EDB Postgres Distributed

Version 3.6

Built at 2024-04-17T21:37:34

1 EDB Postgres Distributed

EDB Postgres Distributed provides loosely-coupled multi-master logical replication using a mesh topology. This means that you can write to any server and the changes are sent directly, row-by-row to all the other servers that are part of the same mesh.

EDB Postgres Distributed consists of several components that make the whole cluster work.

Postgres server

Two different Postgres distributions can be used:

- PostgreSQL - open source
- EDB Postgres Extended Server - PostgreSQL compatible and optimized for replication

What Postgres distribution and version is right for you depends on the features you need. See the feature matrix in Choosing a Postgres distribution for detailed comparison.

BDR

A Postgres server with the BDR extension installed is referred to as a BDR node. BDR nodes can be either data nodes or witness nodes.

Witness nodes don't participate in data replication and are only used as a tie-breaker for consensus.

Available as two editions, BDR Standard provides essential multi-master replication capabilities for delivering row level consistency to address high availability and/or geographically distributed workloads. BDR Enterprise adds advanced conflict-handling and data-loss protection capabilities.

BDR Enterprise

To provide very high availability, avoid data conflicts, and to cope with more advanced usage scenarios, the Enterprise edition provides the following extensive additional features:

- Eager replication provides conflict free replication by synchronizing across cluster nodes before committing a transaction
- Commit at most once consistency guards application transactions even in the presence of node failures
- Conflict-free replicated data types (CRDTs) provide mathematically proven consistency in asynchronous multi-master update scenarios
- Column level conflict resolution enables per column last-update wins resolution to merge updates
- Transform triggers execute on incoming data for modifying or advanced programmatic filtering
- Conflict triggers provide custom resolution techniques when a conflict is detected

BDR Enterprise requires EDB Postgres Extended v11 (formerly known as 2ndQuadrant Postgres) which is SQL and on-disk compatible with PostgreSQL.

> **Note**
>
> The documentation for the latest stable 3.6 release is available here:
>
> BDR 3.6 Enterprise Edition
>
> **This is a protected area of our website, if you need access please contact us**

BDR Standard

The Standard edition provides loosely-coupled multi-master logical replication using a mesh topology. This means that you can write to any node and the changes will be sent directly, row-by-row to all the other nodes that are part of the EDB Postgres Distributed cluster.

By default BDR uses asynchronous replication to provide row-level eventual consistency, applying changes on the peer nodes only after the local commit.

The following are included to support very high availability and geographically distributed workloads:

- Rolling application and database upgrades to address the largest source of downtime
- DDL replication supports changes to application schema, ideal for use in continuous release environments
- Sequence handling provides applications different options for generating unique surrogate ids that a multi-node aware
- Tools to assess applications, monitor operation, and verify data consistency

BDR Standard requires PostgreSQL v10 or v11.

> **Note**
>
> The documentation for the latest stable 3.6 release is available here:
>
> BDR 3.6 Standard Edition
>
> **This is a protected area of our website, if you need access please contact us**

pglogical 3.6

BDR depends on the pglogical 3 extension to provide the replication channel upon which BDR builds.

> **Note**
>
> The documentation for the latest stable 3.6 release is available here:
>
> pglogical 3.6
>
> **This is a protected area of our website, if you need access please contact us**

2 BDR Release Notes

BDR 3.6.33 (2022 Nov 16)

This is a maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Fix bdr.alter_table_conflict_detection() (BDR-2650)

Correct the function to use the 'detector_args' argument.

Upgrades

This release supports upgrading from the following versions of BDR:

- 3.6.20 and higher

BDR 3.6.32 (2022 Aug 23)

This is a maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Catchup replication slot cleanup during PARTing of a node (BDR-2368, RT82884)
  When parting a node, the catcup replication slot may be left behind if the source node used for catching up changes. Clean up these replication slots once the catchup finishes. Similarly clean up the catchup information from BDR catalogs.

- Cleanup replication slot when bdr_init_physical fails (BDR-2364, RT74789)
  If `bdr_init_physical` aborts without being able to join the node, it will leave behind an inactive replication slot. Remove such a replication slot when it is inactive before an irregular exit.

Improvements

- Allow consumption of the reserved galloc sequence slot (BDR-2367, RT83437, RT68255)
  The galloc sequence slot reserved for future use by background allocator can be consumed in the presence of consensus failure.

Upgrades

This release supports upgrading from the following versions of BDR:

- 3.6.20 and higher

BDR 3.6.31 (2022 May 17)

This is a maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Make ALTER TABLE lock the underlying relation only once (RT80204)
  This avoids the ALTER TABLE operation falling behind in the queue when it released the lock in between internal operations. With this fix, concurrent transactions trying to acquire the same lock after the ALTER TABLE command will properly wait for the ALTER TABLE to finish.

- Reduce log for `bdr.run_on_all_nodes` (BDR-2153, RT80973)
  Don't log when setting `bdr.ddl_replication` to off if it's done with the "run_on_all_nodes" variants of function. This eliminates the flood of logs for monitoring functions.

Improvements

- Use 64 bits for calculating lag size in bytes (BDR-2215)

Upgrades

This release supports upgrading from the following versions of BDR:

- 3.6.20 and higher

BDR 3.6.30 (2022 Feb 15)

This is a maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Ensure loss of CAMO partner connectivity switches to Local Mode immediately
  This prevents disconnected partner from being reported as CAMO ready.

- Fix the cleanup of bdr.node_pre_commit for async CAMO configurations (BDR-1808)
  Previously, the periodic cleanup of commit decisions on the CAMO partner checked the readiness of it's partner, rather than the origin node. This is the same node for symmetric CAMO configurations, so those were not affected. This release corrects the check for asymmetric CAMO pairings.

Upgrades

This release supports upgrading from the following versions of BDR:

- 3.6.20 and higher

BDR 3.6.29 (2021 Dec 14)

This is a maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Switch from CAMO to Local Mode only after timeouts (EE, RT74892)
  Do not use the `catchup_interval` estimate when switching from CAMO protected to Local Mode, as that could induce inadvertent

switching due to load spikes. Use the estimate only when switching from Local Mode back to CAMO protected (to prevent toggling forth and back due to lag on the CAMO partner).

- Prevent duplicate values generated locally by galloc sequence in high concurrency situations when the new chunk is used (RT76528)
  The galloc sequence could have temporarily produce duplicate value when switching which chunk is used locally (but not across nodes) if there were multiple sessions waiting for the new value. This is now fixed.

- Ensure that the group slot is moved forward when there is only one node in the BDR group
  This prevents disk exhaustion due to WAL accumulation when the group is left running with just single BDR node for prolonged period of time. This is not recommended setup but the WAL accumulation was not intentional.

- Advance Raft protocol version when there is only one node in the BDR group
  Single node clusters would otherwise always stay on oldest support protocol until another node was added. This could limit available feature set on that single node.

Improvements

- Reduce frequency of CAMO partner connection attempts (EE)
  In case of a failure to connect to a CAMO partner do not retry immediately (leading to a fully busy pglogical manager process), but throttle down repeated attempts to reconnect to once per minute.

- Ensure CAMO configuration is checked again after a reconnect (EE)

Upgrades

This release supports upgrading from following versions of BDR:

- 3.6.20 and higher

BDR 3.6.28.1 (2021 Nov 19)

This is a security and maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Please make sure you read the pglogical 3.6.28 release notes for the important security and physical promotion data loss fixes provided there.

Resolved Issues

- Ensure `bdr.camo_local_mode` _delay is actually being applied (BDR-803)
  This artificial delay allows throttling a CAMO node that is not currently connected to its CAMO partner to prevent it from producing transactions faster than the CAMO partner can possibly apply. In previous versions, it did not properly kick in after `bdr.global_commit_timeout` amount of lag, but only 1000 times later (due to erroneously comparing seconds to milliseconds).

- Fix potential FATAL error when using global DML locking with CAMO (BDR-1675, BDR-1655)

- Fix lag calculation for CAMO local mode delay (BDR-1681)
  The lag calculation could have easily overflowed causing the CAMO to behave wrongly on partner disconnect.

- Fix `bdr.alter_sequence_set_kind` to accept a bigint as a start value (RT74294)
  The function was casting the value to an int thus getting bogus values when bigint was used.

- Fix node replication slot handling on node name reuse (RT71888)
  Cleanup local info slot name when the slot is dropped, this will prevent data loss by not dropping the slot during `bdr_init_physical` when it joins a node re-using the same name.

- Don't stop consensus worker when manager exists (RT73539)
  Killing the consensus after configuration changes would destabilize Raft.

- Fix ignored `--recovery-conf` option in `bdr_init_physical`.
  `bdr_init_physical` was completely ignoring the value of the `--recovery-conf` option if provided. It would append to a `recovery.conf` if one already existed in the target data dir but totally ignored the supplementary config provided on the command line, if any.

Improvements

- Allow user to specify a different `postgres.auto.conf` file. (BDR-1400)
  Added command-line argument `--postgresql-auto-conf` to be used when the user needs to specify a different `postgres.auto.conf` file.

- Log LSN when advancing replication origin during join Make `bdr_process_node_state_join_start()` log the LSN it advances the replication origin for the catchup join-target node to when BDR is started by `bdr_init_physical` with a `bdr.init_physical_lsn` set. This is critical information for diagnosing issues with physical join.

- Replication slot monitoring improvements (BDR-720)
  - Differentiate decoder slot in `bdr.node_slots`.
  - Include node and group information in `bdr.node_slots` when origin and target are in different node group.
  - Teach `bdr.monitor_local_replslots` what slots to expect.

- Improve documentation of the backup/restore procedure (RT72503, BDR-1340)
  Recommend against dropping the extension with cascade because it may drop user columns that are using CRDT types and break the sequences. It's better do use `drop_node` function instead.

Upgrades

This release supports upgrading from following versions of BDR:

- 3.6.20 and higher

BDR 3.6.27 (2021 Aug 12)

This is a security and maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Please make sure you read the pglogical 3.6.27 release notes for the important security and physical promotion data loss fixes provided there.

Resolved Issues

- Make sure `bdr_init_physical` does not leave temporary replication slots behind (BDR-191, RT70355)
  We now use native temporary slots in PostgreSQL which are automatically cleaned up.

- Make the consensus worker exit if postmaster dies (BDR1063, RT70024)
  This solves issues with consensus worker hanging after crash of other PostgreSQL process.

- Fix reuse of internal connection pool connections
  Improves behavior with larger number of nodes.

Upgrades

This release supports upgrading from following versions of BDR:

- 3.6.20 and higher

BDR 3.6.26 (2021 May 13)

This is a security and maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Please make sure you read the pglogical 3.6.26 release notes for the important security and physical promotion data loss fixes provided there.

Resolved Issues

- Fix crash in `bdr.run_on_all_nodes` when remote node returned an error

Other Changes

- Add `bdr.alter_subscription_skip_changes_upto()` (BDR-195)
  This function allows moving subscription ahead without replicating the changes which is useful for error recovery.

- Add a new documentation appendix which shows example of how to do table rewriting DDL safely with BDR (RTRT69514, RT69340).

- Improve coexistence with BDR 3.7 in the same BDR group (during upgrades).

Upgrades

This release supports upgrading from following versions of BDR:

- 3.6.20 and higher

BDR 3.6.25 (2021 Feb 11)

This is a security and maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Don't display additional node connection string in `node_summary` and `local_node_summary` views (RT69564)
  BDR only supports one connection string for each node.

- Fix "bdr node ... not found" in replay progress update (RT69779)
  Nodes that have pending progress info globally might no longer exist locally, and if that's the case is safe to ignore the progress update for them.

- Update state journal on node creation (RM20111)
  This solves corner cases where we could sometimes fail to join or part a node if the node with same name previously existed and was removed.

- Fix issues with stopping supervisor process when blocked by network call (RM20311)
  This issue could have resulted in PostgreSQL refusing to stop if the supervisor process was stuck on a network call.

Improvements

- Add new param detector_args to `bdr.alter_table_conflict_detection` (RT69677)
  To allow additional parameters for individual detectors.

- Currently the only additional parameter is `atttype` for the `row_version` detection method which allows using `smallint` and `bigint`, rather than just the default integer for the column type.

Upgrades

This release supports upgrading from following versions of BDR:

- 3.6.20 and higher

BDR 3.6.24 (2020 Dec 03)

This is a security and maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Unblock replay progress updates after a failing CAMO connection (EE) (RT69493, RM19924)
  After a connection to the configured CAMO partner (or origin) failed, for example if the CAMO partner simply has not been started at the start of the origin node, it was not properly reset. This not only prevented further connection attempts and prevented CAMO from operating, but also blocked replay progress updates from the blocked node. Which in turn prevents the group slot from advancing, meaning WAL data piled up on all nodes.

- This release not only corrects the cleanup of the CAMO connection state to enable retries, but also decouples replay progress updates from the CAMO connection. Should there ever be another issue with CAMO connections, replay progress updates and the group slot should not be affected.

- Ensure bdr_init_physical works on matching BDR versions (RT69520, RM19975)
  To initialize a new node from a physical copy, the new node must use the same BDR version as the node the physical backup was copied from. This release ensures bdr_init_physical checks and gives a useful error otherwise.

- Relax the safety checks in bdr.drop_node a bit again (RT69639)
  The check to prevent premature removal introduced in 3.6.23 turned out to be too strict and prevented dropping nodes as soon as any one peer had properly dropped the node already. This release relaxes the check again to allow dropping on all peer nodes.

Improvements

- Local node information in `bdr.node_summary` (RM20002, RT69541)
  Allow user to see information related to the local node even before the node is part of a `node_group`. This extends also to the `bdr.local_node_summary` view.

Upgrades

This release supports upgrading from following versions of BDR:

- 3.6.20 and higher

BDR 3.6.23 (2020 Nov 12)

This is a security and maintenance release for BDR 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Resolve CAMO transactions within a local transaction (RT69404)
  On rare occasions, the final commit stage of a CAMO or Eager All Node transaction still needs to do lookups in the system catalog. This led to a segfault, because BDR did not properly wrap this operation in a (read-only) transaction with full access to system catalogs.

