Monitoring and Scaling Postgres at Datadog

Seth Rosenblum, Datadog @SethRosenblum

Seth Rosenblum
Data Reliability Engineering Lead

- Kafka
- Elasticsearch
- Cassandra
- Postgres!

@SethRosenblum

@datadoghq
SaaS-based monitoring
Trillions of data points per day
We’re hiring!
https://jobs.datadoghq.com

Collecting data is cheap; not having it when you need it can be expensive

Collecting data is cheap; not having it when you need it can be expensive

...So instrument all the things!

Metrics

What metrics do we gather?

connections
commits
rollbacks
disk_read
buffer_hit
rows_returned
rows_fetched
rows_inserted
rows_updated
rows_deleted
database_size
deadlocks
temp_bytes

temp_files
bgwriter.checkpoints_timed
bgwriter.checkpoints_requested
bgwriter.buffers_checkpoint
bgwriter.buffers_clean
bgwriter.maxwritten_clean
bgwriter.buffers_backend
bgwriter.buffers_alloc
bgwriter.buffers_backend_fsync
bgwriter.write_time
bgwriter.sync_time
locks
seq_scans

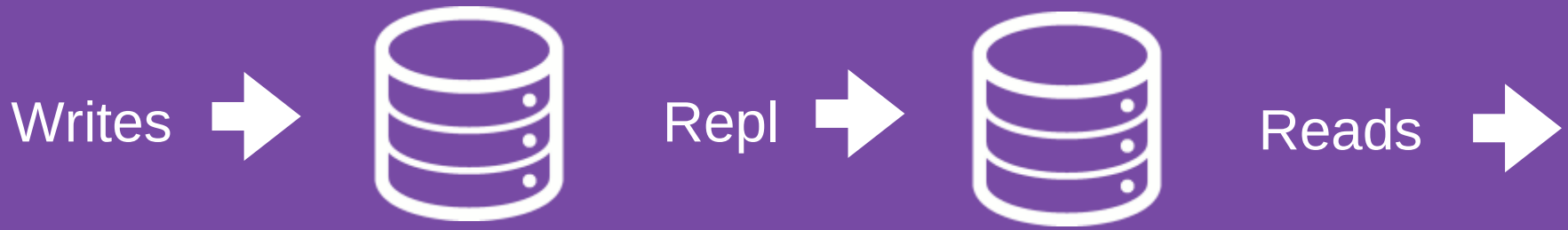
seq_rows_read
index_scans
index_rows_fetched
rows_hot_updated
live_rows
dead_rows
index_rows_read
table_size
index_size
total_size
table.count
max_connections
percent_usage_connections

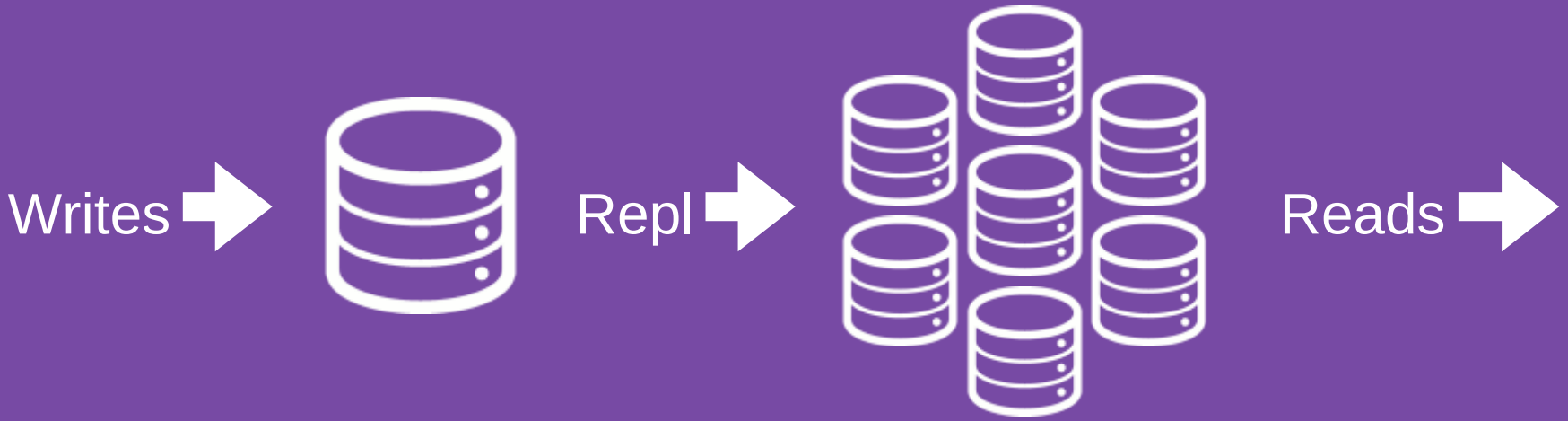
replication_delay
replication_delay_bytes
heap_blocks_read
heap_blocks_hit
index_blocks_read
index_blocks_hit
toast_blocks_read
toast_blocks_hit
toast_index_blocks_read
toast_index_blocks_hit

