Monitor More on PostgreSQL

Introduction of New Features of pg_statsinfo

Kyotaro Horiguchi
Open source software center, NTT

Table of Contents

What is pg_statsinfo?

1.1. Motivation

Many valuable clues will have disappeared before found to be usable.

- sar / *stat / sosreport give valuable information about the **environment** on which PostgreSQL runs but,
- They offer **nothing about** PostgreSQL internals. We sometimes find clues for the cause of trouble from server logs and every server status at the time of reporting.
- **If the contents of the system catalogs or various statistics were recored, we could even find signs of trouble before it occurs.**

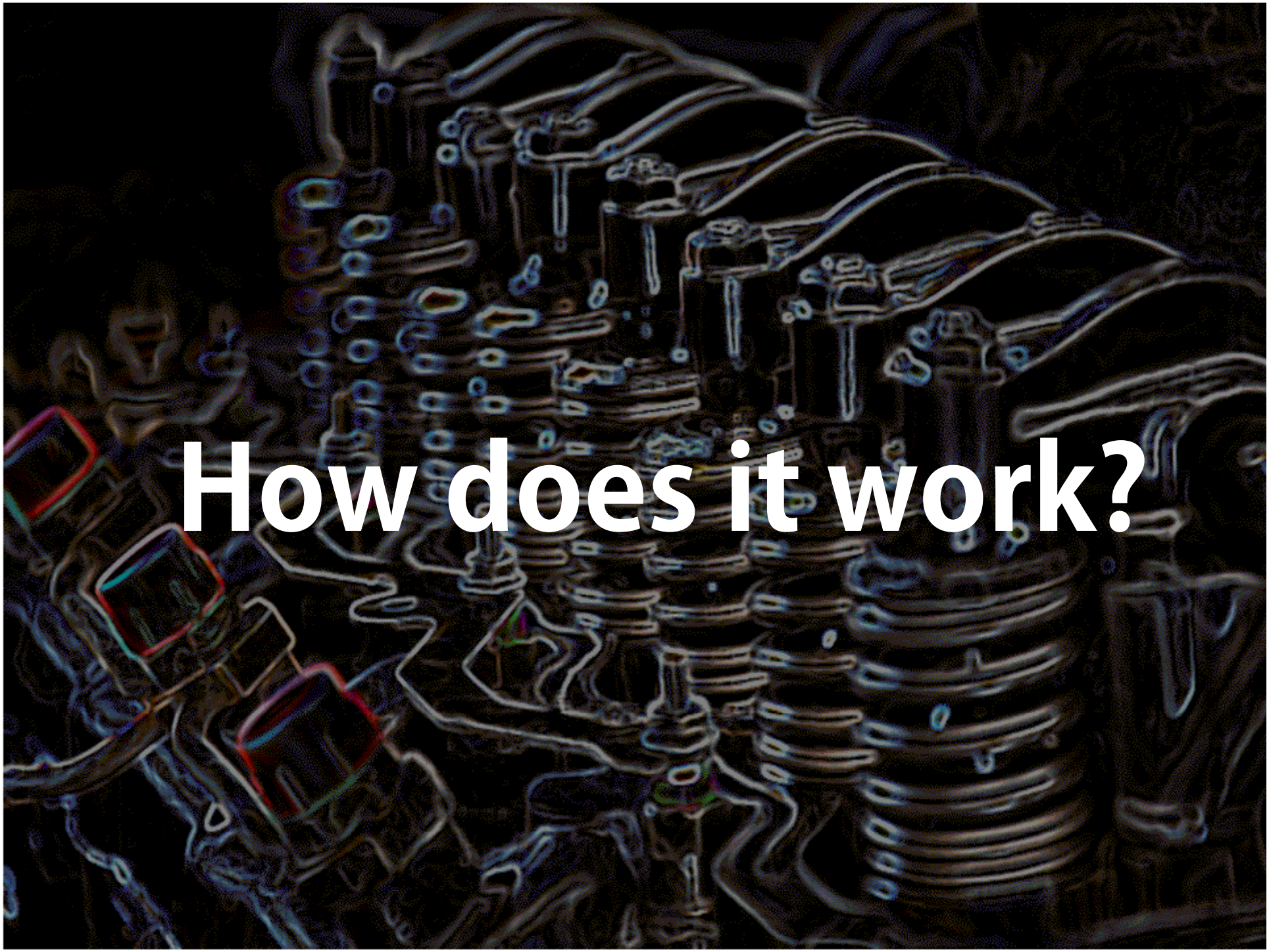
NTT

1. 2. What is pg_statsinfo?

