OpenSCG

BIGSQL

Self Healing PostgreSQL with Predictive Analysis

- Rajesh Madiwale(Database Consultant)

BIGSQL

Introduction

I am Rajesh Madiwale with 4+ Years of exp on various DB technologies like PostgreSQL,Greenplum,Mongodb,Mysql.

Email_id : rajesh.madiwale@openscg.com

Linkedin : https://www.linkedin.com/in/rajesh-madiwale-81761940

Phone No : +91-9866423573

Designation : Database Consultant

About OpenSCG

- Started Operations in early 2010
- Around 130 member team
- Headquartered in East Brunswick, NJ
- Presence in
  - Rochester, NY
  - San Mateo, CA
  - Hyderabad, India
  - Bangalore, India

PostgreSQL Experts

Long Community Involvement & Leadership

AWS Consulting Partner

Agenda

- Regular duties of a PostgreSQL DBA
- Data capture for predictive analysis
- What is Self Healing?
- What should be automated and what should not be?
- Tools to enable automatic maintenance on PostgreSQL

Regular duties of a PostgreSQL DBA

Data capture for predictive analysis

- There are different data capturing techniques we are using to capture the data at regular intervals.
    - Custom scripts
    - Open source monitoring solutions

Understand snapshotting

- It is technically impossible to go back to a certain point in time in database

- To view the problem causing areas, without capturing a snapshot of the DB activity(usage metrics only)

- Snapshotting is a solution implemented by several database vendors like Oracle, SQL Server, etc to capture the diagnostic data/usage metrics of a database for proactive and predictive analysis.

- For PostgreSQL, pgstatspack is available to capture the DB and OS level stats.

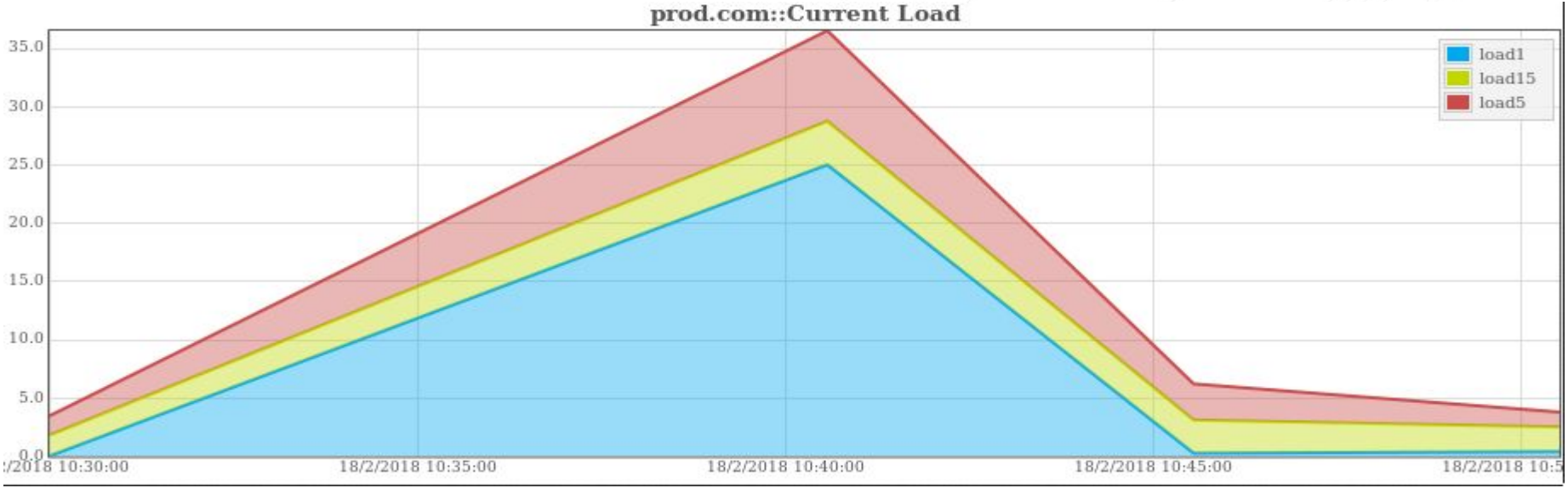
Understanding the data captured by the snapshotting tools

