Research and Implementation of Database Concepts

Kickoff

Martin Boissier, Thomas Bodner, Stefan Halfpap, Jan Koßmann, Dr. Michael Perscheid, Dr. Daniel Ritter, Marcel Weisgut
Enterprise Platform and Integration Concepts

EPIC Teaching Activities

<table>
<tr><th></th><th>Winter Term</th><th>Summer Term</th></tr>
<tr><td rowspan="2">Bachelor</td><td>Scalable Software Engineering
(Lecture, 5th term, revised SWT II)</td><td>Foundation of Business Software
(Lecture, 4th term)</td></tr>
<tr><td colspan="2">Bachelor’s Project (5th and 6th term)
Online Marketplace Simulation: A Testbed for Self-Learning Agents</td></tr>
<tr><td rowspan="4">Master</td><td>Trends and Concepts in the Software Industry II
(Seminar with Prof. Plattner and target-actual comparison of enterprise software at customers)</td><td>Trends and Concepts in the Software Industry I
(Lecture with Prof. Plattner and industry partners)

Trends and Concepts in the Software Industry III
(Optional Project Seminar)</td></tr>
<tr><td>Data-driven Decision Support
(Lecture and Project)</td><td>Causal Inference
(Lecture and Project)</td></tr>
<tr><td>Research and Implementation of Database Concepts
(Research Seminar)</td><td>Build Your Own Database
(Lecture and Project)</td></tr>
<tr><td colspan="2">Master Projects

Data-driven Decision Support   Autonomous Data Management</td></tr>
</table>

What to expect?

- Better understand how database systems work
- Learn how to familiarize yourself with a larger code base
- Work in small teams on a larger project

Same as in the Develop your own Database (DYOD) seminar

- Gain experience in systems development
- Improve your C++(20) skills

Less of a focus than in DYOD

- Research experience
- Related work, Conduct experiments, visualize results, communicate findings

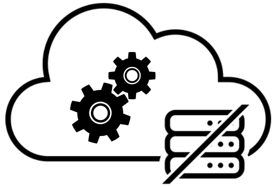
New in this seminar

How does this relate to Develop your own Database?

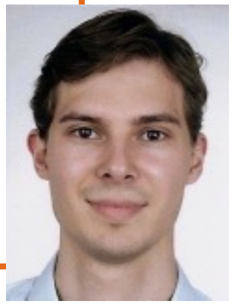
- We found that thesis students often have little experience in communicating their results
- This seminar is supposed to be a „thesis light“, including literature research, implementation, designing and executing experiments, and presenting the results in speech and writing
- It is both suitable for those students who have taken DYOD and for those who have not
- BUT: No weekly meetings with the entire group, thus no DBMS/C++ introduction
  - Previous experience, e.g, from Trends and Concepts or the DBS lectures is helpful
  - DYOD slides and sprint documents are available if you want to read up on details
- More research-oriented, i.e., the projects are proposals, not full specifications

Who are we?

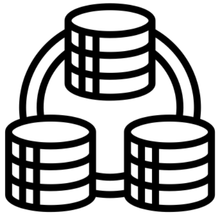
Cloud Data Management



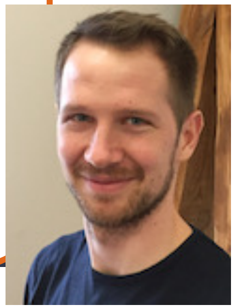
Thomas Bodner



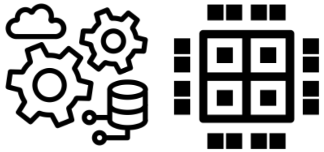
Query-Driven
Data Allocation



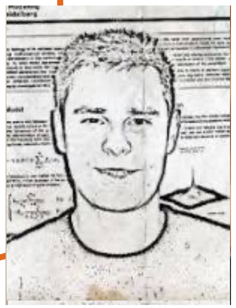
Stefan Halfpap



In-Memory, Cloud DB &
Next Gen. Hardware



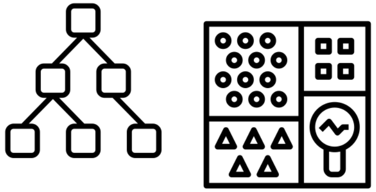
Daniel Ritter



Compression

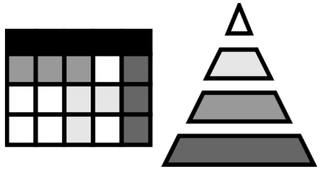
Martin Boissier

Unsupervised DB
Optimization

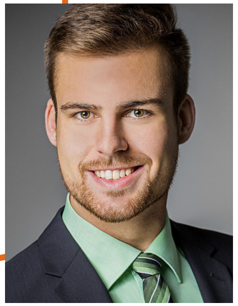


Jan Kossmann

Data Management on
Modern Storage Technol.



