Speeding up Query Execution

Andres Freund
PostgreSQL Developer & Committer
Citus Data – citusdata.com - @citusdata

http://anarazel.de/talks/pgconf-us-2017-03-29/jit.pdf

Motivation

Scalability
Efficiency
Concurrency

OLTP
vs
OLAP
vs
HTAP

Online Transactional Processing
vs
Online Analytical Processing
vs
Hybrid Transactional / Analytical Processing

Amdahl's Law

Two independent parts **A** **B**

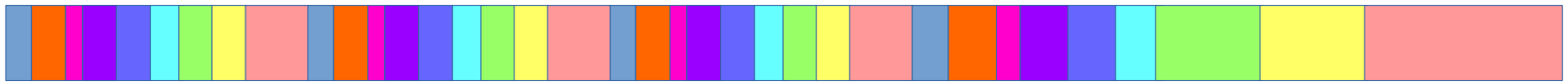
Original process

Make **B** 5x faster

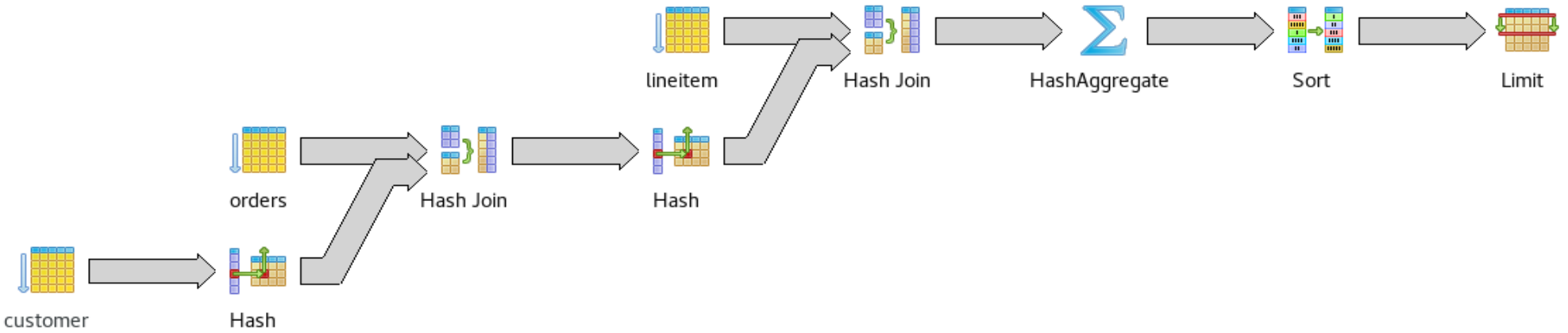
Make **A** 2x faster

https://commons.wikimedia.org/wiki/File:Optimizing-different-parts.svg

A
B
C
D
E
F
G
H
I

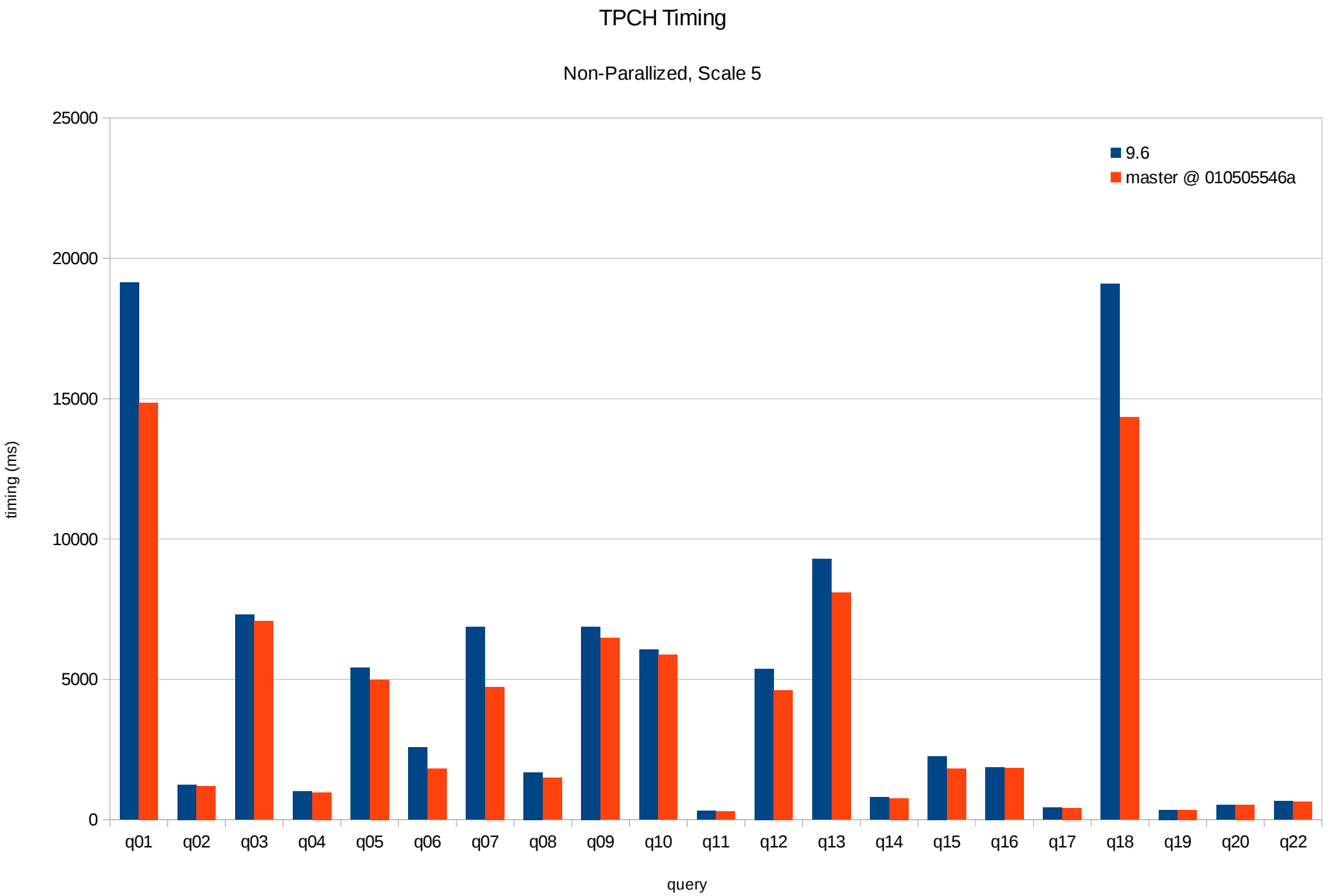


Batch Tuple Processing

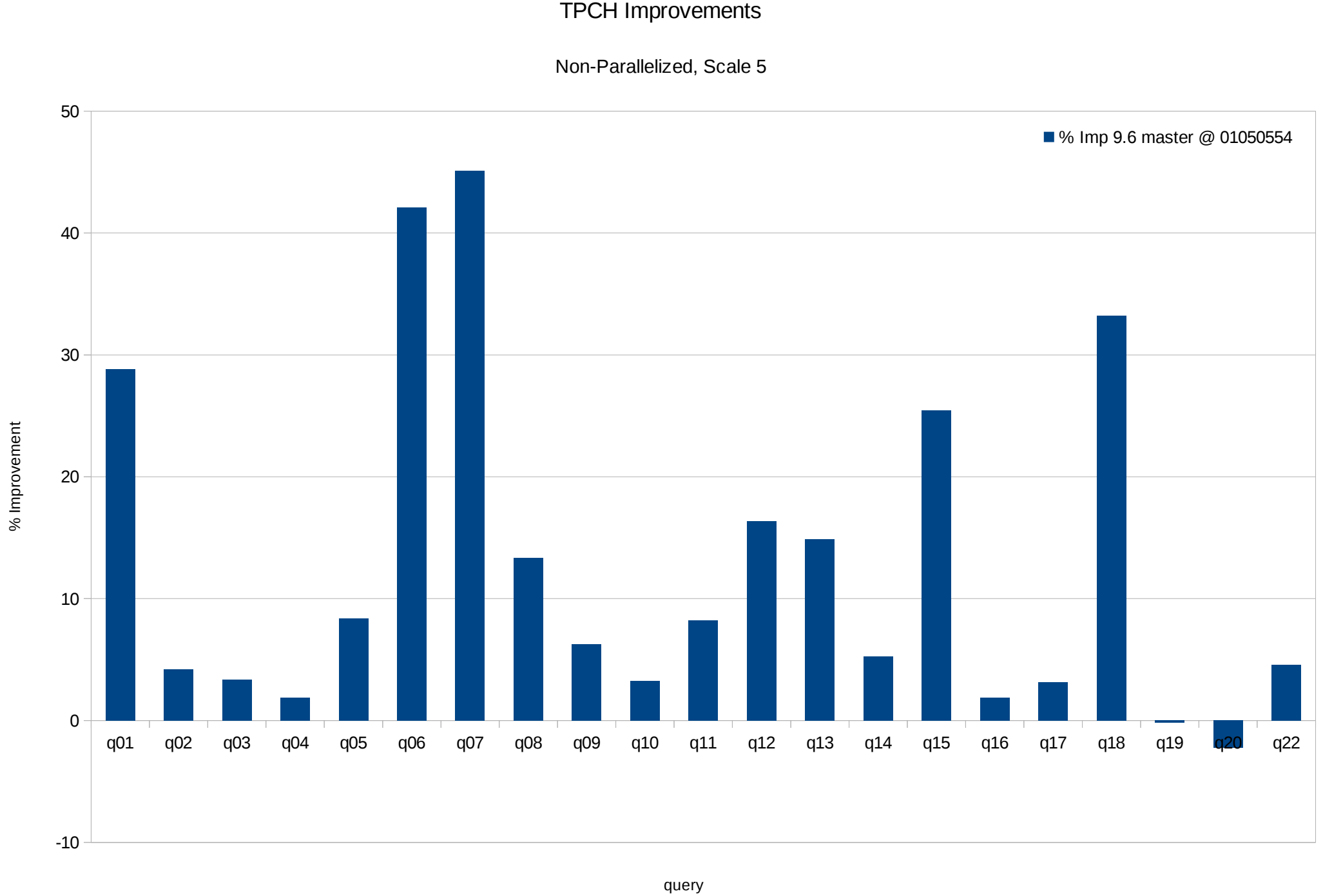


```
SELECT
    l_returnflag,
    l_linestatus,
    sum(l_quantity) as sum_qty,
    sum(l_extendedprice) as sum_base_price,
    sum(l_extendedprice * (1 - l_discount)) as sum_disc_price,
    sum(l_extendedprice * (1 - l_discount) * (1 + l_tax)) as sum_charge,
    avg(l_quantity) as avg_qty,
    avg(l_extendedprice) 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;
```

```
Sort  (cost=2153027.37..2153027.39 rows=6 width=68)
      (actual time=19742.591..19742.591 rows=4 loops=1)
 Sort Key: l_returnflag, l_linestatus
 Sort Method: quicksort  Memory: 25kB
 ->  HashAggregate  (cost=2153027.19..2153027.29 rows=6 width=68)
                    (actual time=19742.557..19742.559 rows=4 loops=1)
       Group Key: l_returnflag, l_linestatus
       ->  Seq Scan on lineitem  (cost= rows=29727516 width=36)
                                 (actual time=0.012..4514.186 rows=29713242 loops=1)
             Filter: (l_shipdate <= '1998-09-18 00:00:00'::timestamp without time zone)
             Rows Removed by Filter: 286553
Planning time: 0.326 ms
Execution time: 19742.705 ms
```


