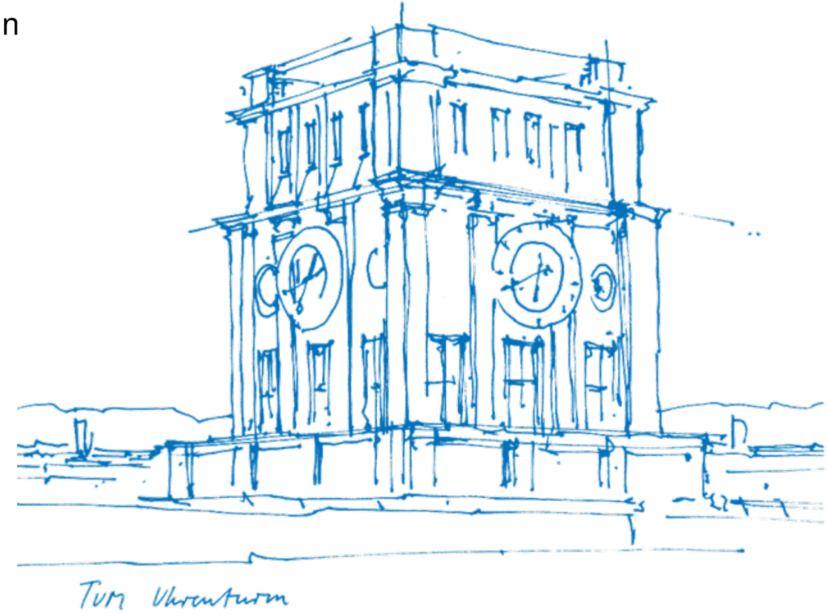
Chair III: Database Systems
Chair XXV: Data Science and Engineering
Department of Informatics
Technical University of Munich

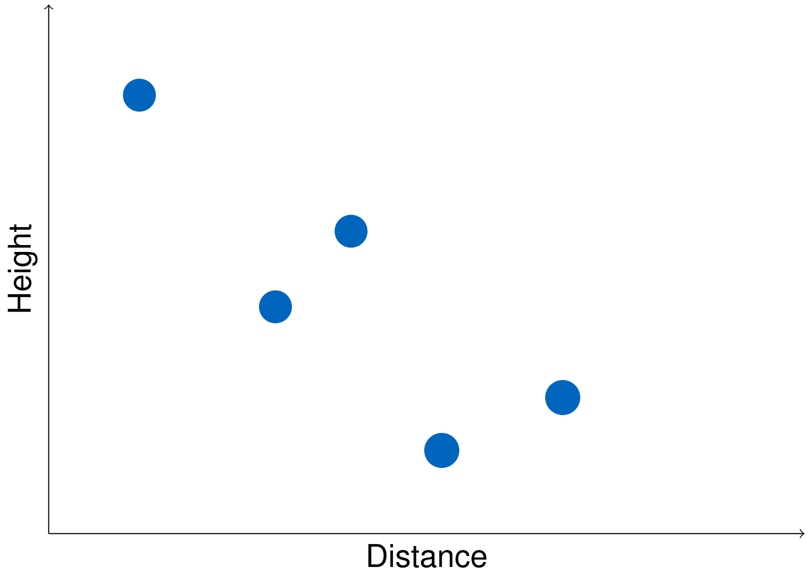
ARTful Skyline Computation for In-Memory Database Systems

Maximilian E. Schüle, Alex Kulikov, Alfons Kemper, Thomas Neumann

Lyon, France, August 26, 2020



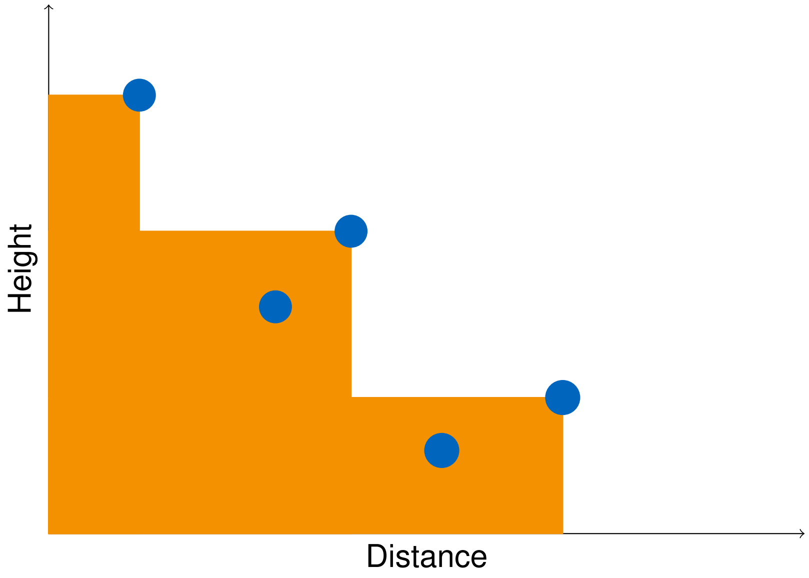
The Skyline-Algorithm



- Skyline algorithm finds interesting tuples within multi-dimensional data sets
- Output: all tuples that are not dominated by any other
- Example: skyline of skyscrapers (height, distance)
- Formally: p dominates q if p is at least as good as q in every dimension, and superior in at least one:

$$p \succ q \Leftrightarrow \forall i \in [n].p[i] \succeq q[i] \wedge \exists j \in [n].p[j] \succ q[j].$$

The Skyline-Algorithm



- Skyline algorithm finds interesting tuples within multi-dimensional data sets
- Output: all tuples that are not dominated by any other
- Example: skyline of skyscrapers (height, distance)
- Formally: p dominates q if p is at least as good as q in every dimension, and superior in at least one:

$$p \succ q \Leftrightarrow \forall i \in [n].p[i] \succeq q[i] \wedge \exists j \in [n].p[j] \succ q[j].$$

Skyline in SQL

```
SELECT * FROM inputtable q WHERE NOT EXISTS (
  SELECT *
  FROM inputtable p
  WHERE p.d1 <= q.d1 AND ... AND p.dn<=q.dn
     AND (p.d1 < q.d1 OR ... OR p.dn<q.dn ))
```

Listing 1: Skyline query in SQL on a table *inputtable* with attributes d_1, ... , d_n.

```
SELECT * FROM inputtable i WHERE ... GROUP BY ... HAVING ...
SKYLINE OF [DISTINCT] d1 [MIN | MAX], ... , dn [MIN | MAX]
ORDER BY ...
```

Listing 2: Skyline extension of SQL: d_1, ... , d_n are the dimensions; *MIN* and *MAX* specify whether each dimension has to be minimised or maximised.

$$p \succ q \Leftrightarrow \forall i \in [n].p[i] \succeq q[i] \wedge \exists j \in [n].p[j] \succ q[j].$$

- Skyline expressible in SQL
- Language extension proposed by Börzsönyi et. al.
- But not integrated in any database system

Skyline Algorithms

- non-categorical algorithms
  - Naive-nested-loops (NNL): progressive output
- categorical algorithms
  - Skyline-using-tree-sorting (ST-S): maintains sorted tree
  - SARTS (Skyline using ART Sorting-based): ST-S + ART

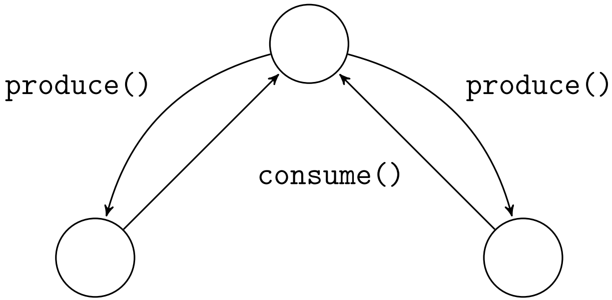
Algorithm 1 Parallel NNL

Input: Tuple List T
Output: Skyline *skyline*

```
parallel_for each tuple t ∈ T do
    is_not_dominated ← True
    for each tuple d ∈ T\{t} do
        if dominates(d, t) then
            is_not_dominated ← False
            break
    if is_not_dominated then
        Add t to skyline
```

