HandlerSocket and similar Technologies - NoSQL for MySQL

FOSDEM, Brussels
5./6. February 2011

Oli Sennhauser

Senior MySQL Consultant at FromDual

oli.sennhauser@fromdual.com

Contents

NoSQL for MySQL

- NoSQL Trends
- SQL Overhead
- HandlerSocket
- NDB-API
- BLOB Streaming Engine
- Handler Interface
- Graph Storage Engine

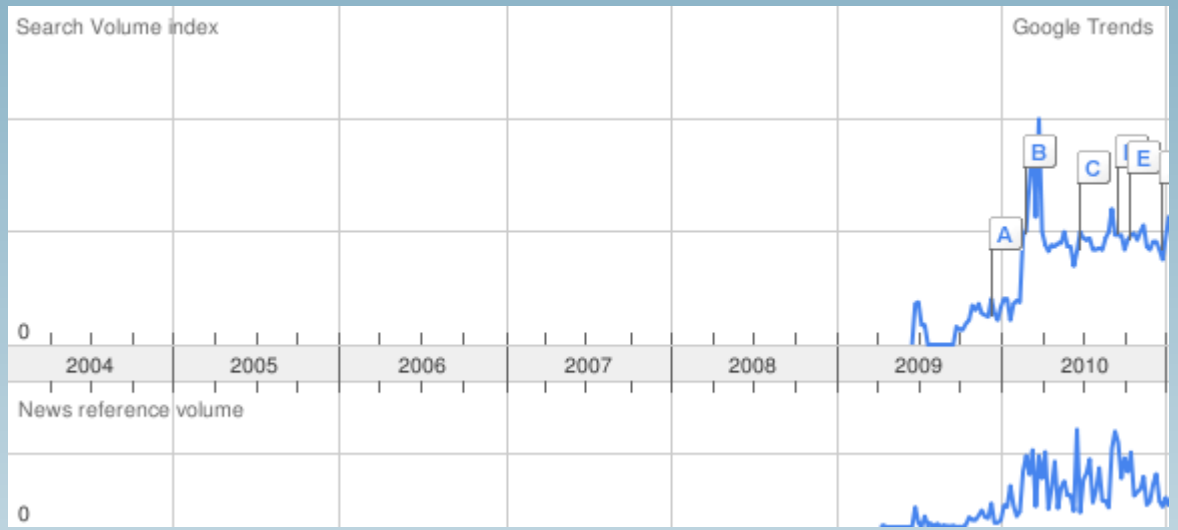
About FromDual

- We provide neutral and vendor independent:
  - Consulting (on-site and remote)
  - Remote-DBA / MySQL Operation Services
  - Support
  - Training (DBA, Performance Tuning, Scale-Out, High Availability, MySQL Cluster)
- We are consulting partner of the Open Database Alliance (ODBA.org)
- Oracle Silver Partner (OPN)

