ORACLE

Hash Join in MySQL 8.0

Øystein Grøvlen
Senior Principal Engineer
Oracle
February, 2022

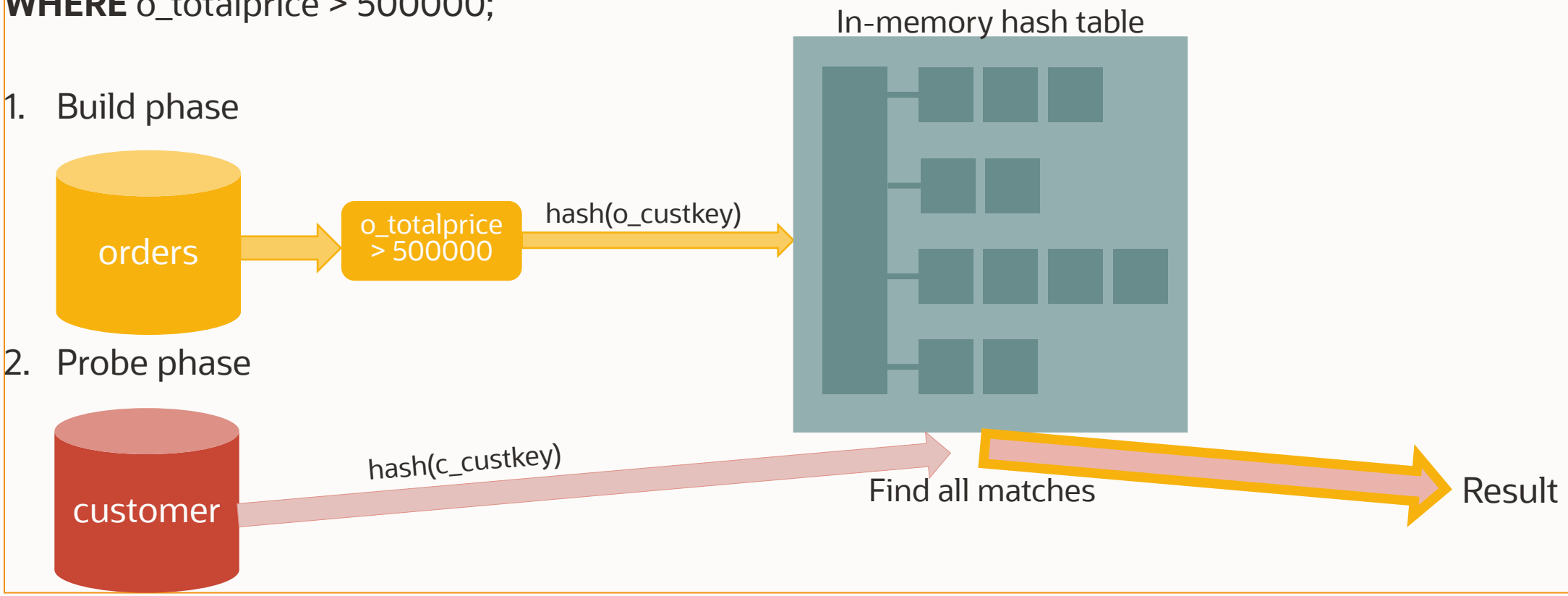
Agenda

1. What is hash join?
2. Hash join in MySQL 8.0
3. When to use hash join
4. How to use hash join

Hash Join

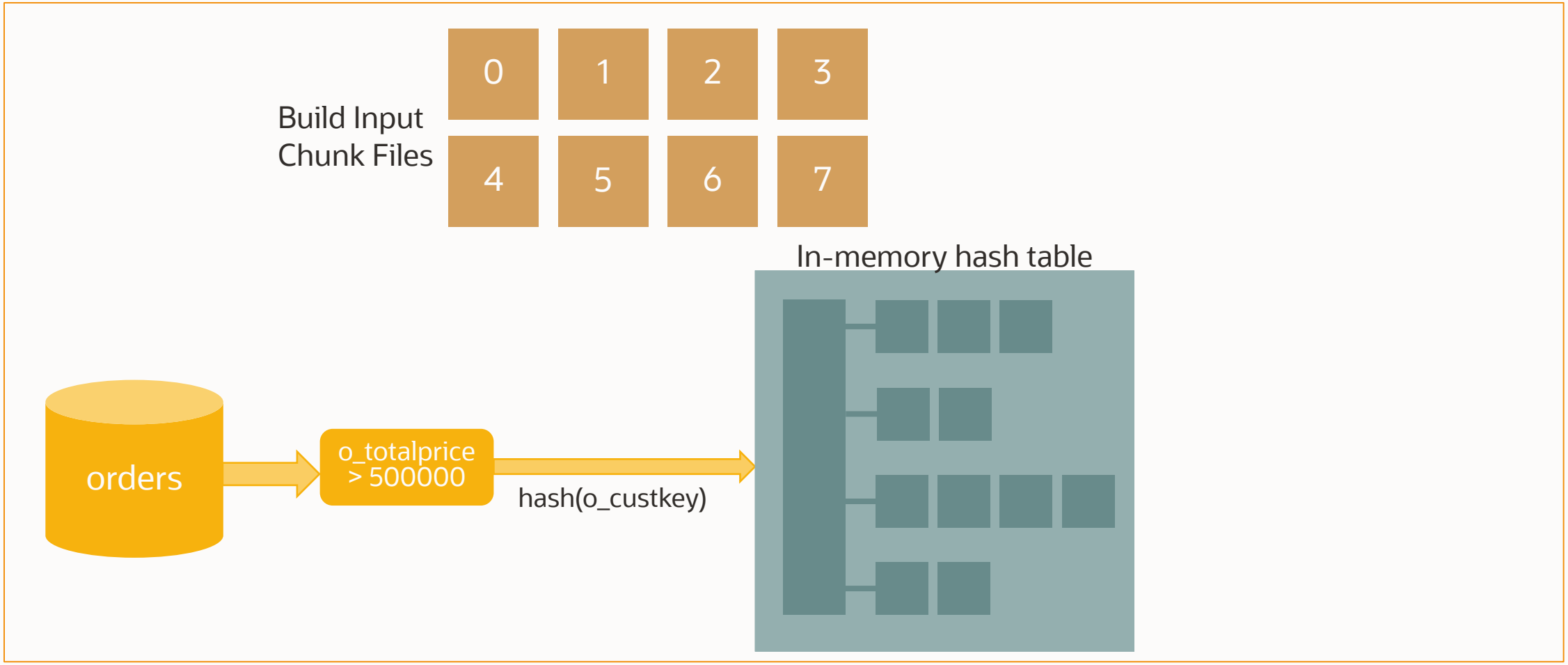
Classic Hash Join

SELECT c_name, c_address, o_orderdate, o_totalprice
FROM orders **JOIN** customer **ON** o_custkey = c_custkey
WHERE o_totalprice > 500000;



