Zürcher Hochschule
für Angewandte Wissenschaften

School of
Engineering

InIT Institut für angewandte
Informationstechnologie

Data-Driven Financial Risk Modeling at Scale with Apache Spark

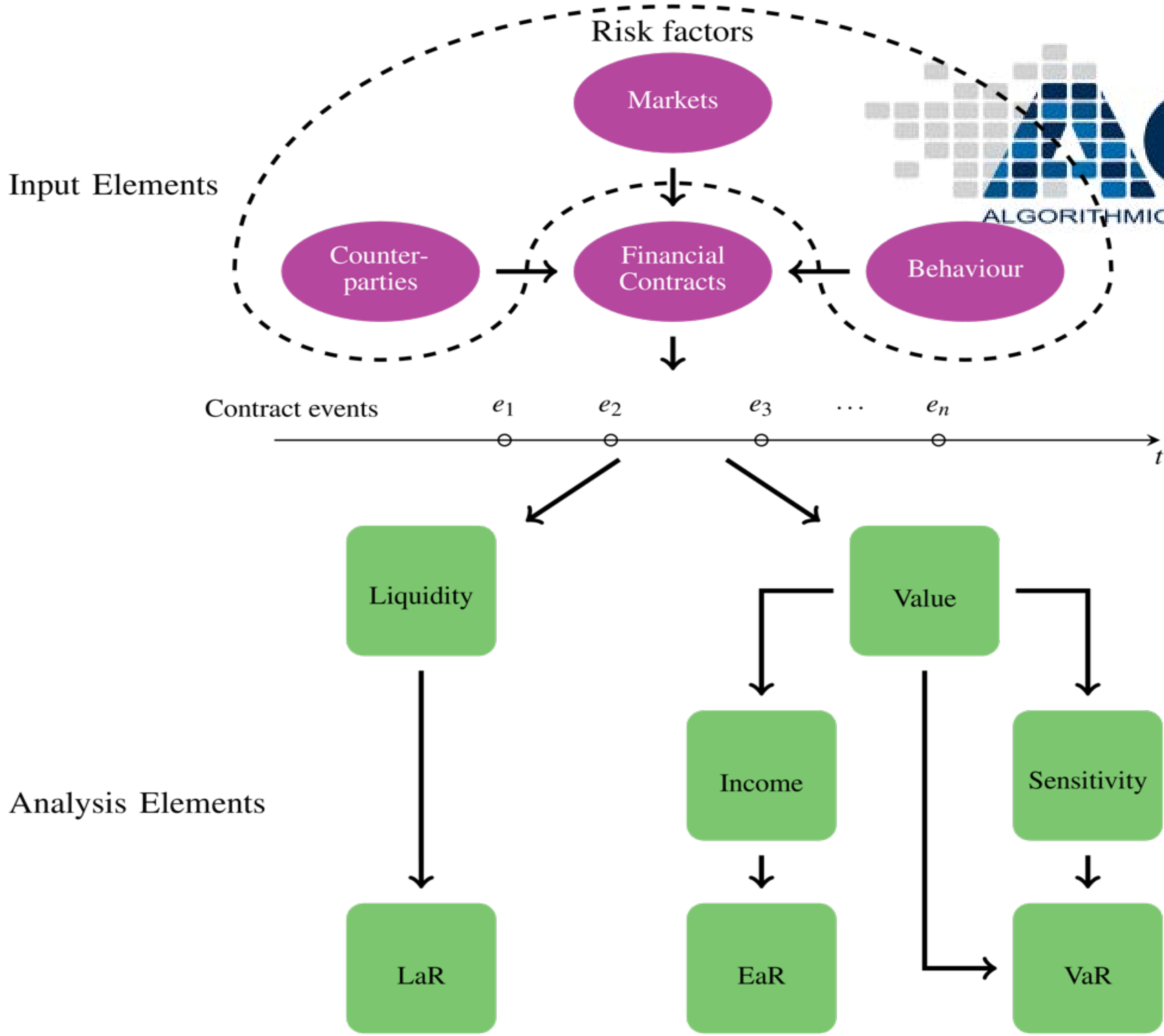
Prof. Dr. Kurt Stockinger

(joint work with Nils Bundi, Wolfgang Breymann and Jons Heitz)

Zurich University of Applied Sciences

Artificial Intelligence in Industry and Finance
Winterthur, September 6, 2018

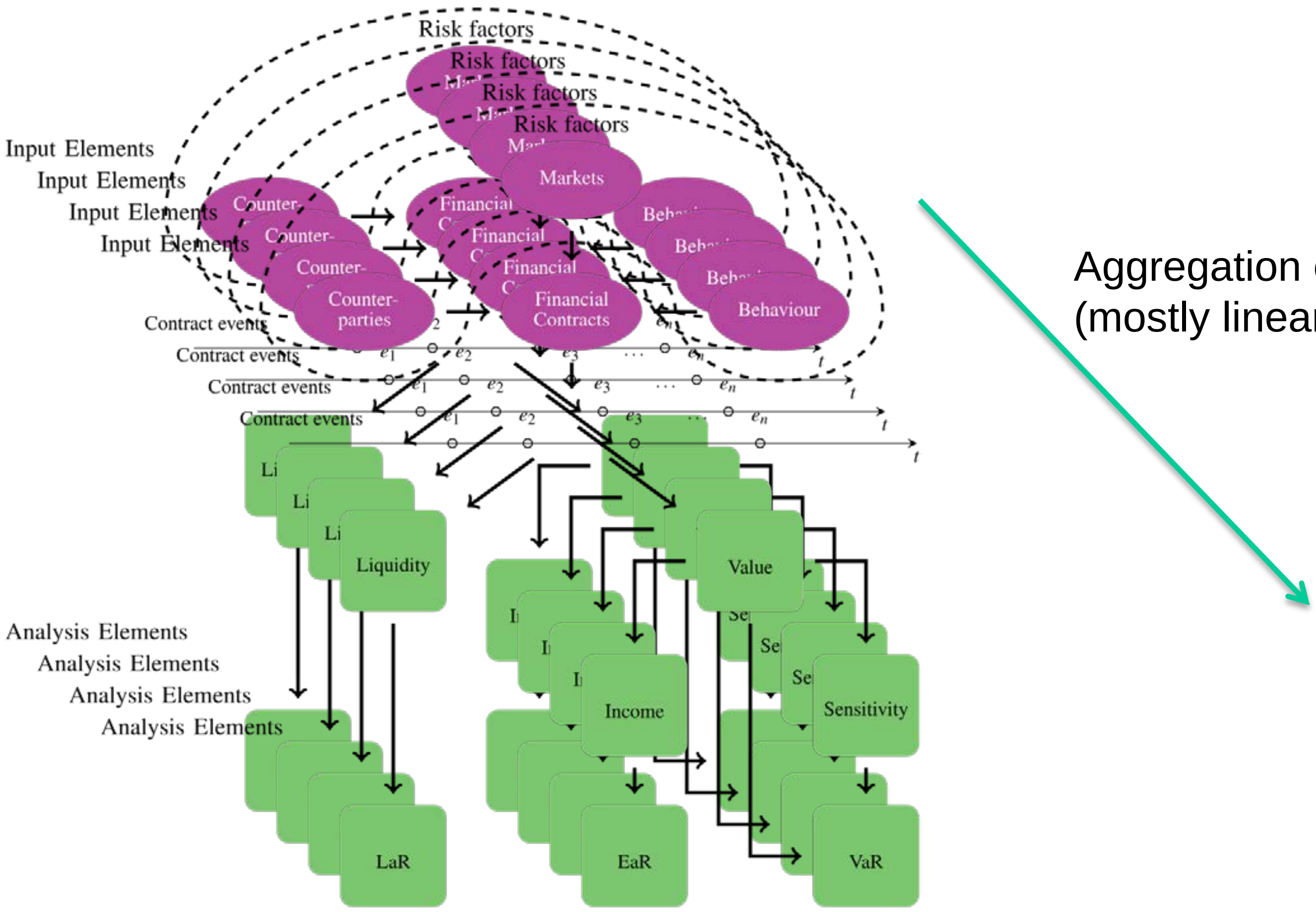
DatFRisMo: Data-Driven Financial Risk Modeling