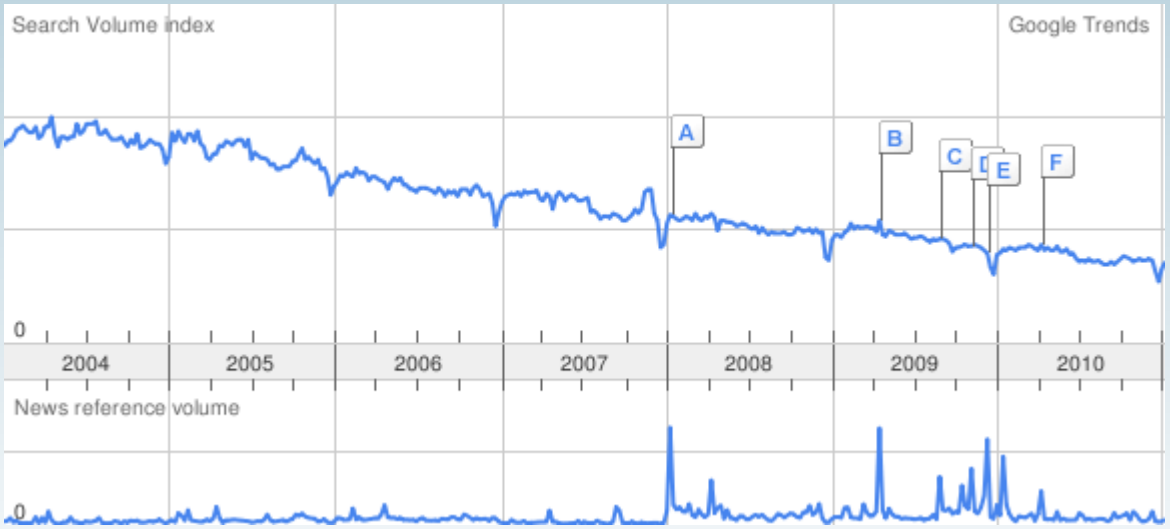
http://www.fromdual.com

Trends

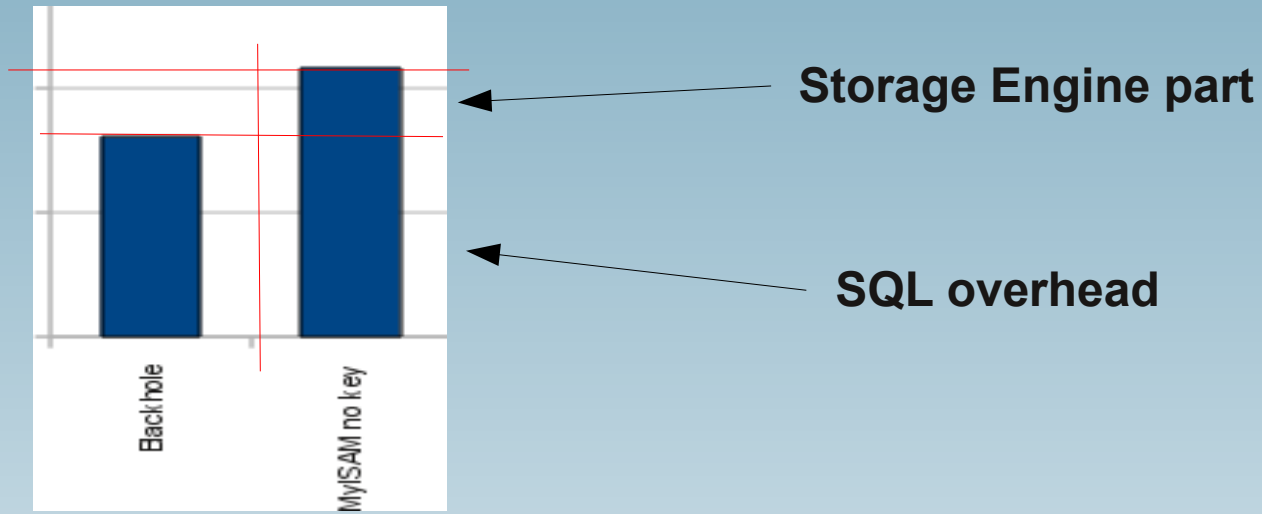
NoSQL



MySQL

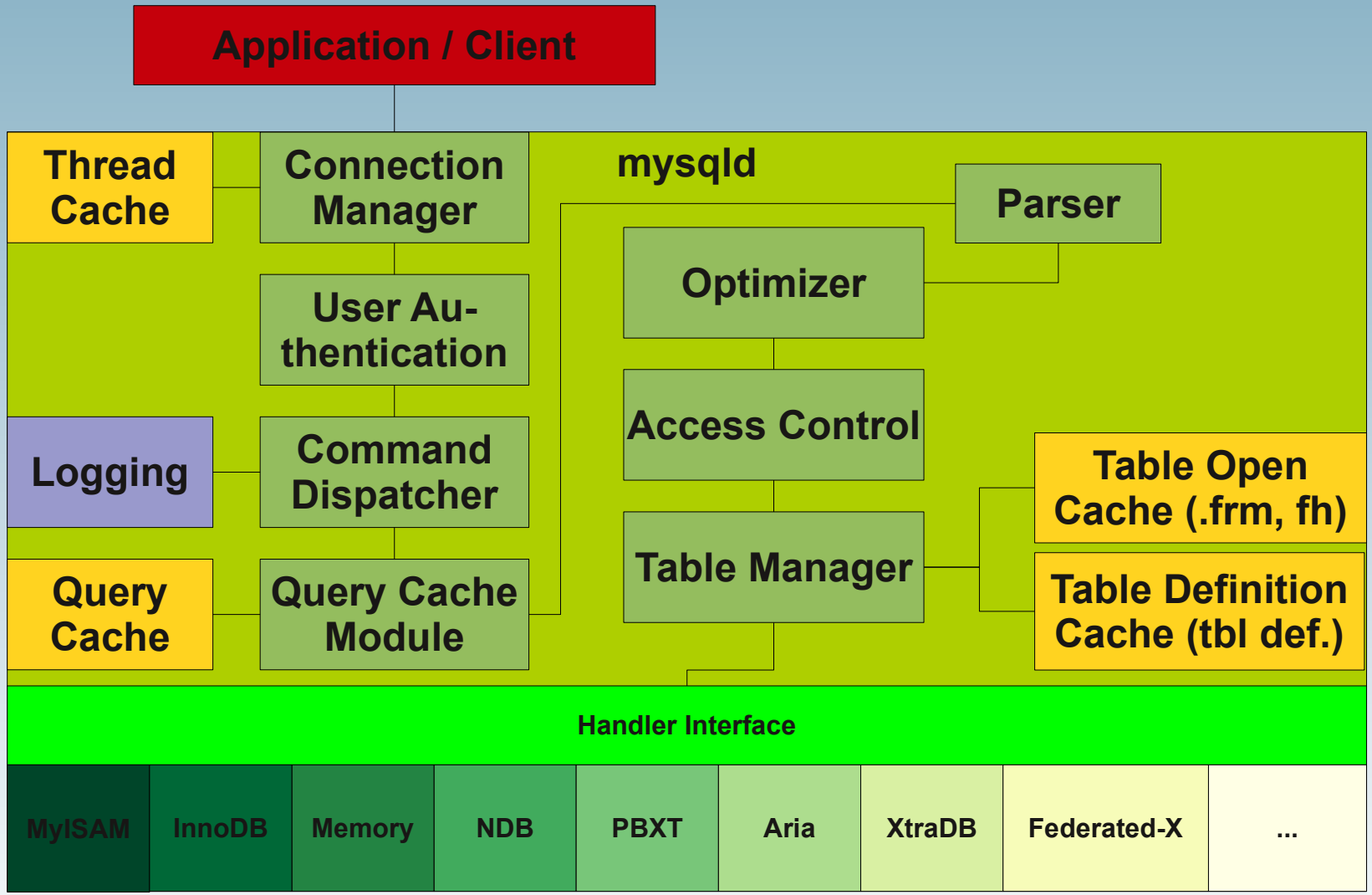


What is the problem?



