Adaptive filter scheduling in the Parquet decoder

`pushdown_filters=on` shouldn't be a per-query lottery.

Adrian Garcia Badaracco · DataFusion · May 2026


center

`pushdown_filters=on` is a per-query lottery

Mode	Decode pattern per row group	Cost when filter is unselective
`pushdown=off`	download projection + filter, apply filter in memory via `FilterExec`	IO to download and compute to decode and then mask projection columns
`pushdown=on`	download 1 filter's columns at a time -> iteratively accumulate mask -> download projection	smaller IOs, more computation overall

`pushdown_filters` on by default has been an open ask for years (#3463) — blocked by exactly this lottery (#20324).

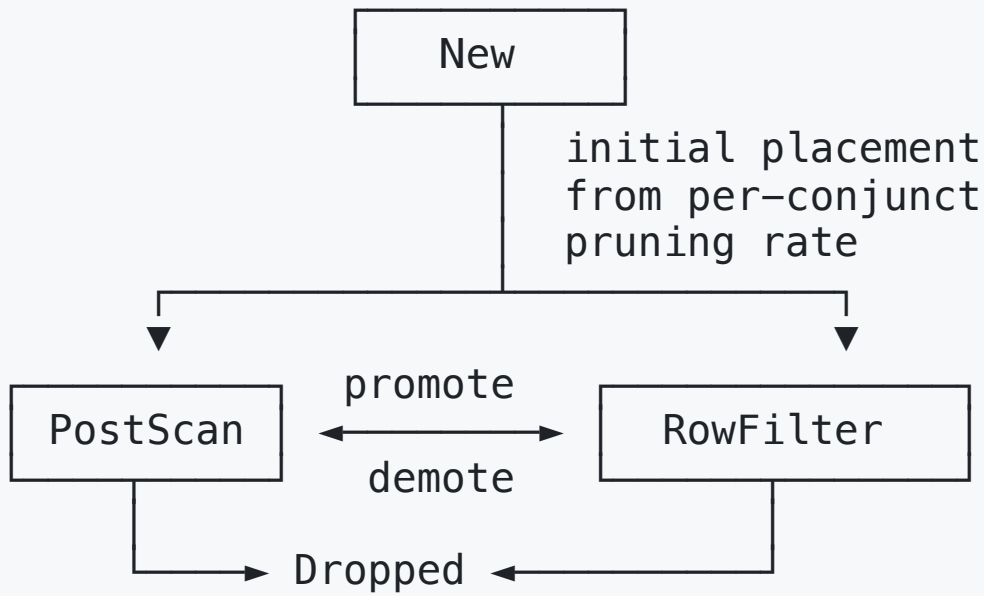
When pushdown wins big

```
SELECT *
FROM hits
WHERE "URL" LIKE '%google%';
```

When pushdown loses big

```
SELECT "SearchEngineID", "SearchPhrase"
FROM hits
WHERE "SearchPhrase" <> '';
```

