PostgreSQL Performance Troubleshooting

By: Karl Arao

(best viewed in MS Word -> View -> Web Layout)

Table of Contents

[Step 1) Get dbstate output 1](#_Toc13497071)

[Step 2) Get explain plan of the slow SQL 1](#_Toc13497072)

[\timing 1](#_Toc13497073)

[EXPLAIN ANALYZE + COST 1](#_Toc13497074)

[Step 3) Monitoring Postgres workload 1](#_Toc13497075)

[Zip ASH data for remote troubleshooting 1](#_Toc13497076)

# Step 1) Get dbstate output

Script by Abel Macias, Dimas Chbane

* Connect using psql

|  |
| --- |
| psql -h localhost |

* List all databases

|  |
| --- |
| \l # this list all databases  Example output:  kristofferson.a.arao=# \l  List of databases  Name | Owner | Encoding | Collate | Ctype | Access privileges  ----------------------+----------------------+----------+-------------+-------------+-----------------------  kristofferson.a.arao | kristofferson.a.arao | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |  postgres | postgres | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 |  template0 | postgres | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 | =c/postgres +  | | | | | postgres=CTc/postgres  template1 | postgres | UTF8 | en\_US.UTF-8 | en\_US.UTF-8 | =c/postgres +  | | | | | postgres=CTc/postgres  (4 rows) |

* Show current database

|  |
| --- |
| kristofferson.a.arao=# SELECT current\_database();  current\_database  ----------------------  kristofferson.a.arao  (1 row) |

* Connect to the database

|  |
| --- |
| \c <db name> # connect to a database  Example output:  kristofferson.a.arao=# \c postgres  You are now connected to database "postgres" as user "kristofferson.a.arao". |

* Run the pg\_dbstate.sql script

|  |
| --- |
| \i pg\_dbstate.sql # run the sql script  postgres=# \i pg\_dbstate.sql  Timing is on.  Time: 1.709 ms  Time: 1.345 ms  Time: 0.192 ms  Tuples only is on.  Expanded display is on.  Time: 0.247 ms  Time: 0.470 ms  Time: 5.222 ms  Time: 3.889 ms  Time: 1.759 ms  Time: 12.733 ms  Expanded display is off. |

* Rename the dbstate.txt to the database name

|  |
| --- |
| ls dbstate.txt pg\_dbstate.sql  dbstate.txt pg\_dbstate.sql  mv dbstate.txt dbstate.txt.postgresql |

# Step 2) Get explain plan of the slow SQL

* Connect using psql

|  |
| --- |
| psql -h localhost |

* Connect to the database

|  |
| --- |
| \c <db name> # connect to a database  Example output:  kristofferson.a.arao=# \c postgres  You are now connected to database "postgres" as user "kristofferson.a.arao". |

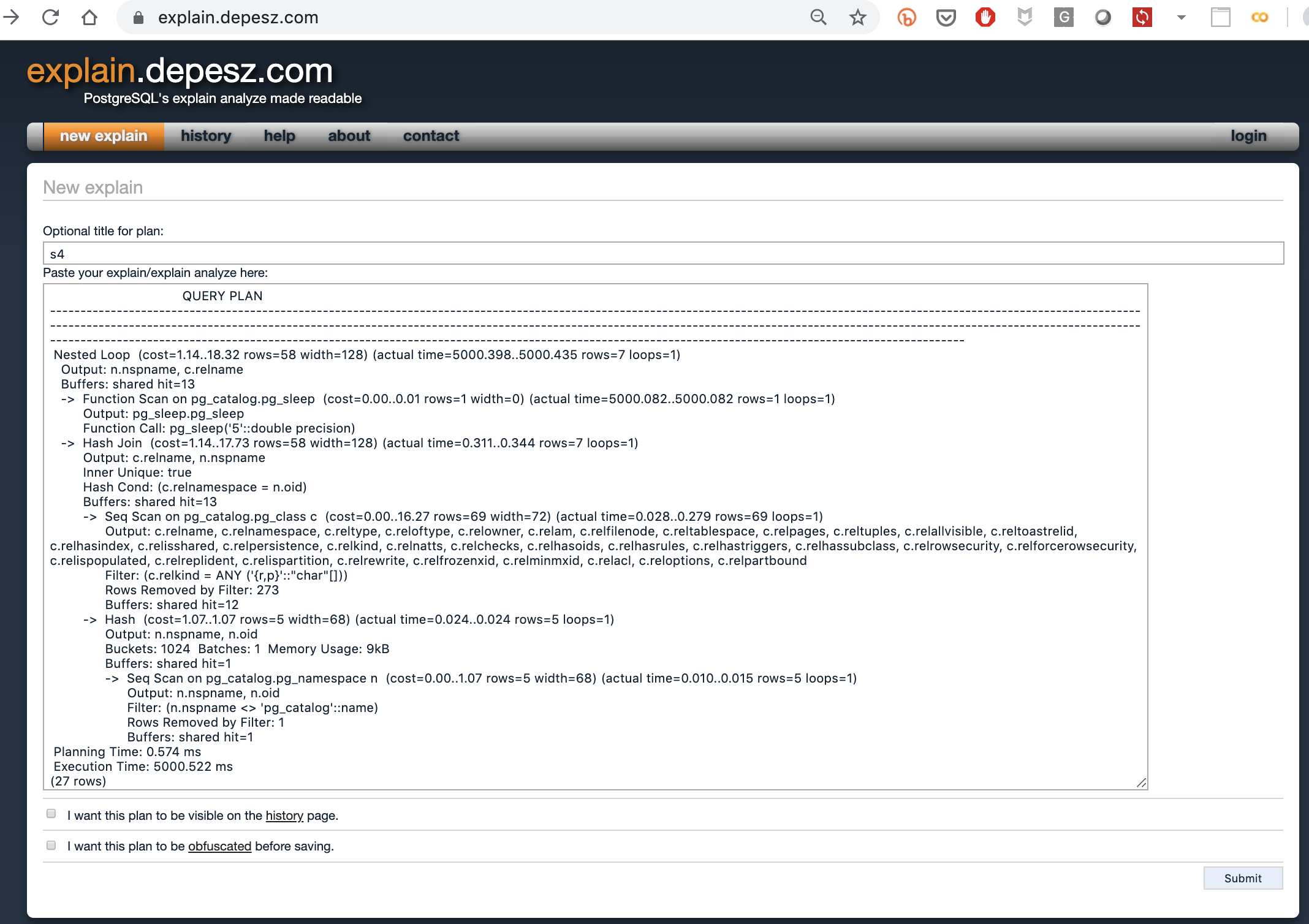
* Run the EXPLAIN command with the slow SQL and spool to text

