MySQL
State of the Dolphin

Geir Høydalsvik
MySQL Engineering

Percona Live Europe
Amsterdam, October 1, 2019
Safe harbor statement

The following 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, timing, and pricing of any features or functionality described for Oracle’s products may change and remains at the sole discretion of Oracle Corporation.
MySQL 8 is Great

NoSQL + SQL = MySQL
MySQL 8.0: One Giant Leap for SQL

“This is a landmark release as MySQL eventually evolved beyond SQL-92 and the purely relational dogma. Among a few other standard SQL features, MySQL now supports window functions (over) and common table expressions (with). Without a doubt, these are the two most important post-SQL-92 features.”
MySQL 8.0: Light Years from 5.x

@MySQL 8 is light years away from 5.x versions. You now have:

- CTE and Recursive CTE
- Window Functions
- SKIP LOCKED, NO WAIT
- Hash Joins (Coming in 8.0.18)
- Explain Analyze giving you the Actual Plan (Coming in 8.0.18)
MySQL 8.0: Fast Adoption

April 2018
MySQL 8.0
MySQL 5.7,66%
MySQL 5.6,25%

August 2019
MySQL 8.0,26%
MySQL 5.7,47%
MySQL 5.6,24%
MySQL 8.0: Releases in 2019

8.0.14 (Jan 2019)
  LATERAL DERIVED TABLES
  PARALLEL READ OF INDEX
  CONSISTENCY LEVELS (GR)

8.0.16 (Apr 2019)
  CHECK CONSTRAINTS
  AUTO-UPGRADE
  AUTO-REJOIN (GR)

8.0.17 (Jul 2019)
  CLONE
  JSON ARRAY INDEXES
  JSON SCHEMA

8.0.18 (Oct 2019)
  HASH JOIN
  EXPLAIN ANALYZE
  …
MySQL 8.0: Hash Join

- Typically faster than nested loop for large result sets
- In-memory if possible
- Spill to disk if necessary
- Used for inner equi-joins
  - Extendable to outer, semi and anti joins
- Replaces Block Nested Loop in query plans
- Hints to force use of hash join or nested loop
MySQL 8.0: Hash Join Performance

- BNL compared to hash join
- Force BNL/hash join in DBT-3/TPC-H
  DBT-3/TPC-H without indexes
  Optimizer selects BNL
  Automatic conversion to hash join
- Hash join is much faster than BNL
- Can't expect same improvement when indexes are available
MySQL 8.0: EXPLAIN ANALYZE

- Instruments and executes the query
  - Estimated cost
  - Actual execution statistics
    - Time to return first row
    - Time to return all rows
    - Number of rows returned
    - Number of loops
- Uses the new tree output format also available in EXPLAIN
MySQL 8.0: The complete list of new features

https://mysqlserverteam.com/the-complete-list-of-new-features-in-mysql-8-0/
Meet Ada

Hello!

- Meet Ada, the DevOps
- Ada is smart
- Ada is using the MySQL Shell
MySQL Shell: Modern

- Colorful Prompt Themes
- Autocompletion
- Syntax Highlighting
- Context Sensitive Help
- Command History
- Pager, less/more
- Output Formats
MySQL Shell: Flexible

- SQL, JavaScript, Python
- Interactive & Batch
- SQL Client
- Document Store
- InnoDB Cluster Admin
MySQL Shell : Extendible

- **Utilities**
  - upgradeChecker()
  - importJSON()
  - importTable()

- **Reporting Framework**
  - \show \watch

- **User Defined Plugins**
  - JS or Python

I adapted it to my prod environment!
MySQL CLONE

WHY IS CLONE SUCH A BIG DEAL?

• Puts the power of fast instance provisioning into the hands of everybody
• Reduces the complex provisioning procedure to a few simple steps
• Even happens automatically when needed when using InnoDB Cluster
• Can be done fully remotely

NEW IN 8.0.17

Ada

CLONE makes my life easy!
MySQL CLONE Directly from SQL

Example:
Provision a *new slave* (RECIPIENT)
from an *existing master* (DONOR)