- Better validate inputs to administration functions (RM19276, RM18108, RM17994, RT69013)
  When intentionally fed with invalid input (mostly NULL), some of the administrative functions of BDR crashed, leading to a restart of the Postgres node. This would constitute a denial of service attack for roles with either `bdr_application` or `bdr_superuser` privileges.

- Add a warning when trying to join known broken BDR versions (RT69088)
  BDR versions 3.6.19 and older are susceptible to data-loss when joining a BDR cluster. BDR now identifies this situation and logs a warning at join time.

- Handle missing column gracefully for ALTER COLUMN TYPE (RM19389, RT69114)
  The fix just throws same error Postgres would when ALTER COLUMN TYPE with non-existent column is executed.

- Fix JOINING state handling on consensus request timeout (RT69076)
  The timeout during JOINING state handling could result in node unable to join the BDR group. The retry logic now handles this state correctly.

- Change the snapshot restore to follow Raft paper more closely
  Fixes potential metadata consistency issues when node returns after longer period of downtime.

- Fix potential crash in bdr.resynchronize_table_from_node (RM19527)

- Extend `bdr.drop_node` with a check preventing premature removal (RM19280)
  The function `bdr.drop_node` by default now checks whether the node to drop has been fully parted on all nodes prior to allowing to drop its metadata. An additional force argument allows to disable this check and effectively exposes the previous behavior.

- Validation of `replication_set_remove_table` input (RT69248, RM19620)
  Check if the relation Oid passed as the function argument is valid.

Other Changes

- Improve error messages (RM19483)
- Minor documentation clarifications and language fixes (RM19825, RM19296, RT69401, RT69044)

BDR 3.6.22 (2020 Oct 01)

This is a security and maintenance release for BDR 3.6 which also includes various minor features.

Resolved Issues

- Correct a hang in `bdr.wait_for_apply_queue` (RM11416, also affects CAMO)
  Keepalive messages possibly move the LSN forward. In an otherwise quiescent system (without any transactions processed), this may have led to a hang in `bdr.wait_for_apply_queue`, because there may not be anything to apply for the corresponding PGL writer, so the `apply_lsn` doesn't ever reach the `receive_lsn`. A proper CAMO client implementation uses `bdr.logical_transaction_status`, which in turn uses the affected function internally. Thus a CAMO switch- or fail-over could also have led to a hang. This release prevents the hang by discarding LSN increments for which there is nothing to apply on the subscriber.

- Correct a problem with Raft snapshot writing when a 3.6.21 node became leader of a cluster that also included nodes on version 3.6.20 or earlier.

- Nodes that were parted but not subsequently dropped (so they still appear in the `bdr.node` table) can cause it as well. The issue can cause DDL locking to time out, galloc sequence allocation to stop, and other issues related to loss of functioning Raft-based consensus. If this problem is encountered, the following error will be seen in logs:

  `ERROR: invalid snapshot: record length [...] doesn't match expected length [...] for chunk "ddl_epoch"`

  and `SELECT bdr.get_raft_status()` will intermittently report `ERROR: could not find consensus worker when called.`

  To check if this issue could be affecting you, check `bdr.node` for any parted nodes with `proto_version_max` less than 13. Also query

  ```
  SELECT status->>'protocol_version' FROM bdr.get_raft_status() status;
  ```

  to see if the result is less than 13. If either is true and your cluster has 3.6.21 nodes you could be affected by the issue. To prevent or solve it you are affected by it, upgrade immediately to 3.6.22.

  This issue does not arise when all nodes are on 3.6.20 or older.

- Fix a problem with NULL values in `bdr.ddl_epoch` catalog (RM19046). Release 3.6.21 added a new `epoch_consumed_lsn` column to `bdr.ddl_epoch` catalog. Adding a new column would set the column value to NULL in all existing rows in the table. But the code failed to handle the NULL values properly. This could lead to reading garbage values or even memory access errors. The garbage values can potentially lead to global lock timeouts as a backend may wait on a LSN which is far into the future. We fix this by updating all NULL values to '0/0' LSN, which is an invalid value representation for LSN. The column is marked NOT NULL explicitly and the code is fixed to never generate new NULL values for the column.

- Allow consensus protocol version upgrades despite parted nodes (RM19041)
  Exclude already parted nodes from the consensus protocol version negotiation, as such nodes do not participate in the consensus protocol any more. Ensures the newest protocol version among the set of active nodes is used.

Improvements

- Numerous fixes for galloc sequences (RM18519, RM18512) The "nextval" code for galloc sequences had numerous issues:
  - Large INCREMENT BY values (+ve or -ve) were not working correctly
  - Large CACHE values were not handled properly
  - MINVAL/MAXVAL not honored in some cases The crux of the issue was that large increments or cache calls would need to make multiple RAFT fetch calls. This caused the loop retry code to be invoked multiple times. The various variables to track the loops needed adjustment.

- Error out if INCREMENT BY is more than galloc chunk range (RM18519)
  The smallint, int and bigint galloc sequences get 1000, 1000000, 1000000000 values allocated in each chunk respectively. We error out if the INCREMENT value is more than these ranges.

- Extend `bdr.get_node_sub_receive_lsn` with an optional committed argument
  The default behavior has been corrected to return only the last received LSN for a committed transaction to apply (filtered), which is the original intent and use of the function (e.g. by HARP). Passing a false lets this function return the unfiltered most recent LSN received, matching the previous version's behavior. This change is related to the hang in `bdr.wait_for_apply_queue` mentioned above.

- Fix tracking of the last committed LSN for CAMO and Eager transactions (RM13509)
  The GUC `bdr.last_committed_lsn` was only updated for standard asynchronous BDR transactions, not for CAMO or Eager ones.

BDR 3.6.21 (2020 Aug 14)

This is a security and maintenance release for BDR 3.6 which also includes various minor features.

Resolved Issues

- SECURITY: Qualify calls to unnest from `bdr.logical_transaction_status` (RM18359)
  Required to protect CAMO users from attack risks identified in CVE-2018-1058, when the user application avoids the insecure coding practices identified there. See BDR Security chapter for further explanation. `bdr.logical_transaction_status` is typically only used by an application taking advantage of the CAMO feature.

- SECURITY: Qualify a call to an inequality operator in `bdr_init_physical` (RM18359)
  Since 3.6.20, the `bdr_init_physical` utility uses the `<>` inequality operator. Similar to the above entry, the operator is now called by its fully qualified name to avoid attack risks identified in CVE-2018-1058.

- SECURITY: Recent PostgreSQL 11.9 et al reported CVE-2020-14349. The related risk is the same as CVE-2018-1058 and was already handled in BDR3.6.19, so no further action has been taken in this release.

- Re-add the "local_only" replication origin (RT68021)
  Using `bdr_init_physical` may have inadvertently removed it due to a bug that existing up until release 3.6.19. This release ensures to recreate it, if it's missing.

- Eliminate SQL calls to pg_switch_wal from BDR (RT68159, RM17356)
  The helper function `bdr.move_group_slot_all_nodes` as well internal replay progress update handling invoked the Postgres function pg_switch_wal via SQL, thus requiring explicit execute permissions. Instead use direct internal calls from the BDR extension to avoid that requirement.

- Handle NULL arguments passed to functions in BDR schema gracefully (RT68375, RM17994)
  Some of the functions in BDR schema, e.g. `bdr.alter_node_add_log_config()` caused segmentation fault when NULL arguments were passed to those. Those are fixed to handle the NULL arguments appropriately. Those functions which do not expect NULL arguments now throw an error mentioning so. Others handle NULL arguments according to the functionality offered by individual function.

- Prevent violation of NOT NULL constraints in BDR-internal catalogs (RM18335)
  Postgres 11.9 is more strict with NOT NULL constraints for catalog tables. Correct and relax constraints for two internal tables for which BDR stores NULL values in.

- Wait for replication changes triggered by prior epoch (RM17594, RM17802)
  When a node requests a global DDL lock, it now waits until it has seen the replication changes associated with the previous DDL. If the previous DDL was run on the same node, then there won't be any additional wait. But if the previous DDL was run on some other node, then this node must wait until it catches up replication changes from the other node. This improves handling of multiple concurrent DDL operations across the BDR Group which would previously result in global lock timeout, but now may cause additional wait in case of a large replication lag between the pair of nodes.

- Fix incorrect handling of MAXVALUE and MINVALUE for galloc sequences (RM14596)

- Fix concurrent action of nextval/currval when changing sequence kind (RT68438)
  During nextval/currval calls, the sequence kind was fetched prior to locking the sequence, which allowed a concurrently executed call to `bdr.alter_sequence_set_kind()` to influence the returned value incorrectly.

- Disallow SCHEMA and SEQUENCE RENAME for galloc sequences (RM15554)
  The consistent range allocator for galloc sequences identifies the sequence using schema name and sequence name, so changing the name of either of those could break it.

- Adjust chunk size of a galloc seq chunk if type was changed (RT68470, RM18297)

- Don't drop `bdr_local_only_origin` in `bdr_init_physical`
  Also recreate it in upgrade script in case previous version of BDR has dropped it.

- Ensure there is at least one transaction after wal switch in `bdr.move_group_slot_all_nodes` (RT67591, RM17356)
  This will move the group slot forward more hastily on servers with low concurrent activity.

- Fix PART_CATCHUP node state handling (RM17418)
  In 3.6.20 different nodes might have had different ideas about what to do when there is not fully joined node during part of another node which could cause the part operation to stall indefinitely with only possible solution being forced part. 3.6.21 leaves the decision on what to do next on Raft leader and other nodes just follow, so the decision making is consistent across whole BDR group.

- Fix incorrect cache handling for sessions established prior to BDR node creation (RT68499).

Improvements

- Allow use of CRDTs when BDR extension installed but without any node (RM17470)
  Previously, restoring CRDT values on a node with BDR extension installed, but without a node created, would throw an ERROR when the CRDT data type requests the node id. We now store an InvalidOid value when the node id is not available. If the node is subsequently added to a BDR cluster, when the CRDT column is updated, InvalidOid will be replaced by a normal node id.

- Check validity of options when altering a galloc sequence (RM18301, RT68470)
  Galloc sequences do not accept some options, so warn the user in case invalid options are used, as already occurs with timeshard sequences.

- `resynchronize_table_from_node()` freezes the table on target node (RM15987)
  When we use this function the target table is truncated first and then copied into on the destination node. This activity additionally FREEZEs the tuples as the data is loaded. This then avoids a rewriting the target table to set hint bits, as well as avoiding high volume WAL writes which would occur when the table was first used after resyncronizing.

- Create a virtual sequence record on other nodes (RM16008, RT68438, RT68432)
  If we create a galloc sequence and try to use its value in the same transaction block, then because it does not exist yet on other nodes, it used to error out with "could not fetch next sequence chunk" on the other nodes. We solve this by creating a virtual record on the other nodes.

- Add start parameter to bdr.alter_sequence_set_kind() (RT68430)
  Allows specifying new starting point for galloc and local sequences in single step with changing the sequence kind.

- Add an internal consistent consensus-based key-value store for use by HARP (RM17825)
  This is not meant for direct use by users, but enables features for additional tooling.

- Improve latency of global locks in cases where other nodes respond very quickly, such as when there is no replication lag

- Throw errors for configuration parameters that were supported by BDR1 and BDR2, but are no longer supported by BDR3 (RT68529).
  Since PostgreSQL does not throw an ERROR/WARNING while setting an extension qualified, non-existent parameter, users with BDR1/BDR2 experience may fail to notice that the parameter is not supported by BDR3. We now explicitly catch all attempts to use parameter names that match unsupported BDR1/BDR2 configuration parameters and throw an ERROR.

- Document known issues in "Appendix C" section of documentation (RM18169)

BDR 3.6.20 (2020 Jun 12)

This is a security and maintenance release for BDR 3.6 which also includes various minor features.

Additional Actions

- Run LiveCompare following logical join to handle rare divergent errors (RT67307)
  When running a logical join when nodes are down or have large replication lag can cause divergence of concurrent transactions that conflict during join. When running a logical join from a source to a target node, changes made on other nodes that were still in-flight could cause conflicts when applied; if conflicts occurs, conflict resolution might diverge for those rows. Running LiveCompare immediately following the join will clean up any such issues; this can be carefully targeted using a conflict logging table. Required actions are now fully documented.

Resolved Issues

- Ignore self-conflicts during logical join that could lead to data loss (RT67858)
  Changes made to tables during logical join could conflict with the original rows, as a result of rewriting the commit timestamp onto the target node. Explicitly ignore such conflicts seen while in join catchup mode, avoiding the associated data loss.

- Ensure origin position messages cannot be skipped during join (RT67858)
  Progress watermark messages are now written to WAL transaction stream rather than being issued asynchronously via the messaging layer. During join we now wait for the joining node to see at least one watermark to ensure no timing window exists that could lead to skipping the origin position during join, thus avoiding data loss.

- Resilience against idle_in_transaction_session_timeout (RM13649, RT67029, RT67688)
  Set idle_in_transaction_session_timeout to 0 so we avoid any user setting that could close the connection and invalidate the snapshot.

- Allow early part_node to interrupt a hanging join (RT67980, RM16659)
  In case a new node failing to join and not making any progress, attempting to remove it with plain (non-forced) `bdr.part_node` could lead to a hang. Resolve this by skipping the PART_CATCHUP phase for such a node, as it did not possibly produce any transactions to catch up to.

- Warn when joining to BDR group with versions of BDR which have known consistency issues
  The above fixes for join process only work if whole BDR node is upgraded to the 3.6.20 or higher (protocol version 12+), so we warn about joining to groups with older versions. In-place upgrades work normally.

- Fix minor issues detected by Coverity scanner.

Improvements

- Ignore `bdr.assess_update_replica_identity` parameter in writer worker processes (RM15983) (EE)
  Setting this in `postgresql.conf` can cause a potential outage if executed by writer worker processes. Hence, writer worker processes ignore this parameter.

- CentOS 8 is now supported, starting with this release.

- Improve diagnostic logging of the join process (RT67858)
  Include the fact that we retry the join state requests after failure in the error message and raise the log level of slot/origin creation/movement from DEBUG to LOG so that it's always logged by default.

BDR 3.6.19 (2020 May 20)

This is a security and maintenance release for BDR 3.6 which includes also includes various minor features.