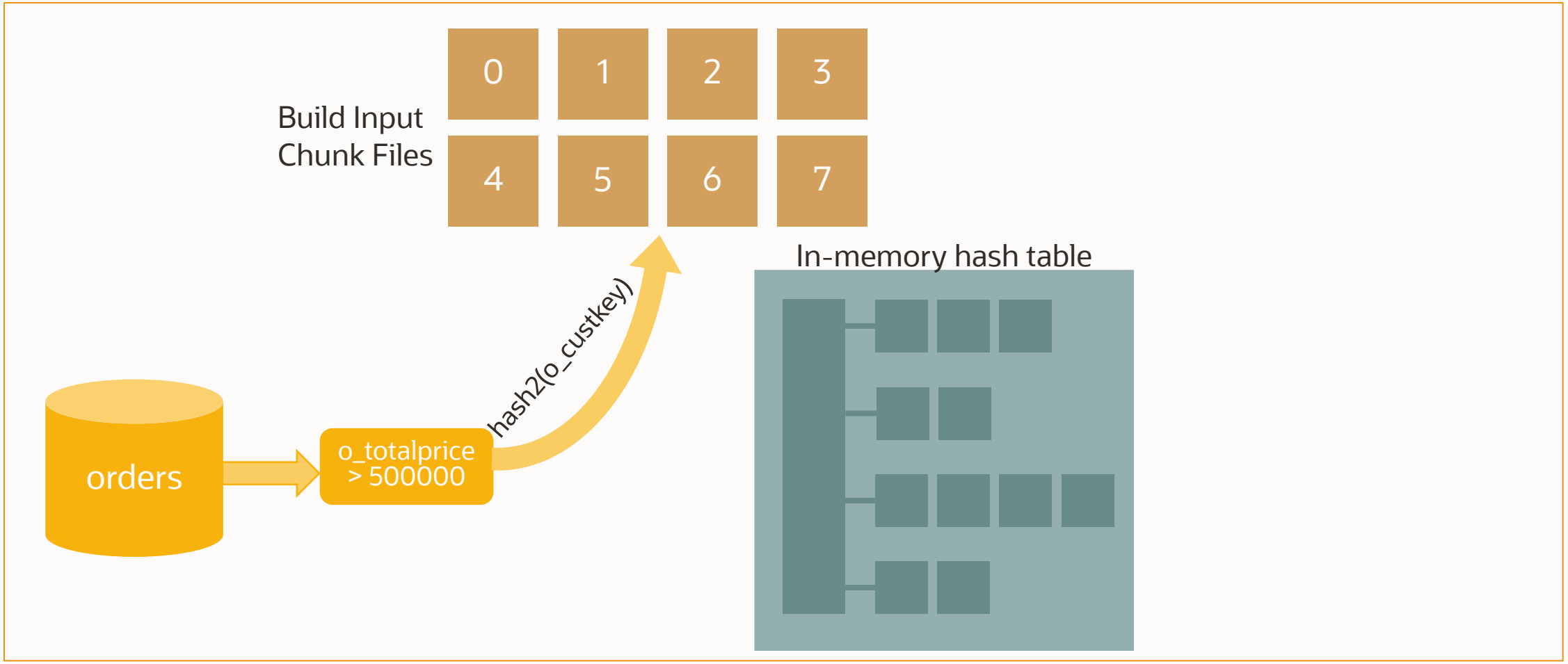
Hybrid Hash Join

Build phase



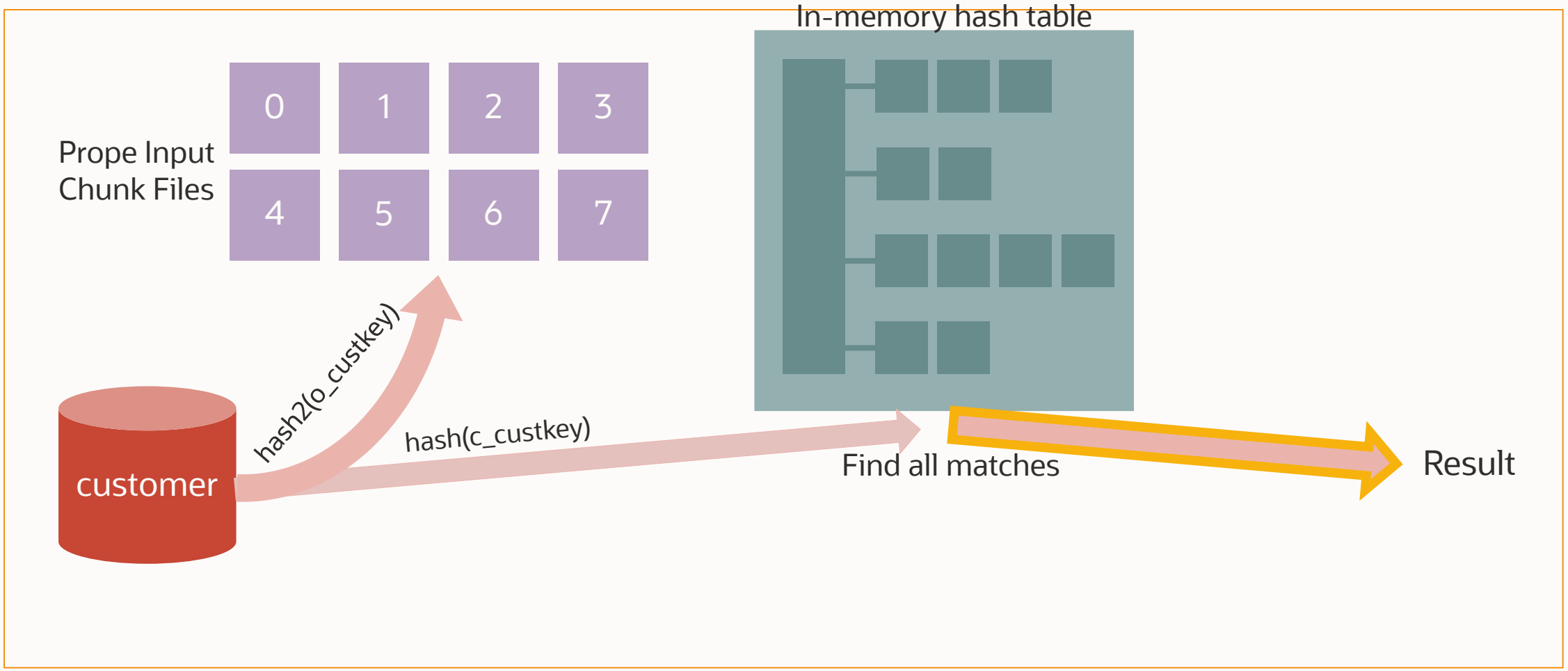
Hybrid Hash Join

Build phase



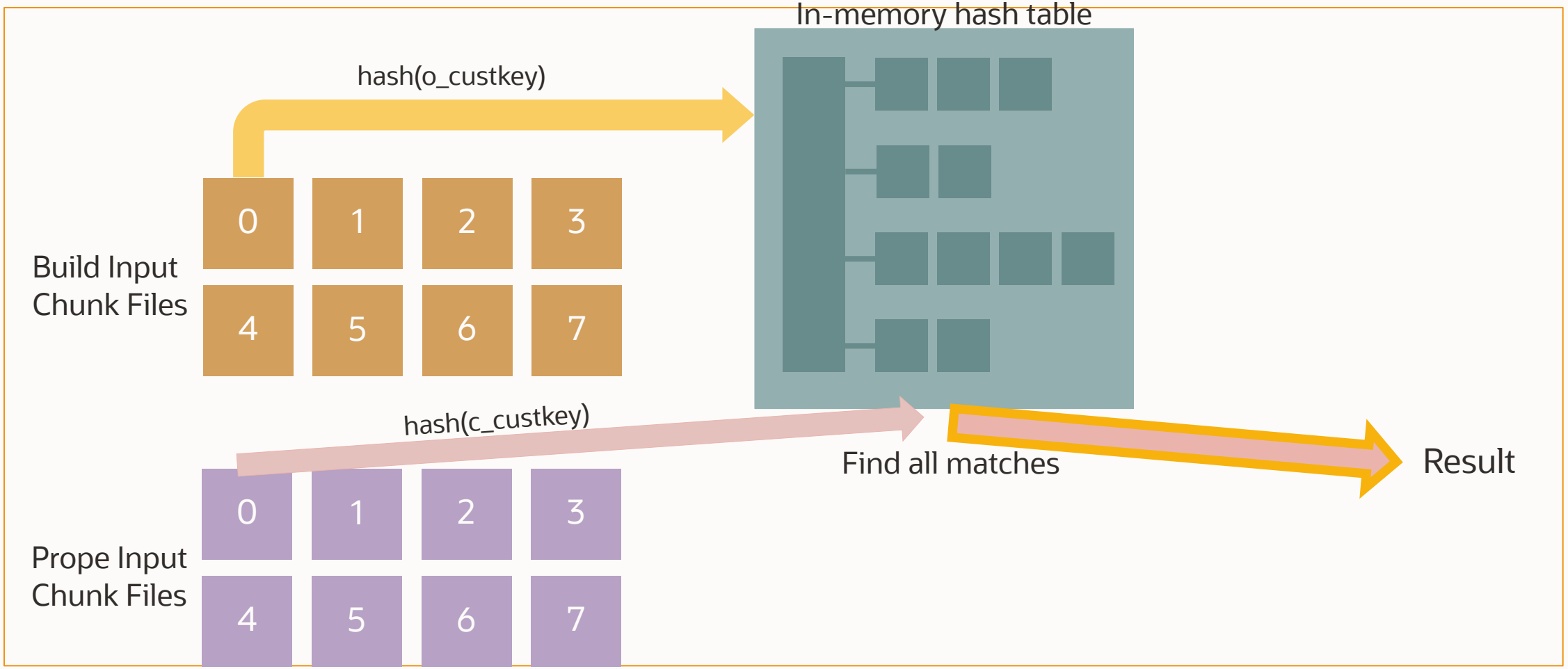
Hybrid Hash Join

Probe phase



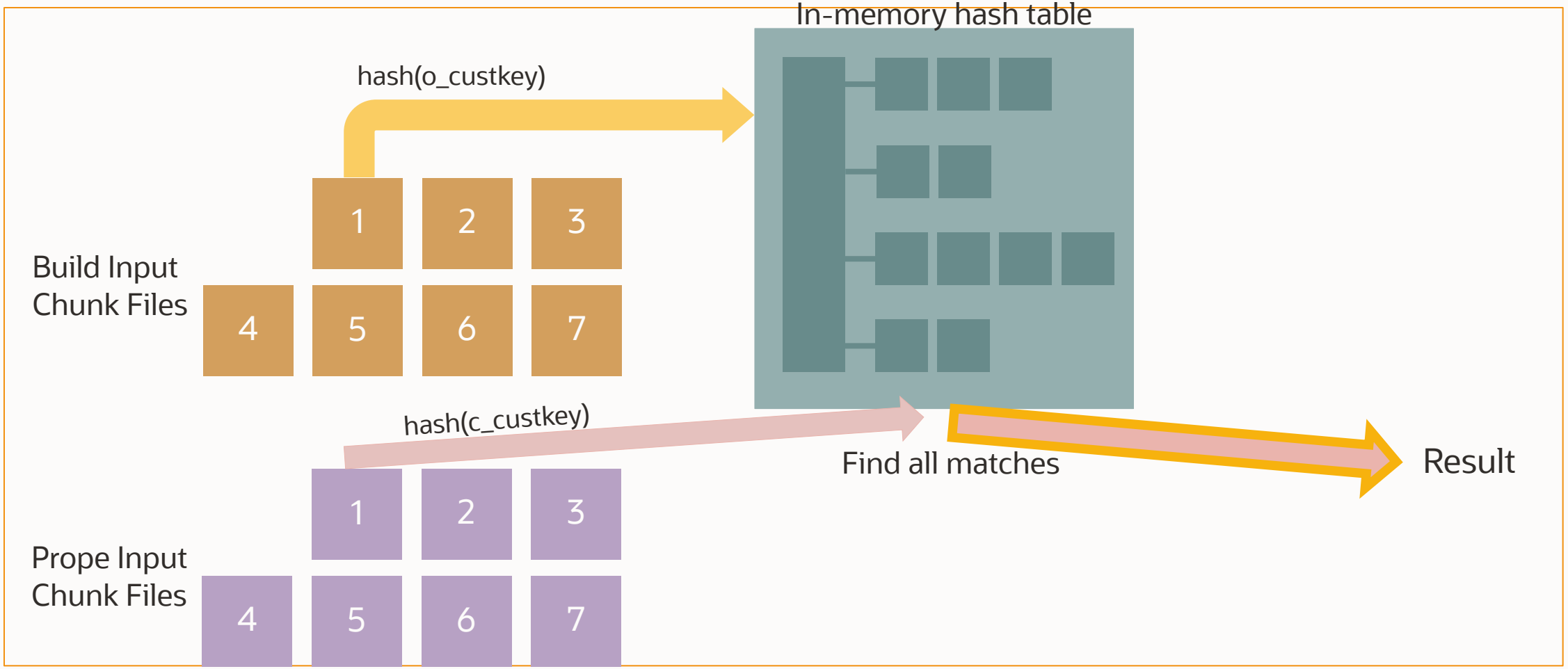
Hybrid Hash Join

Repeat for each pair of chunk files



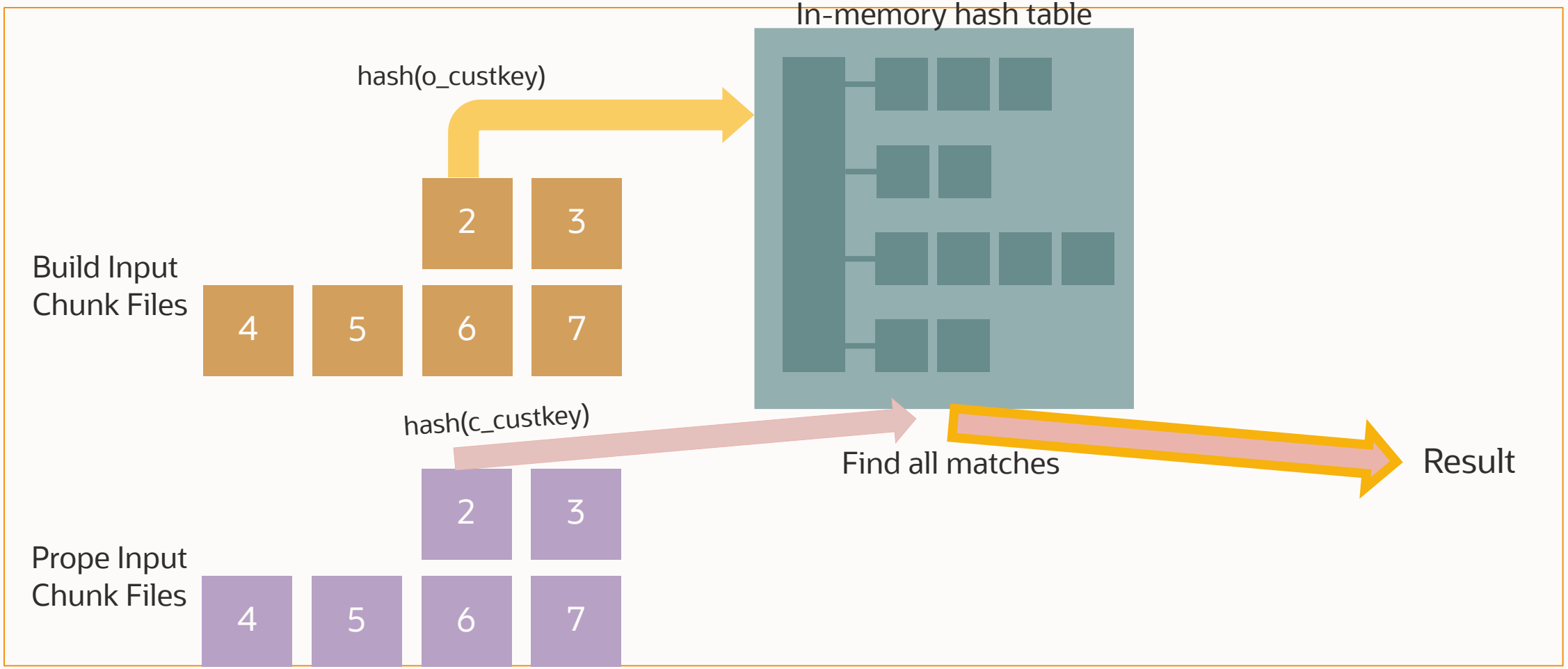
Hybrid Hash Join

Repeat for all pairs of chunk files



Hybrid Hash Join

Repeat for all pairs of chunk files



Hash Join in MySQL 8.0

Join Execution in MySQL 5.7

Supports only Nested-Loops Join

- For each row in left input, find all matching rows in right table

Efficient when

- An index can be used to find the matching rows
