Improving Postgres' Efficiency

Andres Freund

PostgreSQL Developer & Committer

Email: andres@anarazel.de
Email: andres.freund@enterprisedb.com
Twitter: @AndresFreundTec

anarazel.de/talks/pgconf-eu-warsaw-2017-10-26/efficiency.pdf

Efficiency

Analytics

	Choose Better Plan	Smarter Execution	Execute Plan Faster
Past Versions	Custom Plans (9.2) Join Removal (9.5) Multi-Column Statistics (10)	Prefetching for Bitmap-Scans (8.4) Index-Only Scans (9.2) BRIN (9.5) Grouping Sets (9.5, 10)	Sorting (9.5, 9.6) Better Hash-Tables New Expression Engine (10)
Future Versions	Selectivity Estimation Improvements CTE Inlining	“Cached” Nest-Loop Joins / “Probed” Hash Joins “Block” Nest-Loop Joins	Partial JIT Compilation (11?) Vectorized / Batched Execution (12?)

EDB POSTGRES

The Past

Choose A Better Plan: Multi-Column Statistics

- Selectivity Estimation
- Two Column Selectivity: likelihood(a) * likelihood(b)
- Correlation / Dependencies:
  - WHERE city = 'San Francisco' AND zipcode = '94158'
  - city = 'San Francisco' => 864k / 301M => 1/348
  - zipcode = '94158' => 4792 / 301M => 1/62184
  - Result: 301M * (1/348 * 1/62184) => ~13
- Fix in PostgreSQL 10+:

```
CREATE STATISTICS zips_and_cities
ON (zipcode, city)
FROM us_citizens;
```

