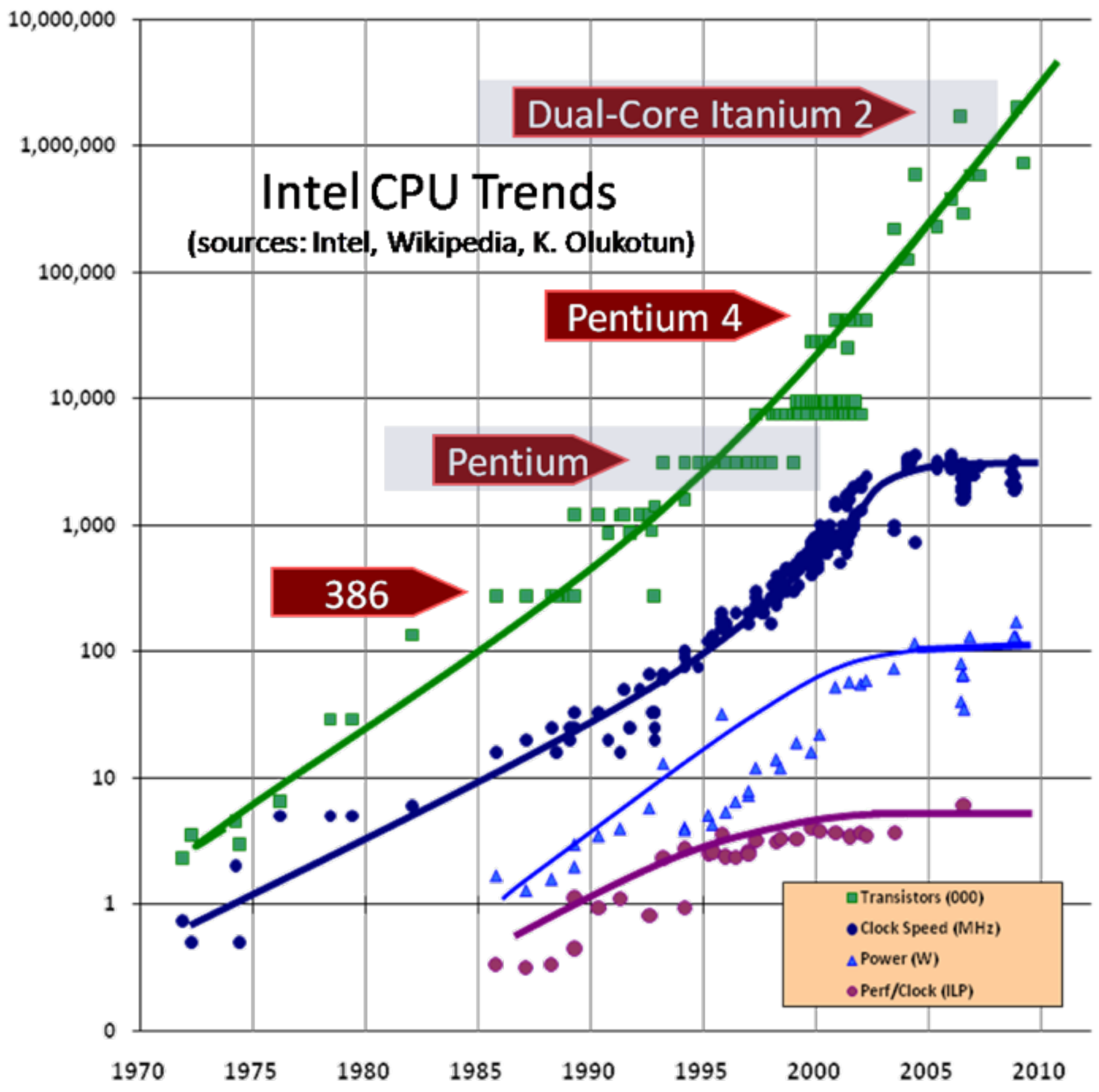
Advanced Topics in In-Memory Computing

Markus Dreseler, Martin Boissier

April 2016

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

"The Free Lunch Is Over"



- Number of transistors per chip increases
- Clock Frequency stalls

[Source: http://www.gotw.ca/publications/concurrency-ddj.htm]

Capacity vs. Speed (latency)

Memory hierarchy:

- Capacity restricted by price/performance
- SRAM vs. DRAM (refreshing needed every 64ms)
- SRAM is very fast but very expensive

Memory is organized in hierarchies

- Fast but small memory on the top
- Slow but lots of memory at the bottom

	technology	latency	size
CPU Register	SRAM	<1 ns	bytes
L1 Cache	SRAM	~ 1 ns	KB
L2 Cache	SRAM	< 10 ns	MB
Main Memory	DRAM	100 ns	GB-TB
Permanent Disc Storage		**~10 000 000 ns** (10ms)	**TB-PB**

Data Processing

In DBMS, on disk as well as in memory, data processing is often:

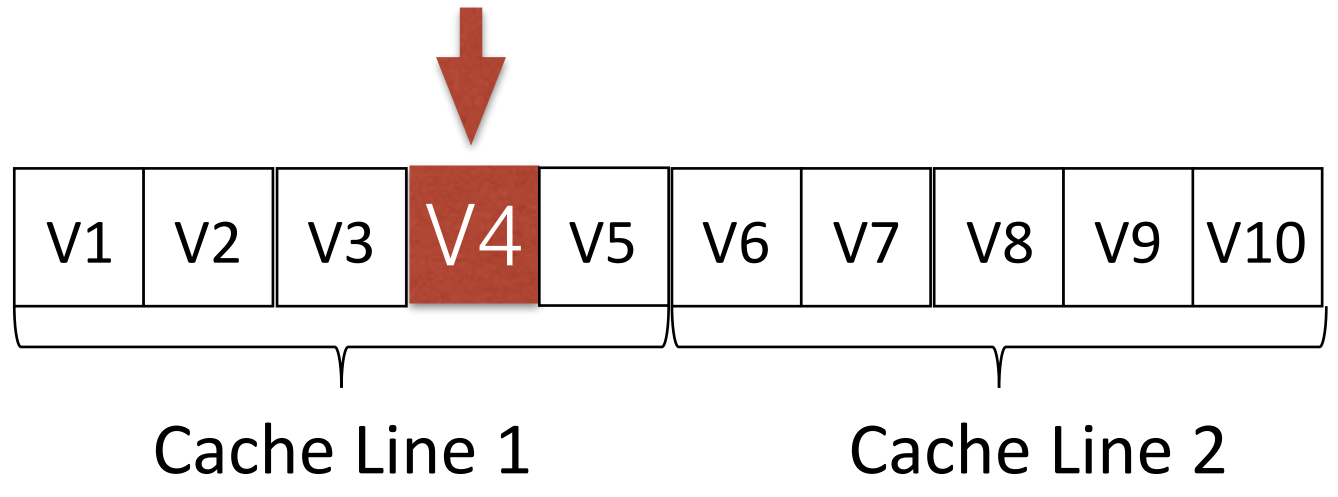
- Not CPU bound
- But bandwidth bound
- “I/O Bottleneck”

➡ CPU could process data faster

Memory Access:

- Not truly random (in the sense of constant latency)
- Data is read in blocks/cache lines
- Even if only parts of a block are requested

