Big Data Management and Apache Flink

Volker Markl
http://www.user.tu-berlin.de/marklv/
http://www.dima.tu-berlin.de
http://www.dfki.de/web/forschung/iam
http://bbdc.berlin

volker.markl@tu-berlin.de
Data Management

- Managing the Complexity of Data Processing
  - Functionality
  - Cost
  - Performance

- Industrialization of the Knowledge Engineering Process

- Study Interactions and Trade-Offs & Build Generic Systems
  - 3P (processing environment, programs, people)
  - Performance metrics
    - Throughput vs. quality measures
    - Human latency vs. system latency

Data Management

© Volker Markl
A (rather incomplete) Data Management timeline

<table>
<thead>
<tr>
<th>Appearance of Relational Databases</th>
<th>First parallel shared-nothing Architectures</th>
<th>Open Source Projects and mainstream Databases</th>
<th>First columnar storage Databases</th>
<th>NoSQL and UDF-based commodity analytics</th>
<th>Alternative MapReduce implementations go mainstream „in-memory“</th>
</tr>
</thead>
<tbody>
<tr>
<td>SQL/OLTP 70s</td>
<td>OLAP/Warehouse 80s</td>
<td>OODBMS 90s</td>
<td>XML-DBMS 00s</td>
<td>„map/reduce“ 2004 - 2009</td>
<td>„in-memory“ 2009 - now</td>
</tr>
<tr>
<td>Ingres</td>
<td>Gamma</td>
<td>DB2</td>
<td>MonetDB</td>
<td>MapReduce Hadoop</td>
<td>Spark</td>
</tr>
<tr>
<td>Oracle</td>
<td>TRACE</td>
<td>Oracle</td>
<td>C-Store</td>
<td>Hadoop</td>
<td>Flink</td>
</tr>
<tr>
<td>System-R</td>
<td>Teradata</td>
<td>BerkeleyDB</td>
<td>Vertica</td>
<td>Giraph</td>
<td>Impala</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MS Access</td>
<td>Aster Data</td>
<td>Hive</td>
<td>HBase</td>
</tr>
<tr>
<td></td>
<td></td>
<td>PostgreSQL</td>
<td>GreenPlum</td>
<td>Hive</td>
<td>SAP Hana</td>
</tr>
<tr>
<td></td>
<td></td>
<td>MySQL</td>
<td>ParAccel</td>
<td>Blu</td>
<td>Blu</td>
</tr>
</tbody>
</table>

1974: SQL

Open Source rises

2003: Internet Explodes

Google publishes MapReduce

Criticism of MapReduce
Some Aspects of the Data Processing Pipeline

Source: bigdatavalue.eu
Data & Analysis: Increasingly Complex!

Data
- data volume too large
- data rate too fast
- data too heterogeneous
- data too uncertain

Analysis
- Volume
- Velocity
- Variability
- Veracity
- Reporting
- Ad-Hoc Queries
- ETL/ELT
- aggregation, selection
- SQL, XQuery
- MapReduce
- Data Mining
- Predictive/Prescriptive
- MATLAB, R, Python
- MATLAB, R, Python
“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
Accuracy

New Technology to the Rescue!
Big Data Analytics Requires Systems Programming

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

R/Matlab: 3 million users
Hadoop: 100,000 users

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

“We will soon have a huge skills shortage for data-related jobs.”
Neelie Kroes (ICT 2013, Nov. 7, Vilnius)

Declarative languages to the rescue!

“We will soon have a huge skills shortage for data-related jobs.”
Wall Street Journal
Declarative Languages to the Rescue! „What“, not „how“ Example: k-Means Clustering

Declarative data analysis program with automatic optimization, parallelization and hardware adaption

65 lines of code
short development time
robust runtime

Hand-optimized code
(data-, load- and system dependent)

486 lines of code
long development time
non-robust runtime
Deep Analysis of „Big Data“ is Key!