- A small subset of the rows will be read
- There are relatively few matches per row
- Most rows can be accessed in memory

Join Execution in MySQL 8.0

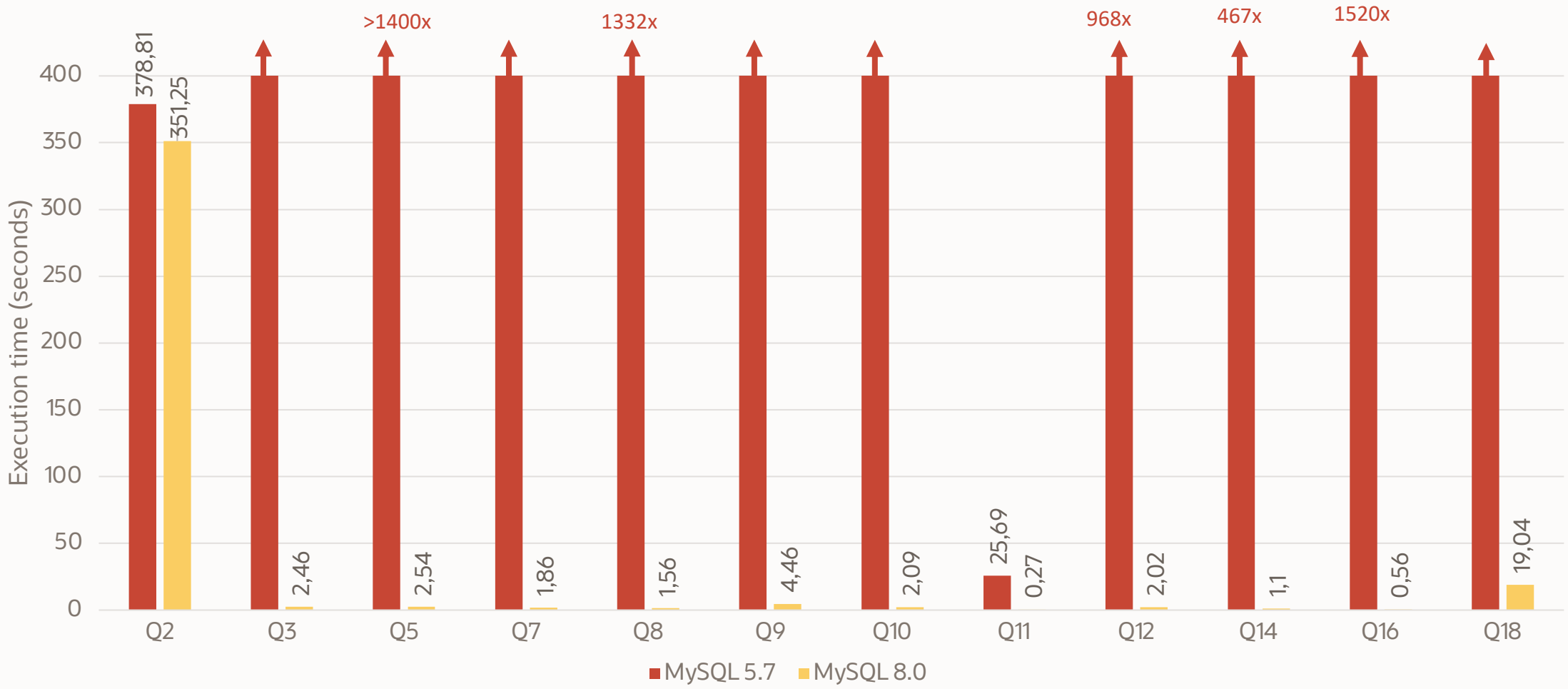
Supports both Nested-Loops Join and Hash Join

Hash Join

- Supports both In-memory and Hybrid Hash Join
- In-memory
  - Memory usage determined by **join_buffer_size** .
- Hybrid Hash Join
  - Max number of chunk files per input: 128
- Automatically chosen where Block Nested Loops Join (BNL) was earlier selected
  - Usually when no indexes are available
  - NO_INDEX hint can be used to force hash join to be used

Hash Join versus Nested-Loops Join (TPC-H)

