LISTEN/NOTIFY scalability benchmark

Table of Contents

Overview

The goal of this benchmark is to get a better understanding of how {a single, a hundred} concurrent NOTIFY backends in combination with {a single, a hundred} concurrent LISTEN backends, affect the pgbench tps, and using perf to understand what the bottleneck for each workload scenario is.

The benchmark has been run on Ubuntu 24.04.2 LTS running in a UTM virtual machine on a Apple M3 Max 128GB RAM.

In the perf results, a drill-down from PostgresMain is shown, where the largest branch is expanded, down to the syscall, to get an idea of what dominates.

master (b5c53b4)

1 x NOTIFY channel_1 to 1 x LISTEN channel_1

```
$ ./listen_script 1
$ pgbench -f ~/notify_channel_1.sql -c 1 -j 1 -T 60 -n bench

tps = 11544.902331 (without initial connection time)

-   98.90%     0.25%  postgres  postgres                [.] PostgresMain
   - 98.64% PostgresMain
      - 39.59% exec_simple_query
         - 32.36% CommitTransactionCommand
            - 32.20% CommitTransaction
               - 23.15% AtCommit_Notify
                  + 22.05% kill
```

```
                   + 0.83% AllocSetAllocFromNewBlock
                + 3.74% PreCommit_Notify
                + 1.30% ResourceOwnerReleaseInternal
                + 0.67% XactLogCommitRecord
           + 2.83% pg_parse_query
           + 1.66% start_xact_command
             0.53% PortalRun
        + 32.54% pq_getbyte
        + 24.39% socket_flush
          0.54% SetCurrentStatementStartTimestamp
```

100 x NOTIFY channel_1 to 1 x LISTEN channel_1

```
$ ./listen_script 1
$ pgbench -f ~/notify_channel_1.sql -c 100 -j 100 -T 60 -n bench

tps = 7494.324353 (without initial connection time)

-   99.23%     0.21%  postgres  postgres                [.] PostgresMain
   - 99.02% PostgresMain
      - 61.83% exec_simple_query
         - 57.73% CommitTransactionCommand
            - 57.61% CommitTransaction
               - 27.40% ResourceOwnerReleaseInternal
                  - 27.17% ProcReleaseLocks
                     - 27.09% LockReleaseAll
                        - 21.82% ProcLockWakeup
                           + 19.28% kill
                             0.89% LockCheckConflicts
                        + 3.75% LWLockRelease
               + 14.03% AtCommit_Notify
               + 11.77% PreCommit_Notify
               + 2.31% XactLogCommitRecord
         + 1.76% pg_parse_query
         + 0.70% start_xact_command
      + 22.45% socket_flush
      + 13.29% pq_getbyte
```

1 x NOTIFY channel_1 to 100 x LISTEN channel_1

```
$ for n in `seq 1 100` ; do ./listen_script 1 ; done
$ pgbench -f ~/notify_channel_1.sql -c 1 -j 1 -T 60 -n bench

tps = 798.089837 (without initial connection time)

-   99.75%     0.02%  postgres  postgres                [.] PostgresMain
   - 99.73% PostgresMain
      - 62.41% pq_getbyte
         - pq_recvbuf
```

```
            - 62.40% secure_read
               - 42.07% ProcessClientReadInterrupt
                  - 41.90% ProcessNotifyInterrupt
                     - 34.08% socket_flush
                        - 34.03% internal_flush_buffer
                           - 34.02% secure_write
                              - 27.75% WaitEventSetWait
                                 + 18.20% epoll_pwait
                                 + 7.59% WaitEventSetWait
                                 + 1.79% drain
                              + 5.81% __send
                     + 3.02% CommitTransactionCommand
                     + 2.90% asyncQueueReadAllNotifications
                     + 1.44% StartTransactionCommand
               + 19.31% WaitEventSetWait
               + 0.82% recv
      + 36.48% exec_simple_query
      + 0.62% socket_flush
```