➡ Potential waste of bandwidth

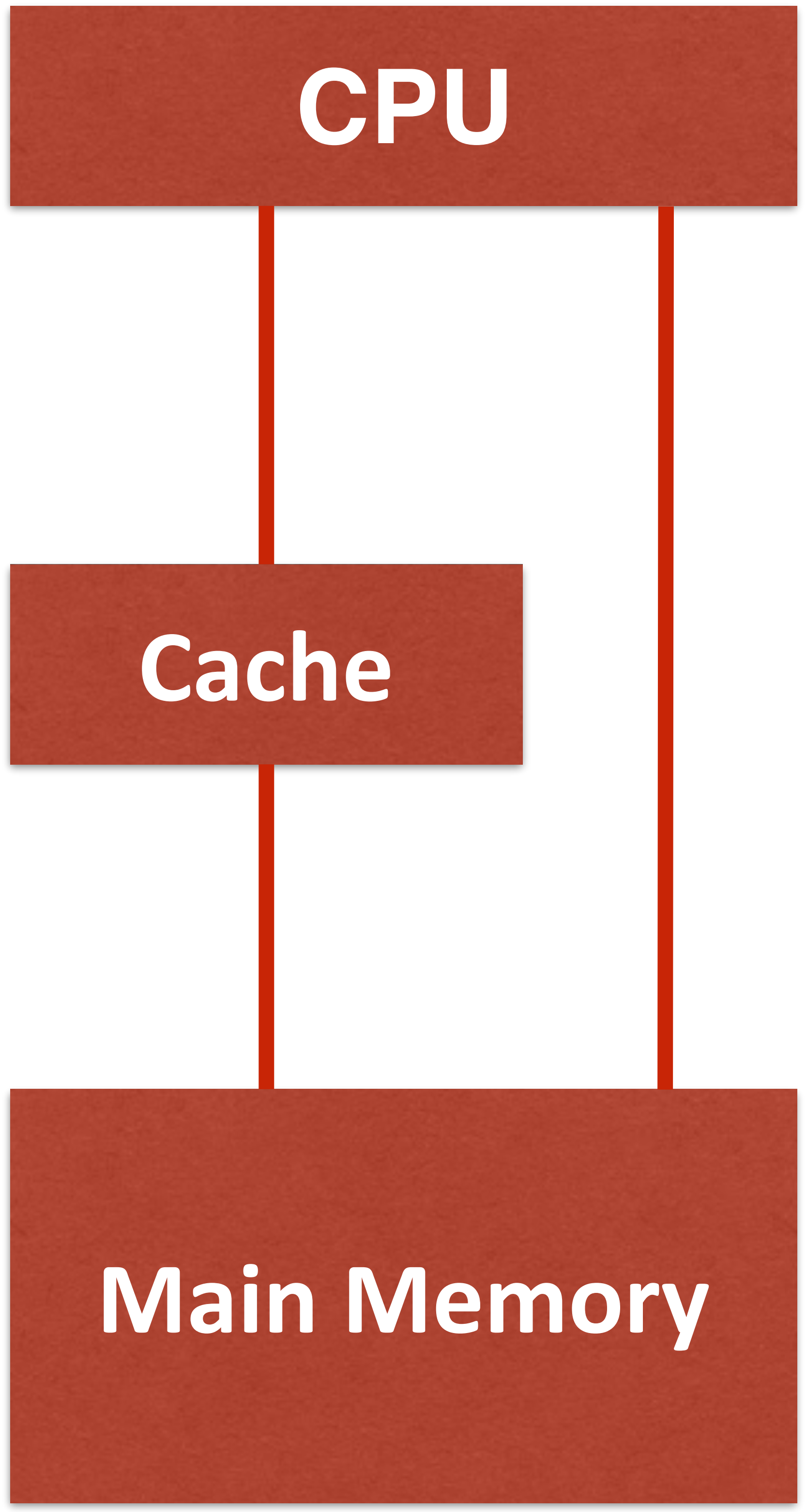


Memory Hierarchy

Cache
Small but fast memory, which keeps data from main memory for fast access.

Cache performance is crucial

- Similar to disk cache (e.g. buffer pool)
- **But:** Caches are controlled by hardware.



Cache hit
Data was found in the cache.
Fastest data access since no lower level is involved.

Cache miss
Data was not found in the cache. CPU has to load data from main memory into cache (miss penalty).

Locality is King!

- To improve cache behavior
  - Increase cache capacity
  - Exploit locality
    - Spatial: related data is close (nearby references are likely)
    - Temporal: Re-use of data (repeat reference is likely)

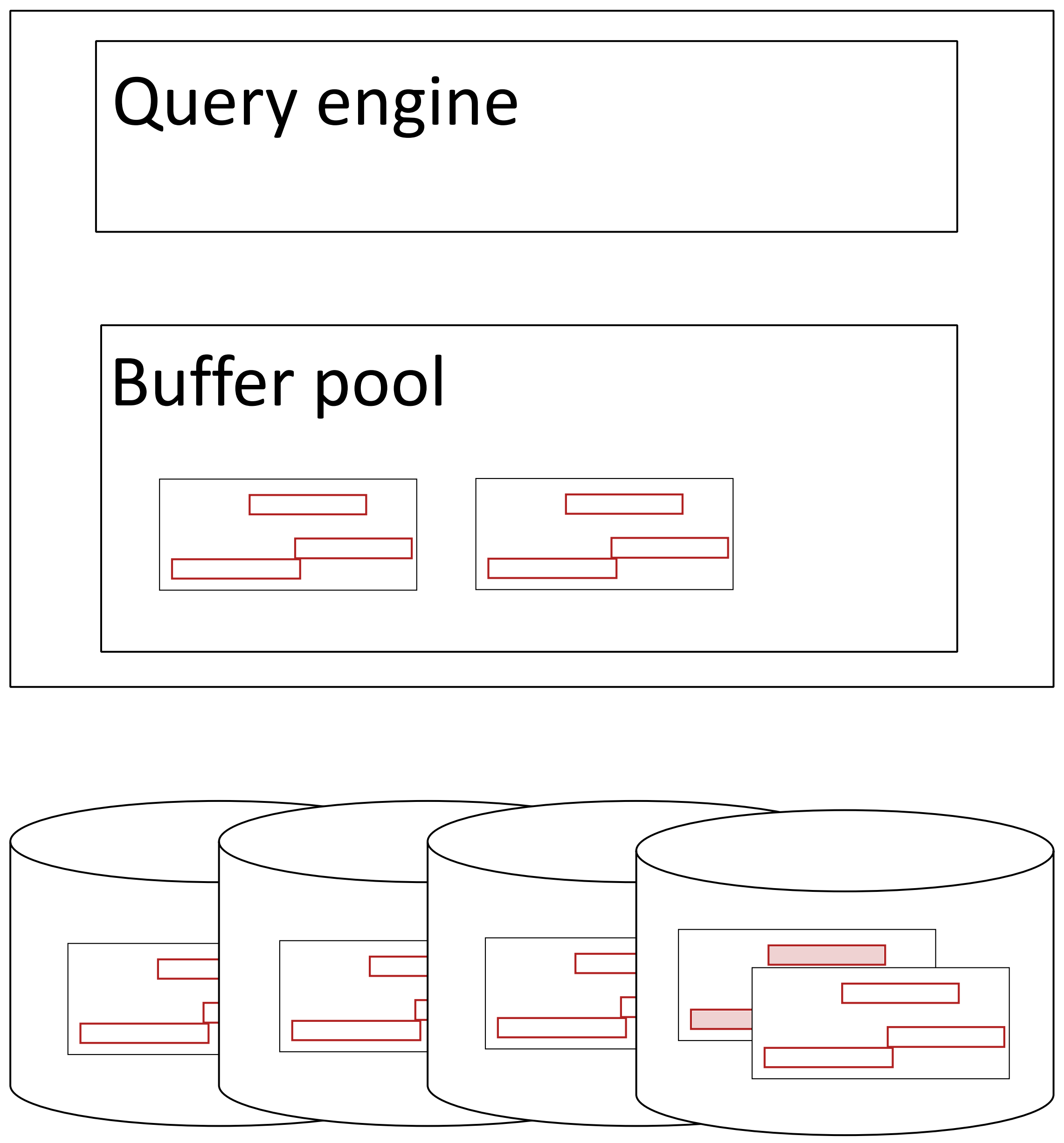
- To improve locality
  - Non random access (e.g. scan, index traversal):
    - Leverage sequential access patterns
    - Clustering data to a cache lines
    - Partition to avoid cache line pollution (e.g. vertical decomposition)
    - Squeeze more operations/information into a cache line
  - Random access (e.g., hash joins):
    - Partition to fit in cache (cache-sized hash tables)

