pg_profile - PostgreSQL historic workload reporting tool

Andrey Zubkov, Senior DBA

Postgres Professional

May 11, 2021

Problem

- Which workload causes the most resource consumption in a database?
- What is the cause of that performance hit since last week when system performed well?

Approaching the problem

For a Postgres database we can do the following:

- Setup detailed logging with further log analysis
- Collect performance statistics and track changes

This talk is about a tool, implementing the second approach

Statistics approach pros and cons

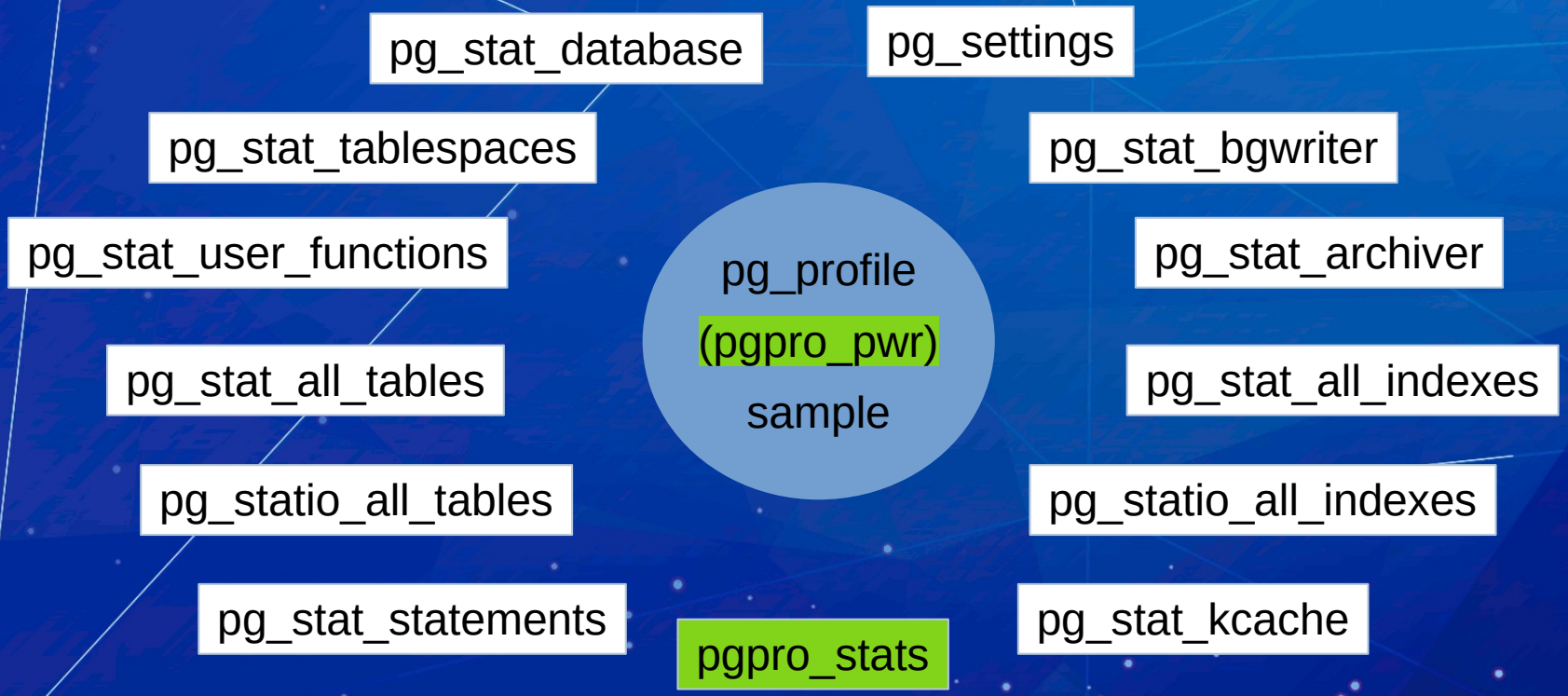
+ Track all statements (even very short)
+ Track database object statistics
+ Avoid huge log-file writing and analysis
- No parameter values
- No plans
- Failed statements are invisible

pg_profile concepts

- Pure pl/pgsql, i.e. no binaries/libraries, services, etc.
- Sampling. 1-2 samples per hour
  (no much overhead)
- Build a report between any two samples
- Build a differential report on two intervals

