Faster similarity queries using RDKit, binary fingerprints, and relational databases

Riccardo Vianello

5th RDKit UGM
Basel 26-28 October 2016

Outline

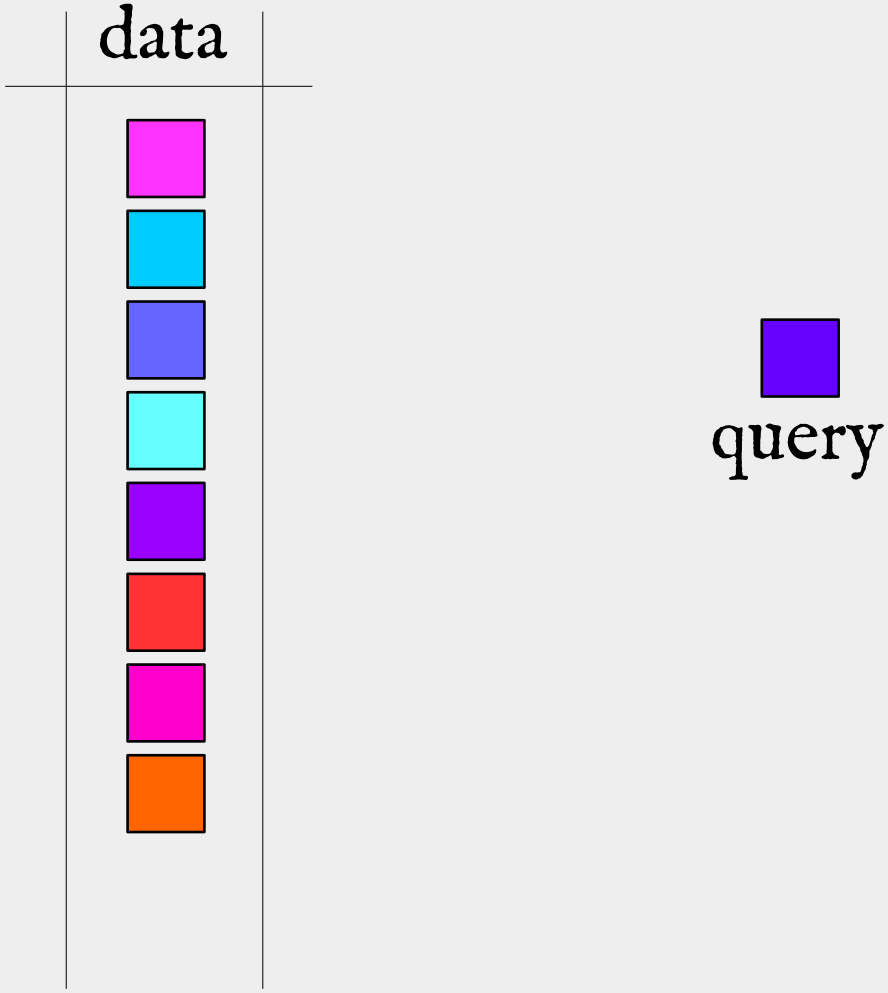
- An illustration of how the sequential and indexed similarity search strategies compare
- A high-level performance comparison between the optimized index implementation and the earlier RDKit release
- Results of a preliminary exercise combining horizontal-scaling and the RDKit cartridge in PostgreSQL

Regarding tests and benchmarks

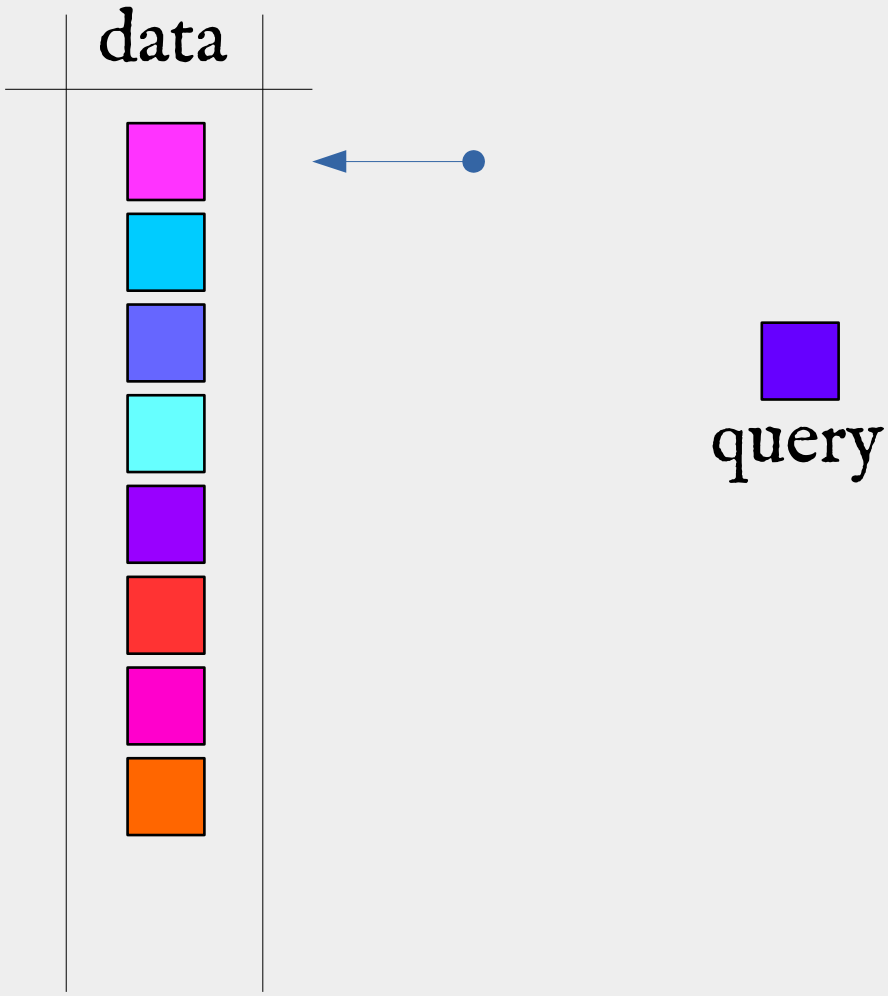
Unless differently specified, all tests have been executed on a i7-4510U @ 2GHz linux laptop w/ 16 GB RAM, using PosgreSQL 9.5 and SQLite 3.13.0

The data sets are random samples extracted from the ZINC drug-like subset.