Motivation

- Hardware has changed
  - TB of main memory are available
  - Cache sizes increased
  - Multi-core CPU’s are present
  - Memory bottleneck increased
  - NUMA (and NUMA on a NUMA?)
- Data / Workload
  - Tables are wide and sparse
  - Lots of set processing
- Traditional databases
  - Optimized for write-intensive workloads
  - Show bad L2 cache behavior

Problem Statement

- DBMS architecture has **not changed** over decades
- Redesign needed to handle the changes in:
  - Hardware trends (CPU/cache/memory)
  - Changed workload requirements
  - Data characteristics
  - Data amount

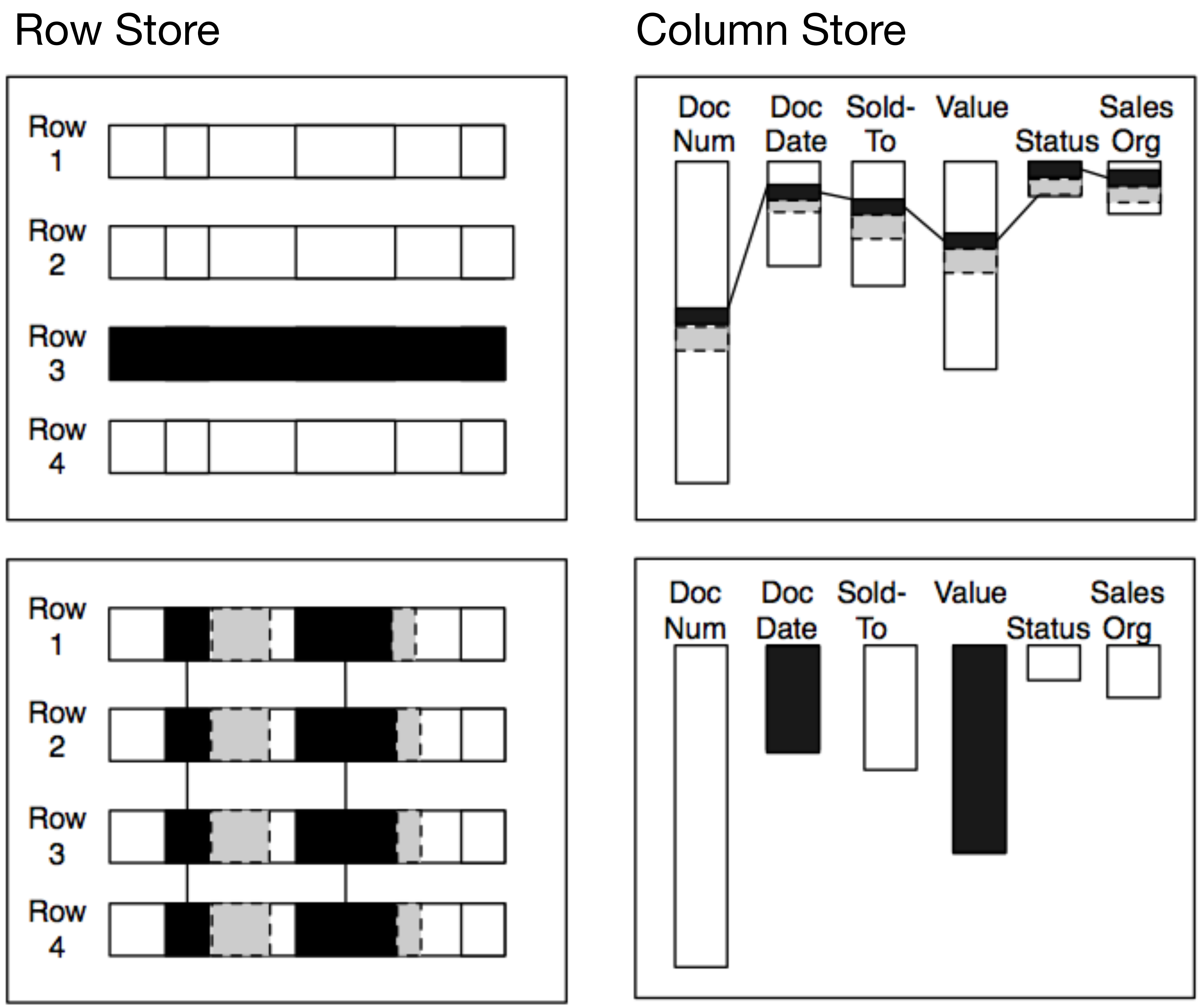


Traditional DBMS Architecture

Row- or Column-oriented Storage

SELECT *
FROM Sales Orders
WHERE Document Number = ‘95779216’

SELECT SUM(Order Value)
FROM Sales Orders
WHERE Document Date > 2016-01-20



Question & Answer

How to optimize an IMDB?

- Exploit sequential access, leverage locality
  - Column store
- Reduce I/O
  - Compression
- Direct value access
  - Fixed-length (compression schemes)
- Late Materialization
- Parallelize

Seminar Organization

HPI Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Objective of the Seminar

- Work on advanced database topics in the context of in-memory databases (IMDB) with regards to enterprise data management
- Learn how to work scientifically
  - Fully understand your topic and define the objectives of your work
  - Propose a contribution in the area of your topic
  - Quantitatively demonstrate the superiority of your solution
  - Compare your work to existing related work
  - Write down your contribution so that others can understand and reproduce your results

Seminar schedule

- Today (**11.04.**): Overview of topics, general introduction
- Thursday (**18.04.**): In-memory DB Basics & HYRISE
- **19.04.**: Send your priorities for topics to *markus.dreseler@hpi.de*
- **Planned Schedule**
  - **Week of 23.05.**: Mid-term presentation
  - **Week of 20.06.**: Final presentation (tbc)
  - **31.07.**: Peer Reviewing (tbc)
  - **07.08.**: Paper hand-in (tbc)
- Throughout the seminar: individual coaching by teaching staff
- Meetings (Room V-2.16)

Final Presentation

- Why a final presentation?
    - Show your ideas and their relevance to others
    - Explain your starting point and how you evolved your idea / implementation
    - Present your implementation, explain your implementations properties
    - Sell your contribution! Why does your work qualify as rocket science?