|  |
| --- |
| # example artificial slow query below with pg\_sleep to 5 seconds  explain (analyze,verbose,costs,buffers, FORMAT text) SELECT schemaname, tablename  FROM pg\_tables, pg\_sleep(5)  WHERE schemaname <> 'pg\_catalog';  # run with \o <filename> to spool to text  Example output:  postgres=# \o explain\_text.txt postgres=#  postgres=# explain (analyze,verbose,costs,buffers, FORMAT text) SELECT schemaname, tablename  postgres-# FROM pg\_tables, pg\_sleep(5)  postgres-# WHERE schemaname <> 'pg\_catalog';  postgres=#  postgres=# \o |

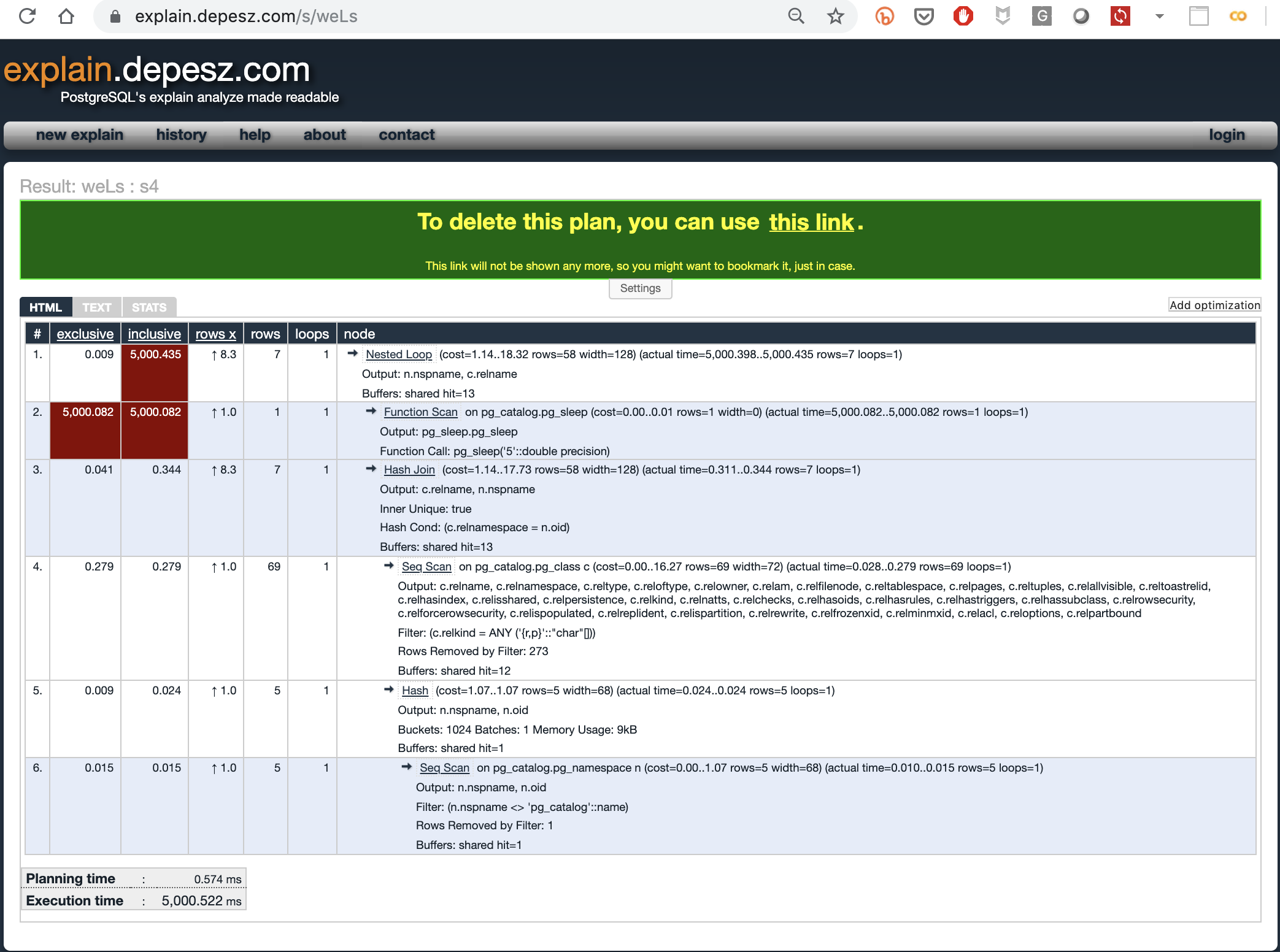
* Query Plan output

|  |
| --- |
| QUERY PLAN  -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  Nested Loop (cost=1.14..18.32 rows=58 width=128) (actual time=5000.398..5000.435 rows=7 loops=1)  Output: n.nspname, c.relname  Buffers: shared hit=13  -> Function Scan on pg\_catalog.pg\_sleep (cost=0.00..0.01 rows=1 width=0) (actual time=5000.082..5000.082 rows=1 loops=1)  Output: pg\_sleep.pg\_sleep  Function Call: pg\_sleep('5'::double precision)  -> Hash Join (cost=1.14..17.73 rows=58 width=128) (actual time=0.311..0.344 rows=7 loops=1)  Output: c.relname, n.nspname  Inner Unique: true  Hash Cond: (c.relnamespace = n.oid)  Buffers: shared hit=13  -> Seq Scan on pg\_catalog.pg\_class c (cost=0.00..16.27 rows=69 width=72) (actual time=0.028..0.279 rows=69 loops=1)  Output: c.relname, c.relnamespace, c.reltype, c.reloftype, c.relowner, c.relam, c.relfilenode, c.reltablespace, c.relpages, c.reltuples, c.relallvisible, c.reltoastrelid, c.relhasindex, c.relisshared, c.relpersistence, c.relkind, c.relnatts, c.relchecks, c.relhasoids, c.relhasrules, c.relhastriggers, c.relhassubclass, c.relrowsecurity, c.relforcerowsecurity, c.relispopulated, c.relreplident, c.relispartition, c.relrewrite, c.relfrozenxid, c.relminmxid, c.relacl, c.reloptions, c.relpartbound  Filter: (c.relkind = ANY ('{r,p}'::"char"[]))  Rows Removed by Filter: 273  Buffers: shared hit=12  -> Hash (cost=1.07..1.07 rows=5 width=68) (actual time=0.024..0.024 rows=5 loops=1)  Output: n.nspname, n.oid  Buckets: 1024 Batches: 1 Memory Usage: 9kB  Buffers: shared hit=1  -> Seq Scan on pg\_catalog.pg\_namespace n (cost=0.00..1.07 rows=5 width=68) (actual time=0.010..0.015 rows=5 loops=1)  Output: n.nspname, n.oid  Filter: (n.nspname <> 'pg\_catalog'::name)  Rows Removed by Filter: 1  Buffers: shared hit=1  Planning Time: 0.574 ms  Execution Time: 5000.522 ms  (27 rows) |

* Paste the explain plan to [https://explain.depesz.com](https://explain.depesz.com/) , click Submit



* The output will show where most of the response time is spent



A few notes:

## \timing

* \timing , equivalent to SET TIME ON in Oracle to show the wall clock time on every SQL execution in psql

|  |
| --- |
| postgres=# \timing  Timing is on.  postgres=# explain (verbose,costs,buffers, FORMAT text) SELECT schemaname, tablename  ..output snipped..  Planning Time: 0.583 ms  Execution Time: 5000.761 ms  (27 rows)  Time: 5002.297 ms (00:05.002)  postgres=# |

## EXPLAIN ANALYZE + COST

* The ANALYZE option of EXPLAIN means to execute the SQL then print the query plan
* Below is query plan with (5 seconds) and without ANALYZE (1 second)

