Skyline query processing

Orestis Gkorgkas

Norwegian University of Science and Technology

August 12, 2015

Outline

Skyline queries

Basic algorithms

Multi-threaded algorithms

Skyline in distributed systems

Skyline on Map-Reduce

Outline

Skyline queries

Basic algorithms

Multi-threaded algorithms

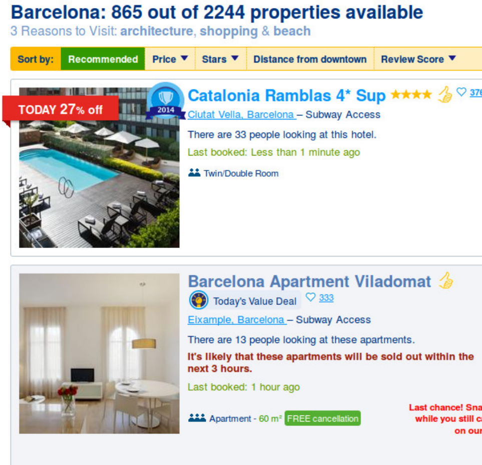
Skyline in distributed systems

Skyline on Map-Reduce

Skyline Queries

Users often have multiple search criteria

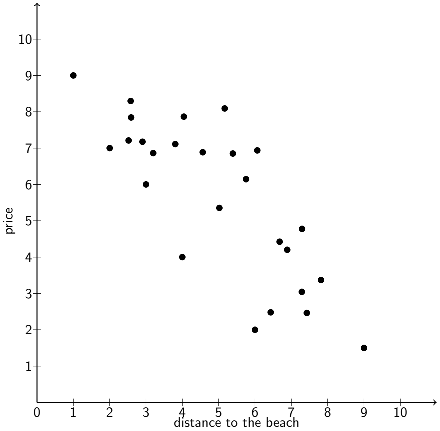
- Price
- Location
- Cleanliness
- Stars
- Overall quality



Skyline Queries

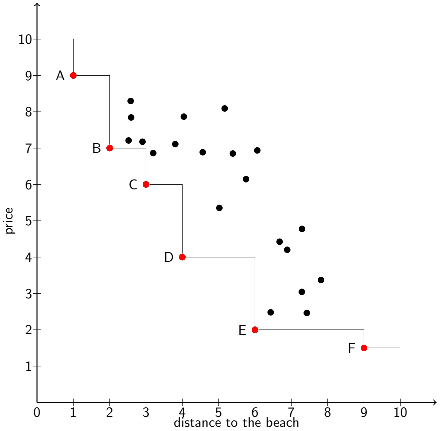
- Items are usually ranked according to a single feature
- Users get overwhelmed by the number of all possible options
  - It becomes difficult to find the item that suits their needs
- **Usually there is no single best item**
- **But there are many options that are definitely worse than others**
- How can we keep only the good options?
  - Skyline queries

Skyline



- If a point **p** is no worse at all dimensions than a point **q** and at least better in one, then **p dominates q**
- The skyline consists of all points that are not dominated by any other point

Skyline



- If a point **p** is no worse at all dimensions than a point **q** and at least better in one, then **p dominates q**
- The skyline consists of all points that are not dominated by any other point

Why skyline queries are important?

- Data exploration
  - They provide an overview of the available data
  - They include good options for any use preference
- Multi-criteria decision analysis
  - Especially useful when client's processing resources are limited
- Route planning
  - Evaluation of multiple traveling costs

Skyline queries variations

- Constraint skyline queries
  - Constrains are imposed to the range of some of the dimensions
  - e.g. Skyline for the hotels are that are not further than 500m from beach
- Skylines over a subset of dimensions
- Dynamic skylines
  - Each dimension corresponds to a function
  - e.g. Restaurants that are the closest to my current position their closing time is as later as possible from current time
- Skylines over joins
- Skyband queries
- ...

Skyline queries over different settings

- Centralized processing
  - Single-thread algorithms
    - Index-free
    - Index-based
  - Multi-threaded algorithms
  - Skylines on GPU
- Distributed processing
  - Peer-to-peer systems
  - Skyline on Map-Reduce

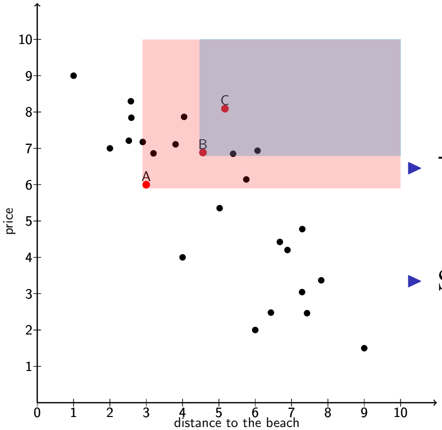
A naive algorithm

- Pairwise comparison of all points
- Can be implemented as a nested SQL query
- High complexity ($O(n^2)$)

```
SELECT * FROM HOTELS outer
WHERE outer NOT EXISTS (
SELECT * FROM HOTELS inner
inner.distance <=outer.distance AND
inner.price <= outer.price AND
(inner.distance <outer.distance OR
inner.price < outer.price
));
```

