Selbst-Adaptive Big Data Architekturen als Grundlage für Ressourcen-Optimale Verarbeitung

Klaus Schmid, Holger Eichelberger
{schmid, eichelberger}@sse.uni-hildesheim.de
Self-adaptive Big Data Architectures

• Big Data
  – Processing of large and complex data sets
  – Too difficult for traditional data processing applications
  – 3V: Volume, Velocity, Volatility

• Problem:
  – Volatile stream characteristics (several orders of magnitude)
  – Soft real-time processing
  – Limited resources / Scale-out not possible

• Goal: Sustain quality of data analysis
  – Adaptive processing
  – Lightweight
Motivation

Application Instance: FP7 QualiMaster

Risk identification in financial markets
- Interconnected markets
- Regular risk analysis requested by EU / US law
- Licensed data
- Bursty data streams
  - Financial data
  - Social web

Always optimal processing → too much HW $$$
Adaptive Systems (MAPE-K)

EASy-Producer
→ Tool for Product Lines and Adaptive Systems

Supports
• Variability / Adaptation space modeling
• Constraint analysis
• Derivation of consequence
• Complex instantiation process
Data Analysis Pipeline

Financial source → Financial preprocessing → Correlation computation → Dynamic Graph Compilation

Twitter source → Spam filtering → Event detection → Focus recommender

Result sink
Algorithm Family

- Idea: Exchange algorithms
  - Same functionality
  - Different runtime characteristics
Selbst-Adaptive Big Data Architekturen
für Resourcen-Optimale Verarbeitung

Adaptive System Architecture

QualiMaster Architecture

End-user application

User triggers

Commands, State

Data Management

Coordination

Monitoring / Analysis

Adaptation

Execution Systems

Pipelines

Storm

Hardware

Hadoop

Cloud Scale-Out

Execution Systems

Pipelines

Storm

Hardware

Hadoop

SSE, University of Hildesheim

GI FG Architekturen, 30.06.2016
## Adaptation Mechanisms

Scoped by model of adaptation space / adaptation script

<table>
<thead>
<tr>
<th>Mechanism</th>
<th>QualiMaster / Stream Processing</th>
</tr>
</thead>
</table>
| Exchange of components        | • Aim: Stream transparency  
                                | • Triggered by constraints    
                                | • Upcoming: Decision by performance profile                               |
| Change of parameters          | • Triggered by algorithms   
                                | • Triggered by user           
                                | • Upcoming: Decision by performance profile                               |
| Re-parallelization / migration| • Storm: Rebalance           
                                | • Storm extension             |

- **Re-parallelization / migration**
  - Storm: Rebalance
  - Storm extension

- **Change of parameters**
  - Triggered by algorithms
  - Triggered by user
  - Upcoming: Decision by performance profile
  - Upcoming: Source volume prediction
  - Last resort: Load shedding

- **Exchange of components**
  - Aim: Stream transparency
  - Triggered by constraints
  - Upcoming: Decision by performance profile

- **Run-time Adaptation Mechanisms**
  - Re-parallelization / migration
  - Migration

- **Aim:** Stream transparency

- **Triggered by constraints**

- **Upcoming:** Decision by performance profile

- **Triggered by algorithms**

- **Triggered by user**

- **Upcoming:** Source volume prediction

- **Last resort:** Load shedding
Lessons learned

- Developing adaptive code is complex
- Storm: Good foundation for distributed stream processing
  - Stable installation not trivial
  - Testing is tricky and time consuming
  - Monitoring aggregates too much (but extensible)
  - Small bugs lead to large effects
  - Does not support adaptation
- Technology is developing fast
  - Twitter Heron
  - Apache Spark
  - Supporting frameworks

Documentation!

Model-based development!
Approach

Product-Line based Approach

- Domain-Specific Modeler
- Domain / Variability Model
- Code Generation
- Domain-Specific Infrastructure

-Producer
Domain-specific configuration
Results

- Topological configuration
- Several pipelines generated: 5 demo + 4 test pipelines
- Validation: <250 ms
- Instantiation
  - 4 minutes
  - 30 KLOC in 195 artifacts
  - Deployable artifact: 40 - 150MBytes
  - Integration of algorithms
  - Integration of adaptation mechanisms / monitoring
- Voice of the “user”
  - Clear separation of algorithm/pipelines
  - Generate more
Summary / Results

- Resource optimization requires processing alternatives
- Volatile Big Data requires adaptive processing
- Generative approaches can successfully
  - Create major parts of technical code (30KLOC, 195 artifacts)
  - Integrate complex runtime mechanisms (<110 ms)
  - Create deployable artifacts (40-150MBytes)
  - Relieve Data Analysts from technical work

Output becomes bottleneck!

1684 ticks/s → 1.4M correlations

Project homepage: [http://www.qualimaster.eu](http://www.qualimaster.eu)
Twitter: [@QualiMasterEU](https://twitter.com/QualiMasterEU)

The research leading to these results has received funding from the European Union Seventh Framework Programme [FP7/2007-2013] under grant agreement n° 619525 (QualiMaster).