Smarter Execution: Index-Only Scan

- ```
  SELECT l_shipdate, count(*)
  FROM lineitem
  WHERE l_shipdate BETWEEN '...' AND '...'
  GROUP BY l_shipdate
  ORDER BY count(*)
  LIMIT 1;
  CREATE INDEX "i_l_shipdate" ON lineitem(l_shipdate);
  ```
- Bitmap-Scan: 109900.866 ms, 1099789 buffers accessed
- Index-Only-Scan: 1317.177 ms, 38470 buffers accessed
- Requires: Regular (auto-)vacuum, index over all columns
- PostgreSQL 11: “covering indexes”
  ```

Smarter Execution: Prefetching

- SELECT l_shipdate, SUM(quantity),
  FROM lineitem
  WHERE l_shipdate BETWEEN '...' AND '...'
  GROUP BY l_shipdate
  ORDER BY sum(quantity) DESC
  LIMIT 1;
  CREATE INDEX "i_l_shipdate" ON lineitem(l_shipdate);
- Good on rotational disks, awesome on SSDs
- No-Prefetching: SET effective_io_concurrency = 0;
  Buffers: shared hit=3 read=1119735
  Time: 30032.860 ms
  Average-IO: ~260 MB/Sec
  Utilization: ~60%
- Prefetching: SET effective_io_concurrency = 512;
  Buffers: shared hit=3 read=1119735
  Time: 17688.256 ms
  Average-IO: ~525 MB/Sec
  Utilization: 100.00%

EDB POSTGRES

Faster Execution:
New Hash-Table & Expression Engine

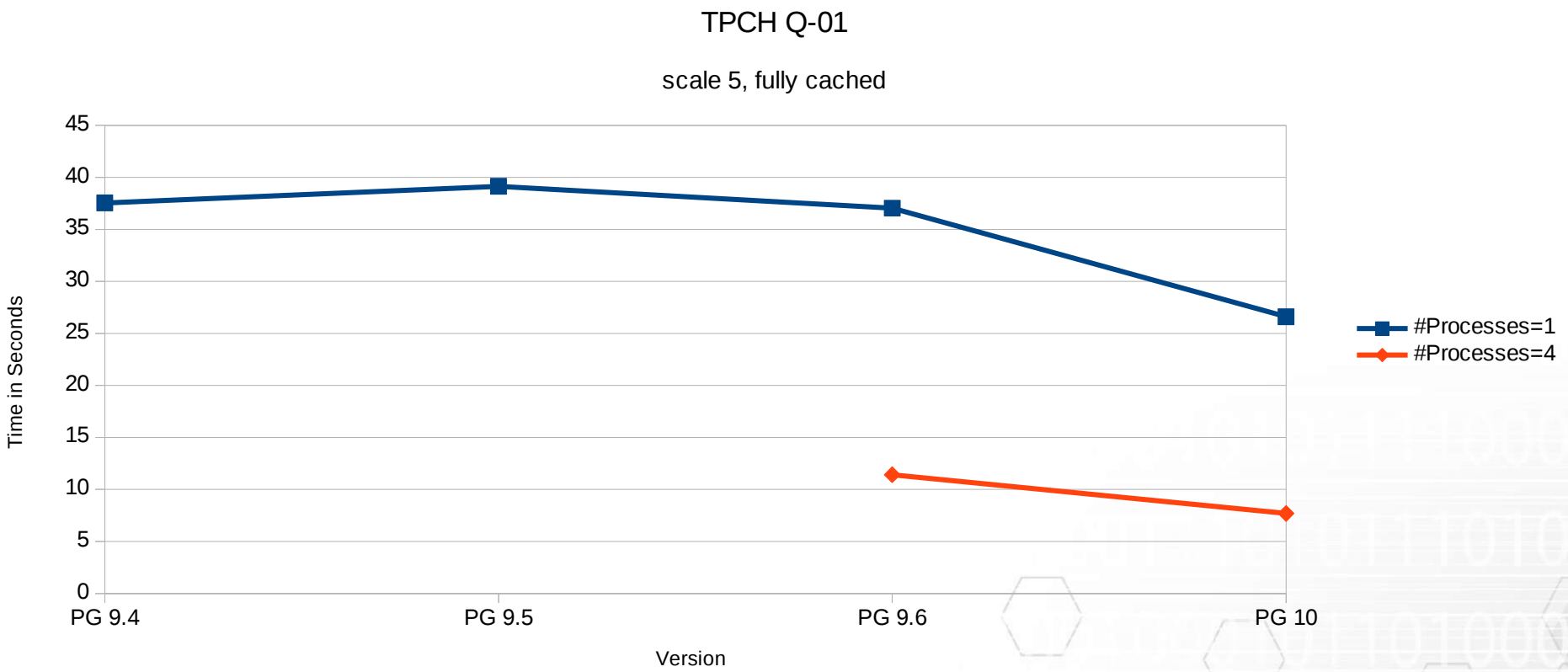
```
SELECT
    l_returnflag,
    l_linestatus,
    sum(l_quantity) AS sum_qty,
    sum(l_extendedprice) AS sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
    avg(l_quantity) AS avg_qty,
    avg(l_extendedprice) AS avg_price,
    avg(l_discount) AS avg_disc,
    count(*) AS count_order
FROM lineitem
WHERE l_shipdate <= date '1998-12-01' - interval '74 days'
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag, l_linestatus;
```

EDB POSTGRES

Faster Execution:
New Hash-Table & Expression Engine

```
Sort  (cost=4313533.34..4313533.36 rows=6 width=68)
  Sort Key: l_returnflag, l_linestatus
  ->  HashAggregate  (cost=4313533.16..4313533.26 rows=6 width=68)
        Group Key: l_returnflag, l_linestatus
        Output: …, sum(l_quantity), sum(l_extendedprice), sum(…), ...
        ->  Seq Scan on lineitem  (cost=0.00..1936427.80 rows=59427634 width=36)
              Filter: (l_shipdate <= '1998-09-18 00:00:00'::timestamp without time zone)
```

Faster Execution:
New Hash-Table & Expression Engine



The Future

Faster Execution: JIT Compilation

