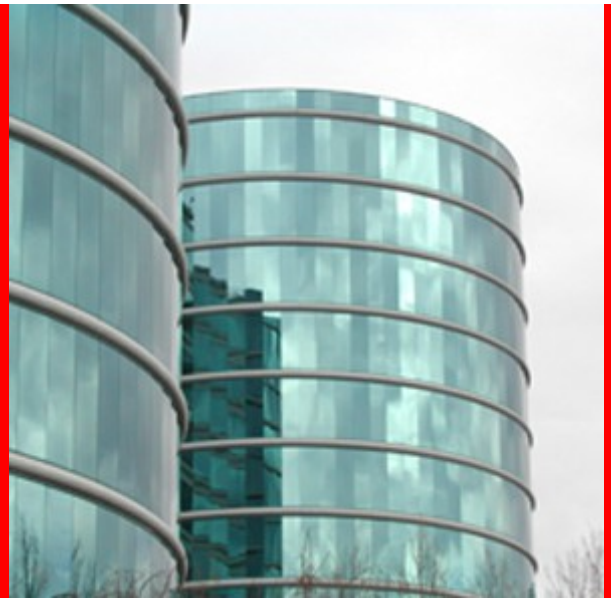


MySQL™

ORACLE®

25x: MySQL Cluster and push-down joins (in pursuit of the holy grail)

Jonas Oreland

25x: MySQL Cluster and push-down joins (in pursuit of the holy grail)

What is your name: Jonas Oreland, Oracle/Sun/MySQL

What is your quest: Making MySQL Cluster superior and affordable to all

What is the air-speed velocity of an unladen swallow: 25x

The knights of Ni: Ole John Aske, Jan Wedvik

ftp://ftp.mysql.com:/pub/mysql/download/cluster_telco/mysql-5.1.44-ndb-7.1.3-spj-preview/mysql-cluster-gpl-7.1.3-spj-preview.tar.gz

25x: MySQL Cluster and push-down joins (in pursuit of the holy grail)

Table of content:

Introduction to MySQL Cluster – part I

What is ndb:

- a distributed hash table with a relational model (rows/columns)
- automatic/configurable horizontal partitioning
- built in configurable redundancy (synchronous replication)
- row level locking
- logging/check pointing
- data stored in main-memory or on disk (with page buffer cache) (configurable on column level)
- online schema change (add column, create/drop index)
- online repartitioning (adding partitions)
- online adding of nodes
- online backup

Introduction to MySQL Cluster – part II

What is MySQL Cluster

ndb and set of connectors and add-ons:

- C/C++ ndbapi, native client library
- SQL MySQL + ha_ndbcluster.cc
- LDAP OpenLDAP + backndb (using ndbapi)
- Java ClusterJPA (using ClusterJ via ndbapi)

- MySQL replication with ha_ndbcluster_binlog.cc (geo redundancy)

Introduction to MySQL Cluster – part III

What are the primitive data access methods supported by ndb

- primary key lookup
- unique key lookup (impl. as 2-way primary key lookup)
- table scan (parallel or pruned) with push down conditions
- index scan (parallel or pruned) with push down (multi) key-ranges and push down conditions

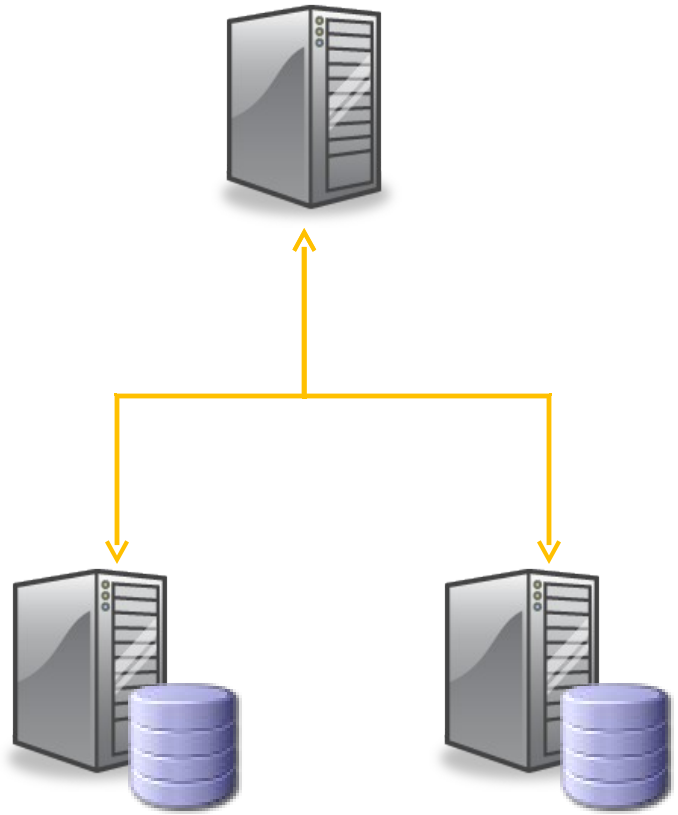
Why joins sometimes are slow with MySQL Cluster – part I

TPC-W getBestSeller

3-way join, subquery, group by, order by

```
SELECT i_id, i_title, a_fname, a_lname
FROM item, author, order_line
WHERE item.i_id = order_line.ol_i_id
  AND item.i_a_id = author.a_id
  AND order_line.ol_o_id > (SELECT MAX(o_id)-3333
                            FROM orders)
  AND item.i_subject = 'COMPUTERS'
GROUP BY i_id, i_title, a_fname, a_lname
ORDER BY SUM(ol_qty) DESC
LIMIT 50;
```

