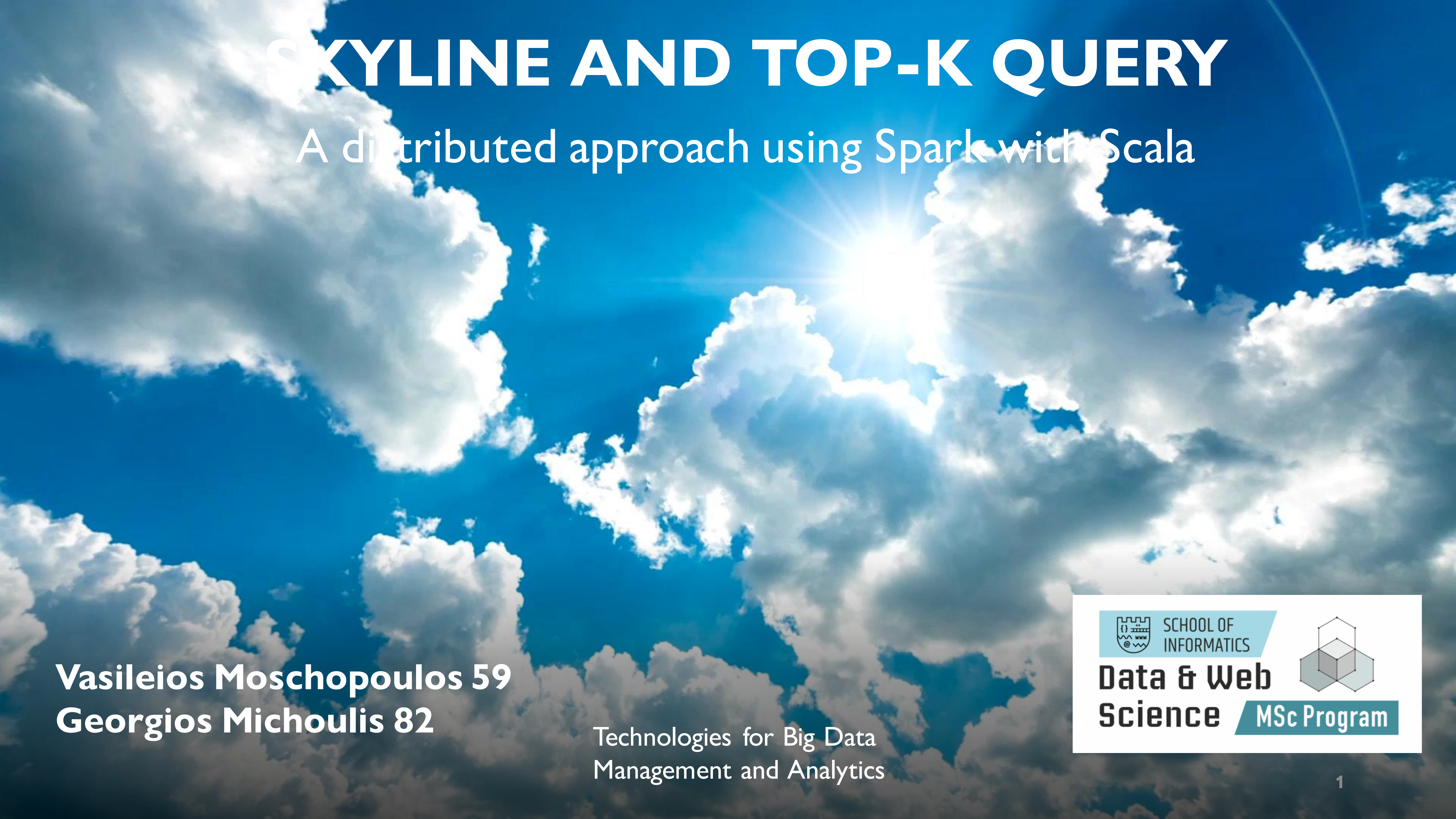


SKYLINE AND TOP-K QUERY

A distributed approach using Spark with Scala

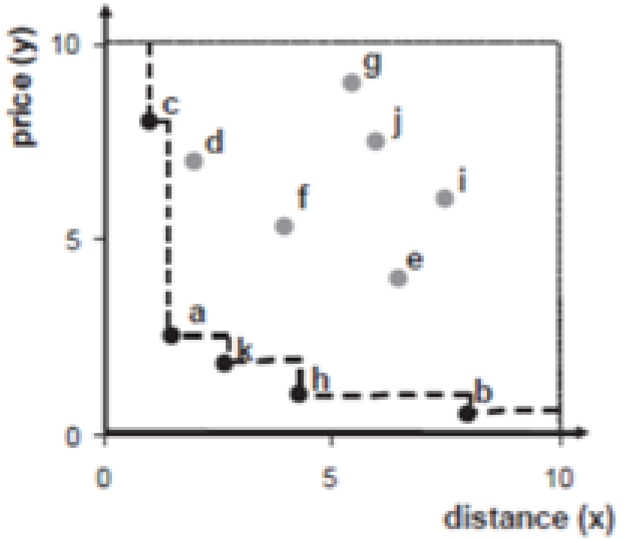
Vasileios Moschopoulos 59
Georgios Michoulis 82

Technologies for Big Data Management and Analytics

index

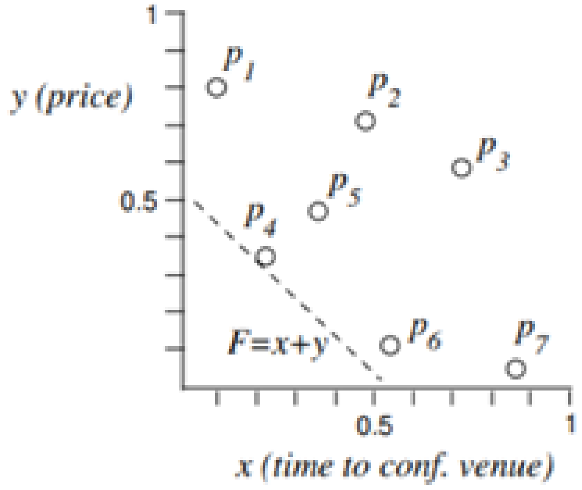
Skyline

- The Skyline operator is the subject of an optimization problem, used in a query to filter results from a database to keep only those objects that are not worse than any other.
- The skyline is a subset of all tuples that are not dominated by any other tuple of the original set. A tuple a dominates another tuple b (a < b) when the values of either of the attributes are greater than or equal to the corresponding values of b.
- The user wants the hotel to be both cheap and close to the beach. However, hotels that are close to the beach may also be expensive. Skyline operator would only present those hotels that are not worse than any other hotel in both price and distance to the beach.
- These two characteristics are most probably highly anticorrelated.