- SQL overhead is up to 70-80% for simple queries (~ 1 ms)!
- with NO SQL → we could be up to 5 times faster
- SQL is made for complex queries
- NoSQL typically solves simple queries

Where does this overhead come from?



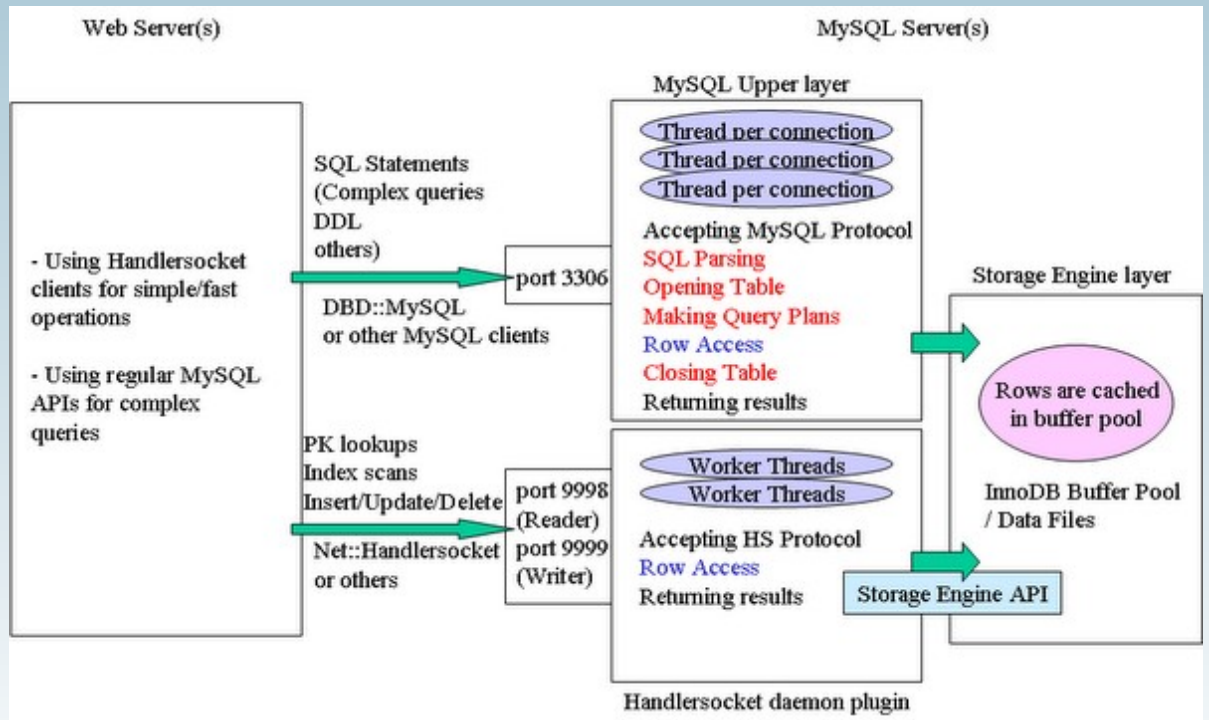
What can we do about?

- HandlerSocket (2010)
- NDB-API (1997!)
- PrimeBase Streaming Engine (2008)
- Handler Interface (2001/2011)
- OQGRAPH SE (2009)

HandlerSocket

- **October 20, 2010, Yoshinori Matsunobu:**

  Using MySQL as a NoSQL - A story for exceeding 750,000 qps on a commodity server

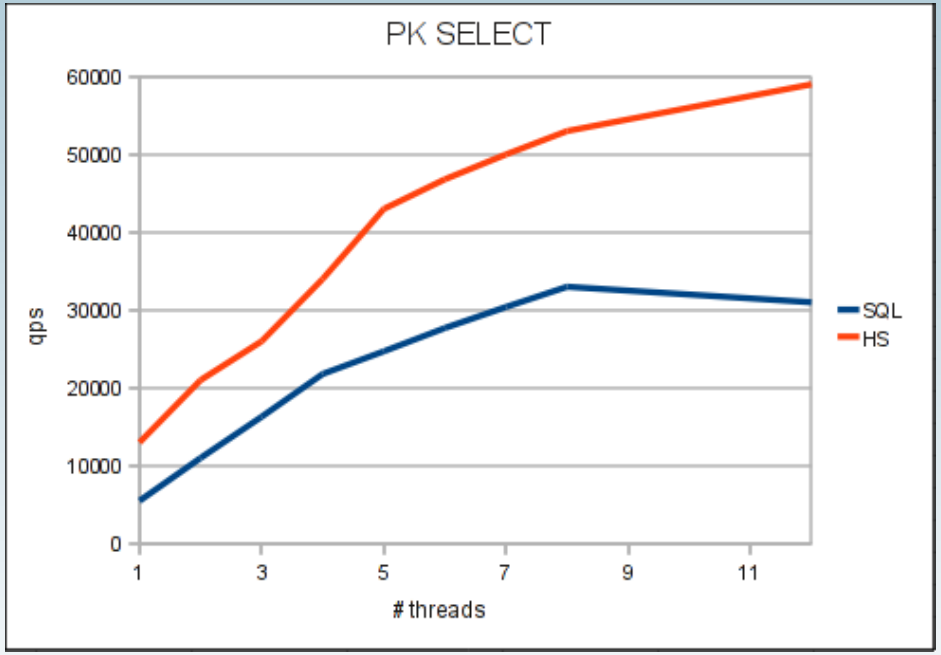


SELECT

```
# SELECT * FROM test.test where id = 42;

use Net::HandlerSocket;

my $args = { host => 'master', port => 9998 };
my $hs = new Net::HandlerSocket($args);

my $res = $hs->open_index(0, 'test', 'test', 'PRIMARY', 'id,data,ts');

$res = $hs->execute_single(0, '=', [ '42' ], 1, 0);
shift(@$res);
for (my $row = 0; $row < 1; ++$row) {
  print "$id\t$data\t$ts\n";
}

$hs->close();
```

