Global Big Data Conference

www.globalbigdataconference.com

Twitter : @bigdataconf

#GlobalNoSQL

Workshop

MySQL and Percona XtraDB Cluster

Day 2

Agenda

- MySQL and PXC Administration
  - Helper Tools (Percona Toolkit)
  - MySQL Server General Settings
  - InnoDB Settings
  - Galera Settings
  - OS Settings
- MySQL Security
  - User Management
  - Host Access Management
- High Availability and common tasks with PXC
  - Schema changes
  - Bonus if time allows: Mix setup with Async Replication

MySQL and PXC Administration

Helper Tools (Percona Toolkit)

- **pt-summary**: Summarizes the status and configuration of a Linux server
- **pt-mysql-summary**: Summarizes the status and configuration of a MySQL database server
- **pt-stalk**: Collect forensic data about MySQL. Can be configured to wait for a trigger condition to occur, then collects data to help diagnose problems
- Example:

```
PTDEST=/tmp/pt/collected/$(hostname)/
mkdir -p "${PTDEST}/samples";

pt-summary > "${PTDEST}/pt-summary.out";

pt-mysql-summary --save-samples="${PTDEST}/samples" -- --user=root
--password=<mysql-root-password>  > "${PTDEST}/pt-mysql-summary.out";

pt-stalk --no-stalk --iterations=2 --sleep=30 --dest="${PTDEST}" -- --user=root
--password=<mysql-root-pass>;
```

Helper Tools (Percona Toolkit)

- **pt-mext**: Look at many samples of MySQL SHOW GLOBAL STATUS side-by-side
- Example:

```
PTDEST=/tmp/pt/collected/$(hostname)/
cd /tmp/pt;

pt-mext -r -- cat <file_name-mysqladmin> > <file_name-mysqladmin>.mext
```

General advice

We need to make sure we understand the nature of PXC/Galera clusters. Clusters based on Galera **are not designed to scale writes**. In fact, sending writes to multiple nodes for the same tables will end up causing certification conflicts that will reflect as deadlocks for the application, adding huge overhead.

Also, since **replication-certification process is single-threaded process** we need to keep the transactions short, COMMITTING them as soon as we are done with an atomic unit of work. Long transactions can stall writes on the other nodes which can be seen in the writer node showing transactions in wsrep pre-commit stage, which means the other nodes are doing certification for a large transaction or that the node is suffering performance problems of some sort, for example swap, full disk, abusively large reads, etc.

General advice

Note that:

- The list of parameters here isn't exhaustive and there are many other that can be changed;
- There is no magic formula for configuration tuning and both hardware and the specific workload plays a lot when tuning parameters;
- Most of the cases we get a lot more performance improvement with a good database logical design than hardware and parameters tuning;
- The use of appropriated column types, indexes and relationship are a much better tuning;
- All that said, **any recommendation here need to be experimented in a test environment before going to production** as they might be adjusted to the specific environment and in some cases **they can cause more harm than good if not properly understood**!

MySQL Server General Settings

- **max_connections**: The maximum permitted number of simultaneous client connections
  - Dynamic: YES
  - Default Value: 151
  - Recommended: Keep it small as possible to avoid CPU contention (context switch, thread concurrency, etc);
- **max_allowed_packet**: The maximum size of MySQL communication buffer
  - Dynamic: YES
  - Default Value: V.5.7: 4M | V.8: 64M
  - Recommended: Highly depends on application needs. Max is 1G
- **skip_name_resolve**: Whether to resolve host names when checking client connections
  - Dynamic: NO
  - Default Value: FALSE
  - Recommended: TRUE

MySQL Server General Settings

- **tmp_table_size**: The maximum size of internal in-memory temporary tables. Does not apply to user-created MEMORY tables
  - Dynamic: YES
  - Default Value: 16M
  - Recommended: Increase gradually but try to keep small and use indexes
- **max_heap_table_size**: The maximum size to which user-created MEMORY tables are permitted to grow
  - Dynamic: YES
  - Default Value: 16M
  - Recommended: Try to keep smaller but will highly depend on application needs but keep in mind it increase memory consumption

PS: The real limit of in-memory temporary tables is the smaller of tmp_table_size and max_heap_table_size. When an in-memory temporary table exceeds the limit, MySQL automatically converts it to an on-disk temporary table!