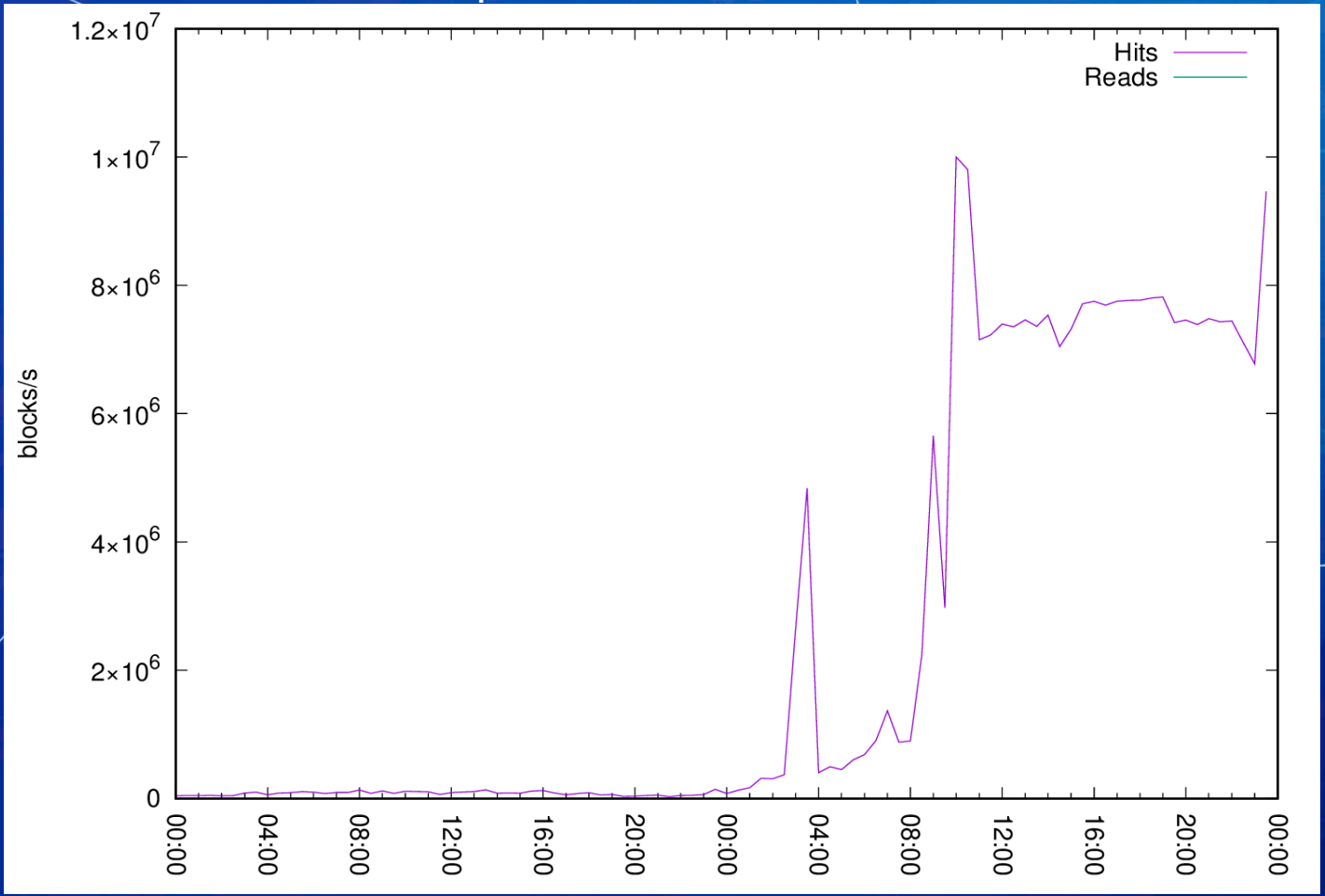
All you need is postgres!