Scaling PostgreSQL at Datadog

Moar Resources!

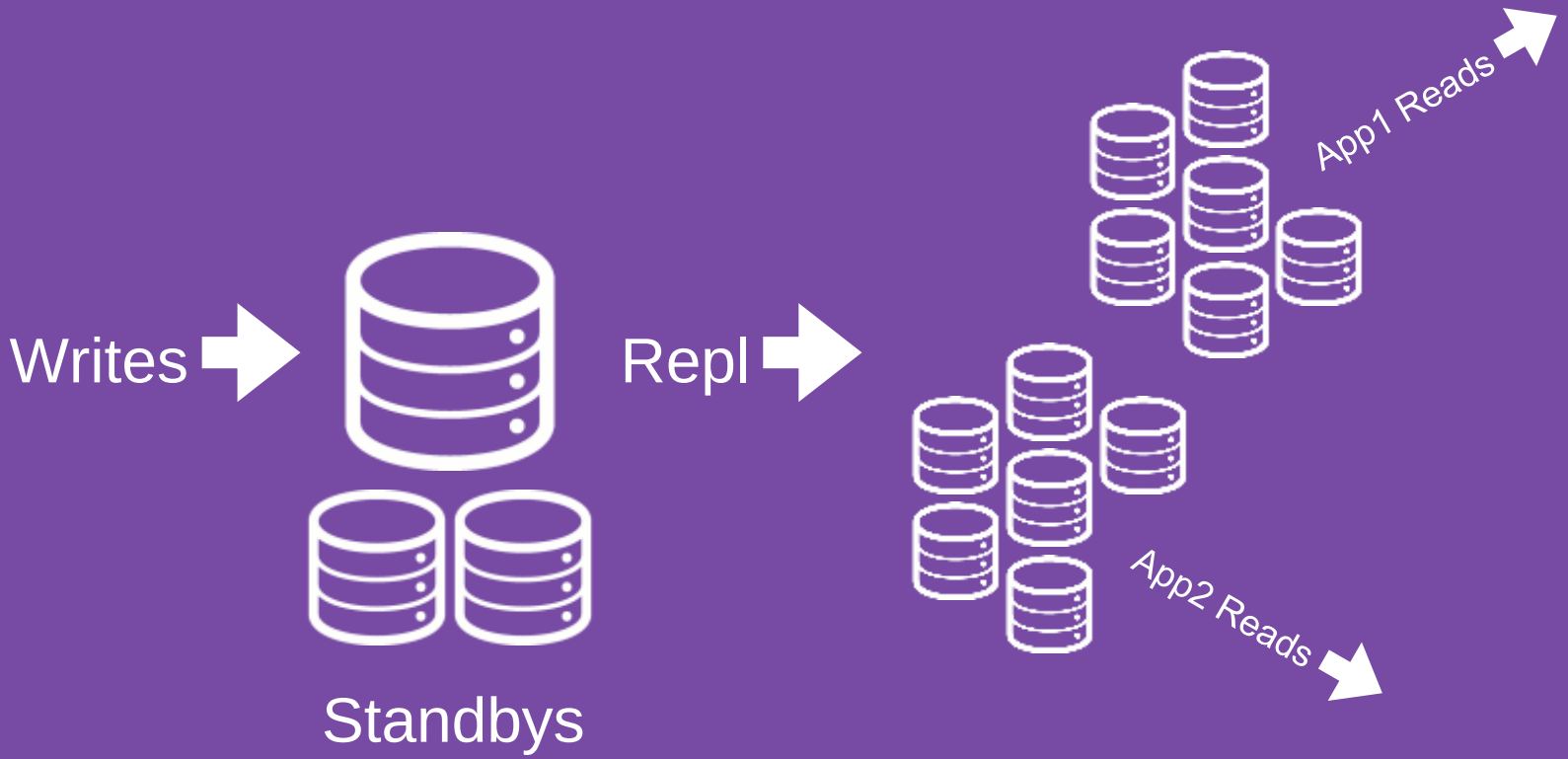
Moar Instances!