- Deep Analytics
  - Python
  - MATLAB
  - R

- Simple Analysis
  - Small Data

- Big Data (3V)
  - Hadoop
  - SQL
A Zoo of Technologies!

http://mattturck.com/wp-content/uploads/2016/03/Big-Data-Landscape-2016-v18-FINAL.png
Challenge: Technologies for Data Science at the Intersection of Data Management and Machine Learning

**DM**

- Declarative Languages
- Automatic Adaption
- Scalable processing

**DA**

- Feature Engineering
- Representation
- Algorithms (SVM, EM, etc.)

**Think DA-algorithms in a scalable way**

- declarative

**Process (iterative) algorithms in a scalable way**

aggregation

relational algebra

UDF

iteration/recursion

linear algebra

graph algebrae etc.
Agenda

• Big Data Management

• Apache Flink
  – Data Stream Analysis
  – Iterations
  – The Flink Community

• Further Aspects
  – Fault Tolerance
  – Declarative Languages
Apache Flink – Big Data Batch and Stream Processing

http://flink.apache.org
http://www.stratosphere.eu
**Stratosphere**: General Purpose Programming + Database Execution

- **Draws on Database Technology**
  - Relational Algebra
  - Declarativity
  - Query Optimization
  - Robust Out-of-core

- **Adds**
  - Iterations
  - Advanced Dataflows
  - General APIs
  - Native Streaming

- **Draws on MapReduce Technology**
  - Scalability
  - User-defined Functions
  - Complex Data Types
  - Schema on Read

Draws on Database Technology

Adds

Draws on MapReduce Technology
Timeline

- **2008**: Initial vision for a big data analytics platform.
- **2009**: DFG Proposal for Stratosphere I
- **2010**: Grant Award, Start of Stratosphere I
- **2010**: DFG Proposal for Stratosphere II
- **2011**: StratoSphere Above the Clouds
- **2012**: Grant Award, Start of Stratosphere II
- **2012**: Spinning Fast Iterative Dataflows paper published
- **2012**: The VLDB Journal paper published
- **2014**: APACHE Flink Incubator Project
- **2014**: Flink Top Level Project
- **2015**: Flink formulated on 21 Meetups Worldwide
- **2015**: 186 Contributors
- **2015**: 6326 Members
- **2015**: 20 Cities
- **2015**: 12 Countries
- **2015**: Flink Community Groups Across Europe
- **2015**: Berlin: 758
- **2015**: Paris: 500
- **2015**: Madrid: 384
- **2015**: Stockholm: 313
- **2015**: Brussels: 279
- **2015**: London: 190

© Volker Markl
What is Apache Flink?

Apache Flink is an open source platform for scalable batch and stream data processing.

A distributed system that you can use to process data
  Like a DBMS but not exactly a DBMS

What kind of data?
  Data that comes in the form of streams

What kind of processing
  Quite flexible. You can use Java/Scala APIs similar to programming with Java collections, the new SQL API, etc

Distributed: runs on many (1000s) of machines and hides this complexity from the user

http://flink.apache.org
Basic application architecture

A replayable log of events with pub/sub functionality

Sources of data (e.g., sensors, logs, ...)

Processing of events

app state

Storage and query systems

Query service

By courtesy of Kostas Tzoumas
case class Path (from: Long, to: Long)
val tc = edges.iterate(10) {
  paths: DataSet[Path] =>
  val next = paths
  .join(edges)
  .where("to")
  .equalTo("from") {
    (path, edge) =>
    Path(path.from, edge.to)
  }
  .union(paths)
  .distinct()
  next
}

Pre-flight (Client)

Type extraction stack

Cost-based optimizer

Dataflow Graph

deploy operators

track intermediate results

Memory manager
Out-of-core algos
Batch & Streaming
State & Checkpoints

Workers

Recovery metadata
Task scheduling

Master

© Volker Markl
Effect of optimization

Run on a sample on the laptop

Execution Plan A

Hash vs. Sort
Partition vs. Broadcast
Caching
Reusing partition/sort

Execution Plan B

Run on large files on the cluster

Execution Plan C

Run a month later after the data evolved
Why optimization?