pg_profile sample contents

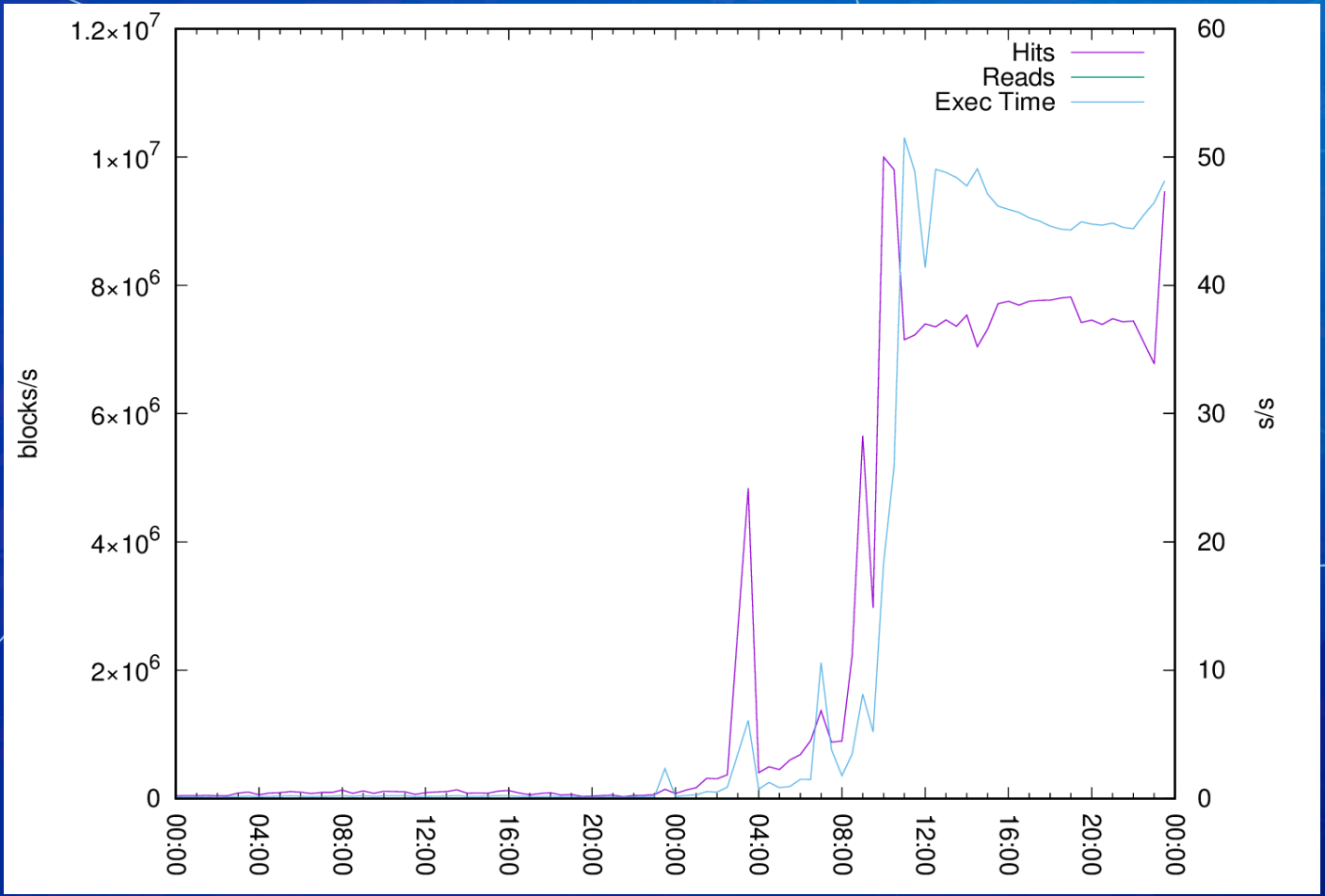


Real issue

unexpected hit rate increase



Issue affects execution time



How can *pg_profile* help us?