Are you ready?
MySQL CLONE Setup the **DONOR**

```sql
mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";
mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";
mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
```
MySQL CLONE Setup the **RECIPIENT**

```sql
mysql> INSTALL PLUGIN CLONE SONAME "mysql_clone.so";
mysql> SET GLOBAL clone_valid_donor_list = "donor.host.com:3306";
mysql> CREATE USER clone_user IDENTIFIED BY "clone_password";
mysql> GRANT BACKUP_ADMIN ON *.* to clone_user;
```
MySQL Connect to **RECIPIENT** and execute CLONE SQL statement

```
mysql> CLONE INSTANCE
    -> FROM clone_user@donor.host.com:3306
    -> IDENTIFIED BY "clone_password";
```
MySQL CLONE Check Status

mysql> select STATE, ... > from performance_schema.clone_status;

+-------------+---------------------+------------+
| STATE       | START TIME          | DURATION   |
+-------------+---------------------+------------+
| In Progress | 2019-07-17 17:23:26 | 4.84 m     |
+-------------+---------------------+------------+
**MySQL CLONE Check Progress**

```sql
mysql> select STATE, ...
    > from performance_schema.clone_progress;
```

<table>
<thead>
<tr>
<th>STAGE</th>
<th>STATE</th>
<th>START TIME</th>
<th>DURATION</th>
<th>Estimate</th>
<th>Done(%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>DROP DATA</td>
<td>Completed</td>
<td>17:23:26</td>
<td>790.86 ms</td>
<td>0 MB</td>
<td>100%</td>
</tr>
<tr>
<td>FILE COPY</td>
<td>Completed</td>
<td>17:23:27</td>
<td>10.33 m</td>
<td>94,729 MB</td>
<td>100%</td>
</tr>
<tr>
<td>PAGE COPY</td>
<td>Completed</td>
<td>17:33:47</td>
<td>15.91 s</td>
<td>11,885 MB</td>
<td>100%</td>
</tr>
<tr>
<td>REDO COPY</td>
<td>Completed</td>
<td>17:34:03</td>
<td>1.07 s</td>
<td>293 MB</td>
<td>100%</td>
</tr>
<tr>
<td>FILE SYNC</td>
<td>In Progress</td>
<td>17:34:04</td>
<td>51.68 s</td>
<td>0 MB</td>
<td>0%</td>
</tr>
<tr>
<td>RESTART</td>
<td>Not Started</td>
<td>NULL</td>
<td>NULL</td>
<td>0 MB</td>
<td>0%</td>
</tr>
<tr>
<td>RECOVERY</td>
<td>Not Started</td>
<td>NULL</td>
<td>NULL</td>
<td>0 MB</td>
<td>0%</td>
</tr>
</tbody>
</table>
MySQL InnoDB Cluster

- configureInstance()
- createCluster()
- addInstance()
- removeInstance()
- rejoinInstance()

And here CLONE is fully automated!
Pre-requisites: Install and start MySQL on 3 servers
mysql-js> dba.configureInstance('clusteradmin@mysql1')

binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
mysql-js>dba.configureInstance('clusteradmin@mysql2')

```
binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
```

Configure instance

MySQL Shell
mysql-js>dba.configureInstance('clusteradmin@mysql3')

binlog_checksum = NONE
enforce_gtid_consistency = ON
gtid_mode=ON
server_id= <unique ID>
mysql-js> cluster=dba.createCluster('PLEU2019')
mysql-js> cluster.status()
{
    "clusterName": "PLEU2019",
    "defaultReplicaSet": {
        "name": "default",
        "primary": "mysql1:3306",
        "ssl": "REQUIRED",
        "status": "OK_NOT_TOLERANT",
        "statusText": "Cluster is NOT tolerant to any failures."
    },
    "topology": {
        ...
    },
    "topologyMode": "Single Primary"
},
"groupInformationSourceMember": "mysql1:3306"
}
mysql-js> cluster.addInstance('clusteradmin@mysql2')

Add instance

MySQL Shell

PLEU201
9

CLONE
mysql-js> cluster.addInstance('clusteradmin@mysql3')
MySQL Shell : Add Instance with CLONE Progress Reporting

A new instance will be added to the InnoDB cluster. Depending on the amount of data on the cluster this might take from a few seconds to several hours.

Adding instance to the cluster...

Monitoring recovery process of the new cluster member. Press AC to stop monitoring and let it continue in background.

Clone based state recovery is now in progress.

NOTE: A server restart is expected to happen as part of the clone process. If the server does not support the RESTART command or does not come back after a while, you may need to manually start it back.

* Waiting for clone to finish...

NOTE: 10.0.0.3:3306 is being cloned from 10.0.0.2:3306

** Stage DROP DATA: Completed

** Clone Transfer

FILE COPY 100% Completed
PAGE COPY 100% Completed
REDO COPY 100% Completed

** Stage RECOVERY:

NOTE: 10.0.0.3:3306 is shutting down...