ACTUS
ALGORITHMIC CONTRACT TYPES UNIFIED STANDARDS

Brammertz, Akkizidis, Breymann, Entin, Rustmann, *Unified Financial Analysis*. Wiley, Chichester, 2009.

An ACTUS Portfolio



Aggregation over contracts (mostly linear operations)

How Can we Solve this Challenge?

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- **Big Data Problem:**
  - Large amounts of contract events (generated cash flows)

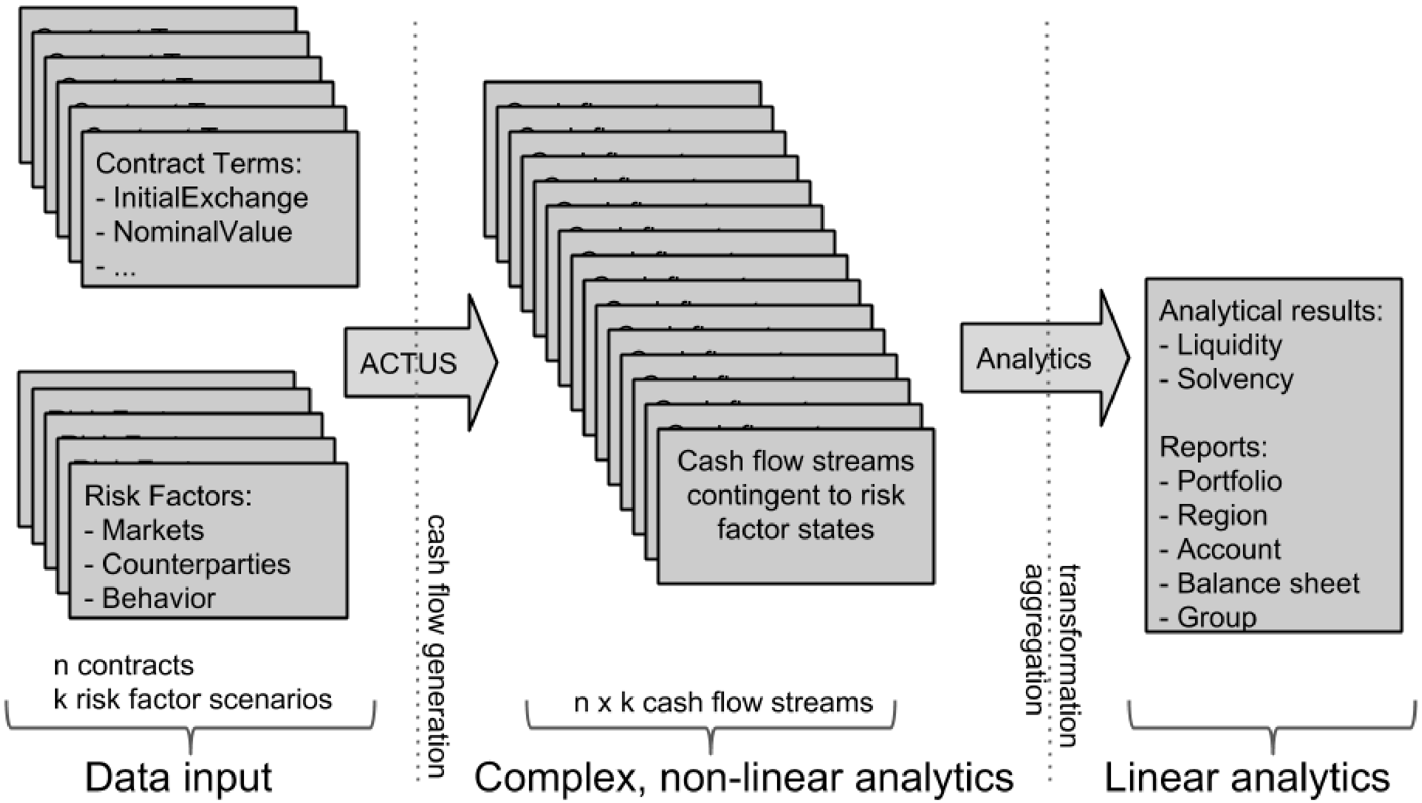
- **Big Computation Problem:**
  - Large-scale Monte-Carlo simulation (risk factors)

Main Research Questions

- Question 1: Can we easily **parallelize** existing financial kernels?
- Question 2: Can financial calculations be **formulated in SQL** and thus be **accelerated** by taking advantage of a SQL Query Optimizer?
- Question 3: What is the **scalability** of running large-scale, real-world financial analytics?

Data Flows in Actus

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie



Zürcher Hochschule
für Angewandte Wissenschaften
zh
aw
School of
Engineering
InIT Institut für angewandte
Informationstechnologie

- **Nominal value:**
  - Measures the (current) notional outstanding of, e.g., a loan
  - Provides basis for exposure calculations in credit- risk departments
- **Fair value:**
  - Quantifies the price of a contract that could be realized in a market transaction at current market conditions
- **Liquidity:**
  - Expected net liquidity flows over some future time periods

Basic measurements necessary for analyzing and managing different types of financial risks

Financial Analytics – More Formal

- **Nominal value:** is $N_i^k = n_i^k(t_0)$ $n_i^k(t_0)$ current notional outstanding

- **Fair value:** $V_i^k = \sum_{t \in T_i^k} d_i^k(t) f_i^k(t)$ $d_i^k(t)$ cash flow, $f_i^k(t)$ discount factor