Writes
Repl
Reads


Writes
Repl
Reads

“Postgres performance-optimizes a lot better when it has a consistent workload”

Josh Berkus


Writes
Repl
App1 Reads
App2 Reads
Standbys

Requirements

- Write master is writeable, read replicas are readable!

Requirements

- Write master is writeable, read replicas are readable!
- Read replicas are up to date and don’t lag

Requirements

- Write master is writeable, read replicas are readable!
- Read replicas are up to date and don’t lag
- Additional read replicas can be provisioned quickly

Solutions

- PostgreSQL!
  - http://bit.ly/pg-repl-docs
- WAL-E
  - https://github.com/wal-e/wal-e

Solutions

- PostgreSQL!
  - http://bit.ly/pg-repl-docs
- ~~WAL-E~~ WAL-G
  - https://github.com/wal-g/wal-g

Requirements

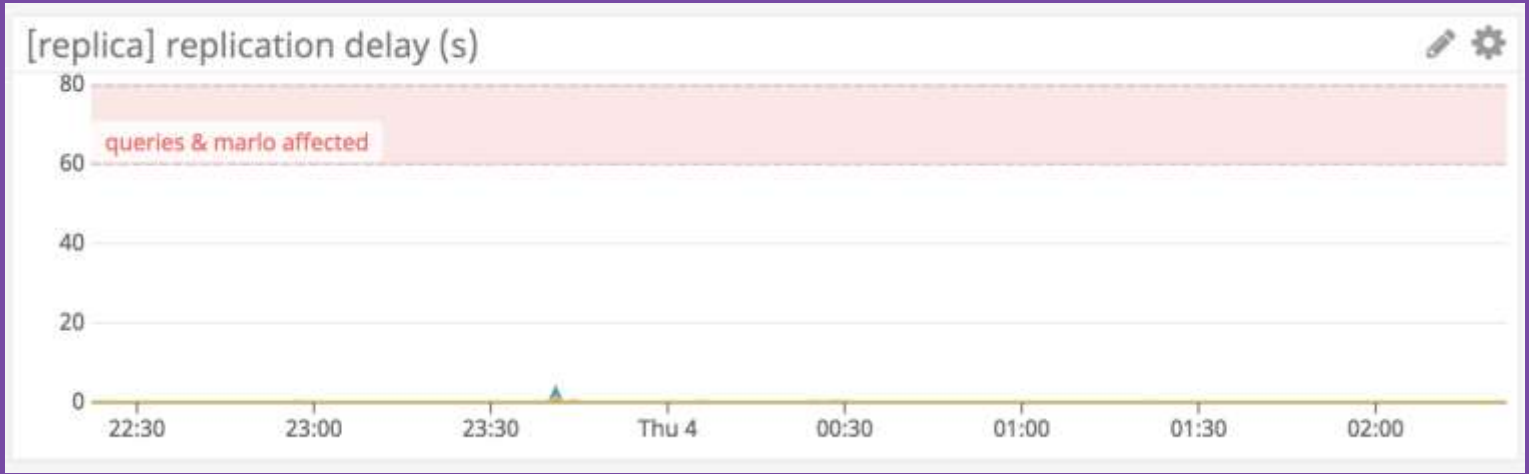
- Write master is writeable, read replicas are readable!
- Read replicas are up to date and don’t lag
- Additional read replicas can be provisioned quickly

What are we alerting on?

- Write master is writeable, read replicas are readable!
  - Up/Down checks
  - Latency

What are we alerting on?