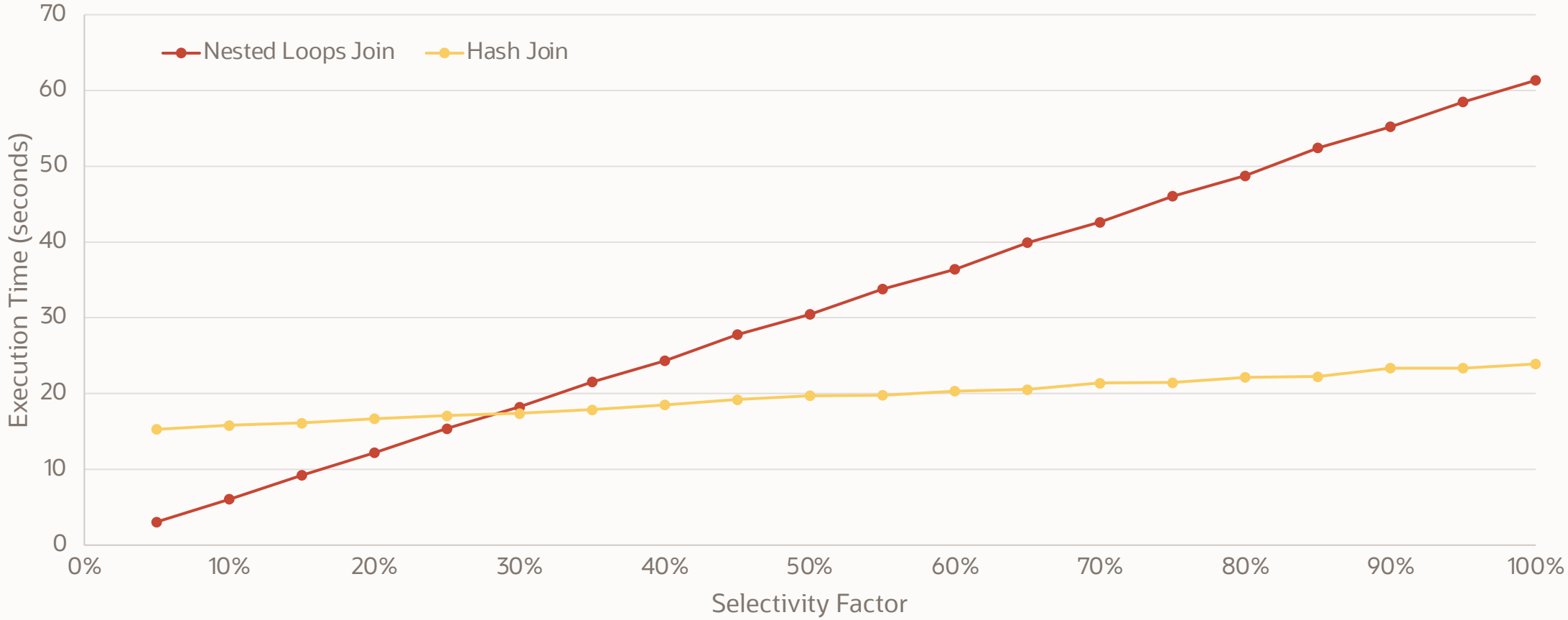
Hash join is much faster than Nested-Loops Join when there are no indexes



Hash Join versus Nested-Loops Join

Hash Join can also be faster than Indexed Nested-Loops Join

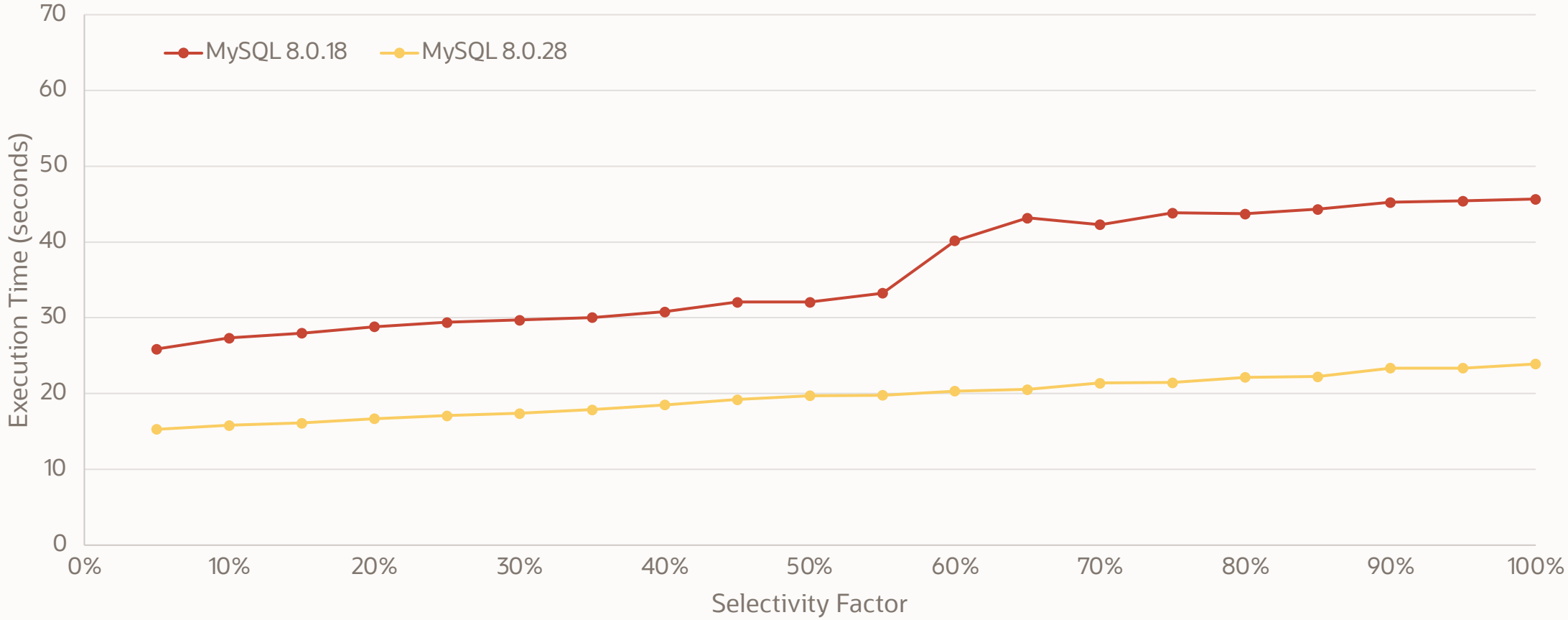
SELECT COUNT(*) FROM orders JOIN lineitem ON l_orderkey = o_orderkey WHERE o_custkey <= 1500000 * *selectivity*



Improved Performance in Later Versions of MySQL 8.0

MySQL 8.0.18 vs MySQL 8.0.28

SELECT COUNT(*) FROM orders JOIN lineitem ON l_orderkey = o_orderkey WHERE o_custkey <= 1500000 * *selectivity*



When to Use Hash Join

No Applicable Index

Hash Join will automatically be selected in 8.0

```
SELECT s_name, c_name
FROM supplier JOIN customer ON s_phone = c_phone;
```

MySQL 5.7

```
mysql [localhost:5736] {msandbox} (dbt3_sf10) > EXPLAIN SELECT s_name, c_name FROM supplier JOIN customer ON s_phone = c_phone;
+----+-------------+----------+------------+------+---------------+------+---------+------+---------+----------+----------------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows    | filtered | Extra                                              |
+----+-------------+----------+------------+------+---------------+------+---------+------+---------+----------+----------------------------------------------------+
|  1 | SIMPLE      | supplier | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  100000 |   100.00 | NULL                                               |
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 1500000 |    10.00 | Using where; Using join buffer (Block Nested Loop) |
+----+-------------+----------+------------+------+---------------+------+---------+------+---------+----------+----------------------------------------------------+
```

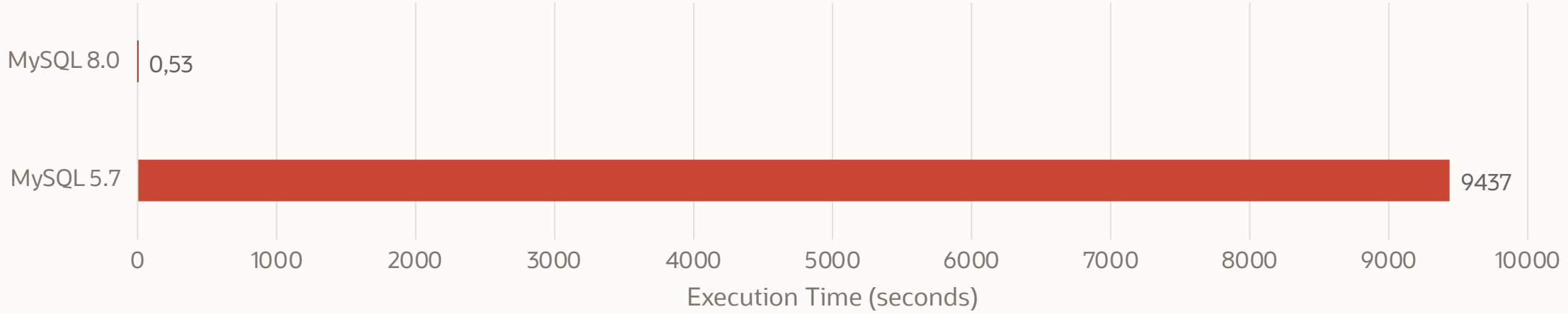
MySQL 8.0

```
mysql [localhost:8028] {msandbox} (dbt3_sf10) > EXPLAIN SELECT s_name, c_name FROM supplier JOIN customer ON s_phone = c_phone;
+----+-------------+----------+------------+------+---------------+------+---------+------+---------+----------+-------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows    | filtered | Extra                                     |
+----+-------------+----------+------------+------+---------------+------+---------+------+---------+----------+-------------------------------------------+
|  1 | SIMPLE      | supplier | NULL       | ALL  | NULL          | NULL | NULL    | NULL |  100000 |   100.00 | NULL                                      |
|  1 | SIMPLE      | customer | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 1500000 |    10.00 | Using where; Using join buffer (hash join) |
+----+-------------+----------+------------+------+---------------+------+---------+------+---------+----------+-------------------------------------------+
```