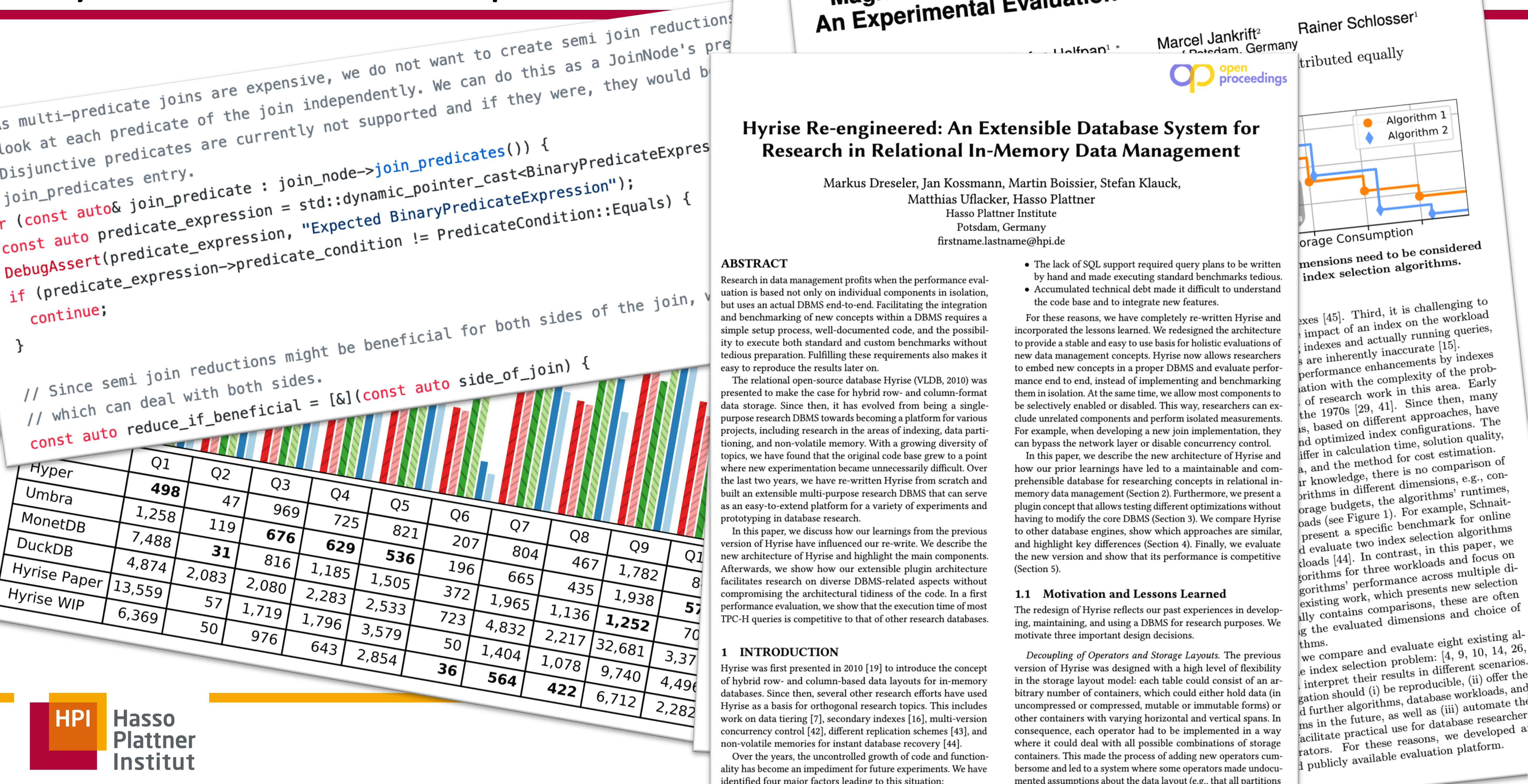
Marcel Weisgut



Hyrise

- An In-Memory Storage Engine for Hybrid Transactional and Analytical Processing
- HYRISE is a research database for the systematic evaluation of new concepts for hybrid transactional and analytical data processing on modern hardware
- Developed with and by HPI students
- Open Source (https://git.io/hyrise)
- System paper published at EDBT’19
- Modern, documented C++20 code base, 93% test coverage
- SQL interface, PostgreSQL network protocol
- Easy to extend via plug-in interface
- Supported benchmarks: TPC-(C|H|DS), JCC-H, Join-Order
- Runs on Intel, AMD, IBM Mainframe, ARM, Apple M1, Raspberry PI

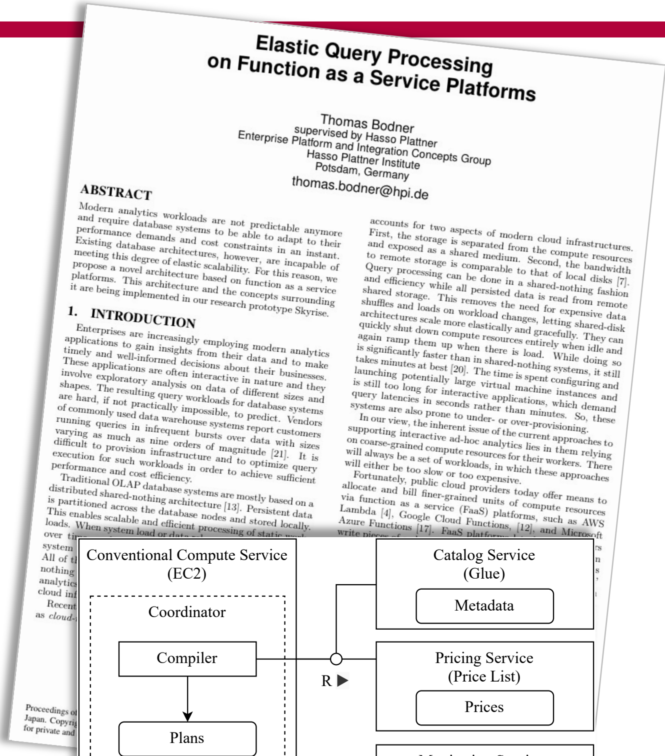
Hyrise in three* pictures



HPI Hasso Plattner Institut

Skyrise

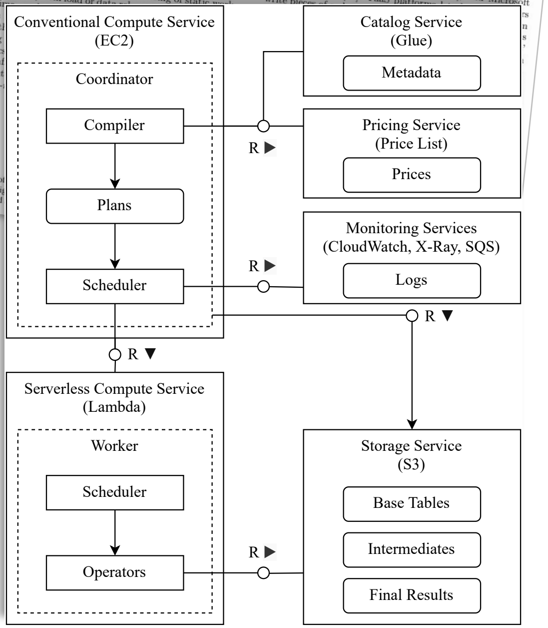
- A serverless query processor for interactive in-situ analytics on cold data
  - **Serverless:** Built on function as a service platforms and object storage
  - **(SQL) query processor**: Relational query execution and optimization
  - **Interactive:** Aims at query latencies in seconds
  - **In-situ:** Processes data without upfront load/align/sort/compress/index/..(ing)
  - **Cold data:** Infrequently accessed TB/PB-scale historical, IoT and Web data
- Initiated in fall 2019 to explore modern cloud infrastructure for databases
  - Exploits scalability, elasticity and reliability of the cloud, deals with its challenges
  - Modern C++ (17), documented, tested (> 90% coverage) codebase
  - Just starting out, plenty of research ahead!
  - Vision paper published at VLDB ’20
  - 3x Master’s theses, 2x seminar papers