|  |
| --- |
| **WITH ANALYZE**  postgres=# \timing  Timing is on.  postgres=# explain (analyze,verbose,costs,buffers, FORMAT text) SELECT schemaname, tablename  postgres-# FROM pg\_tables, pg\_sleep(5)  postgres-# WHERE schemaname <> 'pg\_catalog';  QUERY PLAN  -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  Nested Loop (cost=1.14..18.32 rows=58 width=128) (actual time=5000.472..5000.519 rows=7 loops=1)  Output: n.nspname, c.relname  Buffers: shared hit=13  -> Function Scan on pg\_catalog.pg\_sleep (cost=0.00..0.01 rows=1 width=0) (actual time=5000.171..5000.172 rows=1 loops=1)  Output: pg\_sleep.pg\_sleep  Function Call: pg\_sleep('5'::double precision)  -> Hash Join (cost=1.14..17.73 rows=58 width=128) (actual time=0.296..0.338 rows=7 loops=1)  Output: c.relname, n.nspname  Inner Unique: true  Hash Cond: (c.relnamespace = n.oid)  Buffers: shared hit=13  -> Seq Scan on pg\_catalog.pg\_class c (cost=0.00..16.27 rows=69 width=72) (actual time=0.024..0.272 rows=69 loops=1)  Output: c.relname, c.relnamespace, c.reltype, c.reloftype, c.relowner, c.relam, c.relfilenode, c.reltablespace, c.relpages, c.reltuples, c.relallvisible, c.reltoastrelid, c.relhasindex, c.relisshared, c.relpersistence, c.relkind, c.relnatts, c.relchecks, c.relhasoids, c.relhasrules, c.relhastriggers, c.relhassubclass, c.relrowsecurity, c.relforcerowsecurity, c.relispopulated, c.relreplident, c.relispartition, c.relrewrite, c.relfrozenxid, c.relminmxid, c.relacl, c.reloptions, c.relpartbound  Filter: (c.relkind = ANY ('{r,p}'::"char"[]))  Rows Removed by Filter: 273  Buffers: shared hit=12  -> Hash (cost=1.07..1.07 rows=5 width=68) (actual time=0.023..0.024 rows=5 loops=1)  Output: n.nspname, n.oid  Buckets: 1024 Batches: 1 Memory Usage: 9kB  Buffers: shared hit=1  -> Seq Scan on pg\_catalog.pg\_namespace n (cost=0.00..1.07 rows=5 width=68) (actual time=0.009..0.015 rows=5 loops=1)  Output: n.nspname, n.oid  Filter: (n.nspname <> 'pg\_catalog'::name)  Rows Removed by Filter: 1  Buffers: shared hit=1  Planning Time: 0.583 ms  Execution Time: 5000.761 ms  (27 rows)  Time: 5002.297 ms (00:05.002)  **WITHOUT ANALYZE**  postgres=# explain (verbose,costs,buffers, FORMAT text) SELECT schemaname, tablename  postgres-# FROM pg\_tables, pg\_sleep(5)  postgres-# WHERE schemaname <> 'pg\_catalog';  ERROR: EXPLAIN option BUFFERS requires ANALYZE  Time: 2.771 ms  postgres=#  postgres=#  postgres=# explain (verbose,costs, FORMAT text) SELECT schemaname, tablename  postgres-# FROM pg\_tables, pg\_sleep(5)  postgres-# WHERE schemaname <> 'pg\_catalog';  QUERY PLAN  -------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------  Nested Loop (cost=1.14..18.32 rows=58 width=128)  Output: n.nspname, c.relname  -> Function Scan on pg\_catalog.pg\_sleep (cost=0.00..0.01 rows=1 width=0)  Output: pg\_sleep.pg\_sleep  Function Call: pg\_sleep('5'::double precision)  -> Hash Join (cost=1.14..17.73 rows=58 width=128)  Output: c.relname, n.nspname  Inner Unique: true  Hash Cond: (c.relnamespace = n.oid)  -> Seq Scan on pg\_catalog.pg\_class c (cost=0.00..16.27 rows=69 width=72)  Output: c.relname, c.relnamespace, c.reltype, c.reloftype, c.relowner, c.relam, c.relfilenode, c.reltablespace, c.relpages, c.reltuples, c.relallvisible, c.reltoastrelid, c.relhasindex, c.relisshared, c.relpersistence, c.relkind, c.relnatts, c.relchecks, c.relhasoids, c.relhasrules, c.relhastriggers, c.relhassubclass, c.relrowsecurity, c.relforcerowsecurity, c.relispopulated, c.relreplident, c.relispartition, c.relrewrite, c.relfrozenxid, c.relminmxid, c.relacl, c.reloptions, c.relpartbound  Filter: (c.relkind = ANY ('{r,p}'::"char"[]))  -> Hash (cost=1.07..1.07 rows=5 width=68)  Output: n.nspname, n.oid  -> Seq Scan on pg\_catalog.pg\_namespace n (cost=0.00..1.07 rows=5 width=68)  Output: n.nspname, n.oid  Filter: (n.nspname <> 'pg\_catalog'::name)  (17 rows)  Time: 1.049 ms |