Why joins sometimes are slow with MySQL Cluster – part II

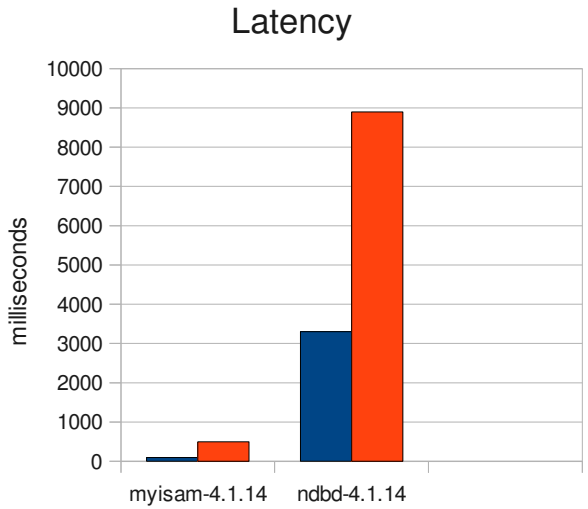


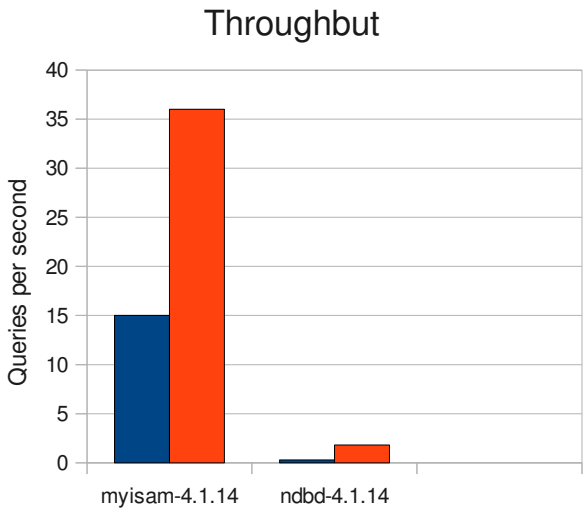
- mysql server
  1xDual Intel 5160 3GHz
- gigabit ethernet
- 2 data-nodes
  2xQuad Intel E5450 3GHz

Why joins sometimes are slow with MySQL Cluster – part III

2004 the saga begins

Why joins sometimes are slow with MySQL Cluster – part IV





- Blue is single thread
- Red is 16-threads

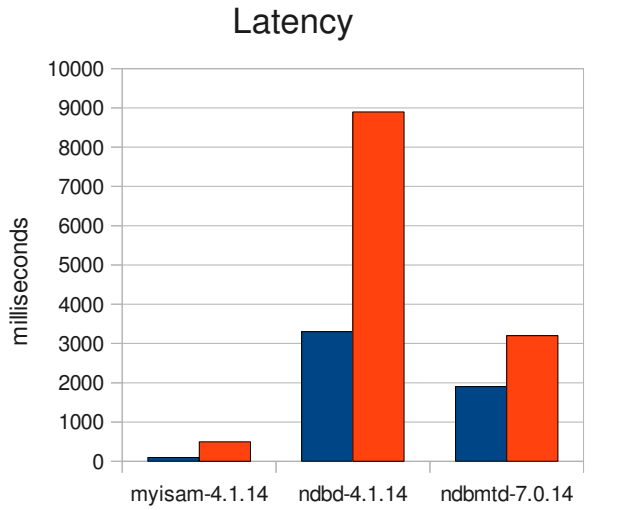
- Left is myisam 4.1.14
- Right is ndbd 4.1.14

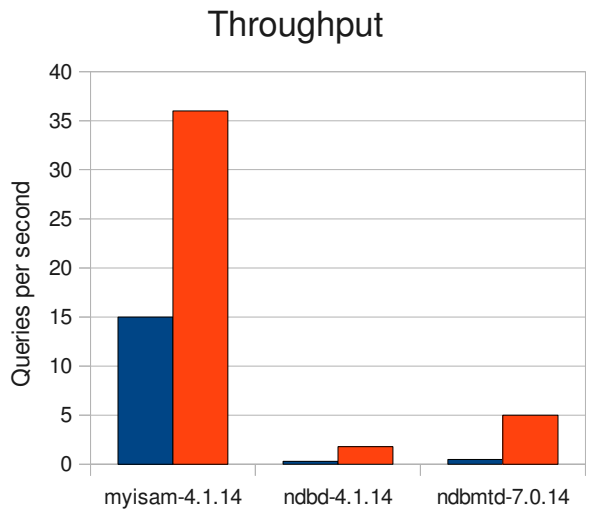
Horror!

Why joins sometimes are slow with MySQL Cluster – part V

Fast forward to 2009

Why joins sometimes are slow with MySQL Cluster – part VI





- Blue is single thread
- Red is 16-threads

- Left is myisam 4.1.14
- Middle is ndbd 4.1.14
- Right is ndbmtd 7.0.14

- Better but still bad!
- No algorithmic changes!

Why joins sometimes are slow with MySQL Cluster – part VII

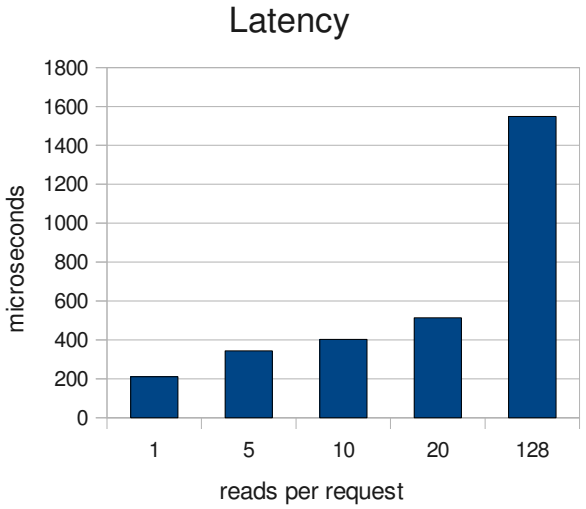
Nested Loop Join

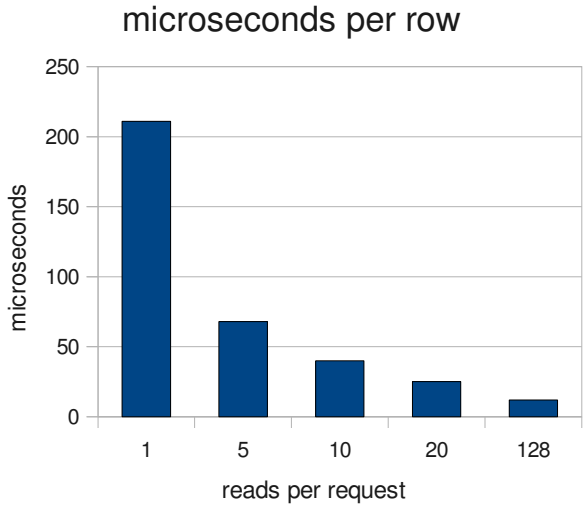
```
FOR EACH ROW <a> in TABLE T1 (matching conditions on T1)
  FOR EACH ROW <b> in TABLE T2 (matching condition on T2 given <a>)
    FOR EACH ROW <c> in TABLE T3 (matching conditions on T3 given <b>)
```

FOR EACH is implemented using one of the 4 primitive data access methods in ndb

NOTICE: Everything is done 1 row at a time. Zero parallelism!

