High replication lag due to ReorderBufferWrite wait event

Last updated by | Francisco Javier Pardillo Martin | Jan 23, 2023 at 8:21 AM PST

Contents

Issue

Logical replication lag in a data-in logical replication configuration.

Investigation/Analysis

We found the wait event "ReorderBufferWrite" always on top on those replication processes in the destination server.

Mitigation

We recommend to review the application transactions or scheduled batch processes that could be executing big transactions (like single delete command affecting thousands of rows) and try to split those into smaller batches. This will make the PostgreSQL engine to not use the disk for those internal logical replication/decoding sorting phases.

RCA Template

The use of the event ReorderBufferWrite in destination server is related to the transaction size in terms of changes including in such transactions, when number of changes in >= 4096 PostgreSQL needs to use disk and causing those events to appear (ReorderBufferWrite, ReorderBufferRead).

You can use parameter log_min_messages=debug2 to find following entries in server logs: "spill %u changes in XID %u to disk", that shows when the PostgreSQL engine needed to use disk for those large transactions.

Please note that using the log_min_messages parameter may affect overall performance, so, remember to revert back to default value after your testing.

Repro Steps

You can reproduce the high replication lag in a logical replication configuration by executing in source server big transactions like:

```
insert into test (select i, i::text from generate_series(1,20000) i);
```

and checking in destination the replication processes that will show the wait event: "ReorderBufferWrite", also, you can enable log_min_messages to debug2 and you will observe those spill messages in destination server: "spill %u changes in XID %u to disk"

Later you can repeat the same amount of work, but in smaller transactions:

```
insert into test (select  i, i::text from generate_series(1,2000) i);

... (10 times)

insert into test (select  i, i::text from generate_series(1,2000) i);
```

And you will not observe such "ReorderBufferWrite" events in target server.

More Information

Check current source code talking about the wait event and the internal fixed number of 4096 for max_changes_in_memory:

https://github.com/postgres/postgres/blob/master/src/backend/replication/logical/reorderbuffer.c

```
/*
 * Maximum number of changes kept in memory, per transaction. After that,
 * changes are spooled to disk.
 *
 * The current value should be sufficient to decode the entire transaction
 * without hitting disk in OLTP workloads, while starting to spool to disk in
 * other workloads reasonably fast.
 *
 * At some point in the future it probably makes sense to have a more elaborate
 * resource management here, but it's not entirely clear what that would look
 * like.
 */
int                     logical_decoding_work_mem;
static const Size max_changes_in_memory = 4096; /* XXX for restore only */
```

How good have you found this content?