```
SELECT SUM(l_extendedprice * l_discount * l_quantity), count(*)
FROM lineitem
WHERE l_shipmode != 'MAIL'
```

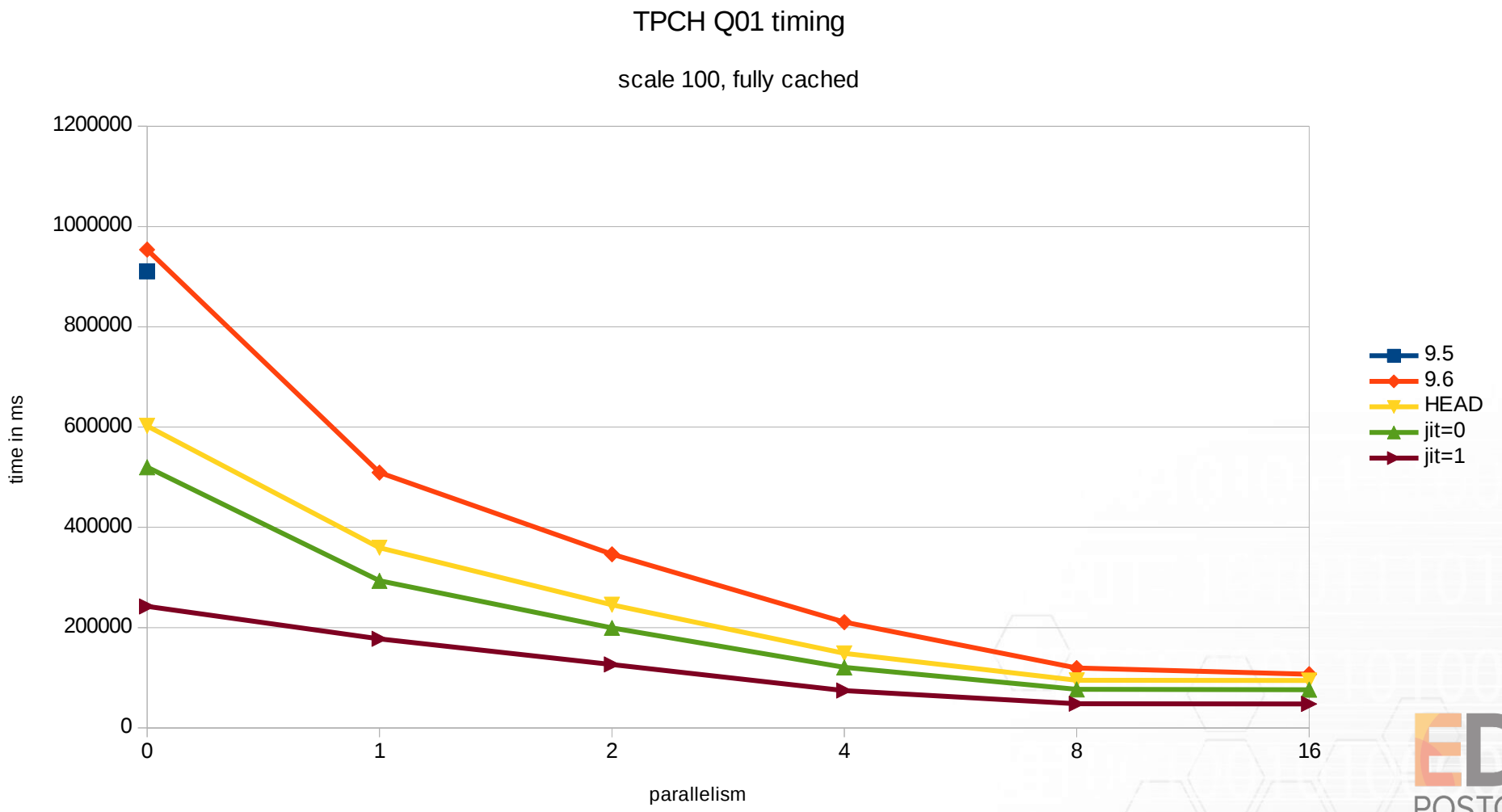
- This executes:
  - tuple deforming / accessing a row's columns
  - bpcharne (character(xx) != character(xx))
  - float8mul
  - int8inc, float8pl

- PG 10, fully cached, best of three:
  - 12856 ms
  - branches: 1616.296 M/sec
