JITing PostgreSQL

Andres Freund

PostgreSQL Developer & Committer

Email: andres@anarazel.de
Email: andres.freund@enterprisedb.com
Twitter: @AndresFreundTec
anarazel.de/talks/edb-pune-jit-2017-11-03/jit.pdf

Motivation

TPC-H Q01

```
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;
```

What about Parallelism?

Efficiency

Identified Bottlenecks

- plain expression evaluation
- aggregate transition values
- tuple deforming
- grouping comparisons & hashing (aggregation, set functions)
- sorting!

What's JIT?

- create executable code
  - when needed
  - without invoking external program
  - takes time
  - optimizer
- linearize Expression Evalution
- remove branches from Tuple Deforming
- inline functions
  - no indirect function calls
  - elide function call overhead
  - cross function optimization

Existing Parts

- Expression evaluation machinery (v10)
- Known tupledescs for scan, left, right tree
  - important for JITed deforming
- Jit infrastructure (lifetime, errors, settings, inlining)
  - lifetime: EState integration
  - errors: resowner integration
  - inlining: create bitcode for postgres code, load at JIT time
- Expand executor to use expression engine more
  - Aggregate transition functions
  - grouping comparisons (aggregates, setops, ...)

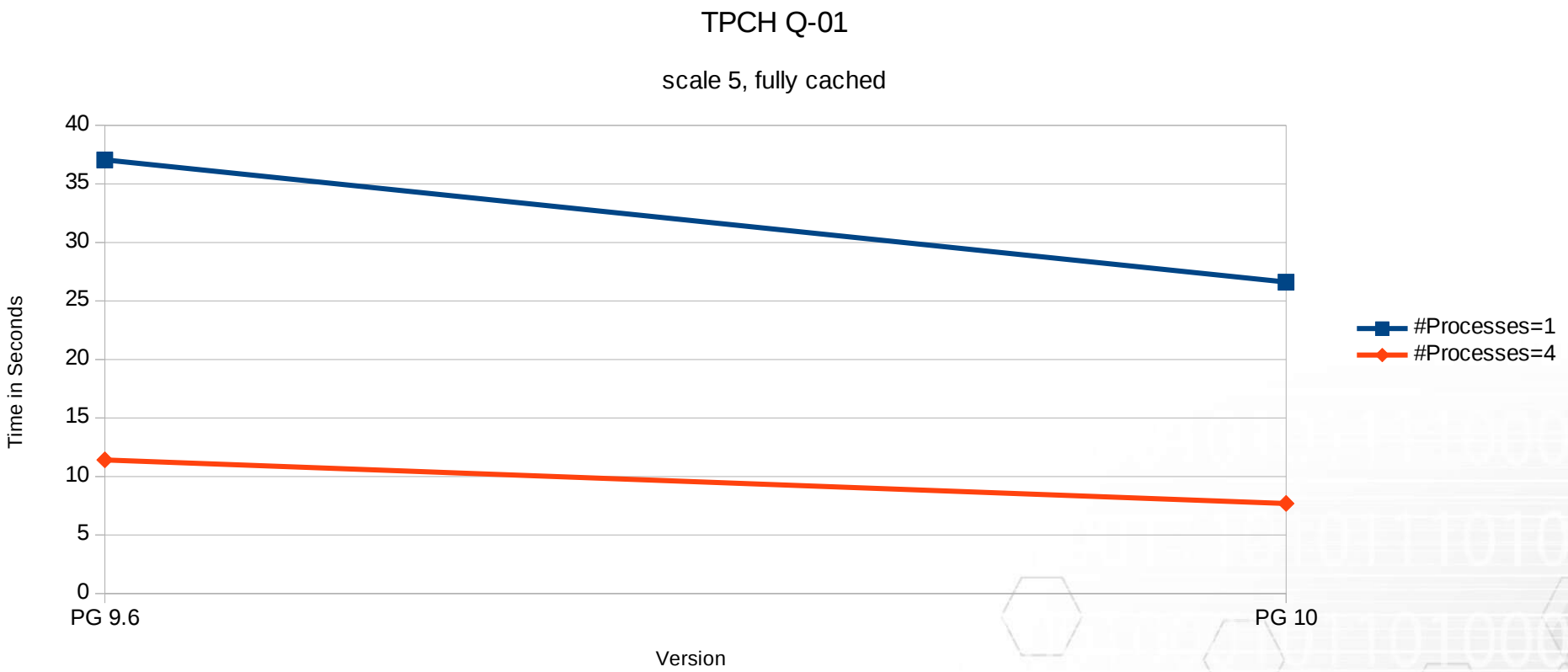
Old Expression Evaluation Engine

- WHERE a.col < 10 AND a.another = 3
  - ExecEvalExpr()
    - ExecEvalAnd()
      - ExecEvalExpr()
        - ExecEvalFunc() # int4lt
          - ExecEvalScalarVar() # a.col
            - slot_getattr
              - slot_deform_tuple # 1
          - ExecEvalConst() # 10
      - ExecEvalExpr()
        - ExecEvalFunc() # int4eq
          - ExecEvalScalarVar() # a.another
            - slot_getattr
              - slot_deform_tuple # 2
          - ExecEvalConst() # 3