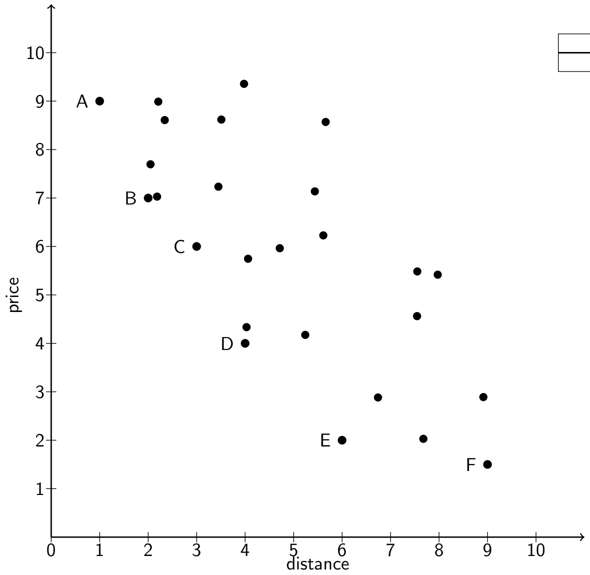
Outline

Properties of skylines



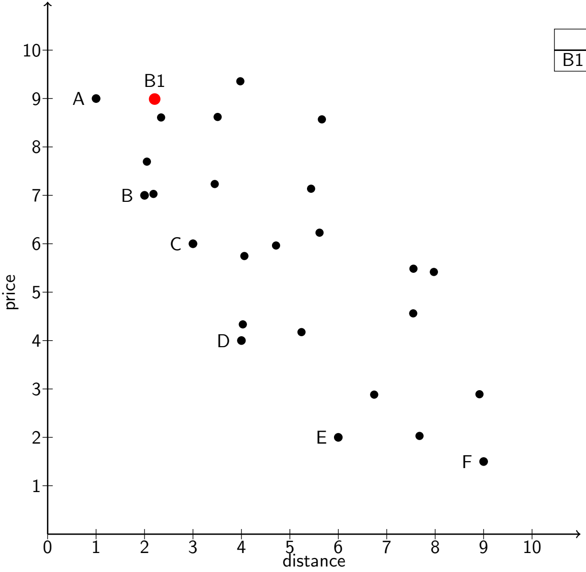
- Transitivity
  - If A **dominates** B and B **dominates** C $\Rightarrow$ A **dominates** C
- $\mathrm{Sk}(A_1 \bigcup A_2) = \mathrm{Sk}(\mathrm{Sk}(A_1) \bigcup \mathrm{Sk}(A_2))$