Monitoring tool for PostgreSQL which have been developed by NTT OSS Center over years.

http://sourceforge.net/projects/pgstatsinfo

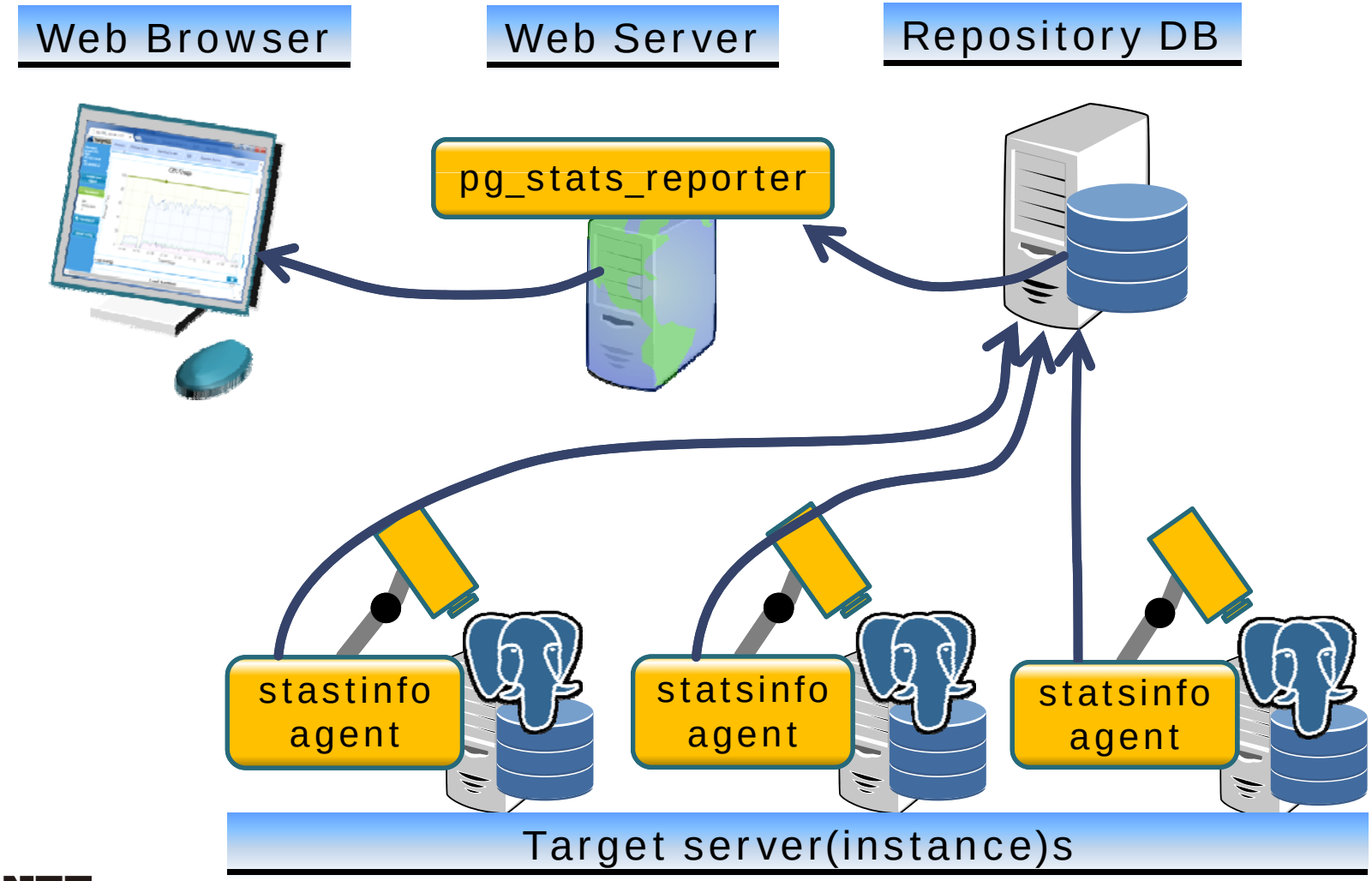
- collects various information from
  - system catalogs/views
  - server logs
  - /proc file system
  - auxiliary extensions (pg_stat_statements, pg_store_plans).
- easy to install
- has a rich viewer


How does it work?