- **Liquidity:** $L_i^k = (l_i^k(\delta_1), l_i^k(\delta_2), \ldots)$

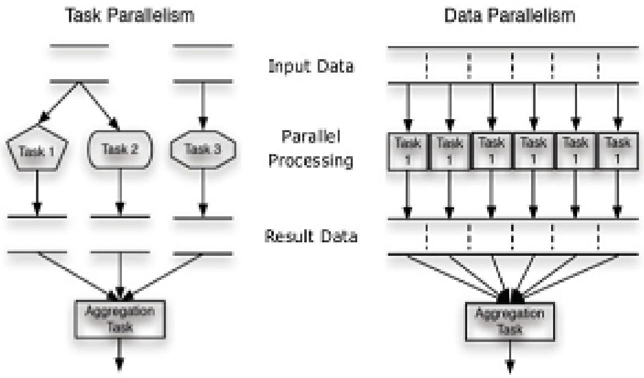
  with

  $$l_i^k(\delta_u) = \sum_{t \in (t_0+\delta_{u-1}, t_0+\delta_u)} f_i^k(t)$$

  $\Delta = \{\delta_1, \delta_2, \ldots, \delta_u, \ldots\}$ time periods

Different Types of Parallelism

- **Task parallelism:**
  - Task is split into subtasks
  - Each subtask is executed on different node of computer cluster



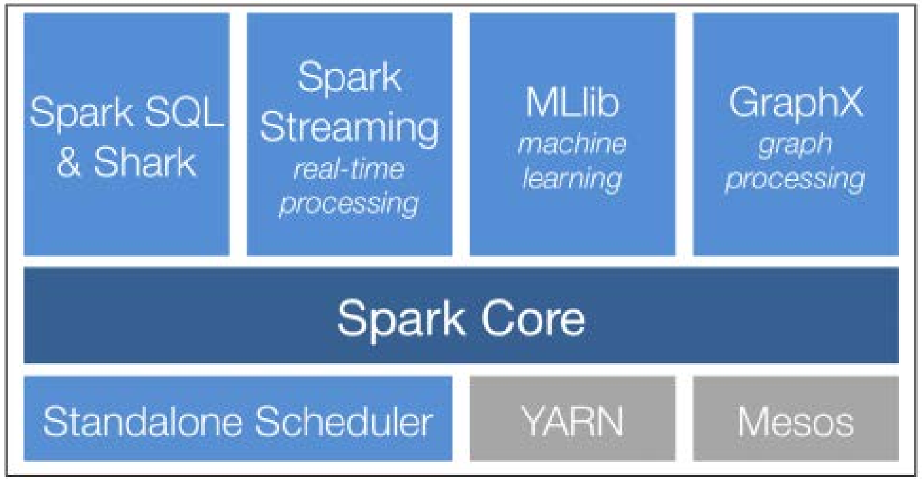
- **Data parallelism:**
  - Data is distributed onto nodes of computer cluster
  - Each node executes some task on different part of data

Financial analytics is an **embarrassingly parallel** problem that can be solved with **data parallelism**

Use Apache Spark Big Data Technology

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- General purpose **cluster computing system**
- Originally **developed at UC Berkeley**, now one of the largest **Apache** projects
- Typically faster than Hadoop due to **main-memory processing**
- High-level APIs in **Java, Scala, Python** and **R**
- **Functionality** for:
  - Map/Reduce
  - SQL processing
  - Real-time stream processing
  - Machine learning
  - Graph processing



User Defined Functions vs. SQL in Spark

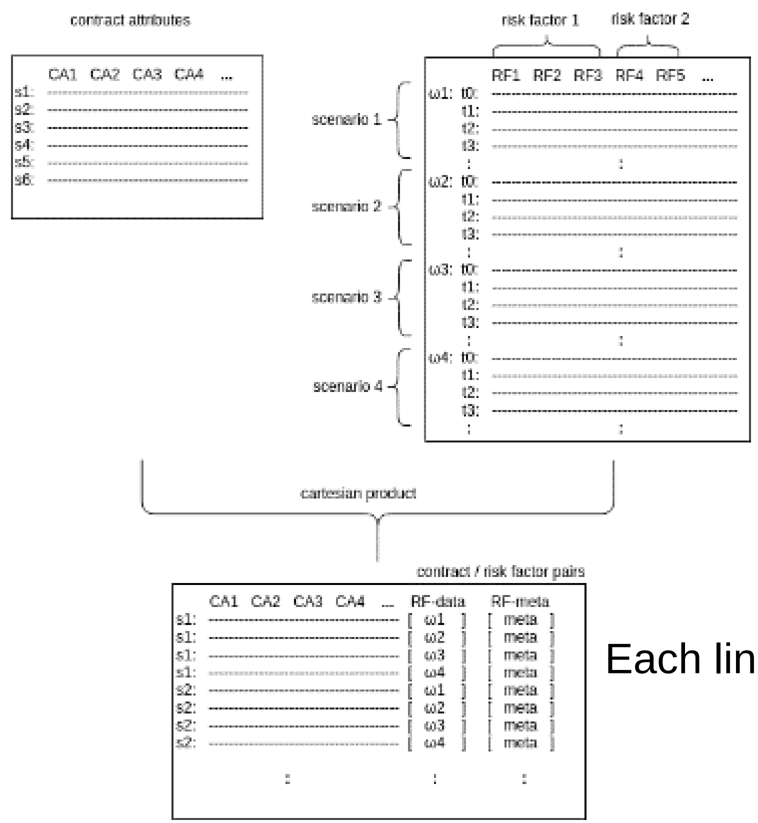
- **User defined function:**
  - Function provided by user (can be any piece of code)

- **SQL:**
  - SQL statement provided by user

- Spark can execute both UDFs and SQL in parallel
- However, **UDFs** are more of a **black box** while **SQL** queries can be accelerated by **SQL Optimizer** (similar to parallel relational databases)
- **Trade-off** between leveraging existing code or re-writing in SQL