No Applicable Index

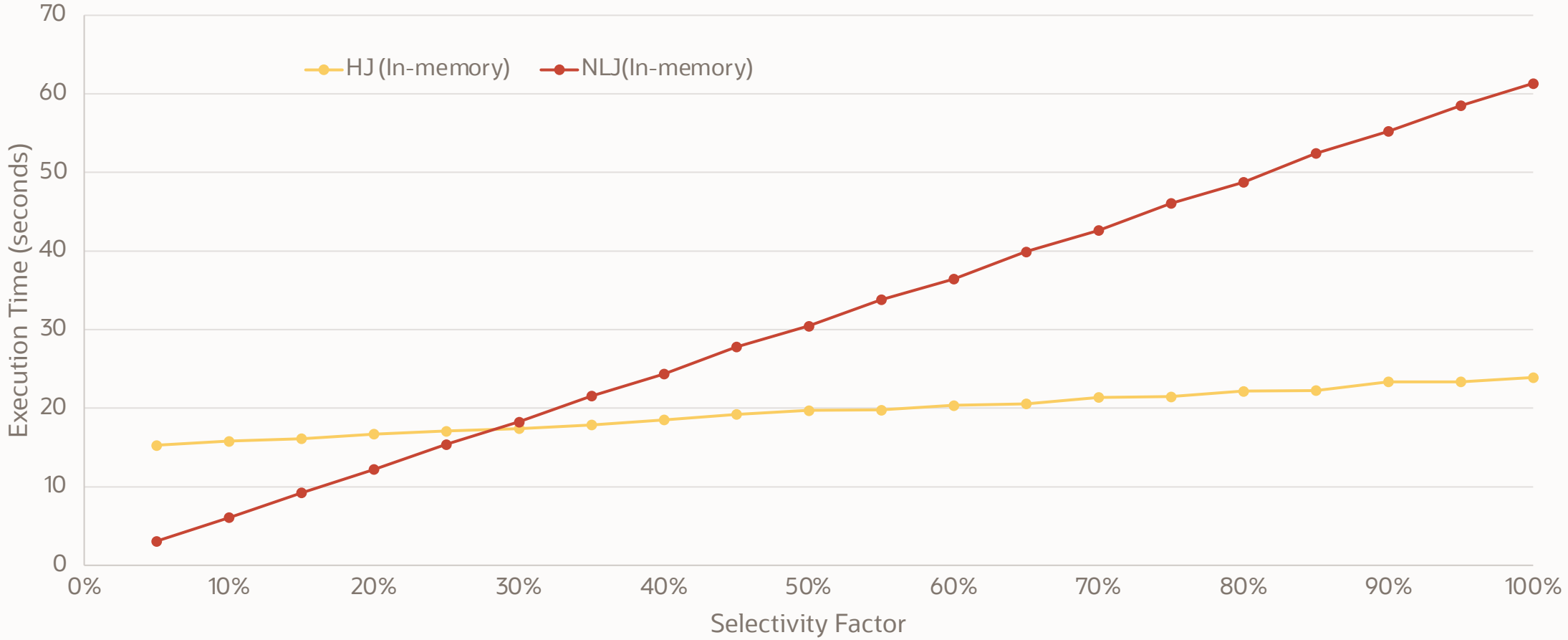
Hash Join will automatically be selected in 8.0

```
SELECT s_name, c_name
FROM supplier JOIN customer ON s_phone = c_phone;
```



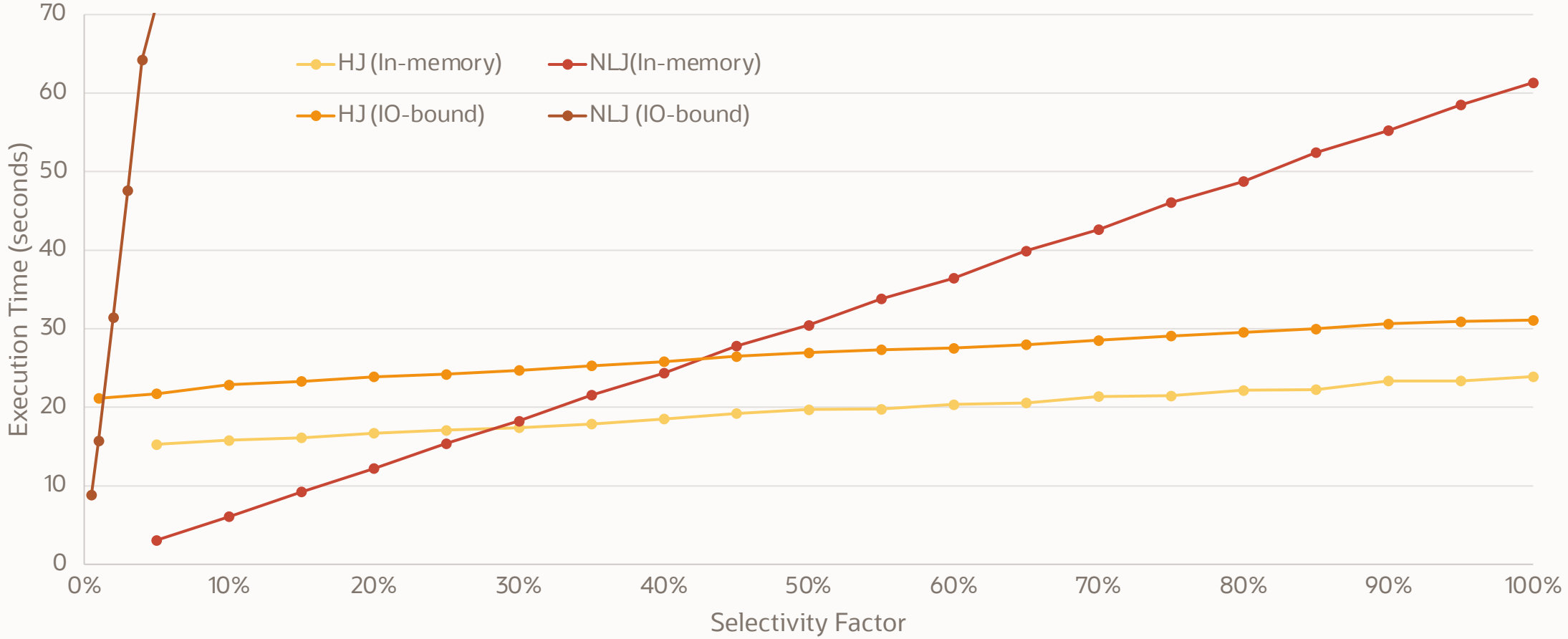
IO-bound queries

SELECT COUNT(*) FROM orders JOIN lineitem ON l_orderkey = o_orderkey WHERE o_custkey <= 1500000 * *selectivity*



IO-bound queries

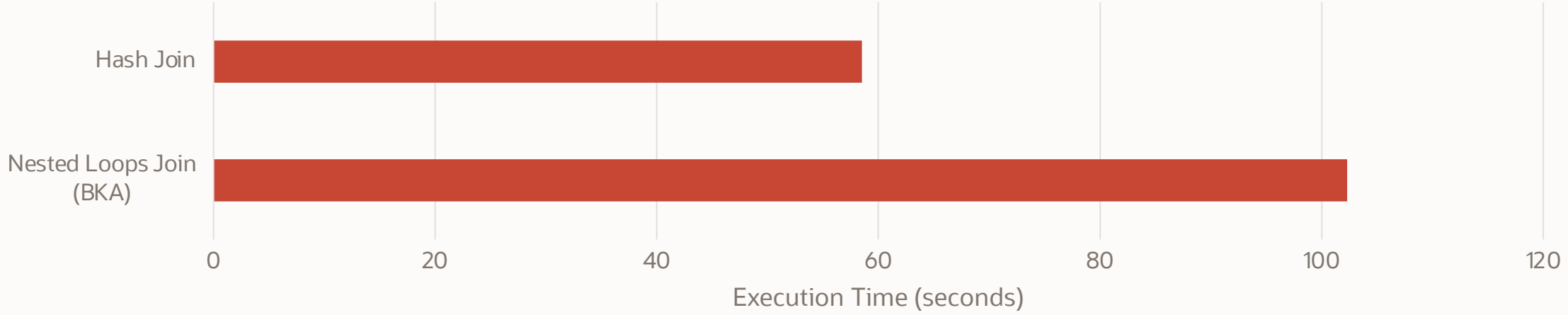
SELECT COUNT(*) FROM orders JOIN lineitem ON l_orderkey = o_orderkey WHERE o_custkey <= 1500000 * *selectivity*