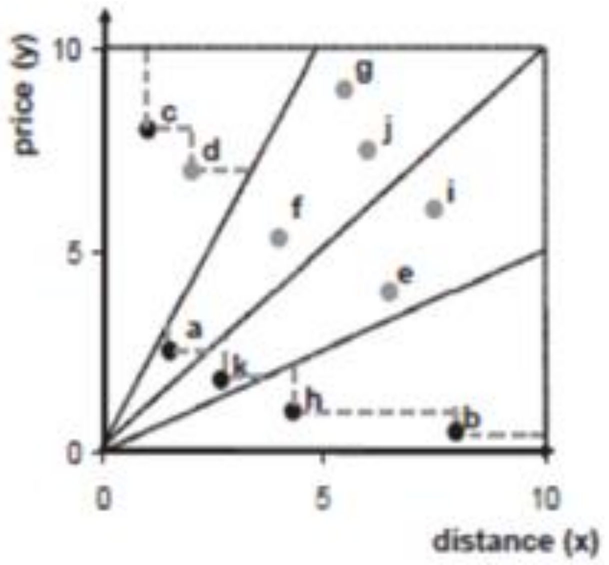
Top-K

- The top-k dominating query was first introduced by Papadias et al. as an extension of the skyline query.
- Provided a set S of d-dimensional objects, the top-k dominating (TKD) query ranks the objects o in S based on the number of objects in S dominated by o and returns the k objects from S that dominate the maximum number of objects.
- For example, when choosing **a hotel** that best fits our needs, it is not easy to merge the "**price**" and "**distance from the beach**" attributes.
- The top-2 hotels for the F = x + y ranking function are p4 and p6. A clear value of the top-k question is that the user can monitor the number of outcomes.



Angular Partitioning

- The Skyline points of the data are a subset of the union of the Skyline points of all the subdivisions.
- The angular partitioning technique converts the Cartesian coordinates of a data set into super spherical space and divides the data space into the desired number of segments according to their angular coordinates.
- The benefits of this approach are the equal distribution of data that is close to the start of the axes. The points that are distributed at the beginning of the axes have a very high likelihood of belonging to the final ridge.
- The Cartesian coordinates for a point x = [x1, x2, ….,xd] converted into a radial coordinate (r) and d-1 are the angular coordinated (ϕi),



$$r = \sqrt{x_n^2 + x_{n-1}^2 + \cdots + x_1^2}$$

$$\tan(\phi_1) = \frac{\sqrt{x_n^2 + x_{n-1}^2 + \cdots + x_2^2}}{x_1}$$

$$\cdots$$

$$\tan(\phi_{d-2}) = \frac{\sqrt{x_n^2 + x_{n-1}^2}}{x_{n-2}}$$

$$\tan(\phi_{d-1}) = \frac{x_n}{x_{n-1}}$$

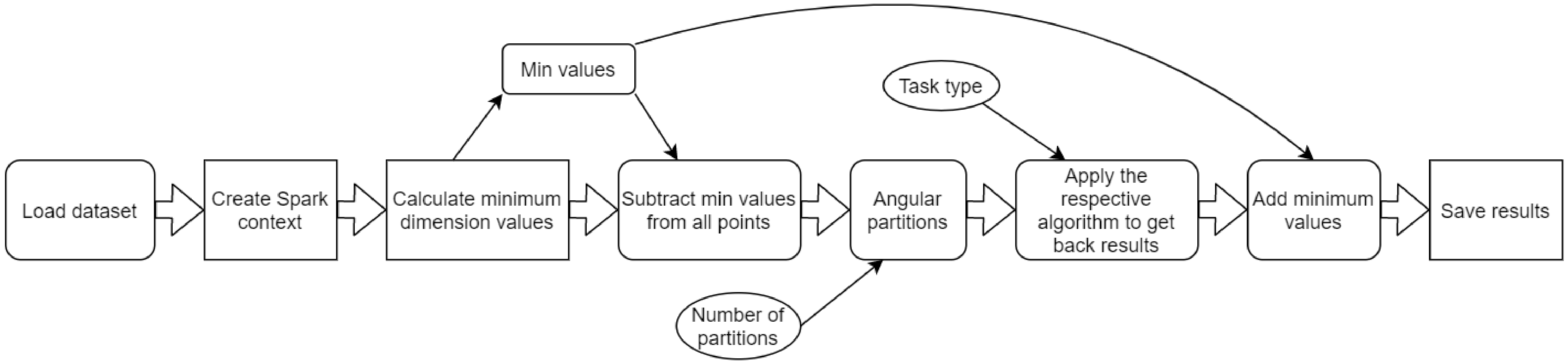
Tasks and data

- We use Spark with Scala and Resilient Distributed Datasets (RDDs) to address three challenges
  1. The calculation of skyline points within a multidimensional dataset
  2. Locating the k points with the highest domination score, i.e., the points that dominate the most over other points
  3. A combination of the previous two challenges, where we aim to locate the skyline points with the highest domination score

- Our data was generated using Python with NumPy
  - Distribution types: Gaussian, uniform, correlated, anticorrelated
  - Dimensionality: 2, 3, 5, 8
  - Cardinality: 1000, 10000, 100000, 1000000