Infos

- Compile yourself (easy!)
- 7.5 times more throughput?!?
- Works with 5.5.8 and MariaDB
- Faster than mem-cached!?!
- In Percona-Server 12.3



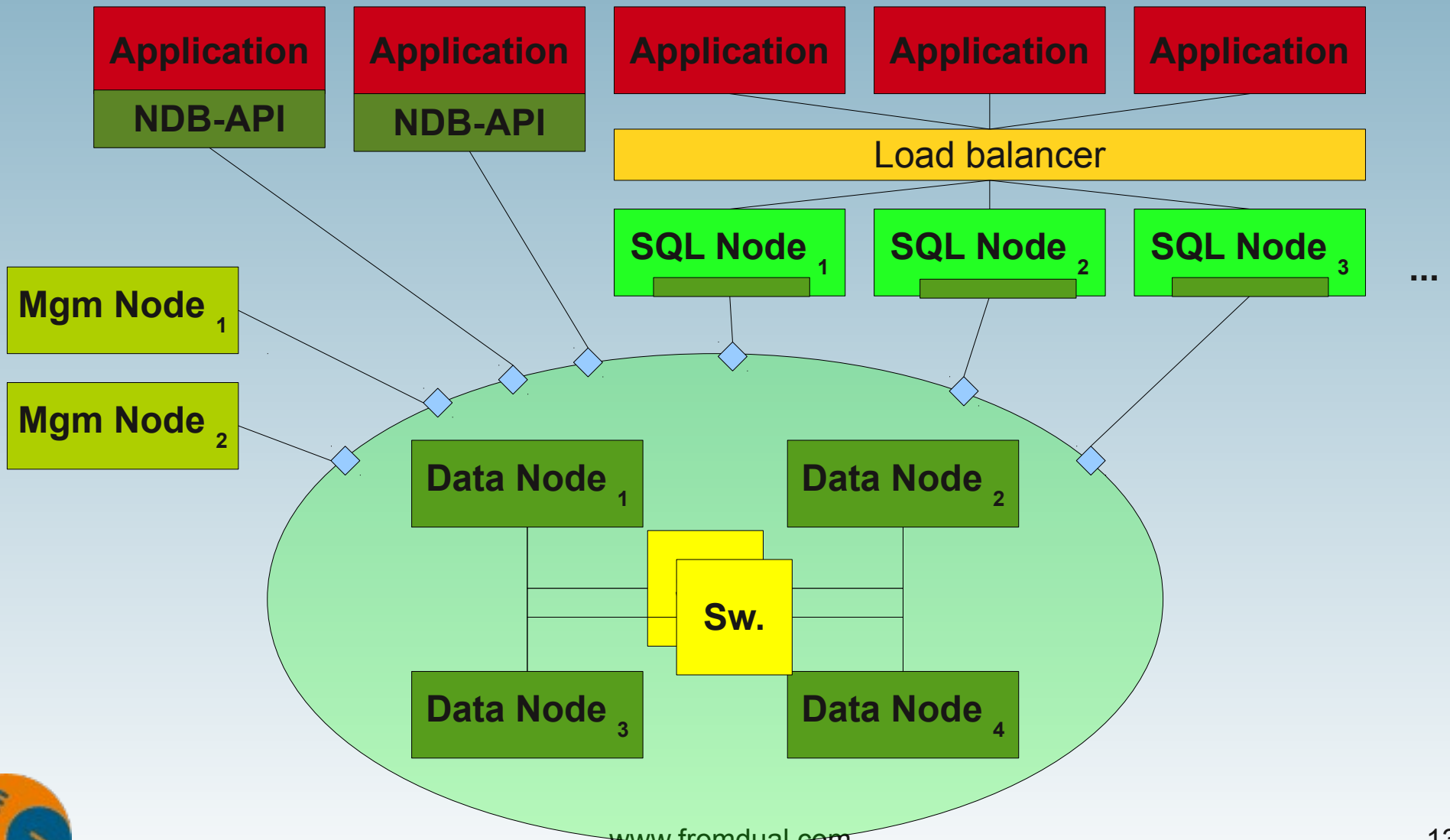
Features / functionality

- Different query patterns (see also handler interface later)
- Lots of concurrent connections
- Highly performant (200 – 700%)
- No duplicate cache (MySQL and memcached)
- No data inconsistency (MySQL and memcache)
- Crash-safe
- SQL access as well (for complex queries like reporting)
- No need to modify/rebuild MySQL ?!?
- Theoretically works for other storage engines as well (I did not test it).