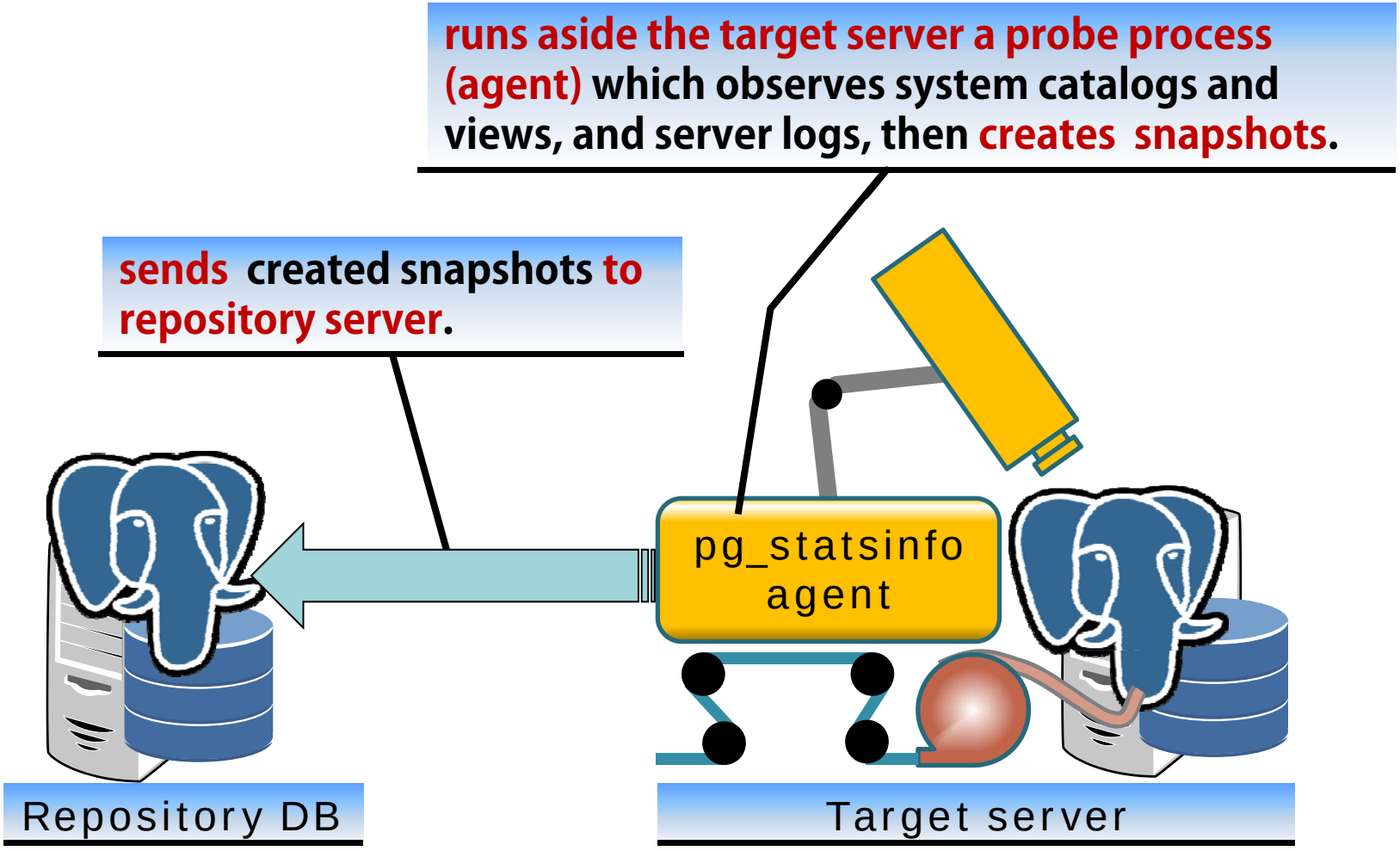
2. How does it work?

- pg_statsinfo consists of three major parts.
  - **pg_statsinfod,** an agent program runs aside Postgresql on the target to probe it and sends the result periodically to repository database.
  - **repository database,** which stores what the probe collected in the form of a time series of snapshots.
  - **pg_stats_reporter,** a graphical viewer which allows users to examine the repository in intuitive way.