Base procedure

- Parallelization:
  1. Algorithms / min value function are applied to all partitions
  2. Partitions are reduced into one partition
  3. The algorithms / min value function are then applied again to extract the final results / min values



Load dataset
Create Spark context
Calculate minimum dimension values
Min values
Subtract min values from all points
Angular partitions
Number of partitions
Task type
Apply the respective algorithm to get back results
Add minimum values
Save results

Angular Partitioner object

- Spark Partitioner object
- Given n number of 90 degree angles, splits the first n-1 angles into three slices and the final one into either two or three slices. The remaining angles are left as one slice.
- Dataset dimension dim cannot be smaller than 2
  Number of partitions should be a power of 3 multiplied by 2 or 1 and no larger than 3^{dim-1}
- $partition_x = a_1 + m_1 a_2 + m_1 m_2 a_3 + \ldots + P_{i=1}^{n-1} m_i * a_n$
  a_i is the slice of i-th angle the point x belongs to, m_i is the number of slices the i-th angle is split into.
  Slice and partition enumeration begin from 0.
- Example:
  3-dimensional point angle coordinates (26.5, 57.6), 9 partitions / 3 slices per 90-degree angle
  $26.5 < 30$, slice 0 of ϕ_1
  $30 < 57.6 < 60$ slice 1 of ϕ_2
  partition $= 0 + 3 * 1 = 3$
- Point angles are calculated using the inverse tangent function

Angular Partitioner object

- The points are coupled into tuples with their respective partition keys
- The partitioner uses the partition keys to distribute the points to their respective partition
- Drawbacks:
    - Some partitions can be left empty, depends on the distribution.

      We counter this by filling empty partitions with infinity value points.
    - Reduced parallelization speed due to this with dense partitions
    - Not all distributions do this equally!

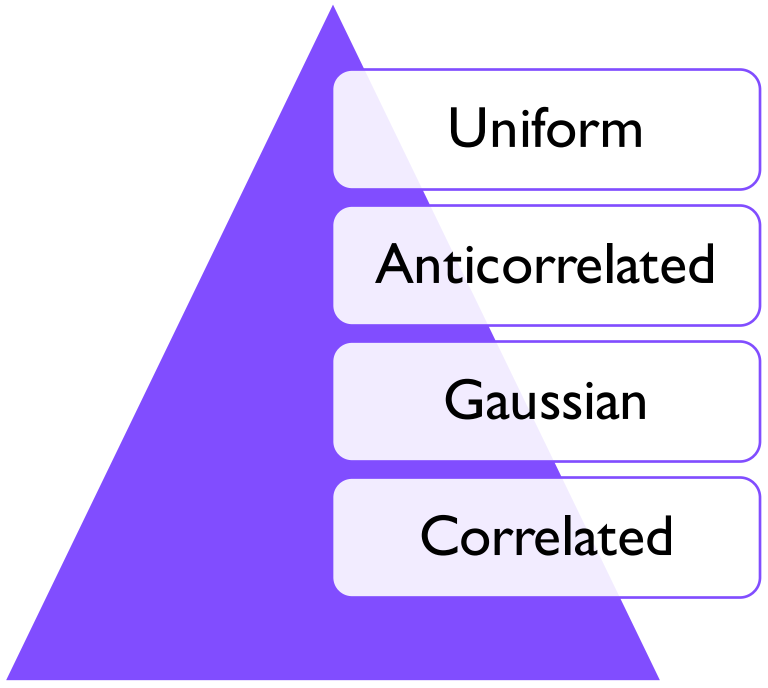
      Uniform distribution is best parallelized, correlated worst due to 'tube-like' structure

Algorithms

- Sort First Skyline (SFS) algorithm
  - Uses a monotone function to sort points (logarithmic sum of coordinates)
  - Points with a lower function score have a higher probability of belonging to the skyline
  - Points are sorted based on their score and then the skyline is calculated
  - Fastest approach, we implement this in all our algorithms
- Top-k dominant points algorithm
  - Calculates the number of points each point dominates (domination score)
  - Filters top-k points
  - O(nlogn) complexity
- Top-k dominant skyline points algorithm
  - Combination of the two
  1. Calculates the domination score
  2. Calculates the skyline
  3. Filters top-k points from the skyline

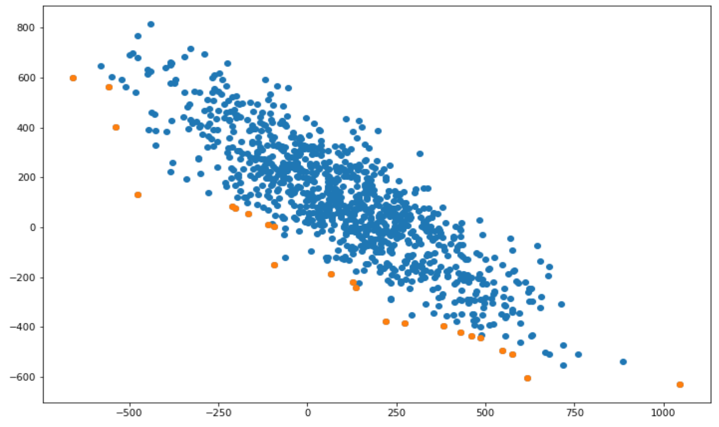
Results

- Angular partitioning into equal slices is suboptimal because not all points are equally distributed across the angles.
- Parallelization effect -> dense partitions take the most time and occupy limited resources.
- All distribution types gather their skylines across the axes as optimally as possible.
- The correlated distribution seems to be gathering the skyline across its length.
- The top k most dominant points are always found near the beginning of the axes.
- A similar behaviour is displayed for the top-k skyline points. The behaviour is most apparent as the cardinality of the dataset increases and as values of k decrease.