# Step 3) Monitoring Postgres workload

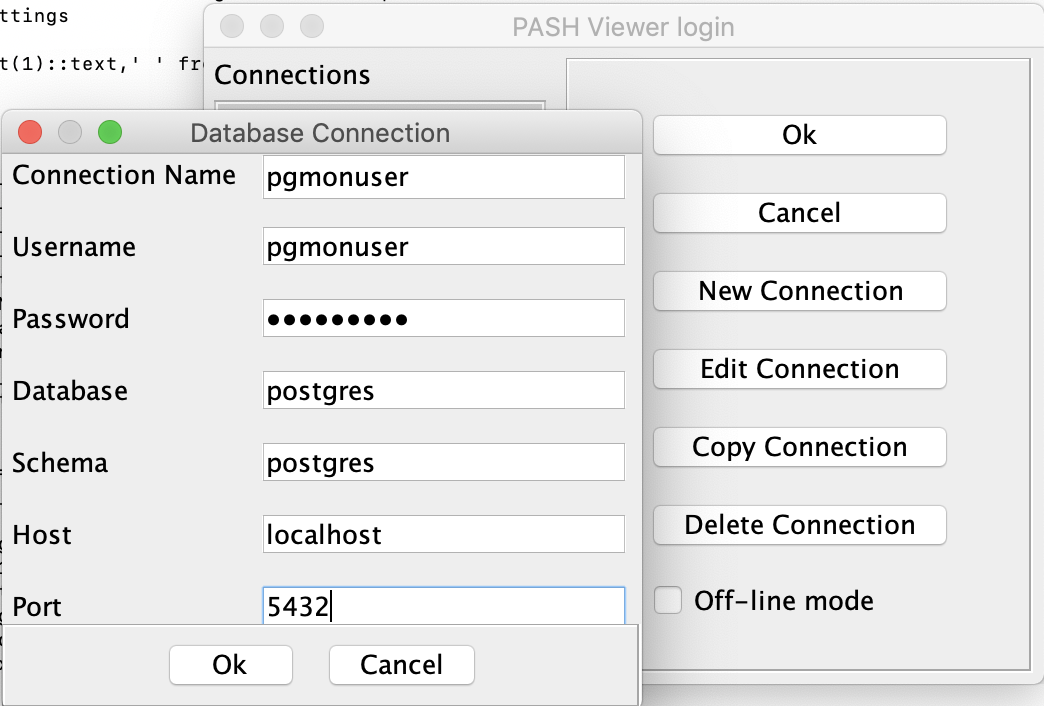
* Download the tool at <https://github.com/dbacvetkov/PASH-Viewer/archive/master.zip>
  + Project page here <https://github.com/dbacvetkov/PASH-Viewer>
* Unzip and read the README.md
* Build the tool

|  |
| --- |
| cd /Users/kristofferson.a.arao/ash\_viewer/PASH-Viewer-master  ./gradlew assembleDist  cd /Users/kristofferson.a.arao/ash\_viewer/PASH-Viewer-master/build/distributions/PASH-Viewer-0.4.1/bin |

* Create the monitoring user PGMONUSER to see query plans

|  |
| --- |
| * from psql connect to postgres database   psql -h localhost # psql  \c postgres # connect to postgres   * create the pgmonuser   postgres=# CREATE USER pgmonuser WITH password 'pgmonuser';  CREATE ROLE  postgres=# GRANT pg\_monitor TO pgmonuser;  GRANT ROLE  postgres=# |