TPCH Timing
Non-Parallized, Scale 5
25000
20000
15000
10000
5000
0
timing (ms)
9.6
master @ 010505546a
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22
query

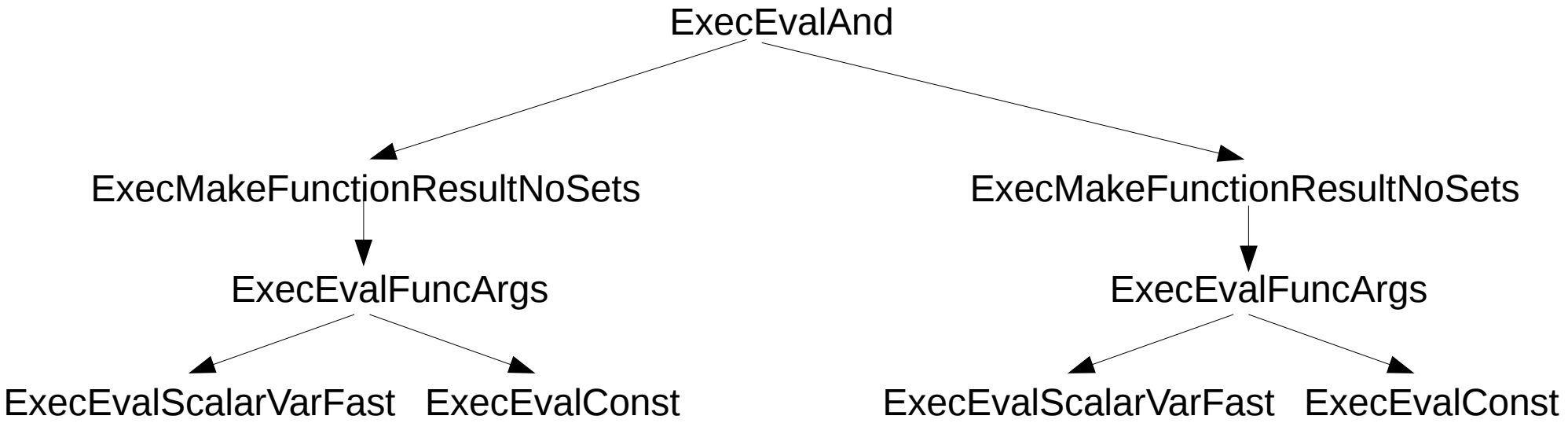
citusdata


TPCH Improvements
Non-Parallelized, Scale 5
% Imp 9.6 master @ 01050554
% Improvement
query
50
40
30
20
10
0
-10
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22