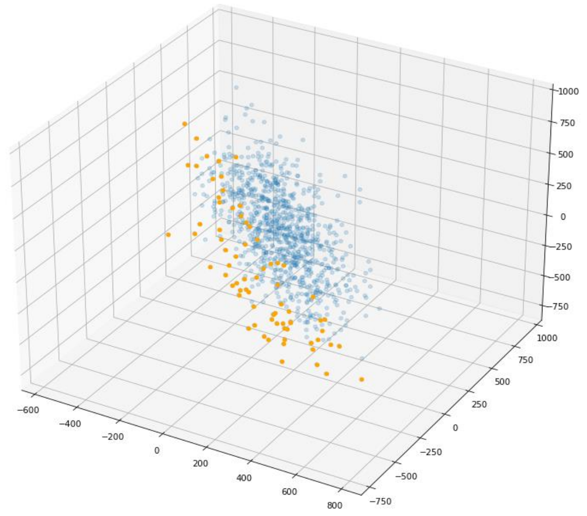


Parallelization ability

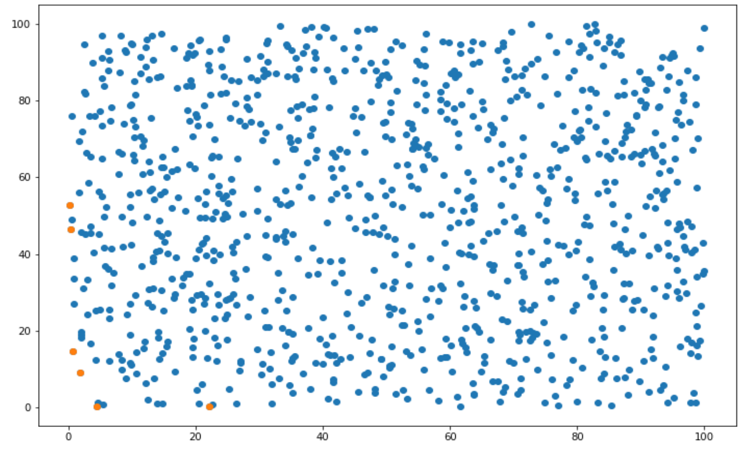
Results



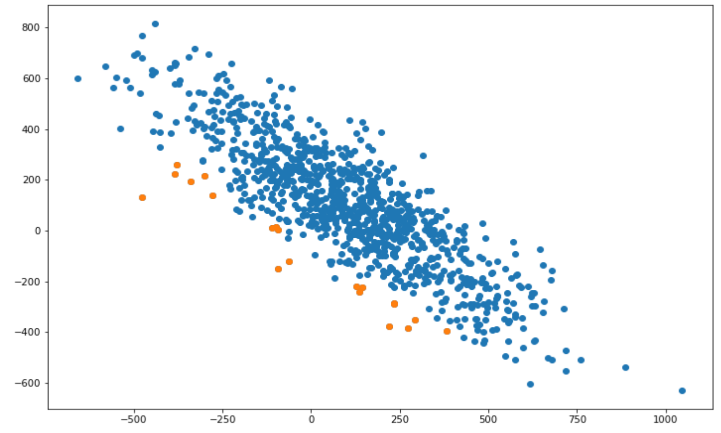
Anticorrelated Skyline



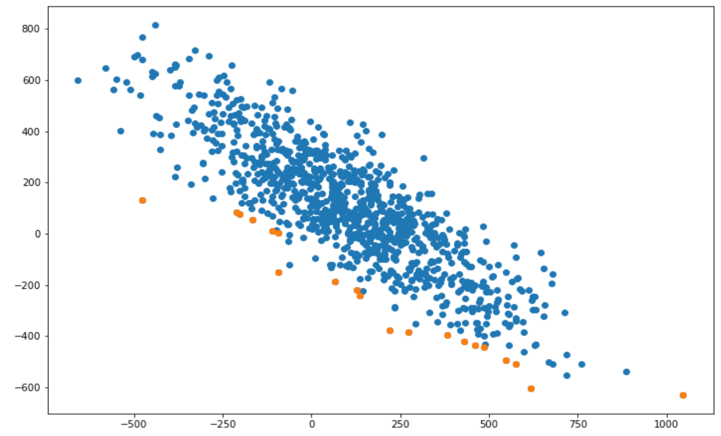
Anticorrelated Skyline 3D



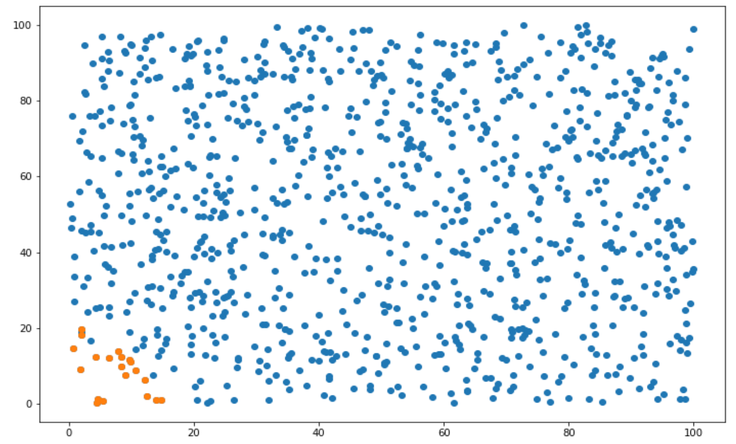