MySQL Server General Settings

- **thread_cache_size**: When a client disconnects, the client's threads are put in a thread cache. This defines its size
  - Dynamic: YES
  - Default Value: -1 (autosizing)
  - Recommended: If the server has hundreds of connections per second we should set thread_cache_size high enough so that most new connections use cached threads. Usually (max_connections / 10) is good enough
- **query_cache_size**: The amount of memory allocated for caching query results
  - Dynamic: YES
  - Default Value: 1M
  - Recommended: 0
- **query_cache_type**: Set the query cache type
  - Dynamic: YES
  - Default Value: 0
  - Recommended: 0

MySQL Server General Settings

- **table_open_cache**: The number of open tables for all threads. Increasing this value increases the number of file descriptors that mysqld requires
  - Dynamic: YES
  - Default Value: V.5.7: 2k | V.8 : 4k
  - Recommended: Check the **Opened_tables** status variable. If the value of Opened_tables is much larger than this variable we need to increase it
- **table_open_cache_instances**: The number of open tables cache instances. Used to improve scalability by reducing contention among sessions
  - Dynamic: YES
  - Default Value: 16 and MAX of 64
  - Recommended: The default value is usually OK but do not have it much larger than the number of CPU cores.
- **table_definition_cache**: Cache table definitions i.e. this is where the CREATE TABLE are cached to speed up opening of tables and only one entry per table
  - Dynamic: YES
  - Default Value: -1, autosized defined by MIN(400 + table_open_cache / 2, 2000)
  - Recommended: # of user-defined tables + 10%, unless 50K+ tables

MySQL Server General Settings

- **slow_query_log**: Whether the slow query log is enabled
  - Dynamic: YES
  - Default Value: OFF
  - Recommended: ON
- **long_query_time**: If the slow query log is enabled, the query is logged to the slow query log file if a query takes longer than this many seconds
  - Dynamic: YES
  - Default Value: 10 seconds
  - Recommended: The default of 10 seconds is usually too high and queries that take longer than 1 second to execute is a good candidate for optimization but if the server is a busy one a value between 3 to 5 seconds should be ok.

MySQL Server General Settings

- **sort_buffer_size**: How large your filesort buffer is. Used to sort the rows without index
  - Dynamic: YES
  - Default Value: 256KB
  - Recommended: Keep the default
- **join_buffer_size**: The minimum size of the buffer that is used for plain index scans, range index scans, and joins that do not use indexes and thus perform full table scans
  - Dynamic: YES
  - Default Value: 256KB
  - Recommended: Keep the default

MySQL Server General Settings

PS: join/sort/read/rnd_read buffers: It is usually best to leave at defaults. They are allocated per-session and all-at-once, so they can use a large amount of memory, and for the vast majority of the workloads setting too large has negative scalability (i.e. the larger the variable is set, the slower DB will perform).

If there is any evidence that a specific query improves with a larger setting for any of these buffers, we simply set the desired size for the particular **session** where that query will run, for example:

```
SET join_buffer_size=131072;
```

Also, note that **sort_buffer_size** variable should be close to CPU cache size and could be set to 1-4MB instead of default 256kb on modern processors.