Resolved Issues

- SECURITY: Set search_path to empty for internal BDR SQL statements (RM15373)
  Also, fully qualify all operators used internally. BDR is now protected from attack risks identified in CVE-2018-1058, when the user application avoids the insecure coding practices identified there. See BDR Security chapter for further explanation.

- SECURITY: Raise required privileges for BDR admin functions (RM15542)
  When executed by roles other than superuser or bdr_superuser:

  - bdr.alter_table_conflict_detection needs table owner
  - bdr.column_timestamps_enable needs table owner
  - bdr.column_timestamps_disable needs table owner
  - bdr.drop_trigger needs table owner
  - bdr.alter_sequence_set_kind needs sequence owner
  - bdr.global_lock_table needs UPDATE, DELETE or TRUNCATE (like LOCK TABLE)
  - bdr.create_conflict_trigger needs TRIGGER permission on the table and EXECUTE permission on the function
  - bdr.create_transform_trigger needs TRIGGER permission on the table and EXECUTE permission on the function

  A new GUC `bdr.backwards_compatibility` allows to skip this newly introduced check for existing clients requiring the former behavior.

- Resolve a hang possible after multiple global lock releases (RT67570, RM14994)
  A bug in the code path for releasing a global lock after a timeout led to overriding the backend's PID with a value of `-1`, also showing up in the `waiters` list of `bdr.global_locks`. This in turn crippled the waiters list and ultimately led to an infinite loop. This release fixes the override, which is the original cause of this hang and correctly removes entries from the lock wait list.

- Correct parsing of BDR WAL messages (RT67662)
  In rare cases a DDL which is replicated across a EDB Postgres Distributed cluster and requires a global lock may cause errors such as "invalid memory alloc request size" or "insufficient data left in message" due to incorrect parsing of direct WAL messages. The code has been fixed to parse and handle such WAL messages correctly.

- Fix locking in ALTER TABLE with multiple sub commands (RM14771) Multiple ALTER TABLE sub-commands should honor the locking requirements of the overall set. If one sub-command needs the locks, then the entire ALTER TABLE command needs it as well.

- Fix bug in example of ALTER TABLE ... ADD COLUMN workaround (RT67668)
  Explain why bdr.global_lock_table() is needed to avoid concurrent changes that cause problems, in that case.

- Fix a hang after promotion of a physical standby (RM15728)
  A physical standby promoted to a BDR node may have failed to start replicating due to the use of stale data from an internal catalog cache.

- Fix crash when bdr.trigger_get_type() is called by itself (RM15592) Calling bdr.trigger_get_type() outside a streaming trigger function would cause a crash. Fixed the function to return NULL when called outside a streaming trigger function.

Improvements

- bdr.trigger_get_origin_node_id() allows preferred-node resolution (RM15105, RT67601)
  Some customers have a requirement to resolve conflicts based upon the node that is the source of the change. This is also known as trusted source, trusted site or AlwaysWins resolution. Previous versions of BDR allowed these mechanisms with 2 nodes; this new function allows this option with any number of nodes. Examples are documented.

- BDR now accepts URIs in connection strings (RM14588)
  All connection strings can now use the format URI "postgresql://... "

- New function bdr.resynchronize_table_from_node() (RM13565, RT67666, RT66968) allows a single table to be truncated and then resynced from a chosen node, while holding a global dml lock. This allows a table to be resynchronized following a data divergence or data corruption without needing to regenerate the whole node. Foreign Keys are removed and re-enabled afterwards.

- Improve filtering of changes made by explicitly unreplicated transactions (RM15557)
  Previously changes made by transactions using bdr.xact_replication = off or by bdr.difference_fix transactions would be sent to the remote node, generating spurious conflicts and wasting effort. Changes are now filtered on the source node instead, improving performance.

- Initial and periodic transaction status checks use async libpq (RM13504) (EE)
  With CAMO enabled, the status of in-flight transactions is checked against a partner node. This uses an standard Postgres connection via libpq, which used to block the PGL manager process. This release changes the logic to use asynchronous libpq to allow the PGL manager to perform other tasks (e.g. process Raft messages) while that status check is performed. This reduces chances of timeouts or deadlocks due to a more responsive PGL manager process.

- Additional message fields assist diagnosis of DDL replication issues (RM15292)

- Clarify documentation regarding privileges required for BDR users (RT67259, RM15533)

BDR 3.6.18 (2020 Apr 16)

This is a maintenance release for BDR 3.6 which includes minor features as well as fixes for issues identified previously.

Improvements

- Add synchronize_structure option to join_node_group (RM14200, RT67243)
  New synchronize_structure option can be set to either 'all' or 'none', which either sychronizes the whole schema or copies no DDL. This allows for rolling application schema upgrades to be performed with a user-managed schema (DDL) change step.

- Make bdr*difference_fix** functions use pre-created local origin (RM14189)*
  *The bdr*difference_fix** family of functions used to create a local origin to carry out conflict fixes. We now pre-create "bdr_local_only_origin" local origin at extension creation time. This same local origin is used by the above functions now.

- Adjust monitored values in bdr.monitor_group_versions() (RM14494)

We no longer report CRITICAL when pglogical version different to bdr version, which is actually not important. We now report WARNING if BDR editions differ between nodes.

- Substantial formatting corrections and spelling check of documentation

Resolved Issues

- Fix node join so it uses only `bdr_superuser` permissions (RM14121, RT67259)
  This affects the `join_target_dsn` connection of the `join_node_group` function, which has been fixed to work with only `bdr_superuser` right for the role used to connect.

- GRANT EXECUTE on bdr.show_subscription_status TO bdr_real_all_stats (RT67360, RM14624)
  This allows both bdr_read_all_stats and bdr_monitor roles to access the bdr.subscription_summary view

- Fix failure of bdr_init_physical to copy data columns using BDR types (RM14522)
  bdr_init_physical now uses bdr.drop_node() rather than DROP EXTENSION, which caused all columns using BDR datatypes such as CRDTs to be silently dropped from tables.

- Fix failure in 3.6.17 upgrade script caused by views referencing CRDTs (RT67505)
  Upgrade script now executed only on tables and mat views. Upgrade failure may give a spurious error such as "ERROR: BDR global lock manager not initialized yet"

- Set non-join subscriptions to CATCHUP state rather than INIT state at startup
  Avoids a rare but possible case of copying metadata twice during node join.

- Fix lookup for a galloc sequence when BDR catalogs are absent. (RT67455, RM14564)
  This might cause a query on a sequence to throw an error like "cache lookup failed for relation ..." when bdr library is added to shared_preload_libraries but BDR extension is not created.

- Allow ALTER TABLE ALTER COLUMN with BDR loaded but not initialized (RM14435)
  With the BDR extension loaded, but no local BDR node created, the DDL replication logic now still allows execution of an ALTER TABLE ALTER COLUMN operation.

- LOCK TABLE warning not shown when BDR node is not created (RM14613)
  Assess LOCK TABLE statement does not show when bdr.assess_lock_statement is set to a value other than 'ignore' until BDR node is created.

- Prevent a NULL dereference in consensus_disable (RM14618)
  `bdr.consensus_disable` expected the consensus process to be running. Fix it to prevent a segfault if that's not the case when the function is called.

BDR 3.6.17 (2020 Mar 31)

This is a maintenance release for BDR 3.6 which includes minor features as well as fixes for issues identified previously.

Improvements

- Allow ALTER TABLE ALTER COLUMN TYPE with rewrite when not replicating DDL (EE) (RM13244)
  In some cases, in controlled DBA environments, it is possible to change the type of a column to an implicitly castable one by adopting a rolling upgrade for the type of this column in a non replicated environment on all the nodes one by one. We allow concurrent activity on this table on other nodes during the rewrite. Also note that such ALTER commands cannot be run within transaction blocks.

- Add conflict logging configuration view (RM13691, RT66898)
  Add `bdr.node_log_config` view that shows information on the conflict logging configuration.

- Add new group monitoring views and functions (RM14014)
  These views and functions report the state of the BDR installation, replication slots and consensus across all nodes in the BDR group.

- Add current state of DDL replication related configuration parameters to log context (RM13637)
  Improves troubleshooting.

Resolved Issues

- Don't drop existing slot for a joining node (RM13310, RT67289, RT66797)
  This could have caused inconsistencies when node was joined using `bdr_init_physical` because it precreated the slot for new node which was supposed to be reused during join, instead it was dropped and recreated. We now keep the slot correctly which ensures there are no inconsistencies.

- Fix restart of CAMO node despite missing partner node (EE) (RM13899, RT67161)
  Prevent an error looking up the BDR node configured as a CAMO origin. In case the node got dropped, it does not exist, but might still be configured for CAMO.

- Fix locking in `bdr.column_timestamps_enable()` (EE) (RT67150)
  Don't hold same transaction and session level locks otherwise `PREPARE`, CAMO and Eager replication can't work for transactions where this is used.

- Don't try to apply BDR conflict resolution to PGL-only subscriptions (RT67178)
  BDR should only be active on BDR subscriptions, not pglogical ones.

- Let the CAMO partner return the final decision, once learned (RM13520)
  If an origin node switches to Local mode, temporarily dropping CAMO protections, it's possible for the CAMO partner to provisionally abort the transaction internally, but actually commit it eventually (to be in sync with the origin node). In earlier releases, this was not recorded leading to the status query function to continue to return an "aborted" result for the transaction. This release allows the final commit decision to override the provisional abort internally (catalog table bdr.node_pre_commit).

- Make CLCD/CRDT data types properly TOAST-able (EE) (RM13689)
  CLCD/CRDT data types were defined as using PLAIN storage. This can become as issue with a table with too many columns or if a large number of nodes are involved. This is now solved by converting these data types to use EXTENDED storage thus allowing for large sized values.

- Ensure duplicate messages are not received during node promotion (RM13972) Send a watermark from join source to the joining node during catchup phase of join to ensure it learns about current replication positions of all other nodes even if there are no data to forward from them during the catchup. Otherwise we might ask for older lsns during the promotion and receive duplicate messages and fail the join.

- Automatically disable CAMO for non-transactional DDL operations (EE)
  Several DDL operations are not allowed within a transaction block and as such cannot reasonably benefit from the protection that CAMO offers. Automatically disable CAMO for these, so as to avoid "cannot PREPARE" errors at COMMIT time.

- Fix errors when `bdr.move_group_slot_all_nodes` is called with no BDR node present in the database (RT67245)
  Allows setting up physical standbys of future BDR master before creating the BDR node.

- Make sure table has a `PRIMARY KEY` when CLCD is turned on (EE)
  This is sanity check that prevents user from enabling CLCD on tables without a `PRIMARY KEY` as that would break the conflict detection for such tables.

BDR 3.6.16 (2020 Mar 05)

BDR 3.6.16 is the sixteenth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified previously.

Improvements

- Add `bdr.alter_table_conflict_detection()` (RM13631)
  This function unifies the UI for changing conflict detection method for individual tables. Allows choice between origin based, row_version based and column level based (EE-only) conflict detection using same interface. The old functions are still supported, although they should be considered deprecated and will be removed in BDR 3.7.

- Add `bdr.default_conflict_detection` configuration option (RM13631)
  Related to the above `bdr.alter_table_conflict_detection()` function, the new configuration option allows setting the default conflict detection method for newly created tables.

- Change how forced part node works (RM13447)
  Forced node part will now first try to get consensus for parting and only do the local change if the consensus fails or if it's called for node which already started consensus based part process but the process has stuck on one of the steps.

- Automatically drop `bdr-enterprise` extension when dropping the `bdr` extension (RM13703)
  This improves usability when trying to drop the bdr extension without cascade, which is useful for example when user wants to keep the pglogical node associated with BDR.

- Improve error reporting when joining node with same name as existing active node (RM13447, RT66940)
  The previous error message was confusing as it made it seem like BDR does not allow node name reuse at all (which it does).

- Set application_name in `bdr_init_physical`
  Helps when diagnosing issues with this tool.

- Improve documentation of `ALTER TABLE` limitations (RM13512, RM13244, RT66940)
  Including new workaround for changing length of varchar columns.

Resolved Issues

- Fix pg_dump for BDR galloc sequences (RM13462, RT67051)
  Galloc sequences internally store extra data in sequence heap; BDR now hides the extra data from SELECTs so that queries on the sequence (which can be normal user query or a query from pg_dump for example) only show the usual sequence information.

- Fix enforcement of REPLICA IDENTITY FULL for CLCD
  Advanced conflict-handling approaches (CLCD, CRDT) require the table to have REPLICA IDENTITY FULL. However due to how the features initially evolved independently, this was not enforced (and documented) properly and consistently. We now correctly enforce the REPLICA IDENTITY FULL for CLCD for every table.

- Fix node name reuse of nodes which were used as join sources for other existing nodes in a BDR group (RM12178, RM13447)
  The source nodes have special handling so we need to make sure that newly joining node is not confused with node of same name that has been parted.

- Apply local states for existing nodes on newly joining node (RT66940)
  Otherwise decision making in during the join process might use wrong state information and miss some tasks like slot creation or subscription creation.

- Correctly clean node-level log filters and conflict resolver configuration (RM13704)
  This solves issues when trying to drop BDR node without dropping associated pglogical node and later recreating the BDR node again.

- Prevent a segfault in Raft on the parted BDR node (RM13705)

BDR 3.6.15 (2020 Feb 14)

BDR 3.6.15 is the fifteenth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified previously.

Improvements

- Keep a permanent log of all resolved CAMO decisions (RM12712)
  Record every decision taken by the CAMO partner when queried by `bdr.logical_transaction_status`, i.e. in the failover case.
- Add functions for enabling/disabling row version tracking (RM12930)
  Easier to use and less error prone interface than manually adding column and trigger.
- Add `currval()` and `lastval()` support for `timeshard` and `galloc` sequences (RM12059)
- Add `pglogical.min_worker_backoff_delay` setting to rate limit background worker re-launches, and `pglogical.worker_tasks` diagnostic view for background worker activity. See pglogical 3.6.15 release notes and documentation for details.

Resolved Issues

- Prevent buffer overrun when copying a TOAST column value inside the walsender output plugin (RT66839)
  This fixes issue that resulted in walsender crashes with certain types of workloads which touch TOASTed columns.
- Fix "type bdr.column_timestamps not found" error when bdr-enterprise extension is not installed when bdr enterprise library is in shared_preload_libraries (RT66758, RM13110)

BDR 3.6.14.1 (2020 Feb 05)

BDR 3.6.14 is an important bug fix release on top of BDR 3.6.14. This release contains a critical crash fix in memory allocation.