Uniform Skyline



Anticorrelated Top-K

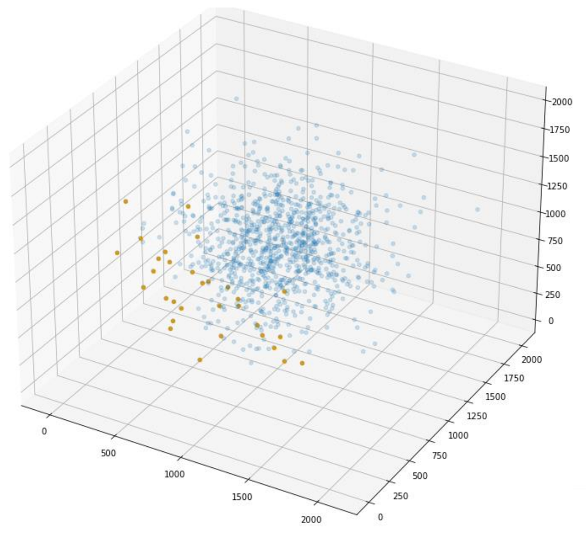


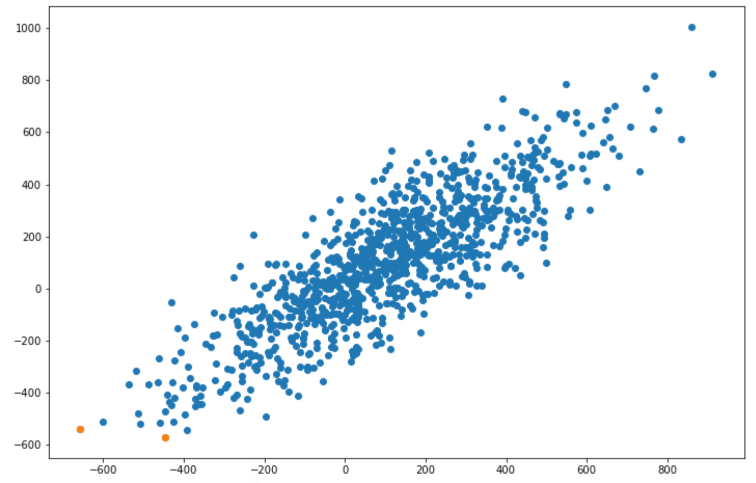
Anticorrelated Skyline –Top-K

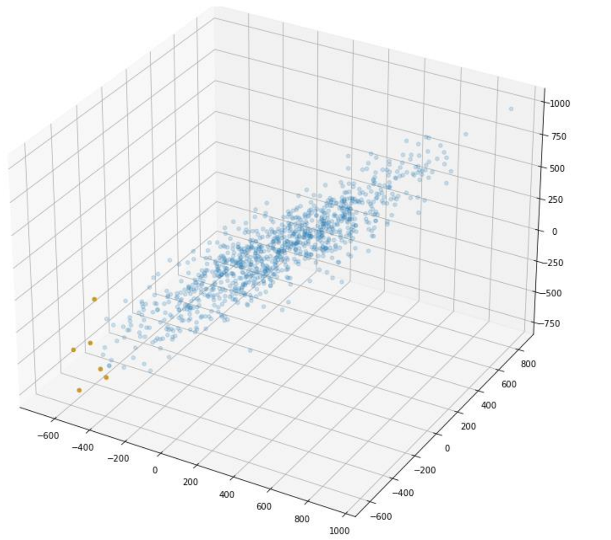


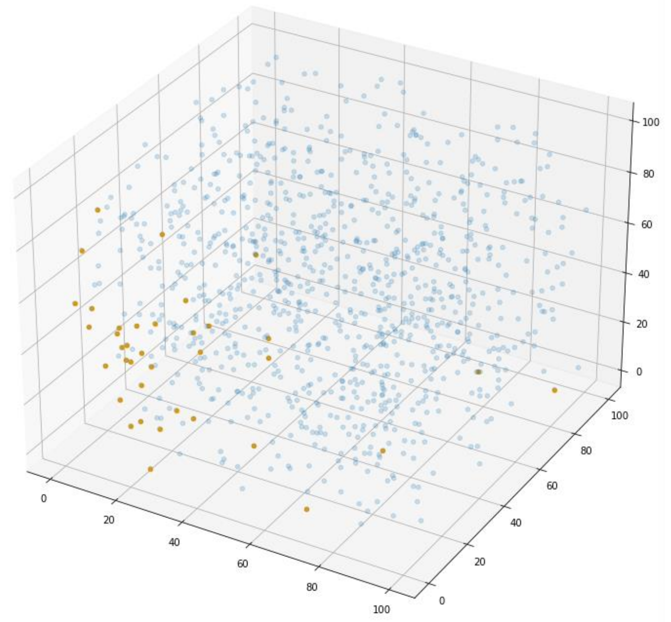
Uniform Top-K

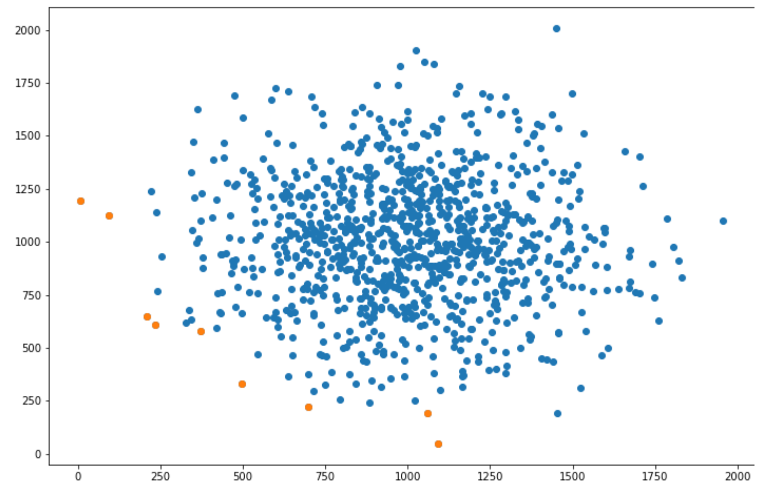