Here are some tables that capture OS and DB metrics:

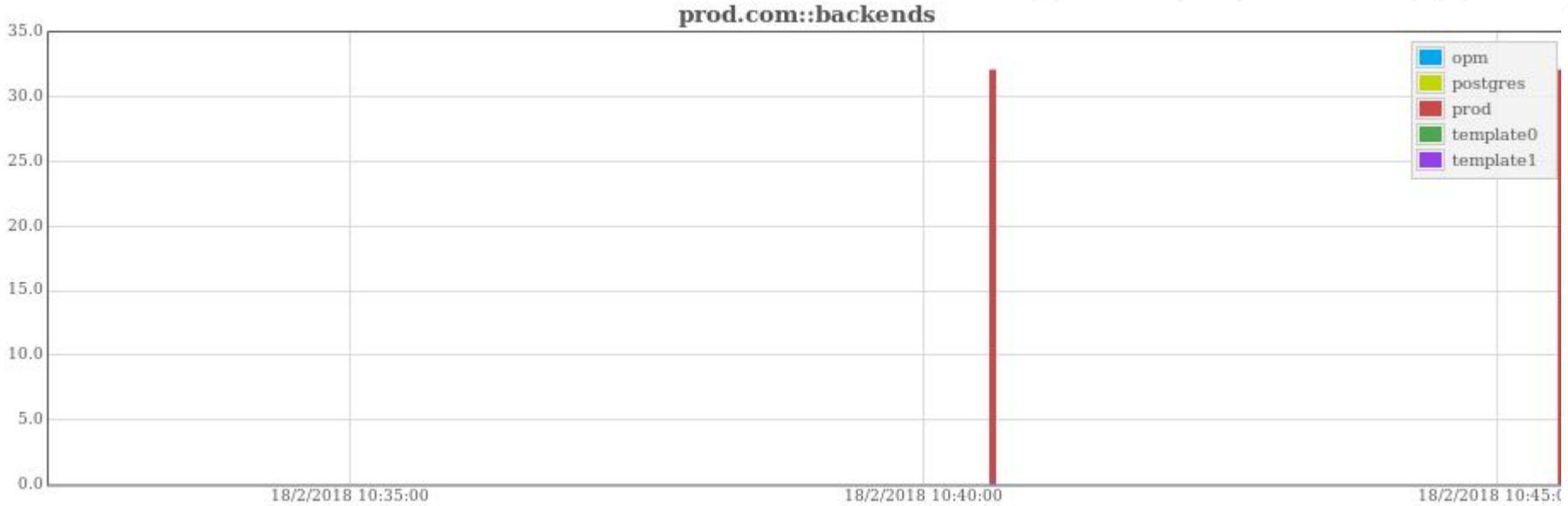
```
snap
snap_cpu
snap_databases
snap_indexes
snap_iostat
snap_load_avg
snap_mem
snap_pg_locks
snap_stat_activity
snap_stat_statements
snap_user_tables
```

Example : Let's understand the graph of Load and Number of connections during period

Load Average:



Number of connections on DB



During that interval what was running on DB which caused spike in Load?

```
prod=# select snap_id,load5,load10 from snap_load_avg where dttm between
'2018-02-18 10:35:00'::timestamp  and '2018-02-18 10:50:00'::timestamp
and load5>10;
 snap_id | load5 | load10
---------+-------+--------
      84 | 24.88 |   8.02
      85 | 14.12 |   7.19
(2 rows)
```

```
prod=# select count(*) from snap_stat_activity where snap_id = 84 and waiting = 't';
 count
-------
    30
(1 row)
```

```
prod=# select dttm,state,left(query,60) from
snap_stat_activity where snap_id=85 and state in ('active','idle in transaction') limit 5;
              dttm               | state  |                                 left
---------------------------------+--------+-------------------------------------------------------------
 2018-02-18 10:41:11.130239+05:30 | active | SELECT snapshots.save_snap()
 2018-02-18 10:41:11.130239+05:30 | active | UPDATE pgbench_accounts SET abalance = abalance + -4366 WHER
 2018-02-18 10:41:11.130239+05:30 | active | UPDATE pgbench_accounts SET abalance = abalance + -2323 WHER
 2018-02-18 10:41:11.130239+05:30 | active | UPDATE pgbench_accounts SET abalance = abalance + 257 WHERE
 2018-02-18 10:41:11.130239+05:30 | active | UPDATE pgbench_accounts SET abalance = abalance + 2527 WHERE
(5 rows)
```

What is Self Healing?