citusdata

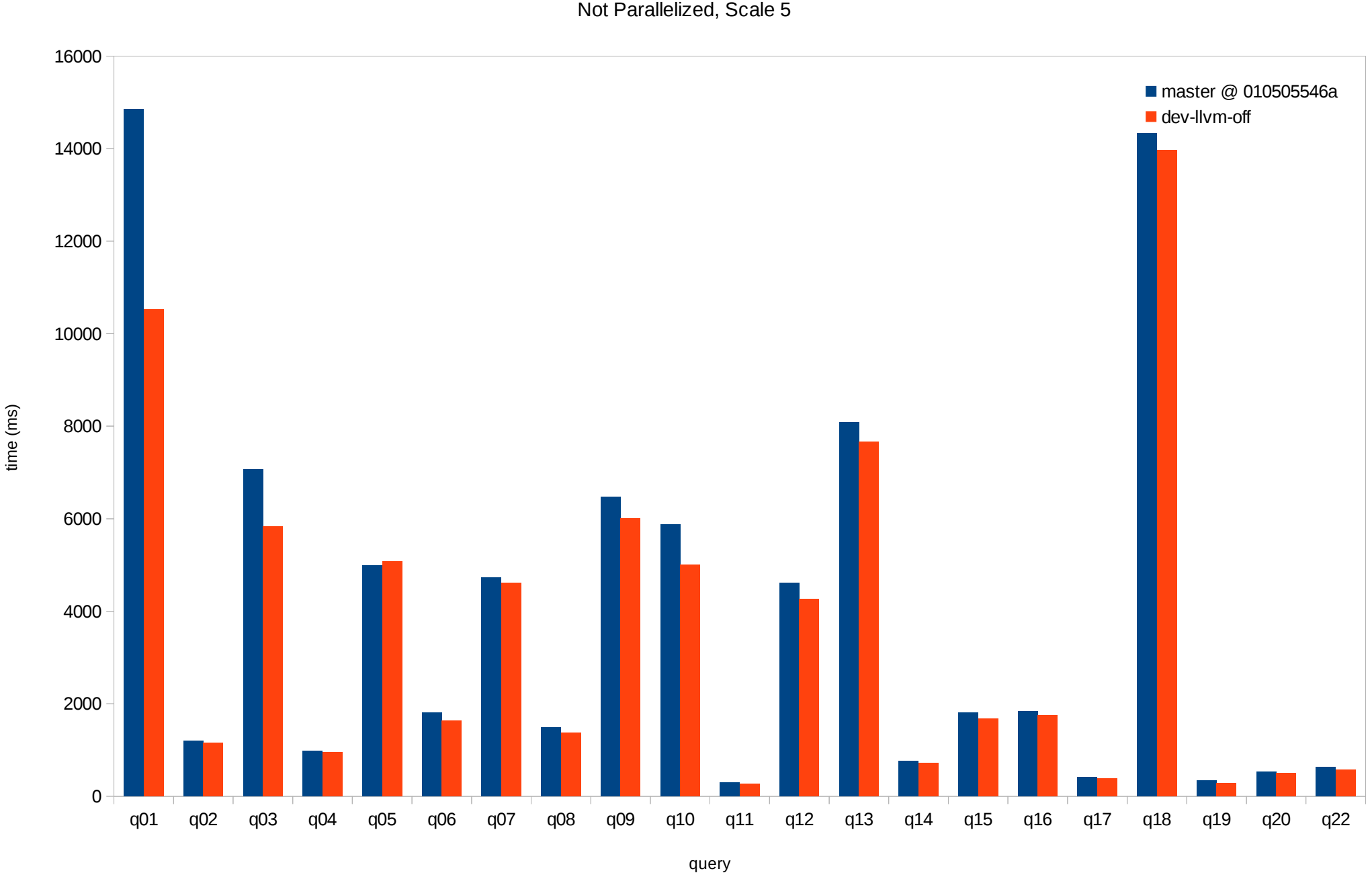
Expression Evaluation

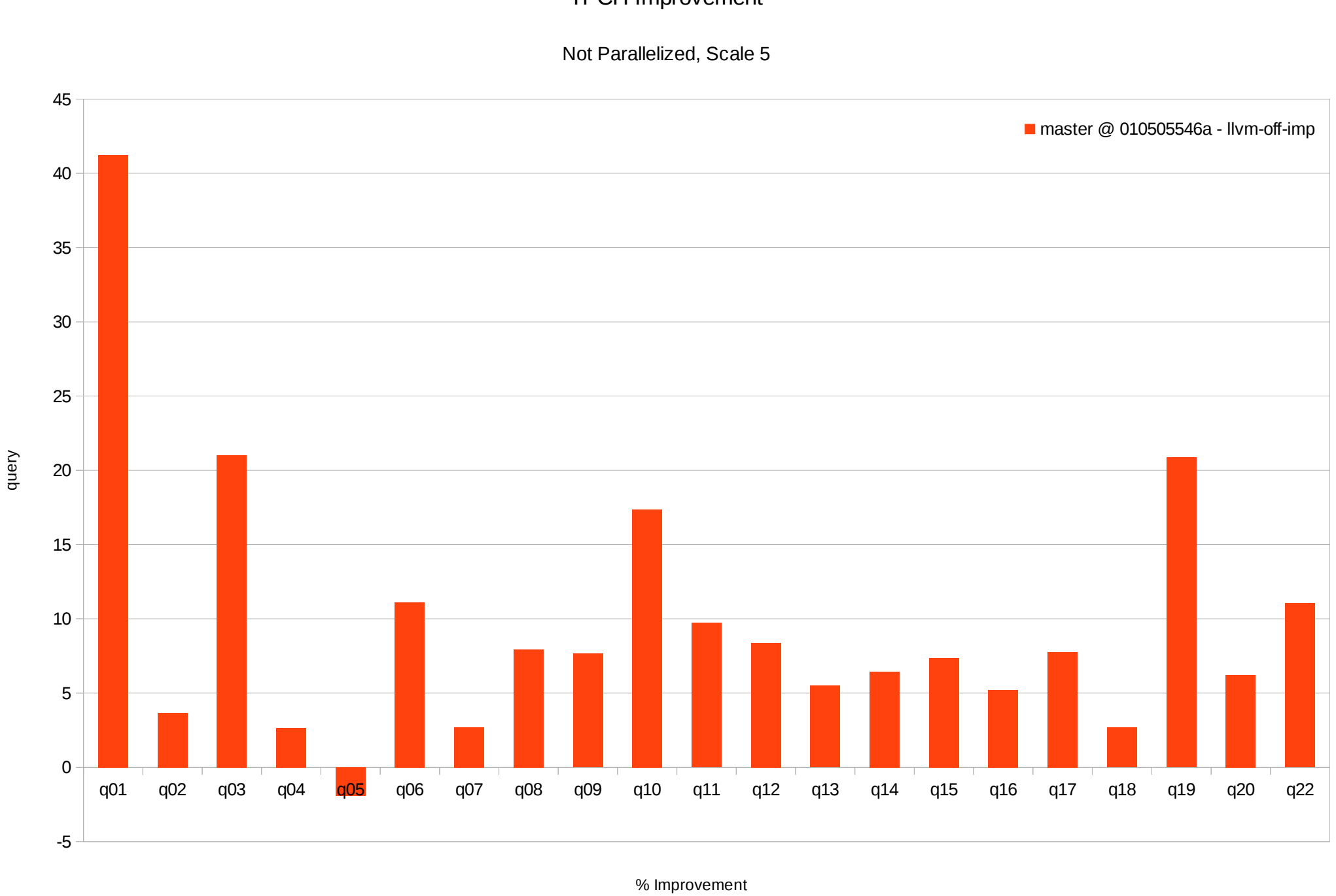
- SELECT list, of, columns, a + b as expr
- WHERE clauses
- GROUP BY clauses
- ...


ExecEvalAnd
ExecMakeFunctionResultNoSets
ExecMakeFunctionResultNoSets
ExecEvalFuncArgs
ExecEvalFuncArgs
ExecEvalScalarVarFast
ExecEvalConst
ExecEvalScalarVarFast
ExecEvalConst

Expression Evaluation Methodology

- Old:
  - Tree Walk
  - Callback Based
  - Recursive
- New
  - “Byte code” generated internally
  - No / Fewer Callbacks
  - “Direct Threaded”
  - Non-Recursive