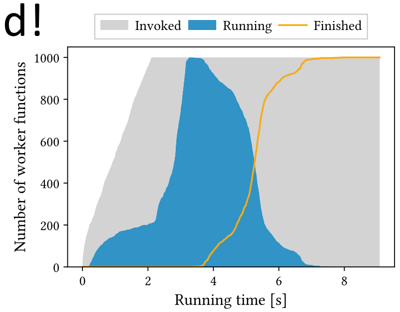

Elastic Query Processing
on Function as a Service Platforms

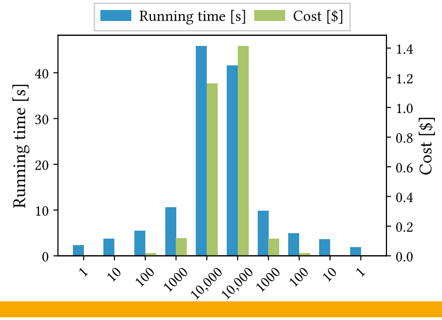
Thomas Bodner
supervised by Hasso Plattner
Enterprise Platform and Integration Concepts Group
Hasso Plattner Institute
Potsdam, Germany
thomas.bodner@hpi.de

ABSTRACT

1. INTRODUCTION







Comparison

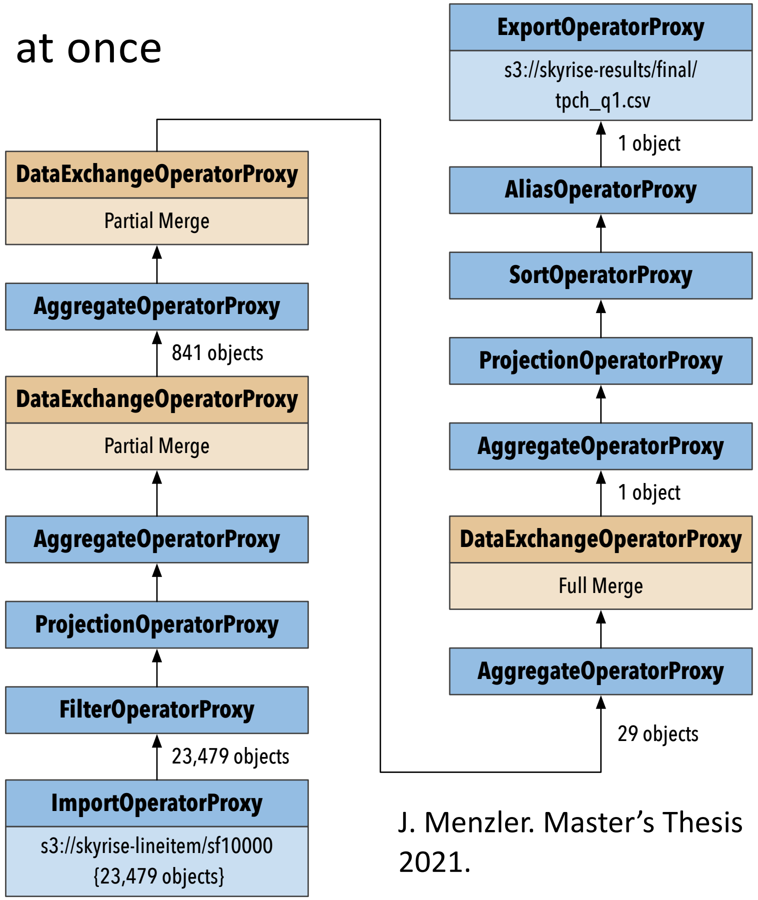
	Hyrise	Skyrise
Target workload	HTAP on hot to warm data	Interactive OLAP on cold data
Dataset size sweetspot	Gigabytes to a few Terabytes	Gigabytes to (hundreds of) Terabytes
Architecture	Scale-up within large bare metal machines	Independent scale-out of decoupled FaaS-based compute and cloud object storage
Pricing model	Pay upfront for machine and provisioning, pay as you go for maintenance and energy	Cloud object storage is $23/TB/month, pay as you go per query, as an example TPC-H Q1 @ SF1000 is currently $0.16

Research Topics

1. In-Memory Pipelined Query Execution
2. Analyzing Traces of Serverless Query Execution
3. Incorporating Distributed Plans into Query Optimization
4. Learned Indexes on Dynamic Data
5. Efficient and Accurate Histograms
6. Database Node Placement in the Cloud
7. Partial Indexes
8. Dynamic Data Placement Algorithms

In-Memory Pipelined Query Execution

- Initially, Skyrise adopted Hyrise‘s materialized execution model
  - Each operator consumes its input all at once and then produces its output all at once
  - Easy to reason about and only option for cross-worker processing
  - Intermediate query execution results may exihibit large footprint
  - No opportunity for parallelism along query pipelines of operators
- We study a hybrid materialized/pipelined execution model
  - Workers have little memory capacity and run single query pipeline each
  - Intermediates are materialized across and pipelined within workers
  - Extend operator set (import, filter, projection, ..) to work on „chunks“
  - Analyze worker main memory usage and query pipeline runtimes
- Prior experience with cloud services beneficial