Resolved Issues

- Prevent buffer overrun when copying a TOAST column value inside the walsender output plugin (RT66839) This fixes issue that resulted in walsender crashes with certain types of workloads which touch TOASTed columns.
- Fix "type bdr.column_timestamps not found" error when bdr-enteprise extension is not installed when bdr enterprise library is in shared_preload_libraries (RT66758, RM13110)

BDR 3.6.14 (2020 Jan 31)

BDR 3.6.14 is a critical maintenance release of the BDR 3.6 series. This release includes major fixes for CAMO and other features as well as minor new features.

Improvements

- Add `bdr.camo_local_mode_delay` to allow throttling in CAMO Local mode (RM12402)
  Provides a simple throttle on transactional throughput in CAMO Local mode, so as to prevent the origin node from processing more transactions than the pair would be able to handle with CAMO enabled.

- Add `bdr.camo_enable_client_warnings` to control warnings in CAMO mode (RM12558)
  Warnings are emitted if an activity is carried out in the database for which CAMO properties cannot be guaranteed. Well-informed users can choose to disable this if they want to avoid such warnings filling up their logs.

- Warn on unrecognized configuration settings

- Move 'loading BDR' message earlier in startup messages

- Significantly enhance docs for Row Version Conflict Detection (RT66493)

- Clarify docs that NOTIFY is not possible with CAMO/Eager

- Add `global_lock_request_time`, `local_lock_request_time` and `last_state_change_time` columns to `bdr.global_locks` view for lock monitoring and diagnostic use.

- Add SQL functions for export/import of consensus snapshot (RM11433)
  These functions allow for manual synchronization of BDR system catalogs in case of corruption or user mistake.

Resolved Issues

- UPDATEs skipped on the partner node because remote_commit_ts set incorrectly (RM12476)
  Commit timestamps were unset in some CAMO messages, leading to losing last-update-wins comparisons that they should have won, which meant some UPDATEs were skipped when an UPDATE happened concurrently from another master. This doesn't occur normally in an AlwaysOn cluster, though could occur if writes happen via a passive master node.

- Only resolve those prepared transactions for which controlling backend is gone (RM12388)
  This fixes a race condition between the pglogical manager process and the user backend running a CAMO transaction. A premature attempt by the manager process to resolve a prepared transaction could lead to the transaction getting marked as aborted on the partner node, whereas the origin ends up committing the transaction. This results in data divergence. This fix ensures that the manager process only attempts to resolve prepared transactions for which the controlling user backend has either exited or is no longer actively managing the CAMO transaction. The revised code also avoids taking ProcArrayLock, reducing contention and thus improving performance and throughput.

- Prevent premature cleanup of commit decisions on a CAMO partner. (RM12540)
  Ensure to keep commit or abort decisions on CAMO or Eager All Node transactions in bdr.node_pre_commit for longer than 15 minutes if there is at least one node that has not learned the decision and may still query it. This eliminates a potential for inconsistency between the CAMO origin and partner nodes.

- Resolve deadlocked CAMO or Eager transactions (RM12903, RM12910)
  Add a `lock_timeout` as well as an abort feedback to the origin node to resolve distributed deadlocking due to conflicting primary key updates. This also prevents frequent restarts and retries of the PGL writer process for Eager All Node and sync CAMO transactions.

- Fix potential divergence by concurrent updates on toasted data from multiple nodes (RM11058)
  This can occur when an UPDATE changes one or more toasted columns, while a concurrent, but later UPDATE commits on a different node. This occurs because PostgreSQL does not WAL log TOAST data if it wasn't changed by an UPDATE command. As a result the logically decoded rows have these columns marked as unchanged TOAST and don't contain the actual value. Fix is handled automatically on BDR-EE, but on BDR-SE additional triggers need to be created on tables that publish updates and that have toastable data (this is also done automatically). The additional check has a small but measurable performance overhead. Logged data will increase in affected cases only. We recommend tuning `toast_tuple_target` to optimize storage. Tables with `REPLICA IDENTITY FULL` are not affected by this issue or fix.

- Properly close connections after querying camo partner to avoid leak. (RM12572)

- Correct `bdr.wait_for_apply_queue` to respect the given LSN (RM12552)
  In former releases, the `target_lsn` argument was overridden and the function acted the same as if no `target_lsn` had been given.

- Ignore progress messages from unknown nodes (RT66461)
  Avoids problems during node parting.

- Make bdr.xact_replication work with ALTER TABLE and parallel query (RM12489)

BDR 3.6.12 (2019 Dec 18)

BDR 3.6.12 is the twelfth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified previously.

Improvements

- Apply `check_full_row` on `DELETE` operations (RT66493)
  This allows detection of `delete_recently_updated` conflict even if the `DELETE` operation happened later in wall-clock time on tables with full row checking enabled.

- Improve Global DML lock tracing
  Add more information to the Global DML Lock trace to help debugging global locking issues more effectively.

- Validate replication sets at join time. (RM12020, RT66310)
  Raise an ERROR from `bdr.join_node_group()` if the joining node was configured to subscribe to non-default replication sets by using `bdr.alter_node_replication_sets()` before join but some of the subscribed-to replication sets are missing.

  On prior releases the joining node might fail later in the join process and have to be force-parted. Or it might appear to succeed but join with empty tables.

Resolved Issues

- Fix crash in `bdr.run_on_all_nodes` (RM12114, RT66515)
  Due to incorrect initialization the `bdr.run_on_all_nodes` could have previously crashed with segmentation fault in presence of `PARTED` nodes.

- Don't broadcast the epoch consumed WAL messages (RM12042)
  Broadcasting the message to all nodes could result in some nodes moving the Global DDL Lock Epoch forward in situations where it wasn't safe to do so yet, resulting in lowered protection against concurrent DML statements when running a statement that requires a Global DML Lock.

- Fix global locking on installations with multiple BDR nodes on single PostgreSQL instance
  The global locking could get spurious timeouts because the lock messages contained wrong node id if there were more than one BDR node on a single PostgreSQL instance.

- Fix typos in some example SQL in docs

BDR 3.6.11 (2019 Dec 04)

BDR 3.6.11 is the eleventh minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified

previously.

Improvements

- Support APIs for PostgreSQL 11.6
- Allow the use of "-"(hyphen) character in the node name (RM11567, RT65945)
  If a pglogical3 node would have been created with a hyphen in the node name BDR couldn't create the node on that database.
- Don't generate `update_origin_change` conflict if we know the updating node has seen the latest local change (RM11556, RT66145)
  Reduces conflict resolution overhead and logging of `update_origin_change` when the conflict can be shown to be false-positive. This does not completely remove false-positives from `update_origin_change` but reduces their occurrence in presence of UPDATES on older rows. This reduces conflict log spam when origin changes for older rows. Also, conflict triggers will be called significantly fewer times.
- Extend `bdr.wait_for_apply_queue` to wait for a specific LSN (RM11059, RT65827)
- Add new parameter `bdr.last_committed_lsn` (RM11059, RT65827)
  Value will be reported back to client after each COMMIT, allowing applications to perform causal reads across multiple nodes.
- Add status query functions for apply and received LSN (RM11059, RM11664)
  New functions `bdr.get_node_sub_receive_lsn` and `bdr.get_node_sub_apply_lsn` simplify fetching the internal information required for HAproxy health check scripts.
- Add sum() and avg() aggregates for CRDT types (RM11592, RT66168)
- Speed up initial synchronization of replication slots on physical standby (RM6747)
- Add `bdr.pg_xact_origin` function to request origin for an xid (RM11971)
- Add `bdr.truncate_locking` configuration option which sets the `TRUNCATE` command's locking behavior (RT66326)
  This configuration option determines whether (when `true`) `TRUNCATE` obeys the `bdr.ddl_locking` setting which is the new, safe behavior or if (when `false`, the default) never does any locking, which is the old, potentially unsafe behavior.
- Allow conflict triggers to see commit timestamp of `update_recently_deleted` target rows (RM11808, RT66182)

Resolved Issues

- Add hash/equality opclass for the column_timestamps data type (RT66207)
  REPLICA IDENTITY FULL requires comparison of all columns of a tuple, hence column_timestamps data type must support equality comparisons.
- Correct conflict docs for BDR-EE (RT66239, RM9670)
  Changes made in BDR3.5 were not correctly reflected in conflict docs
- Don't check protocol version for galloc sequences during initial sync (RM11576, RT65660)
  If galloc sequences already exist, bdr_init_physical doesn't need to recheck protocol versions.
- Fix galloc sequence chunk tracking corruption on lagging nodes (RM11933, RT66294)
  In presence of node with lagging consensus the chunk tracking table would diverge on different nodes which then resulted in wrong chunks being assigned on consensus leader change. As a result node might start generating already used sequence numbers. This fix ensures that the table never diverges.
- Fix galloc sequence local chunk information corruption (RM11932, RT66294)

Make sure we correctly error out when in all cases request of new chunk has failed, otherwise we might assign bogus chunks to the sequence locally which would result in potentially duplicate sequence numbers generated on different nodes.

- Fix a case where the consensus worker event loop could stall in the message broker when trying to reconnect to an unreachable or unresponsive peer node by being more defensive about socket readability/writeability checks during the libpq async connection establishment phase. (RM11914)

  This issue is most likely to arise when a peer node's underlying host fails hard and ceases replying to all TCP requests, or where the peer's network blackholes traffic to the peer instead of reporting a timely ICMP Destination Unreachable message.

  Effect of the issue on affected nodes would result in operations which require consensus to either stall or not work at all - those include: DDL lock acquisition, Eager transaction commit, calling `bdr.get_consensus_status()` function, galloc sequence chunk allocation, leader election and BDR group slot advancing. This could have been visible to users as spurious lock timeout errors or increased lag for the BDR group slot.

- Fix a race condition with global locks and DML (RM12042)
  Prevent mismatching ordering of lock operations against DML with three or more concurrently writing nodes. This allows to properly protect a TRUNCATE against concurrent DML from multiple writer nodes.

- Repeat reporting of `local_node_id` to support transparent proxies (EE) (RM12025, RM12033)
  With CAMO enabled, BDR reports a `bdr.local_node_id` GUC to the client. To fully support transparent proxies like HAproxy, BDR now reports this value once per transaction in combination with `transaction_id`, to ensure a client doesn't ever return incorrect results from PQparameterStatus() because of a stale cache caused by missing a transparent connection switch.

- Fix global DDL and DML lock recovery after instance restart or crash (RM12042)
  Previous versions of BDR might not correctly block the writes against global lock if the node or apply worker restarted after the lock was acquired. This could lead to divergent data changes in case the protected command(s) were changing data concurrently.

- Fix global DDL and DML lock blocking of replication changes (RM12042)
  Previous versions of BDR would continue replication of changes to a locked table from other nodes. This could result in temporary replication errors or permanent divergent data changes if the transaction which acquired the global lock would be applied on some nodes with delay.

- Fix hang in cleanup/recovery of acquired global lock in the apply worker
  The apply worker which acquired global lock for another node could on exit leak the hanging lock which could then get "stolen" by different backend. This could cause the apply worker to wait for lock acquisition of same lock forever after restart.

- Don't hold back freezeLimit forever (EE) (RM11783)
  The Enterprise Edition of BDR holds back freeze point to ensure enough info is available for conflict resolution at all times. Make sure that we don't hold the freeze past xid wraparound warning limit to avoid loss of availability. Allow the limit to move forward gracefully to avoid risk of vacuum freeze storms.

- Properly close connections in `bdr.run_on_all_nodes`
  Removes log spam about connection reset by peer when `bdr.run_on_all_nodes` is used.

- Clarify docs that CREATE MATERIALIZED VIEW is not supported yet. (RT66363)

BDR 3.6.10 (2019 Oct 29)

BDR 3.6.10 is the tenth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified previously.

Improvements

- Add new optional performance mode for CAMO - remote_write (EE) (RM6749)

This release enables a CAMO remote_write mode offering quicker feedback at time of reception of a pre-commit message from the CAMO partner, rather than only after the application of the transaction. Significantly better performance in exchange for small loss of robustness.

- Defer switching to CAMO mode until the partner has caught up (EE) (RM9605/RT65000/RT65827)
  In async mode for improved availability, CAMO allows to switch to a local mode in case the CAMO partner is not reachable. When switching back, it may have to catchup before it can reasonably confirm new transactions from its origin. The origin now uses an estimate of the catchup time to defer the switch back to CAMO mode to eliminate transactions timing out due to the CAMO partner still catching up.

- Add functions wait_for_apply_queue and wait_for_camo_partner_queue (EE)
  Allows to wait for transactions already received but currently queued for application. These can be used to prevent stale reads on a BDR node replicated to in `remote_write` mode.

- Improve monitoring of replication, especially around catchup estimates for peer nodes (EE) (RM9798)
  Introduce two new views `bdr.node_replication_rates` and `bdr.node_estimates` to get a reasonable estimate of how far behind a peer node is in terms of applying WAL from this local node. The `bdr.node_replication_rates` view gives an overall picture of the outgoing replication activity in terms of the average apply rate whereas the `bdr.node_estimates` focuses on the catchup estimates for peer nodes.

- Support Column-Level Conflict Resolution for partitioned tables (EE) (RM10098, RM11310)
  Make sure that newly created or attached partitions are setup for CLCD if their parent table has CLCD enabled.

- Take global DML lock in fewer cases (RM9609).
  Don't globally lock relations created in current transaction, and also relations that are not tables (for example views) as those don't get data via replication.

Resolved Issues

- Disallow setting `external` storage parameter on columns that are part of a primary key (RM11336).
  With such a setting, any `UPDATE` could not be replicated as the primary key would not get decoded by PostgreSQL.

- Prevent ABA issue with `check_full_tuple = true`. (RM10940, RM11233)
  We only do the full row check if `bdr.inc_row_version()` trigger exists on a table from now on to prevent ABA issue when detecting conflict on UPDATEs that didn't change any data when `check_full_tuple` is set to `true`.

BDR 3.6.9 (2019 Oct 10)

BDR 3.6.9 is the ninth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified previously.