InnoDB Settings

- **innodb_buffer_pool_size**: The size of the memory area where InnoDB caches table and index data
  - Dynamic: YES
  - Default Value: 128MB
  - Recommended: Typically 70%-80% of your server's memory but can be higher if more memory available or lower to leave enough memory to OS if otherwise
- **innodb_buffer_pool_instances**: The number of regions that the InnoDB buffer pool is divided into
  - Dynamic: NO
  - Default Value: 8 (or 1 if innodb_buffer_pool_size < 1GB). Max of 64
  - Recommended: Keep the default unless the server has enough memory and CPU cores to justify the change
- **innodb_log_buffer_size**: The size of the buffer that InnoDB uses to write to the log files on disk
  - Dynamic: NO
  - Default Value: 16MB
  - Recommended: A larger buffer enables large transactions to run without write the log to disk before the transactions commit. Can start with 64MB and increase incrementally

InnoDB Settings

- **innodb_log_file_size**: The size in bytes of each log file in a log group
  - Dynamic: NO
  - Default Value: 48MB
  - Recommended: The larger the value, the less checkpoint flush activity is required by buffer pool, saving disk I/O. The drawback is the recovery process will take longer if the database was abnormally shutdown (crash or killed, either OOM or accidental)
- **innodb_flush_log_at_trx_commit**: Controls the balance between strict ACID compliance for commit operations and higher performance
  - Dynamic: YES
  - Default Value: 1
  - Recommended: It is safe to set to 0 on PXC as IST or SST will recover from crash
- **innodb_flush_method**: The method used to flush data to InnoDB data files and log files
  - Dynamic: NO
  - Default Value: 0
  - Recommended: If using a RAID with battery-backed cache, DIRECT_IO helps relieve I/O pressure. If storage is SAN based, O_DSYNC might be faster for a read-heavy workload with mostly SELECT statements

Galera Settings

- **wsrep_slave_threads**: The number of threads that can apply replication transactions in parallel
  - Dynamic: YES
  - Default Value: 1
  - Recommended: The default, 1, underutilizes multi-core boxes. A value of half to double the number of cores can be applied but better start with smaller values and increase incrementally
- **wsrep_retry_autocommit**: The number of retries the node attempts when an autocommit query fails
  - Dynamic: YES
  - Default Value: 1
  - Recommended: Depends on application needs
- **wsrep_sst_donor**: A list of nodes (using their wsrep_node_name values) that the current node should prefer as donors for SST and IST
  - Dynamic: YES
  - Default Value: 1

If the value is empty, the first node in SYNCED state in the index becomes the donor and will not be able to serve requests during the state transfer. To consider other nodes if the listed nodes are not available, add a comma at the end of the list, for example:

```
wsrep_sst_donor=node1,node2,
```

If you remove the trailing comma from the previous example, then the joining node will consider only node1 and node2

Galera Settings - gcache

- When a transaction is executed on one node it is replicated to another node(s) of the cluster. This transaction is then copied over from the group channel to Galera-Cache (GCache) followed by apply action;
- It is a circular (RingBuffer) file and the cache can be discarded immediately once the transaction is applied, but retaining it can help promote a node as a DONOR node serving write-sets for a newly booted node;
- If the transaction write-set is large enough not to fit in the RingBuffer File (actually large enough not to fit in half of the RingBuffer file) then an independent page (physical disk file) is allocated to cache the write-sets;
- In short, GCache acts as a temporary storage for replicated transactions.

Galera Settings - gcache

The relevant parameters for us today are:

- **gcache.size**: The size of the gcache file
- **gcache.page_size**: The size of the standard page stored into the gcache file.
- **gcache.keep_pages_size**: defines total size of allocated pages to keep. For example, if keep_pages_size = 10M then N pages that add up to 10M can be retained. If N pages add to more than 10M, then pages are removed from the start of the queue until the size falls below set threshold. A size of 0 means don’t retain any page.

Galera Settings - calculating gcache size

We need to check how many bytes are written every minute to be able to calculate an appropriated gcache size. The variables we are interested to check are:

- wsrep_replicated_bytes: Total size (in bytes) of writesets sent to other nodes
- wsrep_received_bytes: Total size (in bytes) of writesets received from other nodes;

```
show global status like 'wsrep_received_bytes';
show global status like 'wsrep_replicated_bytes';
select sleep(60);
show global status like 'wsrep_received_bytes';
show global status like 'wsrep_replicated_bytes';
```

Bytes per minute: (second wsrep_received_bytes – first wsrep_received_bytes) + (second wsrep_replicated_bytes – first wsrep_replicated_bytes)

If we want to allow one hour of maintenance (or downtime) of a node, we need to increase the gcache to the size of 60 min. If we want more time, just make it bigger.