The proposal: per-filter, adaptive, in-decoder



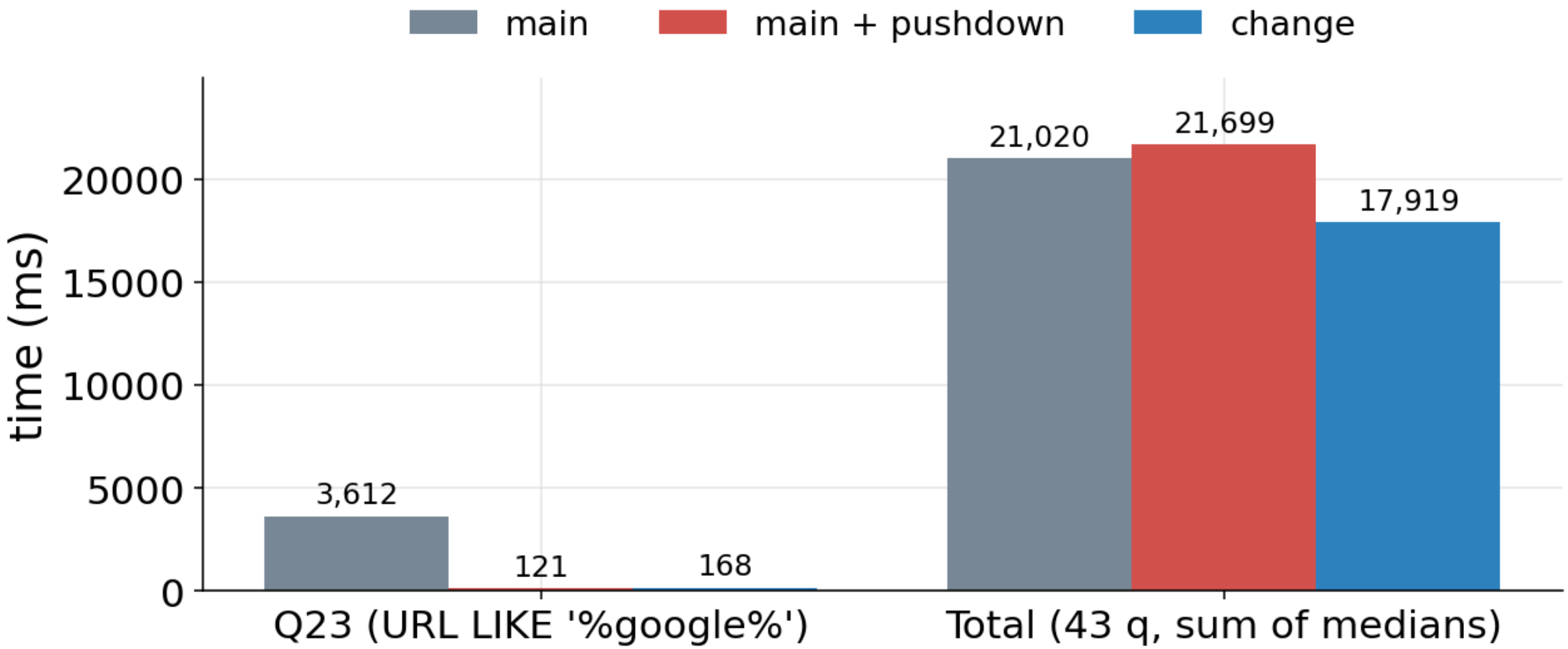

```
per filter:
  - rows seen / matched
  - eval time (ns)
  - bytes seen
  - Welford running stats


decision metric:
  scatter-aware
  bytes-saved-per-second
  with one-sided CI

 (Dropped only for
 OptionalFilterPhysicalExpr)
```

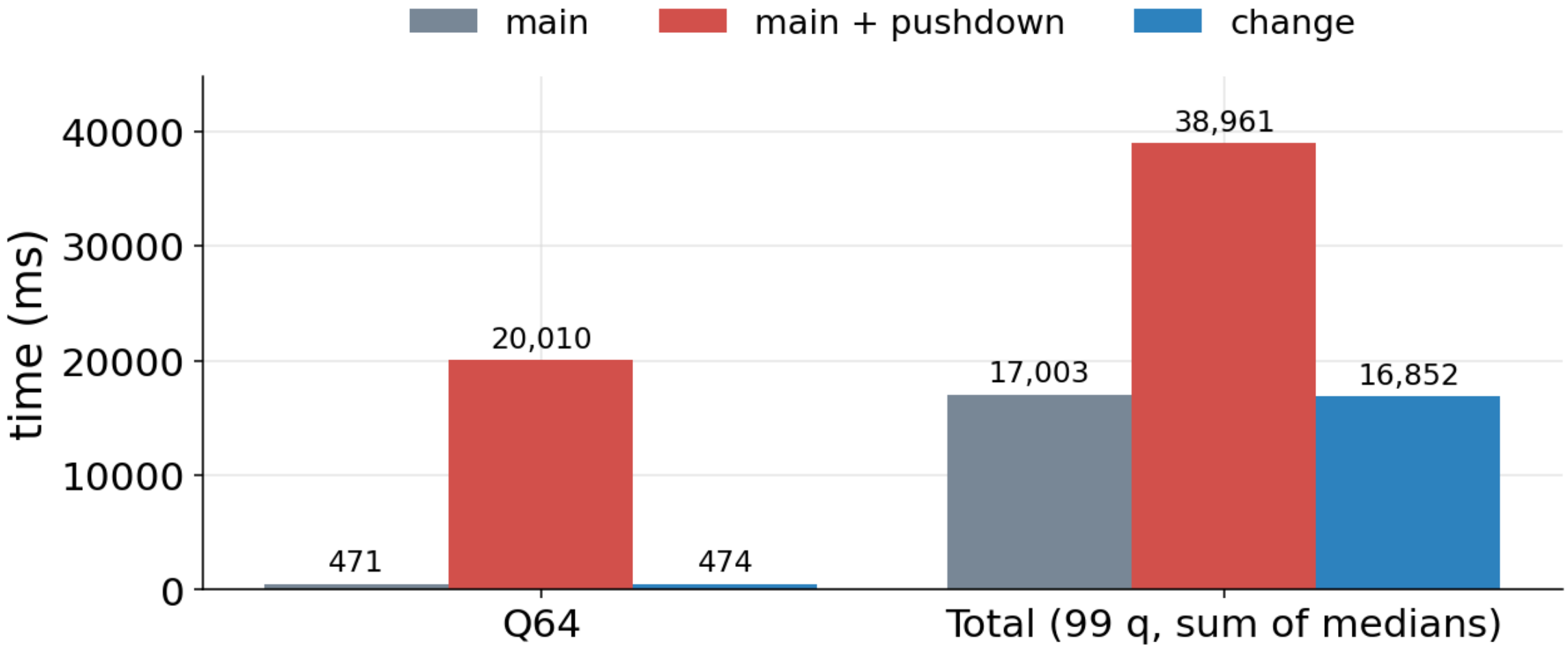
Decoder swaps strategy at every row-group boundary — same `ParquetPushDecoder`, same `BoxStream`, fresh `RowFilter`. `PushBuffers` carries through, so already-fetched bytes that survive the swap are reused.