Do you want to hand-tune that?
P. Carbone, A. Katsifodimos, A. Ewen, V. Markl and e. al., "Apache Flink™: Stream and Batch Processing in a Single Engine.,”

DATA STREAM ANALYSIS
Life of data streams

- **Create**: create streams from event sources (machines, databases, logs, sensors, …)

- **Collect**: collect and make streams available for consumption (e.g., Apache Kafka)

- **Process**: process streams, possibly generating derived streams (e.g., Apache Flink)
Stream Analysis in Flink

```
case class Count(symbol: String, count: Int)
val defaultPrice = StockPrice("", 1000)

// Use delta policy to create price change warnings
val priceWarnings = stockStream.groupBy("symbol")
  .window(Delta.of(0.05, priceChange, defaultPrice))
  .mapWindow(sendWarning _)

// Count the number of warnings every half a minute
val warningsPerStock = priceWarnings.map(Count(_, 1))
  .groupBy("symbol")
  .window(Time.of(30, SECONDS))
  .sum("count")
```

Defining windows in Flink

- **Trigger policy**
  - When to trigger the computation on current window
- **Eviction policy**
  - When data points should leave the window
  - Defines window width/size
- **E.g., count-based policy**
  - Evict when \#elements > n
  - Start a new window every n-th element
- **Built-in: Count, Time, Delta policies**
Checkpointing / Recovery

• Flink acknowledges batches of records
  – Less overhead in failure-free case
  – Currently tied to fault tolerant data sources (e.g., Kafka)

• Flink operators can keep state
  – State is checkpointed
  – Checkpointing and record acks go together

• Exactly one semantics for state
Checkpointing / Recovery

Pushes checkpoint barriers through the data flow

Data Stream

After barrier = Not in snapshot
Before barrier = part of the snapshot

(backup till next snapshot)

Chandy-Lamport Algorithm for consistent asynchronous distributed snapshots

Operator checkpoint starting

Checkpoint done

checkpoint in progress

Checkpoint done

Checkpoint done

Checkpoint done

ITERATIONS IN DATA FLOWS
⇒ MACHINE LEARNING ALGORITHMS
Iterate by looping

- for/while loop in client submits one job per iteration step
- Data reuse by caching in memory and/or disk
Iterate in the Dataflow

- **initial solution** → **partial solution** → **X** → **Y** → **partial solution** → **iteration result**

  - **Replace**
  - **Step function**
  - **other datasets**
Large-Scale Machine Learning

Factorizing a matrix with 28 billion ratings for recommendations

Optimizing iterative programs

- Pushing work „out of the loop“
- Caching Loop-invariant Data
- Maintain state as index

STATE IN ITERATIONS
→ GRAPHS AND MACHINE LEARNING
Iterate natively with deltas

- Initial workset
- Partial solution
- Merge deltas
- Replace
- Iteration result
Effect of delta iterations…

# of elements updated

iteration
... very fast graph analysis

Performance competitive with dedicated graph analysis systems

... and mix and match ETL-style and graph analysis in one program

A Benchmark Result

Performed by Yahoo! Engineering, Dec 16, 2015

[...]Storm 0.10.0, 0.11.0-SNAPSHOT and Flink 0.10.1 show sub-second latencies at relatively high throughputs[...]. Spark streaming 1.5.1 supports high throughputs, but at a relatively higher latency.

Flink achieves highest throughput with competitive low latency!