Peer Reviewing

- Each student will be assigned a colleague's paper version
  (approximately two weeks before paper hand-in)
  - Review will be graded
  - Annotate PDF for easy fixes (e.g., typos)
  - Short summary (2-3 pages) about the paper's content and notes to the author how to further improve his paper

- Expected to be done one week before paper hand-in

Final Documentation

- 7-9 pages, IEEE format [1]

- Suggested Content: Abstract, Introduction into the topic, Related work, Implementation, Experiment/Results, Interpretation, Future Work

- Important!
  - Related work needs to be cited
  - Quantify your ideas / solutions with measurements
  - All experiments need to be reproducible (code, input data) and the raw data to the experiment results must be provided

[1] http://www.ieee.org/conferences_events/conferences/publishing/templates.html

Grading

- 6 ECTS
- Grading:
  - 30% Presentations (Mid-term 10% / Final 20%)
  - 30% Results
  - 30% Written documentation (Paper)
  - 10% Peer Review

Topic Assignment

- Each participant sends list of top two topic areas in order of preference to lecturers by 19.04. (*markus.dreseler@hpi.de*)
- Topics are assigned based on preferences and skills by teaching team

HYRISE

- Open source IMDB
- Hosted at https://github.com/hyrise
- C++11
- Query Interface: Query plan or stored procedures

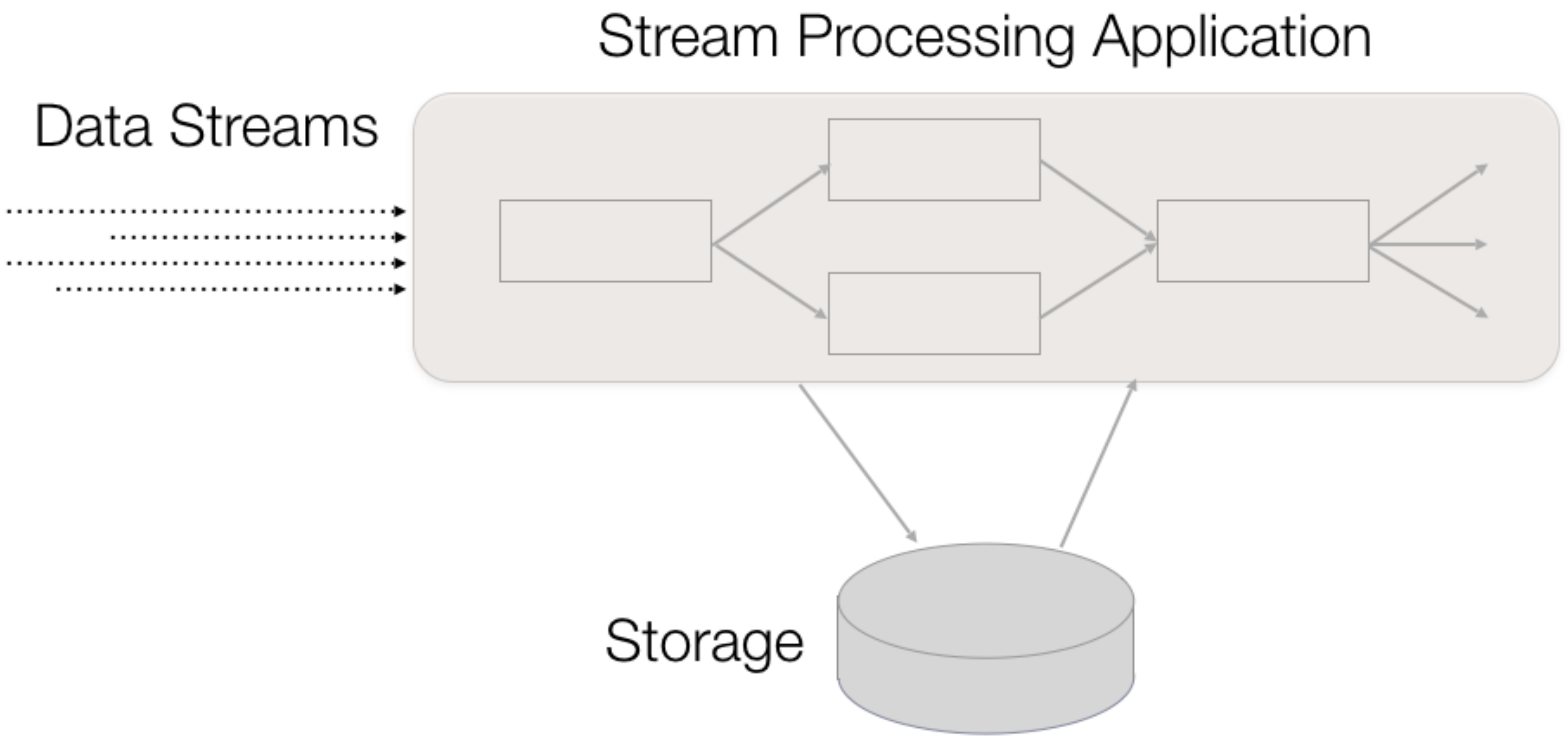
Recommended Papers for Intro

- Plattner, VLDB 2014: *The Impact of Columnar In-Memory Databases on Enterprise Systems*
- Krueger et al. VLDB 2012: *Fast Updates on Read-Optimized Databases Using Multi-Core CPUs*
- Grund et al. VLDB 2010: *HYRISE—A Main Memory Hybrid Storage Engine*