- Read replicas are up to date and don’t lag
  - Write master standby availability
  - Write master standby replication lag
  - Read replica lag


[replica] replication delay (s)
80
60
40
20
0
queries & marlo affected
22:30
23:00
23:30
Thu 4
00:30
01:00
01:30
02:00

What are we alerting on?

- Additional read replicas can be provisioned quickly
  - Base backups are functioning properly

[i-0c3c018173bac31be] base-backup failed in 3524.65s

>>>> CMD <<<<

```
envdir /etc/wal-e/ /var/lib/postgresql/wal_e_venv/bin/wal-e backup-push
```

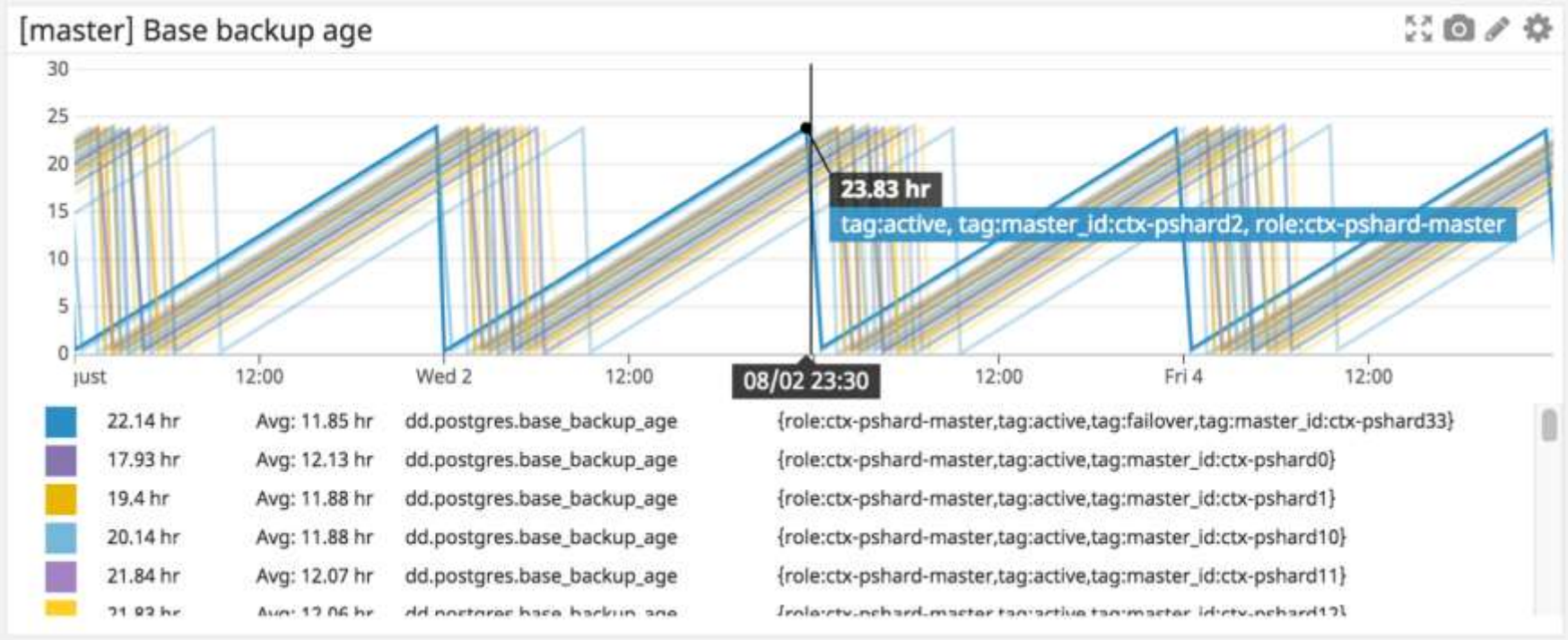
>>>> EXIT CODE <<<<

1

>>>> STDERR <<<<

```
botocore.vendored.requests.packages.urllib3.connectionpool INFO     Starting new HTTP
connection (1): 169.254.169.254
botocore.vendored.requests.packages.urllib3.connectionpool INFO     Starting new HTTP
connection (1): 169.254.169.254
botocore.vendored.requests.packages.urllib3.connectionpool INFO     Starting new HTTPS
connection (1): s3.amazonaws.com
botocore.vendored.requests.packages.urllib3.connectionpool INFO     Starting new HTTPS
connection (2): s3.amazonaws.com
botocore.vendored.requests.packages.urllib3.connectionpool INFO     Resetting dropped
connection: s3.amazonaws.com
```

When do we care that a backup has failed?