* Waiting for server restart... ready
* 10.0.0.3:3306 has restarted, waiting for clone to finish...
* Clone process has finished: 60.63 MB transferred in about 1 second (-30.63 MB/s)

Incremental distributed state recovery is now in progress.

* Waiting for distributed recovery to finish...

NOTE: '10.0.0.3:3306' is being recovered from '10.0.0.2:3306'
* Distributed recovery has finished

The instance '10.0.0.3' was successfully added to the cluster.

MySQL Shell

PLEU201

FINISHED in one second
mysql-js> cluster.status()

{ "clusterName": "PLEU2019",
  "defaultReplicaSet": {
    "name": "default",
    "primary": "mysql1:3306",
    "ssl": "REQUIRED",
    "status": "OK",
    "statusText": "Cluster is ONLINE and can tolerate up to ONE failure.",
    "topology": {
      ...CUT...
    },
    "topologyMode": "Single Primary"
  },
  "groupInformationSourceMember": "mysql1:3306"
}

MySQL Shell

Status?
FAULT
TOLERANT

PLEU201
9
# mysqlrouter --bootstrap clusteradmin@mysql1 --user=routeradmin
# systemctl start mysqlrouter

Starting mysqlrouter and adding r/w load

MySQL Shell

R/W Load

MySQL Router

PLEU201
9
mysql1# kill -9 $(pidof mysqld)

Testing...
Killing primary mysqld...

MySQL Shell
OK, mysql1 left the group and mysql2 became the new primary.
MySQL: How to regain Fault Tolerance?

1. **Automatic, self healing**
   - Binlog (if GTID available in group)
   - CLONE (otherwise)

2. **Manual fix**
   - Self healing failed, e.g. network failure
   - `rejoinInstance()`
   - CLONE

3. **Replace with new instance (permanent failure)**
   - `removeInstance()`
   - `addInstance()`
   - CLONE
mysql-js> cluster.rejoinInstance('clusteradmin@mysql1')

rejoinInstance()
MySQL: Fault Tolerant Again

THANK YOU!

MySQL Shell

R/W Load

MySQL Router

PLEU201

9
Meet the MySQL Team at the Conference

Sunny Bains
Luis Soares
Kenny Gryp
Norvald Ryeng
Frédéric Descamps
Dimitri Kravtchuk
Ståle Deraas
Pedro Gomes
Georgi Kodinov
Geir Høydalsvik
MySQL Talks at the Conference (Tuesday)

11:00 AM - 11:50 AM Room 07 Dimitri Kravtchuk
MySQL 8.0 Performance: Scalability & Benchmarks

5:00 PM - 5:50 PM Room 07 Sunny Bains
InnoDB Management and Scalability Improvements in MySQL 8.0

5:00 PM - 5:50 PM Room B Pedro Gomez, Frederick Descamps
MySQL InnoDB Cluster - Advanced Configuration & Operation

6:00 PM - 6:50 PM Room B Norvald Ryeng
MySQL 8.0.18: Latest Updates
MySQL Talks at the Conference (Wednesday)

10:00 AM - 10:25 AM Room 10 Frederick Descamps
MySQL Shell: The Best DBA Tool? Extend the Shell With the New Extension Infrastructure

1:30 PM - 2:20 PM Room B Luis Soares
MySQL 8.0: The New Replication Features

3:30 PM - 3:50 PM Room 09 Norvald Ryeng
JSON Array Indexes in MySQL

2:30 PM - 3:20 PM Room 10 Staale Deraas
Automatic Upgrade and New Error Logging in MySQL 8.0

2:30 PM - 3:20 PM Room 07 Georgi Kodinov
How to Instrument Your Code in Performance Schema
Join us on MySQL Community Slack

https://lefred.be/mysql-community-on-slack/

We have 3 nodes A, B, C. A is primary R/W and at 9:00 AM A went down and B took over and at 11 AM B and C went down. Last backup was from 11:45 PM from last night.

In this scenario, we need to merge the writes that happened on A and B to restore until 11 AM.

lefred 8:43 PM
You need to restore backup and replay binlogs from B or C.

No because B has the writes of A.

When A went down, it doesn't have committed anything that B or C do not have.
Follow us on Social Media

https://www.facebook.com/mysql

https://twitter.com/mysql

https://www.linkedin.com/company/mysql