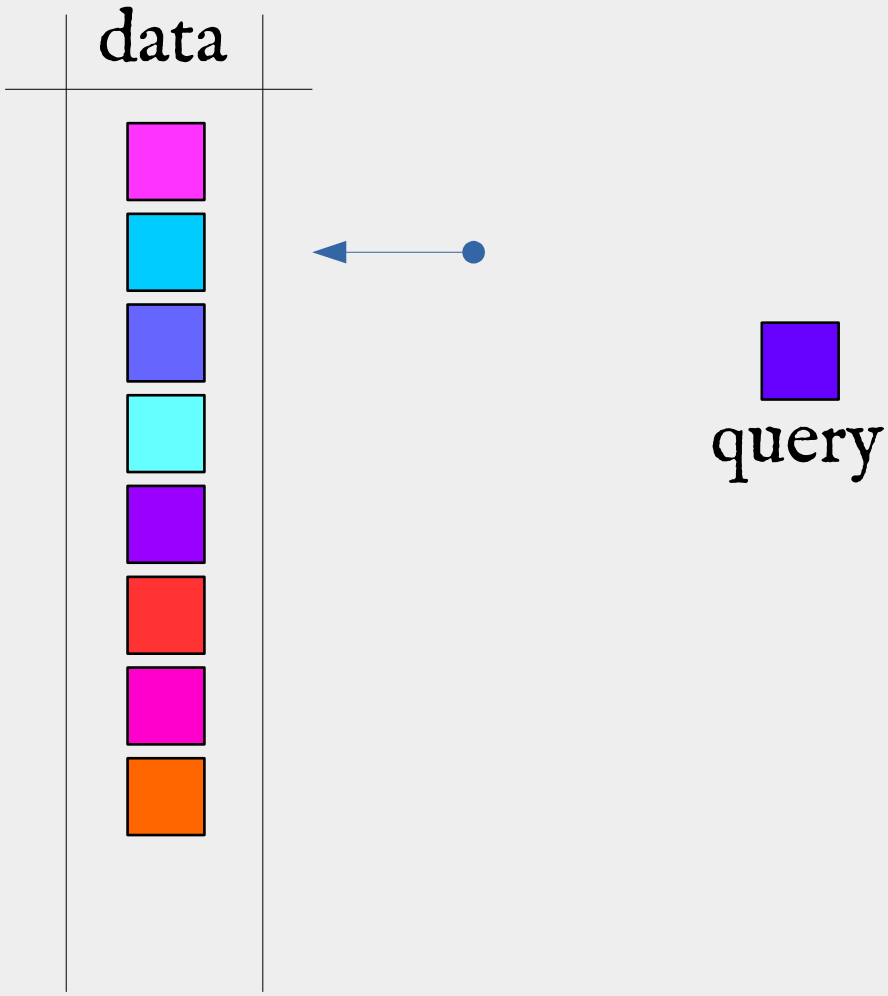
Sequential scan



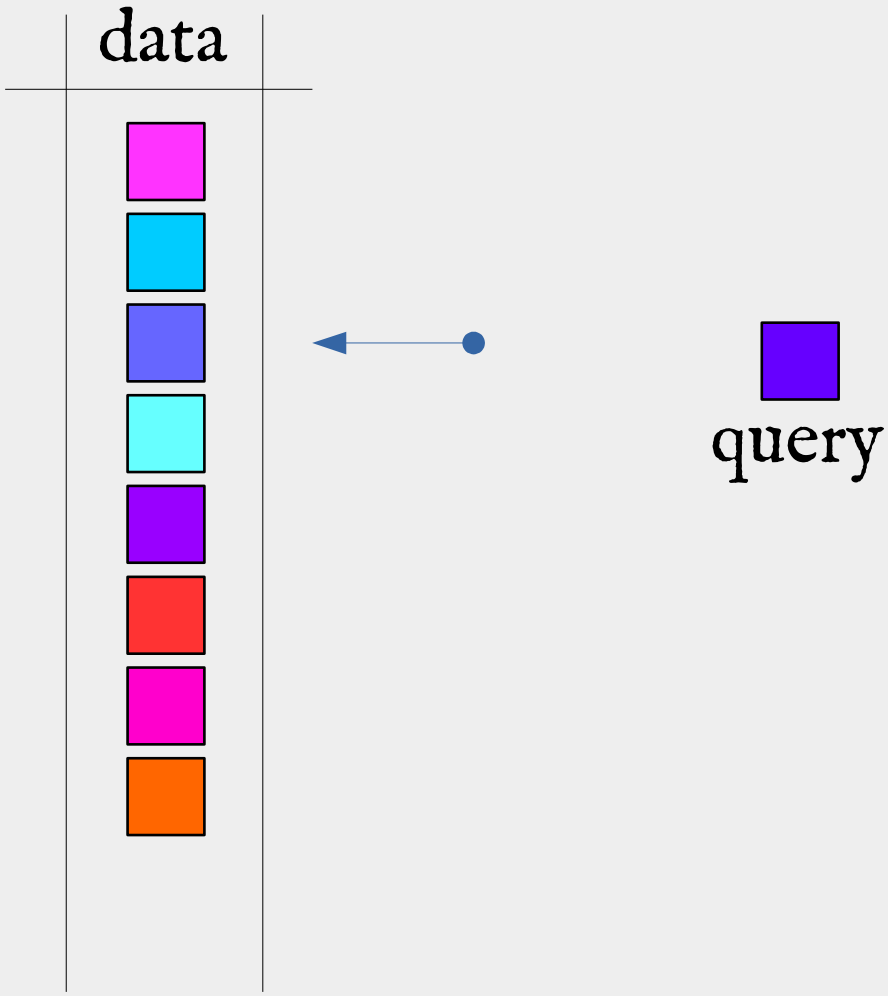
Sequential scan



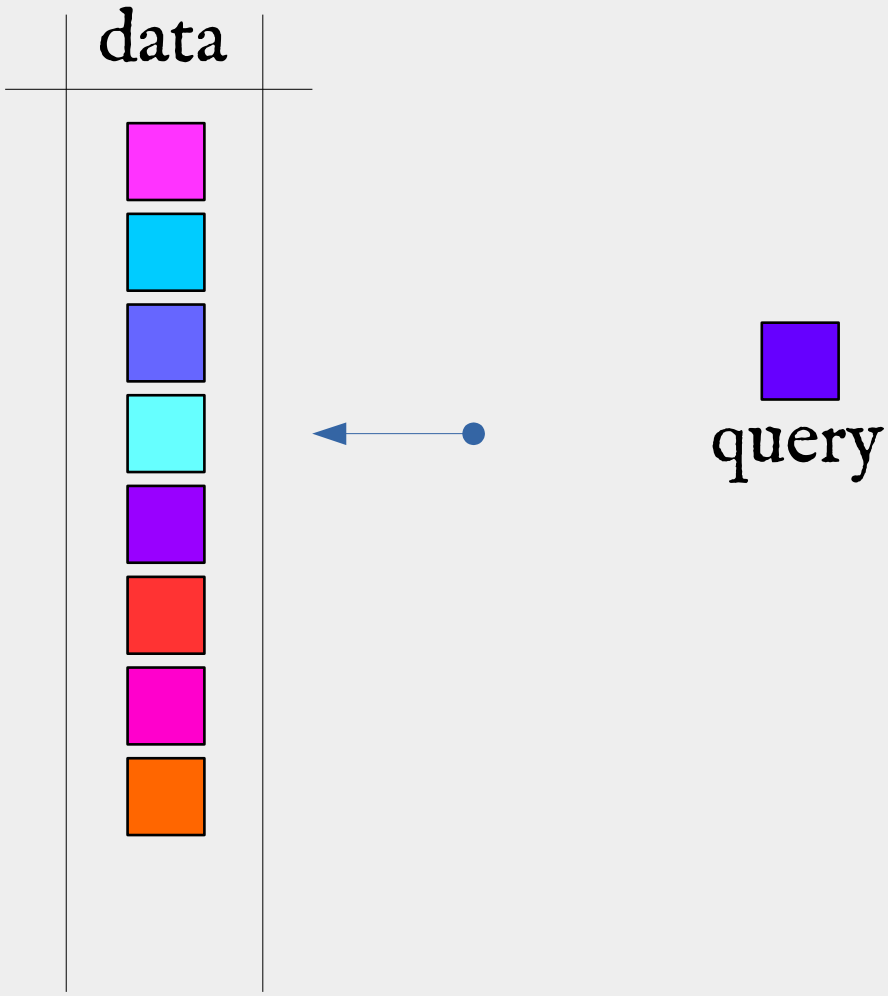
Sequential scan



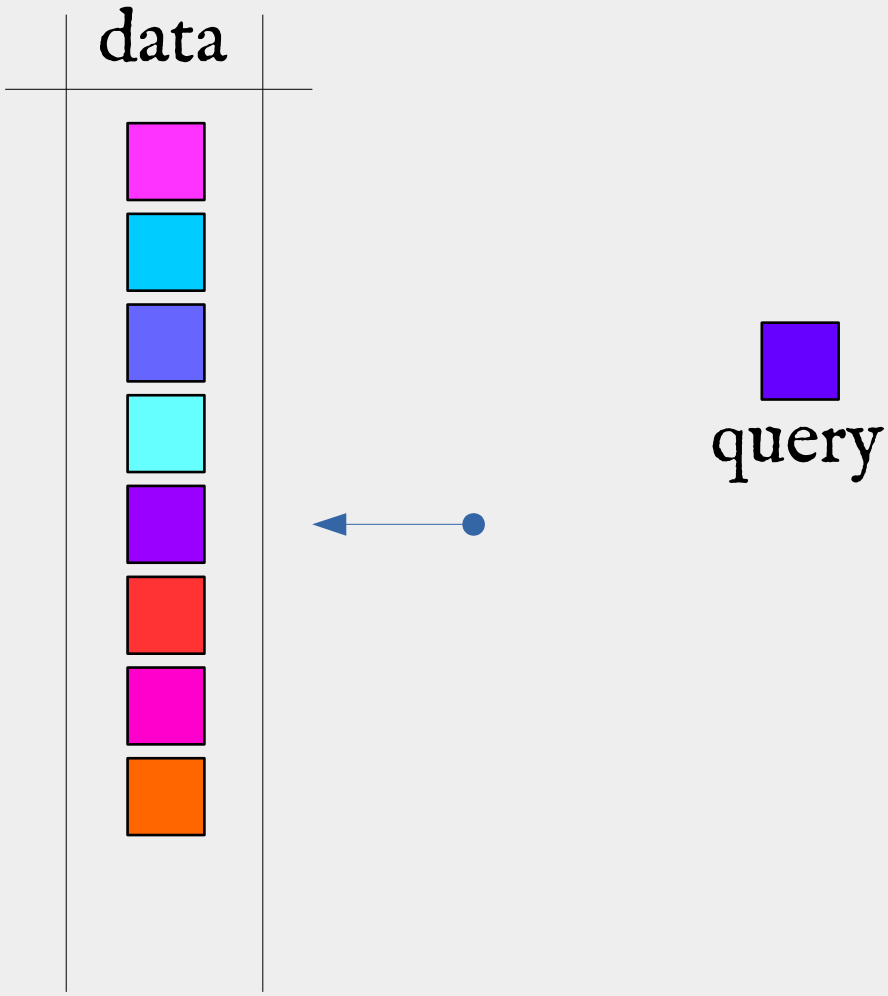
Sequential scan



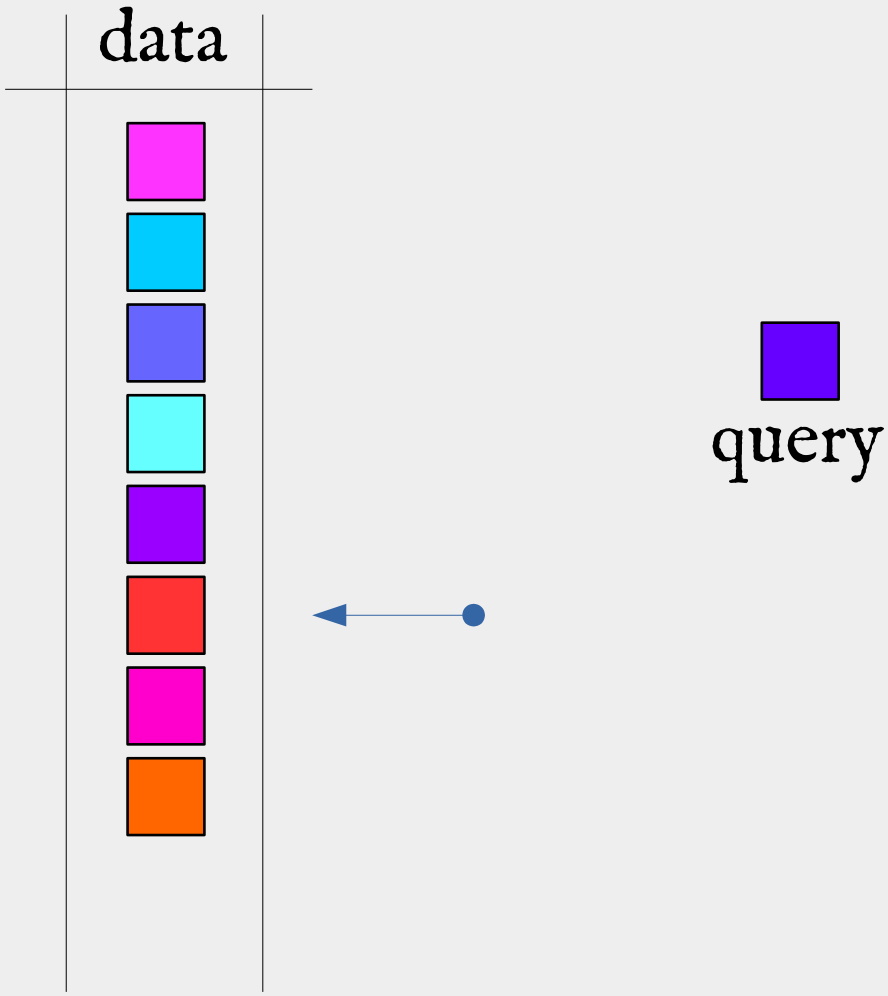
Sequential scan



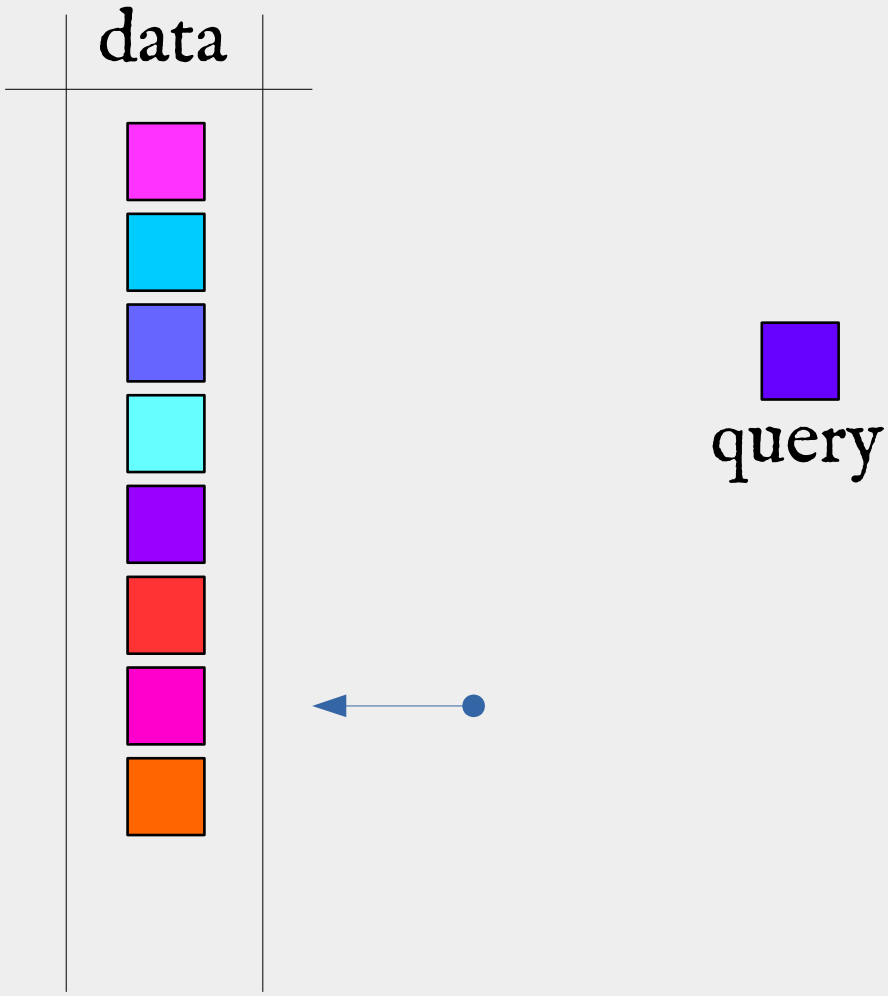
Sequential scan



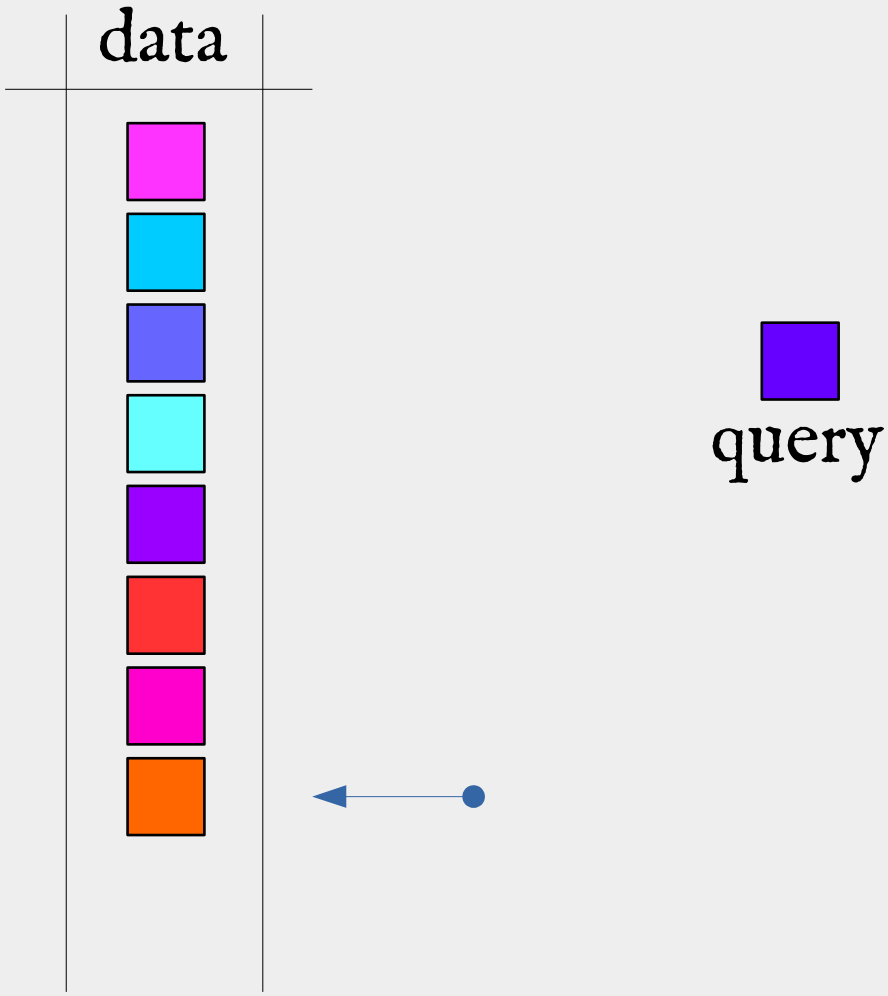
Sequential scan



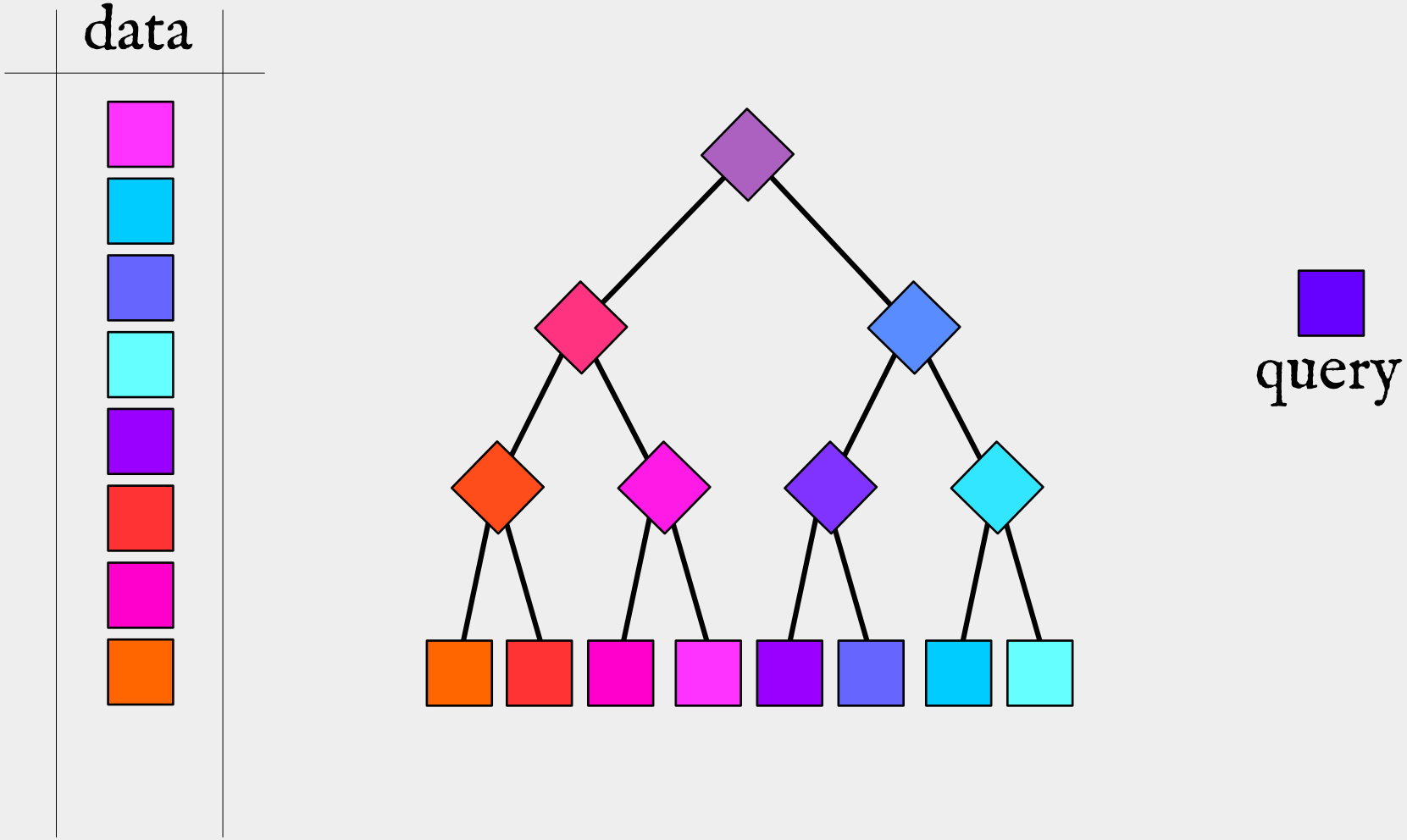