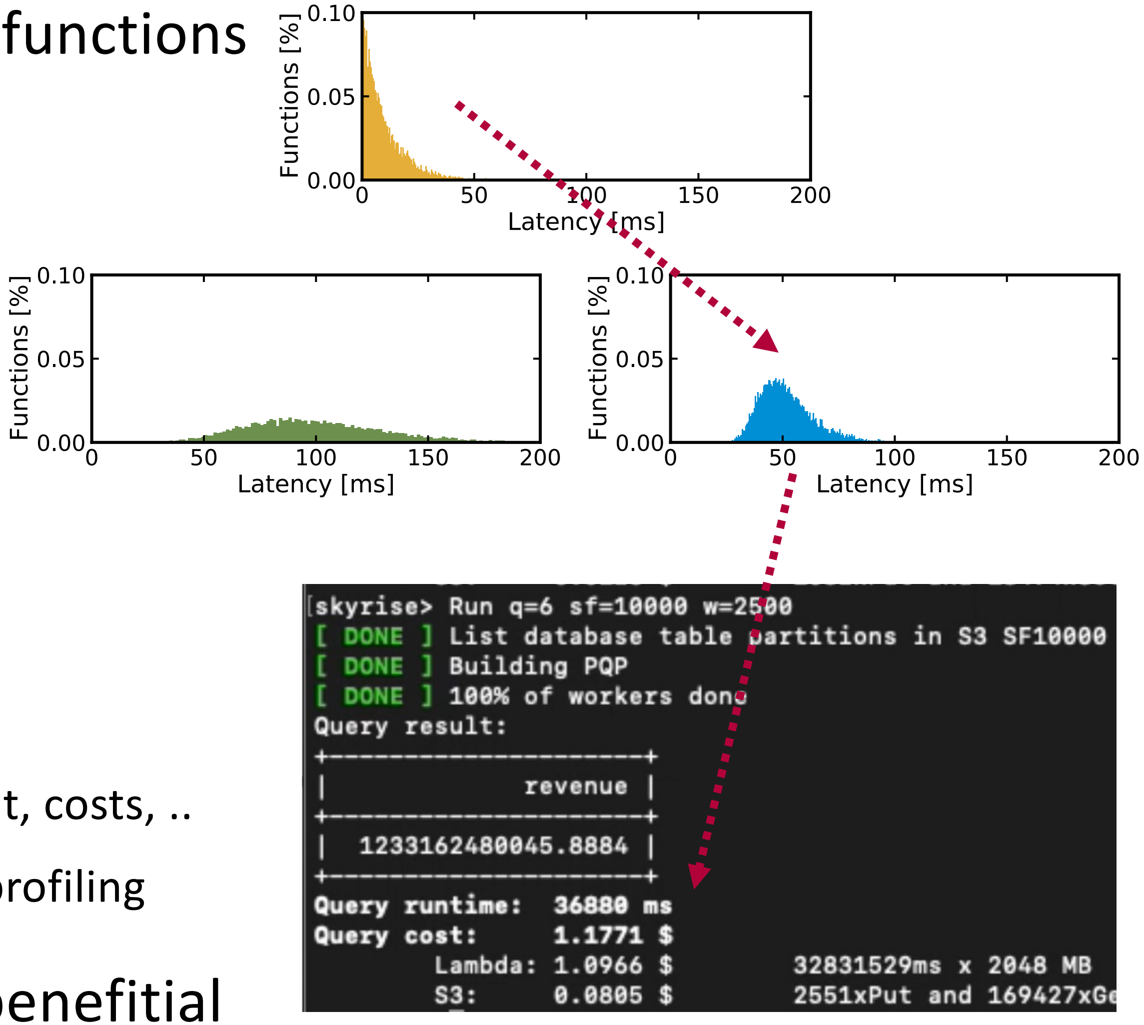


J. Menzler. Master’s Thesis 2021.

P. A. Boncz, M. Zukowski, N. Nes. MonetDB/X100: Hyper-Pipelining Query Execution. CIDR 2005.
M. Perron, R. C. Fernandez, D. DeWitt, S. Madden. Starling: A Scalable Query Engine on Cloud Function. SIGMOD 2020.

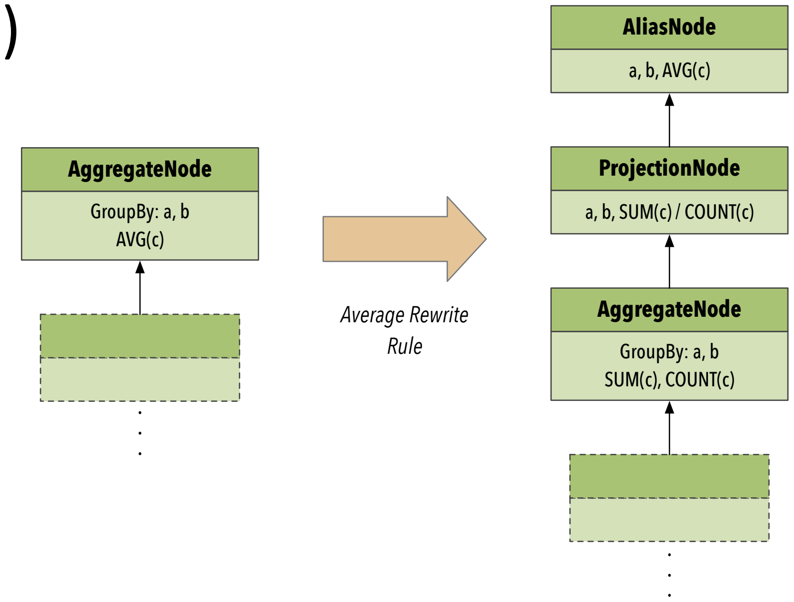
Analyzing Traces of Serverless Query Execution