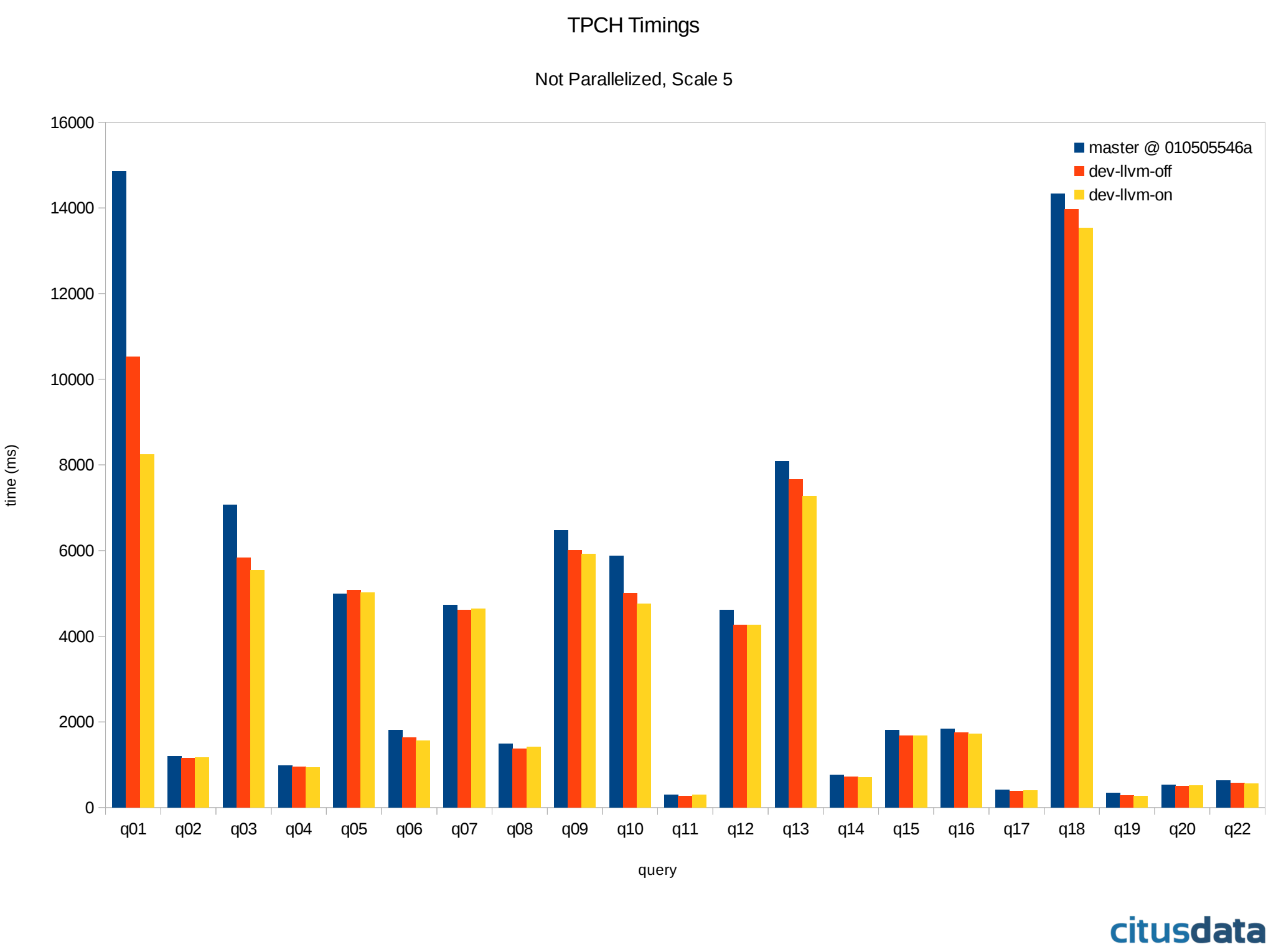

TPCH Timings
Not Parallelized, Scale 5
master @ 010505546a
dev-llvm-off
time (ms)
query
0
2000
4000
6000
8000
10000
12000
14000
16000
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22
citusdata


TPCH Improvement
Not Parallelized, Scale 5
master @ 010505546a - llvm-off-imp
45
40
35
30
25
20
15
10
5
0
-5
query
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22
% Improvement