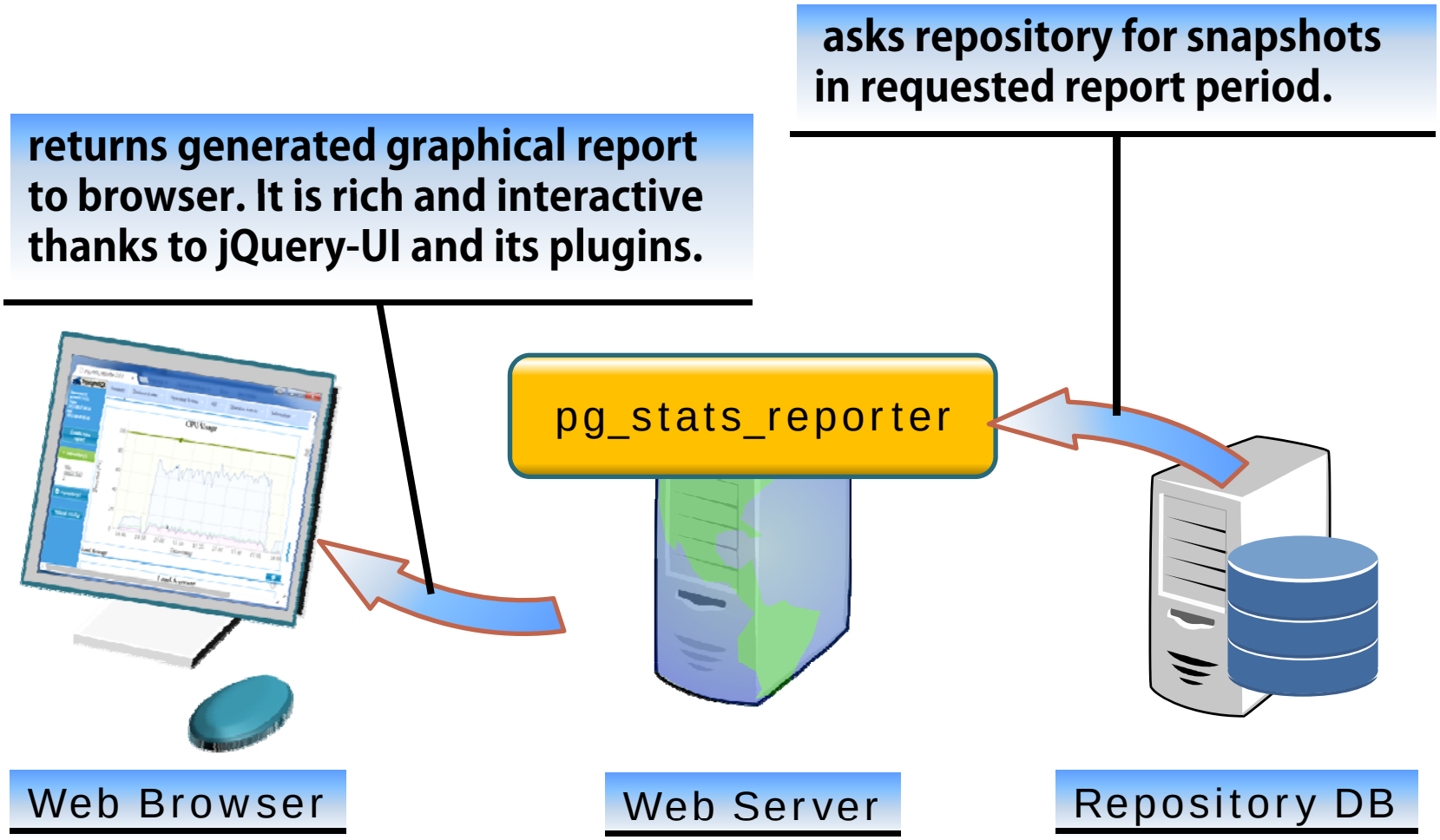
2.1. Example configuration



2.2. pg_statsinfo in detail



2.3. pg_stats_reporter in detail




Similar Products

3. Similar products (1/3)

pgFouine and **pgBadger** offer the similar functionality to pg_statsinfo but also different in some points.

The advantages of pg_statsinfo are,

- Additional information available from system catalogs/views, some extensions, and /proc.
- Statement logs are not required so server logs does not bloat.

pgFouine: http://pgfouine.projects.pgfoundry.org/
pgBadger: http://dalibo.github.io/pgbadger/

NTT

3. Similar products (2/3)

The disadvantages are,

- Required to install into every target server, including additional extensions.
  pg_stat_statements and pg_store_plans are necessary to make it fully functional.

- Some extent of performance reduction is inevitable.
  DBT-2 benchmark slowed by less than 1% in TPS, but workloads where the most queries are vary short would be more affected.

3. Similar products (3/3)

Postgres Toolkit

- A multitool to help DBAs to manage PostgreSQL servers, consists of a dozen of small tools.
- One of them, pt-stat-snapshot, is a tool which leaves snapshots of some performance views of PostgreSQL.

http://postgres-toolkit.launchrock.com/


New Features

4.1. New features - as of version 3.0

The previous version of pg_statsinfo 3.0 and pg_stats_reporter 3.0 had the following new features.

- Stores server logs in repository
- Records autovacuum/analyze statistics
- Records alerts in repository