Galera Settings - Flow Control

Flow control is a replication feedback mechanism that allows any node in the cluster to instruct the group when it needs replication to pause and when it is ready for replication to continue. This prevents any node in the synchronous replication group from getting too far behind the others in applying replication.

It may sound counter-intuitive to heard (see) the last sentence, "**synchronous replication node getting too far behind the others**" but we need to remember that Galera uses synchronous certification bu asynchronously apply the changes.

Galera's replication is synchronous to the point of ensuring transactions are copied to all nodes and global ordering is established, but apply and commit is asynchronous on all but the node the transaction is run on.

It's important to note that Galera prevents conflicts to the transactions that have been certified but not yet applied.

Galera Settings - gcs.fc_limit

- Pauses the replication if *wsrep_local_recv_queue* queue exceeds this number of writesets;
- Defaults to 16 transactions. This effectively means that this is as far as a given node can be behind committing transactions from the cluster;
- For master-slave setups, writes go to only one node, this number can be increased considerably;
- When gcs.fc_master_slave is set to NO (multi-master), the gcs.fc_limit parameter is dynamically modified;

Galera Settings - gcs.fc_master_slave

- Defines whether there is more than one source of replication;
- As said before, when this parameter value is set to NO (multi-master), the gcs.fc_limit parameter is dynamically modified to give more margin for each node to be a bit further behind applying writes;
- The gcs.fc_limit parameter is modified by the square root of the cluster size, that is, in a four-node cluster it is two times higher than the base value. This is done to compensate for the increasing replication rate noise.

Galera Settings - gcs.fc_factor

- fc_factor addresses when flow control should be released;
- The factor is a number between 0.0 and 1.0, which is multiplied by the current fc_limit (adjusted by the above calculation if fc_master_slave=NO);
- For example, a value of 0.5 means the *wsrep_local_recv_queue* has to fall below 50% of the fc_limit before replication is resumed and flow control released;

A configuration example can be seen below:

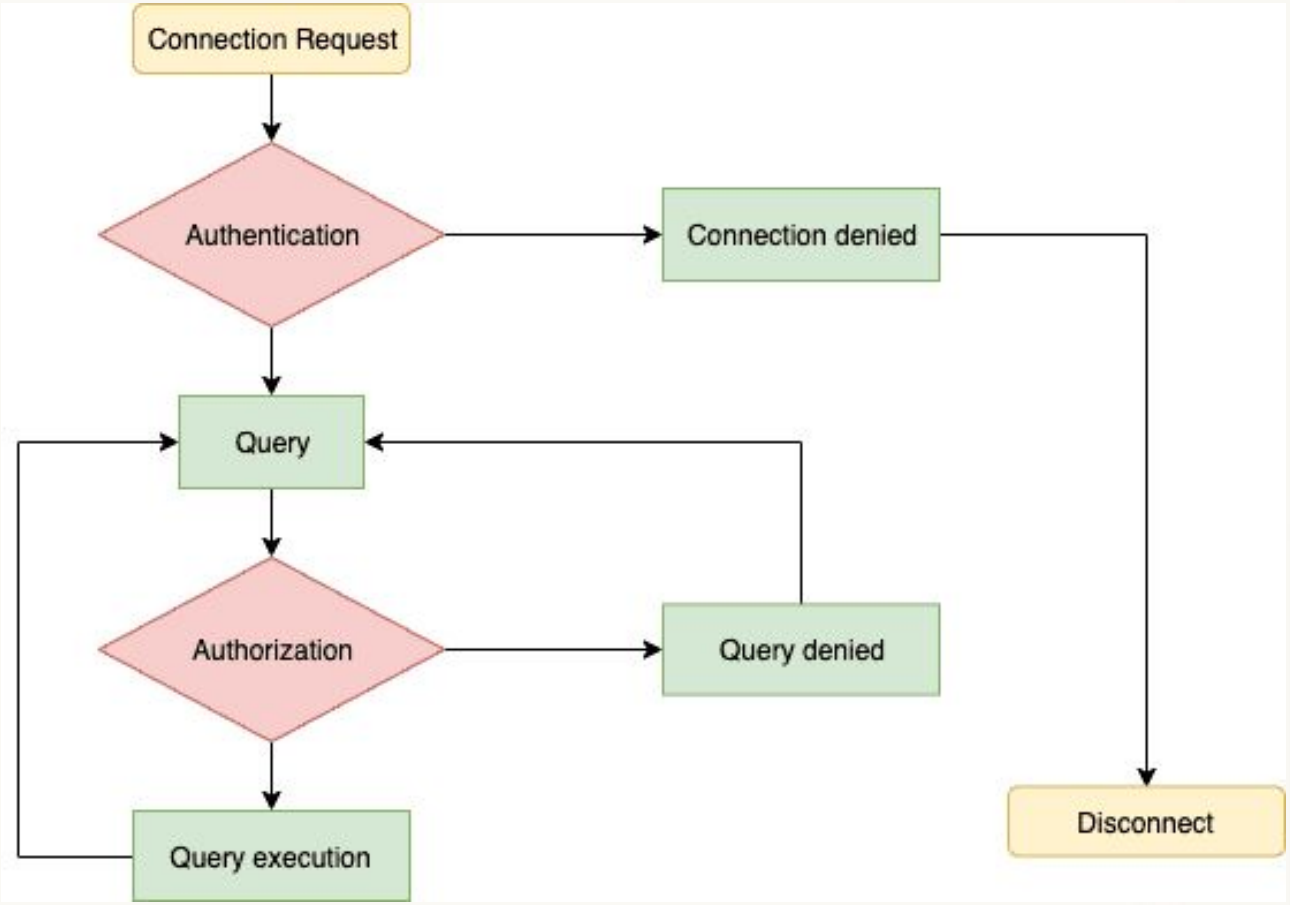
```
set global wsrep_provider_options="gcs.fc_limit=500; gcs.fc_master_slave=YES;
gcs.fc_factor=0.5";
```

OS Settings

- **Transparent Huge Pages**: Databases use small memory pages, and the Transparent Huge Pages tend to become fragmented and impact performance. The recommendation is to disable THP;
- **Swappiness**: Swappiness basically sets the weight of the Linux file cache pages, for Innodb which has its own cache, this can cause memory issue. The recommendation is to set it to 1;
- **IO Scheduler**: The default IO scheduler (CFQ) performs poorly for databases setup. The recommendation is to use deadline for RAID with spindles and noop for SSDs;
- **NUMA interleaving**: Non-uniform memory access (NUMA) is a memory design where an SMP's system processor can access its own local memory faster than non-local memory (the one assigned local to other CPUs). This may result in suboptimal database performance and potentially swapping. When the buffer pool memory allocation is larger than size of the RAM available local to the node, and the default memory allocation policy is selected, swapping occurs. Make sure to set innodb_numa_interleave=1

MySQL Security

User Management



User Management

- Authentication: Verifies the user's identity;
- Authorization: Verifies the user's privileges;
- MySQL includes several components and plugins that implement security features:
  - Plugins for authenticating attempts by clients to connect to MySQL Server
  - A password-validation component for implementing password strength policies
  - Keyring plugins that provide secure storage for sensitive information;
  - Percona uses MySQL Audit API to implement audit file
- The mysql database contains the information for all user accounts on the server;
- Can query the mysql.user table to view user identification info;
- The same table lists all user info, including privileges:

```
SELECT * FROM mysql.user WHERE user='root';
```