citusdata

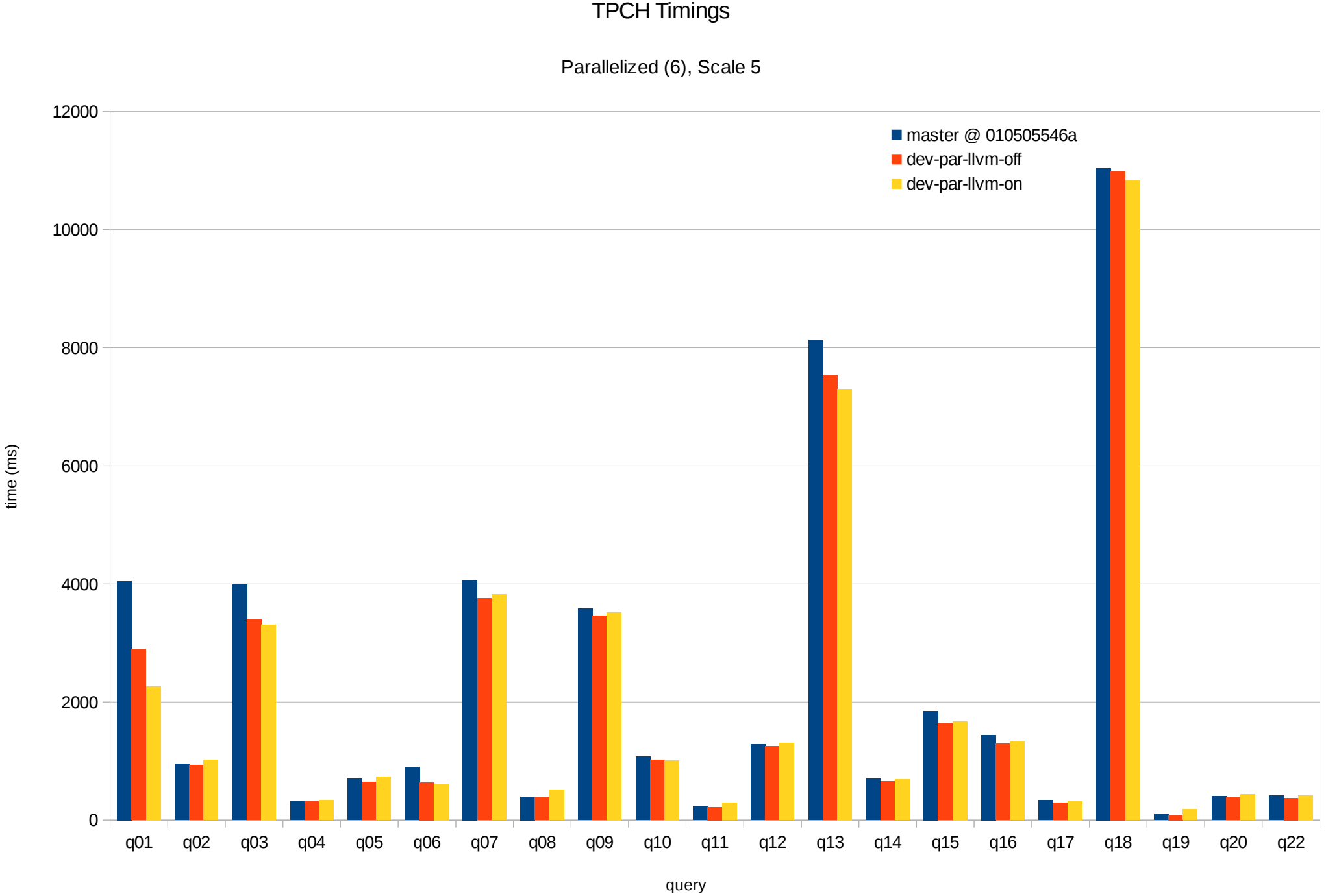
Just In Time Compilation

- Interpretation has a lot of “jumps”
- Interpretation calls a lot of “unknown functions”
- Native Code doesn’t have those Issues
- Only do so when beneficial
  - Generating a native function is expensive (~0.5-5ms)
- Can be used in a lot of places