More Skyline Results



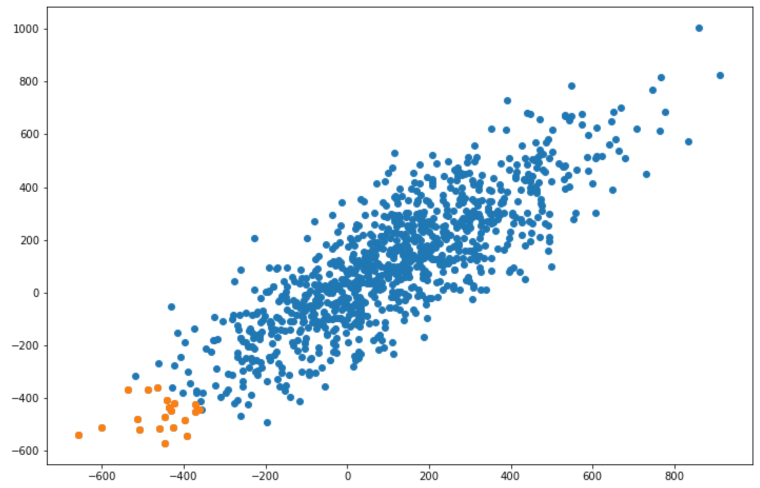


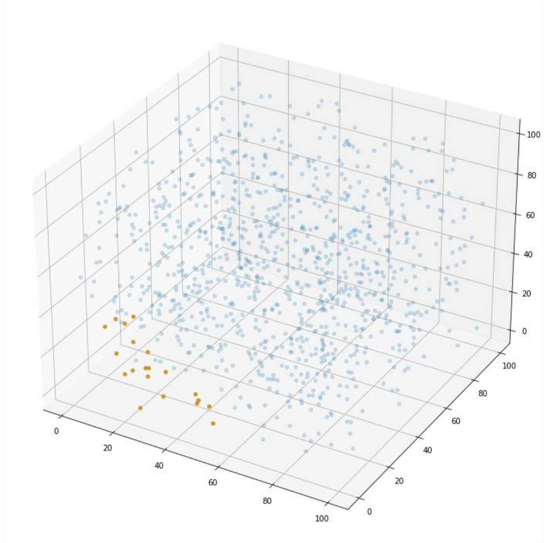


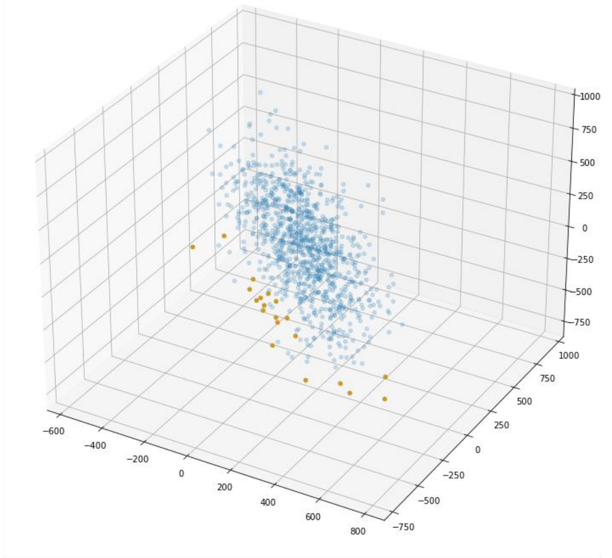


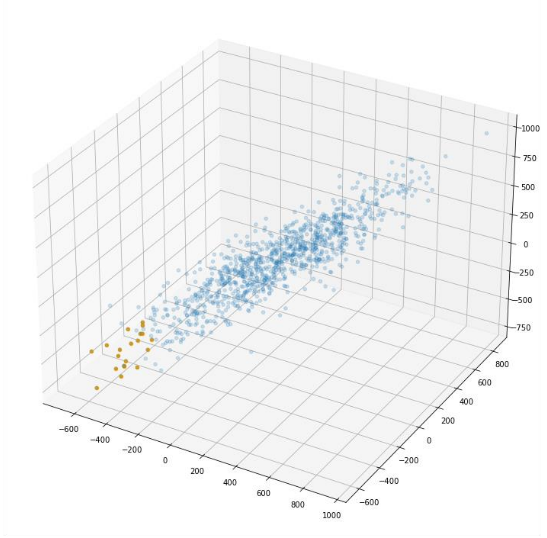


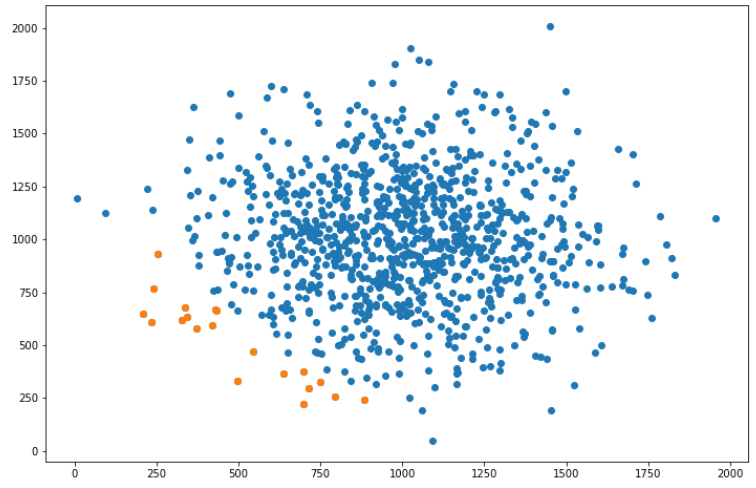
More Top-K Results