[master] Base backup age
30
25
20
15
10
5
0
23.83 hr
tag:active, tag:master_id:ctx-pshard2, role:ctx-pshard-master
12:00
Wed 2
12:00
08/02 23:30
12:00
Fri 4
12:00
22.14 hr Avg: 11.85 hr dd.postgres.base_backup_age {role:ctx-pshard-master,tag:active,tag:failover,tag:master_id:ctx-pshard33}
17.93 hr Avg: 12.13 hr dd.postgres.base_backup_age {role:ctx-pshard-master,tag:active,tag:master_id:ctx-pshard0}
19.4 hr Avg: 11.88 hr dd.postgres.base_backup_age {role:ctx-pshard-master,tag:active,tag:master_id:ctx-pshard1}
20.14 hr Avg: 11.88 hr dd.postgres.base_backup_age {role:ctx-pshard-master,tag:active,tag:master_id:ctx-pshard10}
21.84 hr Avg: 12.07 hr dd.postgres.base_backup_age {role:ctx-pshard-master,tag:active,tag:master_id:ctx-pshard11}

[Triggered] [postgres] ctx-pshard-master base backup is too old on master_id:ctx-pshard14

#account:prod •••

The last wal-e base backup taken by this host is too old, it should not exceed 24 hours by more than a few minutes. It may be failing or just taking a long time. If we failed over this postgres master recently, it may have interrupted a base backup.

1. See if the process is still running on the host `ps awux|grep backup-push` . If it is it may just be taking unusually long.
2. Try running the command in `/etc/cron.d/wal_e_base_backup` . Run it in a screen as it may take a few hours, but it will let you know what's failing.

More information can be found here: https://github.com/DataDog/devops/wiki/Postgres#backup
slack-bourbon-ops
pagerduty-Ops-Bourbon

Monitoring to Improve Performance

Slow Queries

	Name	Hits	Avg Latency ↓	Total time
☆	WITH sub_contexts SELECT key,...	90	2.17 s	196 s
☆	WITH sub_contexts SELECT key,...	58	1.49 s	86.7 s
☆	WITH sub_contexts SELECT key,...	49	1.09 s	53.2 s
☆	WITH sub_contexts SELECT key,...	194	759 ms	147 s
☆	WITH sub_contexts SELECT key,...	45	750 ms	33.7 s
☆	WITH sub_contexts SELECT key,...	22	740 ms	16.3 s
☆	WITH sub_contexts SELECT key,...	70	559 ms	39.1 s

