# Quassel IRC - Bug #1429

## Backlog fetching for channels that have not had any activity at all in a long time takes an extreme amount of time when using PostgreSQL.

03/02/2017 12:03 AM - ReimuHakurei

| | | | |
|---|---|---|---|
| **Status:** | Resolved | **Start date:** | 03/01/2017 |
| **Priority:** | Normal | **Due date:** | |
| **Assignee:** | | **% Done:** | 0% |
| **Category:** | Quassel Core | **Estimated time:** | 0.00 hour |
| **Target version:** | 0.12.4 | | |
| **Version:** | 0.12.4 | **OS:** | Any |

**Description**

My QuasselCore has a large database - right now, I have 73,778,201 rows in the backlog table. I also have a lot of buffers - several of which for channels which no longer exist on networks which also no longer exist. For obvious reasons, there has been no activity in those channels for a long time.

One such buffer is bufferid 170, whose most recent message was messageid 62,238,289. According to PostgreSQL logs, Quassel generates the following query for it (or rather, close to this; Quassel uses prepared statements according to the log):

```
SELECT messageid, time,  type, flags, sender, message FROM backlog LEFT JOIN sender ON backlog.sen
derid = sender.senderid WHERE bufferid = 170 ORDER BY messageid DESC LIMIT 5;
```

This query, according to EXPLAIN ANALYZE, takes 174 seconds (!) to run:

```
                                                                                  QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------
 Limit  (cost=1.00..28.34 rows=5 width=112) (actual time=173945.226..174009.210 rows=5 loops=1)
   ->  Nested Loop Left Join  (cost=1.00..14617727.96 rows=2673195 width=112) (actual time=173945.
223..174009.203 rows=5 loops=1)
         ->  Index Scan Backward using backlog_pkey on backlog  (cost=0.57..3082649.25 rows=267319
5 width=68) (actual time=173891.479..173912.633 rows=5 loops=1)
               Filter: (bufferid = 170)
               Rows Removed by Filter: 12429359
         ->  Index Scan using sender_pkey on sender  (cost=0.43..4.31 rows=1 width=52) (actual tim
e=19.211..19.215 rows=1 loops=5)
               Index Cond: (backlog.senderid = senderid)
 Planning time: 0.642 ms
 Execution time: 174009.380 ms
(9 rows)
```

Is a LEFT JOIN actually needed in this case? After adding the following index:

```
CREATE INDEX "backlog_messageid_backwards" ON backlog USING btree (messageid DESC);
```

...the same query, using a normal JOIN instead of a LEFT JOIN, instead takes 6.6 seconds:

```
reimu_quassel=> EXPLAIN ANALYZE SELECT messageid, time,  type, flags, sender, message FROM backlog
 JOIN sender ON backlog.senderid = sender.senderid WHERE bufferid = 170 ORDER BY messageid DESC LI
MIT 5;
                                                                               QUERY PLAN
-----------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------
 Limit  (cost=1.00..28.35 rows=5 width=112) (actual time=6582.896..6583.055 rows=5 loops=1)
   ->  Nested Loop  (cost=1.00..14624504.64 rows=2673212 width=112) (actual time=6582.894..6583.05
3 rows=5 loops=1)
         ->  Index Scan Backward using backlog_pkey on backlog  (cost=0.57..3082667.12 rows=267321
2 width=68) (actual time=6582.614..6582.718 rows=5 loops=1)
               Filter: (bufferid = 170)
               Rows Removed by Filter: 12429831
         ->  Index Scan using sender_pkey on sender  (cost=0.43..4.31 rows=1 width=52) (actual tim
```

```
e=0.035..0.036 rows=1 loops=5)
               Index Cond: (senderid = backlog.senderid)
 Planning time: 2.285 ms
 Execution time: 6583.132 ms
(9 rows)
```

In any case, some sort of optimization for backlog fetching in cases like this is clearly needed.

### History

#### #1 - 03/02/2017 10:26 AM - EgS

ReimuHakurei wrote:

> Is a LEFT JOIN actually needed in this case?

That is a very good question. I think it is not needed.

I can't recall precisely, but I guess the the original intent was to be on the safe side if a matching sender entry would not exist, or perhaps it was thought that no sender was created for server messages. Anyways, as a sender is created for any message about to be logged, such a situation should never occur and a regular JOIN should suffice as you suggested.

> After adding the following index:
>
> [...]
>
> ...the same query, using a normal JOIN instead of a LEFT JOIN, instead takes 6.6 seconds:

Did you also test the regular JOIN without the new index? Perhaps that will already suffice, and it would be an easier fix as it does not require a schema update.

Thanks and cheers,
Marcus

#### #2 - 03/02/2017 07:11 PM - ReimuHakurei

EgS wrote:

> Did you also test the regular JOIN without the new index? Perhaps that will already suffice, and it would be an easier fix as it does not require a schema update.

Yes, the 174 seconds is with the new index.

#### #3 - 03/02/2017 07:12 PM - ReimuHakurei

Oh, misread your question.

Yes, the regular JOIN speeds it up without the new index, but not by as much on my machine. It speeds it up to ~20 seconds.

#### #4 - 03/02/2017 07:37 PM - ReimuHakurei

If a column were to be added to the buffer table, 'latestmsgid', and "AND backlog.messageid <= [lastmsgid]" is added to the query using a normal JOIN, with the index, the query time drops to 19ms on my test system. However, EXPLAIN ANALYZE does not seem to actually be using the new index in this case; it reports only using backlog_pkey. 19ms is definitely preferable to 174ms.

This is definitely the optimal solution. In the next few days, I'll see if I can make a patch to the core to add this behavior to see if it actually performs that much in practice.

#### #5 - 03/03/2017 09:30 PM - EgS

I guess that latestmsgid would not be necessary if lastseenmsgid would be properly set? (cf. Bug #1431)

Anyways: thanks for your work! Let me know if there is a pull request to be merged.

Cheers,
Marcus

#### #6 - 03/04/2017 05:07 AM - ReimuHakurei

This, and #1431, should be fixed in my pull request here:

https://github.com/quassel/quassel/pull/273

#### #7 - 03/04/2017 03:40 PM - EgS

*- Status changed from New to Resolved*

Merged Merged https://github.com/quassel/quassel/pull/273