Improvements

- Parameters to help BDR assessment by tracking certain application events (EE) bdr.assess_update_replica_identity = IGNORE (default) | LOG | WARNING | ERROR Updates of the Replica Identity (typically the Primary Key) bdr.assess_lock_statement = IGNORE (default) | LOG | WARNING | ERROR Two types of locks that can be tracked are:

  ```
  * explicit table-level locking (LOCK TABLE ...) by user sessions
  * explicit row-level locking (SELECT ... FOR UPDATE/FOR SHARE) by user sessions
  ```

  (RM10812,RM10813)

Resolved Issues

- Fix crash MIN/MAX for gsum and pnsum CRDT types (RM11049)

- Disallow parted nodes from requesting bdr.part_node() on other nodes. (RM10566, RT65591)

BDR 3.6.8 (2019 Sep 23)

BDR 3.6.8 is the eighth minor release of the BDR 3.6 series. This release includes a fix for a critical data loss issue as well as fixes for other issues identified with previous releases.

Improvements

- Create the `bdr.triggers` view (EE) (RT65773) (RM10874)
  More information on the triggers related to the table name, the function that is using it, on what event is triggered and what's the trigger type.

Resolved Issues

- Loss of TOAST data on remote nodes replicating UPDATEs (EE) (RM10820, RT65733)
  A bug in the transform trigger code path has been identified to potentially set toasted columns (very long values for particular columns) to NULL when applying UPDATEs on remote nodes, even when transform triggers have never been used. Only BDR-EE is affected and only when tables have a toast table defined and are not using REPLICA IDENTITY FULL. BDR3 SE is not affected by this issue. LiveCompare has been enhanced with damage assessment and data recovery features, with details provided in a separate Tech Alert to known affected users. This release prevents further data loss due to this issue.

- CAMO: Eliminate a race leading to inadvertent timeouts (EE) (RM10721)
  A race condition led to pre-commit confirmations from a CAMO partner being ignored. This in turn caused inadvertent timeouts for CAMO-protected transactions and poor performance in combination with `synchronous_replication_availability` set to `async`. This fixes an issue introduced with release 3.6.7.

- CAMO: Properly handle transaction cancellation at COMMIT time (EE) (RM10741)
  Allow the COMMIT of a CAMO-protected transaction to be aborted (more gracefully than via node restart or PANIC). Enable run-time reconciliation with the CAMO partner to make the CAMO pair eventually consistent.

- CAMO: Ensure the status query function keeps CAMO enabled. (EE) (RM10803)
  The use of the `logical_transaction_status` function disabled CAMO for the entire session, rather than just for the query. Depending on how a CAMO client (or a proxy in between) used the session, this could lead to CAMO being inadvertently disabled. This has been fixed and CAMO remains enabled independent of calls of this function.

- Eager: cleanup stale transactions. (EE) (RM10595)
  Ensures transactions aborted during their COMMIT phase are cleaned up eventually on all nodes.

- Correct TransactionId comparison when setting VACUUM freeze limit.
  This could lead to ERROR: cannot freeze committed xmax for a short period at xid wrap, causing VACUUMs to fail. (EE) (RT65814, RT66211)

BDR 3.6.7.1

This is a hot-fix release on top of 3.6.7.

Resolved Issues

- Prevent bogus forwarding of transactions from a removed origin. (RT65671, RM10605)
  After the removal of an origin, filter transactions from that origin in the output plugin, rather than trying to forward them without origin information.

BDR 3.6.7 (2019 Aug 30)

BDR 3.6.7 is the seventh minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified previously.

Improvements

- CAMO and Eager switched to use two-phase commit (2PC) internally.
  This is an internal change that made it possible to resolve a deadlock and two data divergence issues (see below). This is a node-local change affecting the transaction's origin node exclusively and has no effect on the network protocol between BDR nodes. BDR nodes running CAMO now require a configuration change to allow for enough `max_prepared_transactions`; see [Upgrading] for more details. Note that there is no restriction on the use of temporary tables, as exists in explicit 2PC in PostgreSQL.

- Add globally-allocated range sequences (RM2125)
  New sequence kind which uses consensus between nodes to assign ranges of sequence numbers to individual nodes for each sequence as needed. Supports all of smallint, integer and bigint sequences (that includes serial column type).

- Implement Multi-Origin PITR Recovery (EE) (RM5826)
  BDR will now allow PITR of all or some replication origins to a specific point in time, providing a fully consistent viewpoint across all subsets of nodes. For multi-origins, we view the WAL stream as containing multiple streams all mixed up into one larger stream. There is still just one PIT, but that will be reached as different points for each origin separately. Thus we use physical WAL recovery using multiple separate logical stopping points for each origin. We end up with one LSN "stopping point" in WAL, but we also have one single timestamp applied consistently, just as we do with "single origin PITR".

- Add `bdr.xact_replication` option for transaction replication control
  Allows for skipping replication of whole transaction in a similar way to what `bdr.ddl_replication` does for DDL statements but it affects all changes including `INSERT/UPDATE/DELETE`. Can only be set via `SET LOCAL`. Use with care!

- Prevent accidental manual drop of replication slots created and managed by BDR

- Add `bdr.permit_unsafe_commands` option to override otherwise disallowed commands (RM10148)
  Currently overrides the check for manual drop of BDR replication slot in the Enterprise Edition.

- Allow setting `bdr.ddl_replication` and `bdr.ddl_locking` as `bdr_superuser` using the `SET` command
  This was previously possible only via the wrapper functions `bdr.set_ddl_replication()` and `bdr.set_ddl_locking()` which are still available.

- Improve performance of consensus messaging layer (RM10319, RT65396)

Resolved Issues

- Delete additional metadata in `bdr.drop_node` (RT65393, RM10346)
  We used to keep some of the local node info behind which could prevent reuse of the node name.

- Correctly synchronize node-dependent metadata when using `bdr_init_physical` (RT65221, RM10409)
  Synchronize additional replication sets and table membership in those as well as stream triggers and sequence information in

`bdr_init_physical`, in a similar way to logical synchronization.

- Fix potential data divergence with CAMO due to network glitch (RM#10147)
  This fixes an data inconsistency that could arise between the nodes of a CAMO pair in case of an unreachable or unresponsive (but operational) CAMO partner and a concurrent crash of the CAMO origin node. An in-flight COMMIT of a transaction protected by CAMO may have ended up getting committed on one node, but aborted on the other, after both nodes are operational and connected.

- Fix a potential deadlock between a cross-CAMO pair (RM#7907)
  With two nodes configured as a symmetric CAMO pair, it was possible for the pair to deadlock, if both nodes were down and restarting, but both having CAMO transactions in-flight.

- Fix potential data divergence for Eager transaction in face of a crash (RM#9907)
  In case of a crash of the origin node of an Eager transaction just before the final local commit, such a transaction could have ended up aborted on the origin but committed on all other nodes. This is fixed by using 2PC on the origin node as well and properly resolving in-flight Eager transaction after a restart of the origin node.

- Correct handling of fast shutdown with CAMO transactions in-flight (RM#9556)
  A fast shutdown of Postgres on a BDR node that's in the process of committing a CAMO-protected transaction previously led to a PANIC. This is now handled gracefully, with the in-flight CAMO transaction still being properly recovered after a restart of that node.

BDR 3.6.6 (2019 Aug 08)

BDR 3.6.6 is the sixth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified previously.

Improvements

- Add `bdr.drop_node()` (RM9938)
  For removing node metadata from local database, allowing reuse of the node name in the cluster.

- Include `bdr_init_physical` in BDR-SE (RM9892)
  Improves performance during large node joins - BDR-EE has included this tool for some time.

- Enhance bdr_init_physical utility in BDR-EE (RM9988) Modify bdr_init_physical to optionally use selective pg_basebackup of only the target database as opposed to the earlier behavior of backup of the entire database cluster. Should make this activity complete faster and also allow it to use less space due to the exclusion of unwanted databases.

- TRUNCATE is now allowed during eager replicated transactions (RM9812)

- New `bdr.global_lock_table()` function (RM9735).
  Allows explicit acquire of global DML lock on a relation. Especially useful for avoidance of conflicts when using TRUNCATE with concurrent write transactions.

- New conflict type `update_pkey_exists` (RM9976)
  Allows conflict resolution when a `PRIMARY KEY` was updated to one which already exists on the node which is applying the change.

- Reword DDL locking skip messages and reduce log level
  The previous behavior was too intrusive.

- Add `bdr.apply_log_summary` (RM6596)
  View over `bdr.apply_log` which shows the human-readable conflict type and resolver string instead of internal id.

- Add `bdr.maximum_clock_skew` and `bdr.maximum_clock_skew_action` configuration options (RM9379)

For checking clock skew between nodes and either warning or delaying apply in case the clock skew is too high.

Resolved Issues

- Move CRDT type operators from public schema to `pg_catalog` (RT65280, RM10027)
  Previously BDR operators were installed in public schema, preventing their use by servers implementing stricter security policies. No action required.

- Remember if unsupported Eager Replication command was run in current transaction.
  This allows us to prevent situations where an unsupported command was run while Eager Replication was turned off and later in the transaction the Eager Replication is turned on.

- Fix the " `!` " operator for `crdt_pnsum` data type (RM10156)
  It's the operator for resetting the value of the column, but in previous versions the reset operation didn't work on this type.

BDR 3.6.5 (2019 Jul 19)

BDR 3.6.5 is the fifth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.4.

Improvements

- Allow late enabling of CAMO (RM8886)
  The setting `bdr.enable_camo` may now be turned on at any point in time before a commit, even if the transaction already has an id assigned.

- Add version-2 KSUUIDs which can be compared using simple comparison operators (RM9662)

- New `delete_recently_updated` conflict type (RM9673/RT65063) Triggered by `DELETE` operation arriving out of order - the `DELETE` has an older commit timestamp than the most recent local `UPDATE` of the row. Can be used to override the default policy of `DELETE` always winning.

- Make bdr admin function replication obey DDL replication filters (RT65174)
  So that commands like `bdr.replication_set_add_table` don't get replicated to a node which didn't replicate `CREATE TABLE` in the first place.

- Don't require group replication set to be always subscribed by every node (RT65161)
  Since we now apply DDL replication filters to admin functions, it's no longer necessary to force group replication set to be subscribed by every node as other replication sets can be configured to replicate the admin function calls.

- Allow a few more DDL operations to skip the global DML lock
  The following DDL operations have been optimized to acquire only a global DDL lock, but not the DML one:

  ```
  - ALTER TABLE .. ALTER COLUMN .. SET STATISTICS
  ```

  - ALTER TABLE .. VALIDATE CONSTRAINT
    - ALTER TABLE .. CLUSTER ON
    - ALTER TABLE .. RESET

  - CREATE TRIGGER

- Add new BDR trigger that resolves Foreign Key anomalies on DELETE (RM9580)
- Add new function bdr.run_on_all_nodes() to assist monitoring and diagnostics (RM9945)
- Extend the CAMO reference client in C
  Allow setting a `bdr.commit_scope` for test transactions.
- Prevent replication of CLUSTER command to avoid operational impact
- To assist with security and general diagnostics, any DDL that skips replication or global DDL locking at user request will be logged. For regular users of non-replicated and/or non-logged DDL this may increase log volumes. Some log messages have changed in format. This change comes into effect when `bdr.ddl_locking = off` and/or `bdr.ddl_replication = off`.
- Greatly enhance descriptions of BDR admin functions with regard to (RM8345)
  their operational impact, replication, locking and transactional nature
- Detailed docs to explain concurrent Primary Key UPDATE scenarios (RM9873/RT65156)
- Document examples of using bdr.replication_set_add_ddl_filter() (RM9691)

Resolved Issues

- Rework replication of replication set definition (RT64451)
  Solves the issue with the replication set disappearing from some nodes that could happen in certain situations.
- Acquire a Global DML lock for these DDL commands for correctness (RM9650)
  - CREATE UNIQUE INDEX CONCURRENTLY
  - DROP INDEX CONCURRENTLY
  - bdr.drop_trigger() admin function since adding or removing any constraint could allow replication-halting DML
- Correctly ignore nodes that are parting or parted in the Raft code (RM9666/RT64891)
  Removes the excessive logging of node membership changes.
- Don't try to do CRDT/CLCD merge on `update_recently_deleted` (RM9674)
  It's strictly row-level conflict; doing a merge would produce the wrong results.
- Allow BDR apps that require standard_conforming_strings = off (RM9573/RT64949)
- Use replication slot metadata to postpone freezing of rows (RM9670) (EE-only)
  Otherwise an update_origin_change conflict might get undetected after a period of node downtime or disconnect. The SE version can only avoid this using parameters.
- Correct bdr_wait_slot_confirm_lsn() to wait for the LSN of last commit, rather
  than the LSN of the current write position. In some cases that could have released the wait earlier than appropriate, and in other cases it might have been delayed.

BDR 3.6.4 (2019 Jun 20)

BDR 3.6.4 is the fourth minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.3.

The Highlights of BDR 3.6.4

- Apply statistics tracking (RM9063)
  We now track statistics about replication and resource use for individual subscriptions and relations and make them available in the `pglogical.stat_subscription` and `pglogical.stat_relation` views. The tracking can be configured via the `pglogical.stat_track_subscription` and `pglogical.stat_track_relation` configuration parameters.
- Support CAMO client protocol with Eager All Node Replication

Extend `bdr.logical_transaction_status` to be able to query the status of transactions replicated in global commit scope (Eager All Node Replication). Add support for Eager All Node Replication in the Java CAMO Reference client.