- Self Healing is not JUST fixing the issues but, it is also considered as, identifying the problem by looking at the patterns (if any) and based on the pattern automate the fixes.

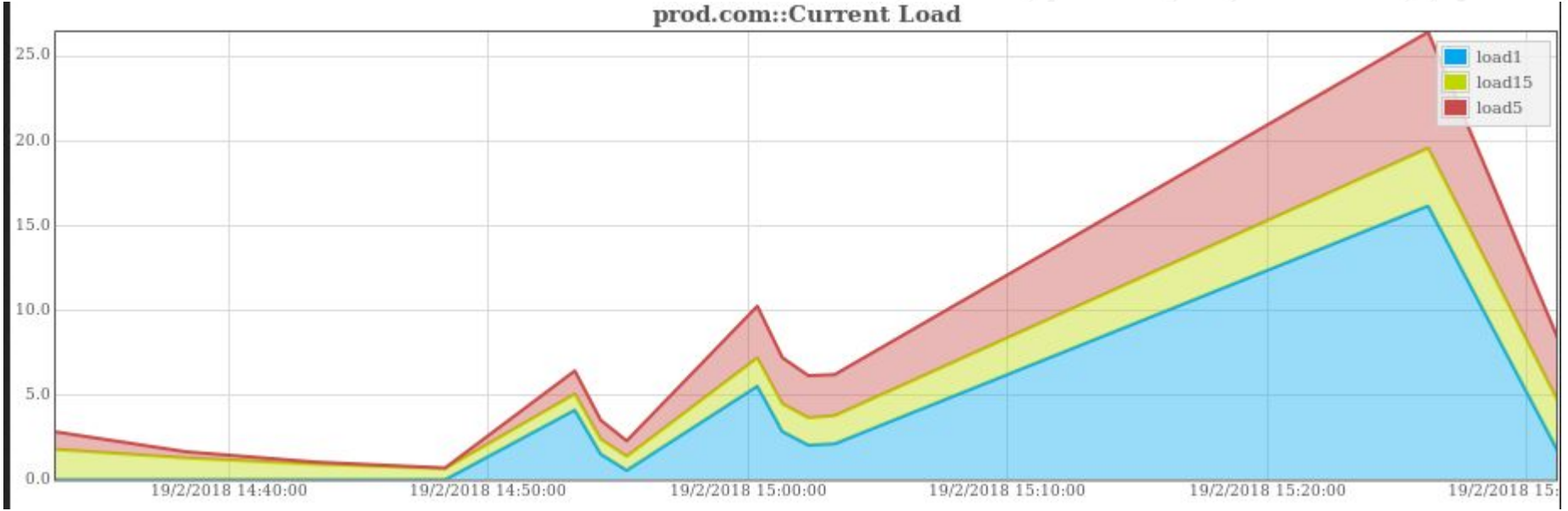
What is Self Healing?

Divided in two Steps

1. Identifying the issue by automated way
2. Fixing it using automation

Case Studies :

1: Load observed during backup



```
prod=# select * from snap_load_avg where load5>3 and dttm > '2018-02-19 15:28:00';
 snap_id |            dttm            | load5 | load10 | load15
---------+----------------------------+-------+--------+--------
     199 | 2018-02-19 15:29:30.030936 |   5.3 |   4.77 |   3.17
     200 | 2018-02-19 15:30:02.128708 |  3.74 |   4.44 |   3.11
(2 rows)
```

```
select application_name,datname,left(query,60) from snap_stat_activity where snap_id = 199
prod-# and  state='active';
 application_name | datname |                                                  left
------------------+---------+-------------------------------------------------------------
 pg_dump          | prod    | COPY public.pgbench_accounts (aid, bid, abalance, filler) TO
 pgbench          | prod    | SELECT abalance FROM pgbench_accounts WHERE aid = 1851419;
 pgbench          | prod    | SELECT abalance FROM pgbench_accounts WHERE aid = 12402832;
 pgbench          | prod    | SELECT abalance FROM pgbench_accounts WHERE aid = 17698276;
 pgbench          | prod    | SELECT abalance FROM pgbench_accounts WHERE aid = 8564836;
                  | prod    | SELECT snapshots.save_snap()
 pgbench          | prod    | SELECT abalance FROM pgbench_accounts WHERE aid = 20136615;
(7 rows)
```