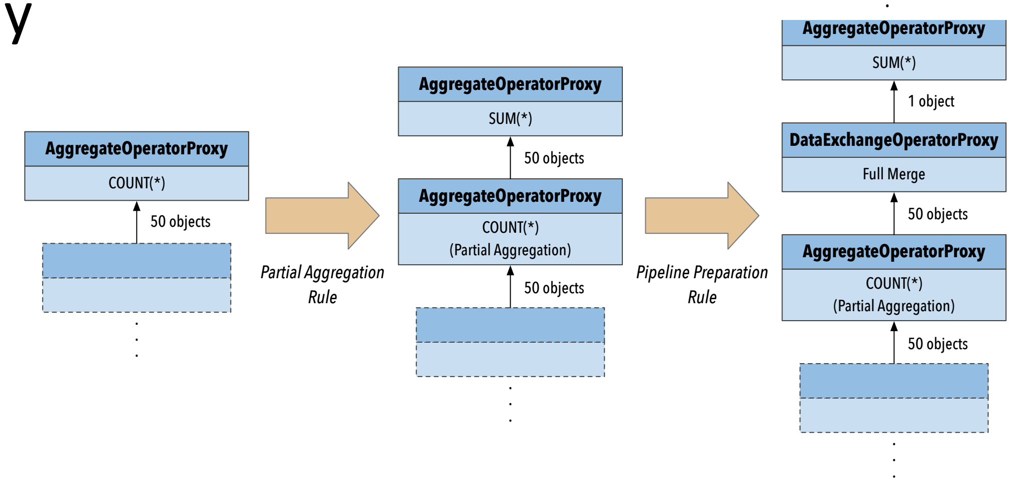
- Skyrise executes queries in parallel across cloud functions
  - Cloud functions run pipelines of query operators
  - Concurrency to several thousand function invocations
  - Skyrise inherits properties of FaaS platform, i.e., elastic scalability, reliability, performance and security isolation across/within queries
  - Skyrise also inherits the observability issue, rendering debugging and profiling cumbersome
- Skyrise collects a multitude of runtime data
  - Operator and operator step transitions, timings, throughput, costs, ..
  - Aggregate data and make it consumable for debugging or profiling
- Prior experience with cloud (function) services benefitial



Incorporating Distributed Plans into Query Optimization

- FaaS Platforms offer massive parallelism (>10,000s of workers)
- To exploit underlying parallelism, Skyrise optimizer must be aware of data partitioning, distribution, and shuffling
- Extend rule sets for both logical and physical query plans, for now based on heuristics
- Systematically evaluate individual effects and interplay
- Prior experience with cloud services benefitial

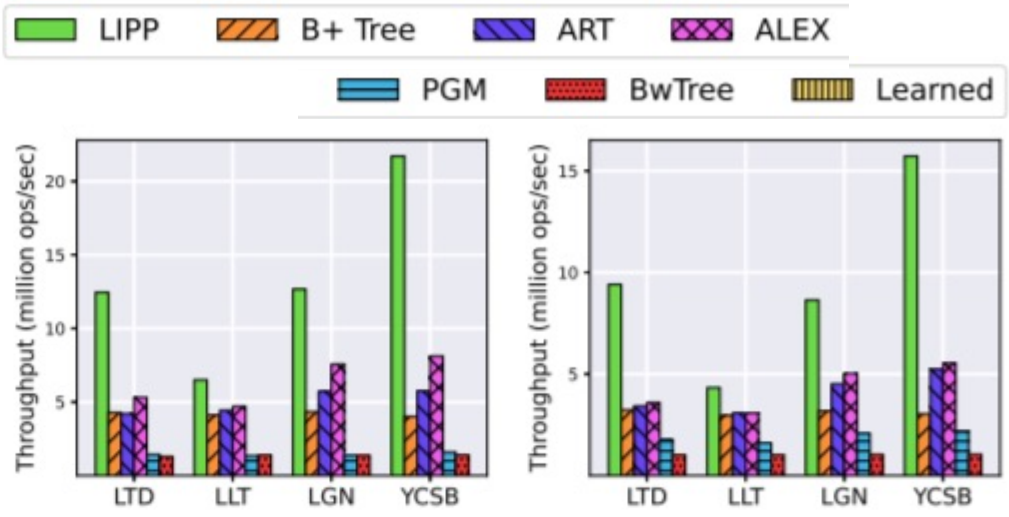




J. Menzler. Query Compilation for Distributed Execution with Cloud Functions. Master's Thesis 2021.
J. Zhou. P. Larson, R. Chaiken. Incorporating Partitioning and Parallel Plans into the SCOPE Optimizer. ICDE 2010.

Learned Indexes on Dynamic Data

- Learned indexes (LIs) with better **performance** than common tree indexes
- LIs supporting **dynamic** data: **PGM**, **ALEX**, **LIPP**
- Datasets (from ALEX paper + new String data)
- Assessment criteria: **index** lookup times, throughput, size; **construction** time + memory



(b) Read-Heavy (c) Write-Heavy

- Tasks:
  - Understand + run dynamic, open source LIs
  - Reproduce results on Integers (compare btree)
  - Extend for further **data types** (e.g. Strings → INT)
  - Select / generate data type-specific datasets
  - Benchmark on String datasets
  - (Stretch: integrate LIs into Hyrise + benchmark)
- Learning potentials:
  - ML-techniques in databases
  - Indexing data
  - Benchmarking
  - (Hyrise index integration)

Wu, J., Zhang, Y., Chen, S., Wang, J., Chen, Y., & Xing, C. Updatable Learned Index with Precise Positions. PVLDB 14(8): 1276-1288 (2021)
Crotty A. Hist-Tree: Those Who Ignore It Are Doomed to Learn. CIDR (2021)