Sequential scan



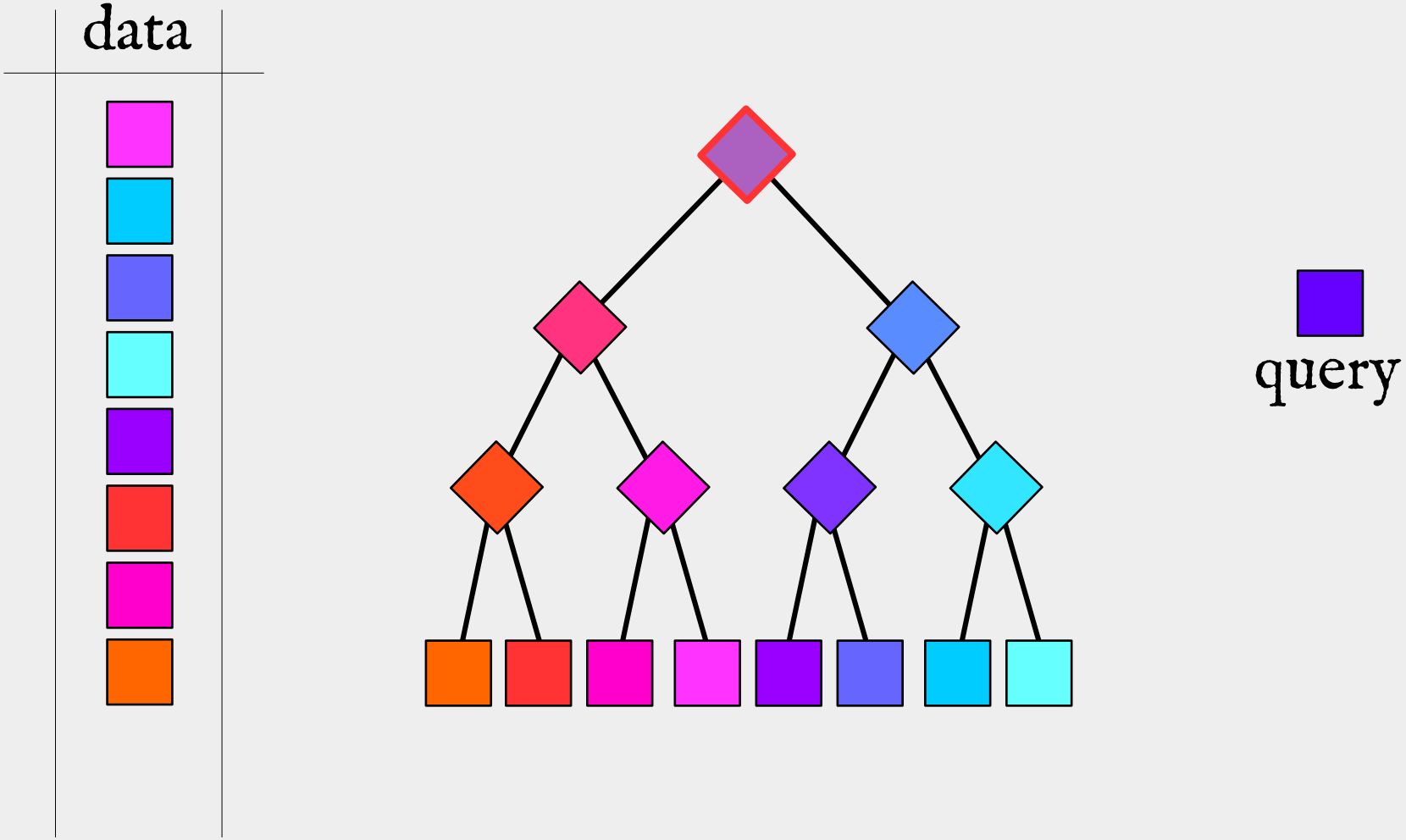
Sequential scan



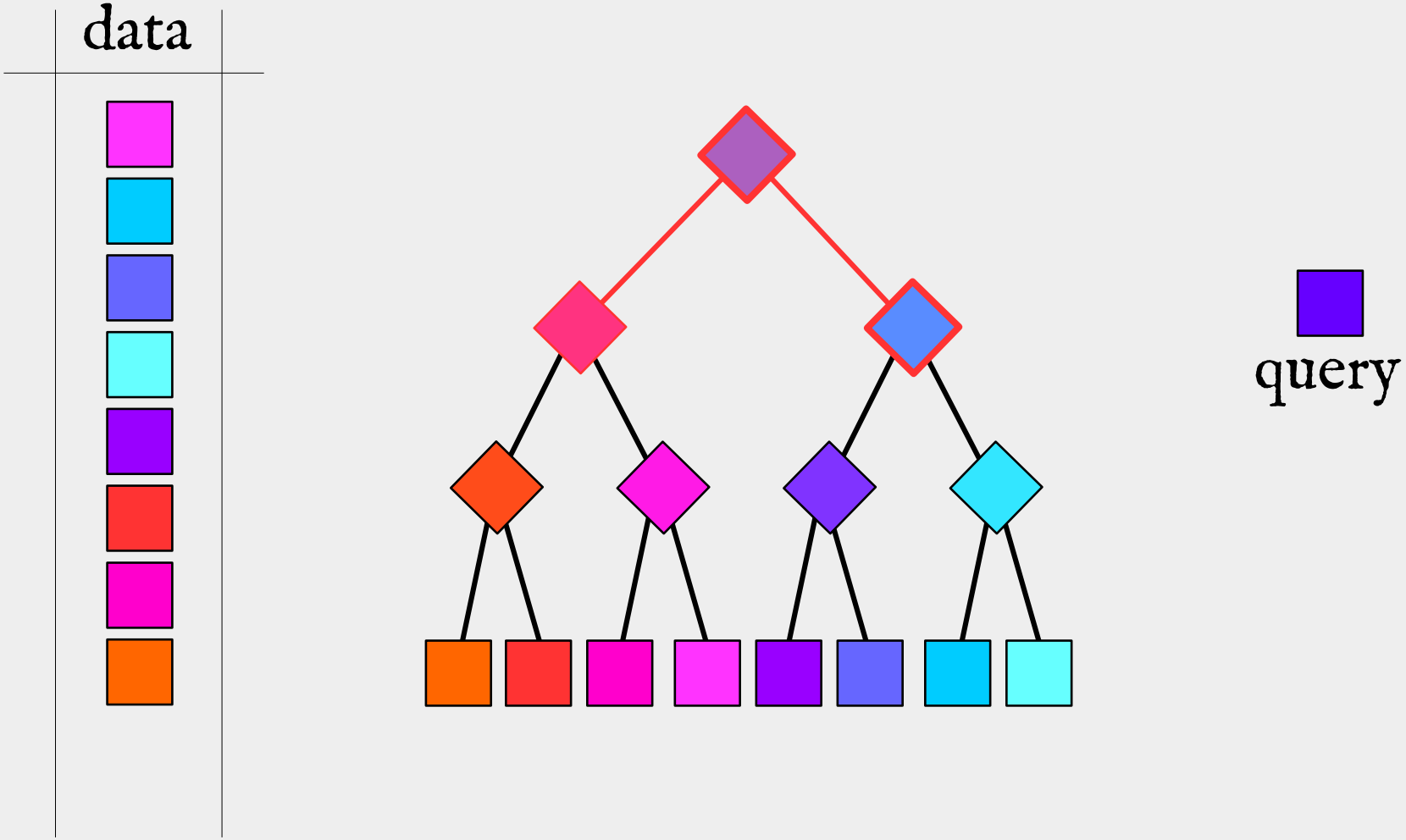
Indexed scan



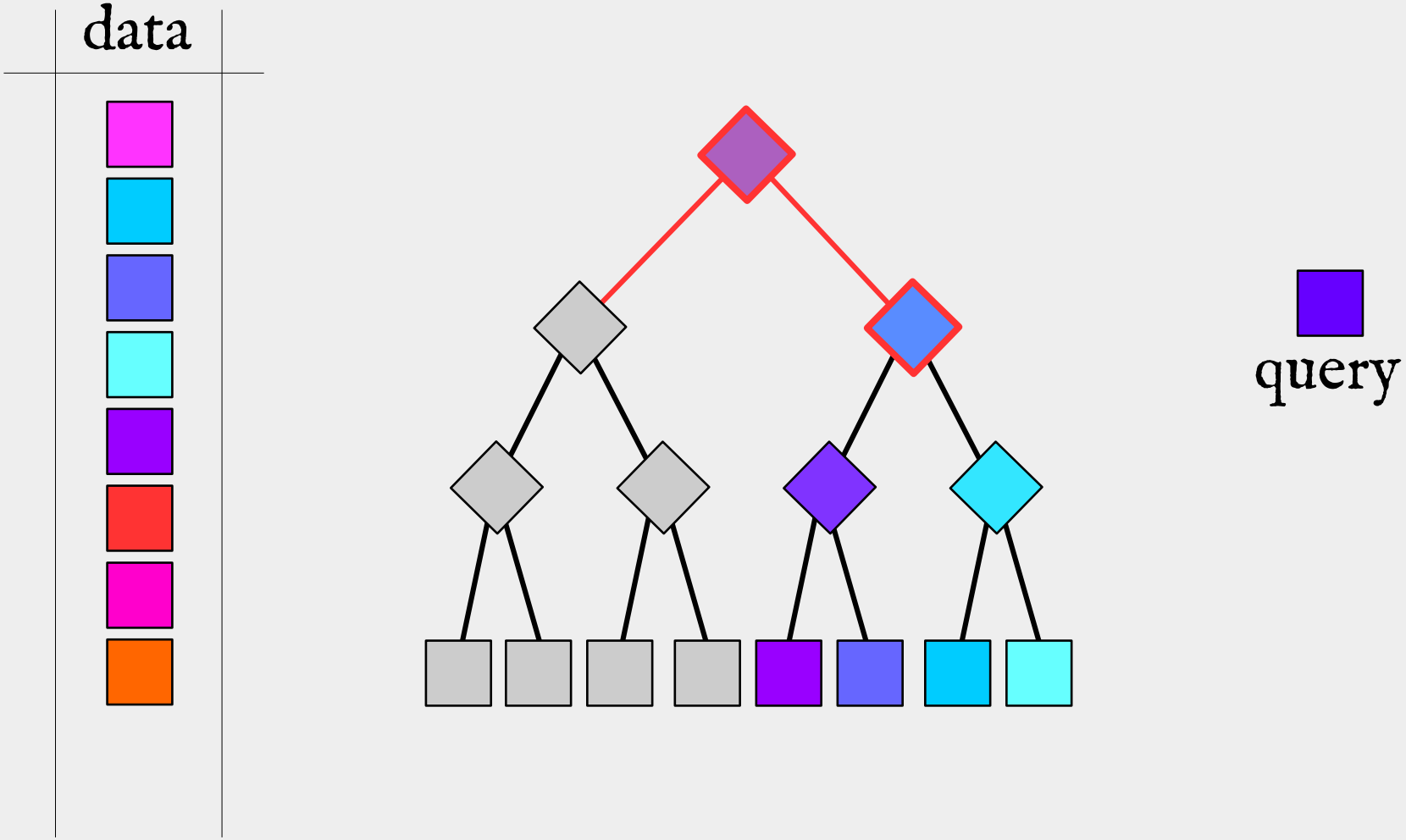
Indexed scan



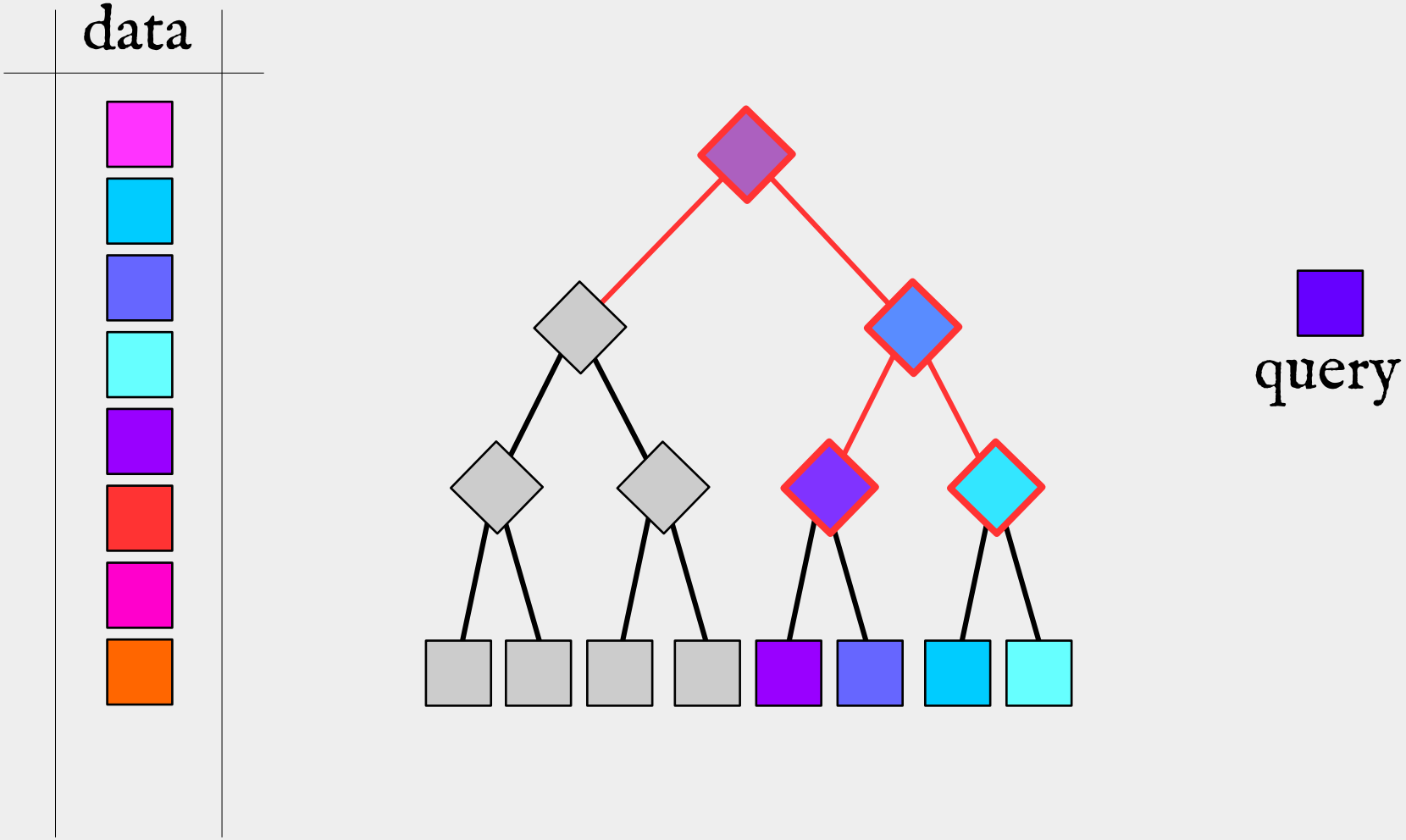
Indexed scan



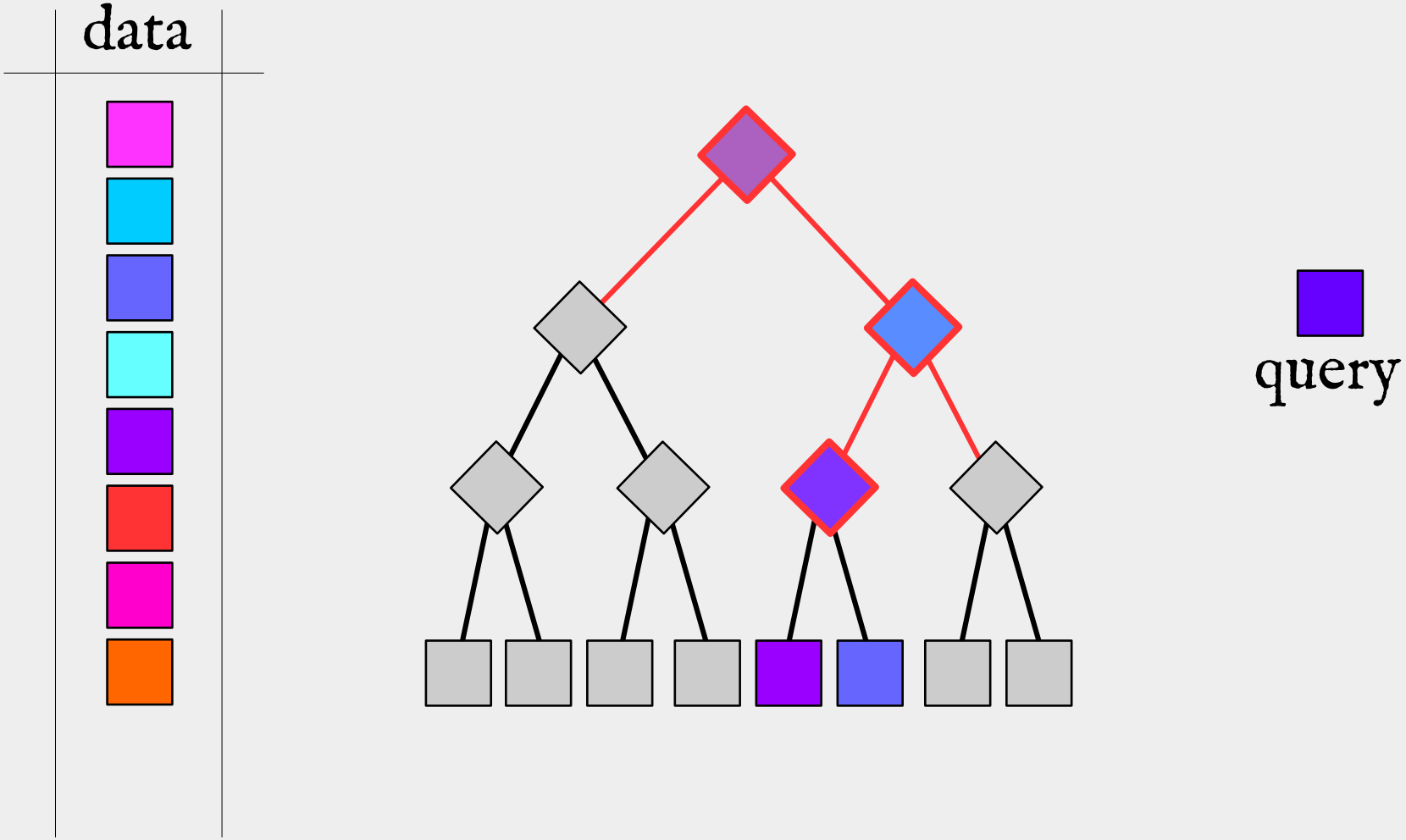
Indexed scan



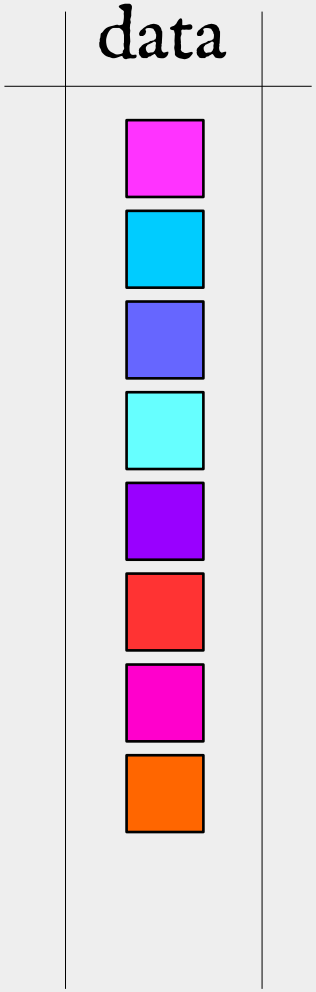
Indexed scan

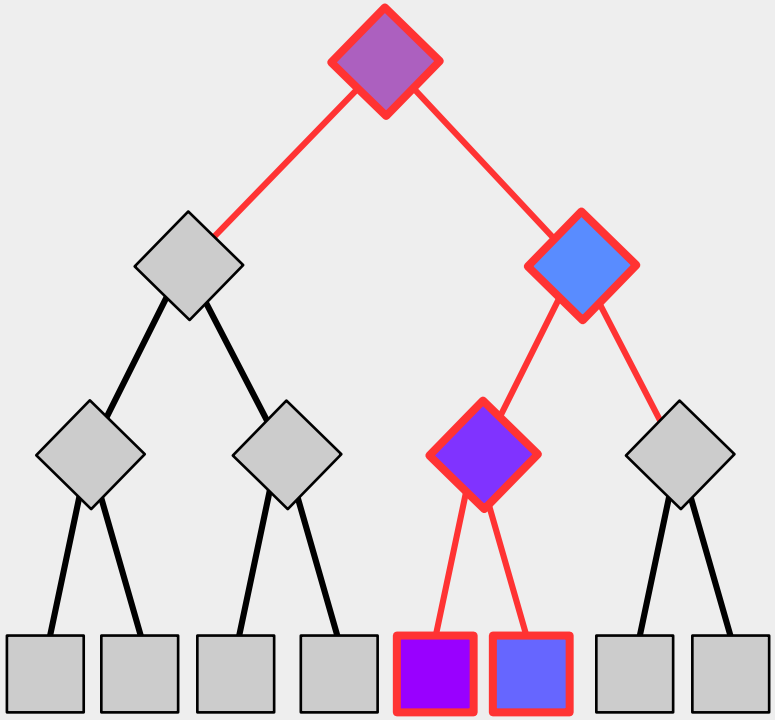


Indexed scan



Indexed scan