- Build a report on known bad time interval

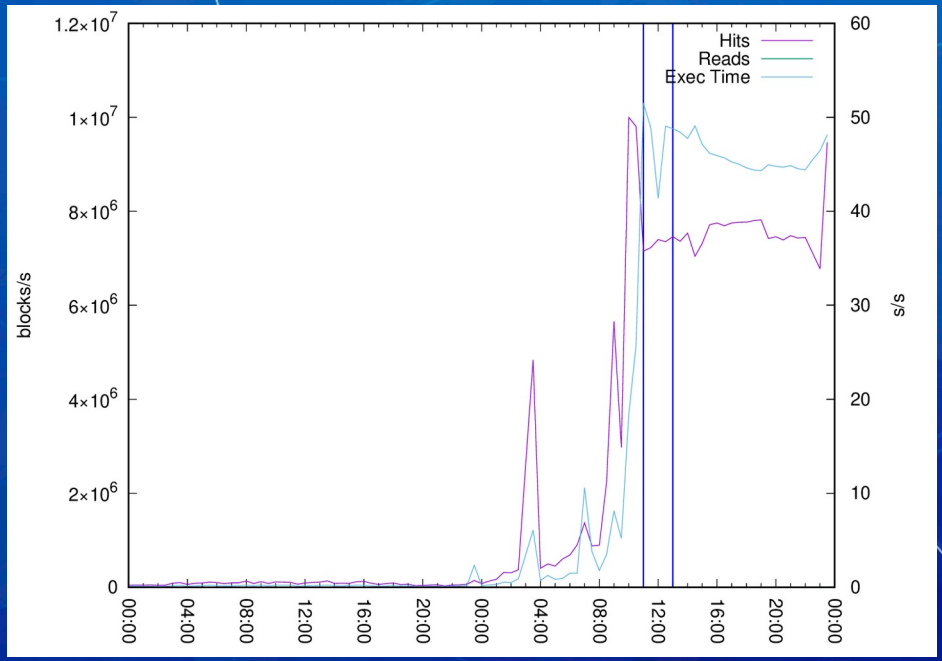
Report contents

- Server statistics
  - Database statistics
  - Statement statistics by database
  - Cluster statistics
  - Tablespace statistics
- SQL Query statistics
  - Top SQL by elapsed time
  - Top SQL by planning time
  - Top SQL by execution time
  - Top SQL by executions
  - Top SQL by I/O wait time
  - Top SQL by shared blocks fetched
  - Top SQL by shared blocks read
  - Top SQL by shared blocks dirtied
  - Top SQL by shared blocks written
  - Top SQL by WAL size
  - Top SQL by temp usage
  - rusage statistics
    - Top SQL by system and user time
    - Top SQL by reads/writes done by filesystem layer
  - Complete list of SQL texts
- Schema object statistics
  - Top tables by estimated sequentially scanned volume
  - Top tables by blocks fetched
  - Top tables by blocks read
  - Top DML tables
  - Top tables by updated/deleted tuples
  - Top growing tables
  - Top indexes by blocks fetched
  - Top indexes by blocks read
  - Top growing indexes
  - Unused indexes
- User function statistics
  - Top functions by total time
  - Top functions by executions
- Vacuum-related statistics
  - Top tables by vacuum operations
  - Top tables by analyze operations
  - Top indexes by estimated vacuum I/O load
  - Top tables by dead tuples ratio
  - Top tables by modified tuples ratio
- Cluster settings during the report interval

Issue time report

Report on 11:00-13:00 should do the trick

```
$ psql -Aqtc \
"SELECT profile.get_report(130,134)" \
-o report_issue.html
$
```



Statements by execution time

Query ID	Database	Exec (s)	%Total	I/O time (s)		Rows	Execution times (ms)				Executions
				Read	Write		Mean	Min	Max	StdErr	
825ec9dfe2 [ba551e58a1220972]	demodb	17677.69	26.10			4976275	436.777	131.332	1268.142	87.207	40473
fc2b6ac0db [7058521854c25be5]	demodb	13814.61	20.40			435	124.600	33.169	523.835	53.414	110872
32e15bfa2b [2c3252b0a9ecf099]	demodb	12796.92	18.90				224.436	62.513	725.970	85.991	57018
9972b38b9c [711d687fdb6583af]	demodb	6819.08	10.07				216.334	63.142	628.219	86.583	31521
581a0cb27e [42c019fd344ccda3]	demodb	6763.17	9.99				2318.536	0.110	4348.005	515.938	2917
476c08c031 [de37a7b16ab1d9ec]	demodb	6257.31	9.24			3100	1098.931	0.012	4284.943	1233.238	5694
bb9daa91f5 [19858c316e39b93a]	demodb	820.60	1.21			19459	42.600	12.135	261.705	25.354	19263

Top SQL by shared blocks fetched

Query ID	Database	blks fetched	%Total	Hits(%)	Elapsed(s)	Rows	Executions
581a0cb27e #5 [42c019fd344ccda3]	demodb	7294930558	41.42	100.00	6763.2		2917
Fc2b6ac0db #2 [7058521854c25be5]	demodb	6232122854	35.38	100.00	13814.6	435	110872
825ec9dfe2 #1 [ba551e58a1220972]	demodb	2583010863	14.66	100.00	17677.7	4976275	40473
32e15bfa2b #3 [2c3252b0a9ecf099]	demodb	774440330	4.40	100.00	12796.9		57018
9972b38b9c #4 [711d687fdb6583af]	demodb	427932206	2.43	100.00	6819.1		31521
615932b6c7 #11 [3865b6f15c706793]	demodb	33263701	0.19	100.00	225.8	77099	962
5fadf658f1 #9 [2e77692b5dff5c21]	demodb	33045697	0.19	100.00	304.9	1268	1381

