
Performance Tuning of MySQL Cluster

October 2011

Johan Andersson

Severalnines AB

johan@severalnines.com

Cell +46 73 073 60 99

severalnines

Agenda



- Application design
- Identifying bottlenecks
- Tuning tricks

Facts



- A single query will never run as fast as on Innodb (served from RAM)
- Network latency is a issue
- More data nodes does not speed up query execution time.

Application Design



- Define the most typical Use Cases
 - List all my friends, session management etc etc.
 - Optimize everything for the typical use case
- Engineer schema to cater for the Use Cases
- Keep it simple
 - Complex access patterns does not scale
 - Simple access patterns do (Primary key and Partitioned Index Scans)
- Note! There is no parameter in config.ini that affects performance – only availability.
 - Everything is about the Schema and the Queries.
 - Tune the mysql servers (sort buffers etc) as you would for innodb.

Simple Access

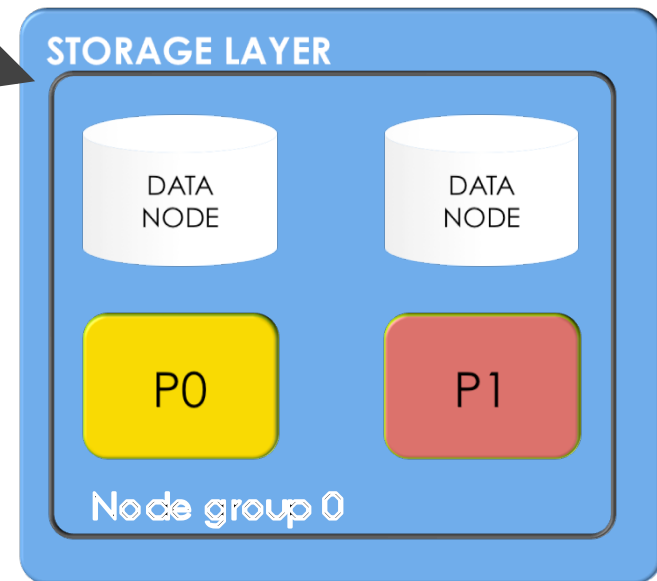


- PRIMARY KEY lookups are HASH lookup $O(1)$
- INDEX searches a T-tree and takes $O(\log n)$ time.
- In 7.2 JOINS are ok, but in 7.1 you should try to avoid them.

Setup



NETWORK!



<u>subid</u>	<u>data</u>	
1	A	Partition 0
3	B	
2	C	Partition 1
4	D	

subscriber

Identifying Bottlenecks

- A lot of CPU is used on the data nodes
 - Probably a lot of large index scans and full table scans are used.
 - A lot of CPU is used on the mysql servers
 - Probably a lot of GROUP BY/DISTINCT or aggregate functions.
 - Hardly no CPU is used on either mysql or data nodes
 - Probably low load
 - Time is spent on network (a lot of “ping pong” to satisfy a request).
 - System is running slow in general
 - Disks (io util), queries, swap (should never happen)
-

Need To Add Data Nodes?

- (adding mysql servers is easy)
 - `top -Hd1`
 - Is any of data nodes threads at 100%?
 - Yes: add more data nodes (online)
 - No: do nothing
-

Detecting Problems

- ▣ Here is a standard method for how to attack the problem.
- ▣ Performance tuning is a never-ending loop:
BEGIN
 - Capture information – e.g, slow query log
 - Change long_query_time if needed
 - EXPLAIN the queries
 - What indexes are used?
 - Are tables JOINed in the correct order (small to big)
 - Re-run the optimized typical use cases using `bencher/`
`mysqlslap`
GOTO BEGIN;
END;
- ▣ Never tune unless you can measure and test!
- ▣ Don't optimize unless you have a problem!

Enable Logging

- ❑ Slow query log
 - ❑ `set global slow_query_log=1;`
 - ❑ `set global long_query_time=0.01;`
 - ❑ `set global log_queries_not_using_indexes=1;`
- ❑ General log (if you don't get enough info in the Slow Query Log)
 - ❑ Activate for a very short period of time (30-60seconds) – intrusive
 - ❑ Can fill up disk very fast – make sure you turn it off.
 - ❑ `set global general_log=1;`
- ❑ Use Severalnines ClusterControl
 - ❑ Includes a Cluster-wide Query Monitor.
 - ❑ Query frequency, EXPLAINS, lock time etc.
 - ❑ Performance Monitor and Manager.

Query Tuning

```
mysql> explain select * from t1 where a=2 and ts='2011-10-05 15:32:11';
```

```
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| id | select_type | table | type | possible_keys | key | key_len | ref | rows |
Extra |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
| 1 | SIMPLE | t1 | ref | idx_t1_a,idx_t1_a_ts | idx_t1_a | 9 | const | 10 | Using
where |
+----+-----+-----+-----+-----+-----+-----+-----+-----+
```

□ Use FORCE INDEX(..) ...

```
mysql> explain select * from t1 FORCE INDEX (idx_t1_a_ts) where a=2 and ts='2011-10-05
15:32:11';
```

```
+ | 1 | SIMPLE | t1 | ref | idx_t1_a_ts | idx_t1_a_ts | 13 | const,const | 10 | Using
where |
```

1 row in set (0.00 sec)

□ ..to ensure the correct index is picked!