* Enter connection details, use the PGMONUSER to connect



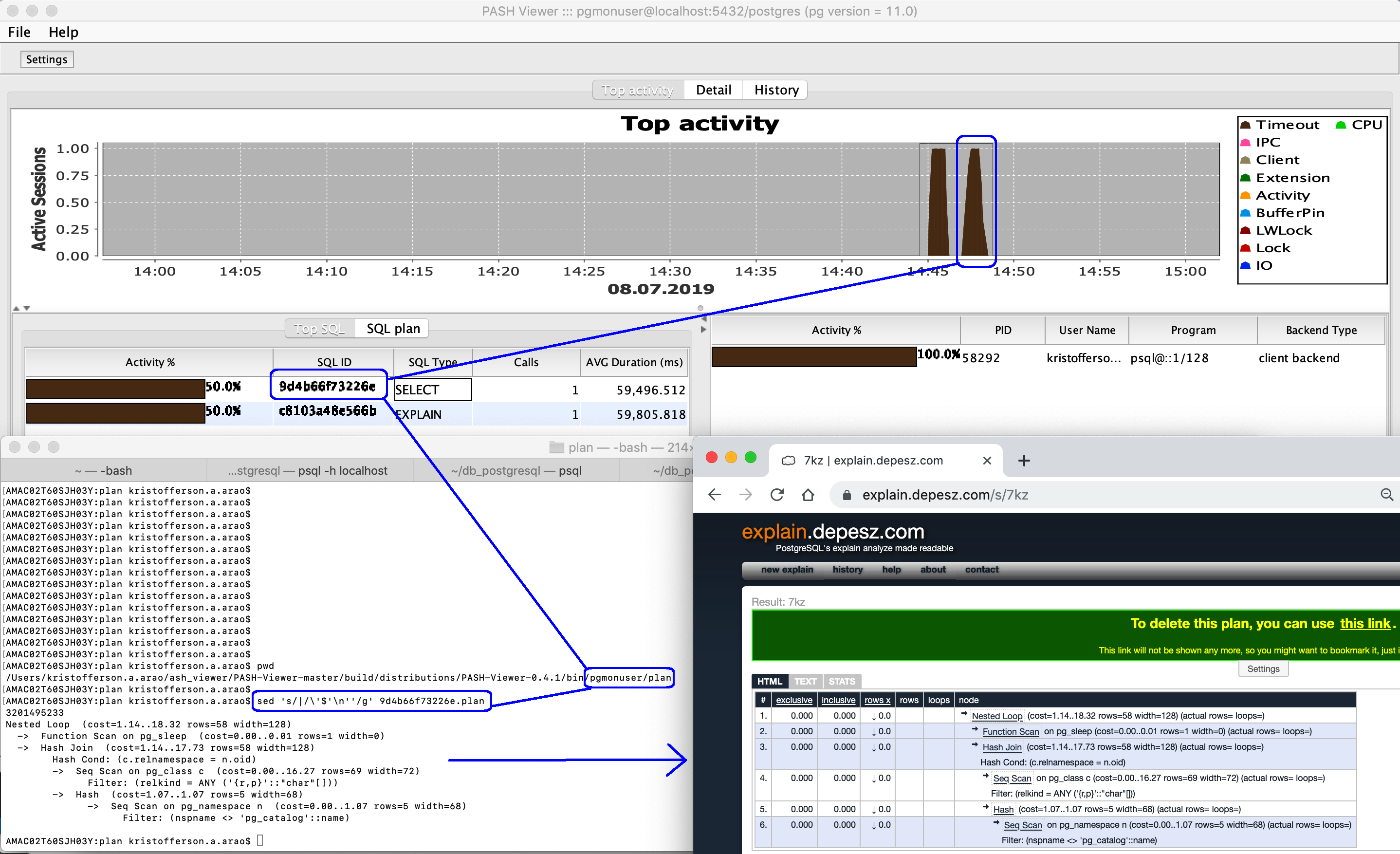
* Run PASH-Viewer

|  |
| --- |
| ./PASH-Viewer |

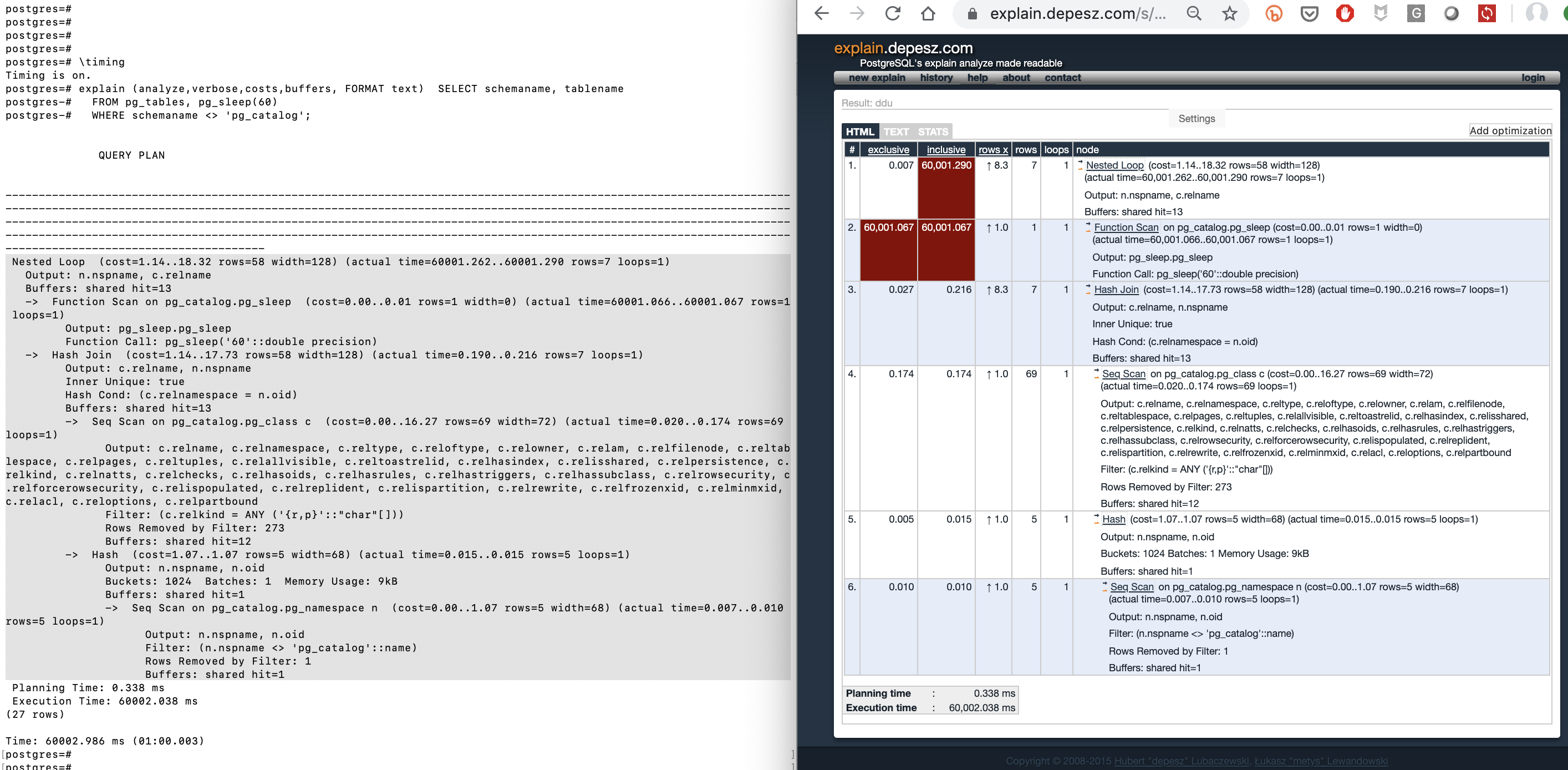
* The PASH-Viewer will create a directory for each connection name
* Inside the directory are two subdirectories
  + Plan directory - where SQLs and plans are stored
  + .jdb data files - for storing historical workload and raw data. This can be zipped and shared for remote troubleshooting

|  |
| --- |
| ./PASH-Viewer  $ pwd  /Users/kristofferson.a.arao/ash\_viewer/PASH-Viewer-master/build/distributions/PASH-Viewer-0.4.1/bin/pgmonuser  $ ls -ltr  total 0  drwxr-xr-x 4 kristofferson.a.arao 562225435 128 Jul 8 14:43 08072019144335  drwxr-xr-x 4 kristofferson.a.arao 562225435 128 Jul 8 14:54 plan |