Similarity search tree implementation

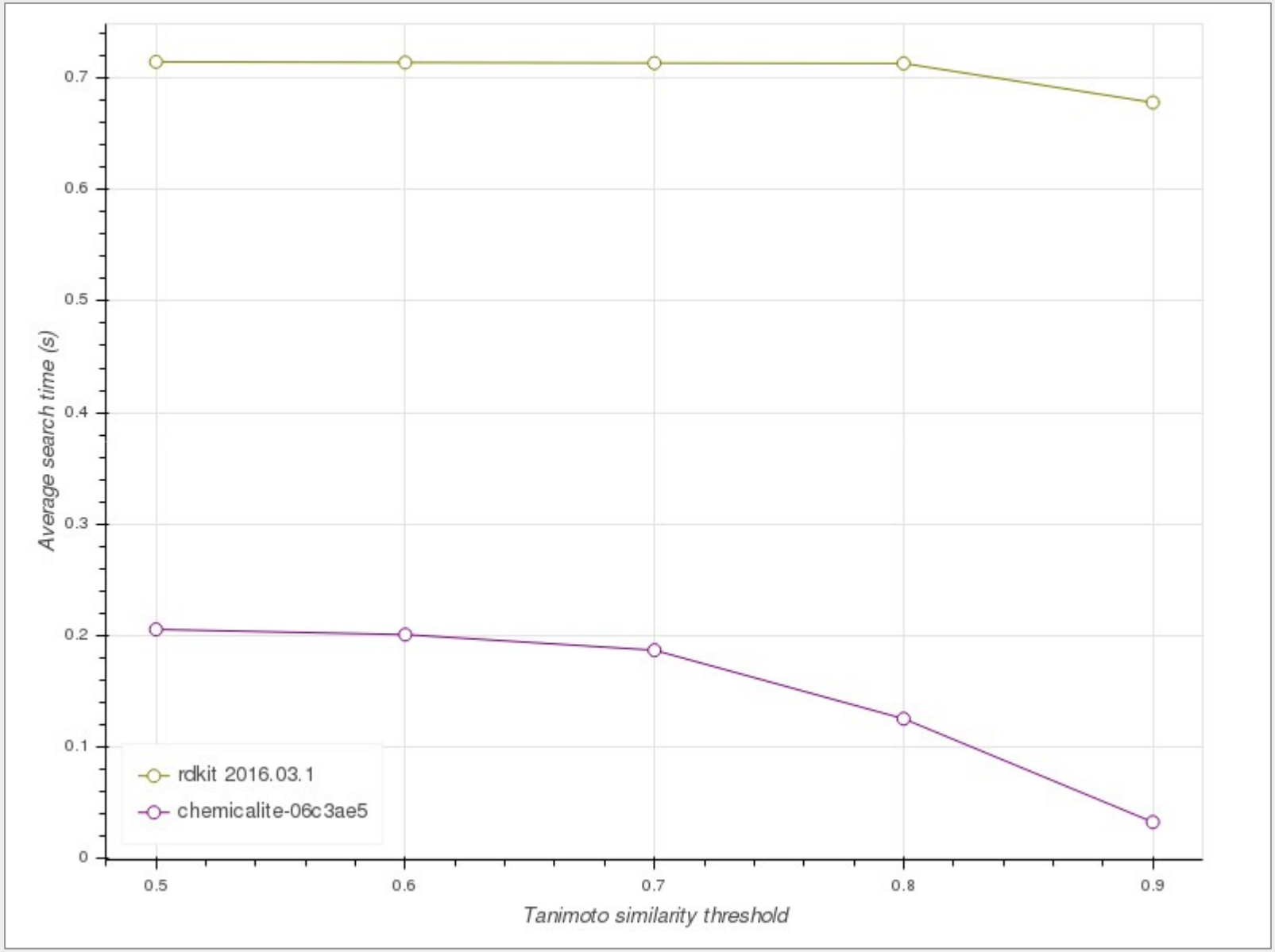
- Distributes the values on leaf nodes so that items that are similar to each other are in the same sub-trees
- Fills the inner nodes with aggregated information that allows computing upper bounds to the similarity within the sub-tree

Useful reviews of bfp similarity search theory and practice

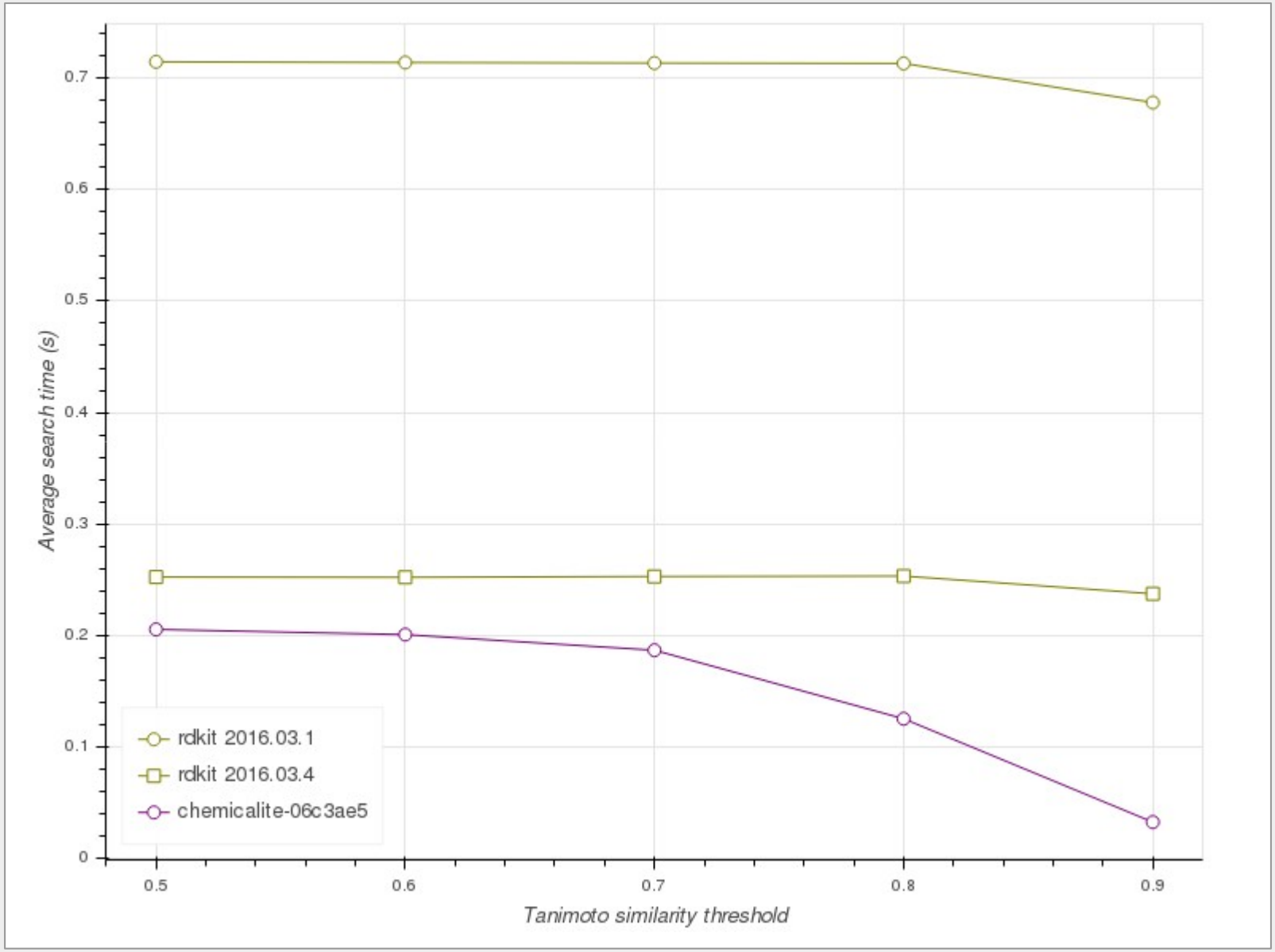
- "Update: Faster population counts", Dalke A., https://goo.gl/S5NfX5
- "Chemical similarity search in MongoDB", Swain M., https://goo.gl/RUUEQU

ChemicaLite

- A cheminformatics extension for the SQLite database
- Similar in concept to the RDKit PostgreSQL cartridge, but not as feature-rich
- https://github.com/rvianello/chemicalite



Average execution time of Tanimoto match count queries for different threshold values (1M cpds, morgan bfp, rad. 2, 512 bits)