  - iTLB-load-misses: 126.42% of all iTLB cache hits
- JIT, fully cached, best of three
  - 6526 ms
  - branches: 1053.995 M/sec
  - iTLB-load-misses: 8.42% of all iTLB cache hits

Faster Execution: JIT Compilation

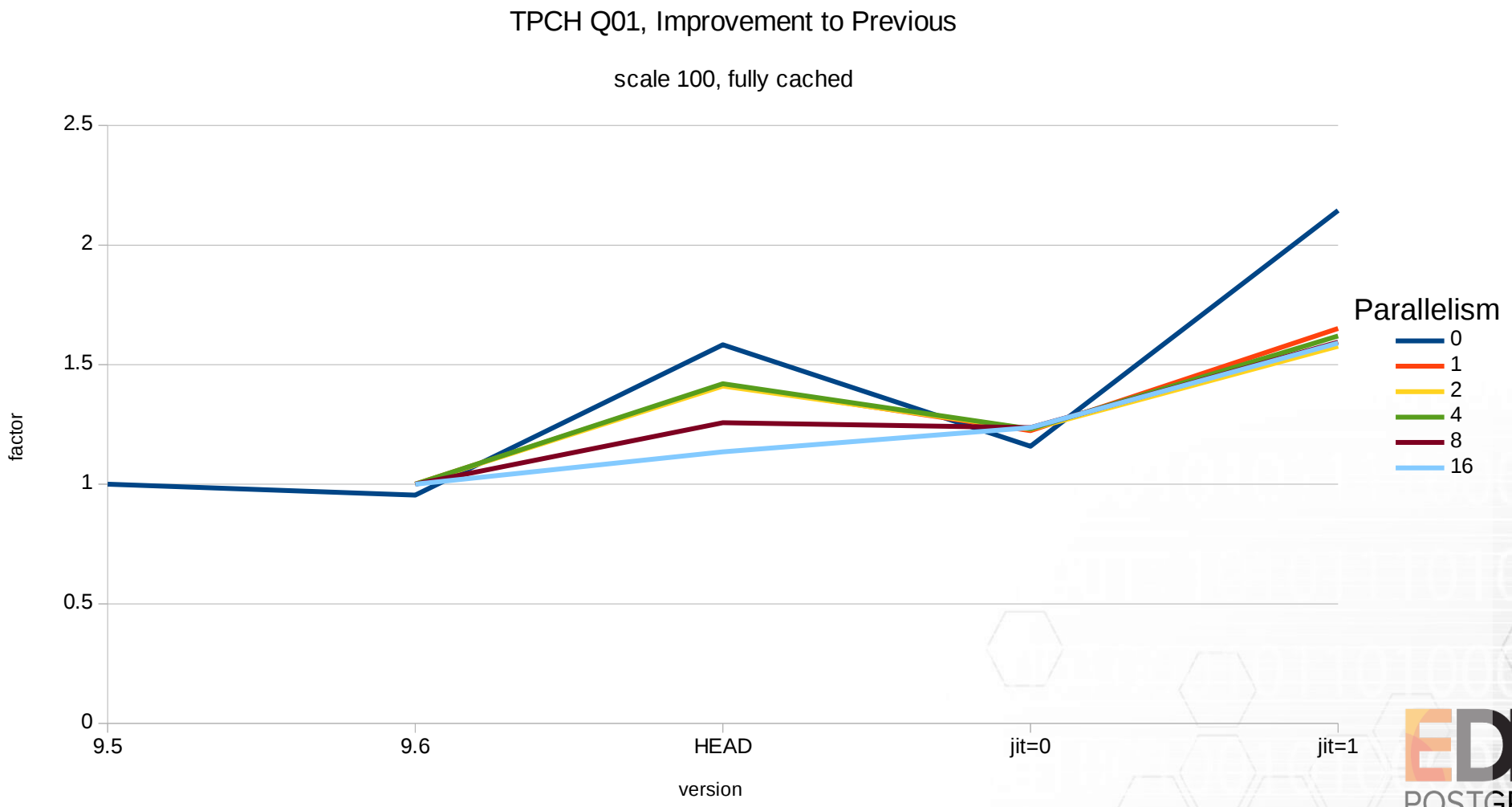
```
SELECT
    l_returnflag,
    l_linestatus,
    sum(l_quantity) AS sum_qty,