New Expression Evaluation Engine

- WHERE a.col < 10 AND a.another = 3
  - EEOP_SCAN_FETCHSOME (deform necessary cols)
  - EEOP_SCAN_VAR (a.col)
  - EEOP_CONST (10)
  - EEOP_FUNCEXPR_STRICT (int4lt)
  - EEOP_BOOL_AND_STEP_FIRST
  - EEOP_SCAN_VAR (a.another)
  - EEOP_CONST (3)
  - EEOP_FUNCEXPR_STRICT (int4eq)
  - EEOP_BOOL_AND_STEP_LAST (AND)

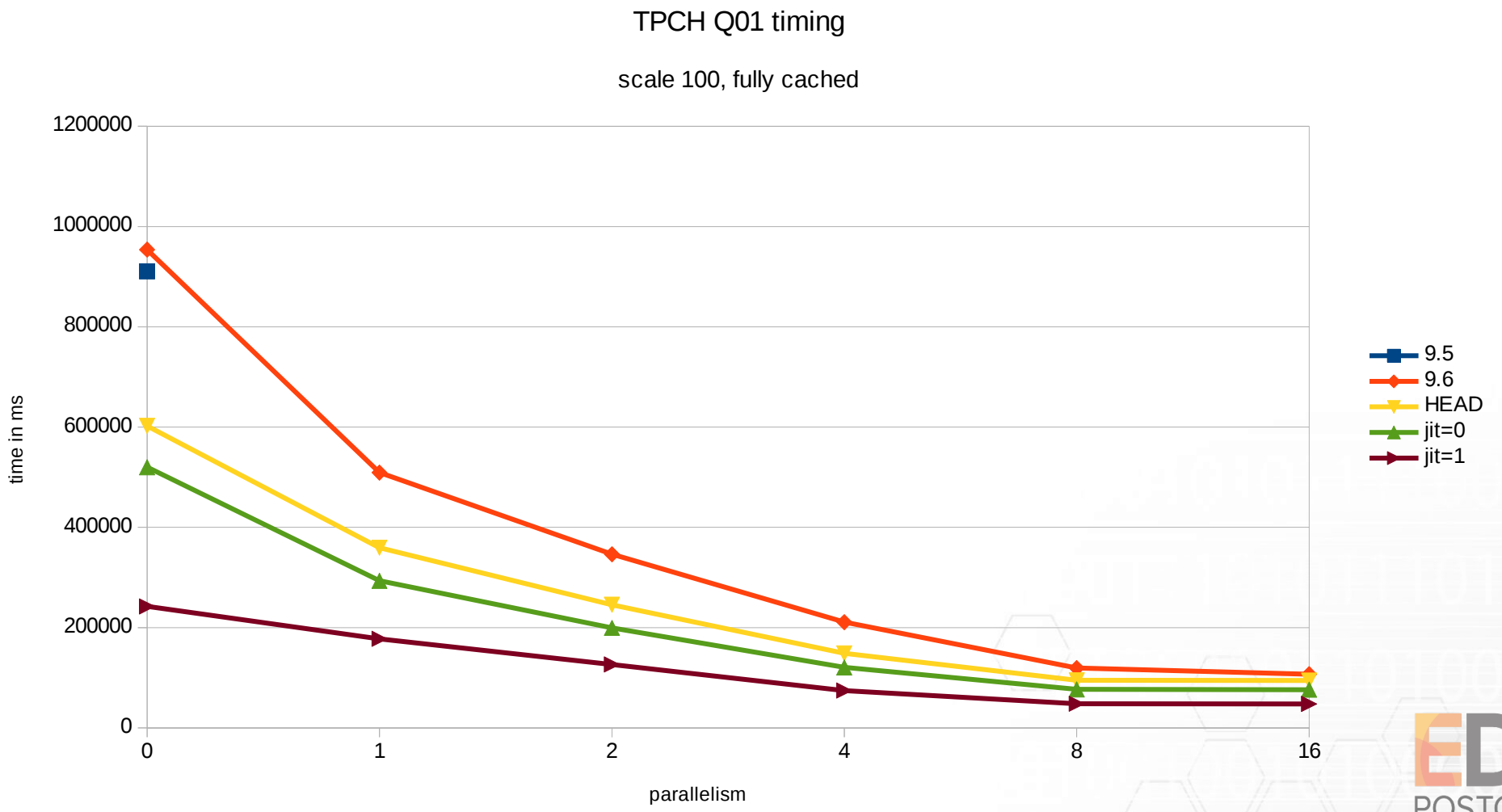
Faster Execution:
New Hash-Table & Expression Engine