Slow Queries

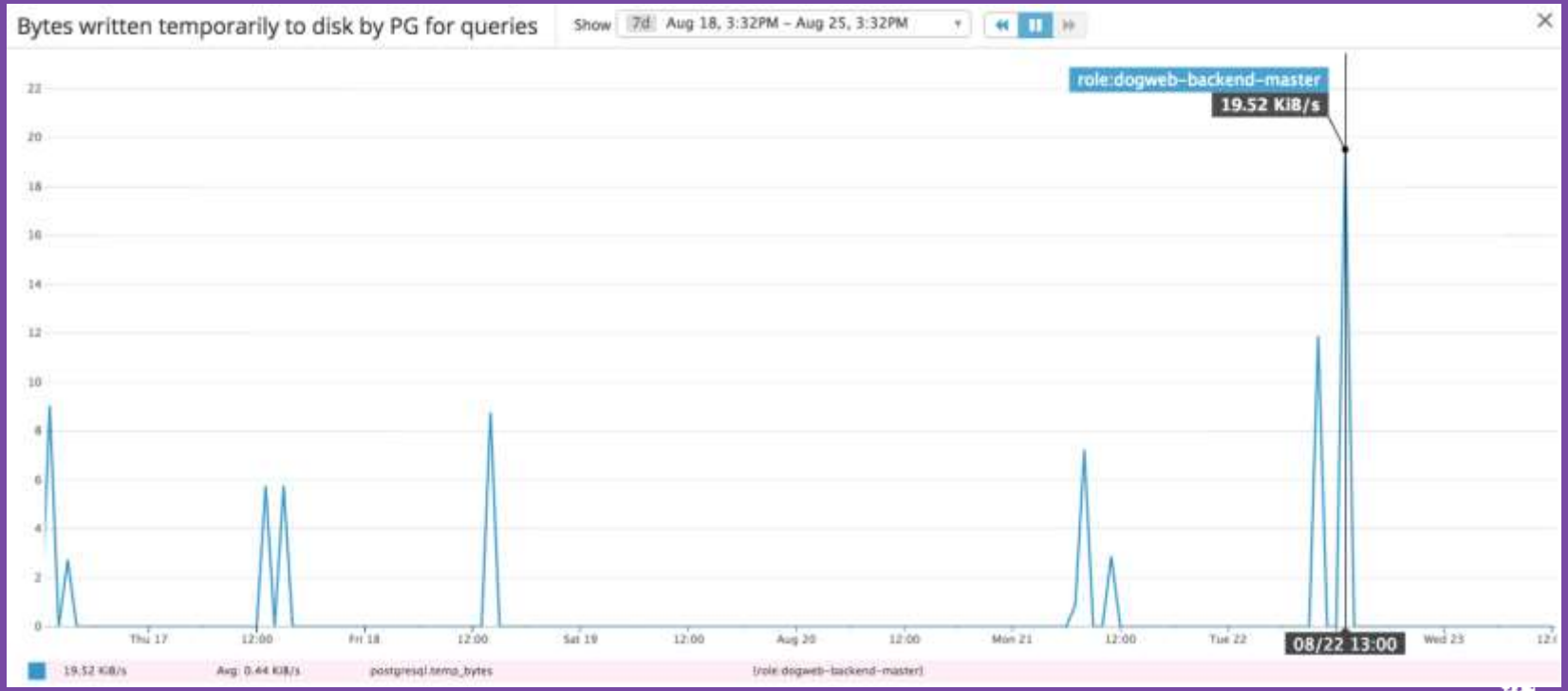
	Name	Hits	Avg Latency ↓	Total time
☆	WITH sub_contexts SELECT key,...	90	2.17 s	196 s
☆	WITH sub_contexts SELECT key,...	58	1.49 s	86.7 s
☆	WITH sub_contexts SELECT key,...	49	1.09 s	53.2 s
☆	WITH sub_contexts SELECT key,...	194	759 ms	147 s
☆	WITH sub_contexts SELECT key,...	45	750 ms	33.7 s
☆	WITH sub_contexts SELECT key,...	22	740 ms	16.3 s
☆	WITH sub_contexts SELECT key,...	70	559 ms	39.1 s