ClickBench partitioned · SSD



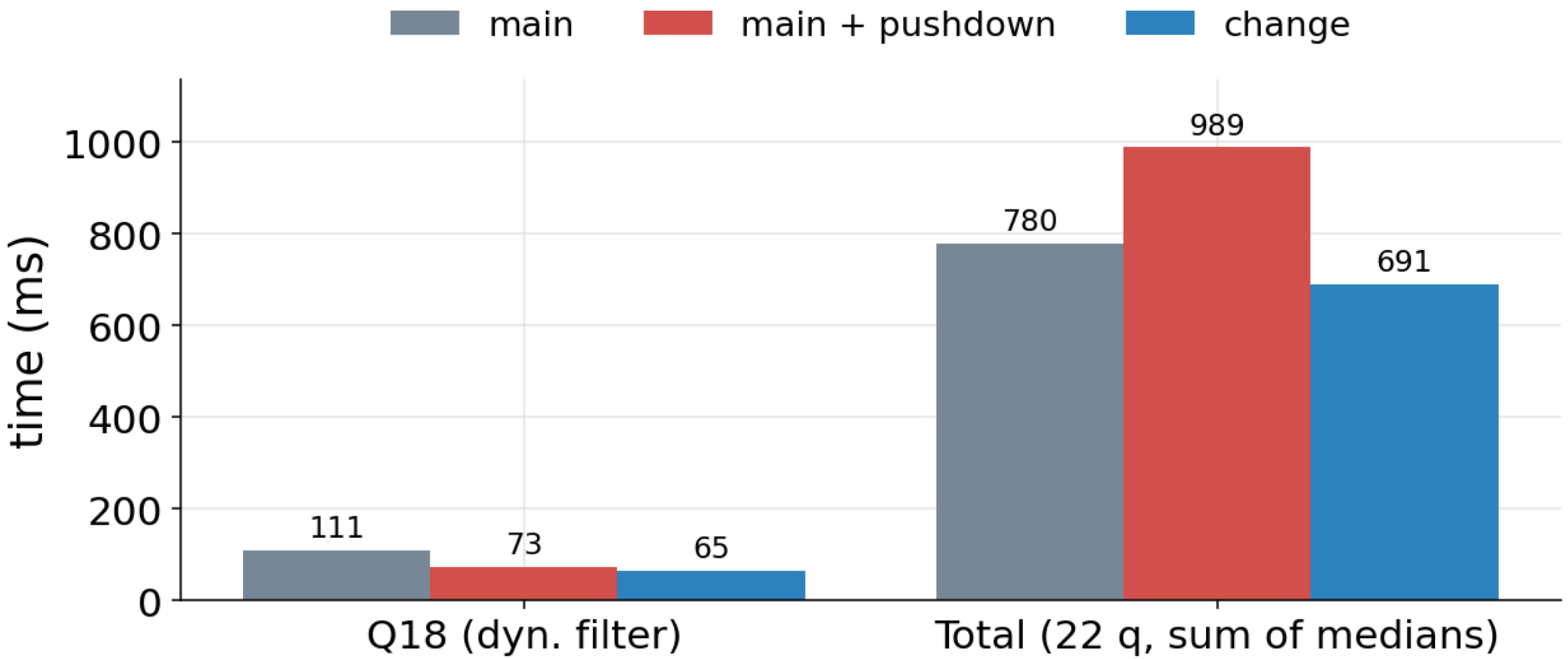
Q23: `SELECT * FROM hits WHERE URL LIKE '%google%' ORDER BY EventTime LIMIT 10`.