Major Data Structure

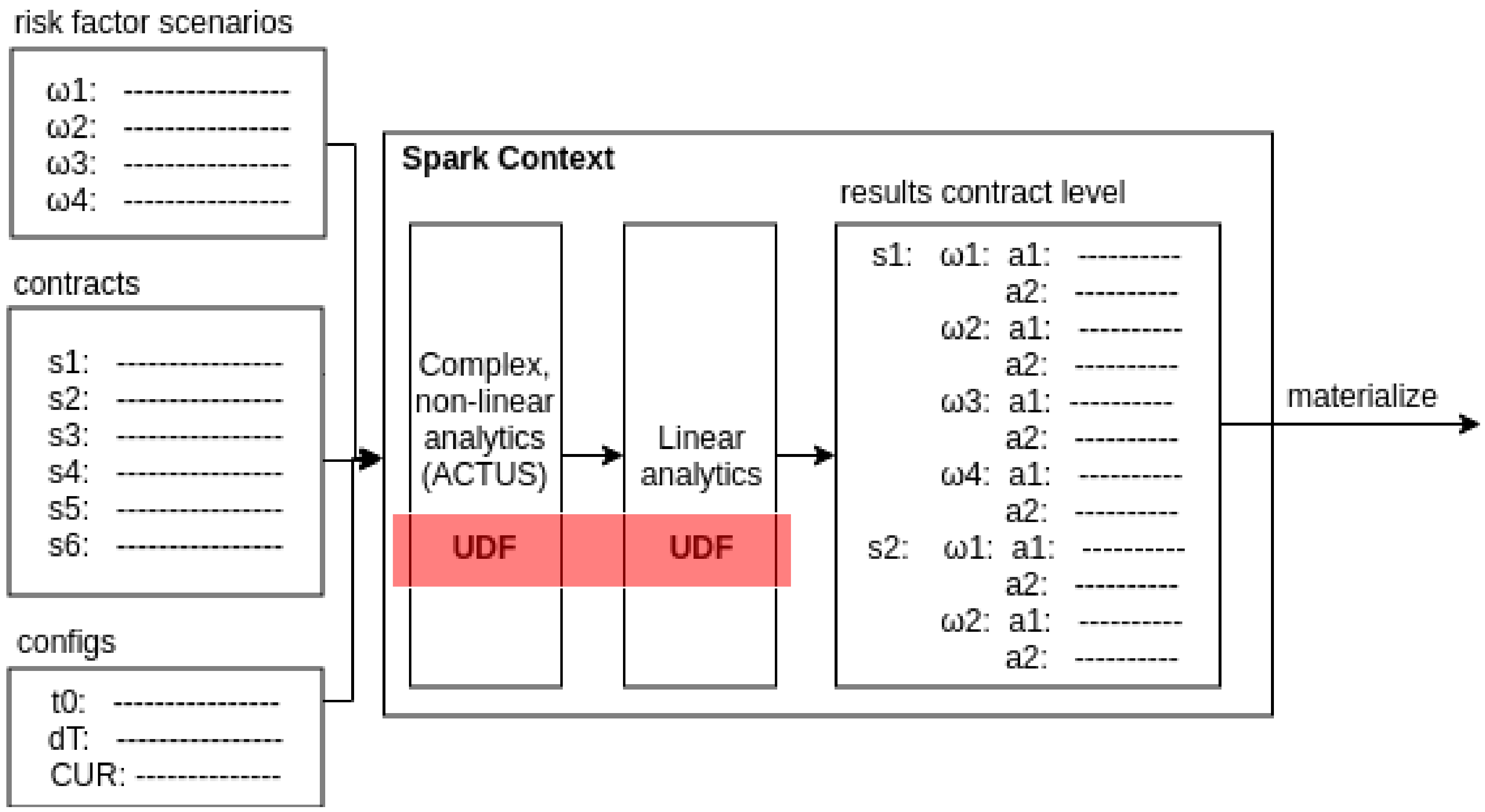
Need a data structure that enables data parallelism based on Spark **DataSet**



Each line can be executed in parallel

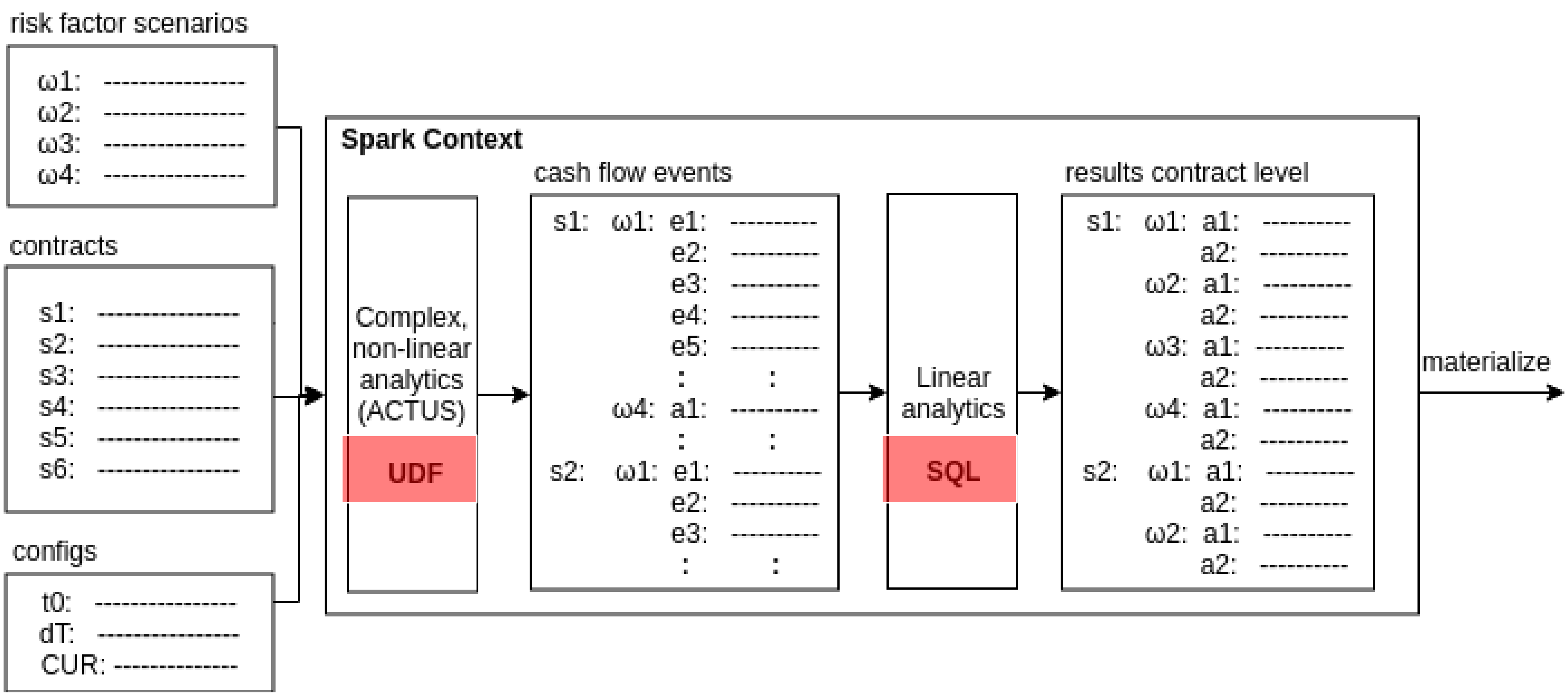
On-the-Fly: Spark-UDF for Non-Linear and Linear Analytics