Performance:
RAM vs Disk

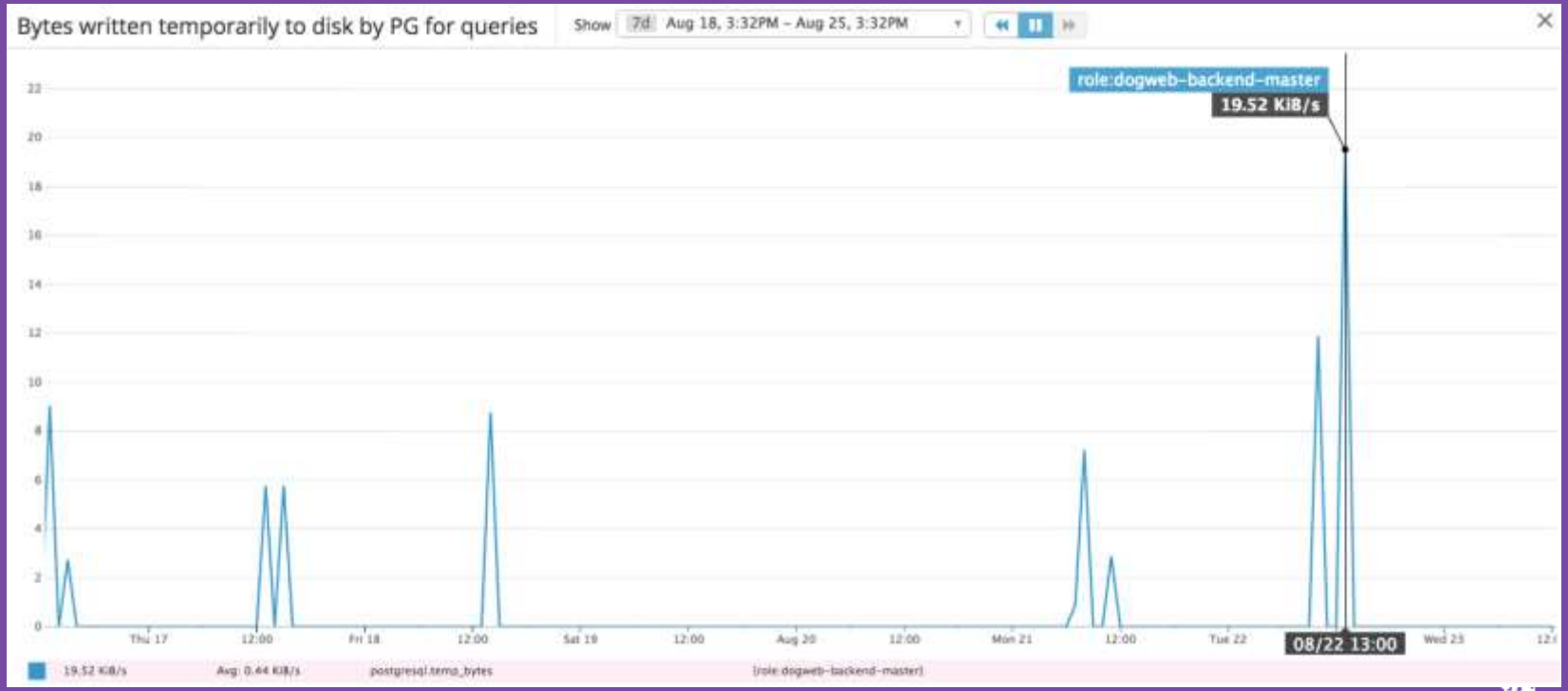
“Aside from `shared_buffers`, the most important memory-allocation parameter is `work_mem`… Raising this value can dramatically improve the performance of certain queries…”

Robert Haas

Finding **Inefficient** Queries



Finding **Inefficient** Queries



EXPLAIN ANALYZE

http://bit.ly/pg-explain

- Explain displays the execution plan

EXPLAIN ANALYZE

http://bit.ly/pg-explain

- Explain displays the execution plan
- Analyze runs it and gathers stats

EXPLAIN ANALYZE

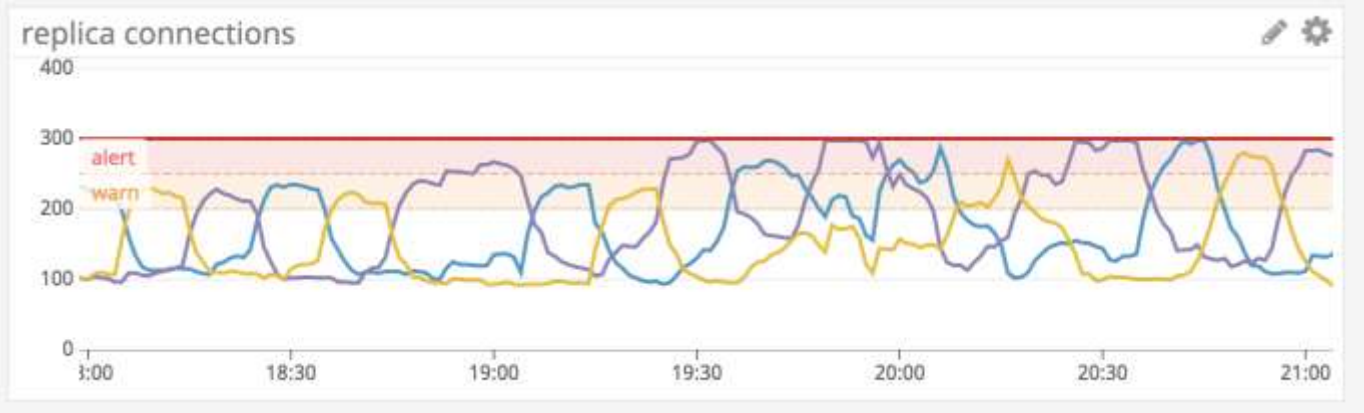
```
Merge Right Join  (cost=25870.55..31017.51 rows=229367 width=92) (actual time=2884.501..5147.047 rows=354834 loops=1)
  Merge Cond: (a.uid = b.uid)
  ->  Index Scan using foo on bar a  (cost=0.00..537.29 rows=9246 width=27) (actual time=0.049..41.782 rows=9246 loops=1)
  ->  Materialize  (cost=25870.49..27204.80 rows=106745 width=81) (actual time=2884.413..3804.537 rows=354834 loops=1)
        ->  Sort  (cost=25870.49..26137.35 rows=106745 width=81) (actual time=2884.406..3099.732 rows=111878 loops=1)
              Sort Key: b.uid
              Sort Method:  external merge  Disk: 8928kB
…
 Total runtime: 5588.105 ms
(14 rows)
```