Why joins sometimes are slow with MySQL Cluster – part VIII



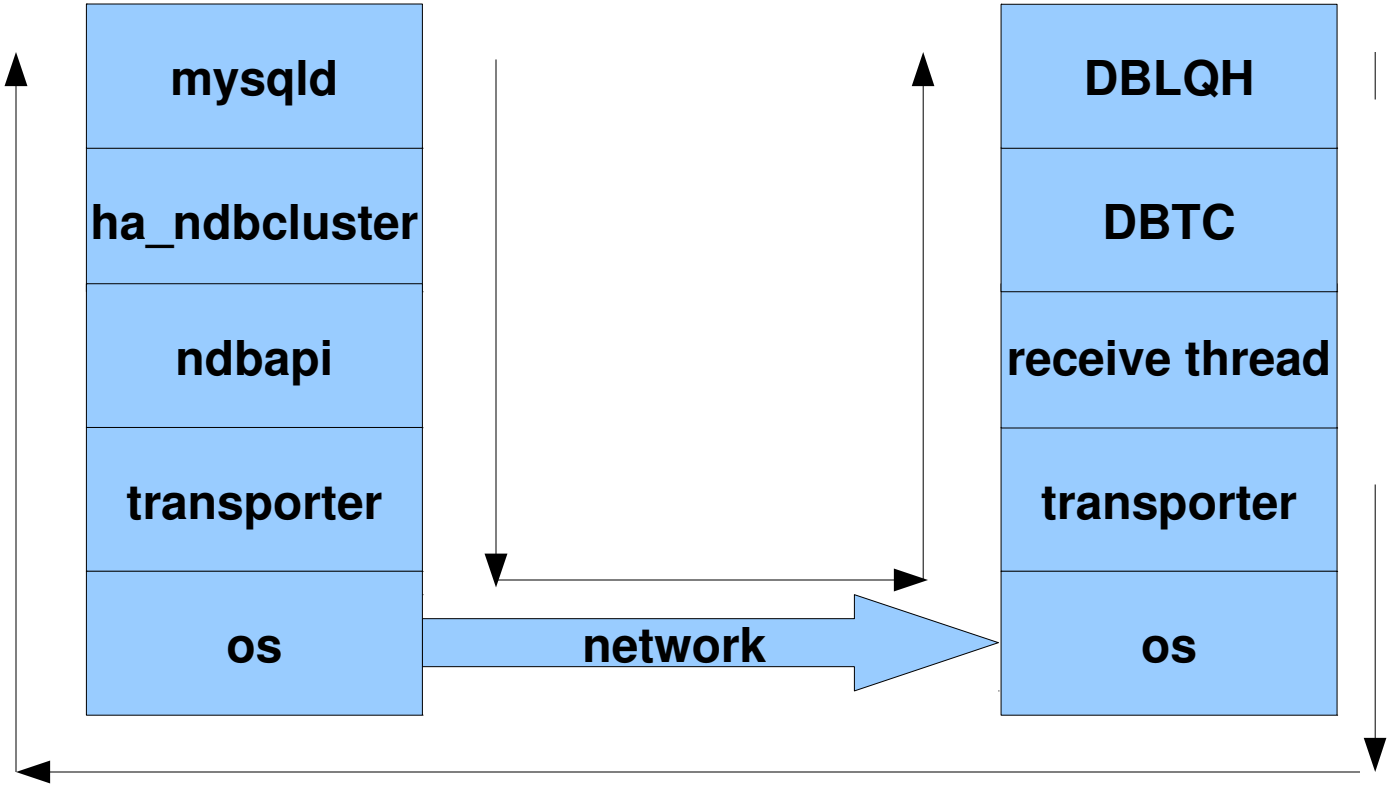


Ping time: 100 microseconds

- Latency for 1 primary key operation is 211 microseconds
- Latency for 128 primary key operations is 1548 microseconds

- Time per row for 1 primary key operations is 211 microseconds
- Time per row for 128 primary key operations is 12 microseconds

Why joins sometimes are slow with MySQL Cluster – part IX



Why joins sometimes are slow with MySQL Cluster – part X

```
mysql> explain SELECT i_id, i_title, a_fname, a_lname FROM item, author, order_line WHERE item.i_id = order_line.ol_i_id AND
item.i_a_id = author.a_id    AND order_line.ol_o_id > (SELECT MAX(o_id)-3333 FROM orders) AND item.i_subject = 'COMPUTERS' GROUP BY
i_id, i_title, a_fname, a_lname ORDER BY SUM(ol_qty) DESC limit 50;
```

select_type	table	type	key	ref	Extra
PRIMARY	**order_line**	**range**	**PRIMARY**	**NULL**	**Using where; Using temporary; Using filesort**
PRIMARY	item	eq_ref	PRIMARY	test.order_line.OL_I_ID	Using where with pushed condition
PRIMARY	author	eq_ref	PRIMARY	test.item.I_A_ID	
SUBQUERY	NULL	NULL	NULL	NULL	Select tables optimized away

```
mysql> select count(*) from order_line where order_line.ol_o_id >
(SELECT MAX(o_id)-3333 FROM orders);
+----------+
| count(*) |
+----------+
|    10006 |
+----------+
1 row in set (0.04 sec) (41090 us e.g 4 us / row)
```

Why joins sometimes are slow with MySQL Cluster – part XI

```
mysql> select count(*) from item, order_line where item.i_subject = 'COMPUTERS' and
item.i_id = order_line.ol_i_id and order_line.ol_o_id > (SELECT MAX(o_id)-3333 FROM
orders);
+----------+
| count(*) |
+----------+
|      420 |
+----------+
```

Latency for 1 primary key operations is 211 microseconds

Query time = 41090 + (10006*211) + (420*211) = 2240976 = **2.2 s**

Why joins sometimes are slow with MySQL Cluster – part IX

So we need to...

cut down the mightiest tree in the forest...with....A HERRING!

BKA – part I

Batched Key Access

```
FOR EACH ROW <a> in TABLE T1 (matching conditions on T1)
  Gather <a0...an>
  FOR EACH ROW <b> in TABLE T2 (matching condition on T2 given <a0..an>)
    Gather <b0...bn>
    FOR EACH ROW <c> in TABLE T3 (matching conditions on T3 given <b0..bn>)
```

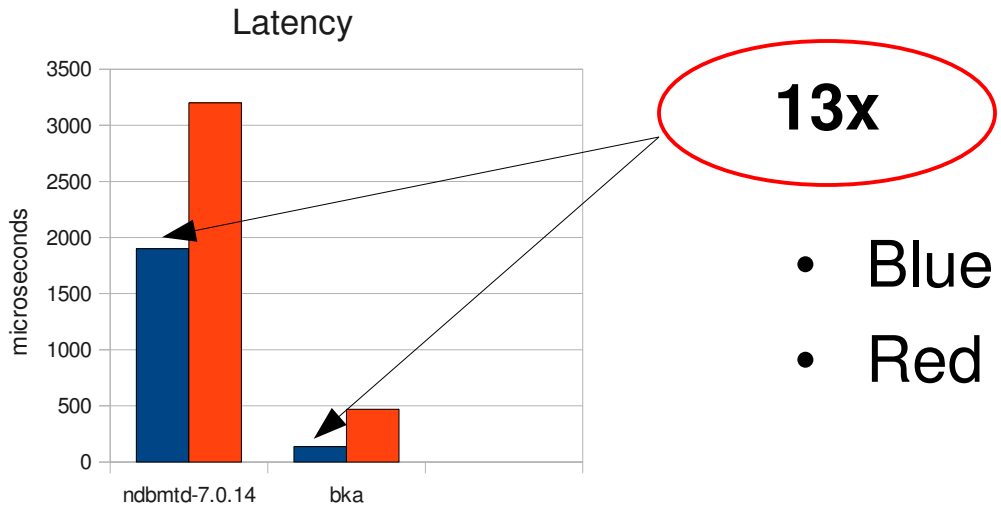
with max n = 128 (as mysql-6.0-bka-preview)