Large Subset of Table

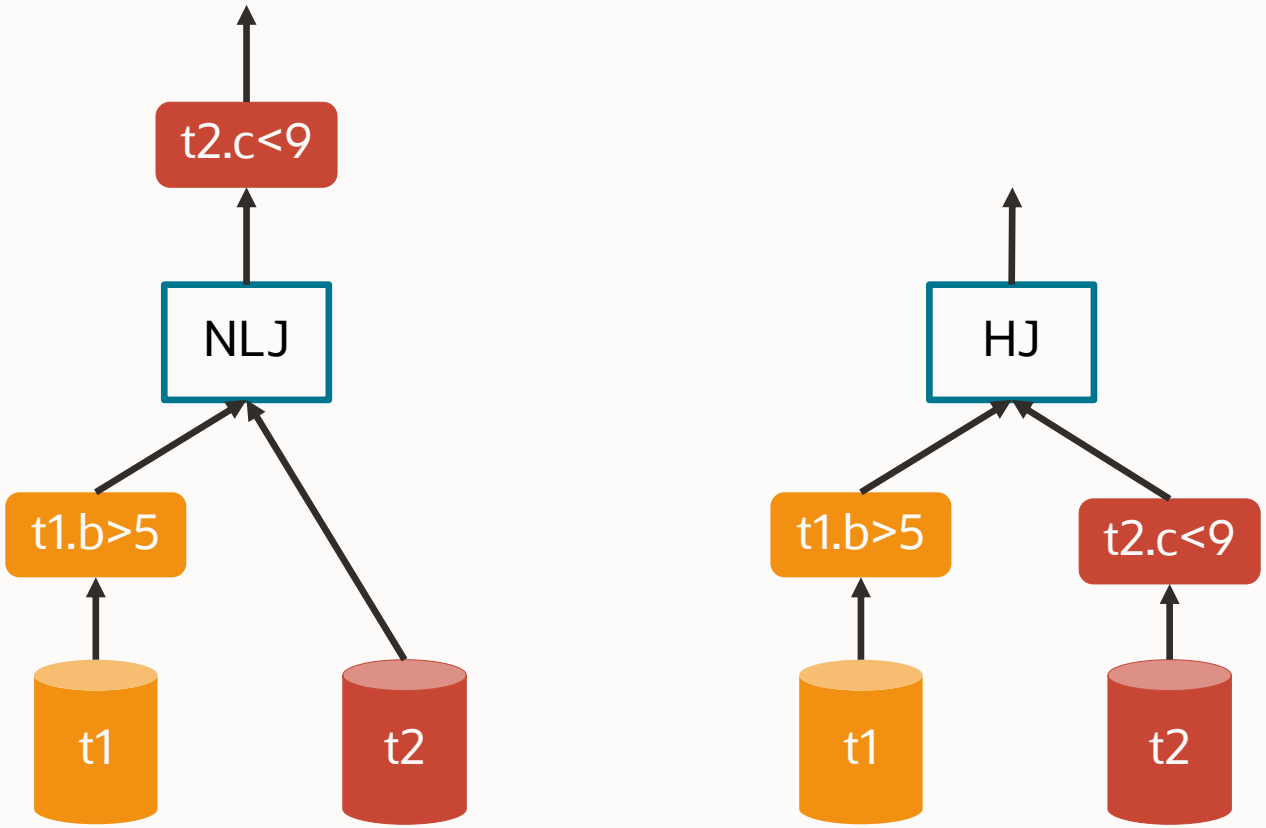
TPC-H Q13 (Customer Distribution Query)

```
SELECT c_count, COUNT(*) AS custdist
FROM (
    SELECT c_custkey, COUNT(o_orderkey) AS c_count
    FROM customer LEFT OUTER JOIN orders
      ON c_custkey = o_custkey AND o_comment NOT LIKE '%express%requests%'
    GROUP BY c_custkey
  ) AS c_orders
GROUP BY c_count
ORDER BY custdist DESC, c_count DESC;
```



Selective Conditions on Multiple Tables

Hash join Enables Early Filtering

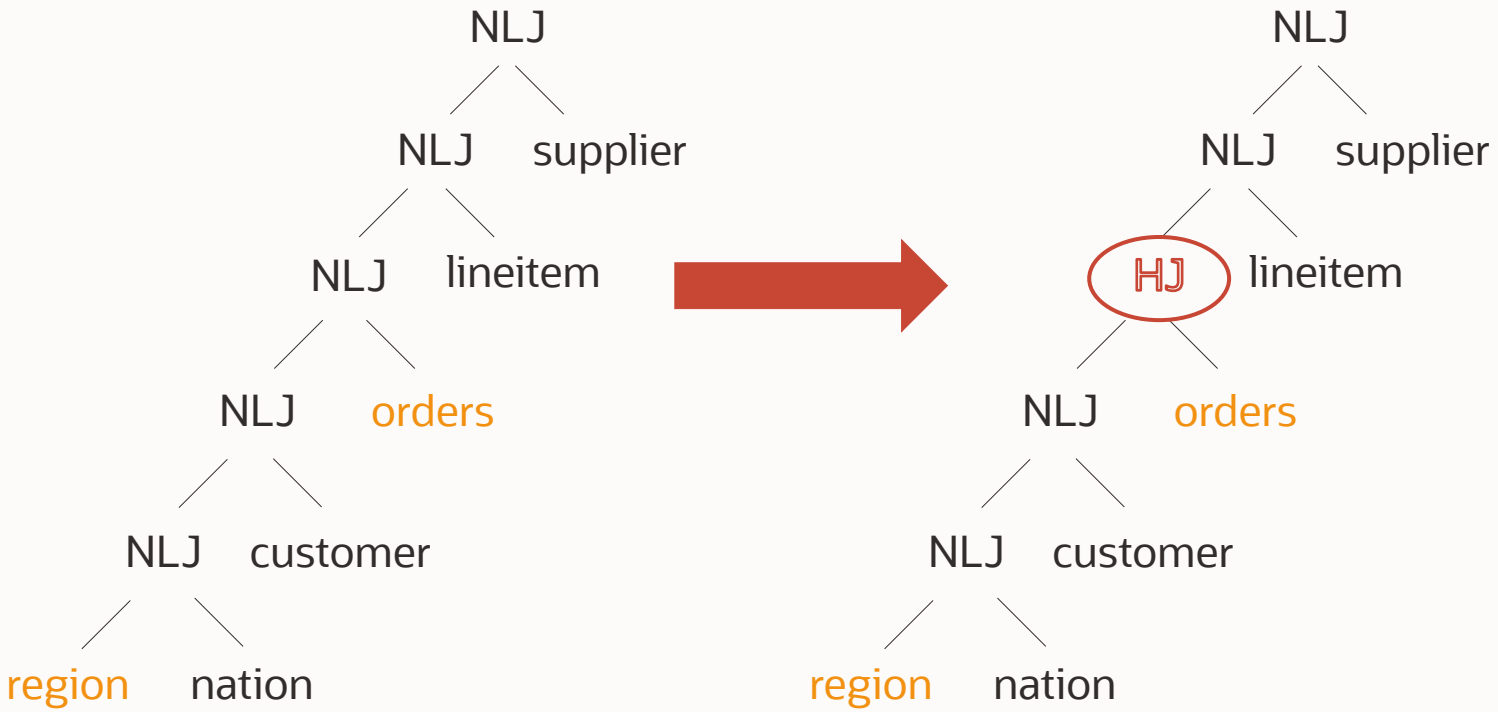


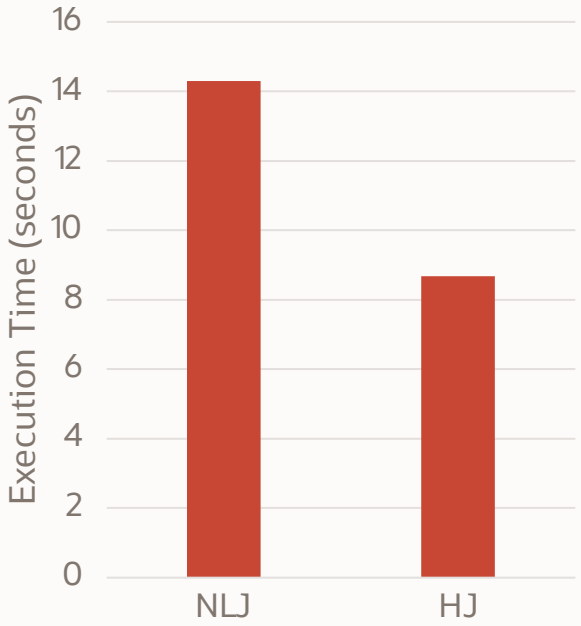

```
SELECT *
FROM t1 JOIN t2 ON t1.a = t2.a
WHERE t1.b > 5 AND t2.c < 9;
```

Selective Conditions on Multiple Tables

TPC-H Q5 (Local Supplier Volume Query)

```
SELECT n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
FROM customer JOIN orders ON c_custkey = o_custkey JOIN lineitem ON l_orderkey = o_orderkey
     JOIN supplier ON l_suppkey = s_suppkey AND c_nationkey = s_nationkey
     JOIN nation ON s_nationkey = n_nationkey JOIN region ON n_regionkey = r_regionkey
WHERE r_name = 'ASIA'
  AND o_orderdate >= '1995-01-01' AND o_orderdate < DATE_ADD('1995-01-01', INTERVAL '1' YEAR)
GROUP BY n_name ORDER BY revenue desc;
```





How to Use Hash Join

EXPLAIN Shows Whether Hash Join is Used

EXPLAIN FORMAT=TRADITIONAL:

```
+----+-------------+----------+------------+------+---------------+------+---------+------+----------+----------+----------------------------------------------+
| id | select_type | table    | partitions | type | possible_keys | key  | key_len | ref  | rows     | filtered | Extra                                        |
+----+-------------+----------+------------+------+---------------+------+---------+------+----------+----------+----------------------------------------------+
|  1 | SIMPLE      | orders   | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 15000000 |     5.02 | Using where                                  |
|  1 | SIMPLE      | lineitem | NULL       | ALL  | NULL          | NULL | NULL    | NULL | 59986052 |     0.00 | Using where; Using join buffer (hash join) |
+----+-------------+----------+------------+------+---------------+------+---------+------+----------+----------+----------------------------------------------+
```

EXPLAIN FORMAT=TREE:

```
-> Aggregate: count(0)  (cost=4513965784213.93 rows=3009309)
  -> Inner hash join (lineitem.l_orderkey = orders.o_orderkey)  (cost=4513965483283.02 rows=3009309)
      -> Table scan on lineitem  (cost=0.19 rows=59986052)
      -> Hash
          -> Filter: (orders.o_custkey <= <cache>((1500000 * 0.05)))  (cost=1530473.77 rows=752502)
              -> Table scan on orders  (cost=1530473.77 rows=15000000)
```