- Stop to support PostgreSQL 8.3

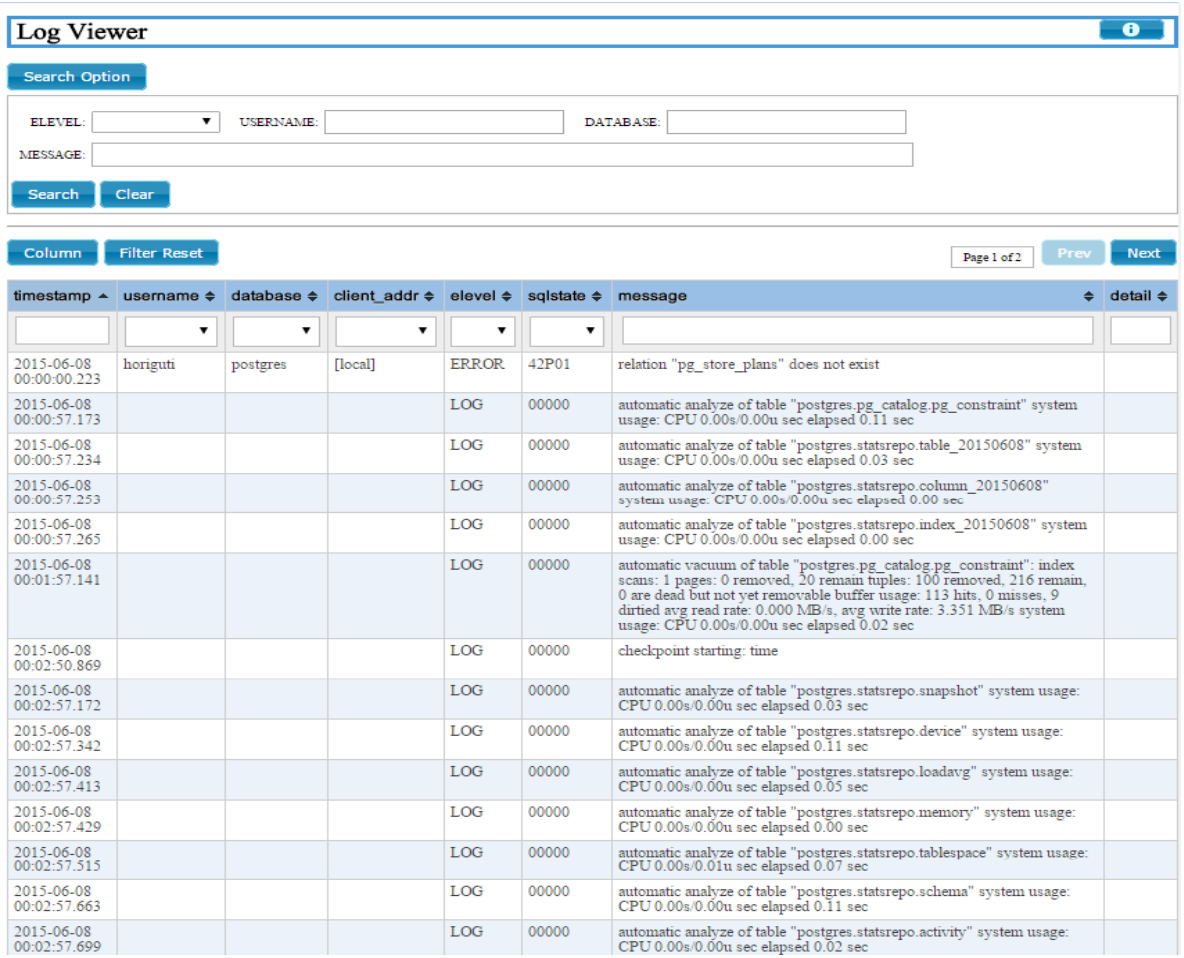
NTT

4.1.1. New features - The Log Viewer

Shows log lines for the given report period and offers on-screen quick filter and instant sorting features.

Selectable columns

Quick filter and sorting



timestamp	username	database	client_addr	elevel	sqlstate	message	detail
2015-06-08 00:00:00.223	horiguti	postgres	[local]	ERROR	42P01	relation "pg_store_plans" does not exist	
2015-06-08 00:00:57.173				LOG	00000	automatic analyze of table "postgres.pg_catalog.pg_constraint" system usage: CPU 0.00s/0.00u sec elapsed 0.11 sec	
2015-06-08 00:00:57.234				LOG	00000	automatic analyze of table "postgres.statsrepo.table_20150608" system usage: CPU 0.00s/0.00u sec elapsed 0.03 sec	
2015-06-08 00:00:57.253				LOG	00000	automatic analyze of table "postgres.statsrepo.column_20150608" system usage: CPU 0.00s/0.00u sec elapsed 0.00 sec	
2015-06-08 00:00:57.265				LOG	00000	automatic analyze of table "postgres.statsrepo.index_20150608" system usage: CPU 0.00s/0.00u sec elapsed 0.00 sec	
2015-06-08 00:01:57.141				LOG	00000	automatic vacuum of table "postgres.pg_catalog.pg_constraint": index scans: 1 pages: 0 removed, 20 remain tuples: 100 removed, 216 remain, 0 are dead but not yet removable buffer usage: 113 hits, 0 misses, 9 dirtied avg read rate: 0.000 MB/s, avg write rate: 3.351 MB/s system usage: CPU 0.00s/0.00u sec elapsed 0.02 sec	
2015-06-08 00:02:50.869				LOG	00000	checkpoint starting: time	
2015-06-08 00:02:57.172				LOG	00000	automatic analyze of table "postgres.statsrepo.snapshot" system usage: CPU 0.00s/0.00u sec elapsed 0.03 sec	
2015-06-08 00:02:57.342				LOG	00000	automatic analyze of table "postgres.statsrepo.device" system usage: CPU 0.00s/0.00u sec elapsed 0.11 sec	
2015-06-08 00:02:57.413				LOG	00000	automatic analyze of table "postgres.statsrepo.loadavg" system usage: CPU 0.00s/0.00u sec elapsed 0.05 sec	
2015-06-08 00:02:57.429				LOG	00000	automatic analyze of table "postgres.statsrepo.memory" system usage: CPU 0.00s/0.00u sec elapsed 0.00 sec	
2015-06-08 00:02:57.515				LOG	00000	automatic analyze of table "postgres.statsrepo.tablespace" system usage: CPU 0.00s/0.01u sec elapsed 0.07 sec	
2015-06-08 00:02:57.663				LOG	00000	automatic analyze of table "postgres.statsrepo.schema" system usage: CPU 0.00s/0.00u sec elapsed 0.11 sec	
2015-06-08 00:02:57.699				LOG	00000	automatic analyze of table "postgres.statsrepo.activity" system usage: CPU 0.00s/0.00u sec elapsed 0.02 sec	

4.1.2. New features - autovacuum stats

Statistics of autovacuum and analyze are useful to check the health of the server or to find signes of trouble.

Autovacuum Overview

Database	Schema	Table	Count	Avg index scans	Avg removed rows	Avg remain rows	Avg remain dead	Avg duration (sec)	Max duration (sec)	Cancels
dbt2	public	district	59	0.119	99.356	121.8	20.373	0.027	0.74	2
dbt2	pg_ca..	pg_st..	1	1	61	491	0	0.06	0.06	0
dbt2	pg_to..	pg_to..	5	1	56.4	81	4.4	1.03	4.79	0
dbt2	public	wareh..	36	0.028	56.167	21.389	10.528	0.005	0.13	0

Analyze Overview

Database	Schema	Table	Count	Total duration (sec)	Avg duration (sec)	Max duration (sec)	Last analyzed	Cancels	Max modified rows
dbt2	public	wareh..	60	2.17	0.036	0.05	2015/3/10 17:12	0	69
dbt2	public	district	60	1.18	0.02	0.13	2015/3/10 17:12	2	246
dbt2	public	new_..	1	0.66	0.66	0.66	2015/3/10 16:51	0	8963

4.2. New features – latest version

The latest version of pg_statsinfo and pg_stats_reporter to have the following new features.