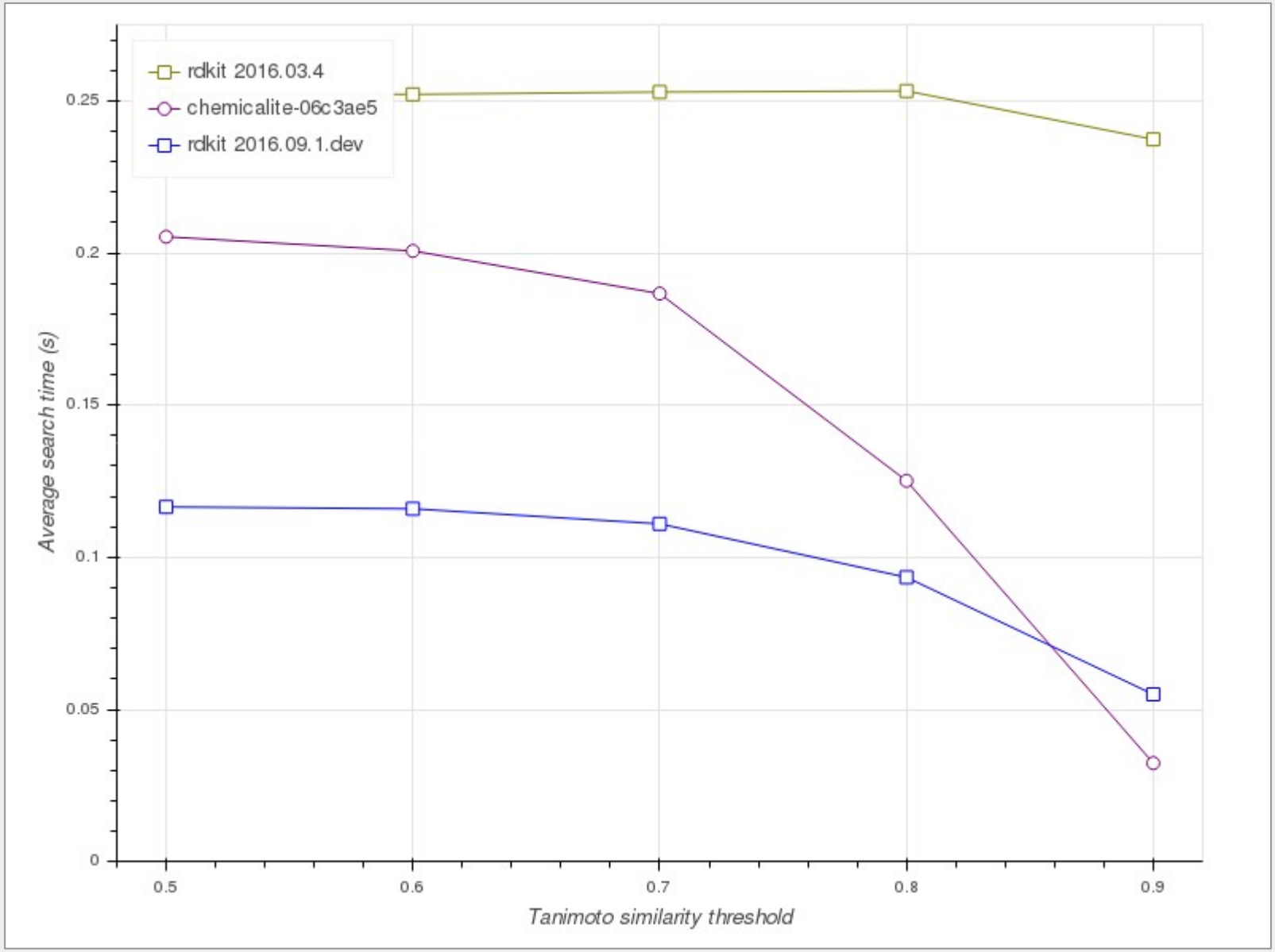


Average execution time of Tanimoto match count queries for different threshold values (1M cpds, morgan bfp, rad. 2, 512 bits)

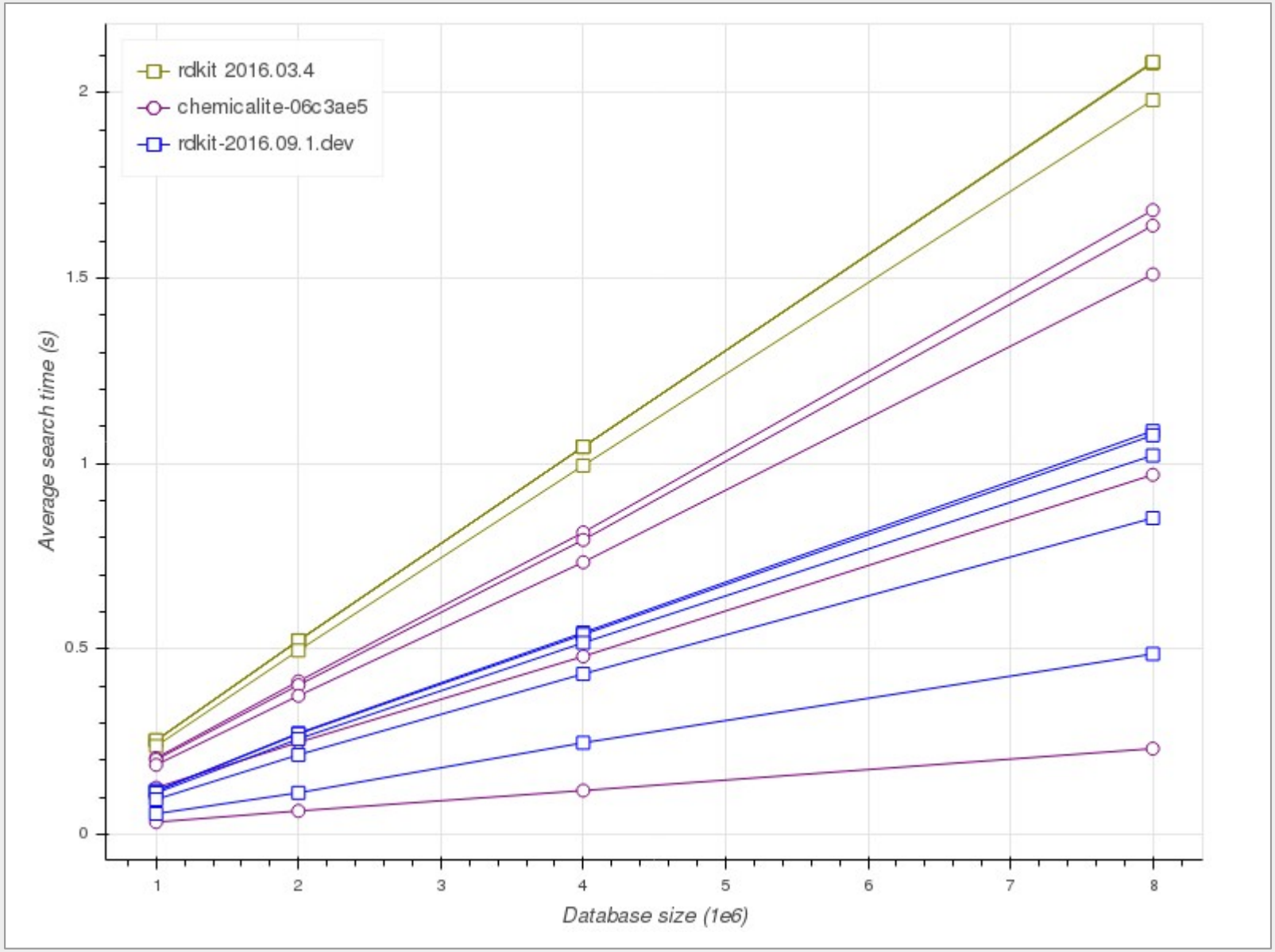
What's new in the RDKit PostgreSQL cartridge

Heavily refactored bfp similarity index implementation

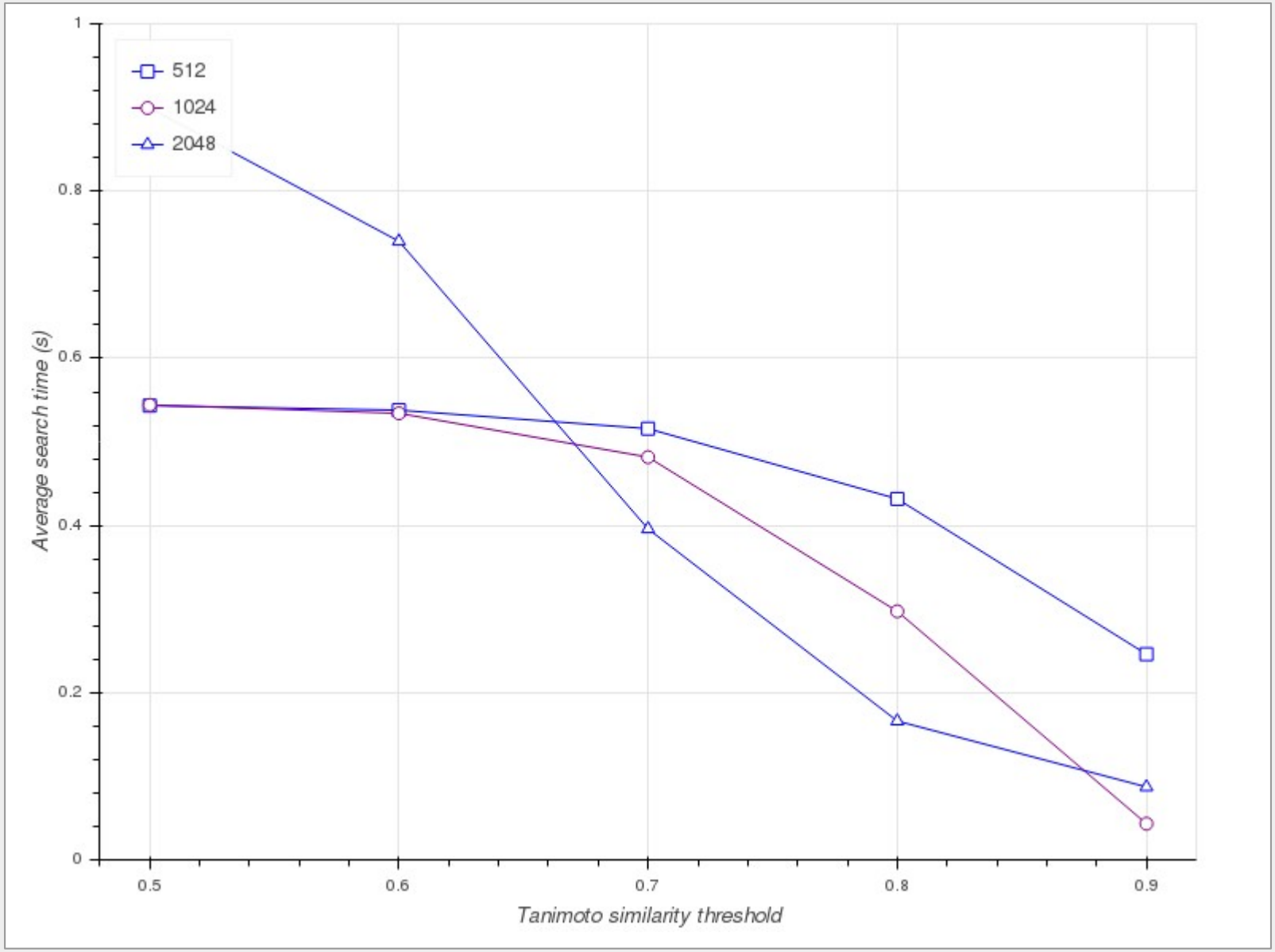
- Consistent use of optimized bfp operations
- The bfp weight/popcount is stored in the index to avoid recomputation
- Improved pruning strategy on the inner tree nodes
- Index-only scan (req. PostgreSQL 9.5+)



Average execution time of Tanimoto match count queries for different threshold values (1M cpds, morgan bfp, rad. 2, 512 bits)



Dependency of average search time on database size
(million of cmpds, morgan bfp, rad. 2, 512 bits)