The whole code is executed as a **user defined function** in Spark



On-the-Fly: Spark-UDF for Non-Linear and Spark-SQL for Linear Analytics

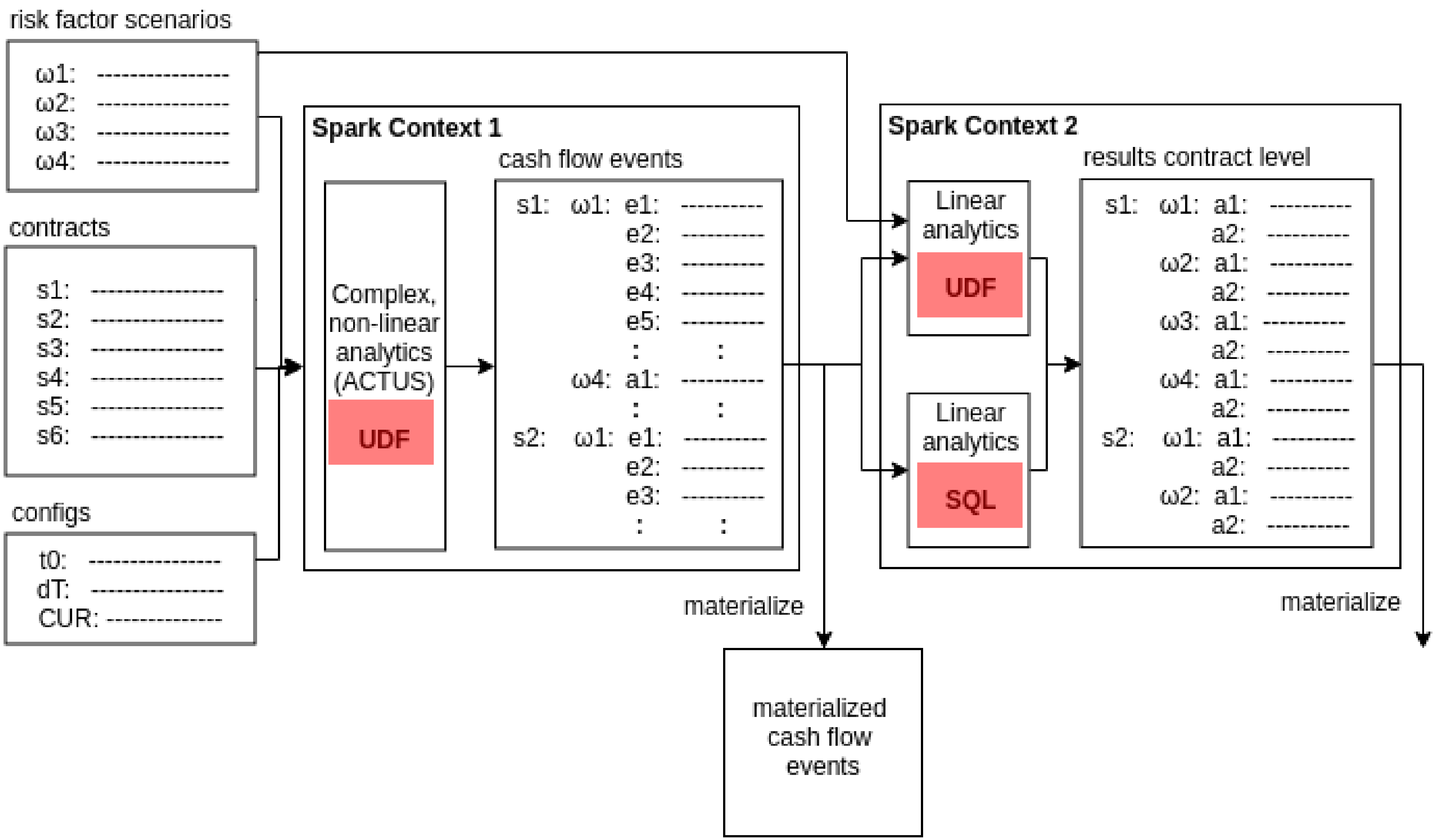
Linear analytics are **rewritten and executed in SQL**



Materialized: Spark-UDF or SQL for Linear Analytics

Zürcher Hochschule für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Cash flow results are materialized



Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

Experimental Environment

- **Software:**
  - ACTUS implemented in Java
  - Apache Spark 2.3 running on Amazon Web Services
  - 96 million financial contracts
  - 1,000 risk factor model
- **Hardware:**
  - Up to 32 machines with 30 GB RAM, 16 vCPUs at 2.5 GHz each
  - Total:
    - 960 GB of distributed RAM
    - 512 vCPU cores

Generate and Count Cash Flows

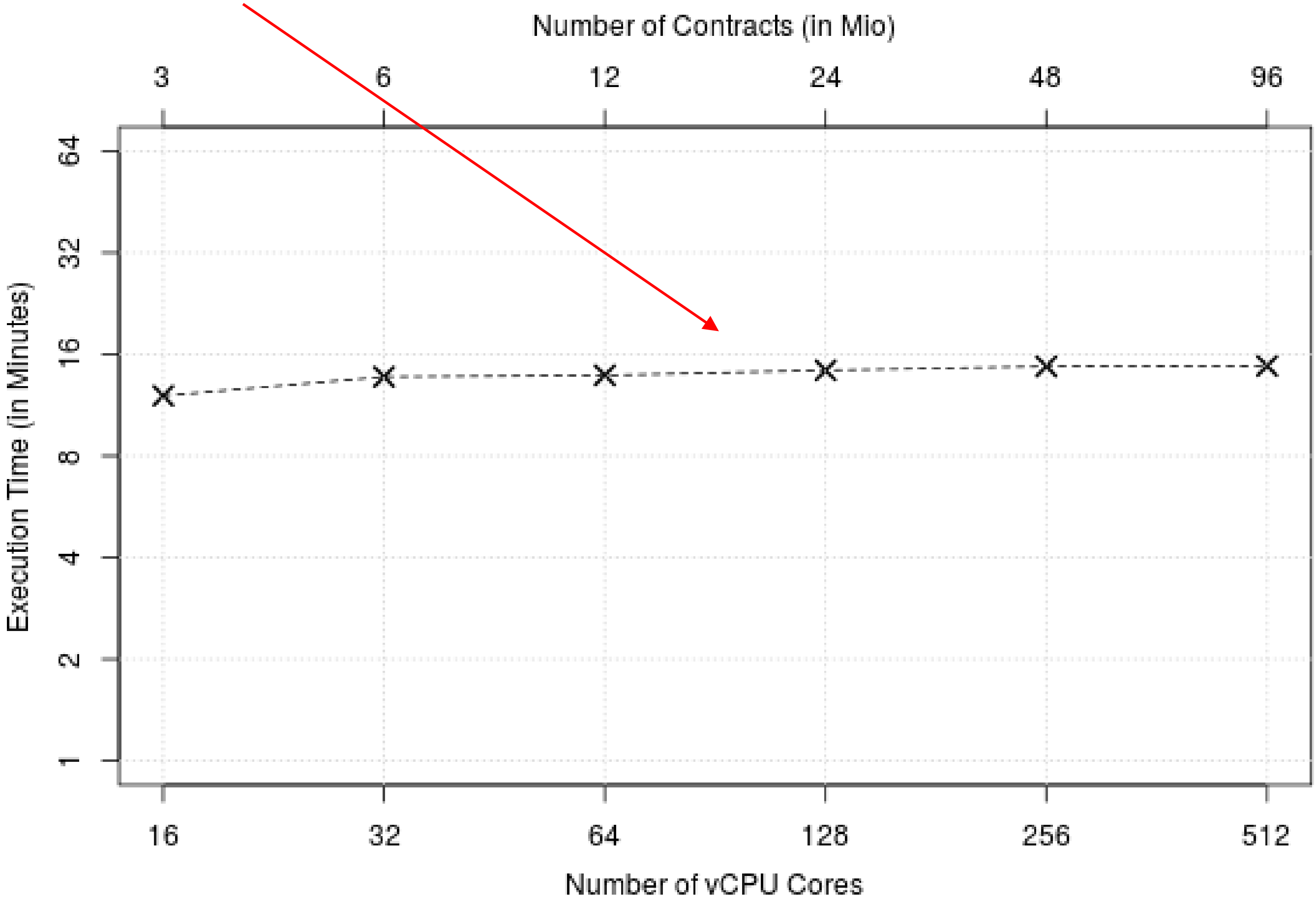
Zürcher Hochschule
für Angewandte Wissenschaften