Skyline within the Producer-Consumer Concept

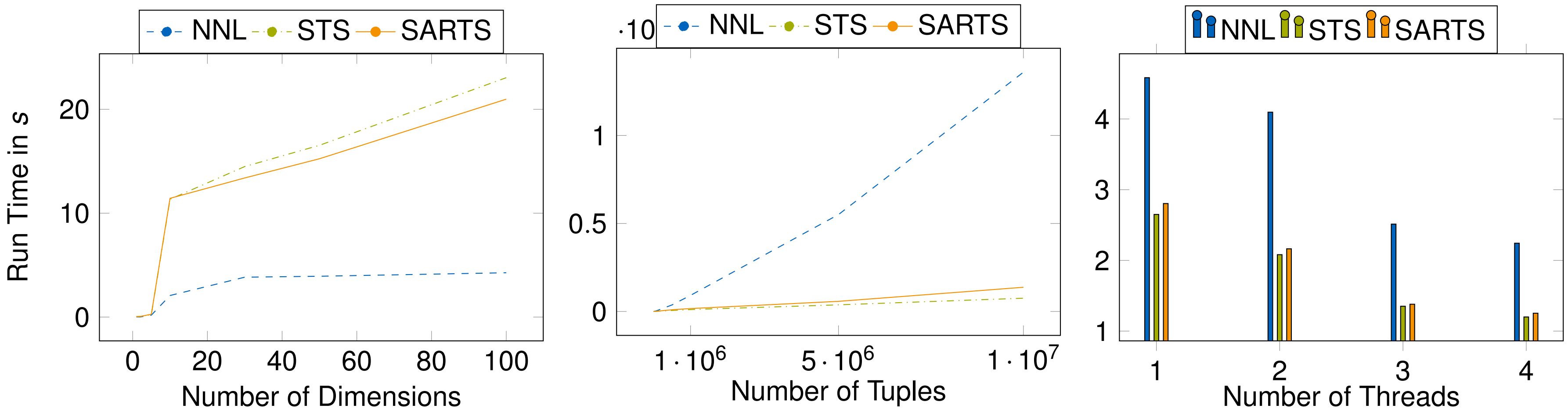
- HyPer: code-generating database system
- Produces LLVM IR (Intermediate Representation)
- Producer-consumer concept:
  - tuples are pushed upwards the target operator
  - progressive output
  - interacts well with naive-nested-loops
- Adaptive-Radix-Tree (ART): index structure, reduces memory consumption



Evaluation: Set-Up

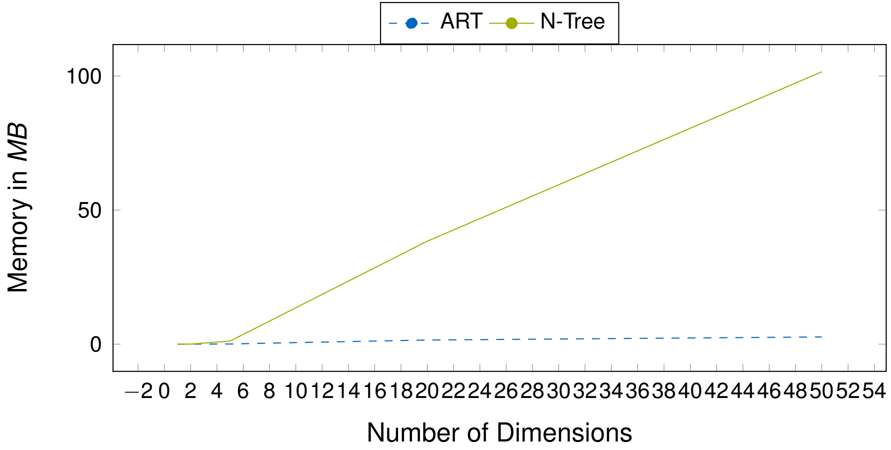
- Linux Mint 18.2 machine
- Intel Core i7-5500U CPU with a 4096 KB cache and 8 GB DDR3L of main-memory.

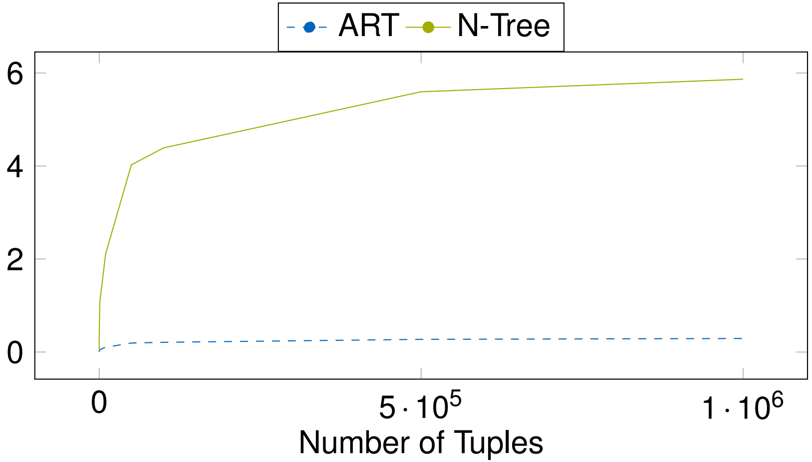
Evaluation: Runtime



- default 5 dimensions, 256 categories, 4 threads and 10,000 input tuples
- SARTS and STS: similar performance

Evaluation: Memory Usage of ART and N-Tree





- 256 categories, 4 threads; left: 1000 input tuples, right: 5 dimensions
- SARTS needs less memory due to the ART

Conclusion

- Progressive Skyline algorithms fit well into the producer-consumer model
- SARTS algorithm using ART reduces overall memory usage in comparison to STS
- Not considered in the measurements: materialisation of tuples

Thank you for your attention!

https://gitlab.db.in.tum.de/alex_kulikov/skyline-computation

https://gitlab.db.in.tum.de/alex_kulikov/skyline-categorical