Percona XtraDB Cluster

- Deadlock Monitoring mit pt-deadlock-logger
- Vorbereitung zur Migration auf Percona XtraDB Cluster
- wsrep Status überwachen
- Percona APT Repository einrichten
- InnoDB Tablespace Cleanup mit pt-online-schema-change

Deadlock Monitoring mit pt-deadlock-logger

Unit-File anlegen:

```
[Unit]
Description=Start the Percona Toolkit Deadlock Logger
After=network.target
After=syslog.target
After=mysql.service

[Service]
Type=simple
StartLimitInterval=5
StartLimitBurst=10
#Environment=PTDEBUG=1
ExecStart=/usr/bin/pt-deadlock-logger --user root --defaults-file /root/.my.cnf --create-dest-table --dest D=percona_schema,t=deadlocks h=dbcluster1-ha1.mydomain.com
Restart=always
RestartSec=120

[Install]
WantedBy=multi-user.target
```

Aktivieren und Starten:

```
systemctl daemon-reload
systemctl enable pt-deadlock-logger.service
systemctl start pt-deadlock-logger.service
```

Alle 30 Sekunden wird nun geprüft, ob im SHOW INNODB STATUS ein Deadlock aufgelaufen ist. Falls ja, wird dieser in die angegebene Tabelle geloggt.

Vorbereitung zur Migration auf Percona XtraDB Cluster

Um eine vorhandene MySQL Datenbank zu Percona XtraDB Cluster zu migrieren müssen einige Voraussetzungen erfüllt sein (siehe hierzu auch die Limitations-Seite bei Percona.

- Alle Tabellen müssen InnoDB Engine sein
- Jede Tabelle sollte einen PRIMARY KEY haben

Alle Tabellen finden, die nicht die InnoDB Engine verwenden

```
USE information_schema;
SELECT CONCAT(TABLE_SCHEMA, '.', TABLE_NAME) AS TABLES_NOT_INNODB
FROM TABLES
WHERE ENGINE != 'InnoDB' AND
      TABLE_SCHEMA NOT IN('mysql','information_schema','performance_schema');
```

Tabellen ohne PRIMARY KEY finden

Dies verhindert folgenden Fehler beim Import eines Dumps:

```
[ERROR] WSREP: Percona-XtraDB-Cluster prohibits use of DML
command on a table (db.taxonomy_index) without an explicit primary key
with pxc_strict_mode = ENFORCING or MASTER

ERROR 1105 (HY000) at line 20655: Percona-XtraDB-Cluster prohibits use
of DML command on a table (db.taxonomy_index) without an explicit
primary key with pxc_strict_mode = ENFORCING or MASTER
```

```
USE information_schema;
SELECT CONCAT(TABLES.TABLE_SCHEMA, '.', TABLES.TABLE_NAME) AS TABLES_WITHOUT_PRIMARY_KEY
FROM TABLES
LEFT JOIN KEY_COLUMN_USAGE AS c
ON (TABLES.TABLE_NAME = c.TABLE_NAME AND
    c.CONSTRAINT_SCHEMA = TABLES.TABLE_SCHEMA AND
    c.CONSTRAINT_NAME = 'PRIMARY')
WHERE TABLES.TABLE_SCHEMA <> 'information_schema' AND
```

```
    TABLES.TABLE_SCHEMA <> 'performance_schema' AND
    TABLES.TABLE_SCHEMA <> 'mysql' AND
    c.CONSTRAINT_NAME IS NULL;
```

Über ALTER TABLE läßt sich diesen Tabellen ein PRIMARY KEY zuweisen. Vorher sollte allerdings sichergestellt sein, dass die Applikation auch damit zurecht kommt.

PRIMARY KEY hinzufügen:

```
USE mydb;
ALTER TABLE address ADD id INT PRIMARY KEY AUTO_INCREMENT;
```

wsrep Status überwachen

Als Script auf einer Cluster-Node ablegen. Gibt dauerhaft den aktuellen Wert der wsrep-Statusvariablen aus:

```
#!/bin/sh
watch -d -n1 -x mysql -B -e "SHOW STATUS WHERE variable_name ='wsrep_local_state_comment' OR
variable_name ='wsrep_cluster_size' OR variable_name ='wsrep_incoming_addresses' OR
variable_name ='wsrep_cluster_status' OR variable_name ='wsrep_connected' OR variable_name
='wsrep_ready' OR variable_name ='wsrep_local_state_uuid' OR variable_name
='wsrep_cluster_state_uuid' OR variable_name ='wsrep_last_committed' OR variable_name
='wsrep_received' OR variable_name ='wsrep_received_bytes';"
```