Topics

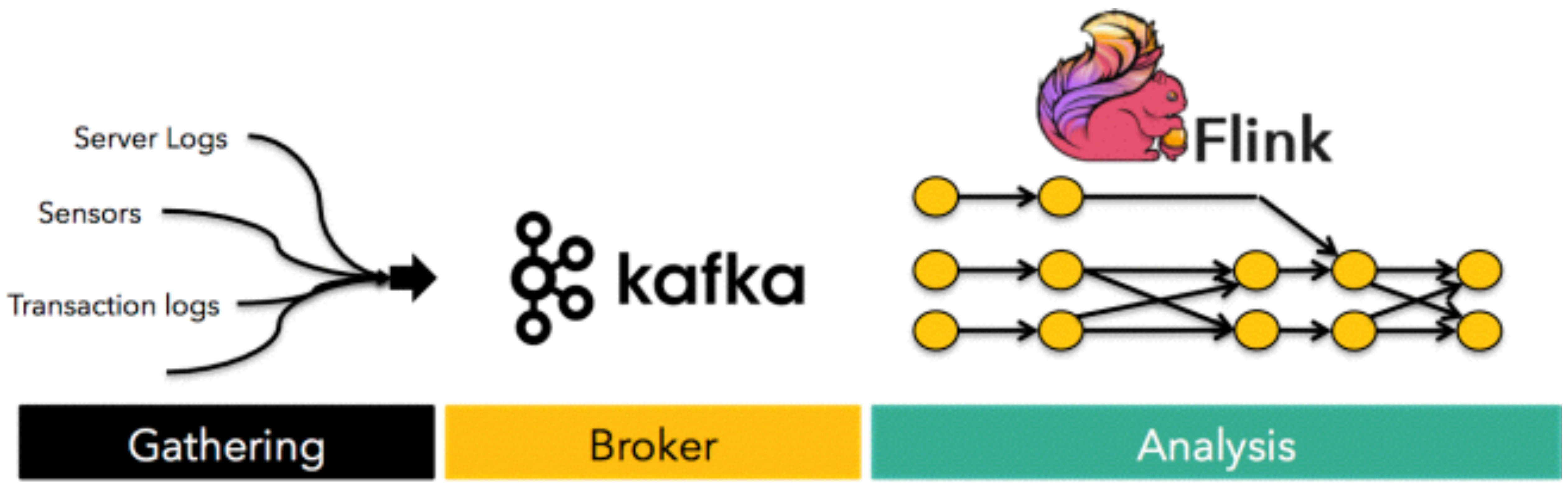
Data Stream Processing

- Aurora
- STREAM
- Samza
- Storm
- Spark (Streaming)
- Flink
- Apex
- …



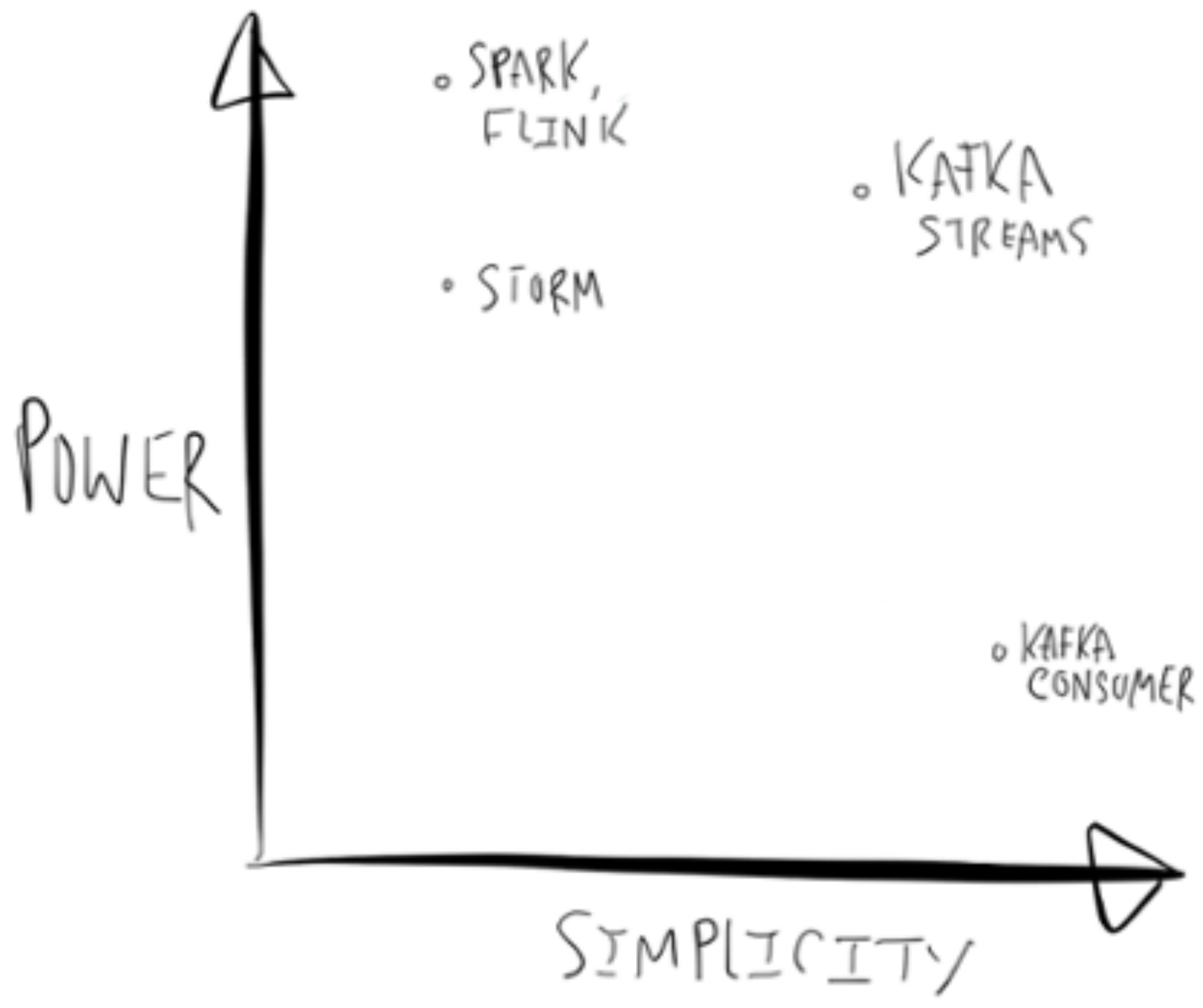
Data Stream Processing

Typical Streaming Architecture



http://www.confluent.io/blog/real-time-stream-processing-the-next-step-for-apache-flink/]

New System - Kafka Streams



http://www.confluent.io/blog/introducing-kafka-streams-stream-processing-made-simple

Evaluation of Kafka Streams

Question

How does Kafka Streams perform compared to other streaming systems like Flink or Spark?

Tasks

- Set-Up of Kafka Stream
- Implementation of Liner Road Benchmark [1]
- Measurement of benchmark runs w/ certain data set sizes and certain degrees of parallelism
- Result evaluation

[1] Arvind Arasu, Mitch Cherniack, Eduardo Galvez, David Maier, Anurag S. Maskey, Esther Ryvkina, Michael Stonebraker, and Richard Tibbetts. 2004. Linear road: a stream data management benchmark. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30* (VLDB '04), Mario A. Nascimento, M. Tamer Özsu, Donald Kossmann, Renée J. Miller, José A. Blakeley, and K. Bernhard Schiefer (Eds.), Vol. 30. VLDB Endowment 480-491.

