Big Data: Challenges and Some Solutions
Stratosphere, Apache Flink, and Beyond

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
Thanks to my team members and students

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

and many more members of the Stratosphere Project, the Berlin Big Data Center, and the Apache Flink community
Data & Analysis: Increasingly Complex!

- **Data**: Volume, Velocity, Variability, Veracity
- **Analysis**: Reporting, Ad-Hoc Queries, ETL/ELT, Data Mining, Predictive/Prescriptive

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

- Algorithms
- ML
- MATLAB, R, Python

- Scalability
“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

Data
Science

Data
Analysis

Scalable
Data
Management

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

New Technology to the Rescue!
A Zoo of Technologies!

http://mattturck.com/wp-content/uploads/2016/03/Big-Data-Landscape-2016-v18-FINAL.png
Big Data Analytics Requires Systems Programming

“Big Data’s Big Problem: Little Talent”
Wall Street Journal
Neelie Kroes (ICT 2013, Nov. 7, Vilnius)

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

Declarative languages to the rescue!
Deep Analysis of „Big Data“ is Key!

Many new companies and products are emerging to enable deep big data analysis; strong European contenders include Apache Flink, Parstream, and Exasol. „New companies“ are the (b)leading users of these technologies, e.g., in the information economy (e.g., Zalando, Amazon, Researchgate, Soundcloud, Spotify). „Traditional Big companies“ are following and still determining strategies (Industrie 4.0, Logistics, Telco, etc.). Most SMEs are not ready yet to capitalize on Big Data.
Machine Learning + Data Management = X

Technology X

Think ML-algorithms in a scalable way

declarative

Process iterative algorithms in a scalable way

Goal: Data Analysis without System Programming!

Mathematical Programming
Linear Algebra
Error Estimation
Active Sampling
Regression Monte Carlo

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

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

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

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

• On Stratosphere and Flink
  – Conception, Initial Contributions
  – Streaming Data Analysis
  – The Flink Community

• On Roman Generals and Big Data Analytics

• On Emma and Mosaics
Apache Flink
# Stratosphere: General Purpose Programming + Database Execution

<table>
<thead>
<tr>
<th>Draws on Database Technology</th>
<th>Adds</th>
<th>Draws on MapReduce Technology</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Relational Algebra</td>
<td>• Iterations</td>
<td>• Scalability</td>
</tr>
<tr>
<td>• Declarativity</td>
<td>• Advanced Dataflows</td>
<td>• User-defined Functions</td>
</tr>
<tr>
<td>• Query Optimization</td>
<td>• General APIs</td>
<td>• Complex Data Types</td>
</tr>
<tr>
<td>• Robust Out-of-core</td>
<td>• Native Streaming</td>
<td>• Schema on Read</td>
</tr>
</tbody>
</table>


What is Apache Flink?

Apache Flink® is an open-source stream processing framework for distributed, high-performing, always-available, and accurate data streaming applications.

- **Key Features:**
  - Bounded and unbounded data
  - Event time semantics
  - Stateful and fault-tolerant
  - Running on thousands of nodes with very good throughput and latency
  - Exactly-once semantics for stateful computations.
  - Flexible windowing based on time, count, or sessions in addition to data-driven windows

- **DataSet** and **DataStream** programming abstractions are the foundation for user programs and higher layers

http://flink.apache.org
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
}

Program

Technology inside Flink

Dataflow Graph

Pre-flight (Client)

Type extraction stack

Cost-based optimizer

Workers

deploy operators

track intermediate results

Memory manager
Out-of-core algos

Batch & Streaming
State & Checkpoints

Recovery metadata
Task scheduling

Workers

Cost-based optimizer

Effect of optimization

Execution Plan A
- Run on a sample on the laptop
- 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
- Execution Plan C
Why optimization?

Do you want to hand-tune that?

F. Hueske, M. Peters, A. Krettek, M. Ringwald, K. Tzoumas, V. Markl, J.C. Freytag: Peeking into the optimization of data flow programs with MapReduce-style UDFs. ICDE 2013: 1292-1295
S. Ewen, K. Tzoumas, M. Kaufmann, V. Markl:

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
- Other datasets
- Step function
- Replace
- Partial solution
- Iteration result
- Solution

Equation: $X,Y$
Large-Scale Machine Learning

Factorizing a matrix with 28 billion ratings for recommendations

(Scale of Netflix or Spotify)

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**
- **workset**
- **partial solution**
- **initial solution**
- **other datasets**
- **Merge deltas**
- **Replace**
- **iteration result**
- **workset**
- **solution**
- **delta set**

- **A**
- **B**
- **X**
- **Y**
Effect of delta iterations...
… very fast graph analysis

Performance competitive with dedicated graph analysis systems

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

“STREAMING DATA” ANALYSIS
Bounded and unbounded data

• Bounded data: a dataset with a natural beginning and end
  – E.g., current position of all trucks in a fleet, content of a data warehouse

• Unbounded data: a dataset without a natural beginning and end
  – E.g., customers of our products, tweets about our product

• Note: few data is bounded by nature; most bounded data is a view over unbounded data

By courtesy of Kostas Tzoumas
Stream and batch processing

• **Stream processing**: continuous processing that continuously produces results
  – E.g., a Java program that connects to a socket and parses the socket contents
  – Apache Flink, Apache Storm

• **Batch processing**: processing that takes finite time to complete and produces results only in the end
  – E.g., sorting a file
  – Apache Hadoop MapReduce, Apache Spark

By courtesy of Kostas Tzoumas
How do they fit together

• Batch processing over bounded data
  – Natural

• Stream processing over bounded data
  – Treats bounded data as subset of stream

• Batch processing over unbounded data
  – Needs a pre-processing stream processing phase that splits stream into chunks

• Stream processing over unbounded data
  – Natural

By courtesy of Kostas Tzoumas
A different view

• What changes faster? Your code or your data?