TPC-DS SF1 · SSD — Q64 carries the day



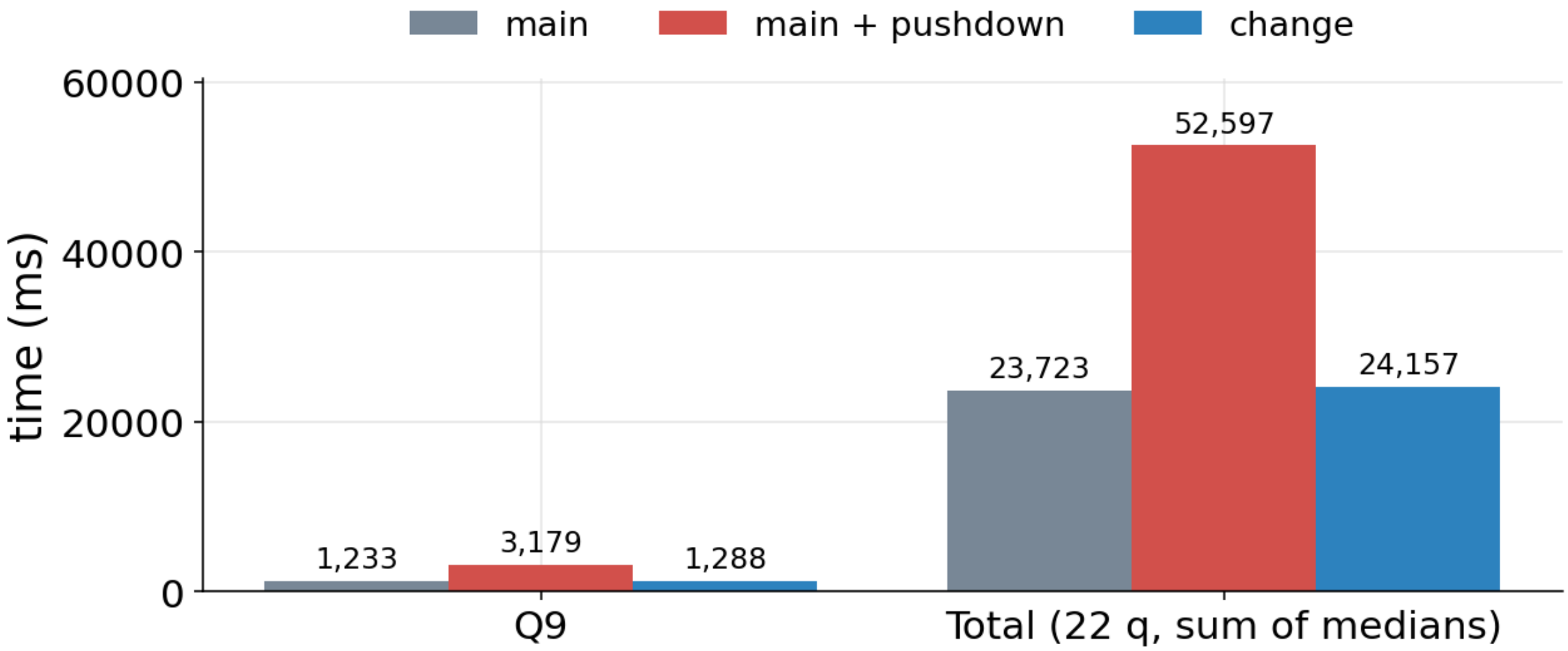
Q64: chained hash joins on `store_sales` publish `key BETWEEN min AND max` dynamic filters that aren't selective enough on this data to pay for row-level. `main + pushdown` runs them all row-level regardless; the change re-evaluates each populated dynamic filter's pruning rate against row-group stats and keeps the unselective ones post-scan.

TPC-H SF1 · SSD — the single-row-group case



TPC-H's `lineitem` is one file of one row group, so the picks have to be right on file open.
The pruning-rate prior promotes the filters that benefit: **Q18**'s `l_quantity IN (subquery)` dynamic filter lands at **0.59× of `main`** (46 of the 89 ms total delta); Q1/Q3/Q19 contribute smaller page-skipping wins.

Switch from SSD to S3: the picture amplifies



Same TPC-H, **simulated S3.** `main + pushdown` regresses **2.2x×** (52.6 s vs 23.7 s). **`change` 24.2 s — 1.02× of `main` ≈ flat; 0.46× of `main + pushdown`.**

Latency multipliers vs SSD: main 30×, main + pushdown 53×, change 35×. `pushdown=on` on `main` issues many small I/Os and pays a round-trip per range; the change avoids that by demoting unhelpful filters to post-scan automatically.

What's missing / what's next

Gap	Mechanism	Fix
Sub-row-group adaptation	swap point is the row-group boundary	arrow-rs `ParquetRecordBatchReader::pause` returning residual `RowSelection`
Cross-partition row-group balance	file is the unit of distribution	row-group-level morselization → PR #21766 (draft)
`pushdown=on` **by default**	requires the change to be at parity everywhere	this change closes the gap; flip #3463 once merged