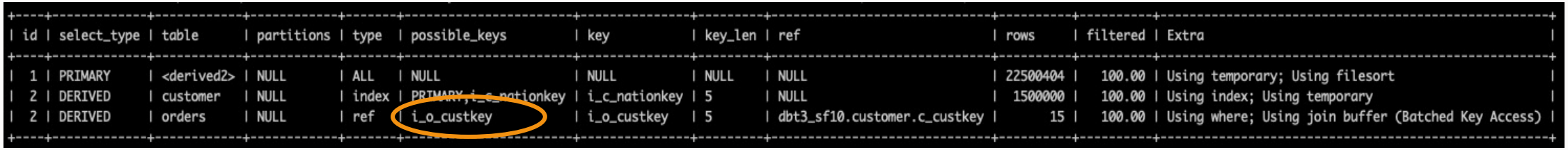
Probe Input

Build Input

Disable Indexes to Force Hash Join

TPC-H Q13 (Customer Distribution Query)

```
SELECT c_count, COUNT(*) AS custdist
FROM (
    SELECT c_custkey, COUNT(o_orderkey) AS c_count
    FROM customer LEFT OUTER JOIN orders
      ON c_custkey = o_custkey AND o_comment NOT LIKE '%express%requests%'
    GROUP BY c_custkey
  ) AS c_orders
GROUP BY c_count
ORDER BY custdist DESC, c_count DESC;
```

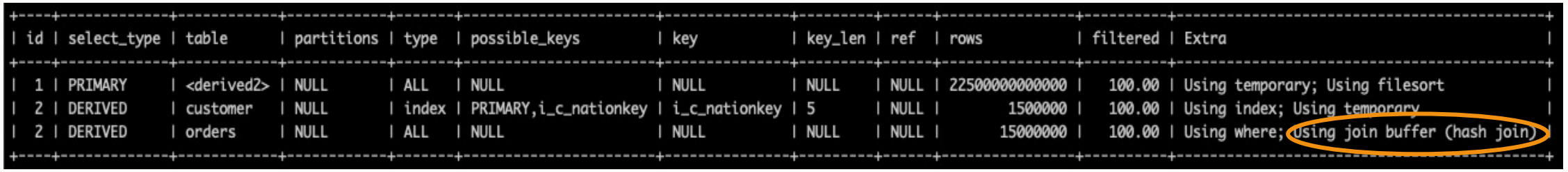


id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	22500404	100.00	Using temporary; Using filesort
2	DERIVED	customer	NULL	index	PRIMARY,i_c_nationkey	i_c_nationkey	5	NULL	1500000	100.00	Using index; Using temporary
2	DERIVED	orders	NULL	ref	i_o_custkey	i_o_custkey	5	dbt3_sf10.customer.c_custkey	15	100.00	Using where; Using join buffer (Batched Key Access)

Use NO_INDEX Hint to Force Hash Join

TPC-H Q13 (Customer Distribution Query)

```
SELECT c_count, COUNT(*) AS custdist
FROM (
    SELECT /*+ NO_INDEX(orders, i_o_custkey) */ c_custkey, COUNT(o_orderkey) AS c_count
    FROM customer LEFT OUTER JOIN orders
      ON c_custkey = o_custkey AND o_comment NOT LIKE '%express%requests%'
    GROUP BY c_custkey
  ) AS c_orders
GROUP BY c_count
ORDER BY custdist DESC, c_count DESC;
```



id	select_type	table	partitions	type	possible_keys	key	key_len	ref	rows	filtered	Extra
1	PRIMARY	<derived2>	NULL	ALL	NULL	NULL	NULL	NULL	22500000000000	100.00	Using temporary; Using filesort
2	DERIVED	customer	NULL	index	PRIMARY,i_c_nationkey	i_c_nationkey	5	NULL	1500000	100.00	Using index; Using temporary
2	DERIVED	orders	NULL	ALL	NULL	NULL	NULL	NULL	15000000	100.00	Using where; Using join buffer (hash join)

Use NO_INDEX Hint to Force Hash Join

TPC-H Q5 (Local Supplier Volume Query)

```
SELECT n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
FROM customer JOIN orders ON c_custkey = o_custkey JOIN lineitem ON l_orderkey = o_orderkey
     JOIN supplier ON l_suppkey = s_suppkey AND c_nationkey = s_nationkey
     JOIN nation ON s_nationkey = n_nationkey JOIN region ON n_regionkey = r_regionkey
WHERE r_name = 'ASIA'
  AND o_orderdate >= '1995-01-01' AND o_orderdate < DATE_ADD('1995-01-01', INTERVAL '1' YEAR)
GROUP BY n_name ORDER BY revenue desc;
```

```
-> Sort: revenue DESC
  -> Table scan on <temporary>
     -> Aggregate using temporary table
        -> Nested loop inner join  (cost=4288570.15 rows=261515)
           -> Nested loop inner join  (cost=2457968.58 rows=5230290)
              -> Nested loop inner join  (cost=1605724.98 rows=1307877)
                 -> Nested loop inner join  (cost=30335.99 rows=300000)
                    -> Nested loop inner join  (cost=1.88 rows=5)
                       -> Filter: (region.r_name = 'ASIA')  (cost=0.75 rows=1)
                          -> Table scan on region  (cost=0.75 rows=5)
                       -> Index lookup on nation using i_n_regionkey (n_regionkey=region.r_regionkey)  (cost=1.12 rows=5)
                    -> Covering index lookup on customer using i_c_nationkey (c_nationkey=nation.n_nationkey)  (cost=1266.82 rows=60000)
                 -> Filter: ((orders.o_orderDATE >= DATE'1995-01-01') and (orders.o_orderDATE < <cache>(('1995-01-01' + interval '1' year))))  (cost=3.75 rows=4)
                    -> Index lookup on orders using i_o_custkey (o_custkey=customer.c_custkey)  (cost=3.75 rows=15)
              -> Filter: (lineitem.l_suppkey is not null)  (cost=0.25 rows=4)
                 -> Index lookup on lineitem using PRIMARY (l_orderkey=orders.o_orderkey)  (cost=0.25 rows=4)
           -> Filter: (supplier.s_nationkey = nation.n_nationkey)  (cost=0.25 rows=0)
              -> Single-row index lookup on supplier using PRIMARY (s_suppkey=lineitem.l_suppkey)  (cost=0.25 rows=1)
```

Use NO_INDEX Hint to Force Hash Join

TPC-H Q5 (Local Supplier Volume Query)

```
SELECT /*+ JOIN_PREFIX(region, nation, customer, orders) NO_INDEX(orders) */
     n_name, sum(l_extendedprice * (1 - l_discount)) as revenue
FROM customer JOIN orders ON c_custkey = o_custkey JOIN lineitem ON l_orderkey = o_orderkey
     JOIN supplier ON l_suppkey = s_suppkey AND c_nationkey = s_nationkey
     JOIN nation ON s_nationkey = n_nationkey JOIN region ON n_regionkey = r_regionkey
WHERE r_name = 'ASIA'
  AND o_orderdate >= '1995-01-01' AND o_orderdate < DATE_ADD('1995-01-01', INTERVAL '1' YEAR)
GROUP BY n_name ORDER BY revenue desc;
```

```
-> Sort: revenue DESC
 -> Table scan on <temporary>
     -> Aggregate using temporary table
         -> Nested loop inner join  (cost=2619833.68 rows=0)
             -> Nested loop inner join  (cost=2619828.53 rows=2)
                 -> Inner hash join (orders.o_custkey = customer.c_custkey)  (cost=2245587.94 rows=0)
                     -> Filter: ((orders.o_orderDATE >= DATE'1995-01-01') and (orders.o_orderDATE < <cache>(('1995-01-01' + interval '1' year))))  (cost=6.89 rows=5)
                         -> Table scan on orders  (cost=6.89 rows=15000000)
                     -> Hash
                         -> Nested loop inner join  (cost=30335.99 rows=300000)
                             -> Nested loop inner join  (cost=1.88 rows=5)
                                 -> Filter: (region.r_name = 'ASIA')  (cost=0.75 rows=1)
                                     -> Table scan on region  (cost=0.75 rows=5)
                                 -> Index lookup on nation using i_n_regionkey (n_regionkey=region.r_regionkey)  (cost=1.12 rows=5)
                             -> Covering index lookup on customer using i_c_nationkey (c_nationkey=nation.n_nationkey)  (cost=1266.82 rows=60000)
                 -> Filter: (lineitem.l_suppkey is not null)  (cost=0.25 rows=4)
                     -> Index lookup on lineitem using PRIMARY (l_orderkey=orders.o_orderkey)  (cost=0.25 rows=4)
             -> Filter: (supplier.s_nationkey = nation.n_nationkey)  (cost=0.00 rows=0)
                 -> Single-row index lookup on supplier using PRIMARY (s_suppkey=lineitem.l_suppkey)  (cost=0.00 rows=1)
```