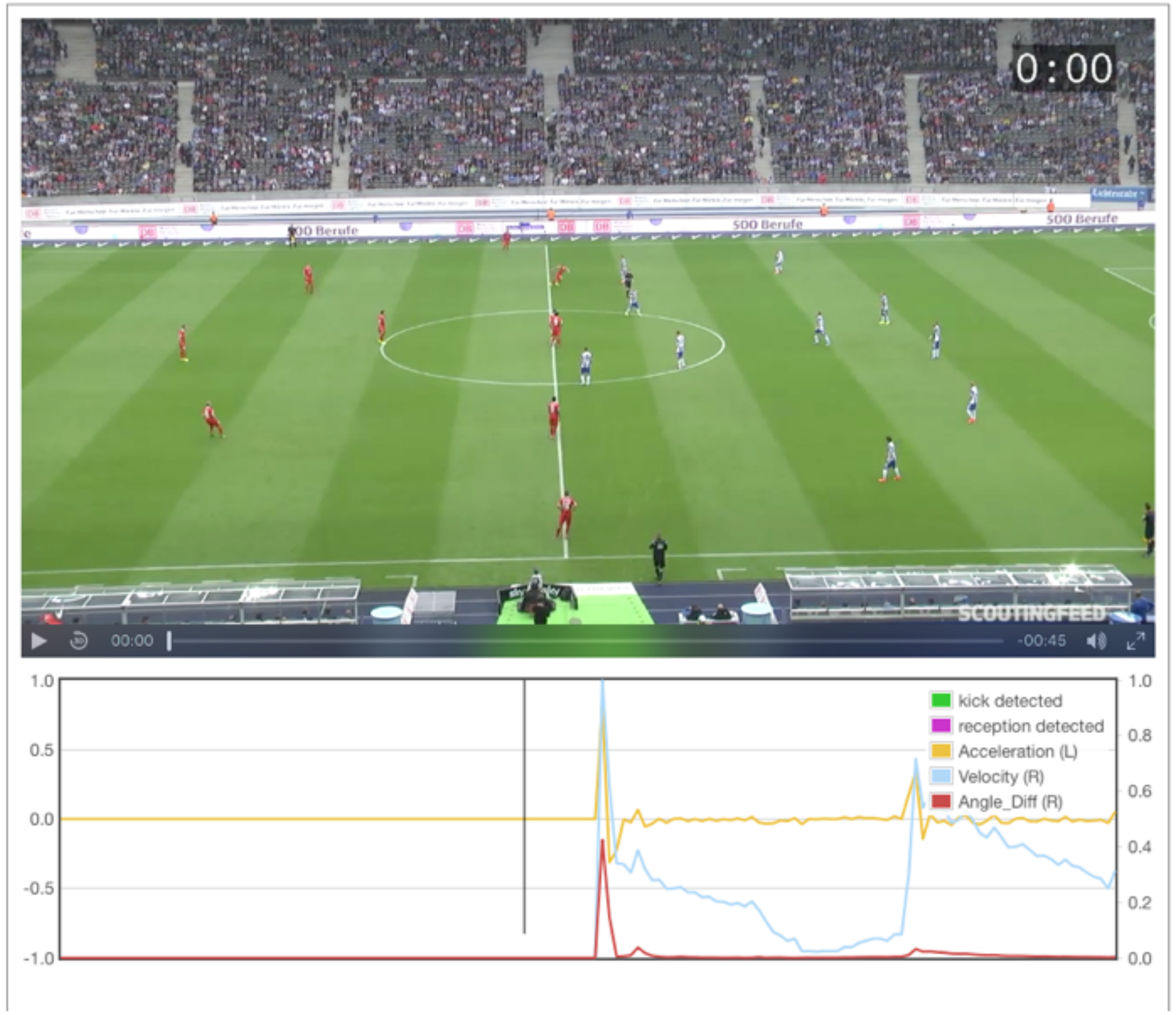
Recognizing Compound Events in Spatio-Temporal Football Data

- The usage of spatio-temporal data increased strongly in recent years (e.g. performance analytics in football)

- Provided data for football games of the German Bundesliga
  - ~1.5 million positional information per game
  - Manually tracked event list

- Problem: the event list is tracked manually, is not synchronized with the positional data, and contains errors

- **Tasks**
  - Implementation and evaluation of algorithms to automatically detect compound events in positional data of football games



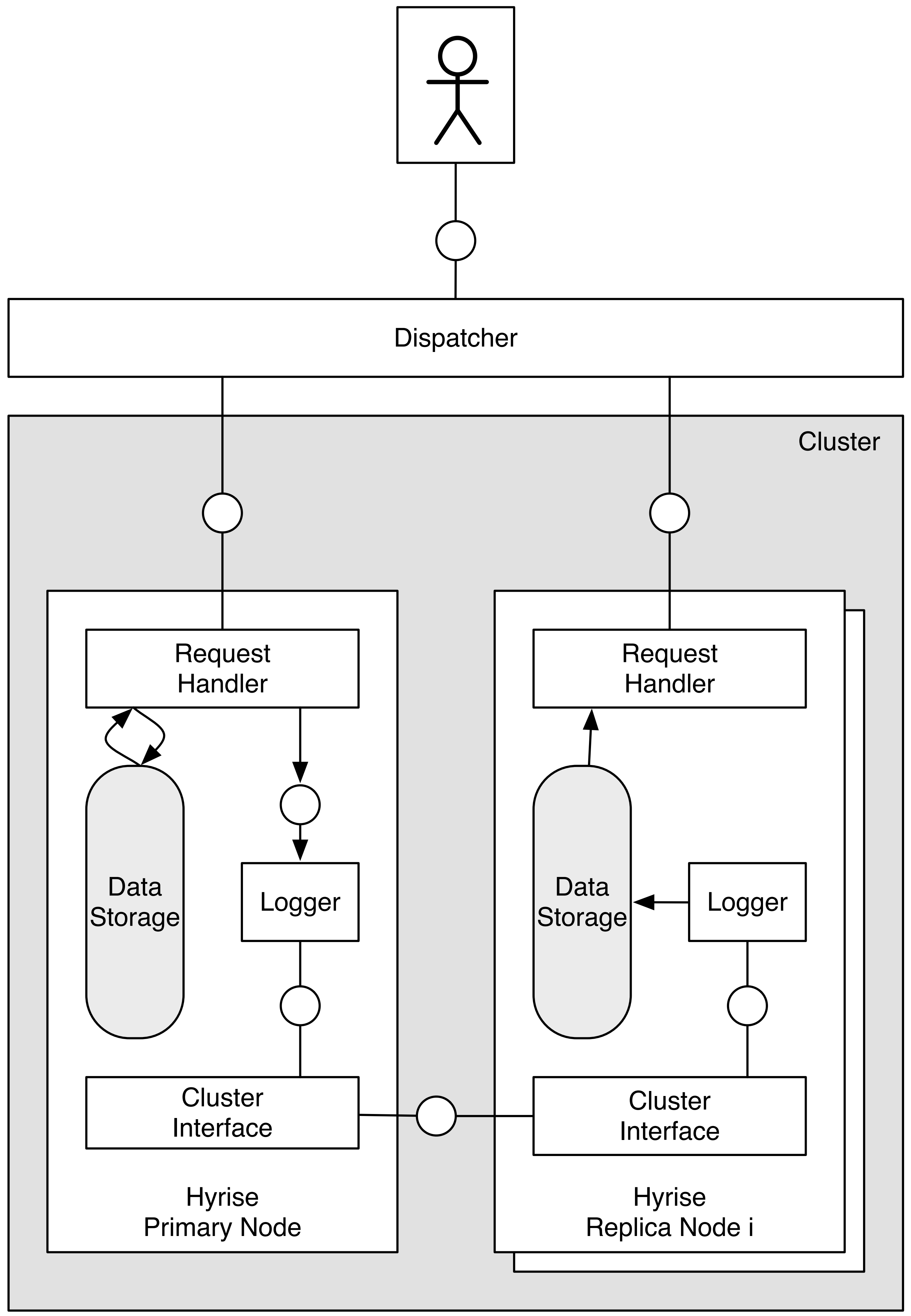
Keven Richly, Christian Schwarz, Rainer Schlosser

Hyrise-R

Implements elastic master replication

Topics

- Heterogeneous (indices, partitions) replicas
- Quick integration of new instances



Using Infiniband to Speed Up Networking

- Many database applications require network operations - especially in distributed and replicated databases
- Latency and throughput are significant factors for performance
- Infiniband is a technology that reduces networking overheads and hands off tasks to the hardware

Using Infiniband to Speed Up Networking

- The Infiniband API (ibverbs) is more verbose than the regular socket API and allows the programmer to fine-tune many aspects of networking
- Goal of this project is to understand the performance characteristics of different approaches, including:
  - Send vs. RDMA Write
  - Scatter-Gather
  - Reliable vs. unreliable networking

SAP HANA Tiering

- Situation
  - In large enterprise databases, only a small subset of all data is regularly accessed
  - Keeping rarely accessed data in main memory is as a waste of resources
- Project
  - Evaluation of a new tiering approach for SAP HANA
  - Less frequently accessed data is stored on disk while fully retaining the OLAP performance of SAP HANA
  - Cooperation with SAP Innovation Center in Potsdam