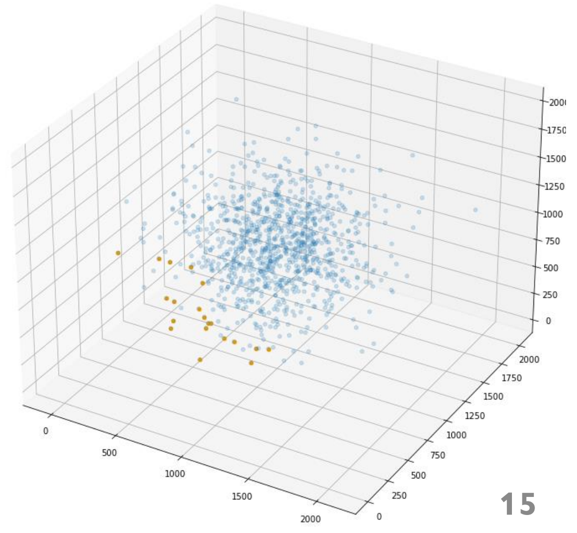




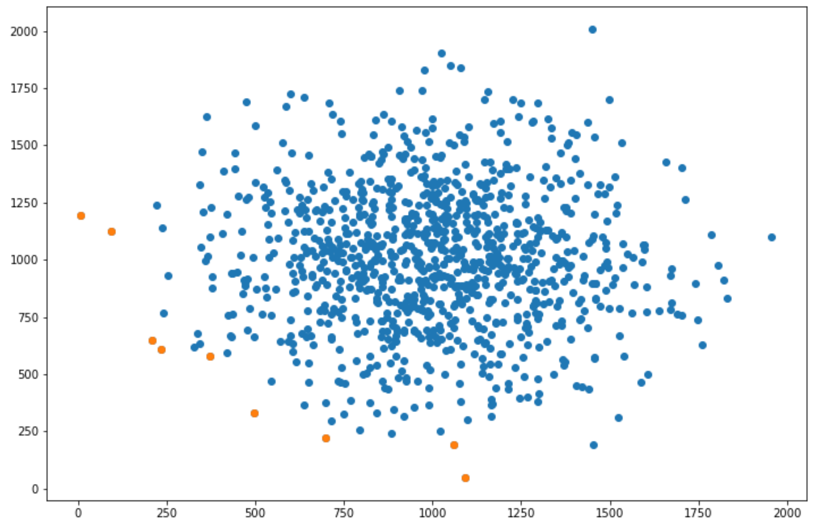
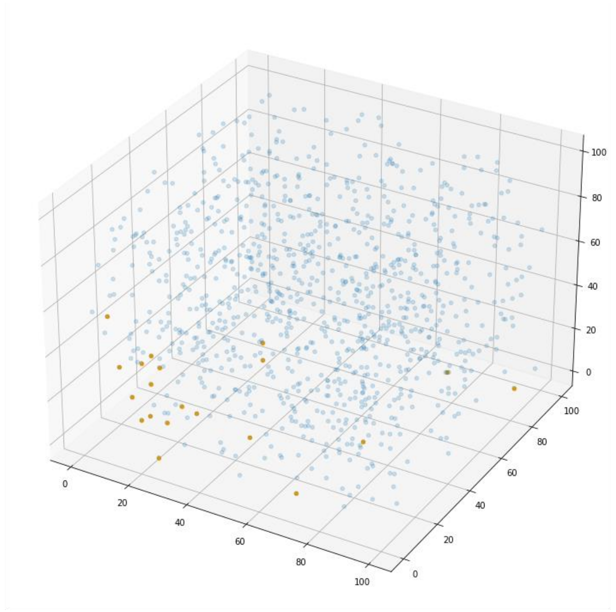
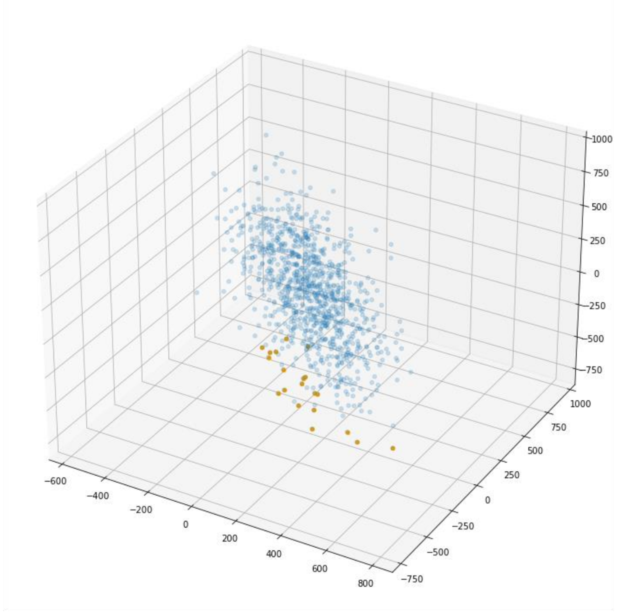
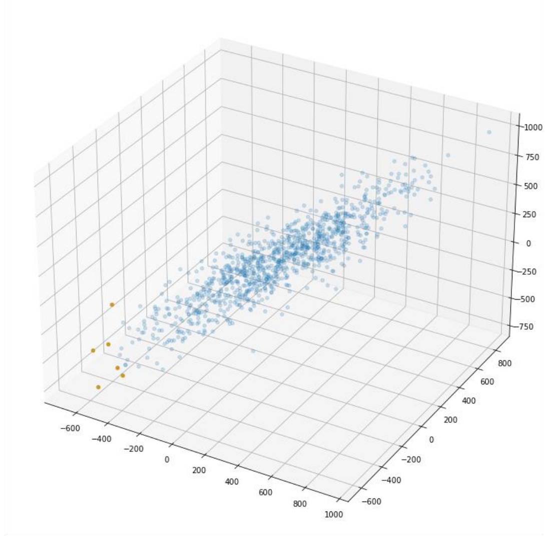
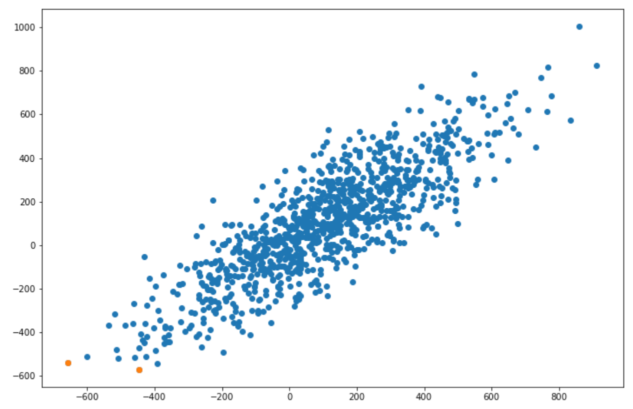
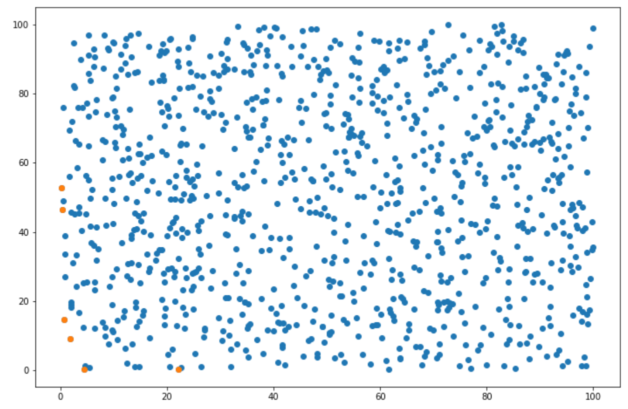