Block Nested Loops



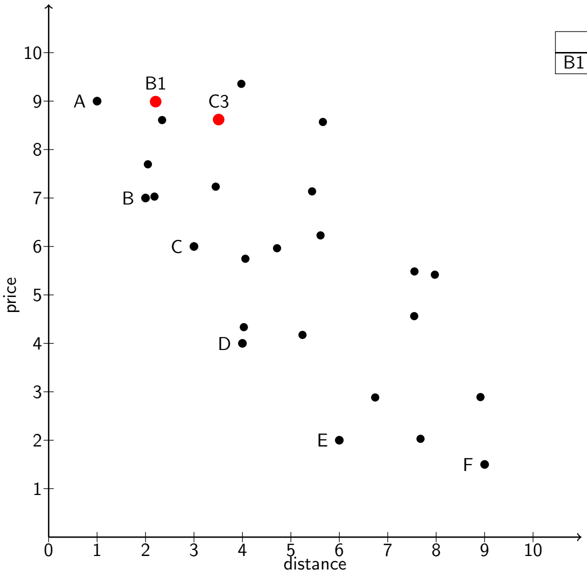
Window		

Block Nested Loops



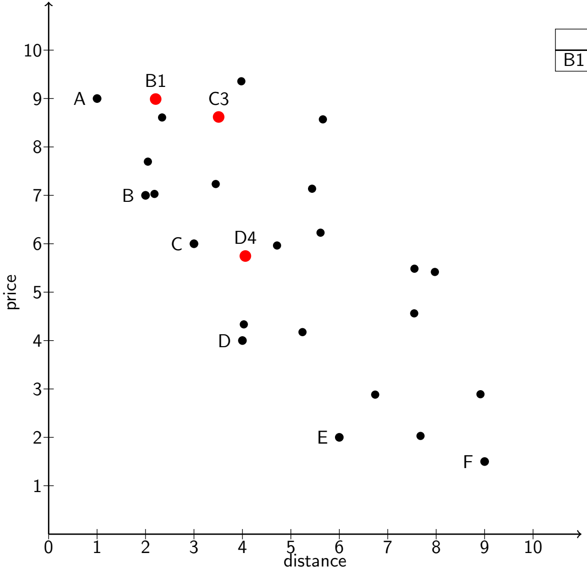
Window		
B1		

Block Nested Loops



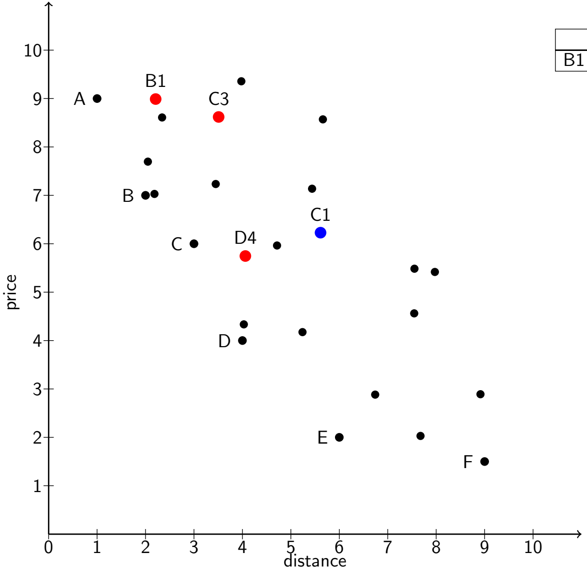
Window		
B1	C3	

Block Nested Loops



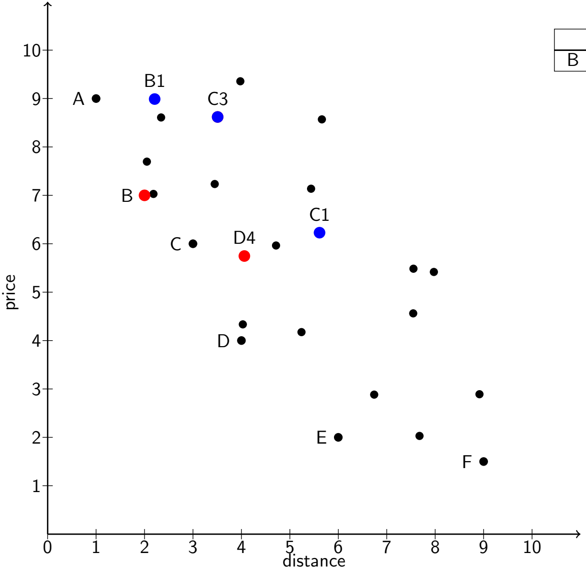
Window		
B1	C3	D4

Block Nested Loops



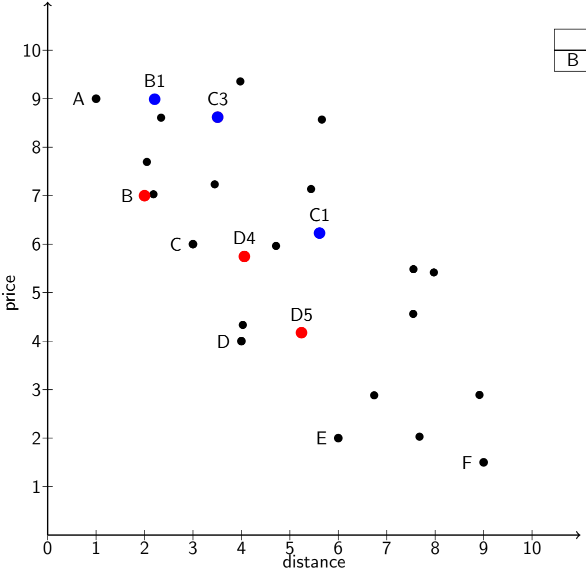
Window		
B1	C3	D4

Block Nested Loops



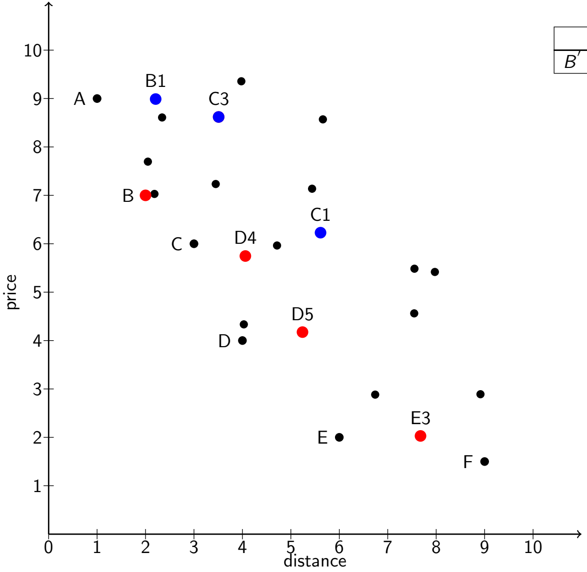
Window		
B		D4

Block Nested Loops



Window		
B	D5	D4

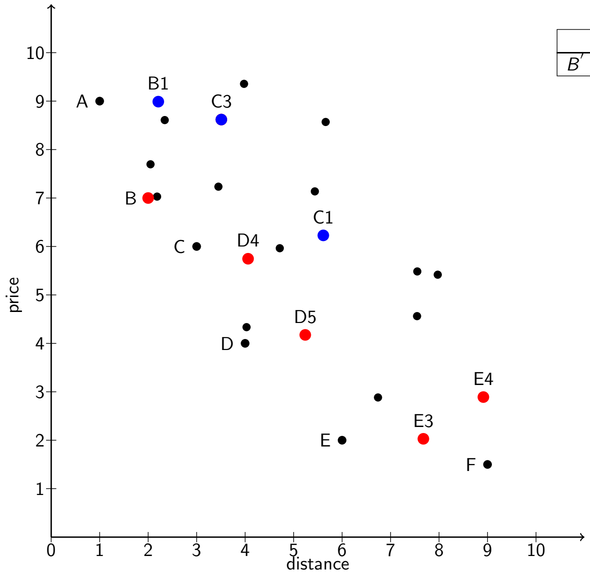
Block Nested Loops



Window		
B'	$D5'$	$D4'$

Temp. file
E3

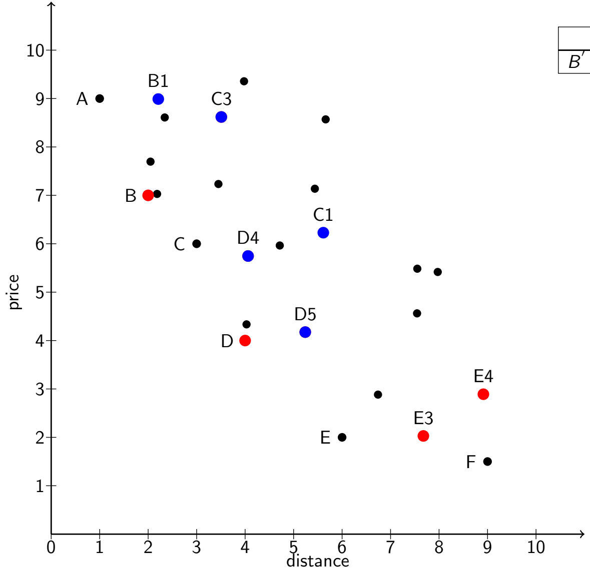
Block Nested Loops



Window		
B'	$D5'$	$D4'$

Temp. file
E3
E4

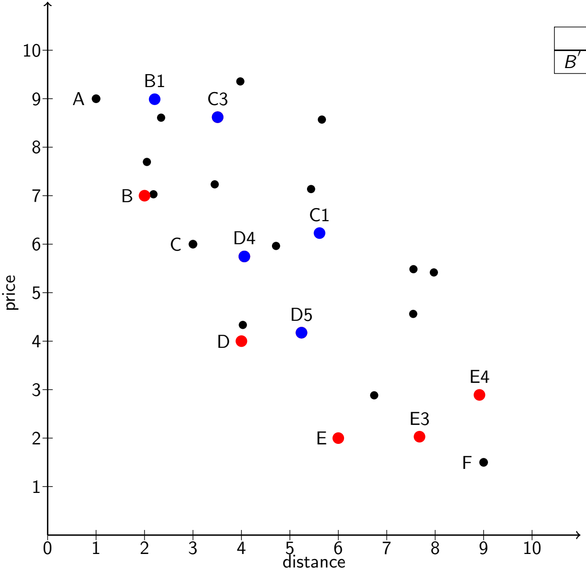
Block Nested Loops



Window		
B'	D	

Temp. file
E3
E4

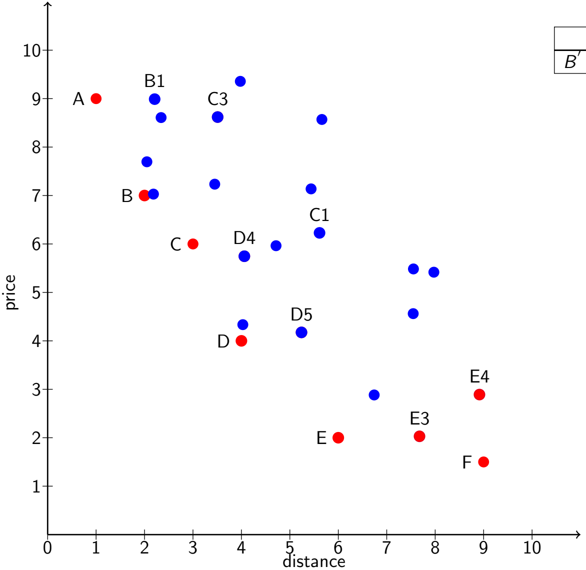
Block Nested Loops



Window		
B'	D	E

Temp. file
E3