THE FLINK COMMUNITY
The Flink Community:
Meetups By Country Concerning Flink

Apache Flink Meetups Worldwide (Data accurate as of 30.5.16)
6326 members strictly focused on Apache Flink (comprising 57%)
4771 members broader in scope, including Flink (comprising 43%)
Flink community (2)

- More than 250 people have contributed code to Flink

By courtesy of Kostas Tzoumas
Code survival in Flink

By courtesy of Kostas Tzoumas
Powered by Flink

Zalando, one of the largest ecommerce companies in Europe, uses Flink for real-time business process monitoring.

King, the creators of Candy Crush Saga, uses Flink to provide data science teams with real-time analytics.

Alibaba, the world's largest retailer, built a Flink-based system (Blink) to optimize search rankings in real time.

Bouygues Telecom uses Flink for real-time event processing over billions of Kafka messages per day.

By courtesy of Kostas Tzoumas
> 20 Companies Using Flink

Alibaba.com
Pragis Bidoop
ResearchGate
otto group
EURANOVA
RADICALBIT
FIRELAYERS

Bouyges Telecom
Treelogic
ing
mbrtargeting
zalando

Atos
dataArtisans
amadeus
> 8 Software Projects Using Flink

**Google Cloud Platform**

**CLOUD DATAFLOW**
A fully-managed cloud service and programming model for batch and streaming big data processing.

**Apache Flink** is a replacement for MapReduce to support large-scale batch workloads and streaming data flows. It eliminates the concept of mapping and reduces and leverages in-memory storage, resulting in significant performance gains over MapReduce.

**CASCADING**

**Apache SAMOA**
Scalable Advanced Massive Online Analysis

**Apache SAMOA** is a distributed streaming machine learning (ML) framework that contains a programming abstraction for distributed streaming ML algorithms.

**Apache Mahout™**

The Apache Mahout™ project’s goal is to build an environment for quickly creating scalable performant machine learning applications.

**Apache MRQL**
MRQL is a query processing and optimization system for large-scale, distributed data analysis, built on top of Apache Hadoop, Hama, Spark, and Flink.

**Apache Beam**

Apache Beam is an open source, unified programming model that you can use to create a data processing pipeline.
> 10 Research Institutions Using Flink
Flink in the ecosystem

By courtesy of Kostas Tzoumas
Smart Data Web

**Scenario**
How can we detect problems, risks and potential bottlenecks in the supply chain based on the knowledge derived from natural language data?

**Results**
Knowledge graph for supply chain management
- continuously updated with relevant facts and events using parallel analysis of textual data streams
- linking company-internal with open knowledge

BMWi Project “Smart Data Web”: http://www.smartdataweb.de/
Smart Data for Mobility

**Scenario**
How do we aggregate and make use of mobility data from multiple modalities (air, road, railway)?

- **timetables**
- **delays**
- **traffic flow**
- **weather**

Scalable data management (aggregation, enhancement, provisioning)

**Prediction & Optimization**

**Big data analytics**

**Flink**

**Results**
- Big data analytics platform providing data and smart mobility services
- Prediction based on historical and real-time data considering a wide range of mobility-related aspects

BMWi Project “Smart Data For Mobility”: http://www.sd4m.net
SePiA.Pro

Scenario
- Service platform for industrial data analytics services for **production chain optimization**.
- Improvement of production planning and execution
- Cross customer analysis of machines from the perspective of a single machine manufacturer.

Data Sources
- Machinery park
- Sensors
- Actuators
- Streaming

Flink Solution
- Data Science Component
- Industrial Analytics Platform

Output
- Parameters for production chain optimization
- Smart Prediction models
- Cross customer analysis results

Results
- Scalable data analytics platform optimized for industrial settings
- Smart services (data, algorithms, structures, policies) for automatic provisioning

BMWi Project SePiA.Pro
STREAMLINE

Scenario
• Transactions related with user actions on content are considered on the next day for new content recommendations.
• The goal is to do context aware content recommendation in real time.