- **Execution plan statistics**
- AUTOANALYZE cancellations statistics
- Some numbers available in PostgreSQL9.4.
  - Number of tuples remain dead after AUTOVACUUM
  - Number of tuples modified since last ANALYZE
  - WAL archive statistics
- **Peak rate of disk reads/writes**
- **Documentation and helps are extensively revised.**

NTT

4.2.1. New features - Plan Statistics

Plan statistics lists plans executed for a query along with the similar numbers with query statistics. This is useful to trace the transition of plans for a query.

Query ID	User	Database	Plan count	Calls	Total time	Time/call	Block rd time	Blockwr time
79661	horiguti	postgres	3	285998	2044.929	0.007	0	0
UPDATE pgbench_branches SET bbalance = bbalance + ? WHERE bid = ?;								
Plan ID	**Calls**	**Total time**	**Time/call**	**Block r time**	**Block w time**	**First call**	**Last call**	
Plan (child row)								
2361498583	164673	1236.273	0.008	0	0	2015/6/9 18:01	2015/6/9 18:09	
Update on pgbench_branches (cost=4.14..8.16 rows=1 width=370) -> Bitmap Heap Scan on pgbench_branches (cost=4.14..8.16 rows=1 width=370) Recheck Cond: (bid = 2) -> Bitmap Index Scan using pgbench_branches_pkey (cost=0.00..4.14 rows=1 width=0) Index Cond: (bid = 2)								
273038856	69201	492.439	0.007	0	0	2015/6/9 18:05	2015/6/9 18:09	
Update on pgbench_branches (cost=0.13..8.15 rows=1 width=370) -> Index Scan using pgbench_branches_pkey on pgbench_branches (cost=0.13..8.15 rows=1 width=370) Index Cond: (bid = 2)								
4132319976	52124	316.217	0.006	0	0	2015/6/9 17:59	2015/6/9 18:01	
Update on pgbench_branches (cost=0.00..8.09 rows=1 width=370) -> Seq Scan on pgbench_branches (cost=0.00..8.09 rows=1 width=370) Filter: (bid = 4)								

4.5. New features – I/O Peak Graph

The I/O peak graph shows the maximum number among the averages for every sampling interval (5 seconds by default), during each snapshot interval. This describes more precise situation where sudden increase in I/O usage took place.


pg_store_plans
Index Cond: ((instid = $2) AND ("timestamp" >= $0) AND
InitPlan 1 (returns $0)
-> Aggregate (cost=27.31..27.32 rows=1 width=8)
-> Bitmap Heap Scan on snapshot (cost=9.99..26.75
Recheck Cond: (snapid >= 3032::bigint)
-> Bitmap Index Scan using snapshot_pkey
InitPlan 2 (returns $1)
-> Aggregate (cost=30.26..30.27 rows=1 width=8)
-> Seq Scan on snapshot snapshot_1 (cost=0.00..27.55
Filter: (snapid <= 3277::bigint)
InitPlan 3 (returns $2)
-> Index Scan using snapshot_pkey on snapshot snapshot_2
Index Cond: (snapid = 3277::bigint)
-> Index Scan using statsrepo_autoanalyze_cancel_idx on
(cost=66.03..74.06 rows=1 width=136)
Index Cond: ((instid = $5) AND ("timestamp" >= $3) AND

5. What is pg_store_plans?

`http://osdn.jp/projects/pgstoreplans/`

- Collects stats of execution plans in similar way to pg_stat_statements.
- Developed based on pg_stat_statements for the purpose of providing pg_statsinfo with the plan stats feature.
- Almost same to pg_stat_plans, but the most significant difference with pg_stat_plans is that pg_store_plans holds explain representation for later use.

5.1. Why pg_store_plans

- Explain representations are required to be held for later use in repository database.
- To reduce required storage, execution plans are stored in the form of “shortened JSON”, which effectively reduces snapshot size, too.


```
{"p":{"t":"b","!":"u","n":"snapshot","a":"snapshot","1":27.63,"2":35.65,"
3":1,"4":33,"l":[{"t":"z","g":"p","h":"I","q":"InitPlan 2 (returns $1)",
"1":27.34,"2":27.35,"3":1,"4":4,"l":[{"t":"h", "h":"I“,"q":"InitPlan 1
(returns $0)","n":"pg_namespace","a":"pg_namespace","1":0.00,
"2":1.10,"3":1,"4":4,"5":"(nspname = 'statsrepo'::name)"},{"t":"h",
"h":"o","n":"pg_class","a":"pg_class","1":0.00,"2":25.79,"3":90,"4":4,
"5":"(relnamespace = $0)"}]},{"t":"i","h":"m","d":"f",
"i":"snapshot_pkey","n":"snapshot","a":"snapshot","1":0.28,"2":8.30,
"3":1,"4":33,"8":"(snapid = 3165::bigint)"}]}}
```


An Example of Internal Format for Plan Strings

5.2. pg_store_plans in detail

Most of the columns of pg_store_plans are the same to pg_stat_statements. The following is the list of columns peculiar to pg_store_plans.