Top tables by blocks fetched

DB	Tablespace	Schema	Table	Heap		Ix		TOAST		TOAST-Ix	
				Blks	%Total	Blks	%Total	Blks	%Total	Blks	%Total
demodb	pg_default	i6c	i6_n_m	9481392382	53.64	357341693	2.02				
demodb	pg_default	i6c	i6_d_t_d	1656550069	9.37	4975941917	28.15				
demodb	pg_default	i6c	i6_d_t_m	696556409	3.94	31248149	0.18	4	0.00	66	0.00
demodb	pg_default	i6c	i6_n_as	91667256	0.52	4863695	0.03				

Top indexes by blocks fetched

DB	Tablespace	Schema	Table	Index	Scans	Blks	%Total
demodb	pg_default	i6c	i6_d_t_d	i6_d_t_d_pk	1654718992	4975805807	28.15
demodb	pg_default	i6c	i6_n_m	i6_n_m_ix1	91626609	322193350	1.82
demodb	pg_default	i6c	i6_n_m	i6_n_m_ix2	88555	34801154	0.20
demodb	pg_default	i6c	i6_d_u	i6_d_u_uk	8529766	25684839	0.15

How can *pg_profile* help us?

- Issue report findings:
  - Leading three statements consumed 65% of time and 91% of blocks
  - Leading two tables with indexes consumed 93% of blocks

How can *pg_profile* help us?

- Issue report findings:
  - Leading three statements consumed 65% of time and 91% of blocks
  - Leading two tables with indexes consumed 93% of blocks

- Let’s build a differential report

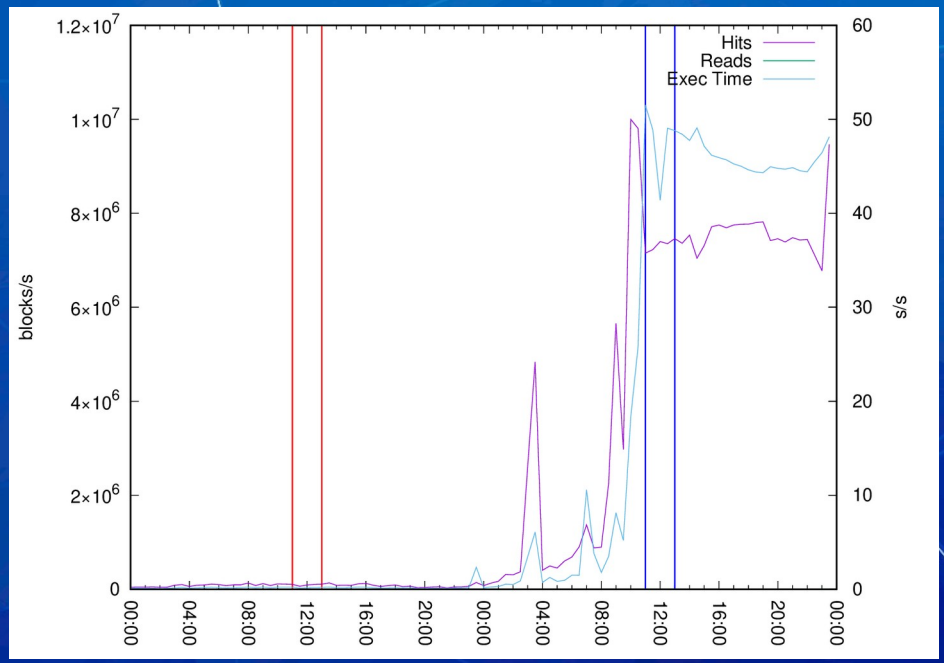
Differential report

- Built on two time intervals
- Statistics of the same objects located one next to other
- The first interval values colored red, and the second interval values colored blue

Differential report

Differential report on 11:00-13:00 *today* with 11:00-13:00 *yesterday*

```
$ psql -Aqtc \
"SELECT
profile.get_report(82,86,130,134)" \
-o diffreport_issue.html
$
```



