Range Filters

Given a set $S$ of $n$ keys, a range filter is a space-efficient data structure that answers range emptiness queries with a false positive probability of at most $\varepsilon$.

$[a, b] \cap S = \emptyset? \quad [a', b'] \cap S = \emptyset? \quad [a'', b''] \cap S = \emptyset?$

- Generalises Bloom filters from point to range queries
- Reduces I/Os of range queries in LSM-based storage engines

Grafite: An Optimal Robust Range Filter

1. Apply a locality-preserving hash function to the keys
   \[ h(x) = (x + q((x/r))) \mod r \]
   Properly chosen to bound FPR and space

2. Compress the hash codes with the Elias-Fano code

3. Solve queries in hash space

**Theorem.** Given a space budget of $B$ bits/key, the query time of Grafite is $O(1)$ and the false positive probability $\varepsilon$ is no more than $\ell/2^{B-2}$, where $\ell$ is the query range size.

Issues with Current Range Filters

- Highly complex
  Sophisticated designs, hard to evaluate and deploy
- Fragile
  Inconsistent FPR and query times across different datasets
- Adversarial queries
  Easy to issue queries that result in false positives (thus I/Os)

Grafite works robustly out of the box by just specifying $B$ or $\varepsilon$.

Experiments with Robust Range Filters

- Grafite
- Rosetta
- REncoder

Space-vs-FPR
Grafite is up to 4 orders of magnitude more effective than REncoder, and up to 5 than Rosetta.

Query time
Grafite is about 1 order of magnitude faster than REncoder, and 2 than Rosetta.

Construction time
Grafite is up to 8 orders of magnitude faster than REncoder and 10 than Rosetta.

Take-home message. If robustness guarantees are needed regardless of input data and future queries, Grafite is the range filter of choice.

Bucketing: A Simple Heuristic Range Filter

1. Divide universe into buckets of equal size $S$
2. Mark non-empty buckets with a compressed bit-vector
3. Solve queries by mapping ranges to bit-vector positions

Bucketing offers, simultaneously, FPR very close to the best-performing heuristic range filters, 5–13× faster query workloads, and 5–24× faster construction.

Experiments with Heuristic Range Filters

- Bucketing
- SBR
- Rosetta
- REncoder
- REncoderSS
- REncoderDS

Take-home message. Heuristic range filters sacrifice robustness to work well on some inputs only, but simpler solutions like Bucketing exist.