□ The difference can be 1 record read instead of any number of records!

Partition Pruning / shard key

- By default, all index scans hit all data nodes – good if big result set – you want as many CPUs as possible to help you.
- For smaller result sets (~a couple of hundred records) Partition Pruning is key for scalability.
 - User-defined partitioning can help to improve equality index scans on part of a primary key.
 - ```
CREATE TABLE t1 (uid,
 fid,
 somedata,
 PRIMARY KEY(uid, fid))
 PARTITION BY KEY(userid);
```
  - All data belonging to a particular `uid` will be on the same partition. Great locality!
  - ```
select * from user where uid=1;
```

 - Only one data node will be scanned (no matter how many nodes you have)

Partition Pruning

```
mysql> show global status like 'ndb_pruned_scan_count';
```

Variable_name	Value
Ndb_pruned_scan_count	0

```
CREATE TABLE t1( ... ) PARTITION BY KEY (userid);
```

An run query, and verify it works:

```
select * from user where userid=1;
```

```
mysql> show global status like 'ndb_pruned_scan_count';
```

Variable_name	Value
Ndb_pruned_scan_count	1

Denormalize Tables Where Possible

- Tables sharing the same PRIMARY KEY can be denormalized.
 - Table T1: <UID, SOME_DATA>
 - Table T2: <UID, SOME_OTHER_DATA>
 - SELECT * from T1,T2 WHERE T1.UID=T2.UID and T2.UID=1 requires two roundtrips.
 - Starting with MySQL Cluster 7.2 only one roundtrip is needed,.
- Denormalize
 - Table T12: <UID,SOME_DATA, SOME_OTHER_DATA>
- Improvement: 2X in throughput

Data Types

BLOBs/TEXTs vs VARBINARY/VARCHAR

- BLOB/TEXT columns are stored in an external hidden table.
 - First 255B are stored inline in main table
 - Reading a BLOB/TEXT requires two reads
 - One for reading the Main table + reading from hidden table
- Change to VARBINARY/VARCHAR if:
 - Your BLOB/TEXTs can fit within an 8052B record
 - (record size is currently 8052 Bytes)
 - Reading/writing VARCHAR/VARBINARY is less expensive

Note 1: BLOB/TEXT are also more expensive in Innodb as BLOB/TEXT data is not inlined with the table. Thus, two disk seeks are needed to read a BLOB.

Note 2: Store images, movies etc outside the database on the filesystem.

Query Tuning

- ❑ Don't trust the OPTIMIZER in MySQL Cluster 7.1 and earlier
 - ❑ Statistics gathering is non-existing
 - ❑ Optimizer thinks there are only 10 rows to examine in each table!
- ❑ You have to do a lot of (up to and including 7.1)
 - ❑ FORCE INDEX / STRAIGH_JOIN to get queries run the way you want
- ❑ Classic example: if you have two similar indexes:
 - ❑ index(a)
 - ❑ index(a,ts)

on the following table

```
CREATE TABLE `t1` (  
  `id` int(11) NOT NULL AUTO_INCREMENT,  
  `a` bigint(20) DEFAULT NULL,  
  `ts` timestamp NOT NULL DEFAULT CURRENT_TIMESTAMP,  
  PRIMARY KEY (`id`),  
  KEY `idx_t1_a` (`a`),  
  KEY `idx_t1_a_ts` (`a`,`ts`)) ENGINE=ndbcluster DEFAULT CHARSET=latin1
```

Ndb_cluster_connection_pool

- ❑ Problem:
 - ❑ A Sendbuffer on the connection between mysqld and the data nodes is protected by a Mutex.
 - ❑ Connection threads in MySQL must acquire Mutex and the put data in SendBuffer.
 - ❑ Many threads gives more contention on the mutex
 - ❑ Must scale out with *many* MySQL Servers.

- ❑ Workaround:
 - ❑ Ndb_cluster_connection_pool (in my.cnf) creates more connections from one mysqld to the data nodes
 - ❑ Threads load balance on the connections gives less contention on mutex which in turn gives increased scalability
 - ❑ Less MySQL Servers needed to drive load!
 - ❑ www.severalnines.com/cluster-configurator allows you to specify the connection pool.
 - ❑ >70 % improvement.

Ndb_cluster_connection_pool

- Gives at least 70% better performance and a MySQL Server that can scale beyond four database connections.
- Set `Ndb_cluster_connection_pool=2x<CPU cores>`
 - It is a good starting point
- One free `[mysqld]` slot is required in `config.ini` for each `Ndb_cluster_connection`.
 - 4 mysql servers, each with `Ndb_cluster_connection_pool=8` requires 32 `[mysqld]` in `config.ini`

several**nines**

Q&A





Thank you for your time!

johan@severalnines.com