NDB-API

- 1997, Mikael Ronström: The NDB Cluster – A parallel data server for telecommunications applications
- November 25, 2008, Jonas Oreland: 950'000 reads per second on 1 datanode

MySQL Cluster



INSERT

```
// INSERT INTO cars VALUES (reg_no, brand, color);

const NdbDictionary::Dictionary* myDict= myNdb->getDictionary();
const NdbDictionary::Table *myTable= myDict->getTable("cars");

NdbTransaction* myTrans = myNdb->startTransaction();
NdbOperation* myNdbOperation = myTrans->getNdbOperation(myTable);

myNdbOperation->insertTuple();
myNdbOperation->equal("REG_NO", reg_no);
myNdbOperation->setValue("BRAND", brand);
myNdbOperation->setValue("COLOR", color);

int check = myTrans->execute(NdbTransaction::Commit);

myTrans->close();
```

SELECT

```
// SELECT * FROM cars;

const NdbDictionary::Dictionary* myDict= myNdb->getDictionary();
const NdbDictionary::Table *myTable= myDict->getTable("cars");

myTrans = myNdb->startTransaction();
myScanOp = myTrans->getNdbScanOperation(myTable);
myScanOp->readTuples(NdbOperation::LM_CommittedRead)

myRecAttr[0] = myScanOp->getValue("REG_NO");
myRecAttr[1] = myScanOp->getValue("BRAND");
myRecAttr[2] = myScanOp->getValue("COLOR");

myTrans->execute(NdbTransaction::NoCommit);

while ((check = myScanOp->nextResult(true)) == 0) {

  std::cout << myRecAttr[0]->u_32_value() << "\t";
  std::cout << myRecAttr[1]->aRef() << "\t";
  std::cout << myRecAttr[2]->aRef() << std::endl;
}
myNdb->closeTransaction(myTrans);
```

Benchmarks and numbers

- ./flexAsynch -ndbrecord -temp -con 4 -t 16 -p 312 -l 3 -a 2 -r 2
- From the MySQL Cluster test suite (src/storage/ndb/test/ndbapi)

- 16 number of concurrent threads (-t)
- 312 number of parallel operation per thread
- 3 iterations (-l)
- 2 attributes per table (8 bytes) (-a)
- 4 concurrent connections (-con)
- 2 number of records (-r ?)
- 1 32-bit word per attribute
- 1 ndbmtd (1 NoOfRepl.)

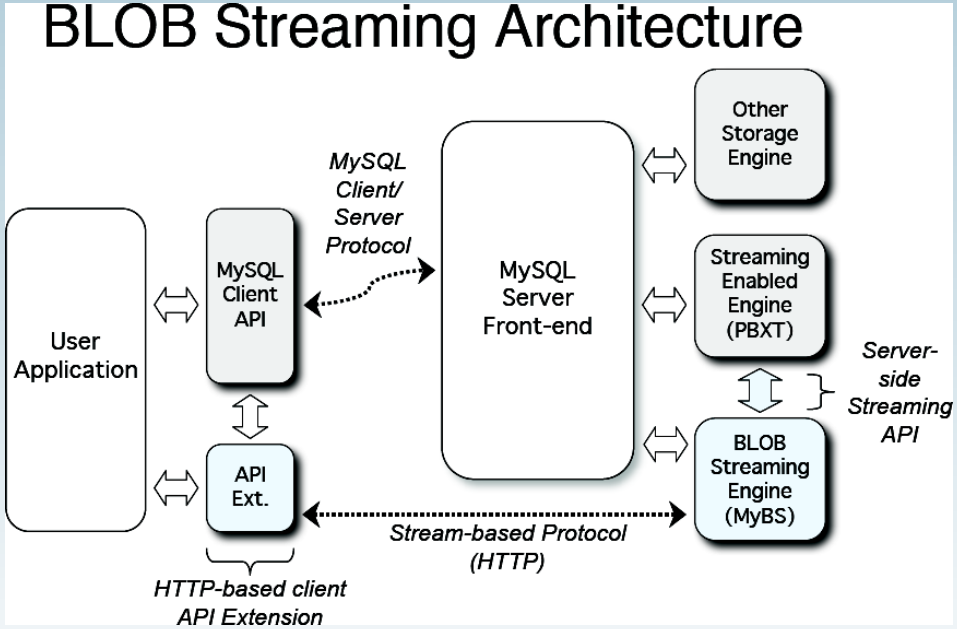
```
insert average: 506506/s min: 497508/s max: 522613/s stddev: 2%
update average: 503664/s min: 495533/s max: 507833/s stddev: 1%
delete average: 494225/s min: 474705/s max: 518272/s stddev: 3%
read   average: 980386/s min: 942242/s max: 1028006/s stddev: 2%
```