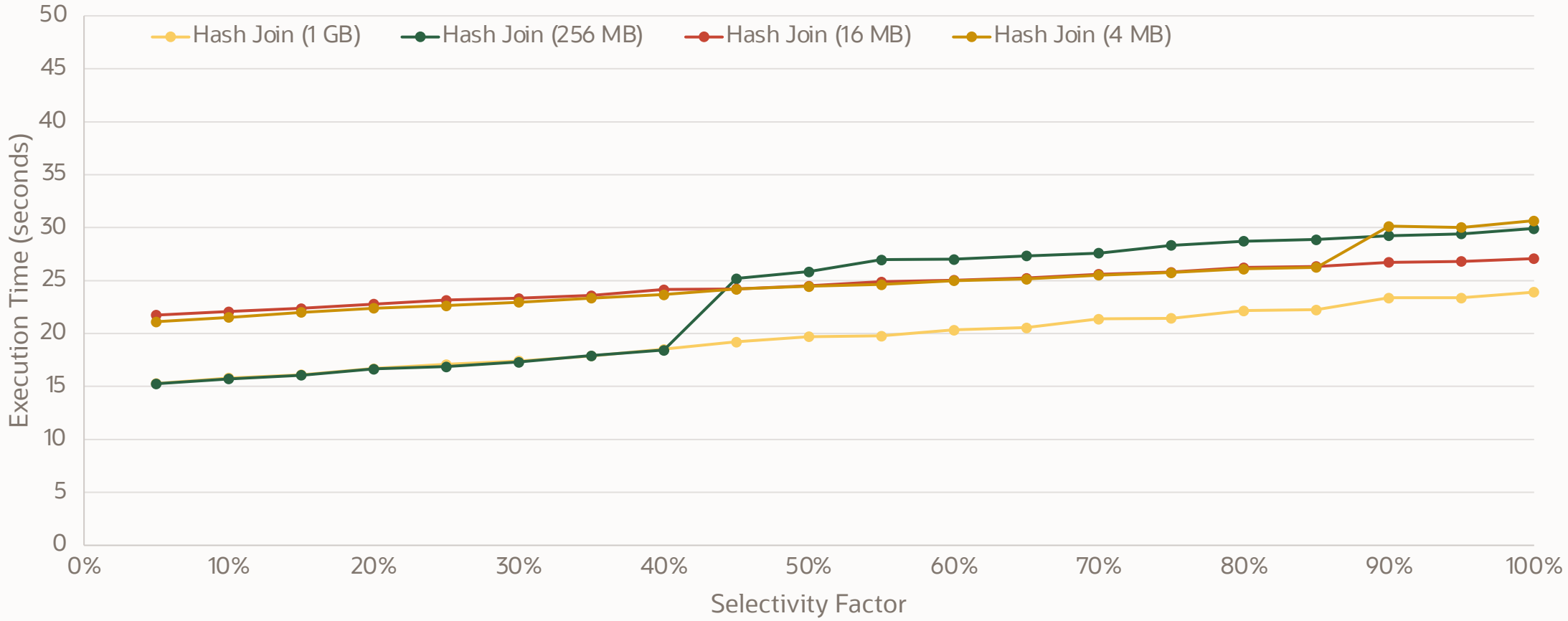
Probe Input

Build Input

Join Buffer Size Matters

But not that much

SELECT COUNT(*) FROM orders JOIN lineitem ON l_orderkey = o_orderkey WHERE o_custkey <= 1500000 * *selectivity*

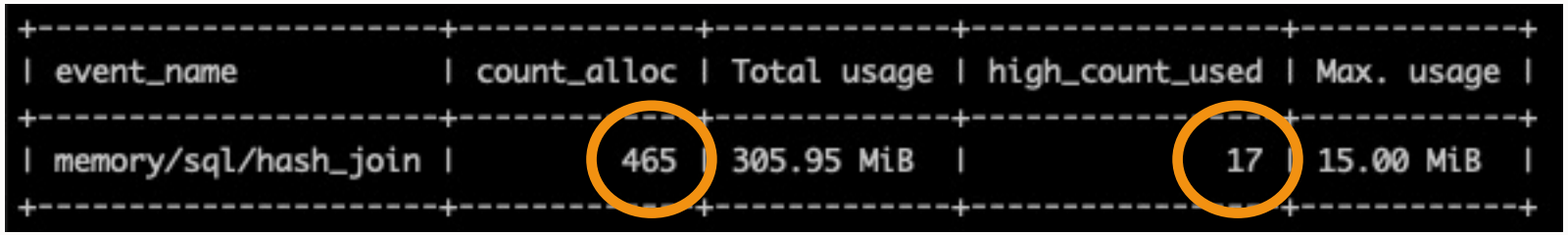


Is Join Buffer Big Enough for In-Memory Hash Join?

Check `performance_schema.memory_summary_global_by_event_name`

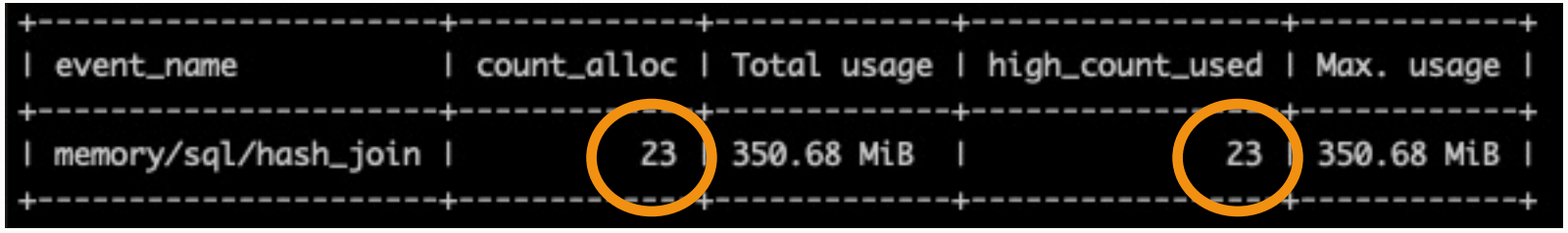
```
TRUNCATE performance_schema.memory_summary_global_by_event_name;
< Run query >
SELECT event_name, count_alloc, format_bytes(sum_number_of_bytes_alloc) "Total usage",
     high_count_used, format_bytes(high_number_of_bytes_used) "Max. usage"
FROM performance_schema.memory_summary_global_by_event_name
WHERE event_name LIKE 'memory/sql/hash_join';
```

SET join_buffer_size = 32*1024*1024;



event_name	count_alloc	Total usage	high_count_used	Max. usage
memory/sql/hash_join	465	305.95 MiB	17	15.00 MiB

SET join_buffer_size = 1024*024*1024;

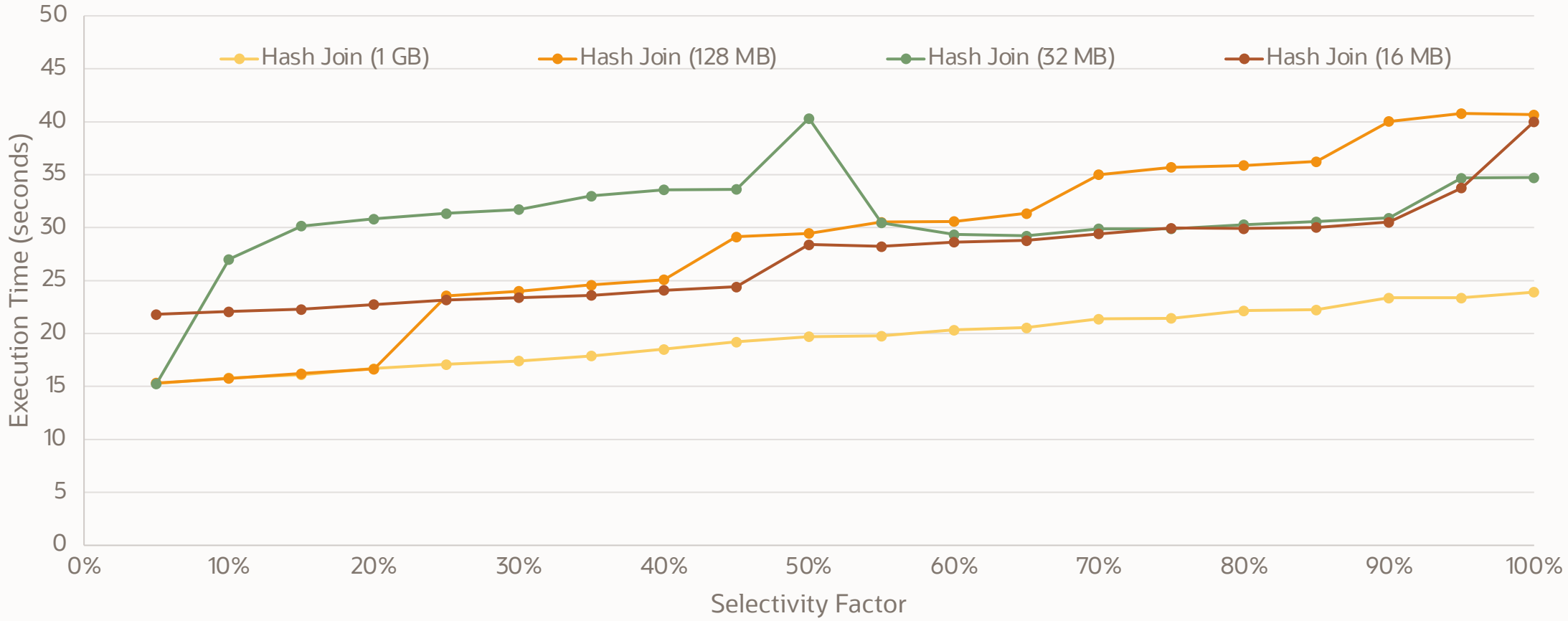


event_name	count_alloc	Total usage	high_count_used	Max. usage
memory/sql/hash_join	23	350.68 MiB	23	350.68 MiB

Add Histograms

Reduced performance without histogram on `o_custkey`

SELECT COUNT(*) FROM orders JOIN lineitem ON l_orderkey = o_orderkey WHERE o_custkey <= 1500000 * *selectivity*



Summary

Summary

- Hash join in MySQL 8.0 gives better join performance when
  - No index is available
  - Query is IO-bound
  - Large part of a table will be accessed
  - Selective conditions on multiple tables
- Use NO_INDEX() hint to force hash join to be used
- Increasing `join_buffer_size` may improve performance
- Remember to create histograms!
  - ANALYZE TABLE … UPDATE HISTOGRAM …

oracle.com