E4

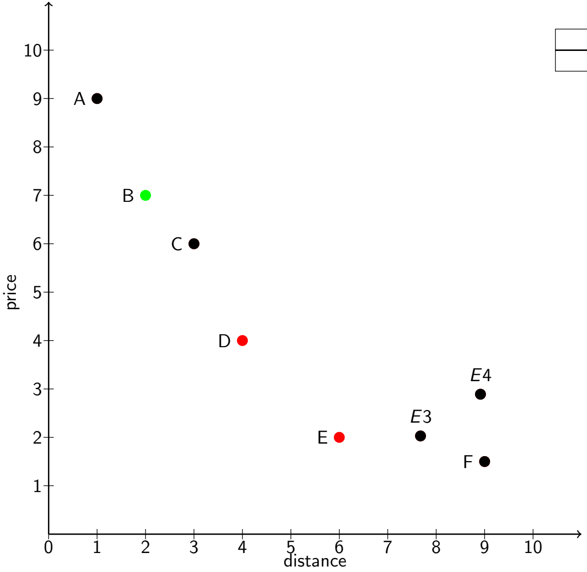
Block Nested Loops



Window		
B'	D	E

Temp. file
E3
E4
A
C
F

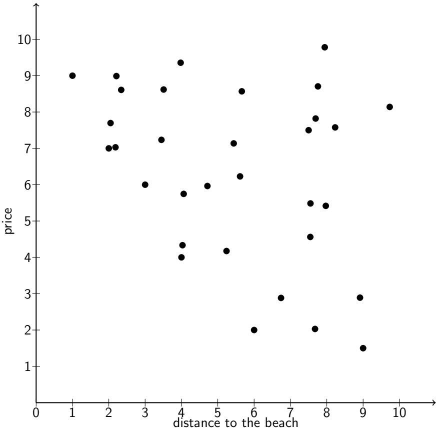
Block Nested Loops



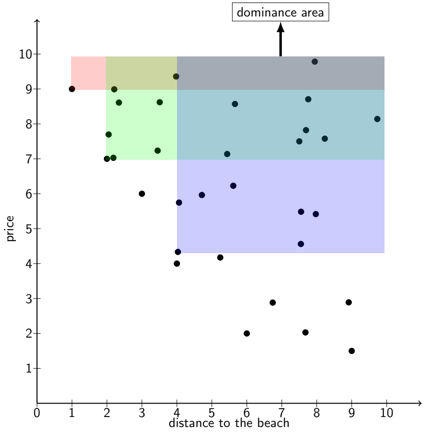
Window		
	D	*E*

Temp. file
E3
E4
A
C
F

How can we improve the performance?

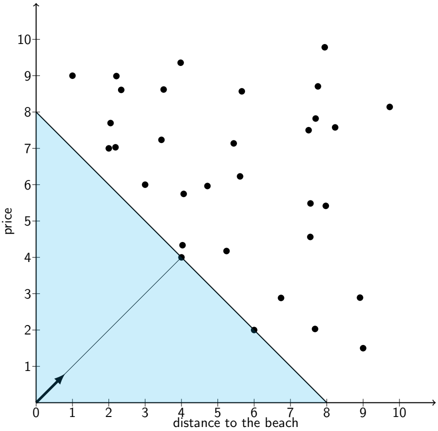


How can we improve the performance?



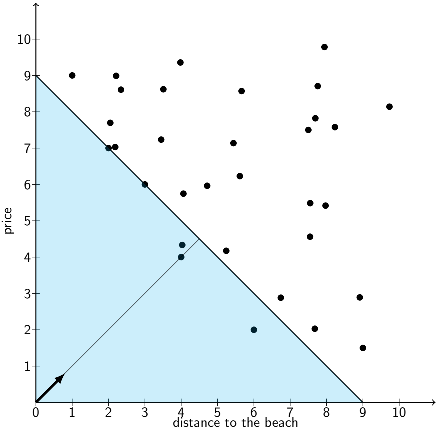
- Select points with **large dominance area**
  - Put “killer” points first in the BNL window

How can we improve the performance?



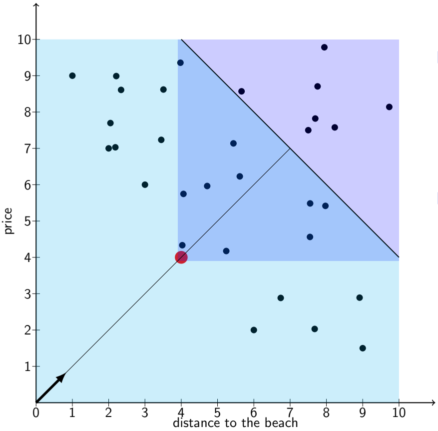
- Select points with **large dominance area**
  - Put “killer” points first in the BNL window
- Perform a sorted access based on a **scoring function**
  - Skyline points always appear before the points they dominate for any monotonic function
  - We do not need to process all objects

How can we improve the performance?