Resolved Issues

- Fix initial data copy of multi-level partitioned tables (RT64809)
  The initial data copy used to support only single level partitioning; multiple levels of partitioning are now supported.
- Don't try to copy initial data twice for partitions in some situations (RT64809)
  The initial data copy used to try to copy data from all tables that are in replication sets without proper regard to partitioning. This could result in partition data being copied twice if both the root partition and individual partitions were published via the replication set. This is now solved; we only do the initial copy on the root partition if it's published.
- Fix handling of indexes when replicating INSERT to a partition (RT64809)
  Close the indexes correctly in all situations.
- Improve partitioning test coverage (RM9311)
  In light of the partitioning related issues, increase the amount of automated testing done against partitioned tables.
- Fix merging of `crdt_pnsum` data type (RT64975)
  The internal index was handled wrongly, potentially causing a segmentation fault; this is now resolved.
- Fix cache lookup failed on database without BDR extension installed (RM9217)
  This could previously lead to errors when dropping tables on a PostgreSQL instance which has the BDR library loaded but does not have the extension installed.
- Fix permission issues on `bdr.subscription_summary` (RT64945)
  No need to have permissions on `pglogical.get_sub_progress_timestamp()` to use this view anymore.
- Cleanup prepared Eager All Node transactions after failures (RM8996)
  Prevents inconsistencies and hangs due to unfinished transactions after node or network failures. Uses Raft to ensure consistency between the nodes for the cleanup of such dangling prepared transactions.

Other Improvements

- The `replicate_inserts` option now affects initial COPY
  We now do initial copy of data only if the table replicates inserts.
- Lower log level for internal node management inside Raft worker (RT64891)
  This was needlessly spamming logs during node join or parting.
- Warn when executing DDL without DDL replication or without DDL locking
  The DDL commands executed without DDL replication or locking can lead to divergent databases and cause replication errors so it's prudent to warn about them.
- Allow create statistics without dml lock (RM9507)
- Change documentation to reflect the correct default settings for the update_missing conflict type.

BDR 3.6.3 (2019 May 30)

BDR 3.6.3 is the third minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.2.

The Highlights of BDR 3.6.3

- Add btree/hash operator classes for CRDT types (EE, RT64319) This allows the building of indexes on CRDT columns (using the scalar value) and the querying of them them using simple equality/inequality clauses, using the in GROUP BY clauses etc.

- Add implicit casts from int4/int8 for CRDT sum types (EE, RT64600)
  To allow input using expressions with integer and CRDT sum types together. For example:

```
CREATE TABLE t (c bdr.crdt_gsum NOT NULL DEFAULT
0);
```

- New `update_recently_deleted` conflict type (RM8574)
  Conflicts are handled differently for the special case of `update_missing` when BDR detects that the row being updated is missing because it was just recently deleted. See UPDATE/DELETE Conflicts in the documentation for details.

- Allow DDL operations in CAMO protected transactions, making automatic disabling of CAMO obsolete (EE, RT64769)

- Add the connection status checking function `bdr.is_camo_partner_connected` for CAMO (EE). See the Commit At Most Once documentation for details.

- Persist the last_xact_replay_timestamp (RT63881)
  So that it's visible even if the subscription connection is down (or remote node is down).

- Major documentation improvements
  Copy-edit sentences to make more sense, add extra clarifying info where the original wording was confusing.

Resolved Issues

- Use group locking for global DML lock (RT64404)
  This allows better cooperation between the global DML locker and the writers which are doing catch up of the remaining changes.

Other Improvements

- Support mixed use of legacy CRDT types and new CRDT types which are in bdr schema
  Implicitly cast between the two so their mixed usage and potential migration is transparent.
- Improve static code scanning
  Every build is scanned both by Coverity and Clang scan-build.
- Log changes of `bdr.ddl_replication` and `bdr.ddl_locking`
  Helps with troubleshooting when divergent DDL was run.
- Rework documentation build procedure for better consistency between HTML and PDF documentation. This mainly changes the way docs are structured into chapters so that there is a single source of chapter list and ordering for both PDF and HTML docs.

BDR 3.6.2 (2019 May 09)

BDR 3.6.2 is the second minor release of the BDR 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.1

The Highlights of BDR 3.6.2

- All the SQL visible interfaces are now moved to the `bdr` schema (EE)
  The CRDT types and per column conflict resolution interfaces are now in the `bdr` schema instead of `bdr_crdt` and `bdr_conflicts`. The types and public interfaces still exist in those schemas for compatibility with existing installations, however their use is not recommended as they are now deprecated and may be removed in a future release. Please use the ones in `bdr` schema. Documentation only contains references to the `bdr` schema now as well.
- Add `bdr.node_conflict_resolvers` view (RT64388)
  Shows current conflict resolver setting for each conflict type on the local node including the defaults.
- Add a CAMO Reference Client implementation in C and Java to the documentation.
- Support DEFERRED UNIQUE indexes
  They used to work only in limited cases before this release.

Resolved Issues

- Fix consensus request timeout during leader election (RT64569)
  The timeout wasn't applied when the leader was unknown leading to immediate failures of any action requiring consensus (for example global DDL locking). This is now resolved.
- Improve cleanup on failure during a DDL locked operation, This speeds up DDL locking subsystem recovery after error so that errors don't create a cascading effect.
- Unify the replication of admin function commands (RT64544)
  This makes the replication and locking behavior of administration function commands more in-line with DDL in all situations, including logical standby.
- Support covering UNIQUE indexes (RT64650)
  Previously, the covering UNIQUE indexes could result in ambiguous error messages in some cases.
- Switch to monotonic time source for Raft timing (RM6390)
  This improves reliability of Raft internal timing in presence of time jumps caused by NTPd and similar. As a result Raft reliability is improved in general.
- Improve locking in the internal connection pooler
  For more reliable messaging between nodes.
- Restore consensus protocol version on restart (RT64526)
  This removes the need for renegotiation every time a consensus worker or a node is restarted, making the features depending on newer protocol version consistently available across restarts.
- Correct automatic disabling and re-enabling of bdr.enable_camo when using DDL in a transaction. Ensure it cannot be manually re-enabled within the same transaction.
- Fix handling of CAMO confirmations arriving early, before the origin starts to wait. This prevents timeouts due to such a confirmation being ignored.

BDR 3.6.1 (2019 Apr 23)

BDR 3.6.1 is the first minor release of the BDR 3.6 series. This release includes minor new features and fixes including all the fixes from 3.6.0.1 and 3.6.0.2.

The highlights of 3.6.1

- Add `bdr.role_replication` configuration option (RT64330)
  The new option controls the replication of role management statements (`CREATE/ALTER/DROP/GRANT ROLE`). This option is dependent on `bdr.ddl_replication` as the role management statements still obey the standard rules of the DDL replication. By default this is set to `on` , meaning that these statements are replicated if executed in a BDR-enabled database.
- Add `--standby` option to `bdr_init_physical` (RM8543, EE)
  Allows the creation of a logical standby using `bdr_init_physical` ; previously only a full blown send/receive node could be created this way.
- Add `last_xact_replay_timestamp` to `bdr.subscription_summary` (RT63881)
  Shows the commit timestamp of the last replayed transaction by the subscription.
- Stop join on unrecoverable error (RT64463)
  Join might fail during the structure synchronization, which currently is an unrecoverable error. Instead of retrying like for other (transient) errors, just part the joining node and inform the user that there was an error.

Resolved Issues

- Improve the trigger security checking (RT64412)
  Allow triggers to have a different owner than the table if the trigger uses bdr or pglogical trigger functions, security definer functions (as those redefine security anyway) and also always allow replication set membership changes during initial replication set synchronization during the node join.
- Make BDR replicated commands obey `bdr.ddl_replication` (RT64479)
  Some of the BDR function calls (like `bdr_conflicts.column_timestamps_enable`) are replicated in a similar way as normal DDL commands including the DDL locking as appropriate. These commands in previous versions of BDR however ignored the `bdr.ddl_replication` setting and were always replicated. This is now fixed. In addition just like normal DDL, these commands are now never replicated from the logical standby.
- Don't try to replicate generic commands on global objects

Several commands on global objects would be replicated even in situations where they shouldn't be because of how they are represented internally. Handling of the following commands has been fixed:
    - `ALTER ROLE/DATABASE/TABLESPACE ... RENAME TO`
    - `ALTER DATABASE/TABLESPACE ... OWNER TO`
    - `COMMENT ON ROLE/DATABASE/TABLESPACE`
    - `SECURITY LABEL ON ROLE/DATABASE/TABLESPACE`

- Properly timeout on CAMO partner and switch to Local mode (RT64390, EE)
  Disregard the connection status of other BDR nodes and switch to Local mode as soon as the designated CAMO partner node fails. Makes the switch to Local mode work in a four or more node cluster.

BDR 3.6.0.2 (2019 Apr 16)

The BDR 3.6.0.2 release is the second bug-fix release in the BDR 3.6 series.

Resolved Issues

- Dynamic disabling of CAMO upon the first DDL (EE, RT64403)
- Fix hang in node join caused by timing issues when restoring Raft snapshot (RT64433)
- Fix the trigger function ownership checks (RT64412)
- Improve behavior of `promote_node` and `join_node_group` with `wait_for_completion := false`

BDR 3.6.0.1 (2019 Apr 11)

The BDR 3.6.0.1 is the first bug-fix release in the BDR 3.6 series.

Resolved Issues

- Support `target_table_missing` conflict for transparent partitioning (EE) (RT64389)
- Fix message broker sometimes discarding messages (common side-effect are DDL locking timeouts)
- Raft protocol negotiations improvements
- Fixed memory leak in tracing code
- Improve synchronous `remote_write` replication performance (RT64397)
- Fixed commit timestamp variant handling of CLCD (EE)
- Re-add support for binary protocol
- Correct Local mode for CAMO with `synchronous_replication_availability = 'async'` (EE)
- Disallow and provide a hint for unsupported operations in combination with CAMO (EE).
- Fix deadlock in `logical_transaction_status` (EE)

BDR 3.6.0 (2019 Apr 04)

The version 3.6 of BDR3 is a major update which brings improved CAMO, performance improvements, better conflict handling and bug fixes.

The highlights of BDR 3.6

- Differentiate BDR RemoteWrite mode and set `write_lsn`
- Significant replication performance improvement
    - Cache table synchronization state
    - Only send keepalives when necessary
    - Only do flush when necessary
    - Serialize transactions in fewer cases in wal sender (2ndQPostgres)

- Improved replication position reporting which is more in line with how physical streaming replication reports it
- Conflict detection and resolution improvements
    - Add new types of conflicts (like `target_table_missing`)
    - Add new types of conflict resolvers
    - Make conflict resolution configurable per node and conflict type
    - Improve conflict detection for updates

- Simplification of CAMO configuration (EE)
- Performance improvements for CAMO (EE)

Resolved issues

- Fix reporting of replay lag (RT63866)
- Fix CRDTs and conflict triggers for repeated UPDATEs of same row in transaction (RT64297)
- Don't try to replicate REINDEX of temporary indexes

Other improvements

- Improved vacuum handling of Raft tables
- Improve and clarify CAMO documentation (EE)

3 pglogical Release Notes

pglogical 3.6.33

This is a maintenance release for PGLogical 3.6 which includes fixes for issues identified previously.

- Don't replicate TRUNCATE as global message (BDR-2821, RT87453)
  The TRUNCATE command now takes the replication set into account.

Upgrades

This release supports upgrading from following versions of pglogical:

- 3.6.23 and higher
- 2.4.0 and 2.4.1
- 2.3.3 and 2.3.4

pglogical 3.6.32

This is a maintenance release for PGLogical 3.6. It is equivalent to 3.6.31, but still gets a release and a version bump to match the BDR version number.

Upgrades

This release supports upgrading from following versions of pglogical:

- 3.6.23 and higher
- 2.4.0 and 2.4.1
- 2.3.3 and 2.3.4

pglogical 3.6.31

This is a maintenance release for PGLogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Keep the `lock_timeout` as configured on non-CAMO-partner BDR nodes (BDR-1916)
  A CAMO partner uses a low `lock_timeout` when applying transactions from its origin node. This was inadvertently done for all BDR nodes rather than just the CAMO partner, which may have led to spurious `lock_timeout` errors on pglogical writer processes on normal BDR nodes.

- Prevent walsender processes spinning when facing lagging standby slots (RT80295, RT78290)
  Correct signaling to reset a latch so that a walsender process does consume 100% of a CPU in case one of the standby slots is lagging behind.

Upgrades

This release supports upgrading from following versions of pglogical:

- 3.6.23 and higher
- 2.4.0 and 2.4.1
- 2.3.3 and 2.3.4

pglogical 3.6.30

This is a maintenance release for PGLogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Push snapshot in SPI writer for every transaction (RT76368)
  This is required in newer versions of Postgres.

Upgrades

This release supports upgrading from following versions of pglogical:

- 3.6.23 and higher
- 2.4.0 and 2.4.1
- 2.3.3 and 2.3.4

pglogical 3.6.29

This is a maintenance release for PGLogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Stop replication in case of a CAMO misconfiguration (RT74906, BDR-1724)
  In case a normal BDR node was treated as a CAMO partner, but it itself is not configured as such via `bdr.camo_partner_of`, the node applied all transactional changes, but did not ever commit them. Correct this to throw a FATAL error and halt replication instead of silently ignoring transactions. This will allow for the node to resume replication once configured properly.

pglogical 3.6.28

This is a maintenance release for pglogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Don't wait on own replication slot when waiting for standby_slot_names (RT74036)
  The walsenders that use slots named in standby_slot_names should not wait for anything, otherwise we might wait forever.

- Close partitions when get replication info about tables in repsets (RT74658)
  The partitions are skipped as the replication is handeld from the parent table, these partitions were not closed, thus issuing the reference leak warning.

- Enabling async conflict resolution for explicit 2PC (BDR-1609, RT71298)
  Continue applying the transaction using the async conflict resolution for explicit two phase commit.

Improvements

- Allow upgrades from pglogical 2.4.1

pglogical 3.6.27

This is a maintenance release for pglogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Don't materialize remote tuple slot for conflict reporting (BDR-734, RT71005)
  Otherwise we might fill in defaults for any missing columns instead of keeping the existing value.

- Don't drop temporary synchronization replication slots which may still be needed by the synchronization connection (BDR-647, RT70760, RT68455, RT68352)
  Instead of doing cleanup periodically in the background, make the walsender that creates the slot responsible for the cleanup. Similar to how native temporary slots work in newer versions of PostgreSQL.

- Fix memory leak in the pglogical COPY handler (BDR-1219, RT72091)
  This fixes memory leak when synchronizing large tables.

- Fix snapshot handling around our internal executor processing (BDR-904)
  Recent changes in PostgreSQL uncovered minor issues in snapshot handling in row filtering and slot manipulation. This is mostly to improve compatibility with latest minor version of PostgreSQL.