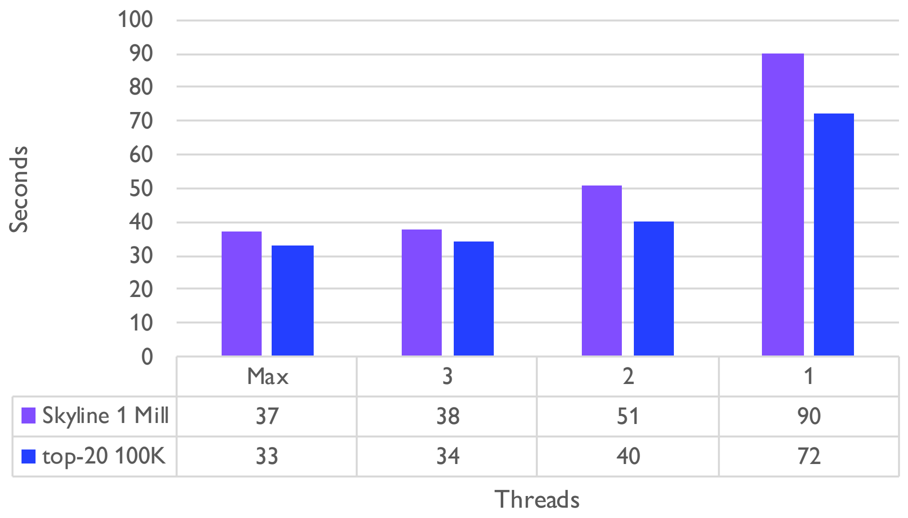


More Top-K Skyline Results

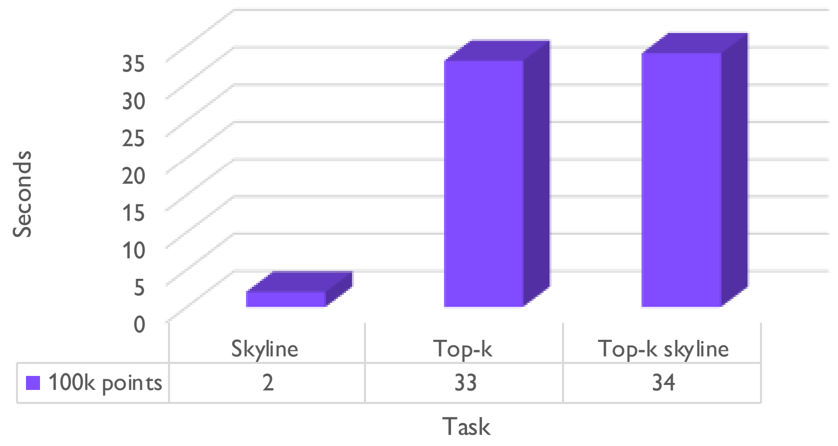
Time comparison

3 dimensions, 9 partitions, uniform



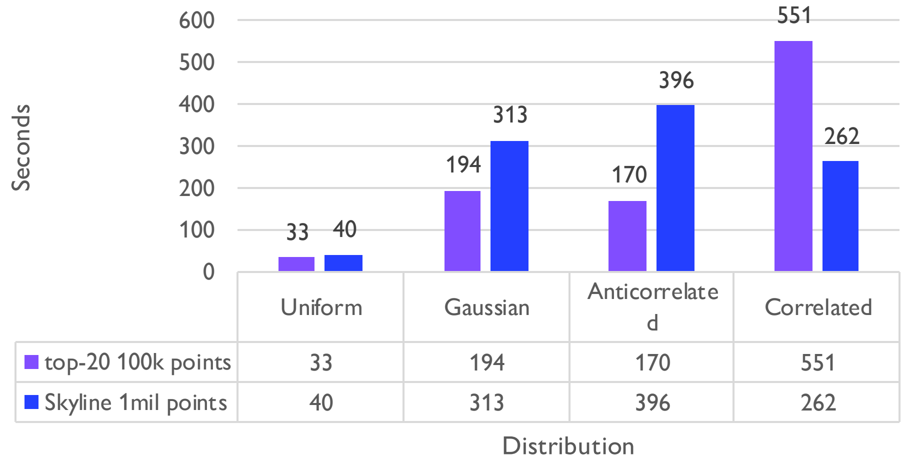
	Max	3	2	1
🟪 Skyline 1 Mill	37	38	51	90
🟦 top-20 100K	33	34	40	72

3 dimensions, 9 partitions, uniform, max threads



	Skyline	Top-k	Top-k skyline
🟪 100k points	2	33	34

3 dimensions, 9 partitions, max threads



	Uniform	Gaussian	Anticorrelated	Correlated
🟪 top-20 100k points	33	194	170	551
🟦 Skyline 1mil points	40	313	396	262

Conclusion

- In practice, skyline and top-k queries are usually issued in a large number of subspaces, each of which includes a small subset of the attributes in the underlying relation.
- Future improvement: Volume calculation
- Spark is being widely used in Big data industry for interactive scaling out batch data processing requirements.
- In addition to that, it is expected to play a key role in the next generation Business Intelligence Applications and in the 4th Industrial revolution.

THANK YOU