIA, ELTE) and companies (e.g., Researchgate, Spotify, Amadeus)

More information: http://flink.apache.org
**Apache Flink:** General Purpose Programming + Database Execution

- **Draws on Database Technology**
  - Declarativity
  - Query optimization
  - Robust out-of-core

- **Add**
  - Iterations
  - Advanced Dataflows
  - General APIs

- **Draws on MapReduce Technology**
  - Scalability
  - User-defined functions
  - Complex data types
  - Schema on read

---


http://flink.apache.org
What can it be used for?

An engine that can **natively** support all these workloads.

**Batch processing**

**Stream processing**

**Machine Learning at scale**

**Graph Analysis**

Flink
Flink in the Analytics Ecosystem

Applications
- Hive
- Cascading
- Giraph
- Mahout
- Pig
- Crunch

Data processing engines
- MapReduce
- Flink
- Spark
- Storm
- Tez

App and resource management
- Yarn
- Mesos

Storage, streams
- HDFS
- HBase
- Kafka
- ...
Flink Program

```scala
val orderLines = filteredOrders join lines
  on { _.id } isEqualTo { _.orderId }
  map { (c, li) -> OrderTotal(c.id, c.shipPriority, li.price) }
val orderTotals = orderLines
  groupBy { pi -> (pi.orderId, pi.shipPriority) }
  combine { _ reduce addRevenues }
```

Data Flow

Program Compiler

Execution Plan

Flink Optimizer

*Picks data shipping and local strategies, operator order*

Execution Graph

Job Graph

Runtime

*Hash- and sort-based out-of-core operator implementations, memory management*

Parallel Runtime

*Task scheduling, network data transfers, resource allocation*
Built-in vs. driver-based looping

Loop outside the system, in driver program

Iterative program looks like many independent jobs

Dataflows with feedback edges

System is iteration-aware, can optimize the job
Effect of optimization

- Execution Plan A: Run on a sample on the laptop
- Execution Plan B: Hash vs. Sort, Partition vs. Broadcast, Caching, Reusing partition/sort
- Execution Plan C: Run on large files on the cluster, Run a month later after the data evolved

Execution Plan A
- Run on a sample on the laptop

Run on large files on the cluster
- Run a month later after the data evolved
Flink Optimizer
Transitive Closure

- What you write is *not* what is executed
- No need to hardcode execution strategies

Flink Optimizer decides:
- Pipelines and dam/barrier placement
- Sort- vs. hash-based execution
- Data exchange (partition vs. broadcast)
- Data partitioning steps
- In-memory caching
Native Streaming

- Flink execution engine is pipelined (streaming)
  - can implement true streaming & batch

- Usually distributed execution engines materializes intermediate results (batch)
  - can only do micro-batch
Micro Batching vs Native Streaming

Discretized Streams (D-Streams)

while (true) {
    // get next few records
    // issue batch computation
}

Native streaming

while (true) {
    // process next record
}
Stream Discretization

• Data is unbound
  – Interested in a (recent) part of it e.g. last 10 days

• Most common windows around: time, and count
  – Mostly in sliding, fixed, and tumbling form

• Need for data-driven window definitions
  – e.g., user sessions (periods of user activity followed by inactivity), price changes, etc.

The world beyond batch: Streaming 101, Tyler Akidau
Great read!
Flink started as the Stratosphere project in 2009, led by TU Berlin.

Entered incubation April 2014 graduated on December 2014.

Now one of the most active big data projects after over a year in the Apache Software Foundation.
Flink Forward 2015 is the first conference to bring together the Apache Flink developer and user community

- **Companies Use cases:**
  - Bouygues Telecom, Huawei, Telefonica, Amadeus Travel Intelligence, Ericsson, Otto Group, ResearchGate

- **Companies analyse and compare Flink with other tools:**
  - CapitalOne, NFLabs IBM, Zalando, MongoDB

- **Open source projects integrated with Flink:**
  - Docker, Apache Mahout, Apache SAMOA, Apache Zeppelin, Apache BigTop, Cascading, MongoDB, Apache Storm, Google Cloud Dataflow.

- **Training Sessions:**
  - DataStream API, DataSet API, Gelly School: Large-scale graph analysis
Flink Forward

Berlin 12/13 Oct 2015
Apache Flink

Fast and reliable large-scale data processing engine

Download  View on Github

Contributors wanted

http://flink.apache.org
Thank You

Contact:
Tilmann Rabl
rabl@tu-berlin.de