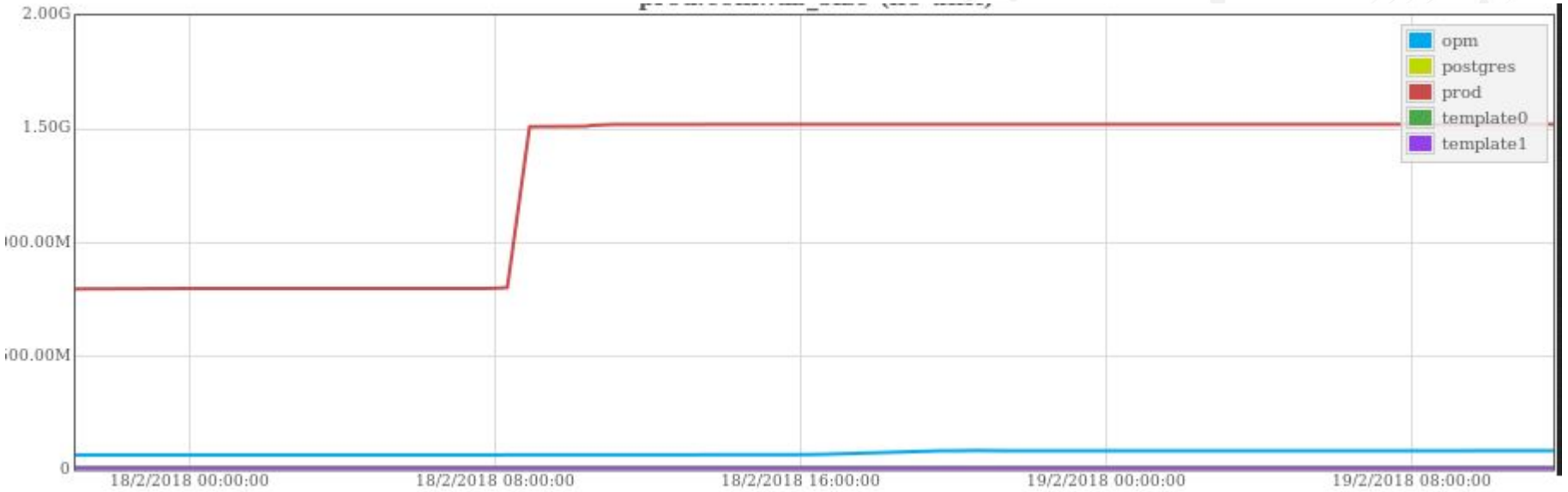
```
-bash-4.1$ ps -ef|grep pg_dump
postgres    811  68170 47 15:42 pts/5    00:00:02 pg_dump -d prod -U postgres -Fc -f prod.dmp -U postgres -v
postgres    831 123663  0 15:42 pts/4    00:00:00 grep pg_dump
-bash-4.1$ kill -TSTP 811
-bash-4.1$
-bash-4.1$
-bash-4.1$ ps -efly|grep pg_dump
T postgres    811  68170 31  80   0  4288 41959 signal 15:42 pts/5    00:00:07 pg_dump -d prod -U postgres -Fc -f prod.dmp -U postgres -v
S postgres    862 123663  0  80   0   836 25816 pipe_w 15:42 pts/4    00:00:00 grep pg_dump
-bash-4.1$
-bash-4.1$
-bash-4.1$ kill -CONT 811
-bash-4.1$
-bash-4.1$ ps -efly|grep pg_dump
R postgres    811  68170 24  80   0  4112 41959 -      15:42 pts/5    00:00:09 pg_dump -d prod -U postgres -Fc -f prod.dmp -U postgres -v
S postgres    907 123663  0  80   0   836 25816 pipe_w 15:43 pts/4    00:00:00 grep pg_dump
-bash-4.1$
```

2: Repairing the bloated indexes

- Collected stats with bloated index query.
- Observed the pattern of bloated index
- Automated through script in a maintenance window every night.