- Select points with **large dominance area**
  - Put “killer” points first in the BNL window
- Perform a sorted access based on a **scoring function**
  - Skyline points always appear before the points they dominate for any monotonic function
  - We do not need to process all objects

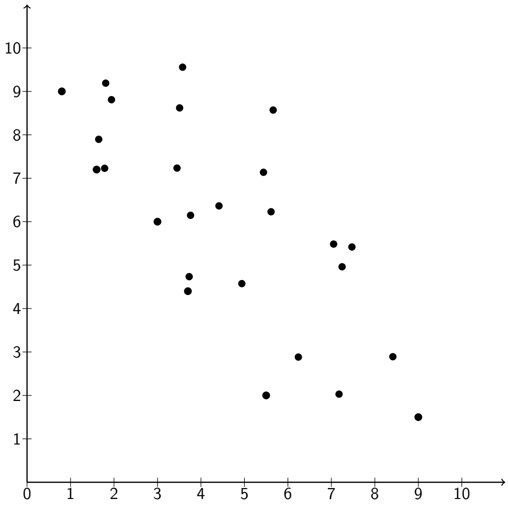
How can we improve the performance?



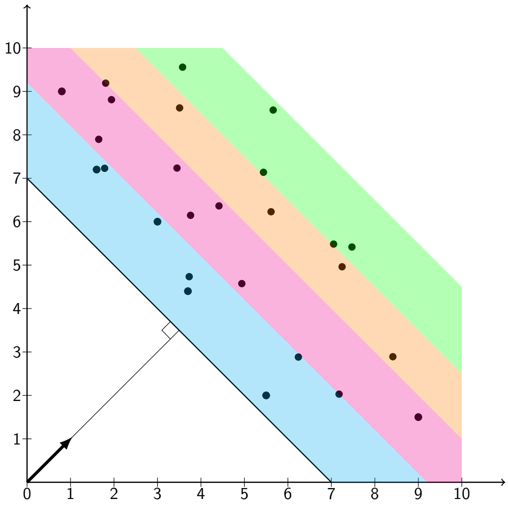
- Select points with **large dominance area**
  - Put "killer" points first in the BNL window
- Perform a sorted access based on a **scoring function**
  - Skyline points always appear before the points they dominate for any monotonic function
  - We do not need to process all objects

Outline

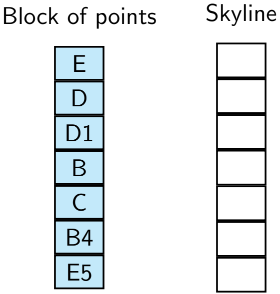
Multi-threaded processing



Multi-threaded processing



Multi-threaded processing



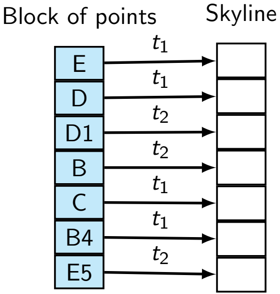
Step 1: Each point is compared independently against the skyline

- If dominated by the skyline it is marked for removal

Step 2: Each surviving point is compared against the points with better score

- Some unnecessary comparisons can be made

Multi-threaded processing



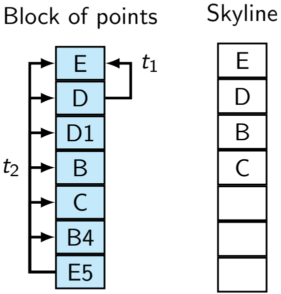
Step 1: Each point is compared independently against the skyline

- If dominated by the skyline it is marked for removal

Step 2: Each surviving point is compared against the points with better score

- Some unnecessary comparisons can be made

Multi-threaded processing



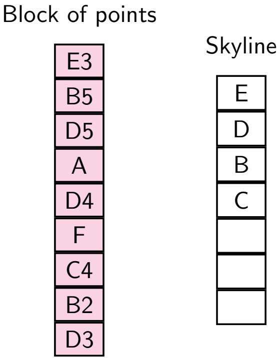
Step 1: Each point is compared independently against the skyline

- If dominated by the skyline it is marked for removal

Step 2: Each surviving point is compared against the points with better score

- Some unnecessary comparisons can be made

Multi-threaded processing



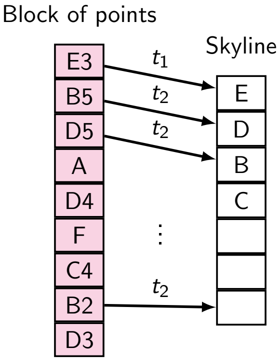
Step 1: Each point is compared independently against the skyline

- If dominated by the skyline it is marked for removal

Step 2: Each surviving point is compared against the points with better score

- Some unnecessary comparisons can be made

Multi-threaded processing



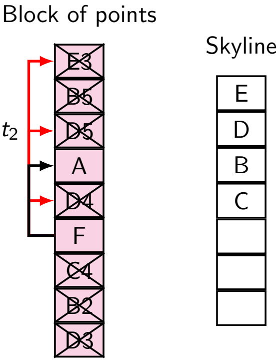
Step 1: Each point is compared independently against the skyline

- If dominated by the skyline it is marked for removal

Step 2: Each surviving point is compared against the points with better score

- Some unnecessary comparisons can be made

Multi-threaded processing



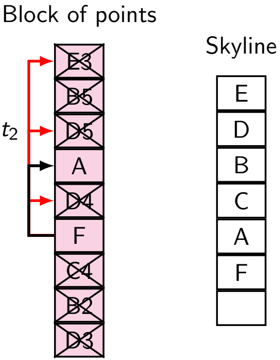
Step 1: Each point is compared independently against the skyline