Tuple Deforming

- Datatypes Known
  - type alignment known
  - possibly previous alignment known
- NOT NULL / NULL is known
- Number of to-be-deformed columns is known
- If there's NOT NULL columns, existance of columns might be known

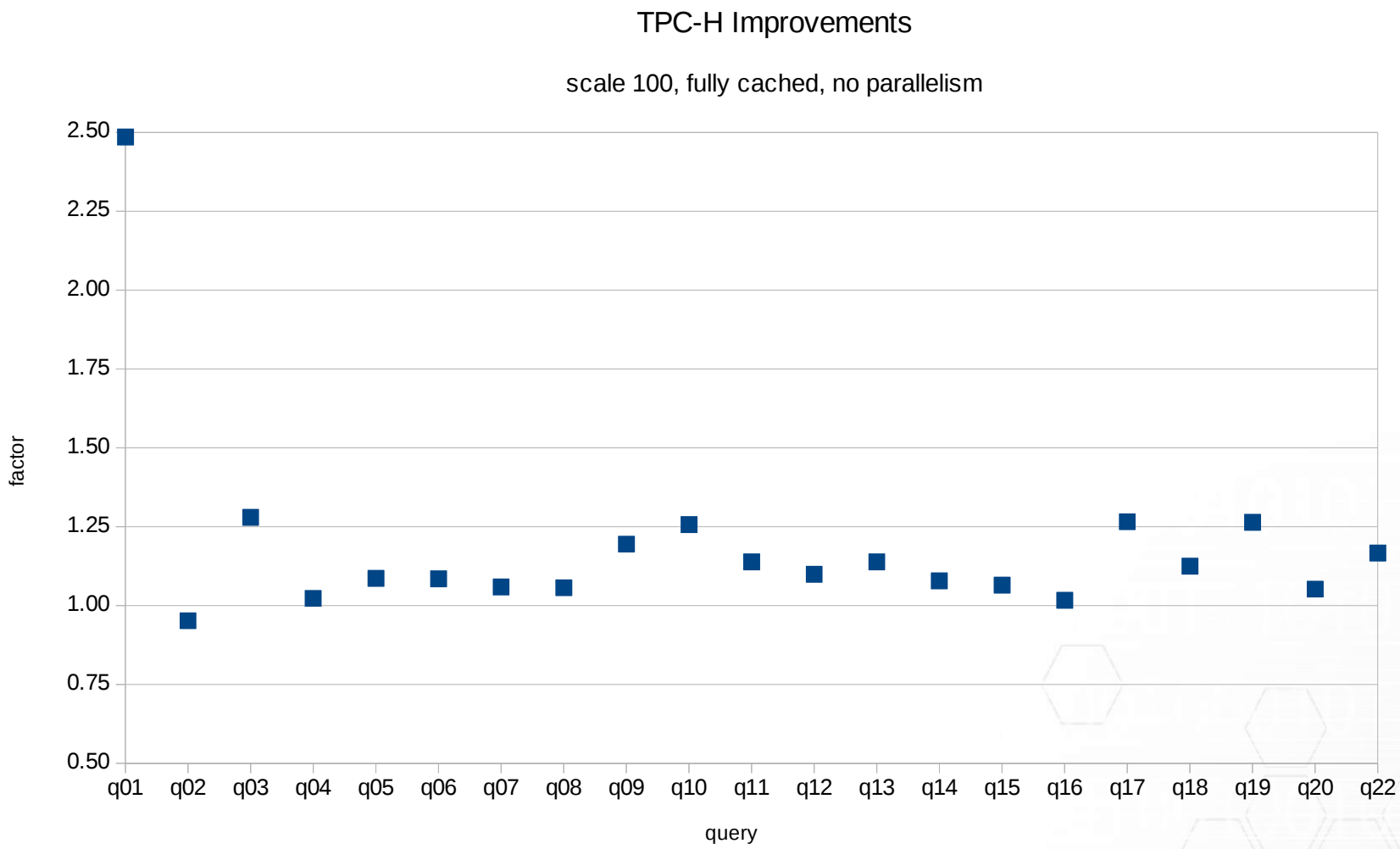
Faster Execution: JIT Compilation



EDB POSTGRES

Example Code

Faster Execution: JIT Compilation



Missing Parts

- planner:
  - when to jit and how much to optimize
- executor:
  - OOM handling
  - expression engine for hashing
  - make tupledescs known in more places
  - determine NOT NULL in upper tupledescs
- JIT:
  - correct inlining implementation
  - allow extensions to do inlining
  - improve code generation
  - Caching!