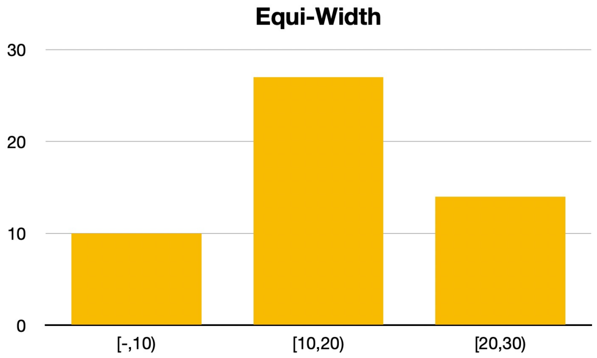
Efficient and Accurate Histograms

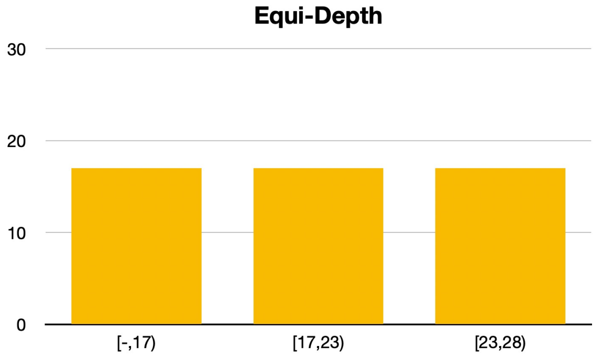
Motivation

- Histograms are database statistics that allow the query optimizer to find efficient and fast query plans
- Improving the accuracy of histograms can have a large positive impact as inefficient query plans are often recognized and avoided
- However, creating and maintaining histograms can be expensive

Current Situation in Hyrise

- Hyrise builds histograms for the entire column
  - Building histograms for a +1TB data set can take hours, even with 240 cores
- The currently used histograms can be inaccurate when data is heavily skewed (often the case in the real world)





Efficient and Accurate Histograms

Goal

- Enable Hyrise to efficiently create histograms for large data sets
- Improve cardinality estimations by using skew-aware histogram types

Evaluation

- Evaluation on synthetic (TPC-H) and real-world (IMDB movie data) data sets
  - What is the accuracy of the evaluated histograms?
  - How efficient is their creation?

Implementation

- Implementation of text book histograms (e.g., equi-width) and max-diff histogram [Hist96]
- Creation of histograms using stable sampling
- Efficient implementation for data sets > 1TB on large server (240 cores and 8 sockets)

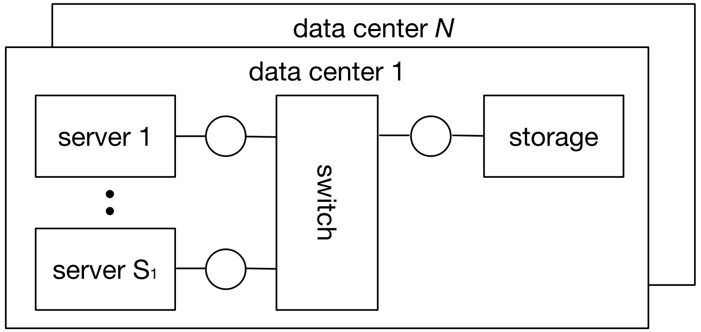
Expected Results

- Thorough implementation and evaluation of different histograms types
- For the histogram type that performs best: efficient and scalable implementation that can ideally find its way to the Hyrise main branch

[Hist96] Viswanath Poosala, Yannis E. Ioannidis, Peter J. Haas, Eugene J. Shekita:
Improved Histograms for Selectivity Estimation of Range Predicates. SIGMOD 1996: 294-305

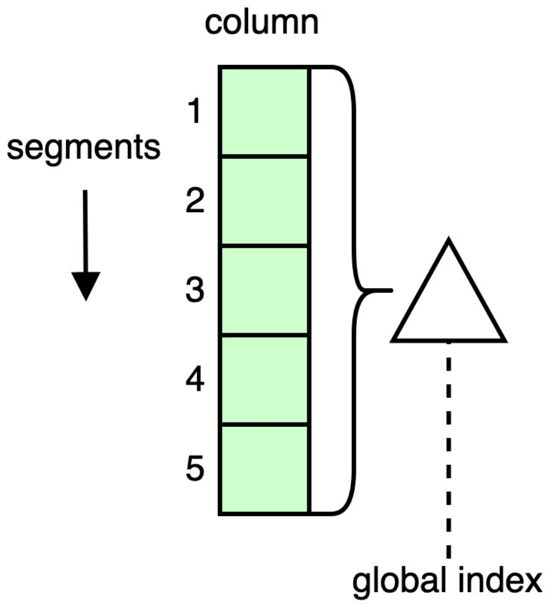
Database Node Placement in the Cloud

- **Motivation**: **Database systems** are increasingly deployed **in the cloud**
- **Problem**: Optimize the **assignment of** (database system) **VMs to physical resources** under **constraints**
  - Problem size: hundreds of servers and thousands of VMs
  - Exemplary VM settings:
    #cores & speed, RAM, storage affinity & anti-affinity rules
  - Exemplary server settings:
    #cores & speed, RAM, connectivity



- **Task**: **Implement and evaluate allocation algorithms** (greedy vs. linear programming based)
- **Learning goals** (specific to this topic):
  - Approaches for solving **optimization problems**, in particular, **linear programming**
  - Characteristics of the architecture of **cloud data centers**