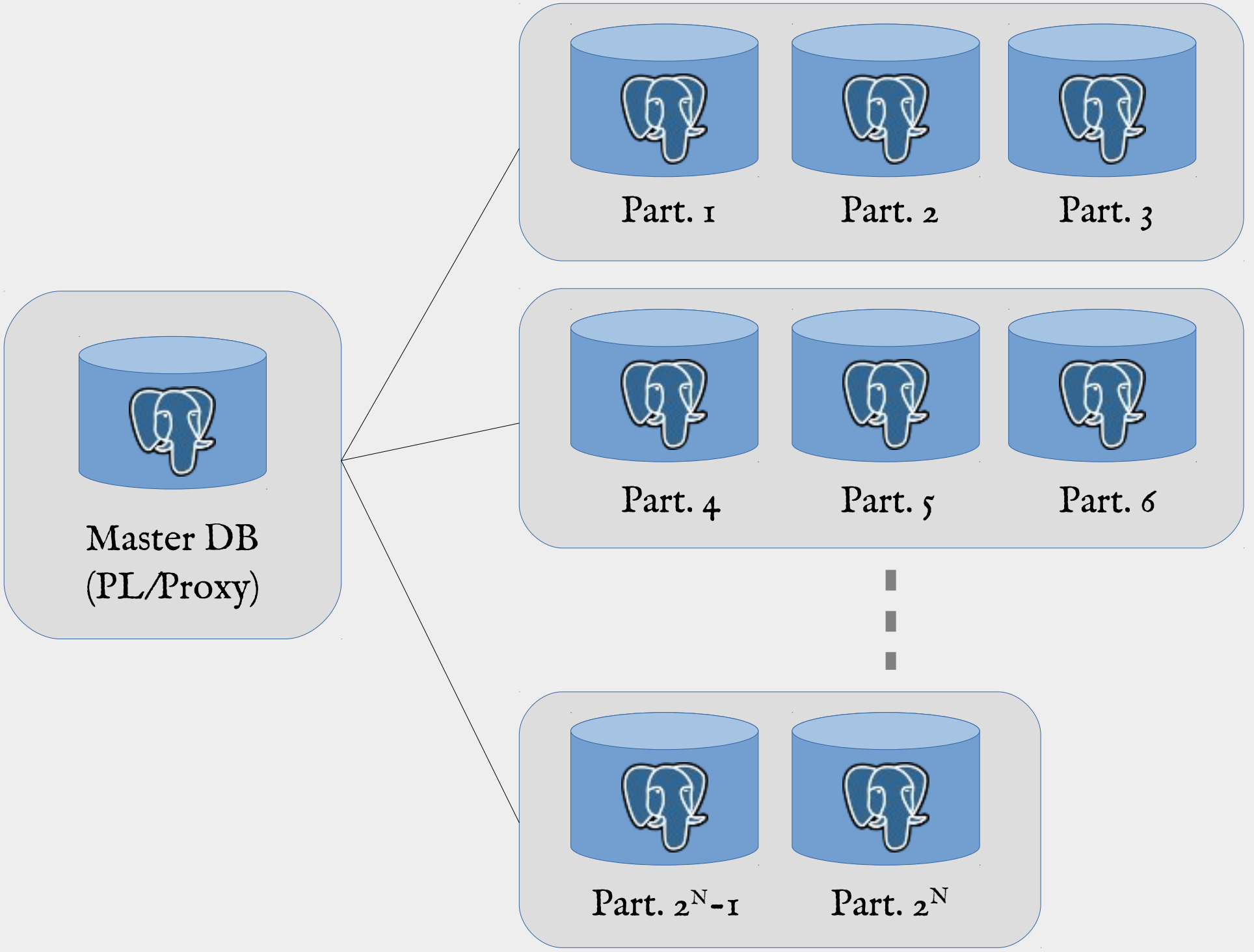


Effect of bfp size on time performances
(4M cmpds, morgan bfp, rad. 2)

Some large-scale solutions for PostgreSQL

- PL/Proxy https://plproxy.github.io

- Citus Data https://www.citusdata.com
- Postgres-XL http://www.postgres-xl.org


Part. 1
Part. 2
Part. 3
Part. 4
Part. 5
Part. 6
Master DB
(PL/Proxy)
Part. 2^N-1
Part. 2^N

On each partition:

```
CREATE TABLE compounds( id serial PRIMARY KEY,
                        smiles text,
                        molecule mol,
                        mbfp bfp
                        );
CREATE INDEX bfpidx ON compounds USING GIST(mbfp);
```

On each partition:

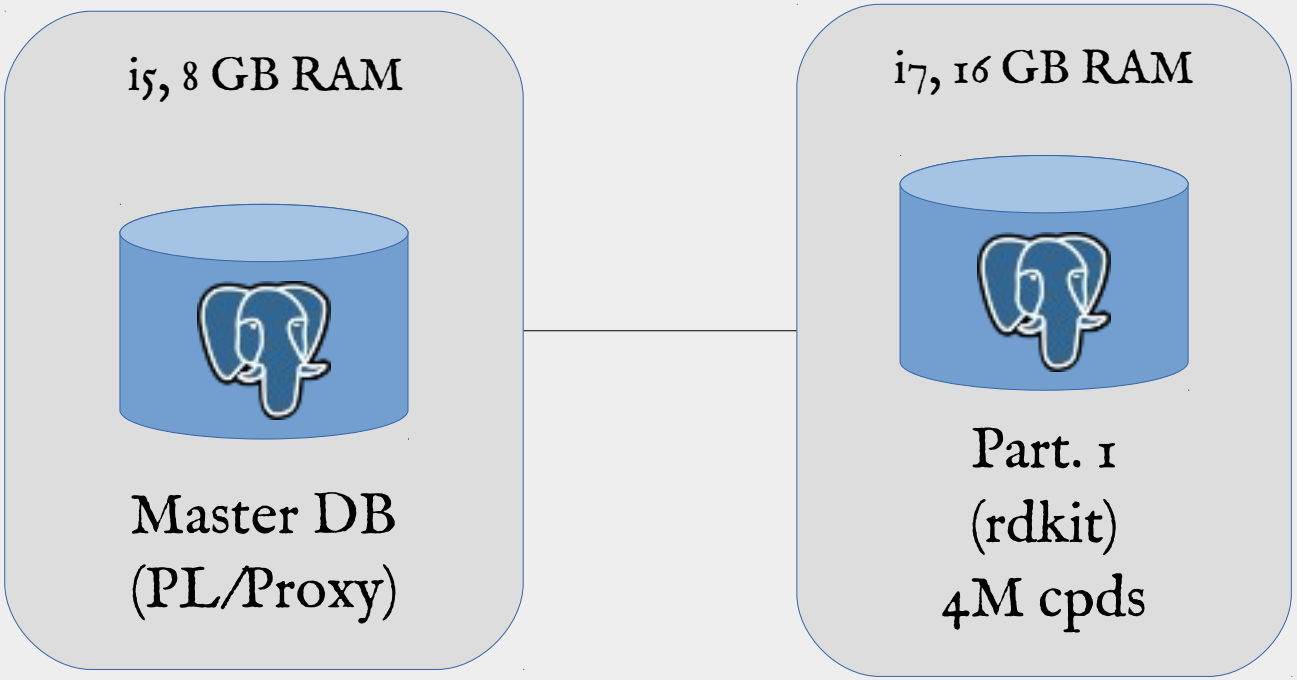
```
CREATE FUNCTION tanimoto_match_count(s text, t numeric)
RETURNS bigint AS $$
    SELECT set_config('rdkit.tanimoto_threshold', t::text, true);
    SELECT count(*) FROM compounds WHERE
        mbfp % morganbv_fp(s::mol);
$$ language sql;
```

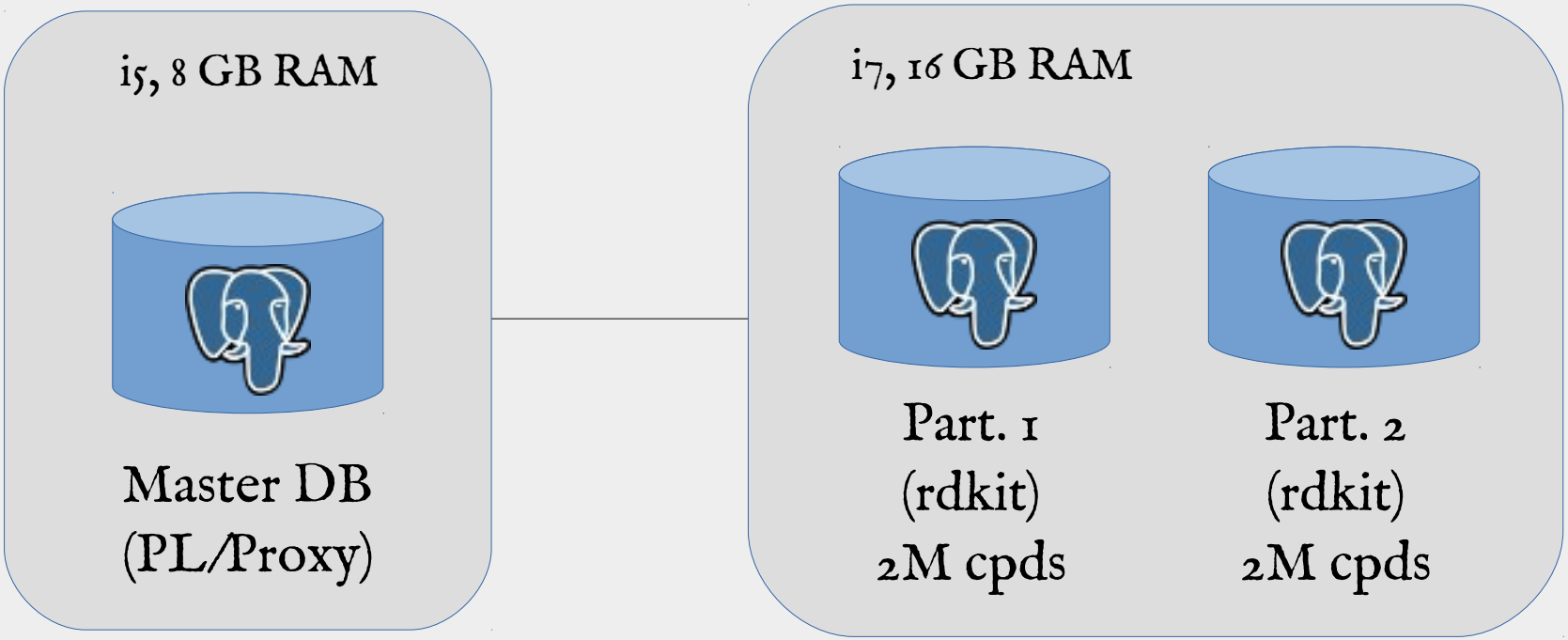
On the master node:

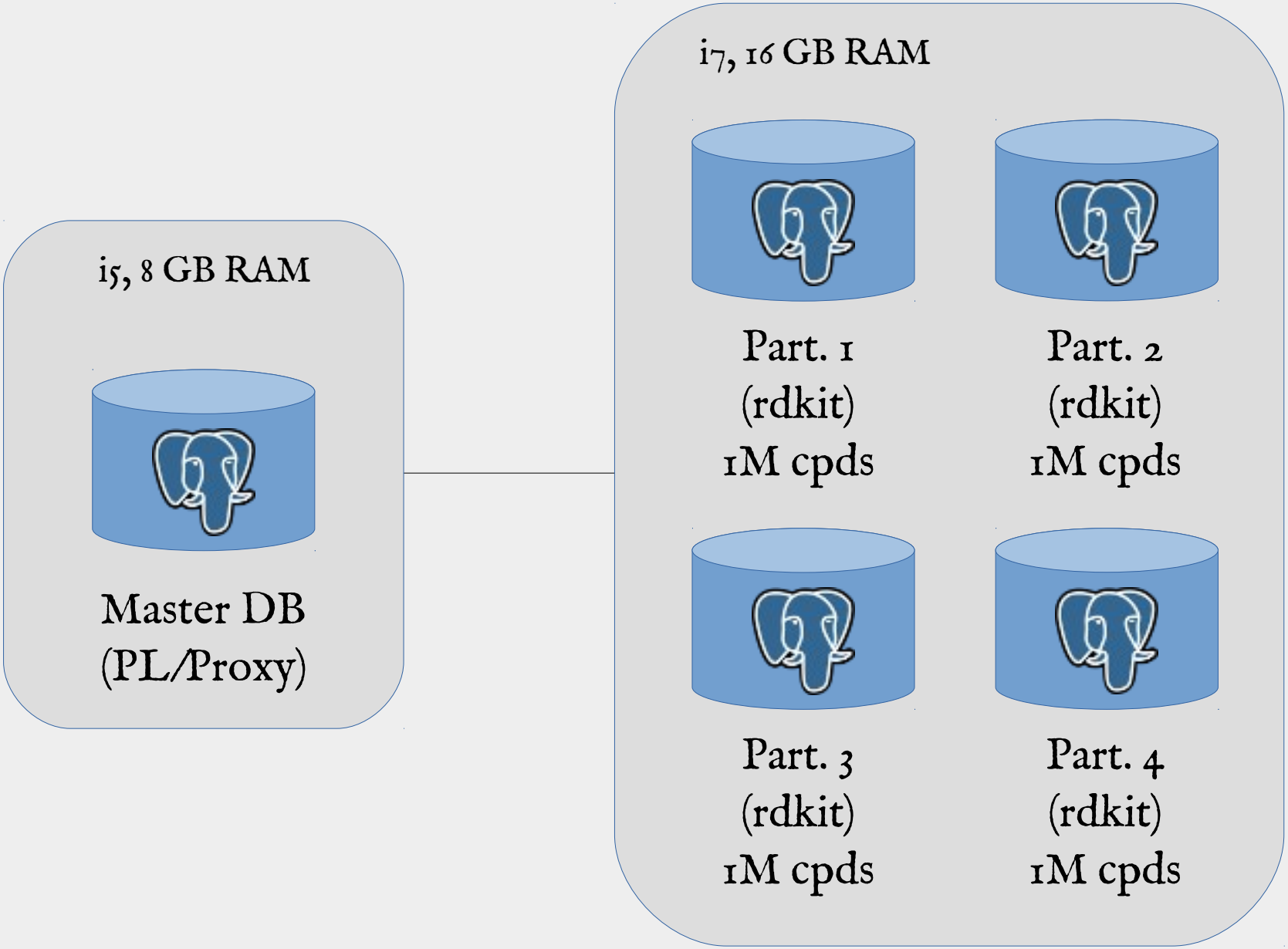
```
CREATE SERVER rdkitcluster FOREIGN DATA WRAPPER
plproxy
OPTIONS (connection_lifetime '1800',
                p0 'dbname=part0 port=6543',
                p1 'dbname=part1 port=6543'
                );
```

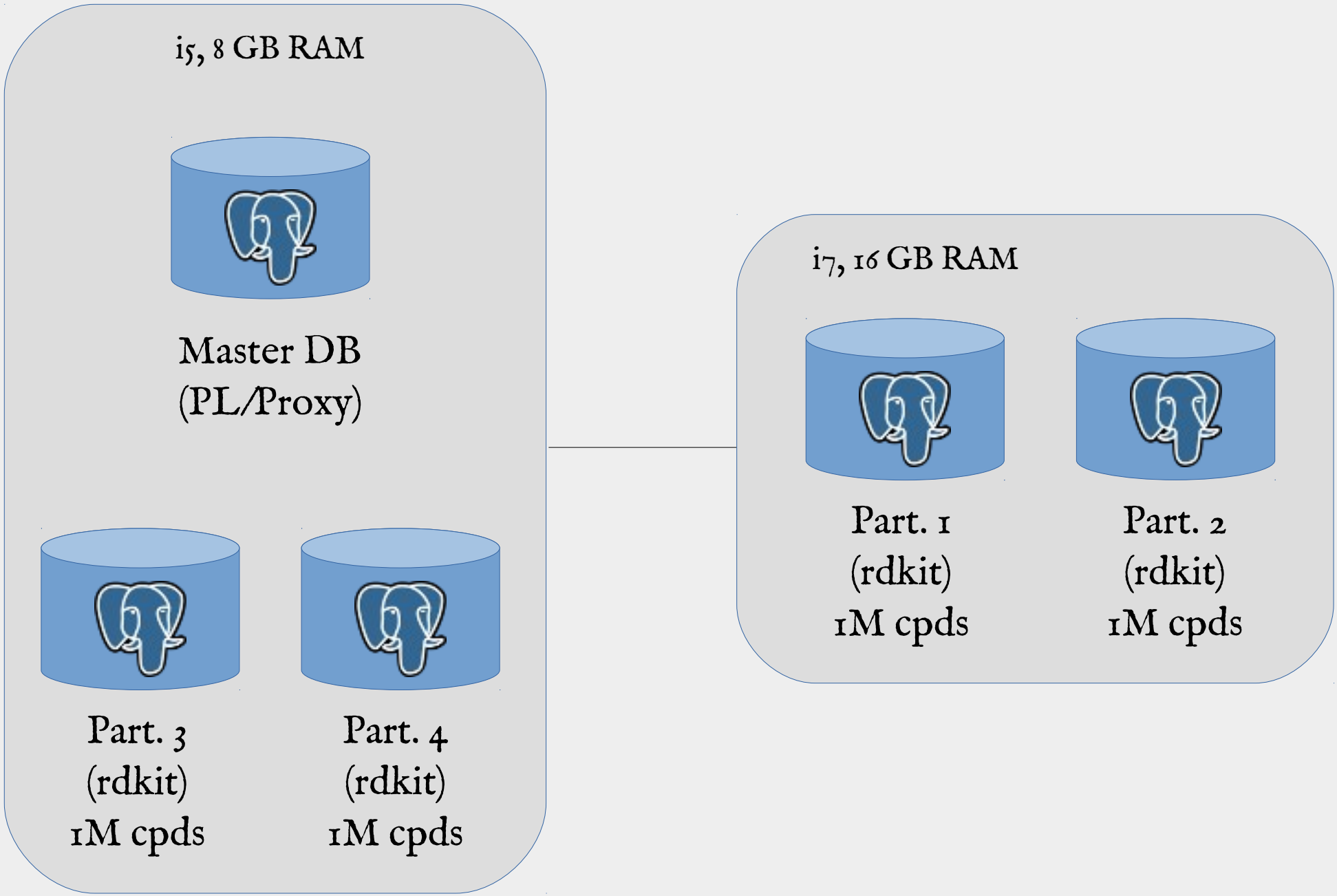
On the master node:

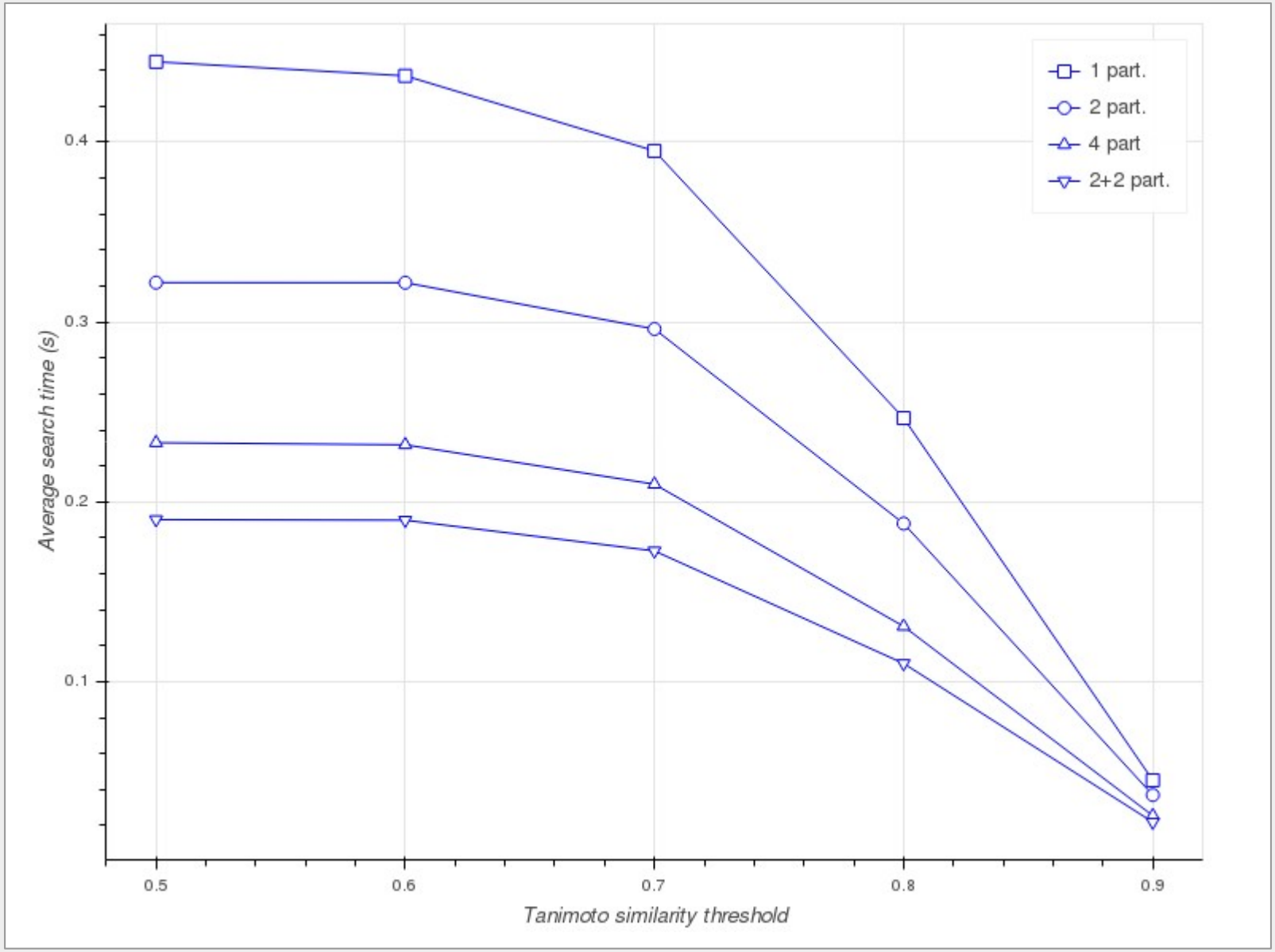
```
CREATE FUNCTION tanimoto_match_count(s text, t numeric)
RETURNS SETOF bigint as $$

    CLUSTER 'rdkitcluster';
    RUN ON ALL;

$$ language plproxy;
```


i5, 8 GB RAM
Master DB
(PL/Proxy)
i7, 16 GB RAM
Part. 1
(rdkit)
4M cpds


i5, 8 GB RAM
Master DB
(PL/Proxy)
i7, 16 GB RAM
Part. 1
(rdkit)
2M cpds
Part. 2
(rdkit)
2M cpds


i7, 16 GB RAM
Part. 1
(rdkit)
1M cpds
Part. 2
(rdkit)
1M cpds
Part. 3
(rdkit)
1M cpds
Part. 4
(rdkit)
1M cpds
i5, 8 GB RAM
Master DB
(PL/Proxy)


i5, 8 GB RAM
Master DB
(PL/Proxy)
Part. 3
(rdkit)
1M cpds
Part. 4
(rdkit)
1M cpds
i7, 16 GB RAM
Part. 1
(rdkit)
1M cpds
Part. 2
(rdkit)
1M cpds



Average execution time of Tanimoto match count queries for different threshold values (4M cpds, morgan bfp, rad. 2, 1024 bits)

Ongoing work

- Evaluate further optimization of the GiST bfp similarity index
- Explore using SP-GiST
- Extend the review and refactoring to other types and operators
- Continue exploring sharding solutions

Takeaways

- The RDKit PostgreSQL cartridge can be significantly faster than we got used to think
- But try not to use a low (< 0.7) similarity threshold
- Test/profile before using the default bfp size
- With a limited effort, sharding may provide an independent performance improvement

Thank you