queryid
- hash of the query calculated by pg_store_plans

planid
- hash of the plan

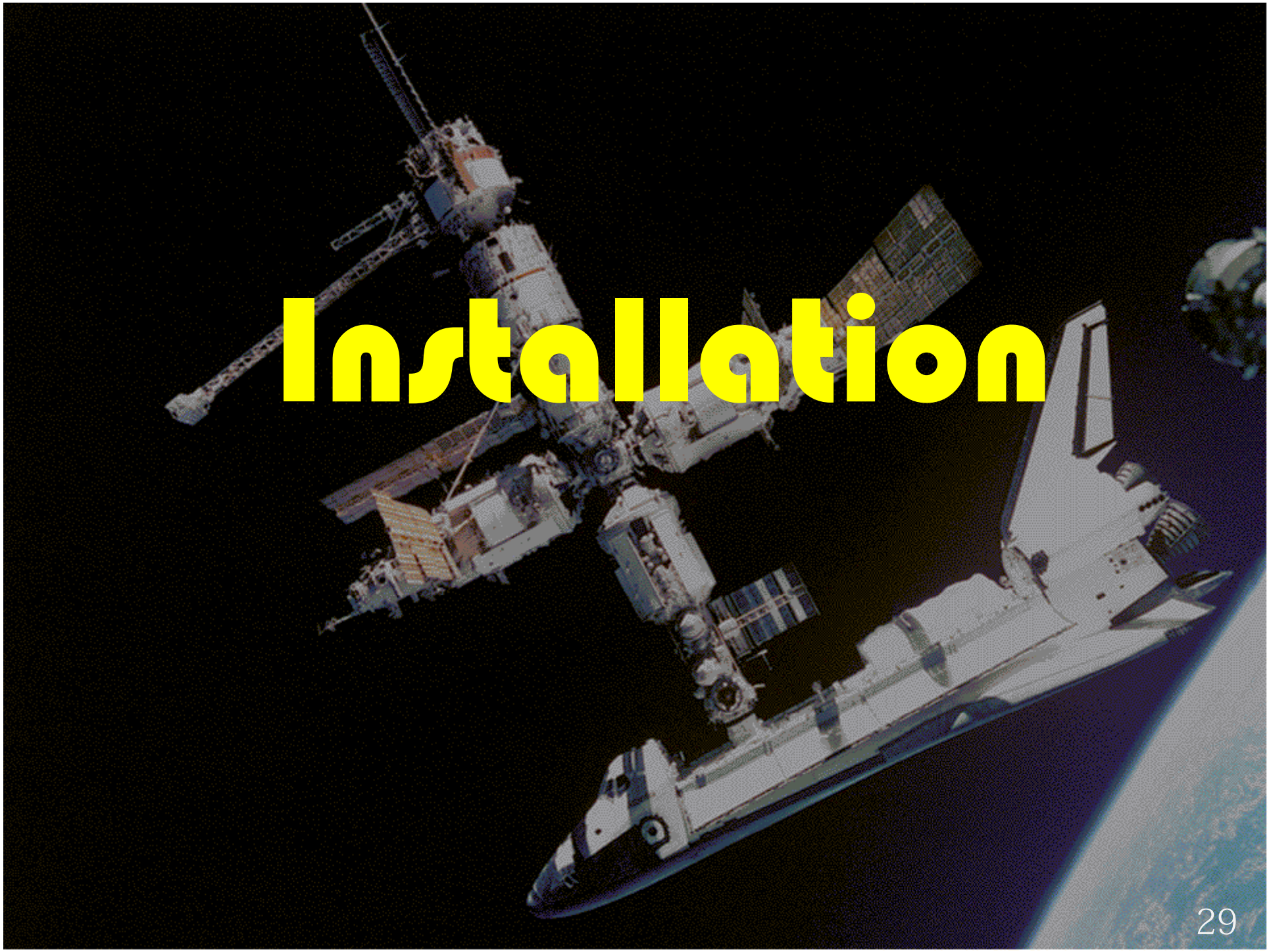
queryid_stat_statements
- query ID calculated by pg_stat_statements, if any

plan
- plan representation

first_call
- time when the plan executed first

last_call
- time when the plan is most recently executed

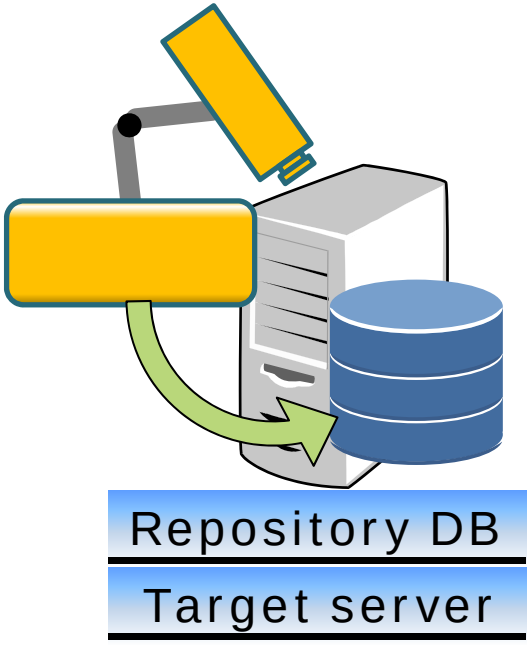
NTT


Installation

6.1. Minimul installation of pg_statsinfo

For the minimal setup, pg_statsinfo needs only two lines to be added to postgresql.conf after installing binaries. pg_statsinfo uses the target server itself as repository.