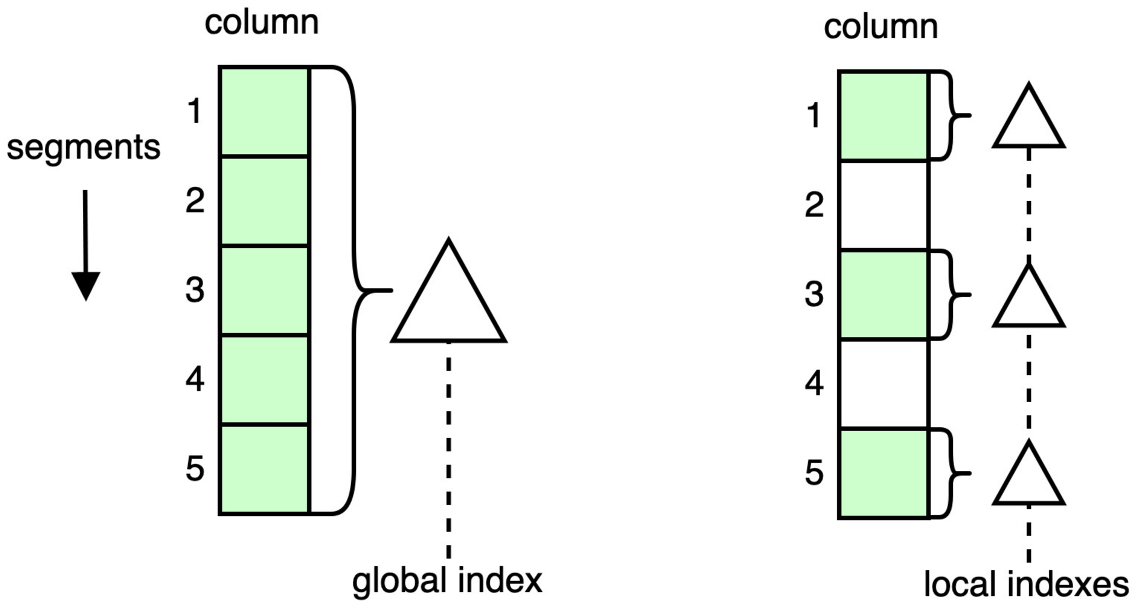
Partial Indexes



Memory Consumption Issue

Indexing all tuples of a table results in a high memory footprint.

Partial Indexes



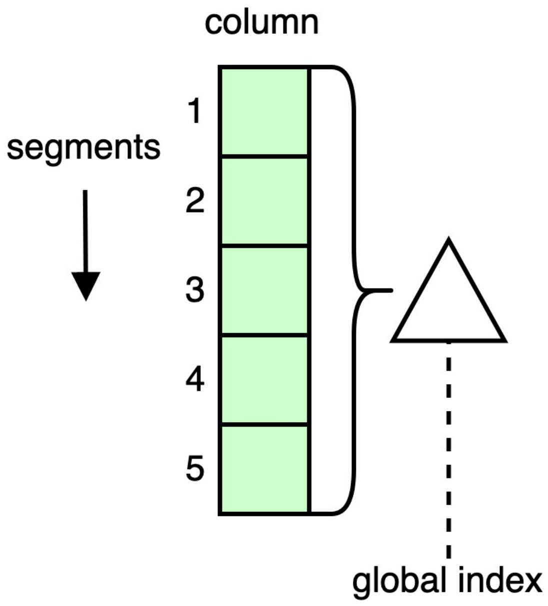
Memory Consumption Issue

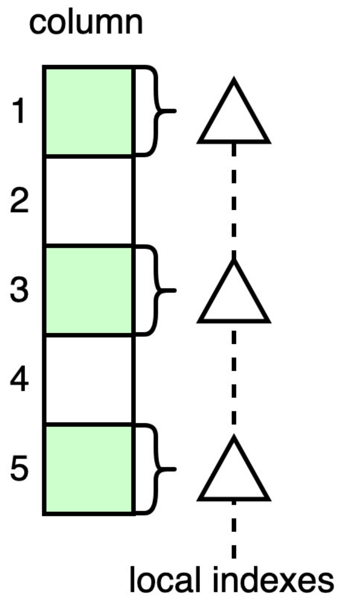
Indexing all tuples of a table results in a high memory footprint.

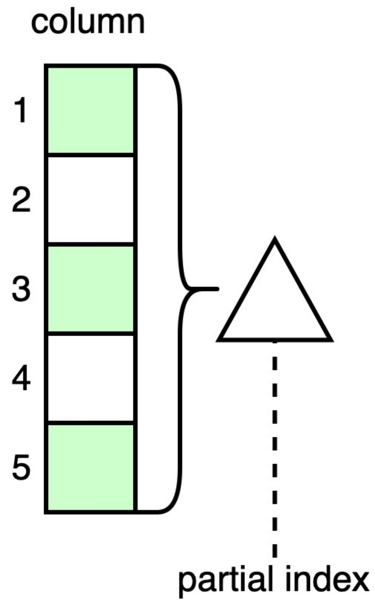
Scalability Issue

The number of lookup operations grows linearly with the number of existing partitions.

Partial Indexes







Memory Consumption Issue

Indexing all tuples of a table results in a high memory footprint.

Scalability Issue

The number of lookup operations grows linearly with the number of existing partitions.

Partial Indexing

- Store index entries of multiple partitions in one global data structure.
- Only a subset of the partitions is indexed.

Hasso Plattner Institut

Partial Indexes

Implementation

- (Partial hash index) – majority implemented in DYOD 21
- Partial B-Tree index
- Index scan operator (currently not compatible with PI)
- Index join operator: fallback join for non-indexed partitions
- Micro benchmarks
- Optimizer rules to use index scans/joins

Evaluation

- Latencies of index lookup operations
- Latencies of index maintenance operations
- Index memory consumption
- Using various benchmarks (micro, TPC-H, JCC-H)
- Performance effects of implemented/modified optimizer rules

Expected Results

- Index implementations (hash and B-Tree)
- Partial index compatible index scan implementation
- Partial index compatible index join implementation
- Optimizer rules to use index scans/joins
- Experimental performance evaluation of partial indexes in comparison to global indexes (used in scans and joins)
- Experimental performance evaluation of Hyrise using the new optimizer rules (using TPC-H and JCC-H)

Dynamic Data Placement Algorithms

Motivation

- Storing data in DRAM allows significantly lower access latencies compared to other data tiers, such as SSDs or HDDs
- DRAM in main-memory databases is limited:
  "[…] the amount of data to be processed keeps growing while DRAM capacity does not" [1]
- To tackle this issue, data can be placed on different data tiers, such as SSDs.

Guiding Questions

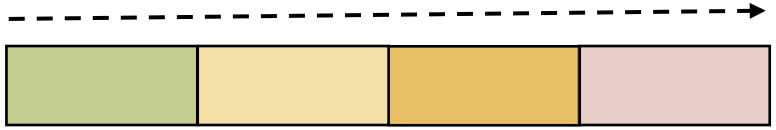
- Which data (structures) should be placed on the slower data tier?
- Given a DRAM budget and a workload, what is an optimal data placement?