Learnings

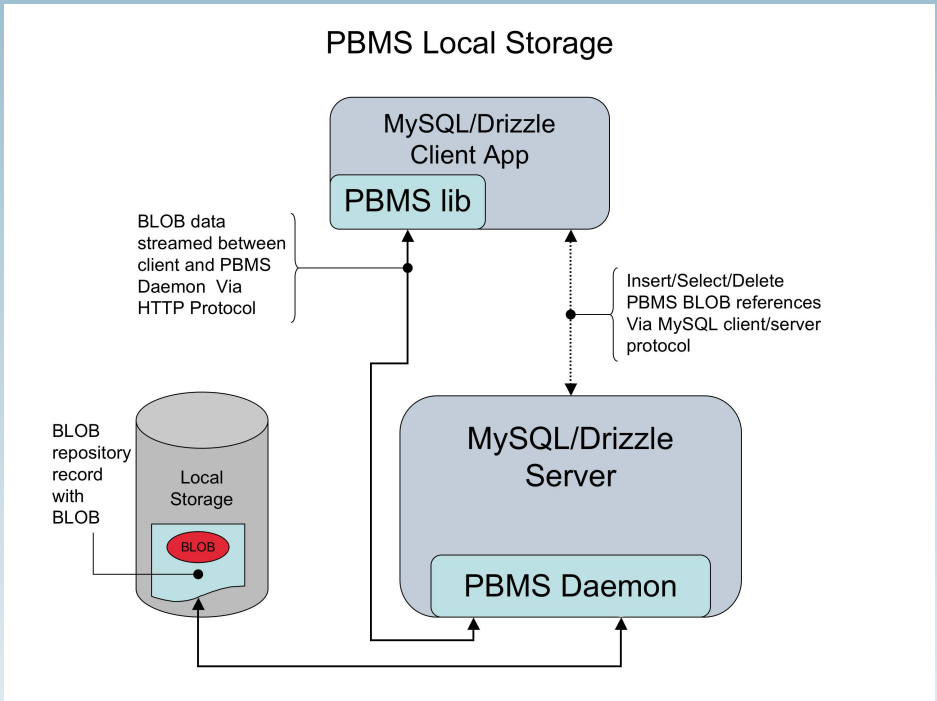
- Observations
  - CPU's not maxed out. "Somewhere" is potential !?!
  - When you overdo: CPU is maxed out and performance drops to 14%!
- The fakes:
  - Look at the parameters!
  - All other setups: I got worse throughput
  - IP instead of localhost: 89% throughput
- Learnings:
  - Do not trust benchmarks you did not fake yourself!

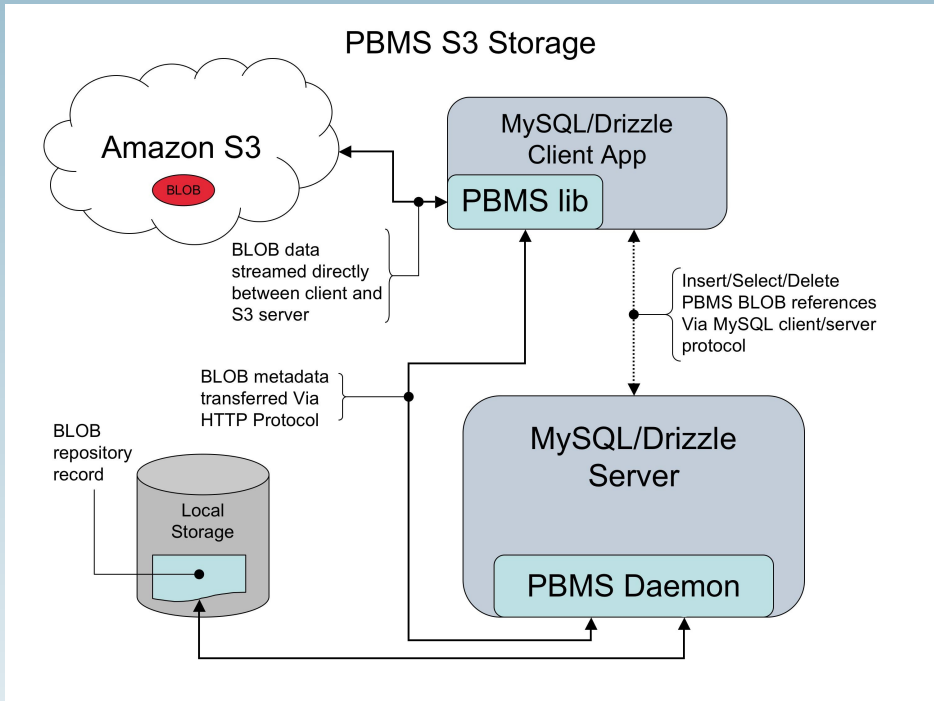
BLOB Streaming Project

- April 2008, Paul McCullagh: Introduction to the BLOB Streaming Project
- March 5 2010, Barry Leslie: Upload 1000+ BLOB's per second!