3: Capacity Planning

Captured data helps DBA to understand the growth rate of table and DB



```
prod=# select dttm,snap_id,datname,pg_size_pretty(dbsize) from snap_databases
prod-# where dttm between '2018-02-18 00:00:00'::date  and '2018-02-19 00:00:00'::date
prod-# and datname='prod' order by 1;
              dttm                 | snap_id | datname | pg_size_pretty
-----------------------------------+---------+---------+----------------
 2018-02-18 00:05:58.590505+05:30 |      18 | prod    | 761 MB
 2018-02-18 09:06:48.512821+05:30 |      37 | prod    | 1436 MB
 2018-02-18 09:07:29.363884+05:30 |      38 | prod    | 1438 MB
 2018-02-18 09:07:49.520395+05:30 |      39 | prod    | 1439 MB
 2018-02-18 09:12:44.406276+05:30 |      49 | prod    | 1440 MB
 2018-02-18 14:45:40.149221+05:30 |     181 | prod    | 1450 MB
 2018-02-18 14:46:11.578528+05:30 |     182 | prod    | 1450 MB
 2018-02-18 14:46:42.968448+05:30 |     183 | prod    | 1450 MB
(8 rows)
```

```
prod=# select datname,left(query,70),state  from snap_stat_activity where snap_id = 37 limit 5;
 datname |                                                                 left                                                                 | state
---------+----------------------------------------------------------------------------------------------------------------------------------------+--------
 prod    | SELECT snapshots.save_snap()                                                                                                           | active
 prod    | UPDATE pgbench_accounts SET abalance = abalance + -4366 WHERE aid = 18                                                                 | active
 prod    | UPDATE pgbench_accounts SET abalance = abalance + -2323 WHERE aid = 28                                                                 | active
 prod    | UPDATE pgbench_accounts SET abalance = abalance + 257 WHERE aid = 1996                                                                 | active
 prod    | UPDATE pgbench_accounts SET abalance = abalance + 2527 WHERE aid = 381                                                                 | active
(5 rows)
```

On Cloud ENV: Automat Script which will add the disk.

Generate Report for tables/Index usage.

# of connects on DB

What can be automated and what should not be ?

Can be Automated:

- Maintenance activities (REINDEX and VACUUM) - schedule
  - Autovacuum(self healing)

- Collecting long running/blocking queries and taking decision on it
  - Talk to customer and set the threshold

What can be automated and what should not be automated ?

- Managing the partitions.
- Automate Vacuum freeze (if > threshold the vacuum freeze it).

What can be automated and what should not be automated ?

Should not be automated:

- Minor/Major upgrade
- Corruption issues

Tools to enable automatic maintenance on PostgreSQL

Using VACUUM FULL or CLUSTER on production db is not always possible.
What can be alternative for VACUUM FULL or CLUSTER?

1: pg_repack
2: pg_squeeze

- Similar to repack but there is no overhead of triggers.
- Works on logical decoding (framework was introduced in PostgreSQL 9.4) and background process.

pg_repack VS pg_squeeze

pg_repack	pg_squeeze
Supports on PostgreSQL 8.3 & above	Supports on PostgreSQL 9.6 only
Only superusers can use the **pg_repack** from CLI	**pg_squeeze** (bgworker) background process can detect bloated tables automatically
Table must have a PRIMARY or UNIQUE KEY	Table must have a PRIMARY or UNIQUE KEY
Required triggers to run pg_repack	Triggers free

Questions ?

!! Thank You !!

Email_id : rajesh.madiwale@openscg.com

Linkedin: https://www.linkedin.com/in/rajesh-madiwale-81761940

OpenSCG BIGSQL