Improvements

- Allow upgrades from pglogical 2.4.0

- Allow binary and internal protocol on more hardware combinations. This currently only affects internal testing.

pglogical 3.6.26

This is a security and maintenance release for pglogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Fix pg_dump/pg_restore execution (CVE-2021-3515) Correctly escape the connection string for both pg_dump and pg_restore so that exotic database and user names are handled correctly.
  Reported by Pedro Gallegos

- Fix potential divergence after physical failover (BDR-365, RT68894, RM19886)
  The `pglogical.standby_slot_names` setting now also protects subscriber standbys using those slots from being behind. Previously the slot on provider could move ahead of subscriber's standby causing data loss on failover of subscriber to that standby. Configuring `pglogical.standby_slot_names` on the subscriber for stanbys that are promotion targets prevents this issue now.

- Fix writer crash caused by concurrent relcache invalidation (RT70549) This crash is caused by a race with concurrent relcache invalidation deliveries. This was triggered by concurrent execution of commands that invalidate whole relcache (for example `VACUUM FULL`).

- Don't re-enter worker error handling loop recursive
  This should help make what happens clearer in any cases where we do encounter errors during error processing.

Other Changes

- Document `pglogical.alter_subscription_set_conflict_resolver`
  This functions allows configuration of conflict resolution used for the subscription.

Upgrades

This release supports upgrading from following versions of pglogical:

- 2.3.3
- 2.3.4
- 3.6.23 and higher

pglogical 3.6.25

This is a security and maintenance release for pglogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Correctly set verbosity of `pg_ctl` command when executed from `pglogical_create_subscriber`
  The `pg_ctl` was always executed in silent mode even when `-v` option was given to `pglogical_create_subscriber` in previous versions of pglogical.

  This allows for getting more meaningful troubleshooting information when analyzing issues with `pglogical_create_subscriber`.

Other Changes

- Optimize utility command processing (RT69617)
  For commands that won't affect any DB objects and don't affect pglogical we can skip the processing early without reading any pgl or system catalogs or calling to DDL replication plugin interfaces.

  This is optimization for systems with large number of such utility command calls (for example using pglogical in transaction pooling).

pglogical 3.6.24

This is a security and maintenance release for pglogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Correct cleanup of dead synchronization slots (RT69227)
  Instead of statistics, use the internal process list to reliably find backends for a given xmin.

pglogical 3.6.23

This is a security and maintenance release for pglogical 3.6 which includes fixes for issues identified previously.

Resolved Issues

- Don't replicate DDL on temporary objects (RM19491, RT69170) Don't replicate CREATE or DROP statements on objects(table, function, procedure, type and sequence) if they are on the "pg_temp" schema.

Other Changes

- Make smart shutdown of writer timeout after half of wal_receiver_timeout (RM19310) Otherwise it might get stuck forever when waiting in Postgres internal code.
- Add initial support for upgrades from 2.3.3
- Drop support for upgrading from long deprecated version 3.2, 3.3, 3.4 and 3.5

pglogical 3.6.22

This is a security and maintenance release for pglogical 3.6 which includes minor features as well as fixes for issues identified previously.

Resolved Issues

- Ensure that `pglogical.standby_slot_names` takes effect when `pglogical.standby_slots_min_confirmed` is at the default value of -1.

  On 3.6.21 and older `pglogical.standby_slot_names` was ignored if `pglogical.standby_slot_names` is set to zero (RM19042).

  Clusters satisfying the following conditions may experience inter-node data consistency issues after a provider failover:

  - Running pglogical 3.0.0 through to 3.6.21 inclusive;
  - Using pglogical subscriptions/or providers directly (BDR3-managed subscriptions between pairs of BDR3 nodes are unaffected);
  - Have a physical standby (streaming replica) of a pglogical provider intended as a failover candidate;
  - Have `pglogical.standby_slot_names` on the provider configured to list that physical standby;
  - Have left `pglogical.standby_slots_min_confirmed` unconfigured or set it explicitly to zero;

  This issue can cause inconsistencies between pglogical provider and subscriber and/or between multiple subscribers when a provider is replaced using physical replication based failover. It's possible for the subscriber(s) to receive transactions committed to the pre-promotion original provider that will not exist on the post-promotion replacement provider. This causes provider/subscriber divergence. If multiple subscribers are connected to the provider, each subscriber could also receive a different subset of transactions from the pre-promotion provider, leading to inter-subscriber divergence as well.

  The `pglogical.standby_slots_min_confirmed` now defaults to the newly permitted value `-1`, meaning "all slots listed in `pglogical.standby_slot_names`". The default of 0 on previous releases was intended to have that effect, but instead effectively disabled physical-before-logical replication.

  To work around the issue on older versions the operator is advised to set `pglogical.standby_slots_min_confirmed = 100` in `postgresql.conf`. This has no effect unless `pglogical.standby_slot_names` is also set.

  No action is generally required for this issue on BDR3 clusters. BDR3 has its own separate protections to ensure consistency during promotion of replicas.

- Fix very rare replication set cache invalidation race condition which could cause crash of a backend or walsender (RM19043, RM19244)

- Fix very rare writer relation cache invalidation race condition which could cause crash of the writer (RM19037)

Improvements

- Add `pglogical.replication_origin_status` view which allows `pglogical_superuser` role to see the status of replication origins.
  This is normally visible only to superuser in PostgreSQL itself.

- Improve `wal_receiver_timeout` handling introduced in 3.6.21
  Don't timeout on nodes that are doing table resynchronization but are oterwise idle.

pglogical 3.6.21

This is a security and maintenance release for pglogical 3.6 which includes minor features as well as fixes for issues identified previously.

Resolved Issues

- Fix segmentation fault encountered when adding a partitioned table with many partitions to replication set (RM15733, RT68352)
  The segmentation fault was caused by a cache entry of one of the partitions invalidated during copying data to the subscriber. This has been fixed by using a valid cache entry for this purpose.

- Fix a failure encountered when adding a large table to replication set (RM18154, RT68455)
  pglogical.replication_set_add_table() may fail to add a large table containing millions of rows to a replication set. Similar failure may be seen if the replication takes longer say due to a slow network between publisher and subscriber. Server logs of the subscriber node will indicate START_REPLICATION SLOT command failing with ERROR 42704 "replication slot does not exist". This failure has been fixed by correcting the logic which periodically removes unused replication slots created for synchronizing table being added to the replication set.

- Fix crash when running replica triggers on partitions (RM18252)

- Prohibit INSERT ON CONFLICT and MERGE (2ndQPostgres) commands on tables without replica identity (RM17323, RT68146)
  These commands might end up doing UPDATE or DELETE which would break replication when table does not have any replica identity.

- Fix memory leak in executor state cache during the initial data COPY (RM17668)
  This was particularly problematic when adding large tables to replication set.

- Fix memory leak in writer INSERT processing (RM17668)
  Resulted in unusually large memory use when applying of INSERTs that affected many rows.

- Fix race condition in invalidation handling of local_sync_status (RM17929)
  This could result in receiver waiting forever for the table resynchronization triggered by `pglogical.alter_subscription_resynchronize_table()` to finish.

- Fix rare race condition where reported flush lsn would be ahead of apply lsn (RM18044)
  This would mostly cause monitoring queries on provider to show odd values.

Improvements

- Document limitations of using `primary_conninfo` for slot synchronization to a standby (RM14612, RT67443)

- Make PGL receiver respect `wal_receiver_timeout` (RM13805, RT67066)
  After an unclean disconnect, the receiver process now terminates once the `wal_receiver_timeout` is exceeded. This allows it to be restarted and then attempt to reconnect. Prior to this release, the TCP expiration time of the OS applied.

pglogical 3.6.20

This is a security and maintenance release for pglogical 3.6 which includes minor features as well as fixes for issues identified previously.

Resolved Issues

- Only process keepalives in writer if they come outside of transaction (RT67858)
  Keepalives sent in middle of forwarded transactions could move wrong origin forward resulting in skipping future transactions from that origin.

- Use timestamp of slot snapshot for initial copy transaction (RM16396)
  We can't set commit timestamp of individual rows correctly when doing logical copy during subscription initialization because that would be too slow (every row would have to have separate transaction). But we can use the knowledge that each row had to be committed at or before the snapshot which we use to read the data was taken. So we find the last commit in that snapshot and use the timestamp of that commit. This helps with time base conflict resolution against the existing data copied during the subscription initialization.

- Keep open the connection until `pglogical_create_subscriber` finishes (RM13649)
  Set `idle_in_transaction_session_timeout` to 0 so we avoid any user setting that could close the connection and invalidate the snapshot.

Improvements

- CentOS 8 is now supported, starting with this release.

pglogical 3.6.19

This is a security and maintenance release for pglogical 3.6 which includes minor features as well as fixes for issues identified previously.

Resolved Issues

- SECURITY: Set search_path to empty for internal PGLogical SQL statements (RM15373)
  Also, fully qualify all operators used internally. PGLogical is now protected from attack risks identified in CVE-2018-1058, when the user application avoids the insecure coding practices identified there.

- Correct parsing of direct WAL messages (RT67762)
  Custom WAL messages emitted by PGLogical (or plugins building on top of it) can be broadcast or direct types. Decoding of the latter was incorrect and could in rare cases (depending on the node name) lead to "insufficient data left in message" or memory allocation errors. Decoding of such direct WAL messages has been corrected.

- Add pglogical.sync_failover_slots() function (RM14318)
  Signal the supervisor process to restart the mechanism to synchronize the failover slots specifyed in the "pglogical.synchronize_failover_slot_name".

- Fix the `--extra-basebackup-args` argument passed to pg_basebackup (RM14808)
  Corrects how the `pglogical_create_subscriber` tool passes on such extra arguments to pg_backbackup.

Improvements

- Add more diagnostic information to pglogical.queue message (RM15292)
  A new key `info` has been added to `pglogical.queue` providing additional information about a queued DDL operation.

pglogical 3.6.18

This is a maintenance release for pglogical 3.6 which includes minor features as well as fixes for issues identified previously.

Improvements

- Warn about failover issues if standby_slot_names is not set (RT66767, RM12973) If pglogical.standby_slot_names is not set and a physical standby is configured; failover to this standby will have data consistency issues as per our documentation. However, the replica could just be a simple read replica. In any case, we now warn on the replica about the potential data corruption/divergence that could result if failover is desired to such a standby.
- Check repsets in create_subscription for pgl2 upstreams also.
- Various improvements to systemtap integration.

Resolved Issues

- Prevent a hang in case of an early error in the PGL writer (RT67433, RM14678)
- Allow postgres to start with pglogical library loaded but activity suspended
  Add start_workers commandline-only GUC to facilitate this.

pglogical 3.6.17

This is a maintenance release for pglogical 3.6 which includes minor features as well as fixes for issues identified previously.

Improvements

- Make the slot synchronization to standby more configurable (RM13111)
  Added several new configuration parameters which tune the behavior of the synchronization of logical replication slots from a primary to a standby PostgreSQL servers. This allows for better filtering, inclusion of non-pglogical replication sets and also using different connection string than physical replication uses (useful when different user or database should be used to collect information about slots).

Resolved Issues

- Fix issue with UPDATEs on partitions with different physical row representation than partition root (RM13539, RT67045)
  The partitions must have same logical row as partition root they can have different physical representation (primarily due to dropped columns). UPDATEs on such partitions need to do special handling to remap everything correctly otherwise constraints and not-updated TOAST columns will refer to wrong incoming data.

- Fix truncation of `\_tmp` slot names in sync slots
  Long slot names could previously cause the temporary slot to be suffixed by `\_tm` rather than the expected `\_tmp` suffix.

Support, Diagnostic and Logging Changes

These changes don't directly change existing behaviour or add new user-facing features. They are primarily of interest to 2ndQuadrant support operations and for advanced diagnostic analysis.

- Expand non-invasive tracing (SystemTap, linux-perf, DTrace, etc) support to cover inspection of the pglogical receiver's input protocol stream, walsender output plugin protocol stream, and other useful events. (RM13517)

- Add a test and debug utility that decodes captured pglogical protocol streams into human-readable form (RM13538)

- Improve error context logging in the pglogical writer to show more information about the transaction being applied and its origin.

- Fix incorrectly reported commit lsn in errcontext messages from the pglogical heap writer (RM13796). This fix only affects logging output. The writer would report the lsn of the original forwarded transaction not the lsn of the immediate source transaction.

- Add subscription, local node and peer node names to heap writer errcontext log output.

pglogical 3.6.16

This is the sixteenth minor release of the Pglogical 3.6 series. This release includes mostly just enables BDR 3.6.16 without any significant changes to pglogical.

pglogical 3.6.15

This is the fifteenth minor release of the Pglogical 3.6 series. This release includes fixes for issues identified previously.

Resolved Issues

- Fix backwards-compatibility to PGLogical 2 (RM13333, RT66919)
  Recent releases performed additional checks during `create_subscription`, which are fine against other PGLogical 3 installations, but not backwards-compatible. This release corrects the check to account for backwards-compatibility.

- Correct a warning about GUC nest level not being reset (EE) (RM13375)
  The addition of the `lock_timeout` in 3.6.14 led to a warning being issued for CAMO and Eager All Node transaction ("GUC nest level = 1 at transaction start"). With this release, GUC nest levels are properly managed and the warning no longer occurs.

Improvements

- Add a new `pglogical.worker_tasks` view that tracks and records pglogical's background worker use. The view exposes information about the number of times a given type of worker has been restarted, how long it has been running, whether it accomplished any useful work, and more. This offers administrators more insight into pglogical's internal activity when diagnosing problems, especially when joined against the `pglogical.worker_error` table.

- Add support for rate-limiting pglogical background worker (re)launches. The new `pglogical.min_worker_backoff_delay` configuration option sets a minimum delay between launches of all types of pglogical background workers so that rapid respawning of workers cannot fill the log files and or excessive load on the system that affects other operations.

  For example, if configured with `pglogical.min_worker_backoff_delay = '500ms'`, pglogical will not retry any given background worker startup more often than twice per second (`1000/500 = 2`).

  A simple fixed-rate factor was deemed to be the most predictable and production-safe initial approach. Future enhancements may add a heuristic delay factor based on worker type, time from start to exit, number of recent launches, etc.

  The launch backoff delay defaults to 0 (off) to prevent surprises for upgrading users.