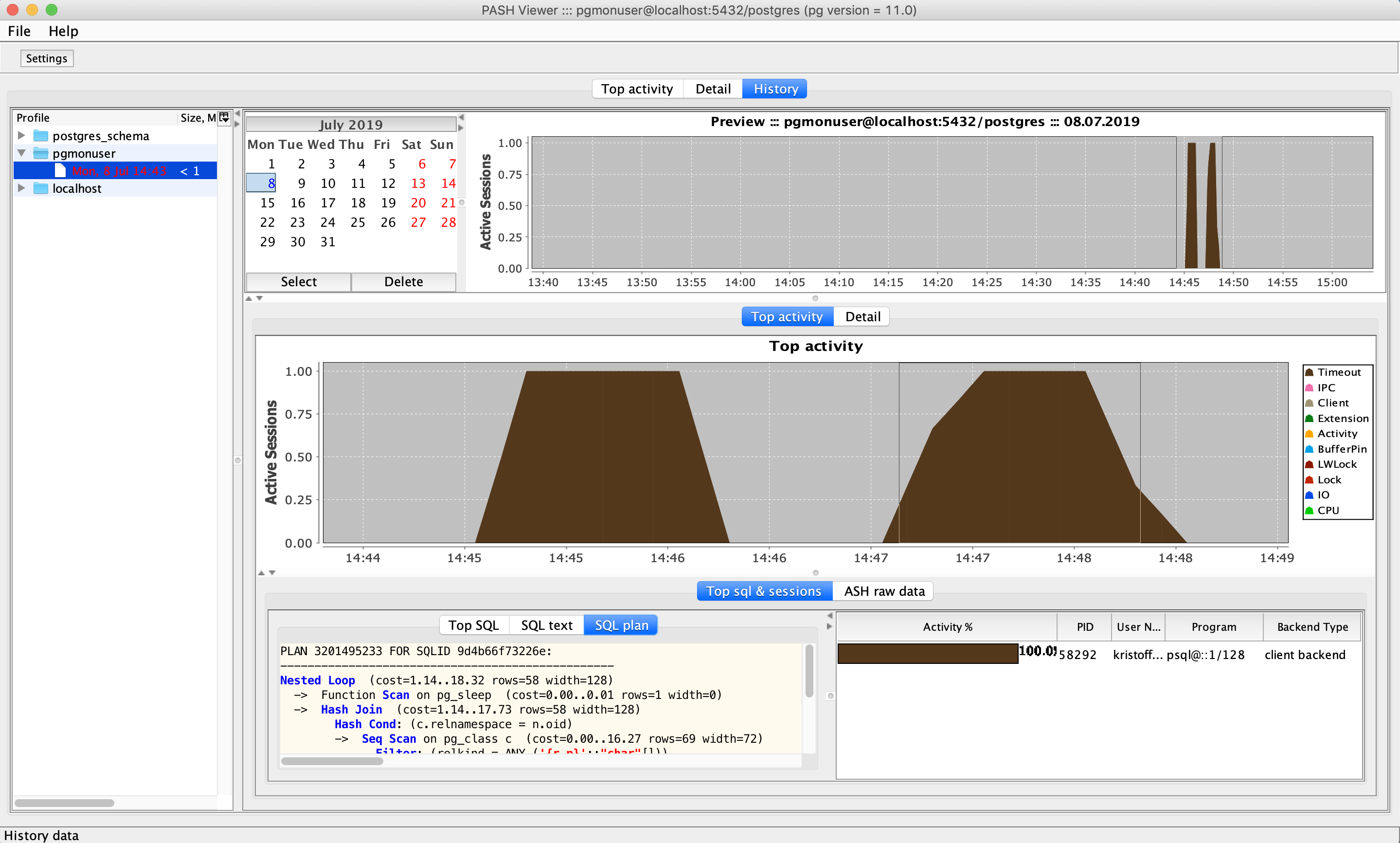
* Below correlates the database load with the running SQL and the underlying execution plan