- If dominated by the skyline it is marked for removal

Step 2: Each surviving point is compared against the points with better score

- Some unnecessary comparisons can be made

Multi-threaded processing



Step 1: Each point is compared independently against the skyline

- If dominated by the skyline it is marked for removal

Step 2: Each surviving point is compared against the points with better score

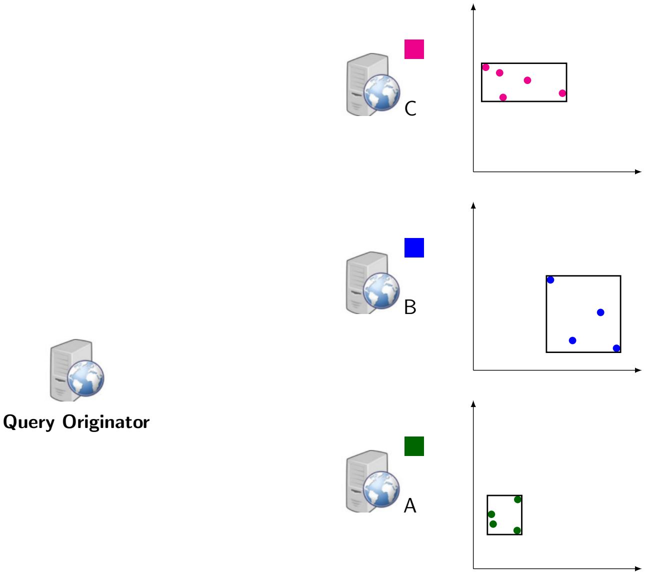
- Some unnecessary comparisons can be made

Outline

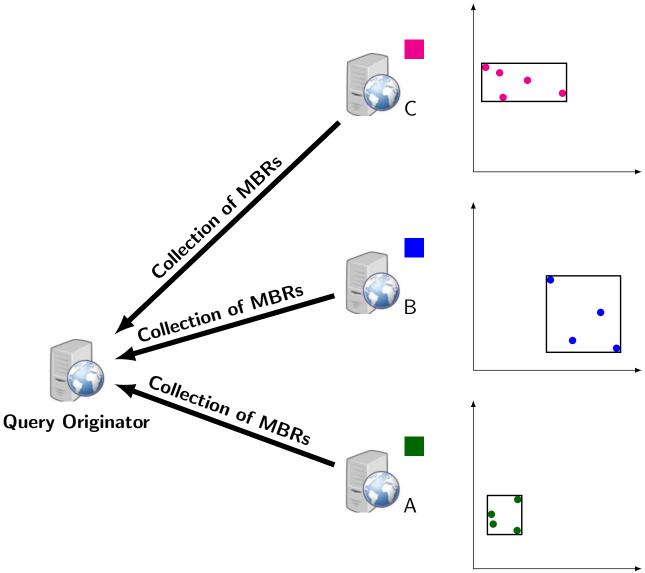
Properties of a good distributed algorithm

- Low latency - Short response time
- Parallelism
- Minimize network traffic
  - Avoid accessing unnecessary nodes
  - Minimize the transfered data
- Minimize local processing time