  A setting of `pglogical.min_worker_backoff_delay = '5s'` or similar is a reasonable starting point, and may become the default in a future release.

Upgrades

The PostgreSQL Global Development Group has phased out support for PostgreSQL 9.4 on all Debian based distributions. Following that, this release covers only PostgreSQL 9.5 and newer. We advise to upgrade to a newer version.

For RedHat based distributions, this release is still available for PostgreSQL 9.4.

pglogical 3.6.14

This is the fourteenth minor release of the Pglogical 3.6 series. This release includes fixes for issues identified previously.

Resolved Issues

- Resolve deadlocked CAMO or Eager transactions (RM12903, RM12910)
  Add a `lock_timeout` as well as an abort feedback to the origin node to resolve distributed deadlocking due to conflicting primary key updates. This also prevents frequent restarts and retries of the PGL writer process for Eager All Node and sync CAMO transactions.

pglogical 3.6.12

This is the twelveth minor release of the Pglogical 3.6 series. This release includes fixes for issues identified previously.

Improvements

- Add infrastructure for `check_full_row` in DELETE operations used by BDR 3.6.12 (RT66493)

- Validate requested replication sets at subscribe time (RM12020, RT66310)
  `pglogical.create_subscription()` now checks that all requested replication sets actually exist on the provider node before returning. If any are missing it will raise an ERROR like: `ERROR:  replication set(s) "nonexistent_repset" requested by subscription are missing on provider` with a DETAIL message listing the full sets requested, etc. On prior releases subscriptions with missing repsets would fail after `pglogical.create_subscription(...)` returned, during initial sync. The failure would only be visible in the logs where it is much less obvious to the user. Or if schema sync was not enable they could appear to succeed but not populate the initial table contents.

Resolved Issues

- Fix a potential deadlock at CAMO partner startup. (RM12187)
  After a restart, the CAMO partner resends all confirmations for recent CAMO protected transactions. In case these fill the internal queue between the receiver and writer processes, a deadlock was possible. This release ensures the receiver consumes pending feedback messages allowing the writer to make progress.

pglogical 3.6.11

This is the eleventh minor release of the Pglogical 3.6 series. This release includes fixes for issues identified previously.

Improvements

- Implement remote_commit_flush for CAMO. (RM11564)
  Additional level of robustness for CAMO, only replying when xact is known committed and flushed on partner node.

- Make receiver-writer shared queues of configurable size. (RM11779)
  Two new GUCs are introduced: pglogical.writer_input_queue_size (default 1MB) pglogical.writer_output_queue_size (default 1MB)

- Add a warning when user tries to set update_origin_change to skip

- Add callback to request replay progress update. (RM6747)

Resolved Issues

- Send TimeZone GUC when replicating DDL (RT66019)
  To ensure that timezone dependent expressions in DDL get evaluated to same value on all nodes.

- Only use isvalid indexes when searching for conflicts (RT66036)
  Indexes currently being created or failed index creations will be ignored, to prevent concurrency issues with change apply and `CREATE INDEX CONCURRENTLY`.

- Fix crash when replication invalidations arrive outside a transaction (RM11159)

- Make the receiver apply the queue before shutting down (RM11778)
  Upon smart shutdown, the PGL writer no longer terminates immediately, requiring queued transactions to be resent, but applies already received transactions prior to shutting down.

pglogical 3.6.10

This is the tenth minor release of the Pglogical 3.6 series. This release includes fixes for issues identified previously.

Improvements

- Add support for a CAMO remote_write mode (RM6749)

Resolved Issues

- COMMIT after initial sync of a table. This avoids treating the first catchup xact as if it was part of the initial COPY, which could lead to strange errors or false conflicts. (RM11284).
- Remove the 4 billion row limit during the initial subscription synchronization (RT66050).
- Cleanup table replication cache when replication set configuration changes.
  Previously we could use stale cache on multiple calls for table replication info on same connection if user changed the configuration in meantime. This could result in initial sync missing replicated table if the configuration was changed while the subscription was being created.
- Remember repsets when caching table replication info.
  If the client calls the table replication info with different parameters, we need to remember them otherwise we might return cached value for wrong replication sets. This could result in initial sync copying data from table which were not supposed to be replicated.

pglogical 3.6.9

This is the ninth minor release of the Pglogical 3.6 series. This release includes minor improvements.

Improvements

- Add support for local, remote_apply and remote_write. (RM11069, RT65801)
  We now accept the use of all the values that PostgreSQL accepts when configuring the "pglogical.synchronous_commit".
- Immediately forward all messages from the PGL receiver back to origin (BDR CAMO)
  Confirmations for CAMO protected transactions flow from the PGL writer applying the transaction back to origin node via the PGL receiver. This process used to consume only one confirmation message per iteration. It now consumes all pending confirmations from the PGL writer and immediately sends them back to the origin. It also decreases latency for BDR CAMO transactions in case confirmations queue up.

pglogical 3.6.8

This is the eigth minor release of the Pglogical 3.6 series. This release includes fixes for issues identified previously.

Resolved Issues

- Use RelationGetIndexAttrBitmap to get pkey columns. (RT65676, RT65797)
  No need to try to fetch pkey columns from index itself, we have relcache interface that does exactly what we need and does so in more performant way.

pglogical 3.6.7.1

This is a hot-fix release on top of 3.6.7.

Resolved Issues

- Fix a protocol violation after removal of an origin. (RT65671, RM10605)
  Removal of a replication subscription may lead to a walsender trying to forward data for unknown origins. Prevent emission of an invalid message in that case.

pglogical 3.6.7

pglogical 3.6.7 is the seventh minor release of the pglogical 3.6 series. This release includes minor new features as well as fixes for issues identified earlier.

Improvements

- Replicate TRUNCATE on a partition if only parent table is published in replication set (RT65335)
  Previously, we'd skip such TRUNCATE unless the partition was also published.
- Generate `target_table_missing` for TRUNCATE which is executed against non-existing table (RT10291)
  Allows for user-configured decision if it should be a replication-stopping issue or not.
- Improve performance of repeated UPDATEs and DELETEs executed on origin node by caching the replication configuration of tables in a user session.
- Reduce CPU usage of receiver worker when writer queue is full (RM10370).

Resolved Issues

- Fix partition replication set membership detection for multi-level partitioned tables
  Replicate changes correctly for multi-level partitioned tables, where only the intermediate partition is part of replication set (not root or leaf partitions).
- Support replication `TRUNCATE CASCADE` on tables referenced by `FOREIGN KEY` (RT65518)
  Previously this would throw error on the subscriber. This will only work if all tables on subscriber which have `FOREIGN KEY` on the table being `TRUNCATEd` are replicated tables. Also it's only supported on PostgreSQL 11 and higher.
- Flush writer between data copy and constraint restore
  Otherwise there could in some rare cases still be unapplied changes when creating constraints during initial synchronization of a subscription, potentially causing deadlocks.
- Fix potential writer queue corruption on very wide (1000+ columns) tables

Upgrades

This release supports upgrading from following versions of pglogical:

- 2.2.0
- 2.2.1
- 2.2.2
- 3.2.0 and higher

pglogical 3.6.6

pglogical 3.6.6 is the sixth minor release of the pglogical 3.6 series. This release includes minor new features as well as fixes for issues identified earlier.

Improvements

- New conflict type `update_pkey_exists` (RM9976)
  Allows resolving conflicts when a `PRIMARY KEY` was updated to one which already exists on the node which is applying the change.
- Add `pglogical.apply_log_summary` (RM6596)
  View over `pglogical.apply_log` which shows the human-readable conflict type and resolver string instead of internal id.
- Improve logging during both the initial data synchronization of a subscription and the individual table resynchronization.

Resolved Issues

- Make sure writer flushes changes after initial data copy (RT65185)
  Otherwise depending on timing and I/O load the subscription might not update positioning info and get data both via initial copy and replication stream catchup that follows.

Upgrades

This release supports upgrading from following versions of pglogical:

- 2.2.0
- 2.2.1
- 2.2.2
- 3.2.0 and higher

pglogical 3.6.5

pglogical 3.6.5 is the fifth minor release of the pglogical 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.4.

Improvements

- Improve tuple lock waits during apply for deletes (RM9569)
  This should improve performance of replication of deletes and updates in contentious situation.

Resolved Issues

- Use consistent table list in initial data copy (RM9651/RT64809) To prevent issues during initial data copy and concurrent table drop.
- Cleanup worker_dsm_handle on worker detach (internal)
  Otherwise we could leave dangling DSM segment handle for a worker after a crash, which could confuse plugins using this API.

- Fix handling of empty eager transactions (RM9550)
  In case no relevant change remains to be applied on a replica node, the prepare of such an empty transaction now works just fine.
- Fix the replication sets output in `pglogical.pglogical_node_info()`
  Previously it could be garbled.
- Reduce log level for messages when resolving ERRCODE_T_R_SERIALIZATION_FAILUREs (RM9439)

Upgrades

This release supports upgrading from following versions of pglogical:

- 2.2.0
- 2.2.1
- 2.2.2
- 3.2.0 and higher

Note that upgrades from 2.2.x are only supported on systems with `pglogical.conflict_resolution` set to `last_update_wins`.

pglogical 3.6.4

pglogical 3.6.4 is the fourth minor release of the pglogical 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.3.

New Features

- Apply statistics tracking (RM9063)
  We now track statistics about replication and resource use for individual subscriptions and relations and make them available in `pglogical.stat_subscription` and `pglogical.stat_relation` views. The tracking can be configured via `pglogical.stat_track_subscription` and `pglogical.stat_track_relation` configuration parameters.
- The `replicate_inserts` option now affects initial COPY
  We now do initial copy of data only if the table replicates inserts.

Resolved Issues

- Fix initial data copy of multi-level partitioned tables (RT64809)
  The initial data copy used to support only single level partitioning, multiple levels of partitioning are now supported.
- Don't try to copy initial data twice for partitions in some situations (RT64809)
  The initial data copy used to try to copy data from all tables that are in replication sets without proper regard to partitioning. This could result in partition data to be copied twice if both root partition and individual partitions were published via replication set. This is now solved, we only do the initial copy on the root partition if it's published.
- Fix handling of indexes when replicating INSERT to a partition (RT64809)
  Close the indexes correctly in all situations.
- Improve partitioning test coverage (RM9311)
  In light of the partitioning related issues, increase the amount of automated testing done against partitioned tables.
- Fix a leak in usage of the relation cache (RT64935)
- Fix a potential queue deadlock between writer and receiver (RT64935, RT64714)

pglogical 3.6.3

pglogical 3.6.3 is the third minor release of the pglogical 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.2.

New Features

- Support `DoNotReplicateId` special origin
  This allows correct handling of "do not replicate" origin which allows skipping replication of some changes. Primarily needed internally for other features.
- Persist the last_xact_replay_timestamp (RT63881)
  So that it's visible even if the subscription connection is down.
- Rework documentation build procedure for better consistency between HTML and PDF documentation
  This mainly changes the way docs are structured into chapters so that there is single source of chapter list and ordering for both PDF and HTML docs.

Resolved Issues

- Invalidate local cache when adding new invalidation
  Fixes visibility of changes in the catalog cache view of the transaction which did those changes. Not triggered yet by any code but will be in the future releases.
- Open indexes after partition routing
  Otherwise we might be opening indexes of the root table rather than the partition, causing issues with handling conflicts for `INSERT` operation replication.

pglogical 3.6.2

pglogical 3.6.2 is the second minor release of the pglogical 3.6 series. This release includes minor new features as well as fixes for issues identified in 3.6.1.

New Features

- Support DEFERRED UNIQUE indexes
  They used to work only in limited cases before this release.
- Support covering UNIQUE indexes (RT64650)
  The use of covering UNIQUE indexes could result in ambiguous error messages in some cases before.
- Add `--log-file` option to `pglogical_create_subscriber` (RT64129) So that log can be saved somewhere other than the current working directory

Resolved Issues

- Fix error message when the database name in the connection string in `pglogical_create_subscriber` is missing (RT64129) The previous message was ambiguous.
- Raise error when unknown parameter was specified for `pglogical_create_subscriber` (RT64129)
  Otherwise mistakes in command line arguments could be silently ignored.
- Solve timing issue with workers exiting while another one tries to start using same worker slot
  Before, we could corrupt the worker information causing the newly starting worker to crash (and having to start again later), this will no longer happen.
- Set statement time on start of every transaction in pglogical workers (RT64572)
  Fixes reporting of `xact_start` in `pg_stat_activity`

pglogical 3.6.1

pglogical 3.6.1 is the first minor release of the pglogical 3.6 series. This release includes minor new features and fixes including all the fixes from 3.6.0.1.

New Features

- Add slot failover documentation
- Add `pglogical.get_sub_progress_timestamp` for retrieving origin timestamp of the last committed change by the subscription

Resolved Issues

- Stop retrying subscription synchronization after unrecoverable error (RT64463)
  If the schema synchronization failed (which is an unrecoverable error) don't keep retrying forever. Instead mark the subscription synchronization as failed and disable the subscription.
- Improve handling and messaging with missing replication sets in output plugin (RT64451)
  Report all missing and found sets and make sure the sets are looked up using current snapshot.

pglogical 3.6.0.1

The pglogical 3.6.0.1 is the first bug-fix release in the pglogical 3.6 series.

Resolved Issues

- Improve synchronous `remote_write` replication performance (RT64397)
- Re-add support for binary protocol

pglogical 3.6.0

The version 3.6 of pglogical is a major update which brings performance improvements, better conflict handling, bug fixes and infrastructure necessary for BDR 3.6.

New Features

- Significant replication performance improvement
  - Cache table synchronization state
  - Only send keepalives when necessary
  - Only do flush when necessary
  - Serialize transactions in fewer cases in walsender (2ndQPostgres)

- Improved replication position reporting which is more in line with how physical streaming replication reports it
- Conflict detection and resolution improvements
  - Add new types of conflicts (like `target_table_missing`)
  - Add new types of conflict resolvers
  - Make conflict resolution configurable by subscription and conflict type

    - Improve conflict detection for updates

Resolved Issues

- Don't try to replicate REINDEX on temporary indexes

Other Improvements

- Fix potential message parsing error for two-phase commits
- Make initial COPY of data interruptible