100 x NOTIFY channel_1 to 100 x LISTEN channel_1

```
$ for n in `seq 1 100` ; do ./listen_script 1 ; done
$ pgbench -f ~/notify_channel_1.sql -c 100 -j 100 -T 60 -n bench

tps = 1314.302478 (without initial connection time)

-   99.78%     0.02%  postgres  postgres               [.] PostgresMain
   - 99.76% PostgresMain
      - 50.35% pq_getbyte
         - 50.34% pq_recvbuf
            - 50.34% secure_read
               - 29.69% ProcessClientReadInterrupt
                  - 29.54% ProcessNotifyInterrupt
                     - 22.70% socket_flush
                        - 22.63% internal_flush_buffer
                           - 22.62% secure_write
                              - 17.92% WaitEventSetWait
                                 + 12.84% epoll_pwait
                                 + 3.88% WaitEventSetWait
                                 + 1.11% drain
                              + 4.35% __send
                     + 2.63% CommitTransactionCommand
                     + 2.58% asyncQueueReadAllNotifications
                     + 1.25% StartTransactionCommand
               + 19.25% WaitEventSetWait
               + 1.23% recv
      + 48.19% exec_simple_query
      + 0.94% socket_flush
```

100 x NOTIFY channel_:client_id to 100 x LISTEN channel_:client_id

```
$ for n in `seq 1 100` ; do ./listen_script $n ; done
$ pgbench -f ~/notify_channel_client_id.sql -c 100 -j 100 -T 60 -n bench

tps = 1419.322468 (without initial connection time)

-   99.81%     0.02%  postgres  postgres               [.] PostgresMain
   - 99.79% PostgresMain
      - 50.42% pq_getbyte
         - 50.41% pq_recvbuf
            - 50.41% secure_read
               - 35.41% WaitEventSetWait
                  + 26.01% epoll_pwait
                  + 7.12% WaitEventSetWait
                  + 2.06% drain
               + 10.18% ProcessClientReadInterrupt
               + 4.39% recv
      + 48.15% exec_simple_query
      + 0.97% socket_flush
```

master (b5c53b4) without heavyweight lock

This is just to give an idea of how the heavyweight lock affects the scalability.

```
diff --git a/src/backend/commands/async.c b/src/backend/commands/async.c
index 4bd37d5beb..47dfe42c9c 100644
--- a/src/backend/commands/async.c
+++ b/src/backend/commands/async.c
@@ -919,8 +919,6 @@ PreCommit_Notify(void)
                 * (Historical note: before PG 9.0, a similar lock on
"database 0" was
                 * used by the flatfiles mechanism.)
                 */
-               LockSharedObject(DatabaseRelationId, InvalidOid, 0,
-                                                AccessExclusiveLock);

                /* Now push the notifications into the queue */
                nextNotify = list_head(pendingNotifies->events);
```

1 x NOTIFY channel_1 to 1 x LISTEN channel_1

```
$ ./listen_script 1
$ pgbench -f ~/notify_channel_1.sql -c 1 -j 1 -T 60 -n bench

tps = 11645.734899 (without initial connection time)

-   99.28%     0.24%  postgres  postgres               [.] PostgresMain
   - 99.04% PostgresMain
      - 39.80% exec_simple_query
```

```
          - 32.54% CommitTransactionCommand
             - 32.38% CommitTransaction
                - 23.53% AtCommit_Notify
                   + 22.41% kill
                   + 0.86% AllocSetAllocFromNewBlock
                + 3.57% PreCommit_Notify
                + 1.09% ResourceOwnerReleaseInternal
                + 0.73% XactLogCommitRecord
          + 2.66% pg_parse_query
          + 1.58% start_xact_command
            0.59% PortalRun
            0.57% CreatePortal
       + 32.42% pq_getbyte
       + 24.52% socket_flush
         0.55% SetCurrentStatementStartTimestamp
```

100 x NOTIFY channel_1 to 1 x LISTEN channel_1

```
$ ./listen_script 1
$ pgbench -f ~/notify_channel_1.sql -c 100 -j 100 -T 60 -n bench

tps = 121615.034209 (without initial connection time)

-   99.81%     0.20%  postgres  postgres                [.] PostgresMain
   - 99.61% PostgresMain
      - 67.21% exec_simple_query
         - 57.33% CommitTransactionCommand
            - 57.23% CommitTransaction
               - 38.04% AtCommit_Notify
                  + 34.87% kill
                  + 1.22% LWLockRelease
                  + 0.85% AllocSetAllocFromNewBlock
                  + 0.76% LWLockAcquire
               + 9.06% PreCommit_Notify
               + 2.54% ResourceOwnerReleaseInternal
               + 1.53% XactLogCommitRecord
                 1.44% TransactionIdSetTreeStatus
               + 0.70% GetCurrentTransactionStopTimestamp
                 0.57% LWLockRelease
                 0.54% ProcArrayEndTransaction
               + 0.53% MemoryContextResetOnly
         + 4.81% pg_parse_query
         + 1.33% start_xact_command
           0.85% CreatePortal
           0.54% PortalRun
      + 16.14% socket_flush
      + 14.43% pq_getbyte
```