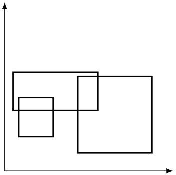
Distributed skyline queries



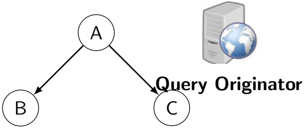
Distributed skyline queries

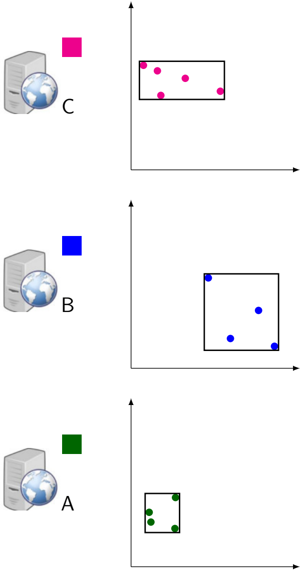


Distributed skyline queries

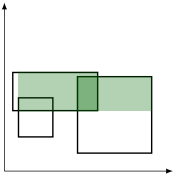


Server makes an execution plan

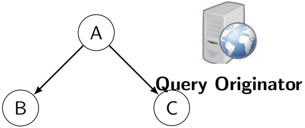


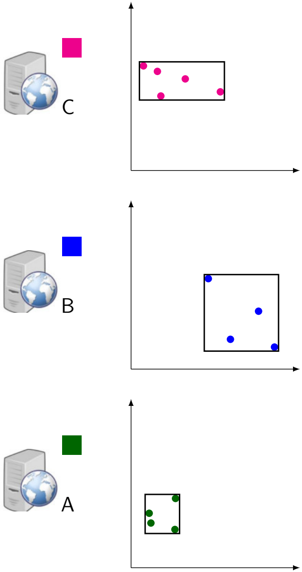


Distributed skyline queries

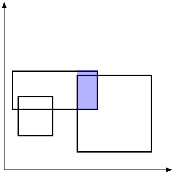


Server makes an execution plan

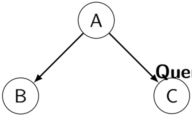




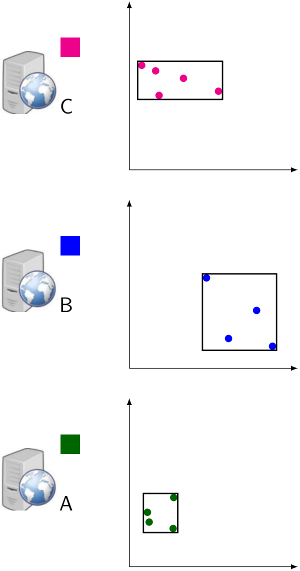
Distributed skyline queries



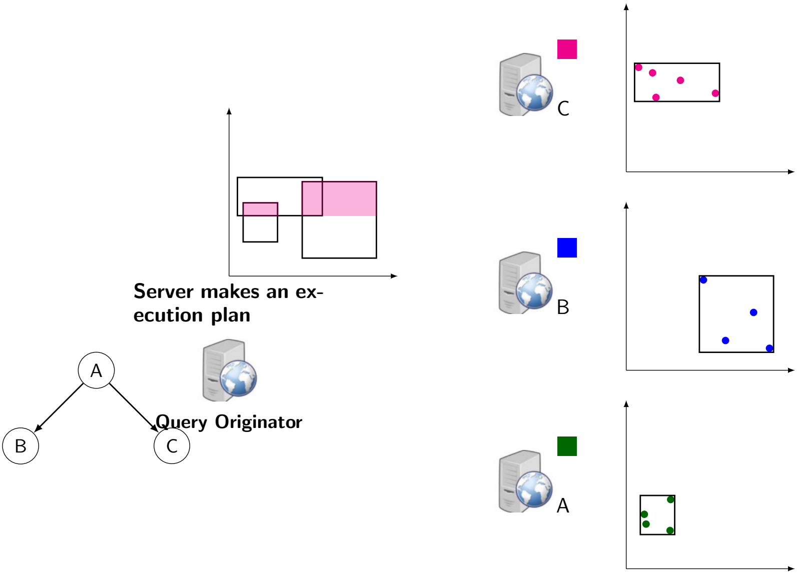
Server makes an execution plan



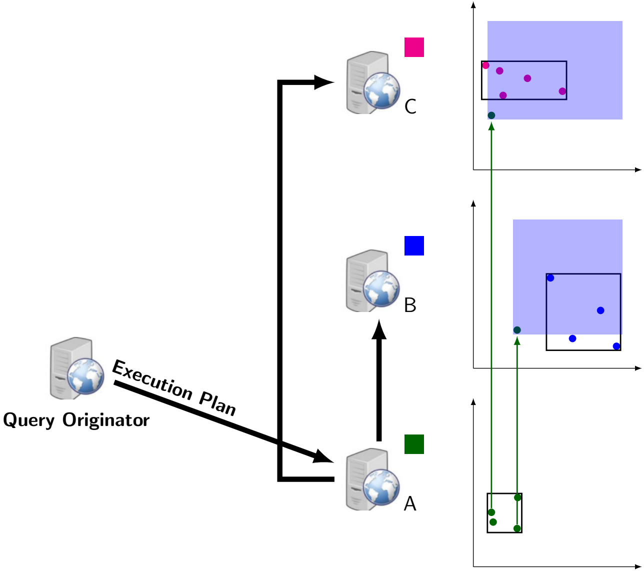
Query Originator



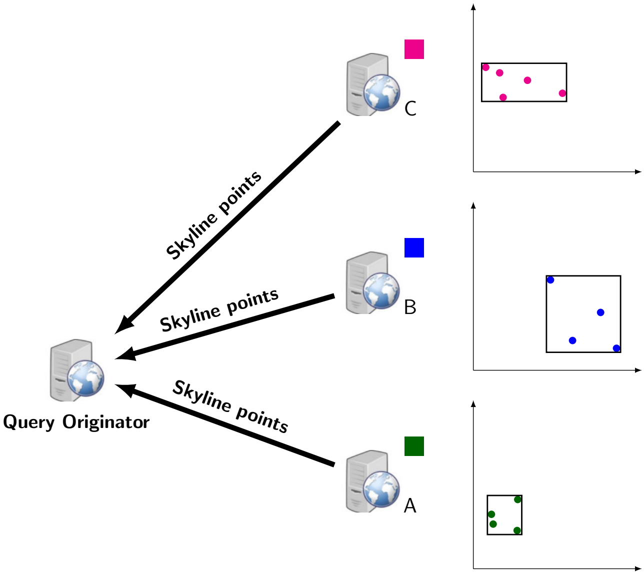
Distributed skyline queries



Distributed skyline queries



Distributed skyline queries

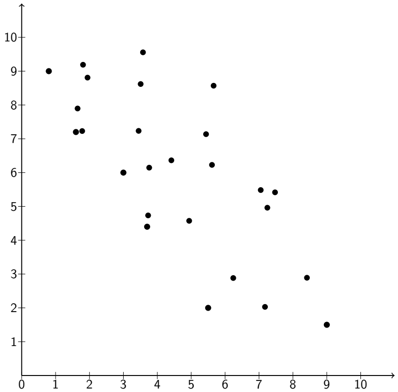


Outline

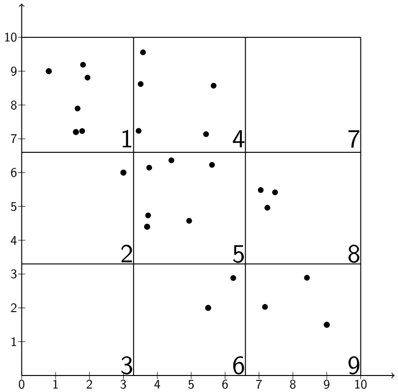
Skyline on Map-Reduce

- Every job is split into a Map and a Reduce phase
  - Map(k_1, v_1)$\rightarrow$ list(k_2, v_2)
  - Reduce($k_2, list(v_2)$) $\rightarrow$ List(v_3)
- The mappers do not communicate with each other