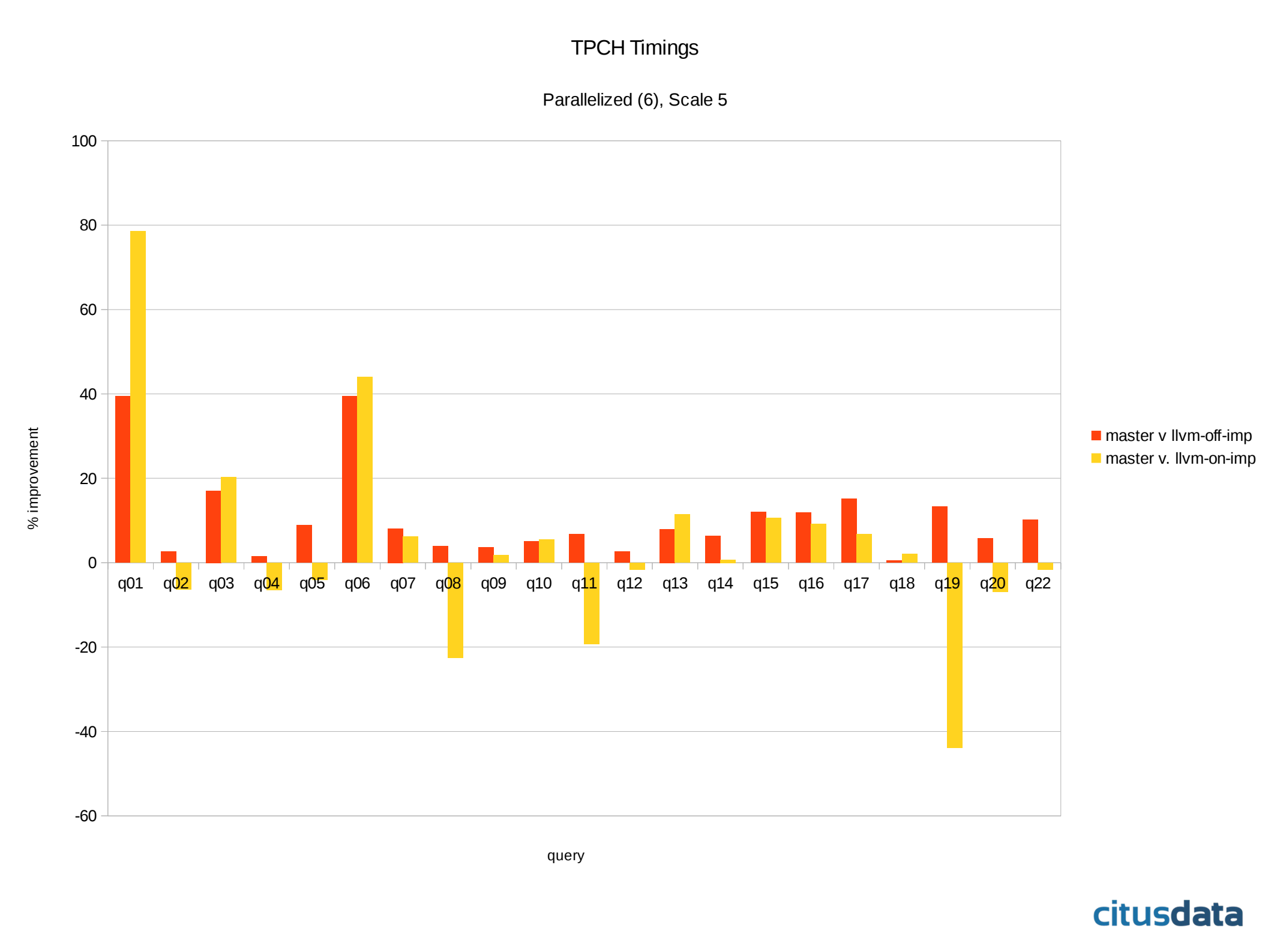

TPCH Timings
Not Parallelized, Scale 5
master @ 010505546a
dev-llvm-off
dev-llvm-on
time (ms)
16000
14000
12000
10000
8000
6000
4000
2000
0
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22
query
citusdata

TPCH Improvement
Not Parallelized, Scale 5
master v. llvm-off-imp
master v. llvm-on-imp
90
80
70
60
50
40
30
20
10
0
-10
query
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22
% Improvement

citusdata


TPCH Timings
Parallelized (6), Scale 5
master @ 010505546a
dev-par-llvm-off
dev-par-llvm-on
time (ms)
query
0
2000
4000
6000
8000
10000
12000
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22

citusdata


TPCH Timings
Parallelized (6), Scale 5
% improvement
100
80
60
40
20
0
-20
-40
-60
q01
q02
q03
q04
q05
q06
q07
q08
q09
q10
q11
q12
q13
q14
q15
q16
q17
q18
q19
q20
q22
query
master v llvm-off-imp
master v. llvm-on-imp
citusdata

Tuple Deforming

- On-Disk → Memory Representation
- Expensive, don’t know columns ahead
- Improving Code: 1 Month, up to 3%
- JIT: ~1 Month, up to 80 %

Better JIT

- Inline functions
  - WHERE (a + b) < c
    - float8pl / int4pl / *pl
    - float8lt / int4lt / *lt
  - AVG(a + b)
    - int8pl
    - int8_avg_accum
    - numeric_poly_avg
  - Function calls are expensive, content cheap to execute
- Better Code

Batch Tuple Processing #2

- Other Bottlenecks Are Gone
- Experiments show 0 - 40% in TPC-H queries

Speeding up Query Execution

Andres Freund
PostgreSQL Developer & Committer
Citus Data – citusdata.com - @citusdata

http://anarazel.de/talks/pconf-us-2017-03-29/jit.pdf