    sum(l_extendedprice) AS sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) AS sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) AS sum_charge,
    avg(l_quantity) AS avg_qty,
    avg(l_extendedprice) AS avg_price,
    avg(l_discount) AS avg_disc,
    count(*) AS count_order
FROM lineitem
WHERE l_shipdate <= date '1998-12-01' - interval '74 days'
GROUP BY l_returnflag, l_linestatus
ORDER BY l_returnflag, l_linestatus;
```

Faster Execution: JIT Compilation

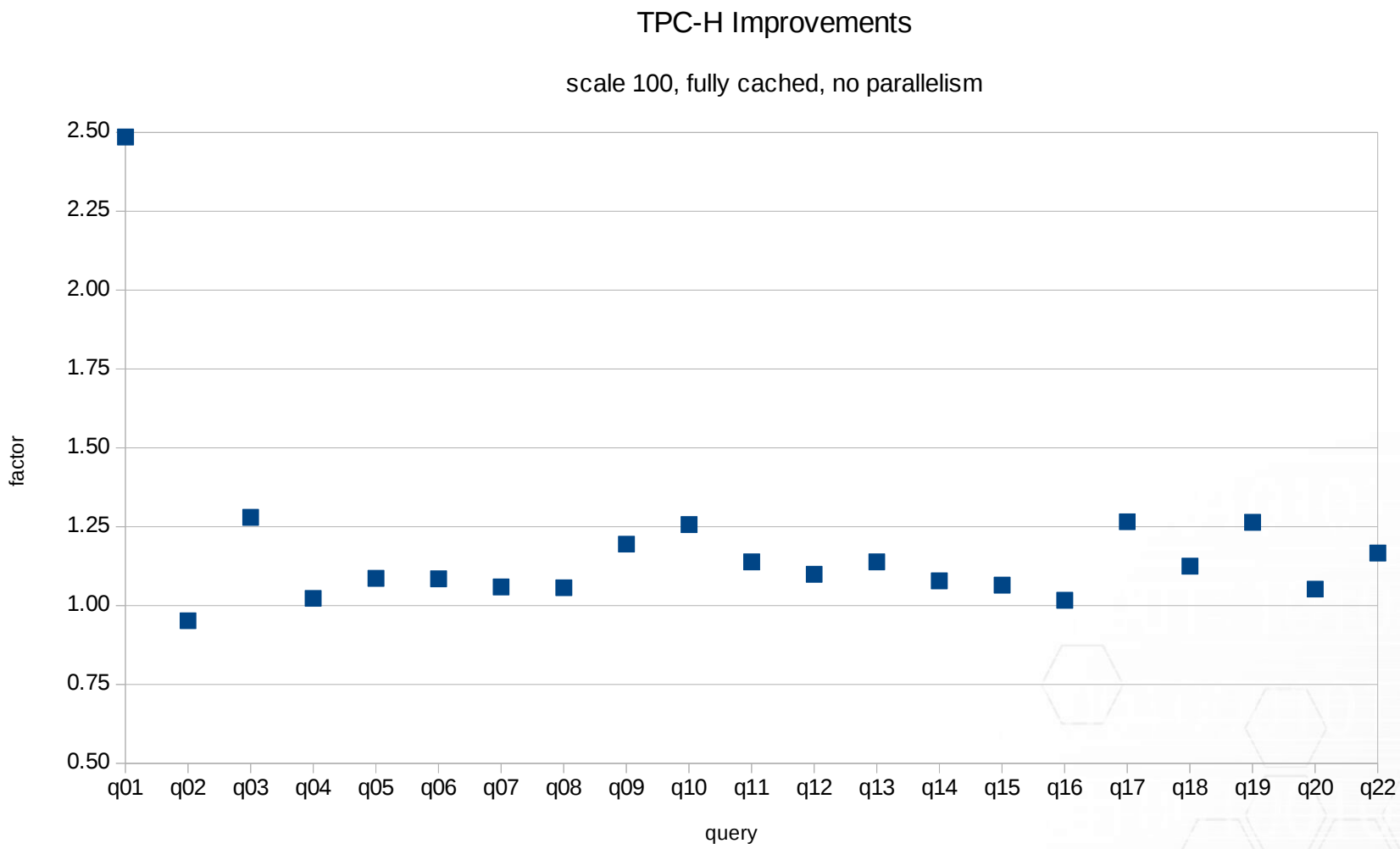


EDB POSTGRES

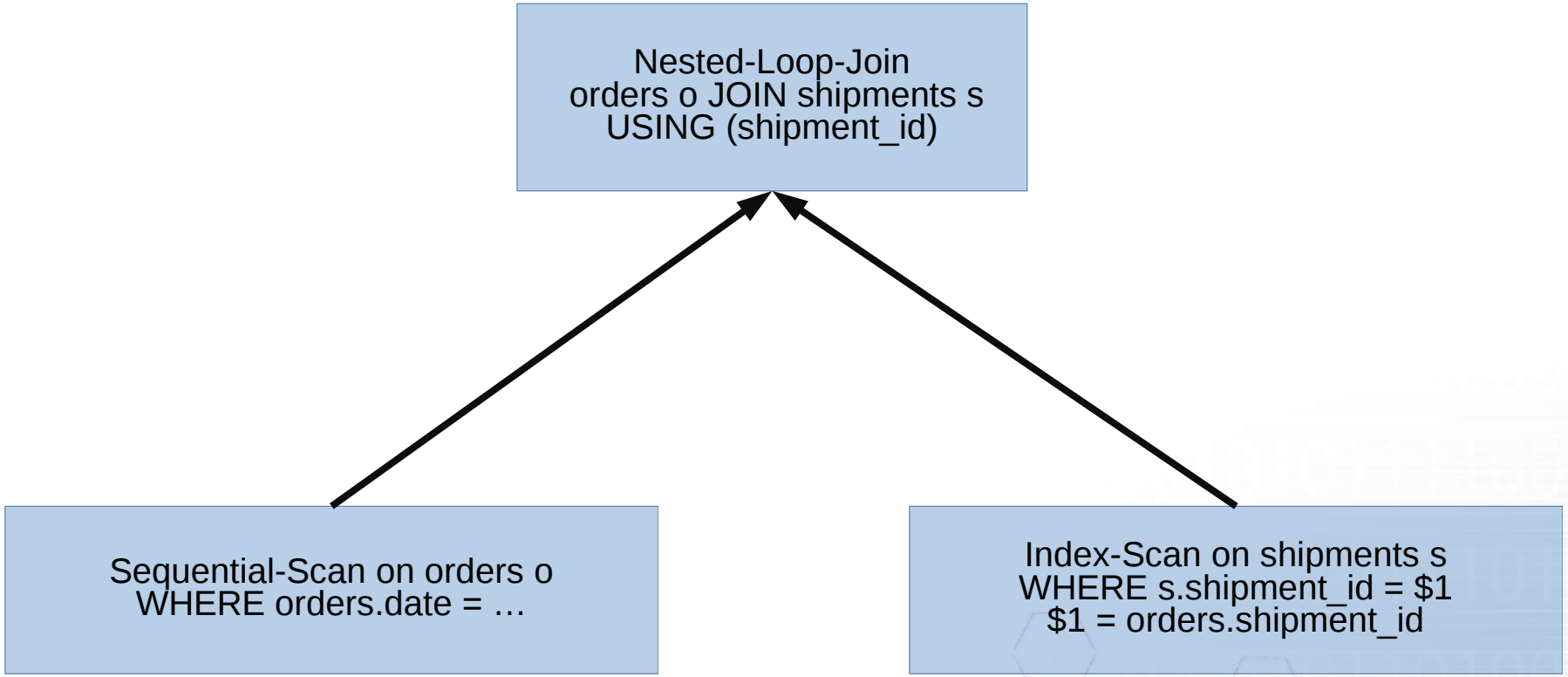
Faster Execution: JIT Compilation



Faster Execution: JIT Compilation

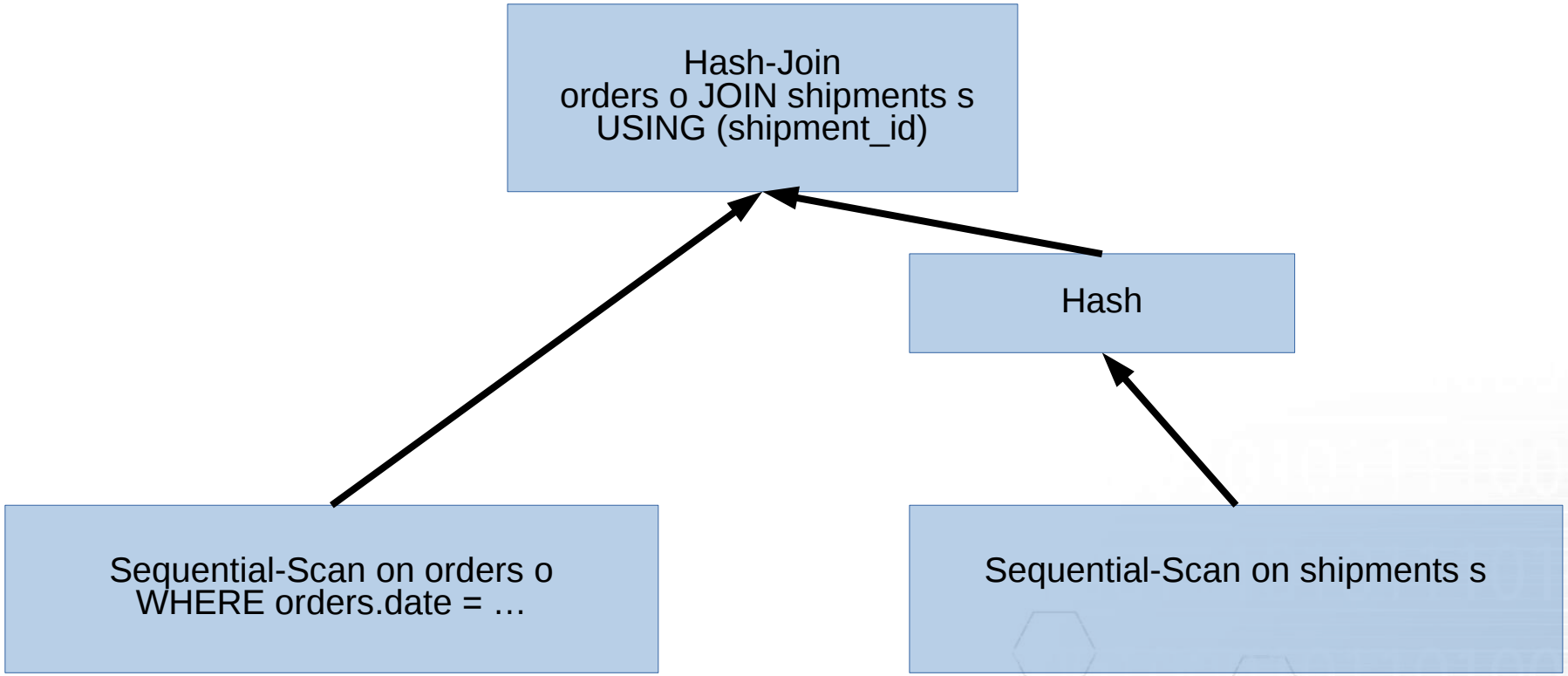


Smarter Execution: “Cached” Nested-Loop Joins

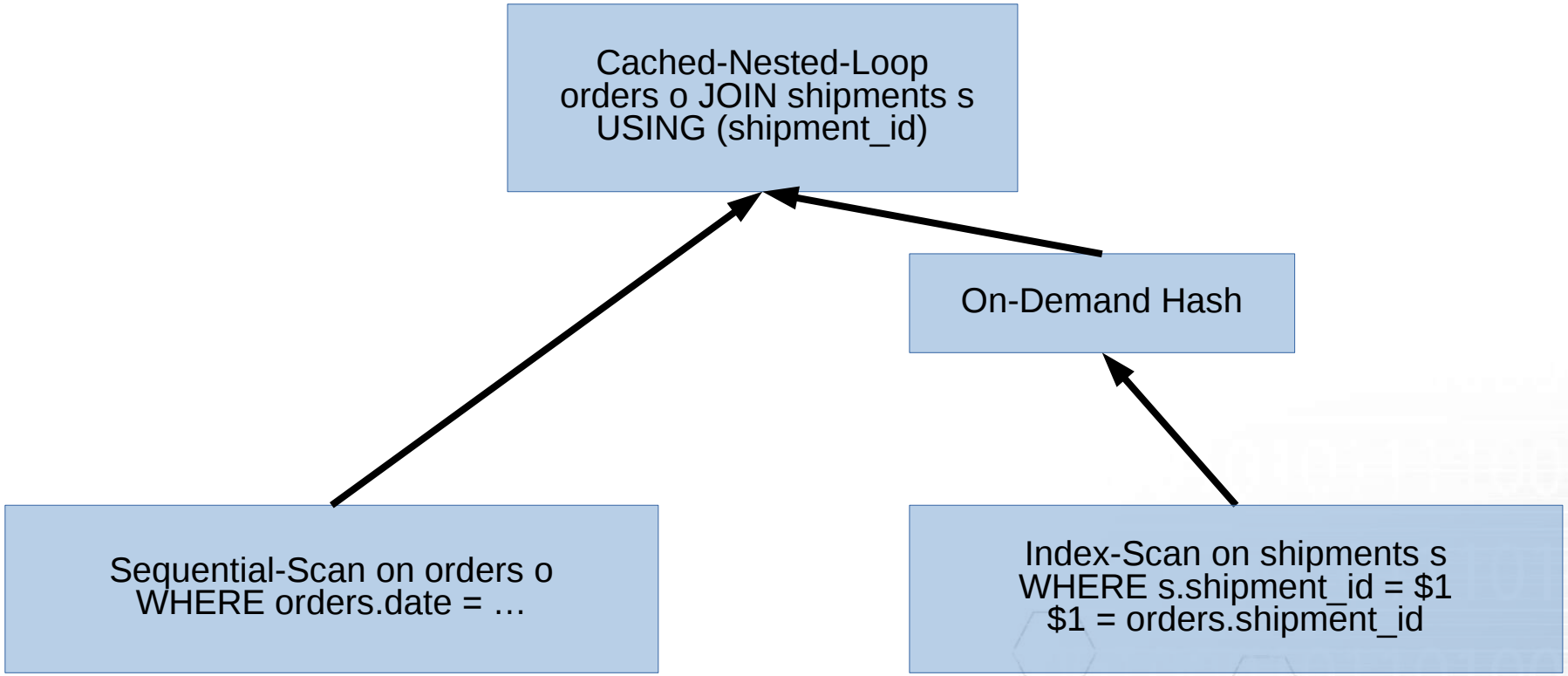


EDB POSTGRES

Smarter Execution: “Cached” Nested-Loop Joins



Smarter Execution: “Cached” Nested-Loop Joins



EDB POSTGRES

Better Planning: CTE Inlining / Barrier

```
WITH per_day AS (
    SELECT l_shipdate, count(*)
    FROM lineitem
    GROUP BY l_shipdate
)
  SELECT *
  FROM per_day
  WHERE l_shipdate = '1998-01-01'
UNION ALL
  SELECT *
  FROM per_day
  WHERE l_shipdate = '1998-01-02';
```

- Non-Inlined, fully-cached: 162376 ms, 2558910 buffer accesses
- Inlined, fully-cached: 148 ms, 2098 buffer accesses

Improving Postgres' Efficiency

Andres Freund

PostgreSQL Developer & Committer

Email: andres@anarazel.de
Email: andres.freund@enterprisedb.com
Twitter: @AndresFreundTec

anarazel.de/talks/pgconf-eu-warsaw-2017-10-26/efficiency.pdf