Query time = 41090 + (10006/128)*1548 + (420/128)*1548 = 167179 us = 167 ms

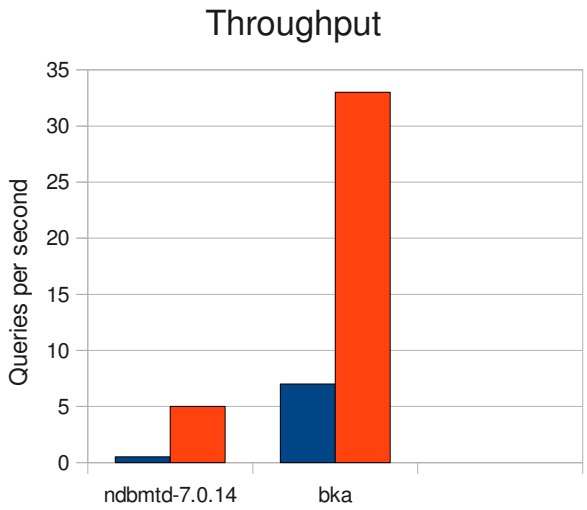
13x

Latency for 128 primary key operations is 1548 microseconds

BKA – part II



- Blue is single thread
- Red is 16-threads



- Left is ndbmtd 7.0.14
- Right is BKA

BKA – part III

So what is wrong with BKA ?

Nothing!
It's great!!

BKA – part IV

Really, what is “wrong” with BKA ?

- it's not released yet
- for low cardinality it does not help at all,
  as it processes 1 table at a time
  e.g `select from T1, T2 where T1.pk = X and T2.pk = T1.a`
- It's “just” a new access method, that can by itself not limit number of rows shipped to mysqld

Distributed push-down joins – part I

What if ?

Distributed push-down joins – part I

What if ?

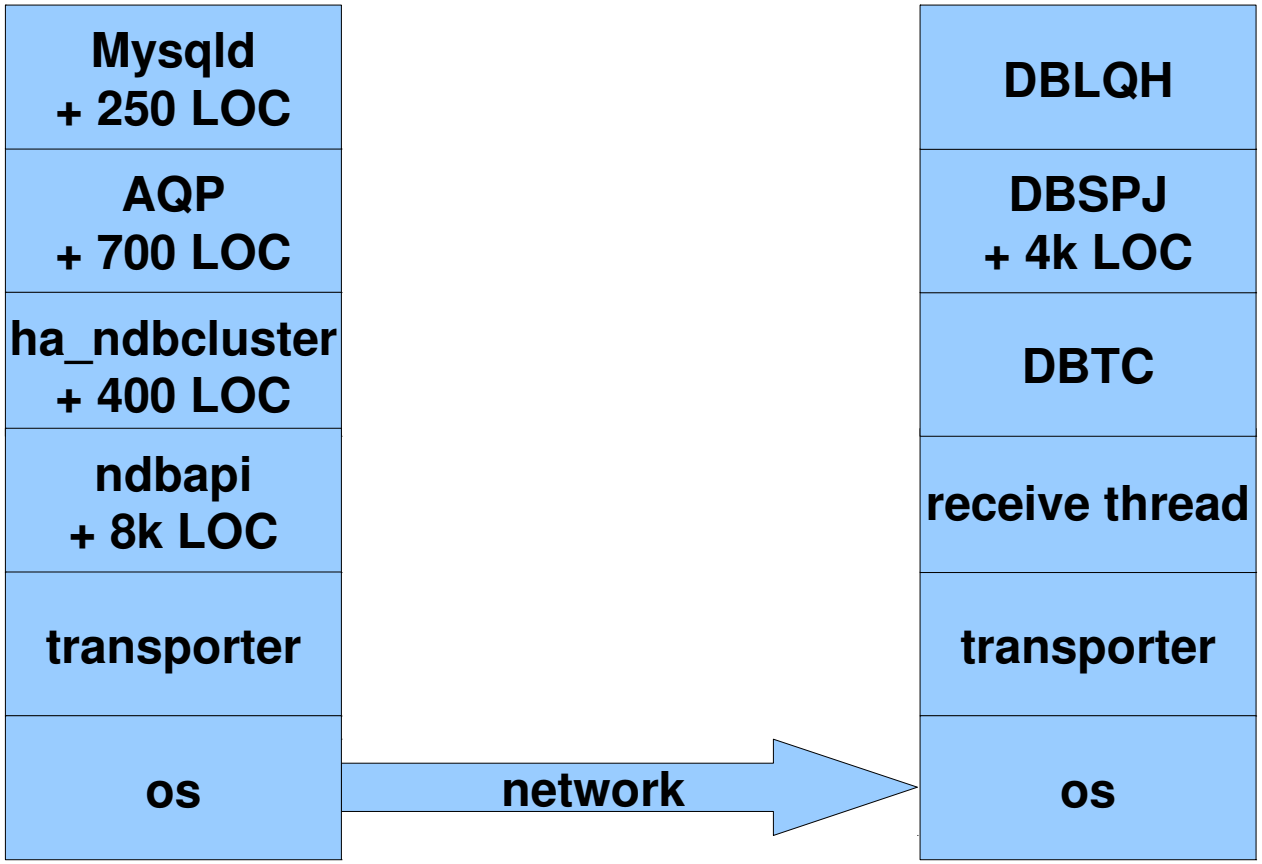
A access method which could combine the existing data access methods,

that could evaluate joins or parts of joins without

transporting all rows to mysqld...

(e.g a killer rabbit!)