Alle wsrep%-Variablen ausgeben:

```
#!/bin/sh
watch -d -n1 -x mysql -B -e "SHOW STATUS WHERE variable_name like 'wsrep%'";
```

Percona APT Repository einrichten

Installationspaket für das Repository holen und installieren

```
apt update
apt install gnupg2
wget https://repo.percona.com/apt/percona-release_latest.$(lsb_release -sc)_all.deb
dpkg -i percona-release_latest.$(lsb_release -sc)_all.deb
apt update
```

Percona Software installieren

```
apt install percona-toolkit xtrabackup percona-server-server
```

InnoDB Tablespace Cleanup mit pt-online-schema-change

Mit der Zeit wird das Datenfile (.ibd) einer InnoDB Tabelle durch das Hinzufügen und Löschen von Daten immer größer. Obwohl die Tabelle eigentlich viel kleiner ist wird der benötigte Plattenplatz nicht immer wieder freigegeben.

Die Voraussetzung für die folgenden Tipps ist, dass `innodb_file_per_table=on` gesetzt und beim PXC die `wsrep_OSU_method=TOI` eingestellt ist.

Bei Tabellen ohne große Last kann das schnell und einfach mit einem `mysqloptimize -A` behoben werden (alle Datenbanken und Tabellen werden "optimiert"). Dabei wird die Tabelle neu erstellt.

Handelt es sich um eine sehr aktive Tabelle und soll der Impact dieser Aktion nicht so groß sein (und zum Beispiel einen Percona Cluster sperren), dann kann man das Tool pt-online-schema-change aus dem Percona Toolkit nutzen. Das Tool kann die Tabellenstruktur ändern ohne Lese-/Schreibzugriffe zu blockieren. Normalerweise wird es verwendet, um z.B. ein neues Feld oder einen Index hinzuzufügen.

Ein `ALTER TABLE ENGINE=InnoDB` bewirkt hier aber das gleiche wie ein `mysqloptimize`. Ich habe mir dafür ein Script geschrieben, das alle Tabellen einer Datenbank durcharbeitet. Einziges Argument ist der Datenbankname.

```
#!/bin/bash

DB=${1}

TABLES=$(mysql -N -B ${DB} -e "show tables;")

for table in ${TABLES}; do
  echo "---------------- Processing ${DB} -- ${table} ----------------"
  pt-online-schema-change --execute --alter "ENGINE=InnoDB" --progress=time,5
D=${DB},t=${table}
  echo
done
```

Das sieht dann so aus:

```
# ./optimize-table-pt-osc.sh my_db
---------------- Processing my_db -- AuthAssignment ----------------
No slaves found.  See --recursion-method if host dbserver-01-a has slaves.
Not checking slave lag because no slaves were found and --check-slave-lag was not specified.
Operation, tries, wait:
  analyze_table, 10, 1
```

```
  copy_rows, 10, 0.25
  create_triggers, 10, 1
  drop_triggers, 10, 1
  swap_tables, 10, 1
  update_foreign_keys, 10, 1
Altering `my_db`.`AuthAssignment`...
Creating new table...
Created new table my_db._AuthAssignment_new OK.
Altering new table...
Altered `my_db`.`_AuthAssignment_new` OK.
2023-08-31T10:49:15 Creating triggers...
2023-08-31T10:49:15 Created triggers OK.
2023-08-31T10:49:15 Copying approximately 157 rows...
2023-08-31T10:49:15 Copied rows OK.
2023-08-31T10:49:15 Analyzing new table...
2023-08-31T10:49:15 Swapping tables...
2023-08-31T10:49:15 Swapped original and new tables OK.
2023-08-31T10:49:15 Dropping old table...
2023-08-31T10:49:15 Dropped old table `my_db`.`_AuthAssignment_old` OK.
2023-08-31T10:49:15 Dropping triggers...
2023-08-31T10:49:15 Dropped triggers OK.
Successfully altered `my_db`.`AuthAssignment`.
```

Bei einem Kunden konnte ich so den benötigten Platz von 450 GB auf 90 GB verringern.