SAP HANA Tiering

- Project Setup
  - Little bit of C++ coding in SAP HANA
    - it's not a lot, still not easy though (it's high performance production C++)
  - development environment already set up
  - *50% is algorithmic work* (optimal sorted order to reduce page cache misses)
  - *30 % is benchmarking* (we've got a nice tool set at hand)

Workload Analyses

- Industry-driven (and synthetic) workloads are known to be way off from real-world workloads
- With a real-world workload at hand, what can we do now?
- **Project Setup**
  - We traced TPC-C and TPC-E and you'll analyze and compare synthetic workloads with a real-world production system
  - Focus of project is a thorough analysis and workload comparison
  - Evaluation framework written in Python and bash

Martin Boissier

Hasso Plattner Institut
IT Systems Engineering | Universität Potsdam

Workload-Driven Partitioning

- Partitioning for Mixed Workload Databases
  - OLxP databases profit from partitioning because scans can be pruned to a subset of partitions, but finding such a partitioning scheme is burdensome
  - A recent master's thesis analyzed an recent automated partitioning approach and applied it to a real-world production system
  - The results were more than promising (in average each query scan skips ~90% of all tuples through partition pruning)
- Project
  - A more recent research paper proposed an another approach
  - You'll implement that new approach (framework is already in place) and compare both approaches (performance, applicability for real-world OLxP systems, their pros and cons, ...)

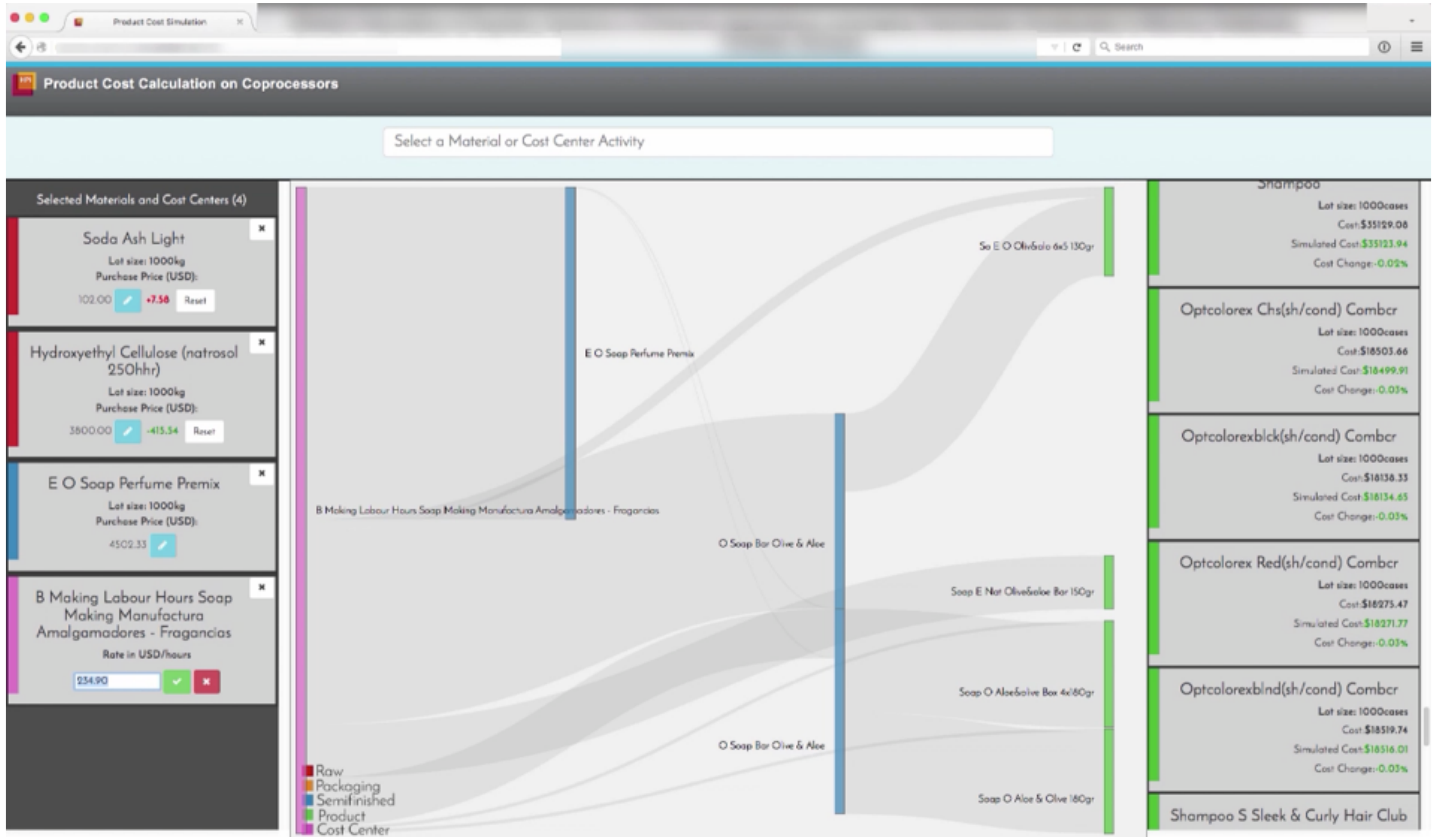
GPU Computing for Enterprise Applications

- Status Quo:
  - Application logic moves closer to the database layer
  - Compute intensive, long running application transactions consume computational power of the database system
  - Classical database tasks have less available resources

- Solution:
  - GPUs and other Coprocessors offer enormous amount of computational resources and on device memory bandwidth
  - Data parallel operations perform very well on these components

Example Application: Product Cost Simulation

- Application calculates cost (and other features) of products are calculated, using a system of linear equations
- Base parameter are stored inside the database system
- Problem is solved with specialized matrix inversion and matrix vector multiplication