Distributed push-down joins – part III



Distributed push-down joins – part II

Nested Loop Join inside DBSPJ

- Start “thread” scanning local partitions for T1
- On row found in T1
  Start “thread” searching for row in T2
- On row found in T2
  Start “thread” searching for row in T3
- When all threads are finished, report back

NOTICE: Everything is asynchronous, as much as possible is performed in parallel

Distributed push-down joins – part IV

MySQL Integration

1.JOIN::prepare

2.JOIN::optimize

3.handler::make_pushed_join(AQP)

4.JOIN::exec

5.JOIN::cleanup

6.JOIN::reinit

Expose query execution plan **after** query optimization

Distributed push-down joins – part V

Abstract Query Plan

MySQL server

Storage Engine

Distributed push-down joins – part VI

```
mysql> show variables like 'ndb_join_pushdown';
+--------------------+-------+
| Variable_name      | Value |
+--------------------+-------+
| ndb_join_pushdown | ON    |
+--------------------+-------+
1 row in set (0.00 sec)
```

Distributed push-down joins – part VII

EXPLAIN!

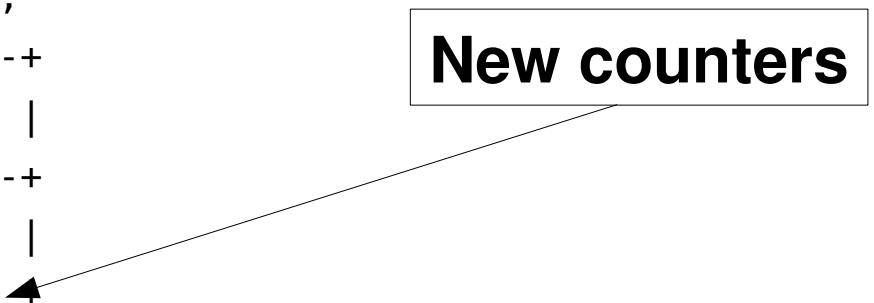
```
mysql> explain SELECT i_id, i_title, a_fname, a_lname FROM item, author, order_line WHERE item.i_id = order_line.ol_i_id AND
item.i_a_id = author.a_id AND order_line.ol_o_id > (SELECT MAX(o_id)-3333 FROM orders) AND item.i_subject = 'COMPUTERS' GROUP BY
i_id, i_title, a_fname, a_lname ORDER BY SUM(ol_qty) DESC limit 50;
+----------+------------+--------+--------------------+-----------------------------------------------------------------------+
| type     | table      | type   | ref                | Extra                                                                 |
+----------+------------+--------+--------------------+-----------------------------------------------------------------------+
| PRIMARY  | order_line | range  | NULL               | Parent of 3 pushed join@1; Using where; Using temporary; Using filesort |
| PRIMARY  | item       | eq_ref | order_line.OL_I_ID | Child of pushed join@1; Using where with pushed condition             |
| PRIMARY  | author     | eq_ref | item.I_A_ID        | Child of pushed join@1                                                |
| SUBQUERY | NULL       | NULL   | NULL               | Select tables optimized away                                          |
+----------+------------+--------+--------------------+-----------------------------------------------------------------------+
4 rows in set (0.00 sec)
```

New keywords

Distributed push-down joins – part VIII

SHOW STATUS LIKE 'NDB_PUSHED%';

```
mysql> show status like 'ndb_pushed%';
+-----------------------------+-------+
| Variable_name               | Value |
+-----------------------------+-------+
| Ndb_pushed_queries_defined  | 1     |
| Ndb_pushed_queries_executed | 1     |
| Ndb_pushed_reads            | 10426 |
| Ndb_pushed_queries_dropped  | 0     |
+-----------------------------+-------+
4 rows in set (0.00 sec)
```



Distributed push-down joins – part IX

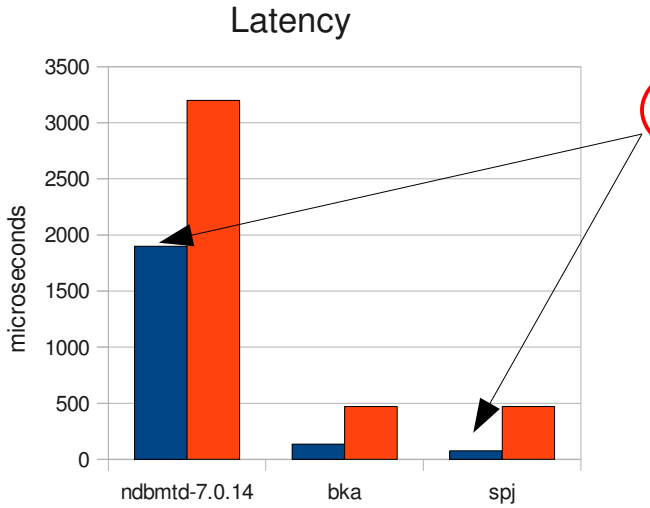
New ndbinfo counters

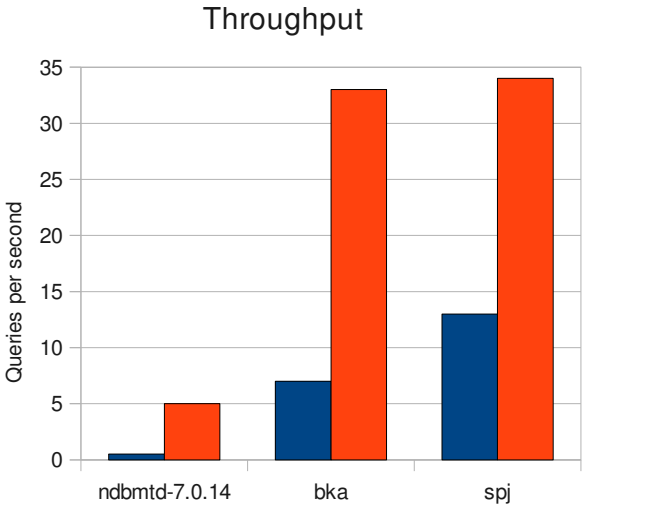
```
mysql> select node_id,counter_name,sum(val) from
ndbinfo.counters where block_name='DBSPJ' group
by node_id, counter_name;
+---------+-----------------------+------+
| node_id | counter_name          | val  |
+---------+-----------------------+------+
|       1 | READS_RECEIVED        |    0 |
|       1 | LOCAL_READS_SENT      | 5216 |
|       1 | REMOTE_READS_SENT     | 4874 |
|       1 | TABLE_SCANS_RECEIVED  |    0 |
|       1 | LOCAL_TABLE_SCANS_SENT |    0 |
|       1 | RANGE_SCANS_RECEIVED  |    4 |
|       1 | LOCAL_RANGE_SCANS_SENT |    4 |
|       2 | READS_RECEIVED        |    0 |
|       2 | LOCAL_READS_SENT      | 4754 |
|       2 | REMOTE_READS_SENT     | 5168 |
|       2 | TABLE_SCANS_RECEIVED  |    0 |
|       2 | LOCAL_TABLE_SCANS_SENT |    0 |
|       2 | RANGE_SCANS_RECEIVED  |    4 |
|       2 | LOCAL_RANGE_SCANS_SENT |    4 |
+---------+-----------------------+------+
14 rows in set (0.13 sec)
```