Data Sources
User actions: download like skip

Flink Solution
Pre-processing data

Output
Real time music recommendation

Results
• Expected increase revenue by 25%
• Reduce time to recommend new content, move from daily batch to real time
• Reduce costs with manual back-office processes such as manually curating content; cost reduction by 60%
• Increase the total number of users consuming (clicking on) recommended content per day; increase by 50%

STREAMLINE, Horizon 2020, Ref: 688191
Scenario

- Defects introduced in early processes of steel production have a great economic impact due to the costs
- The sooner defects are detected, the sooner the process can be modified in order to stop producing defective subsequent coils

Results

- Expected to achieve a reduction of 20% of defect coils and reducing rejected material by 15%.

PROTEUS, Horizon 2020, Ref: 687691
SAGE

**Scenario**
In scientific research areas as vast as brain simulation, gene sequencing, and space weather forecasting, experiments and simulations generate increasingly large data sets. As these scale into the range of exabytes (billions of gigabytes), novel storage, processing, and analytics solutions must be devised to continue deriving insights and innovation in research.

**Results**
Development of a next-generation data storage system supporting current- and future-generation persistent storage media for exascale data processing. Project SAGE will integrate both current- and future-generation storage media within a multi-tiered hierarchy and introduce computational capabilities to the storage layer.
Thanks to my team members and students

- Dr. Stephan Ewen
- Dr. Sebastian Schelter
- Dr. Kostas Tzoumas
- Dr. Asterios Katsifodimos
- Fabian Hüske
- Alexander Alexandrov
- Max Heimel
- Juan Soto

and many more members of the Stratosphere Project, the Berlin Big Data Center, and the Apache Flink community
Evolution of Big Data Platforms

First Generation
Data Warehouses, e.g., relational DBMS

Second Generation
Scale-out, Map/Reduce, UDFs, e.g., Apache Hadoop

Third Generation
In-memory Performance and Improved Programming Model, e.g. Apache Spark

Fourth Generation
In-memory + Out of Core Performance, Declarativity, Optimization of Iterative Algorithms, True Streaming e.g. Apache Flink
Fault tolerance

Pessimistic Recovery:
- Write intermediate state to stable storage
- Restart from such a checkpoint in case of a failure

Problematic:
- High overhead, checkpoint must be replicated to other machines
- Overhead always incurred, even if no failures happen!

➢ How can we avoid this overhead in failure-free cases
Optimistic recovery

- Many data mining algorithms are **fixpoint algorithms**
- **Optimistic Recovery**: jump to a different state in case of a failure, still converge to solution
  - No checkpoints → **No overhead in absense of failures**!
  - algorithm-specific **compensation function** must restore state
All Roads lead to Rome

If you are interested more, read our CIKM 2013 paper:

Sebastian Schelter, Stephan Ewen, Kostas Tzoumas, Volker Markl: "All roads lead to Rome": optimistic recovery for distributed iterative data processing. CIKM 2013: 1919-1928

Declarative Data Processing and Big Data
A Billion $$$ Mantra...

Declarative Data Processing

An effective, formal foundation based on relational algebra and calculus (Codd ’71).
A simple, high-level language for querying data (Chamberlin ’74).
An efficient, low-level execution environment tailored towards the data (Selinger ’79).
With 40+ years of success...

Declarative Data Processing
Is Being Revised

Declarative Data Processing

SQL

Relations

RDBMS

Second-Order Functions

Distributed Collections

Parallel Dataflow Engines
Mosaics of Theories and Systems

• First results
  – Alexander Alexandrov, Andreas Salzmann, Georgi Krastev, Asterios Katsifodimos, Volker Markl: Emma in Action: Declarative Dataflows for Scalable Data Analysis. SIGMOD 2016

• Next Steps
  – Open-Source Release

• Vision (Frontend): Multi-model DSL based on type contracts
  – Collection Processing \( DataBag[A] \)
  – Linear Algebra \( Matrix[A], Vector[A] \)
  – Stream Processing \( Stream[A] \)

• Vision (Backend): Target more execution engines
  – Column Stores
  – GPUs