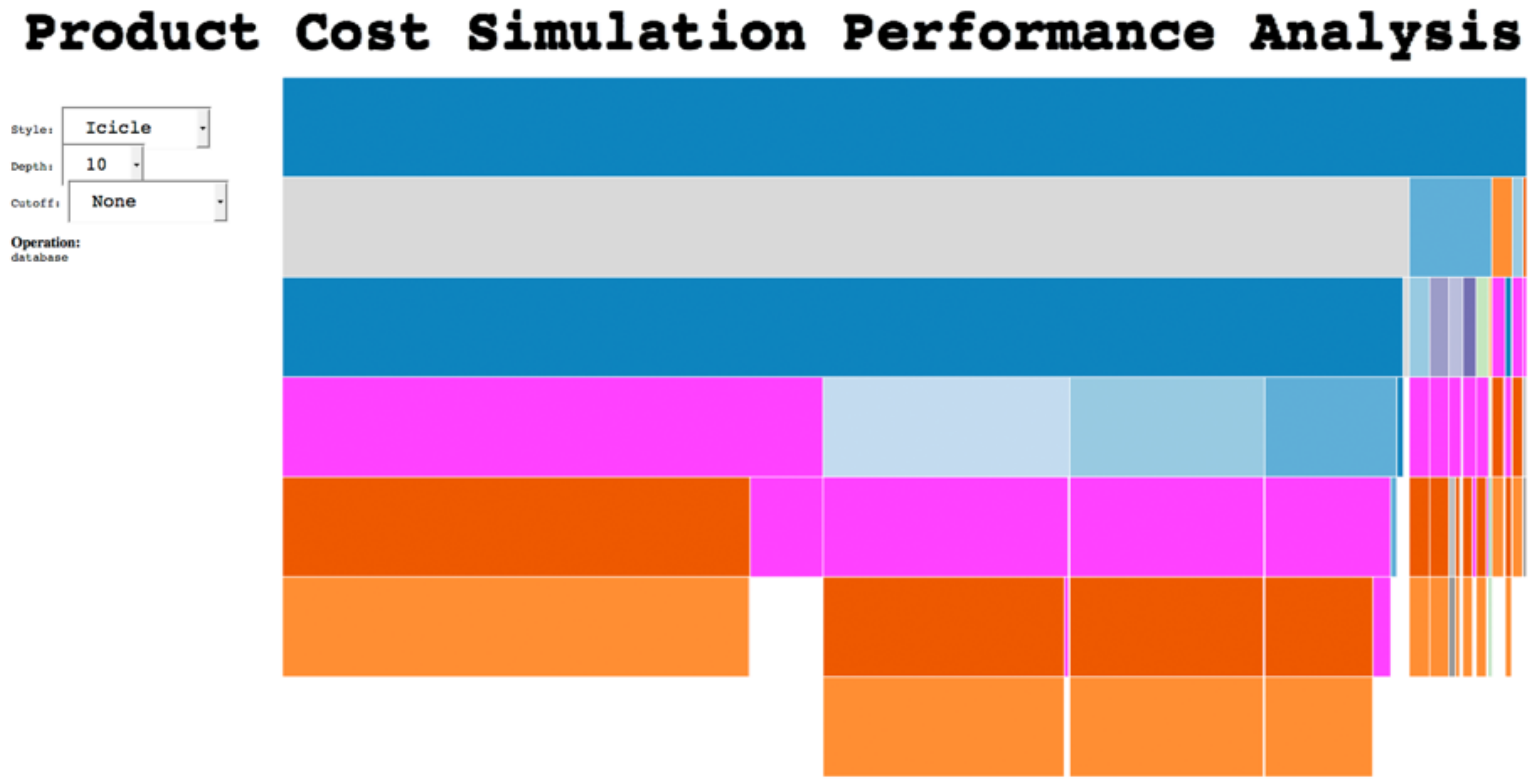


- Implementation can leverage computational resources of *Intel Xeon Phi* and *Nvidia Titan X*

Christian Schwarz

Coprocessor Integration Status

- Database is connected as a separate process
- Data is transferred using local ODBC connection
- Major part of application runtime is spend in database connector
- **Tasks**
  - Improve application runtime by integrating logic into database system
  - Define an execution model suitable for logic integration
  - Using *SAP HANA Advanced Function Library* to integrate code into running database system
  - Measure performance improvement compared to separate process model



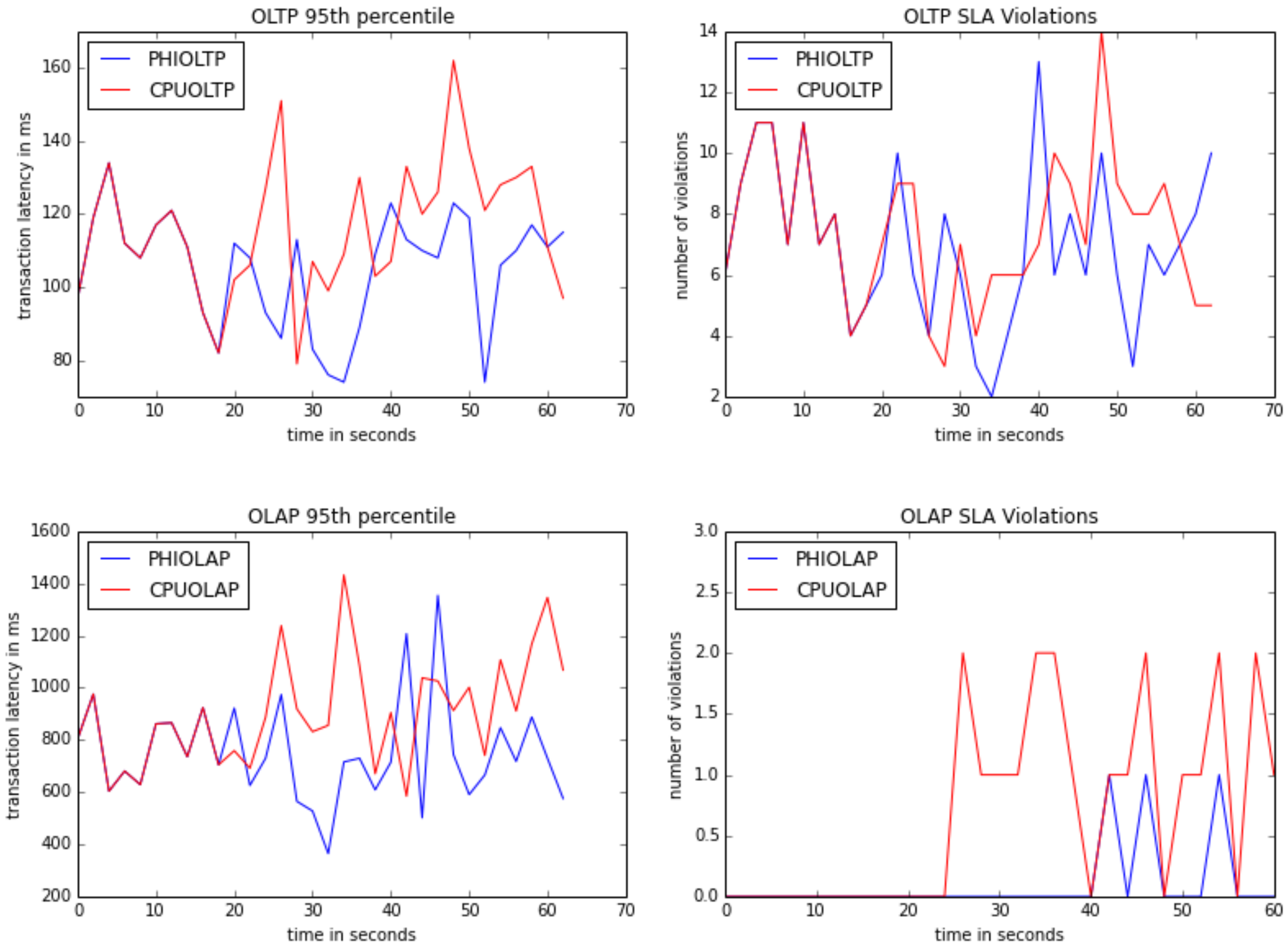
Search: database

Number of Executions	SUM(time) in seconds	AVG(time) in seconds	call
165	63.876	0.387	/bill_of_material/json_materialization /database
165	55.271	0.335	/bill_of_material/json_materialization /database/select
165	55.171	0.334	/bill_of_material/json_materialization /database/select/nanodbc::execute

Christian Schwarz

Coprocessor Impact on Enterprise System

- Integrating complex application logic into database system increases system workload
- Coprocessors add computational resources to the system, helping to handle the new workload
- **Tasks**
  - Define new workload based on existing ones
  - Measure the impact of new applications on benchmark results
  - Evaluate the influence on different query categories (OLTP, OLAP)



Topic Assignment

- Each participant sends list of top two topic areas in order of preference to lecturers by 19.04. (*markus.dreseler@hpi.de*)
- Topics are assigned based on preferences and skills by teaching team