- LOCAL_”X”_SENT
  #”X” sent to local node
- REMOTE_”X”_SENT
  #”X” sent to remote node

- READS
- TABLE_SCANS
- RANGE_SCANS

Distributed push-down joins – part X





- Blue is single thread
- Red is 16-threads

- Left is ndbmtd 7.0.14
- Middle is BKA
- Right is SPJ-7.0.14

Distributed push-down joins – part XI

Limitations - functionality

- No datatype conversions
- No blobs
- No locks
- Only supports `eq_ref` and `const` as access method for child tables

Limitations - performance

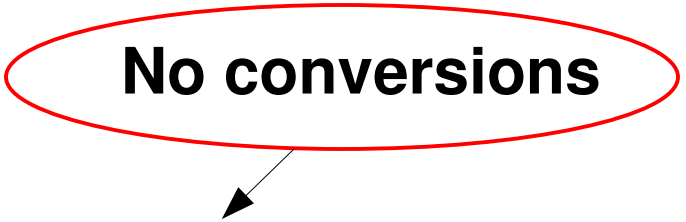
- Only supports `eq_ref` and `const` as access method for child tables
- Only implemented left outer join inside ndb(mt)d, mysqld implements inner join
- Only supports pushed filters on “root” table
- Not multi-threaded (works in ndbmtd, but executed in single thread)

Distributed push-down joins – part XII

Limitations – no datatype conversion

```
mysql> explain SELECT i_title, a_fname FROM item,author WHERE item.i_a_id = author.a_id  AND i_id = 9;
+-------------+--------+--------+---------+---------+-----------------+------+---------------------------+
| select_type | table  | type   | key     | key_len | ref             | rows | Extra                     |
+-------------+--------+--------+---------+---------+-----------------+------+---------------------------+
| SIMPLE      | item   | const  | PRIMARY | 4       | const           |    1 | Parent of 2 pushed join@1 |
| SIMPLE      | author | eq_ref | PRIMARY | 4       | test.item.I_A_ID |    1 | Child of pushed join@1    |
+-------------+--------+--------+---------+---------+-----------------+------+---------------------------+
2 rows in set (0.00 sec)

mysql> alter table author modify column A_ID bigint;
Query OK, 2500 rows affected (2.59 sec)
Records: 2500  Duplicates: 0  Warnings: 0
```




```
mysql> explain SELECT i_title, a_fname FROM item,author WHERE item.i_a_id = author.a_id  AND i_id = 9;
+-------------+--------+--------+---------+---------+-----------------+------+-------------+
| select_type | table  | type   | key     | key_len | ref             | rows | Extra       |
+-------------+--------+--------+---------+---------+-----------------+------+-------------+
| SIMPLE      | item   | const  | PRIMARY | 4       | const           |    1 |             |
| SIMPLE      | author | eq_ref | PRIMARY | 8       | test.item.I_A_ID |    1 | Using where |
+-------------+--------+--------+---------+---------+-----------------+------+-------------+
2 rows in set (0.00 sec)
```

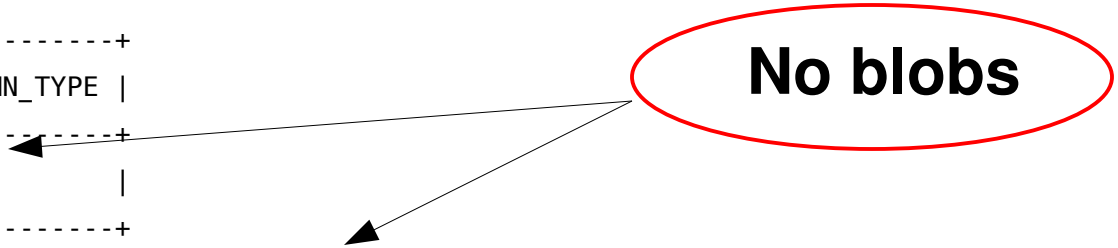
Distributed push-down joins – part XIII

Limitations – no blobs

```
mysql> explain SELECT i_title, a_fname FROM item,author WHERE item.i_a_id = author.a_id  AND i_id = 9;
+-------------+--------+--------+---------+---------+-----------------+------+--------------------------+
| select_type | table  | type   | key     | key_len | ref             | rows | Extra                    |
+-------------+--------+--------+---------+---------+-----------------+------+--------------------------+
| SIMPLE      | item   | const  | PRIMARY | 4       | const           |    1 | Parent of 2 pushed join@1 |
| SIMPLE      | author | eq_ref | PRIMARY | 4       | test.item.I_A_ID |    1 | Child of pushed join@1    |
+-------------+--------+--------+---------+---------+-----------------+------+--------------------------+

mysql> select COLUMN_NAME, COLUMN_TYPE from INFORMATION_SCHEMA.COLUMNS where TABLE_NAME='author' and
COLUMN_NAME='A_BIO';
+-------------+-------------+
| COLUMN_NAME | COLUMN_TYPE |
+-------------+-------------+
| A_BIO       | blob        |
+-------------+-------------+

mysql> explain SELECT i_title, a_fname, a_bio FROM item,author WHERE item.i_a_id = author.a_id  AND i_id = 9;
+-------------+--------+--------+---------+---------+-----------------+------+-------+
| select_type | table  | type   | key     | key_len | ref             | rows | Extra |
+-------------+--------+--------+---------+---------+-----------------+------+-------+
| SIMPLE      | item   | const  | PRIMARY | 4       | const           |    1 |       |
| SIMPLE      | author | eq_ref | PRIMARY | 4       | test.item.I_A_ID |    1 |       |
+-------------+--------+--------+---------+---------+-----------------+------+-------+
```



Distributed push-down joins – part XIV

Limitations – no locks

```
mysql> explain SELECT i_title, a_fname FROM item,author WHERE item.i_a_id = author.a_id  AND i_id = 9;
+-------------+--------+--------+---------+---------+-----------------+------+---------------------------+
| select_type | table  | type   | key     | key_len | ref             | rows | Extra                     |
+-------------+--------+--------+---------+---------+-----------------+------+---------------------------+
| SIMPLE      | item   | const  | PRIMARY | 4       | const           |    1 | Parent of 2 pushed join@1 |
| SIMPLE      | author | eq_ref | PRIMARY | 4       | test.item.I_A_ID |    1 | Child of pushed join@1    |
+-------------+--------+--------+---------+---------+-----------------+------+---------------------------+
2 rows in set (0.00 sec)
```


```
mysql> explain SELECT i_title, a_fname FROM item,author WHERE item.i_a_id = author.a_id  AND i_id = 9 FOR UPDATE;
+-------------+--------+-------+---------+---------+-------+------+-------+
| select_type | table  | type  | key     | key_len | ref   | rows | Extra |
+-------------+--------+-------+---------+---------+-------+------+-------+
| SIMPLE      | item   | const | PRIMARY | 4       | const |    1 |       |
| SIMPLE      | author | const | PRIMARY | 4       | const |    1 |       |
+-------------+--------+-------+---------+---------+-------+------+-------+
2 rows in set (0.00 sec)
```