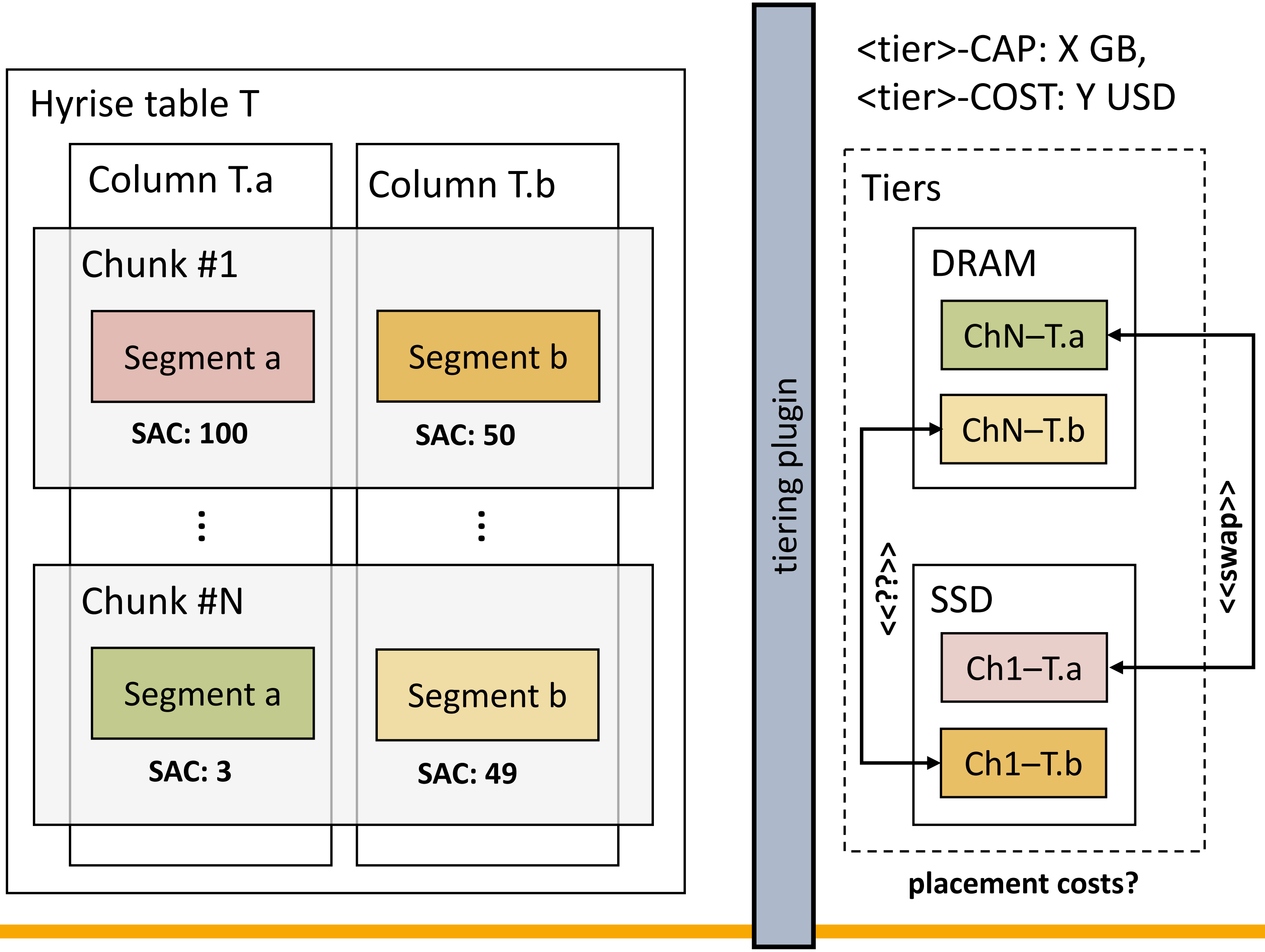
[1] Korolija et al. 2021. *Farview: Disaggregated Memory with Operator Off-loading for Database Engines.* CoRR

Dynamic Data Placement Algorithms

Hyrise automatic tiering

Segment access counter (SAC)
heatmap (low to high)





Dynamic Data Placement Algorithms

Implementation

- Algorithms that determine the optimum data placement for a given workload, having DRAM or SSD as the data tiers (as a Hyrise plugin).
- Micro benchmarks for manual data placement experiments.

Evaluation

- Manual data placements with different shares of segments stored on DRAM/SSD
- TPC-H performance with (a) all segments are stored in main memory, (b) all segments stored on SSD, and (c) segments stored on both DRAM and SSD, according to the developed algorithms
- Metrics: query latencies, memory consumption
- Compare the developed algorithms with a provided reference algorithm

Expected Results

- Different data placement algorithm implementations
- Experimental performance evaluation with segments manually placed on DRAM/SSD
- Experimental performance eval. of the data placement algorithms compared to a reference algorithm

Timeline

Weekly meetings with advisors

25 Oct 2021
Kickoff

31 Oct 2021 by end of day
Submit (your group and) topic preferences

1 Nov 2021
Announcement of topic assignments

Between 1 and 5 Nov 2021
First meeting with your supervisor(s) based on individual arrangement

28 Feb 2022
Final presentations – 20 min + 10 min Q&A

20 Mar 2022
Submission of written report (4 to 8 pages)

Administration

- Specialization areas:
  - ITSE: BPET, OSIS, ITSE-Analyse, ITSE-Maintenance
  - DATA: SCAL
- Official deadline to register was 22 October
- Grading
  - 50% project result and presentation
  - 40% scientific report (4-8 pages ACM format, depending on group size)
  - 10% personal engagement

Bringing groups and topics together

- You are welcome to hang out in this Zoom call after the introduction to figure out groups
- If you have found a topic (and a group), please mail Jan.Kossmann@hpi.de and Daniel.Ritter@guest.hpi.de
  - Include three (or more) topic preferences
  - The assignment algorithm is strategy-proof ;)
- If you have any questions or are still looking for a group partner, please mail us, too

Background reading: https://en.wikipedia.org/wiki/Strategyproofness
https://en.wikipedia.org/wiki/Gale%E2%80%93Shapley_algorithm