• $\text{ddata/dt} \gg \text{dcode/dt}$ is a data streaming problem

• $\text{dcode/dt} \gg \text{ddata/dt}$ is a data exploration problem (and likely to become a data streaming problem later)

Credits Joe Hellerstein
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
Some Benchmark Results

Initially 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.


THE FLINK COMMUNITY
Flink Community as of Today

- Members: 18,884
- Contributors: 313
- Meetups: 41
- Cities: 30
- Countries: 15

Contributors over time:

- Feb 09: 0
- Jun 10: 0
- Nov 11: 0
- Mrz 13: 0
- Jul 14: 0
- Dez 15: 0
- Apr 17: 350

Geographical distribution:

- USA: 10,659 members
- Europe: various cities and countries with different member counts
Some Highly Engaged Users

Largest job has > 20 operators, runs on > 5000 vCores in 1000-node cluster, processes millions of events per second

Complex jobs of > 30 operators running 24/7, processing 30 billion events daily, maintaining state of 100s of GB with exactly-once guarantees

30 Flink applications in production for more than one year. 10 billion events (2TB) processed daily

By courtesy of Kostas Tzoumas
Other Companies in the Flink Community

- otto group
- AMadeus
- Uber
- Zalando
- Pragsis Bidoop
- MediaMath
- Netflix
- Capital One
- Parallel Machines
- Lightbend
- ING-DiBa
- BetterCloud
- RadicalBit
- Line
- ResearchGate
- MUX
- Atos
- Ericsson
- Treelogic
- Euro Nova
- FireLayers

https://flink.apache.org/poweredby.html
Flink in the ecosystem

By courtesy of Kostas Tzoumas
Timeline of Flink

- **2008**: Initial vision for a big data analytics platform
- **2009**: DFG Proposal for Stratosphere I
- **2009**: APACHE Flink Incubator Project
- **2010**: Grant Award Start of Stratosphere I
- **2010**: Spinning Fast Iterative Dataflows paper published
- **2011**: dataArtisans Founded
- **2012**: Stratosphere System paper published
- **2012**: VLDB
- **2014**: The VLDB Journal
- **2015**: 1st Flink Forward Conference in Berlin
- **2015**: Flink Forward 2nd Flink Forward Conference in San Francisco
- **2015**: > 30 Companies using Flink
- **2016**: Flink 1.0
- **2017**: FlinkForward Conference in Berlin
- **2018**: Premier FlinkForward Conference in San Francisco
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

---

S. Schelter, S. Ewen, K. Tzoumas, V. Markl: "All roads lead to Rome": optimistic recovery for distributed iterative data processing. CIKM 2013
Declarative Data Processing and Mosaics
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

© Volker Markl
Fine grained parallelism through deep language embedding (EMMA)

- DataBag
- Matrix
- Stream

Emma Source (backed by Scala AST)

Programming Abstractions

Common Concrete Syntax
(realized as Scala eDSL)

(i) Lifting

Emma Core (backed by Scala AST)

Common Abstract Syntax

e.g. Spark, Flink, MM Engine

(iii) Lowering

Target Language + Runtime

Mosaics

Frontend:

Backends:
Research

1. Unifying Modelling Across Theories
2. Cross Theory Optimization

3. Optimizing Across Engines
4. Predicting and Learning Program Runtimes

5. Optimizing Across Hardware
6. Generating Hardware-Targeted Code
The Five Dimensions of Big Data

- **Law**
  - Ownership
  - Copyright/IPR
  - Liability
  - Insolvency
  - Privacy

- **Society**
  - User Behaviour
  - Social Processes
  - Collaboration
  - Political Processes
  - Public Good

- **Economy**
  - Business Models
  - Benchmarking
  - Open Source & Open Data
  - Deployment Models
  - Information Pricing
  - Information Marketplaces

- **Technology**
  - Data Management
  - Data Processing
  - Statistics/ML
  - Linguistics/Text&Speech
  - HCI/Visualization
  - Novel Computer Architectures
  - Security

- **Application**
  - Systems
  - Frameworks
  - Skills
  - Best-Practices
  - Tools

- **Economy & Business**
  - Public Sector
  - Healthcare
  - Transportation
  - Humanities
  - Sciences

- **Data Management**

- **Data Processing**

- **Statistics/ML**

- **Linguistics/Text&Speech**

- **HCI/Visualization**

- **Novel Computer Architectures**

- **Security**

- **Business Models**

- **Benchmarking**

- **Open Source & Open Data**

- **Deployment Models**

- **Information Pricing**

- **Information Marketplaces**
I would like to thank the **members of the Stratosphere** project, the **Berlin Big Data Center**, and my **national and international collaborators**. In particular, I would like to thank my **former and current students and postdocs** at DIMA/TU Berlin and DFKI. Without them it would not have been possible to realize the visions into concrete software system artifacts. Last, but not least, I would like to thank **all of the contributors and users in the Apache Flink community**, without them the Stratosphere project and the correlated research of the Berlin Big Data Center would not have achieved the worldwide impact that we have experienced over the last few years.