Distributed push-down joins – part XV

Limitations – only eq_ref and const children

```
mysql> explain SELECT i_title, a_fname FROM author, item WHERE author.a_lname = 'KING ARTHUR' AND
item.i_a_id = author.a_id;
+-------------+--------+-------+---------------+-----------------+------+----------------------------------+
| select_type | table  | type  | key           | ref             | rows | Extra                            |
+-------------+--------+-------+---------------+-----------------+------+----------------------------------+
| SIMPLE      | author | range | author_a_lname | NULL            |   10 | Using where with pushed condition |
| SIMPLE      | item   | ref   | item_i_a_id    | test.author.A_ID |    1 | Using where                      |
+-------------+--------+-------+---------------+-----------------+------+----------------------------------+
```

Possible types

- const
- eq_ref
- **system**
- **ref**
- **ref_or_null**
- **index_merge**
- **unique_subquery**
- **index_subquery**
- **range**
- **index**
- **ALL**

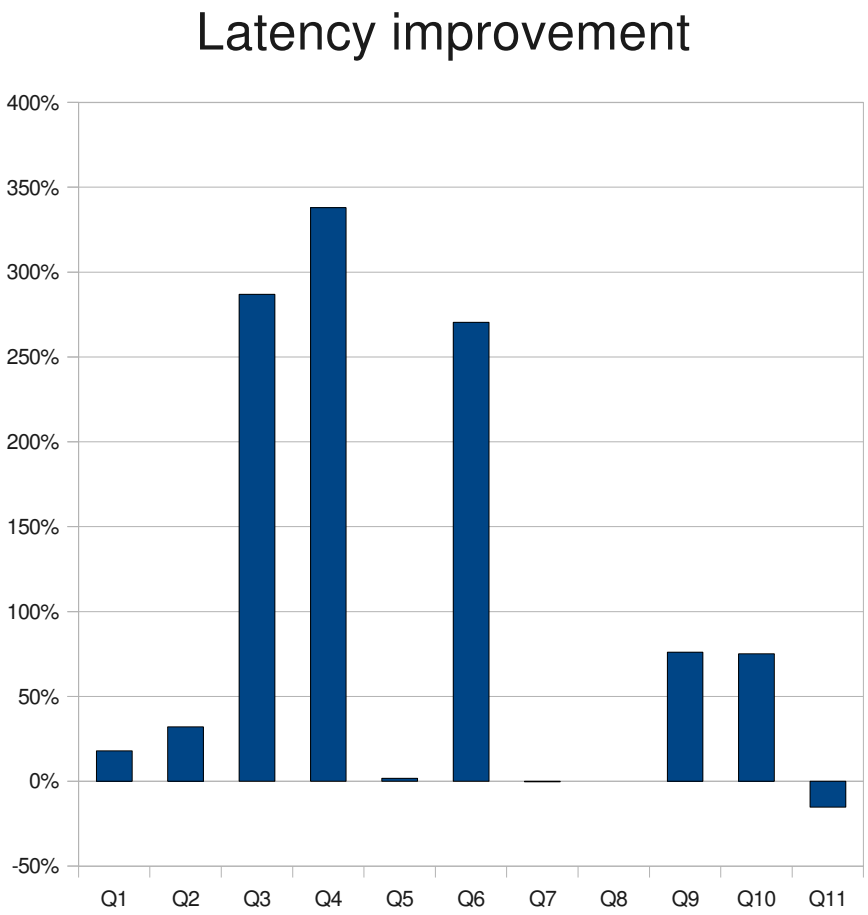
No push :-(

No push:

- **1 to many**
- **Many to many**

"to many"

Distributed push-down joins – part XVI
Test of all TPC-W queries (except getBestSellers)



Summary

- 7 noticable improvement
- 3 queries no change ("to many")
- 1 query 15% slower

So what does this mean?

- Hard to say

My **guess** is that TPC-W is unrealisticly push-friendly

Distributed push-down joins – part XVII

So what is wrong with SPJ ?

Nothing!
It's great!!

Distributed push-down joins – part XVIII

Really, what is wrong with SPJ ?

- it's not released yet
- only supporting `eq_ref` and `const` as child access types will most likely significantly limit pushability

The future! - part I

High hanging fruit

- Pushed aggregates
- More join algorithms

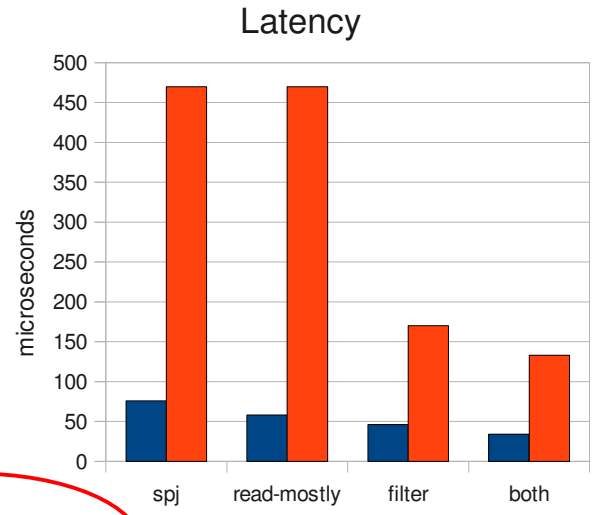
Medium hanging fruit

- “to many” (the holy handgrenade!)
- not only left outer join (in ndb(mt)d)
- “connect by” (not considering SQL)
- 2-way traveling JOIN
- read-mostly tables (fully replicated)

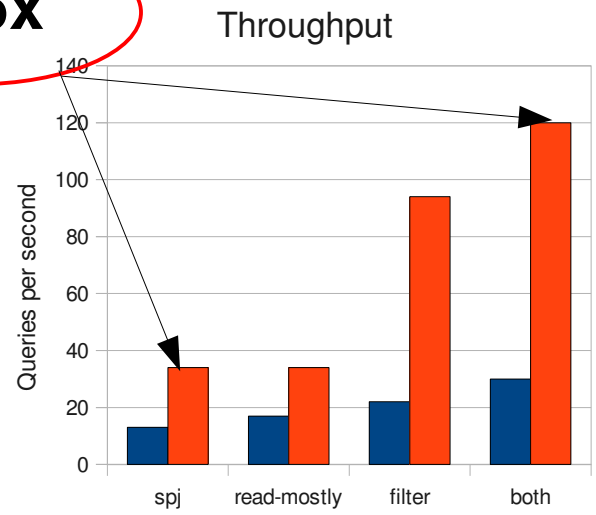
Low hanging fruit

- pushed filters also on child operations
- multi threaded DBSPJ

The future! - part II



3.5x



- Blue is single thread
- Red is 16-threads

Graphs show (left to right)