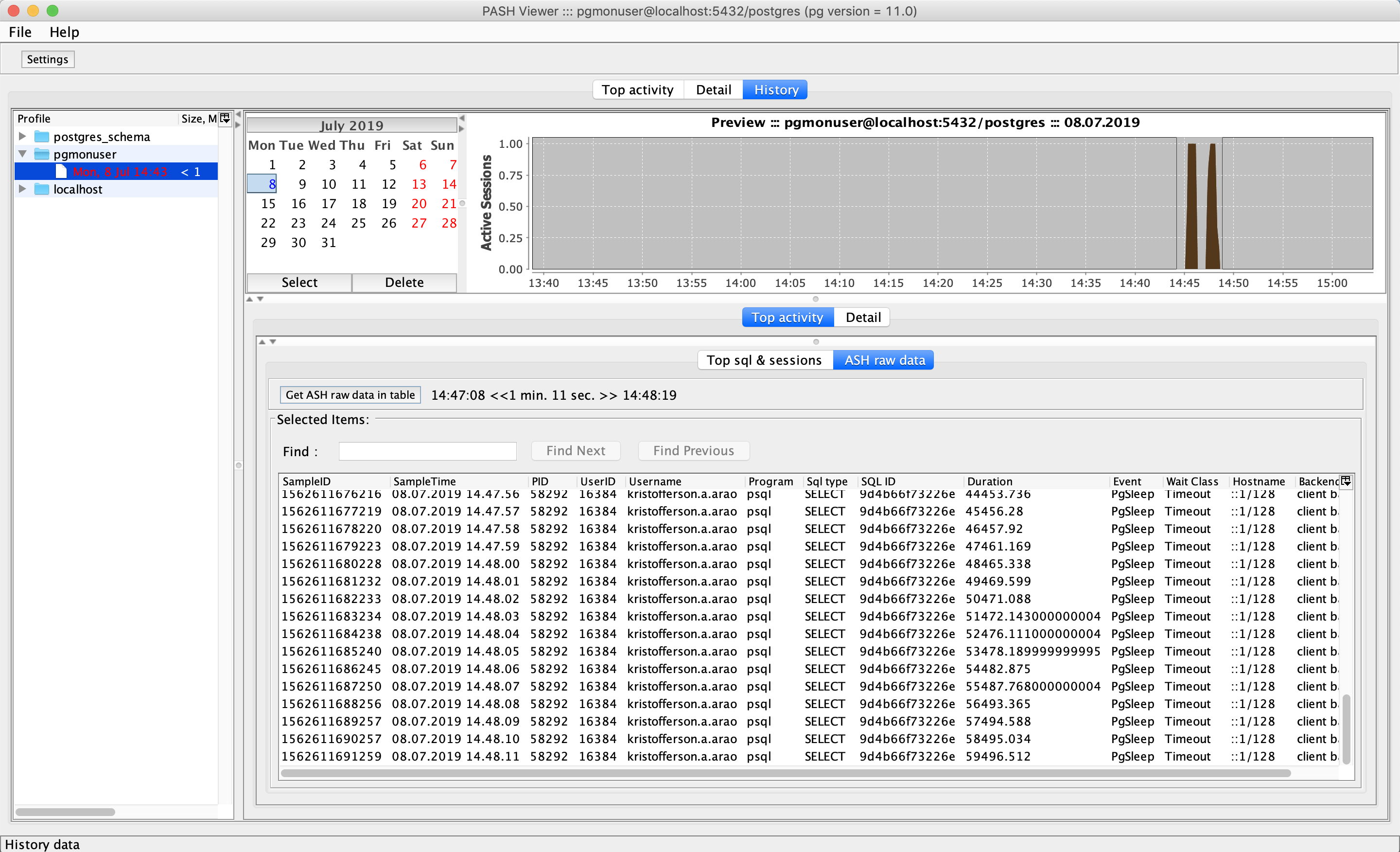
* The query plan captured by PASH-Viewer does not contain row source statistics.
* The EXPLAIN with ANALYZE and COST needs to be run on a separate session to get row source details to see where the time is mostly spent



* Historical view can be viewed using History tab



* Also raw data is available for drill down



A few notes:

## Zip ASH data for remote troubleshooting

* On the PASH-Viewer base directory

|  |
| --- |
| ./PASH-Viewer  $ pwd  /Users/kristofferson.a.arao/ash\_viewer/PASH-Viewer-master/build/distributions/PASH-Viewer-0.4.1/bin  $ ls -ltr  total 24  -rwxr-xr-x 1 kristofferson.a.arao 562225435 2442 Jul 8 01:56 PASH-Viewer.bat  -rwxr-xr-x 1 kristofferson.a.arao 562225435 5388 Jul 8 01:56 PASH-Viewer  drwxr-xr-x 5 kristofferson.a.arao 562225435 160 Jul 8 16:43 pgmonuser |

* Zip the pgmonuser directory and share through email or Sharepoint URL

|  |
| --- |
| $ du -sm pgmonuser/  3 pgmonuser/  $ zip -r pgmonuser pgmonuser  adding: pgmonuser/ (stored 0%)  adding: pgmonuser/plan/ (stored 0%)  adding: pgmonuser/plan/9d4b66f73226e.sql (deflated 18%)  adding: pgmonuser/plan/9d4b66f73226e.plan (deflated 50%)  adding: pgmonuser/08072019164333/ (stored 0%)  adding: pgmonuser/08072019164333/00000000.jdb (deflated 92%)  adding: pgmonuser/08072019164333/je.lck (stored 0%)  adding: pgmonuser/08072019144335/ (stored 0%)  adding: pgmonuser/08072019144335/00000000.jdb (deflated 91%)  adding: pgmonuser/08072019144335/je.lck (stored 0%)  $ ls -ltr pgmonuser.zip  -rw-r--r-- 1 kristofferson.a.arao 562225435 108646 Jul 8 16:45 pgmonuser.zip |