http://bit.ly/pg-auto-explain

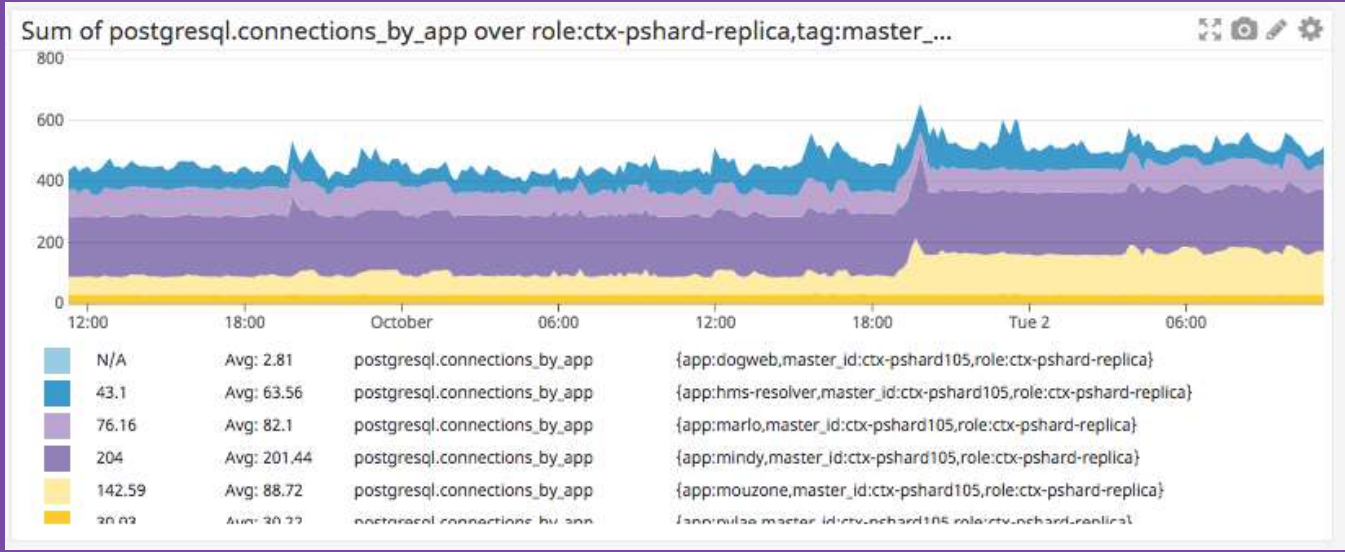
“Aside from `shared_buffers`, the most important memory-allocation parameter is `work_mem`… Raising this value can dramatically improve the performance of certain queries, **but it's important not to overdo it.**”

Robert Haas

Track Connections



Connections By Application



Summary

1. Collect as many metrics as you can, before you need them
2. If the metrics that you have aren’t providing the right value, build ones that do
3. Be aggressive in monitoring slow queries, catch them while they’re easy to find

Resources

- http://dtdg.co/monitor-postgres
- https://dtdg.co/gcp-sql
- https://dtdg.co/postgresql-vacuums

Questions?

Seth Rosenblum

@SethRosenblum

seth@datadoghq.com