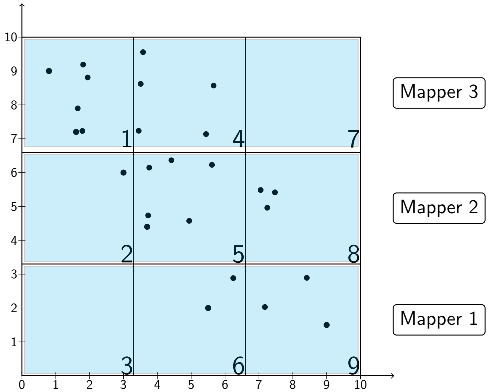
Skyline on Map-Reduce



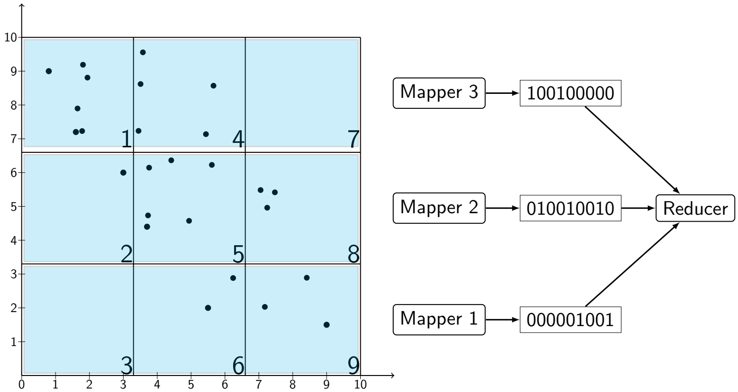
Skyline on Map-Reduce



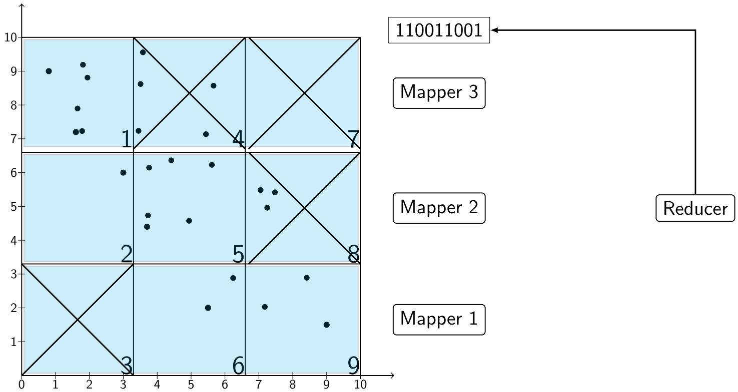
Skyline on Map-Reduce



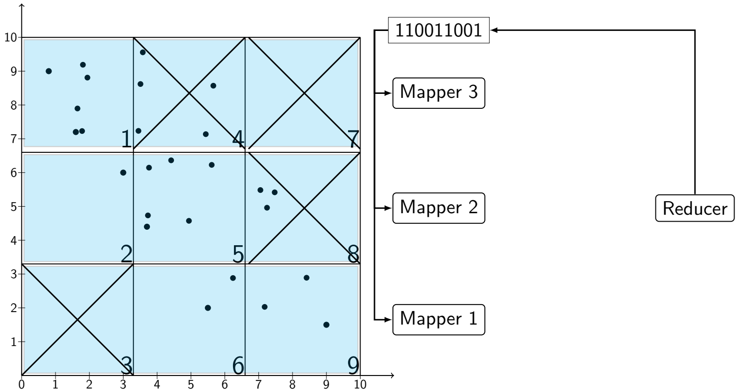
Skyline on Map-Reduce



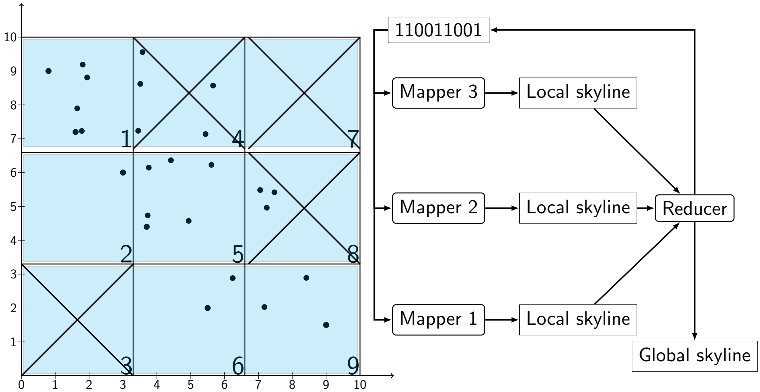
Skyline on Map-Reduce



Skyline on Map-Reduce



Skyline on Map-Reduce



Conclusion

- Skyline queries return not dominated points
- Efficiency can be improved in many ways
  - Presorting and filtering
  - Indexes
    - Indexes cannot always be used
    - e.g. Data streams and custom user queries
  - Parallelization
    - Multi-threading
    - Distributed systems
- There are many aspects we did not cover

Thank you

Thank you for your attention!

References

Ilaria Bartolini, Paolo Ciaccia, and Marco Patella. Efficient sort-based skyline evaluation. *ACM Trans. Database Syst.*, 33(4), 2008.

Stephan Borzsonyi, Donald Kossmann, and Konrad Stocker. The skyline operator. In *Proceedings of ICDE*, pages 421–430, 2001.

Sean Chester, Darius Sidlauskas, Ira Assent, and Kenneth S. Bøgh. Scalable parallelization of skyline computation for multi-core processors. In *31st IEEE International Conference on Data Engineering, ICDE 2015, Seoul, South Korea, April 13-17, 2015*, pages 1083–1094, 2015.

Jan Chomicki, Parke Godfrey, Jarek Gryz, and Dongming Liang. Skyline with presorting: Theory and optimizations. In *In Proceedings of IIWPM*, pages 595–604, 2005.

Katja Hose and Akrivi Vlachou. A survey of skyline processing in highly distributed environments. *VLDB J.*, 21(3):359–384, 2012.

Kasper Mullesgaard, Jens Laurits Pederseny, Hua Lu, and Yongluan Zhou. Efficient skyline computation in mapreduce. In *In Proceedings of EDBT*, pages 37–48, 2014.

Dimitris Papadias, Yufei Tao, Greg Fu, and Bernhard Seeger. Progressive skyline computation in database systems. *ACM Trans. Database Syst.*, 30(1):41–82, 2005.

João B. Rocha-Junior, Akrivi Vlachou, Christos Doulkeridis, and Kjetil Nørvåg. Efficient execution plans for distributed skyline query processing. In *In Proceedings of EDBT*, pages 271–282, 2011.

Shiyuan Wang, Quang Hieu Vu, Beng Chin Ooi, Anthony K. H. Tung, and Lizhen Xu. Skyframe: a framework for skyline query processing in peer-to-peer systems. *VLDB J.*, 18(1):345–362, 2009.

Shiming Zhang, Nikos Mamoulis, and David W. Cheung. Scalable skyline computation using