Database statistics

Database	I	Transactions			Block statistics			Tuples					Size	Growth
		Commits	Rollbacks	Deadlocks	Hit(%)	Read	Hit	Ret	Fet	Ins	Upd	Del		
demodb	1	1554854	1804		99.79	461906	218000570	701175479	90047284	27420	88226	9603	29 GB	7512 kB
	2	3439169	7307		99.99	1615331	17662792017	61934924526	25927080482	225683	357037	48921	29 GB	53 MB
aux	1	284			100.00		25876	162460	8712				7901 kB	
	2	284			100.00		24812	161980	8232				7901 kB	
postgres	1	2963			99.90	475	495350	1563478	89189	33269	736	32028	22 MB	296 kB
	2	2970			99.95	658	1459023	5186650	437424	33524	747	32063	27 MB	368 kB
Total	1	1558101	1804		99.79	462381	218521796	702901417	90145185	60689	88962	41631	29 GB	7808 kB
	2	3442423	7307		99.99	1615989	17664275852	61940273156	25927526138	259207	357784	80984	29 GB	53 MB

Top SQL by execution time

Query ID	Database	I	Exec (s)	%Total	Rows	Execution times (ms)				Executions
						Mean	Min	Max	StdErr	
825ec9dfe2 [ba551e58a1220972]	demodb	1	1.85	0.76	347	0.011	0.005	9.809	0.046	171201
		2	17677.69	26.10	4976275	436.777	131.332	1268.142	87.207	40473
fc2b6ac0db [7058521854c25be5]	demodb	1	1.74	0.71	135	0.010	0.003	11.883	0.067	172062
		2	13814.61	20.40	435	124.600	33.169	523.835	53.414	110872
32e15bfa2b [2c3252b0a9ecf099]	demodb	1	1.08	0.44		0.009	0.003	9.954	0.062	114724
		2	12796.92	18.90		224.436	62.513	725.970	85.991	57018
581a0cb27e [42c019fd344ccda3]	demodb	1	62.76	25.76		965.550	870.881	1085.988	58.479	65
		2	6763.17	9.99		2318.536	0.110	4348.005	515.938	2917
9972b38b9c [711d687fdb6583af]	demodb	1	0.61	0.25		0.010	0.003	7.502	0.057	58624
		2	6819.08	10.07		216.334	63.142	628.219	86.583	31521
476c08c031 [de37a7b16ab1d9ec]	demodb	1								
		2	6257.31	9.24	3100	1098.931	0.012	4284.943	1233.238	5694

Top tables by blocks fetched

DB	Tablespace	Schema	Table	I	Heap		Ix		TOAST		TOAST-Ix	
					Blks	%Total	Blks	%Total	Blks	%Total	Blks	%Total
demodb	pg_default	i6c	i6_n_m	1	536961	0.25	2418551	1.10				
				2	9481392382	53.64	357341693	2.02				
demodb	pg_default	i6c	i6_d_t_d	1	36214123	16.53	108835089	49.68				
				2	1656550069	9.37	4975941917	28.15				
demodb	pg_default	i6c	i6_d_t_m	1	17166613	7.84	1239130	0.57				
				2	696556409	3.94	31248149	0.18	4	0.00	66	0.00
demodb	pg_default	i6c	i6_n_a	1	574752	0.26	692931	0.32				
				2	91667256	0.52	4863695	0.03				

How can *pg_profile* help us?

- Issue report findings:
  - Leading three statements consumed 65% of time and 91% of blocks
  - Leading two tables with indexes consumed 93% of blocks

- Differential report findings:
  - Leading statements executed 4 orders of magnitude longer
  - The slowest query returned 4 orders of magnitude more rows
  - There are 4 orders of magnitude more blocks got from top table

Investigation results

- We’ve got their names
- We’ve got their IDs and texts
- We’ve got absolute numbers

What is next?

- Application optimization
- Query optimization
- Compare results

pgpro_pwr

- Execution statistics at the plan level
- Wait statistics based on wait sampling

Thank you!

pg_profile	https://github.com/zubkov-andrei/pg_profile
Postgres Pro Standard 13	https://bit.ly/3nOmUL8
Postgres Pro Enterprise 13	https://bit.ly/2QOo7q2

Andrey Zubkov,
Postgres Professional, 2021
a.zubkov@postgrespro.ru