zh
aw

School of Engineering

InIT Institut für angewandte Informationstechnologie

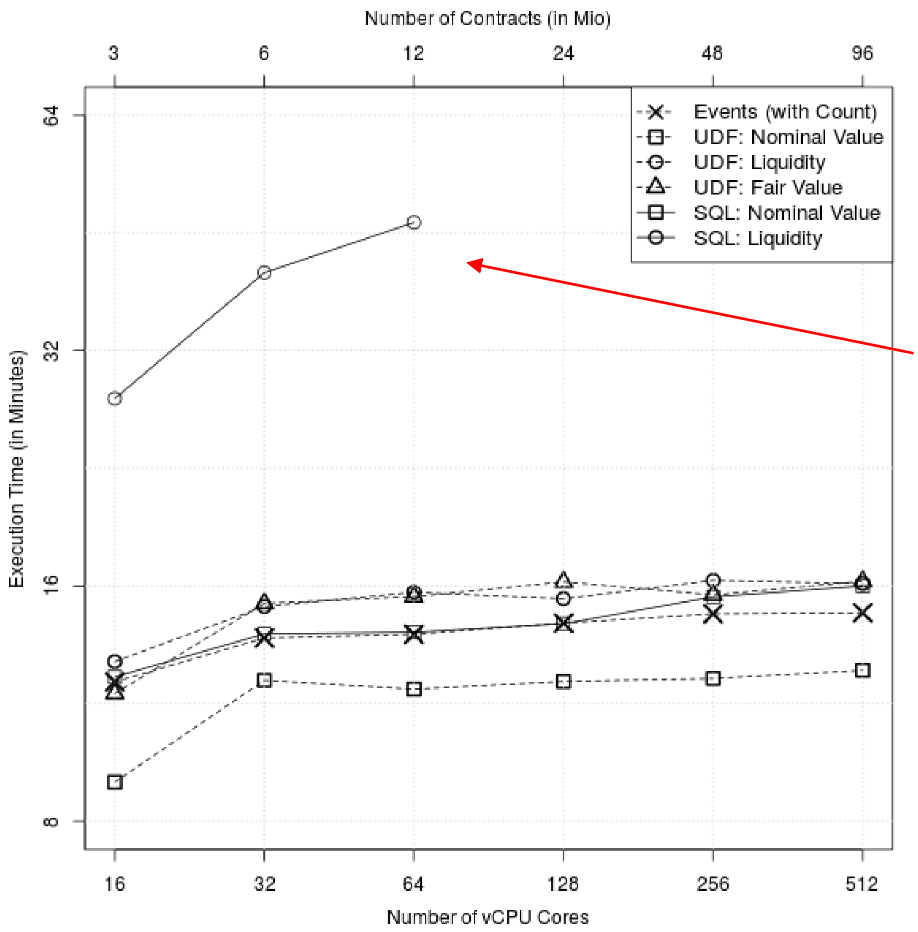
Close to linear scalability



Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

UDF and SQL Analytics – On-the-Fly

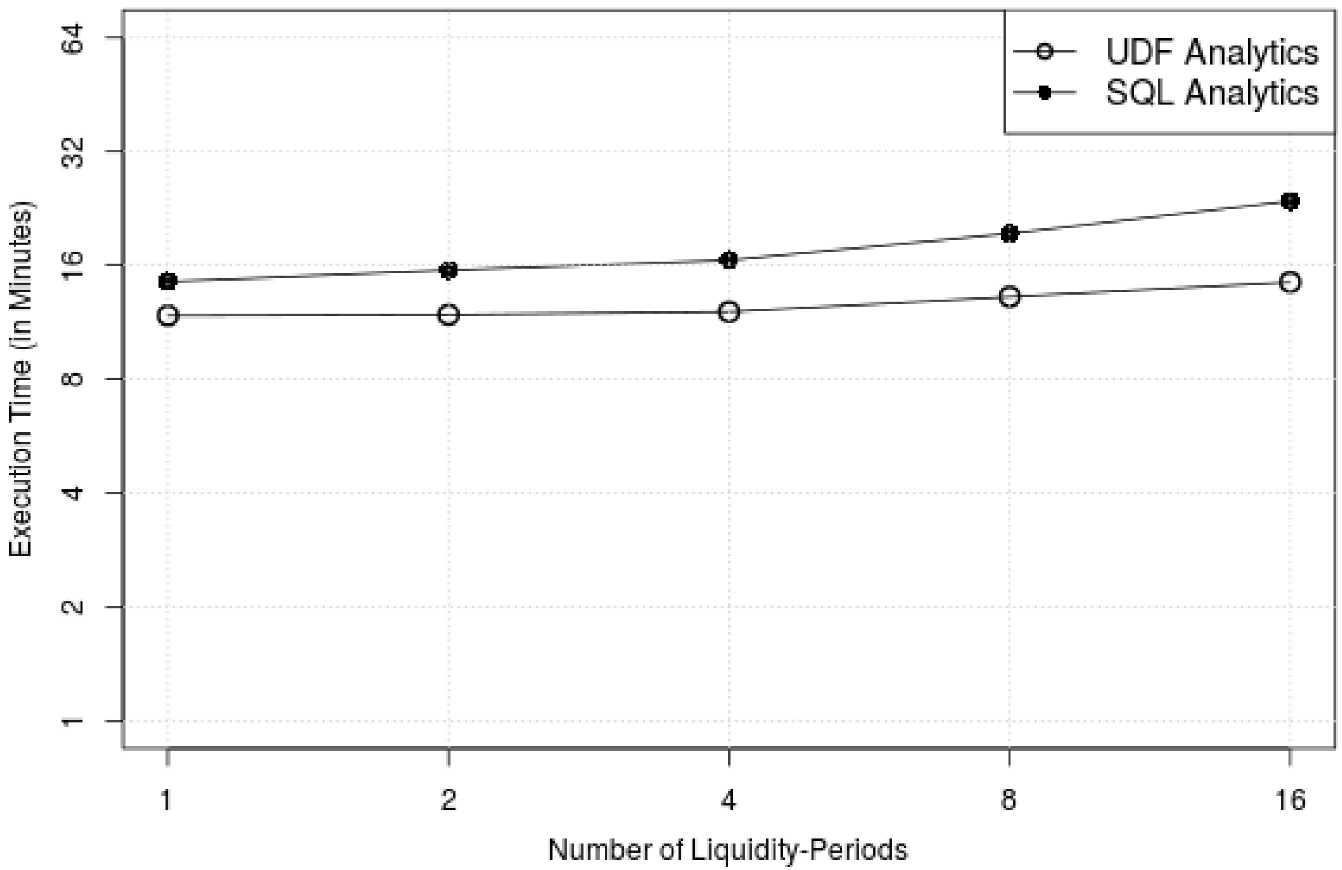
UDF analytics outperform SQL analytics



Spark memory problems
due to large memory footprint
(data needs to be read several times)