1 x NOTIFY channel_1 to 100 x LISTEN channel_1

```
$ for n in `seq 1 100` ; do ./listen_script 1 ; done
$ pgbench -f ~/notify_channel_1.sql -c 1 -j 1 -T 60 -n bench

tps = 801.370038 (without initial connection time)

-   99.79%     0.02%  postgres  postgres              [.] PostgresMain
   - 99.77% PostgresMain
      - 62.94% pq_getbyte
         - pq_recvbuf
            - 62.94% secure_read
               - 42.29% ProcessClientReadInterrupt
                  - 42.13% ProcessNotifyInterrupt
                     - 33.99% socket_flush
                        - 33.93% internal_flush_buffer
                           - 33.92% secure_write
                              - 27.73% WaitEventSetWait
                                 + 18.15% epoll_pwait
                                 + 7.59% WaitEventSetWait
                                 + 1.83% drain
                              + 5.75% __send
                     + 3.21% CommitTransactionCommand
                     + 3.00% asyncQueueReadAllNotifications
                     + 1.43% StartTransactionCommand
               + 19.59% WaitEventSetWait
               + 0.86% recv
      + 36.10% exec_simple_query
```

100 x NOTIFY channel_1 to 100 x LISTEN channel_1

```
$ for n in `seq 1 100` ; do ./listen_script 1 ; done
$ pgbench -f ~/notify_channel_1.sql -c 100 -j 100 -T 60 -n bench

tps = 4095.709407 (without initial connection time)

-   99.79%     0.05%  postgres  postgres               [.] PostgresMain
   - 99.73% PostgresMain
      - 54.22% exec_simple_query
         - 52.98% CommitTransactionCommand
            - 52.95% CommitTransaction
               - 50.49% AtCommit_Notify
                  + 49.85% kill
                 1.17% PreCommit_Notify
      + 43.25% pq_getbyte
      + 1.75% socket_flush
```

100 x NOTIFY channel_:client_id to 100 x LISTEN channel_:client_id

```
$ for n in `seq 1 100` ; do ./listen_script $n ; done
$ pgbench -f ~/notify_channel_client_id.sql -c 100 -j 100 -T 60 -n bench

tps = 3354.541290 (without initial connection time)

-   99.87%     0.03%  postgres  postgres                [.] PostgresMain
   - 99.85% PostgresMain
      - 62.30% exec_simple_query
         - 61.72% CommitTransactionCommand
            - 61.71% CommitTransaction
               - 60.24% AtCommit_Notify
                  + 59.83% kill
                 0.80% PreCommit_Notify
      + 36.09% pq_getbyte
      + 1.21% socket_flush
```

Scripts

The following `expect` script was used to spawn LISTEN connections, that were kept open, and that did SELECT 1 every second, to receive the async notifications, to make it more realistic:

`listen_script`:

```
#!/usr/bin/expect -f
set timeout -1
log_user 0                          ;# suppress stdout/stderr

if {$argc != 1} {
    puts stderr "Usage: $argv0 <channel>"
    exit 64                         ;# EX_USAGE
}
set channel [lindex $argv 0]

if {[fork] != 0} { exit }
disconnect                           ;# stdio → /dev/null

spawn /home/joel/pg-debug/bin/psql -q bench
sleep 1
send "LISTEN channel_$channel;\r"

proc heartbeat {} {
    if {[catch {send "SELECT 1;\r"}]} { exit 2 }   ;# PTY gone → exit
    after 1000 heartbeat
}
after 1000 heartbeat

while 1 {
    expect {
        eof { exit 0 }
        "You are currently not connected to a database." { exit 1 }
```

```
        -re {.*\r?\n} { exp_continue }
    }
}
```

`pgbench` scripts:

notify_channel_1.sql:

```
NOTIFY channel_1;
```

notify_channel_client_id.sql:

```
NOTIFY channel_:client_id;
```