- SPJ as previously
- + w/ emulated read-mostly tables
- + w/ emulated filters on child operations
- + w/ both emulations

The future – part III (what about that swallow)

```
mysql> SELECT COUNT(*) FROM part, lineitem WHERE l_partkey=p_partkey AND
p_retailprice>2050 and l_discount > 0.04;
+----------+
| COUNT(*) |
+----------+
|    20132 |
+----------+
1 row in set (13.47 sec)
```

UC'08 BKA presentation

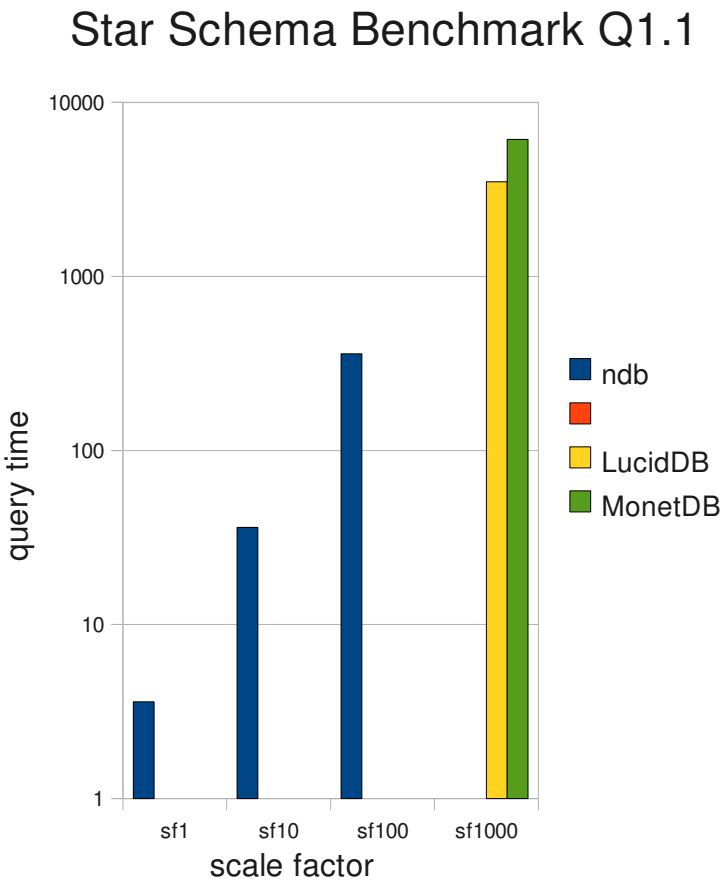
Based on TPC-H

44 times improvement!

```
mysql> set ndb_join_pushdown=off;
Query OK, 0 rows affected (0.00 sec)

mysql> SELECT COUNT(*) FROM part, lineitem WHERE l_partkey=p_partkey AND
p_retailprice>2050 and l_discount > 0.04;
+----------+
| COUNT(*) |
+----------+
|    20132 |
+----------+
1 row in set (9 min 53.03 sec)
```

The future – part III (what about that swallow)

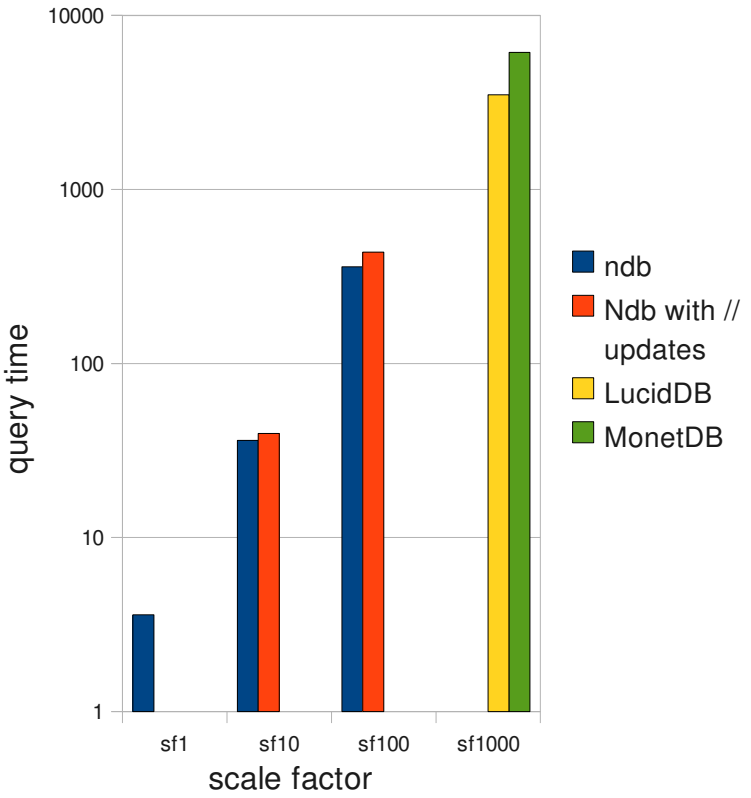


- Star Schema Benchmark is a modified TPC-H
- sf100 = 61Gb
- sf1000 = 610Gb
- LucidDB/MonetDB numbers come from http://www.percona.com/docs/wiki/b

  (using different hardware)

The future – part III (what about that swallow)

Star Schema Benchmark Q1.1



Jonas dreaming!

- MySQL Cluster will never be as fast for DSS as specialized RDBMS:es
- But! Being moderately slower and supporting 50k updates/sec in parallel can make a unique combination!

25x: MySQL Cluster and push-down joins (in pursuit of the holy grail)

Summary:

The buzzwords:	MyCluster is reasonably buzz-word compliant!
The benchmarks:	Joins **can** be slow, but it's unavoidable with current algorithms
The solutions:	SPJ and BKA both shows great potential
The future:	Time will tell!

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions.
The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.

Any resemblance to reality is purely coincidental

25x: MySQL Cluster and push-down joins (in pursuit of the holy grail)

We want you to test!

Caveat: code is known to contain bugs related to node failure handling and should therefor not be put into production

ftp://ftp.mysql.com:/pub/mysql/download/cluster_telco/mysql-5.1.44-ndb-7.1.3-spj-preview/mysql-cluster-gpl-7.1.3-spj-preview.tar.gz