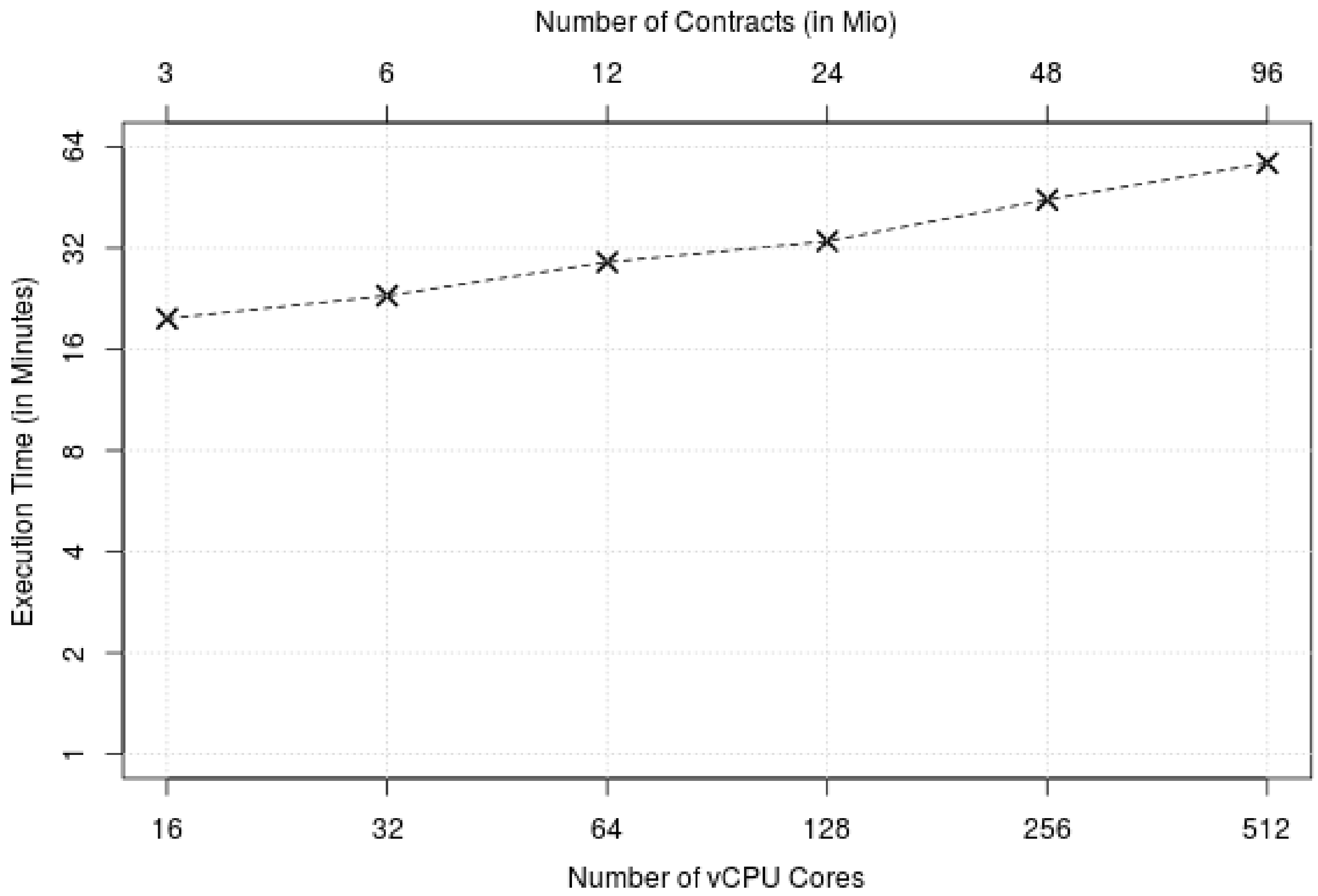
Liquidity Analysis

The more time periods, the longer the execution times



Generating and Materializing Cash Flows

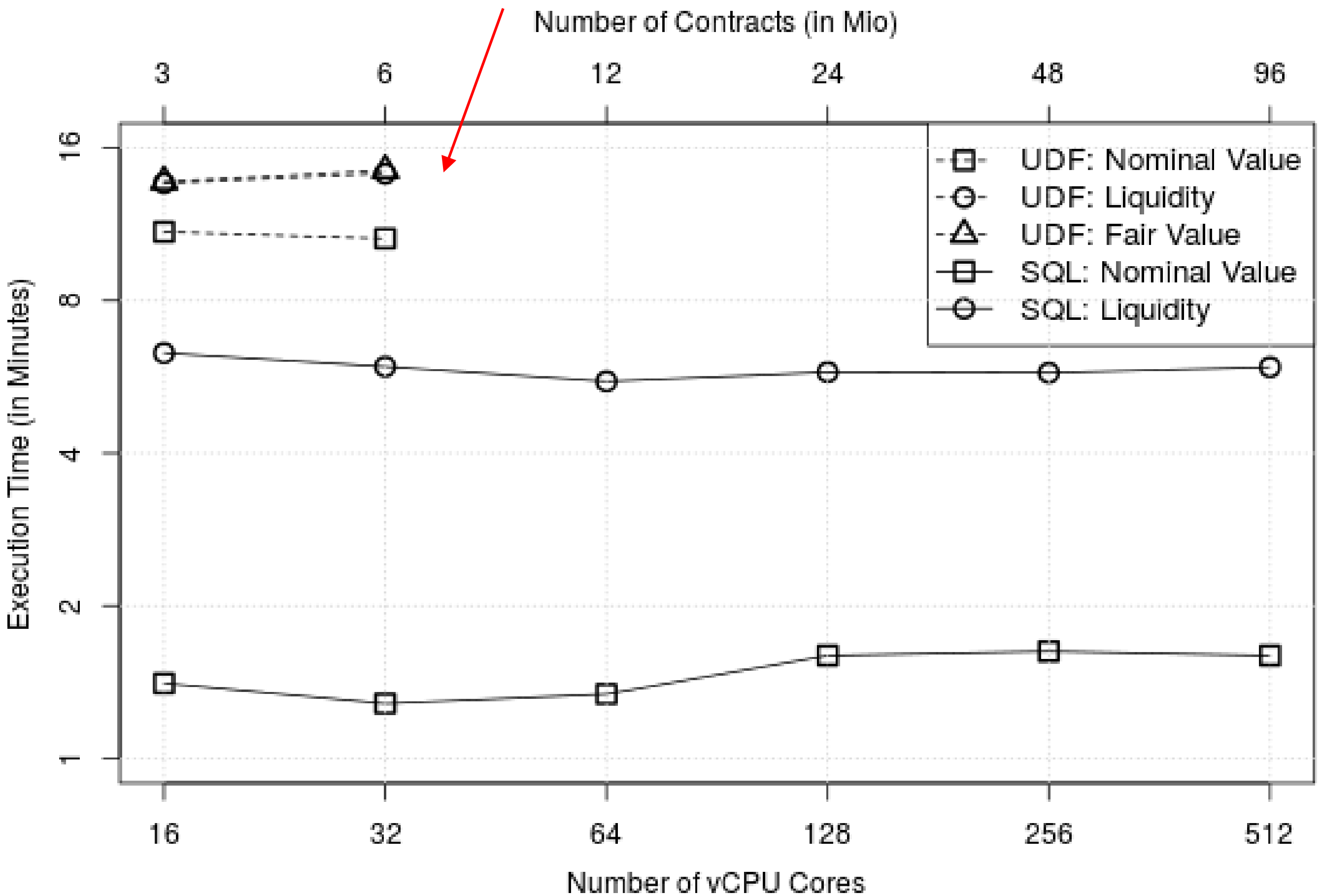
Overhead due to non-parallelized meta data management



UDF and SQL Analytics – Materialized Architecture

SQL analytics outperform UDF analytics

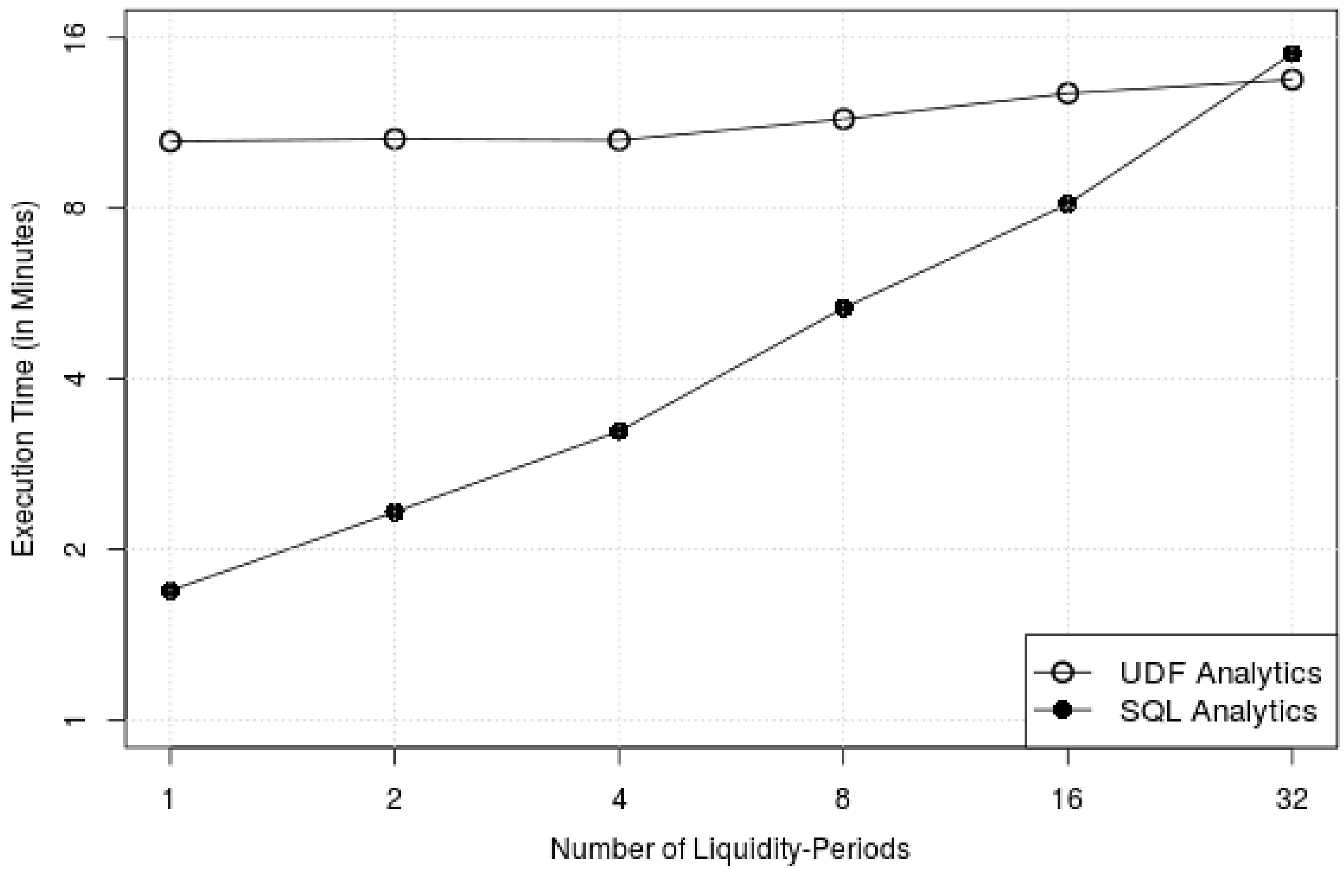
Spark memory problems due to large memory footprint



Liquidity Analyses

Zürcher Hochschule
für Angewandte Wissenschaften
zh
aw
School of
Engineering
InIT Institut für angewandte
Informationstechnologie

SQL analytics outperform UDF up to 16 liquidity periods



Conclusions and Lessons Learned

Zürcher Hochschule
für Angewandte Wissenschaften
School of Engineering
InIT Institut für angewandte Informationstechnologie

- Experiment setup on **up to 512 vCPU** cores on Amazon Web Services
- Most of the experiments show **close to linear scalability**

- **Lesson 1 - Use UDFs for On-the Fly Calculations:**
  - Use UDFs rather than rewrite financial kernel
- **Lesson 2 - Use SQL for iterative calculations on materialized results**
  - When results are materialized, SQL optimizer can improve run time
- **Lesson 3 - Performance tuning of Spark on real- world problems remains challenging**
  - Dynamic memory management for large jobs not ideal
  - Need manual tuning

- **Contact: Kurt Stockinger**