BLOB's local and in the cloud





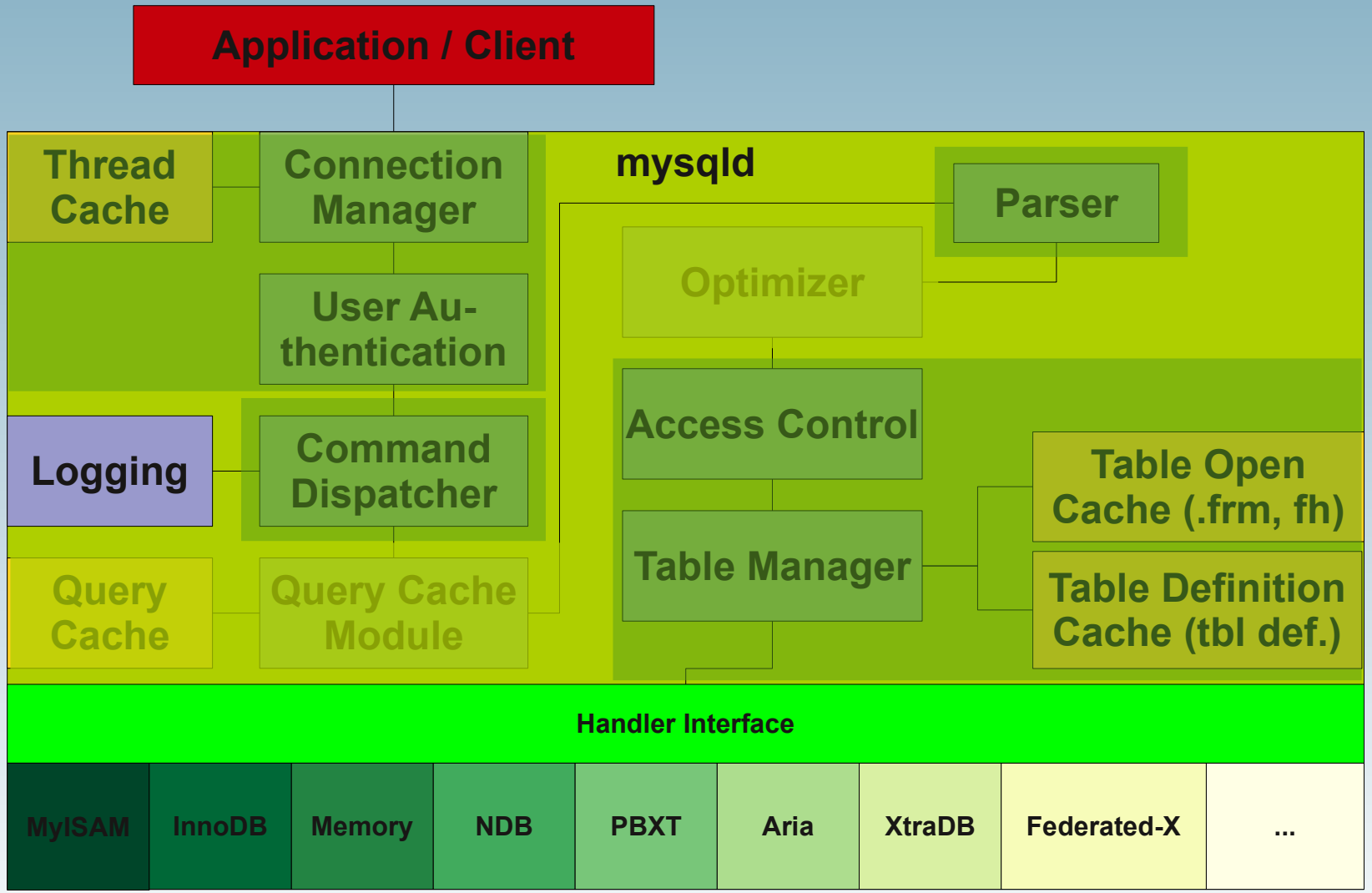
Advantages of BLOB's in the database

- old: RDBMS are not fast in storing BLOB's
  → do NOT store BLOB's in the databases
- new: With NoSQL technologies it becomes much better!
- With PBSE: atomic transactions → No “dangling” references.
- BLOB's in the normal database Backup !?!
- BLOB's can be replicated
- BLOB's in the DB will scale better. Most file systems perform poorly when the number of files exceeds 2 million ?

The Handler Interface

- October 2001, MySQL manual: A new HANDLER interface to MyISAM tables
- December 27, 2010, Stephane Varoqui: Using MySQL as a NoSQL: a story for exceeding 450'000 qps with MariaDB
- January 10, 2011, Stephane Varoqui: 20% to 50% improvement in MariaDB 5.3 Handler Interface using prepared statement

Skipping overhead with Handler Interface



HANDLER Example

```
# MySQL
# SELECT * FROM family;

HANDLER family OPEN;
HANDLER family
   READ `PRIMARY` = (id)
  WHERE id = 1;
HANDLER family CLOSE;


# With MariaDB 5.3

HANDLER family OPEN;
PREPARE stmt
   FROM 'HANDLER family
            READ `PRIMARY` = (id)
           WHERE id = ?';
set @id=1;
EXECUTE stmt USING @id;
DEALLOCATE PREPARE stmt;
HANDLER family CLOSE;

Use persistent connections!!!
```

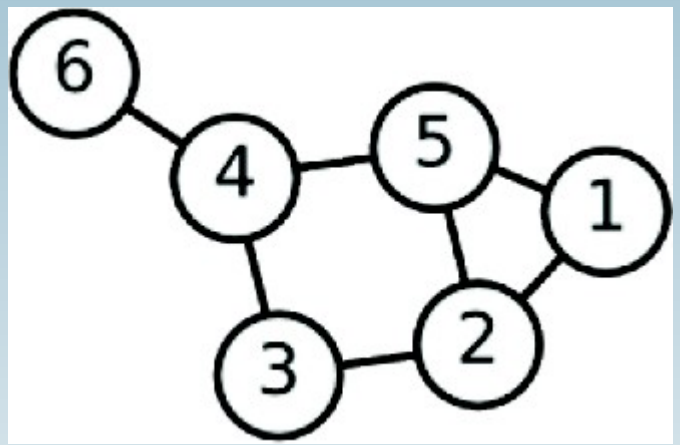
```
HANDLER tbl OPEN

HANDLER tbl READ idx (..., ..., …)
  WHERE ... LIMIT ...

HANDLER tbl READ idx FIRST
  WHERE ... LIMIT ...
HANDLER tbl READ idx NEXT
  WHERE ... LIMIT ...
HANDLER tbl READ idx PREV
  WHERE ... LIMIT ...
HANDLER tbl READ idx LAST
  WHERE ... LIMIT ...

HANDLER tbl READ FIRST
  WHERE ... LIMIT ...
HANDLER tbl READ NEXT
  WHERE ... LIMIT ...

HANDLER tbl CLOSE
```