Host Access Management

- Mysql uses host-based authentication, which means that an account name consists of a username and the name of the client host from which the user must connect to the server;
- Account names have the format ‘user_name’@’host_name‘
- Usernames can be up to 16 characters long;
- The ‘%’ or’_’ wildcard characters can be used to set up an account that enables the user to connect from any host in an entire domain or subnet;

High Availability and common tasks with PXC

Schema change

- Schema changes are one of the big challenges in Galera replication. So, it is important to understand the schema changes operation when using PXC/Galera clusters;
- There are few methods to perform schema changes and the ones we'll discuss here are:
  - Schema changes with "wsrep_OSU_method = TOI"
  - Schema changes with "wsrep_OSU_method = RSU"
  - Schema changes with "ONLINE ALGORITHMS"
  - Schema changes with "pt-osc"

Schema change - TOI

- TOI is the default method (wsrep_OSU_method = TOI) for schema changes;
- All the nodes in the cluster will be pause during the ALTER process. Because the ALTER needs to be replicated on all the nodes. If the ALTER is big it will affect the performance and could be the cause of the downtime;
- Rollback is not possible on schema upgrade;
- You can’t kill the ALTER query immediately during the operation. So, your application may need to wait until the ALTER completion;
- DDL statements are processed in the same order with regard to other transactions in each node;
- The full cluster will be blocked/locked during the DDL operation;
- This guarantees data consistency;

Schema change - RSU

- In this method, DDL statements will not replicate across the cluster nodes. Need to execute the DDL individually on all nodes.
- The node which is executing the DDL will desync from the cluster group. The other nodes in the cluster are still operational and receive the application connections.
- Once the node executes the DDL, it will start to apply the missing writesets.
- In this method, the important thing is the WRITEs should not be performed on that particular table until the schema upgrade completes on all the nodes. Users should be very clear on this because the failure will break the cluster and the data may be unrecoverable.
- Gcache should be good enough to store the writesets.

Schema change - ONLINE ALGORITHMS

So far, we have 3 algorithms:

- INPLACE
- COPY
- INSTANT
- With TOI: “ALGORITHM = INPLACE / COPY” still pauses the cluster during the operation. Galera doesn’t allow transactions when an ALTER TABLE statement is run. So if you are using TOI, any ALTER TABLE will block all transactions on all nodes;
- With RSU: “ALGORITHM = INPLACE/COPY” is still not beneficial on RSU. It pauses the Galera replication and takes the node to Desync
- “ALGORITHM=INSTANT” is supported and faster in RSU. But, still, you can use TOI to avoid the additional work;
- Recommendations is to use the “ALGORITHM = INSTANT ” with TOI wherever we can. But, make sure we have the MySQL 8.x + version. Unfortunately, “ALGORITHM=INSTANT” currently only supports adding new columns.

Schema change - pt-osc

- Pt-osc provides non-blocking schema upgrades on all nodes in one shot;
- This should be used with the TOI method.
- The action flow will be like this:
  - Create a new table “_tablename_new” with the required modification
  - Creates triggers for update the modified rows (insert / update / delete)
  - Copy the records from the original table to the new table using chunk operation.
  - Once the copy is completed, it will swap the table (original → _old, _new → original) and drop the triggers and old table. Direct DDLs (RENAME TABLE, DROP TABLE) will be used for this operation (wsrep_OSU_method=TOI).
- Pt-osc provides several options to perform the effective operations. For example, one control the connections, active threads, load, chunk size etc...
- There is the option “–max-flow-ctrl” for Galera. It will check the average time cluster spent pausing for FC and make the tool pause if it goes over the percentage indicated in the option. By default, the tool will not check the FC.

Bonus if time allows: Mix setup with Async Replication

- Live setup!

Thanks!

- We are done here for the day.
- Feel free to ask any questions during the class, on github at https://github.com/elchinoo/tutorial-pxc or drop me an email at charly.batista@percona.com
- You can also find me at https://www.linkedin.com/in/charlybatista