Characteristics of the Handler Interface

- HANDLER is faster than SELECT:
  - Less parsing involved
  - No optimizer overhead
  - Less query-checking overhead
  - The table does not have to be locked between two handler requests
- No consistent look of the data (dirty reads are permitted)
- Some optimizations possible that SELECT does not allow
- Traverse the database in a manner that is difficult (or even impossible) to accomplish with SELECT

A Graph Storage Engine

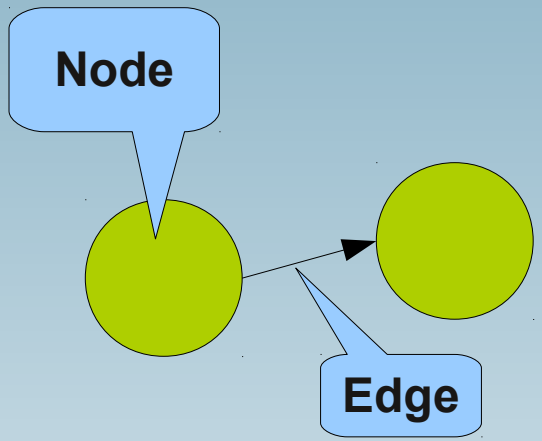
- **May 5, 2009, Arjen Lentz: OQGRAPH Computation Engine for MySQL, MariaDB & Drizzle**



- **It's included in MariaDB 5.1 ff.**
- **And available for MySQL 5.0 ff.**

How does it feel?

- It is similar to MEMORY SE (persistency, locking, trx)
- We talk in:
  - Node/Item/vertex and
  - Edge/connection/link
- Edges have a direction
- We can do networks and hierarchies
  - (family relations, friends of friends, fastest way from a to b)
- To talk to the OQGRAPH SE we use the latches (which algorithm to use)
- It is a computational engine not a storage engine!



Simple example: My family

```
INSERT INTO family VALUES
  (1, 'Grand-grand-ma')
, (2, 'Grand-ma')
, (3, 'Grand-uncle')
, (4, 'Grand-aunt')
, (5, 'Grand-pa')
, (6, 'Mother')
, (7, 'Uncle 1')
, (8, 'Uncle 2')
, (9, 'Father')
, (10, 'Me')
, (11, 'Sister');

INSERT INTO relation (origid, destid)
VALUES
  (1, 2), (1, 3), (1, 4)
, (2, 6), (2, 7), (2, 8)
, (5, 6), (5, 7), (5, 8)
, (6, 10), (6, 11)
, (9, 10), (9, 11);
```

```
SELECT f1.name AS parent, f2.name AS child
  FROM relation AS r
  JOIN family f1 ON f1.id = r.origid
  JOIN family f2 ON f2.id = r.destid;
```

parent	child
Grand-grand-ma	Grand-ma
Grand-grand-ma	Grand-uncle
Grand-grand-ma	Grand-aunt
Grand-ma	Mother
Grand-ma	Uncle 1
Grand-ma	Uncle 2
Grand-pa	Mother
Grand-pa	Uncle 1
Grand-pa	Uncle 2
Mother	Me
Mother	Sister
Father	Me
Father	Sister

Network queries

```
SELECT GROUP_CONCAT(f.name SEPARATOR ' -> ')
       AS path
  FROM relation AS r
  JOIN family AS f ON (r.linkid = f.id)
 WHERE latch = 1
   AND origid = 1
   AND destid = 10
ORDER BY seq;
```

```
+-----------------------------------------+
| path                                    |
+-----------------------------------------+
| Grand-grand-ma -> Grand-ma -> Mother -> Me |
+-----------------------------------------+
```

latch = 1: Find shortest path (Dijkstra)

```
SELECT r.weight, r.seq, f.name
  FROM relation AS r
  JOIN family AS f ON (r.linkid = f.id)
 WHERE r.latch = 2
   AND r.destid = 10;
```

```
+--------+------+----------------+
| weight | seq  | name           |
+--------+------+----------------+
|      3 |    6 | Grand-grand-ma |
|      2 |    5 | Grand-pa       |
|      2 |    4 | Grand-ma       |
|      1 |    3 | Father         |
|      1 |    2 | Mother         |
|      0 |    1 | Me             |
+--------+------+----------------+
```

latch = 2: Find originating nodes
(Breadth-first search)

Resume

- SQL is good for complex queries
- NoSQL is typically for simple queries
- Be careful with performance numbers!
- Architecture / coding becomes more complex
- You can gain better performance

- But it is interesting like hell!

Literature

- Using MySQL as a NoSQL - A story for exceeding 750,000 qps on a commodity server: http://yoshinorimatsunobu.blogspot.com/2010/10/using-mysql-as-nosql-story-for.html
- 950k reads per second on 1 datanode: http://jonasoreland.blogspot.com/2008/11/950k-reads-per-second-on-1-datanode.html
- Scalable BLOB Streaming Infrastructure for MySQL and Drizzle: http://www.blobstreaming.org/
- HandlerSocket: Why did our version not take off? http://pbxt.blogspot.com/2010/12/handlersocket-why-did-out-version-did.html
- Using MySQL as a NoSQL: a story for exceeding 450000 qps with MariaDB: http://varokism.blogspot.com/2010/12/using-mysql-as-nosql-story-for_27.html
- HANDLER Syntax: http://dev.mysql.com/doc/refman/5.5/en/handler.html
- GRAPH Computation Engine – Documentation: http://openquery.com/graph/doc

Q & A

Questions ?

Discussion?

We have some slots free to provide you personal consulting services...