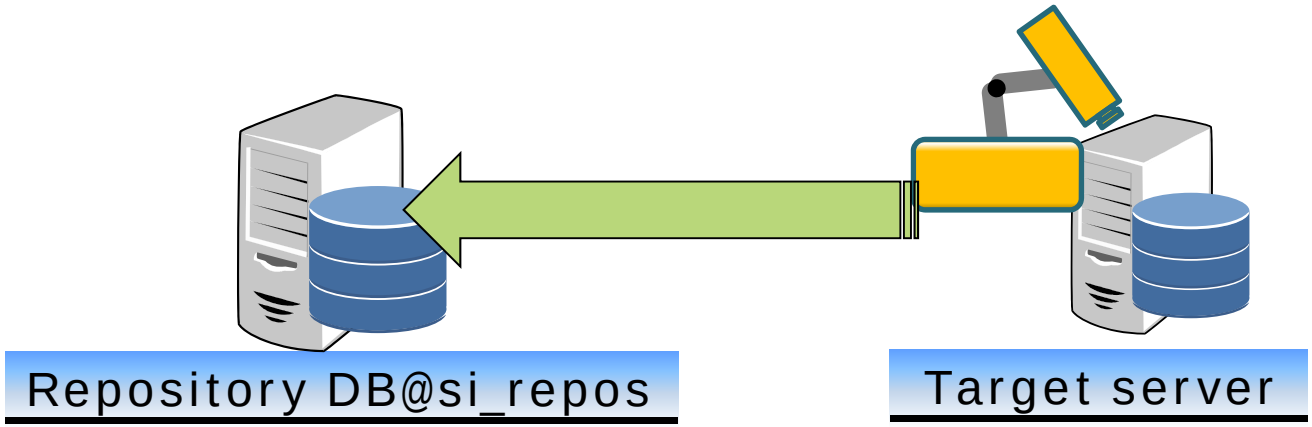
```
shared_preload_libraries = 'pg_statsinfo'
```



6.2. Typical installation of pg_statsinfo

As the more realistic setup, placing repository on dedicated server, collecting additional information and login in as non-superuser, the additional setup looks like this on the target.

```
pg_statsinfo.repository_server = ‘hostname=si_repos dbname=repos user=repos’
shared_preload_libraries = 'pg_statsinfo, pg_stat_statements, pg_store_plans‘
log_checkpoints = yes
log_autovacuum_min_duration = 0
track_io_timing = yes
track_functions = all
track_activities = yes
```



6.3. Installation of pg_stats_reporter

pg_stats_reporter runs as a PHP script on httpd. Some php-related packages are needed to be installed.

```
# yum install httpd php php-pgsql php-intl php-cli
# rpm –ivh pg_stats_reporter-3.1.0-el7.noarch.rpm
```

Firewall and SELinux needs additional setup.

```
# firewall-cmd –-zone=public –-add-service=http
# setsebool -P httpd_can_network_connect_db 1
# semanage fcontext -a -t httpd_sys_rw_content_t /var/www/pg_stats_reporter_lib/cache
# semanage fcontext -a -t httpd_sys_rw_content_t /var/www/pg_stats_reporter_lib/compiled
# restorecon -v /var/www/pg_stats_reporter_lib/cache
# restorecon -v /var/www/pg_stats_reporter_lib/compiled
```

NTT


1
2
3
4
DEMONSTRATION

7. Demonstration

Since it is quite boring to demonstrate placing binaries and SELinux setup, so they are skipped in this demo.

```
## Installing binaries of php, httpd, PostgreSQL,
##                        pg_stat_statements, pg_store_plans,
##                        pg_statsinfo/pg_stats_reporter
firewall-cmd –zone=public –-add-service=http
setsebool -P httpd_can_network_connect_db 1
semanage fcontext -a -t httpd_sys_rw_content_t /var/www/pg_stats_reporter_lib/cache
semanage fcontext -a -t httpd_sys_rw_content_t /var/www/pg_stats_reporter_lib/compiled
restorecon -v /var/www/pg_stats_reporter_lib/cache
restorecon -v /var/www/pg_stats_reporter_lib/compiled
```

7.1 Demo 1 Quick Run

This demo shows the minimal setup to run.
Access to http://localhost/pg_stats_reporter/pg_stats_reporter.php
after the steps following, then you will see the online report.

```
# 1. initialize the cluster
initdb

# 2. add the following to $PGDATA/postgresql.conf
shared_preload_libraries = 'pg_statsinfo, pg_store_plans, pg_stat_statements'

pg_statsinfo.sampling_interval = 1s
pg_statsinfo.snapshot_interval = 3s

log_line_prefix='%m [%p] '
log_checkpoints = yes
log_autovacuum_min_duration = 0
track_io_timing = yes
track_functions = all
track_activities = yes

# 3. start the server and register the additional extensios.
pg_ctl start -w
psql postgres -c 'create extension pg_store_plans'
psql postgres -c 'create extension pg_stat_statements‘

# 4. run a benchmark
pgbench -i postgres
pgbench -c 10 -T 30 postgres
#  - Then, stop pg_statsinfo
pg_statsinfo --stop -d postgres

## 5. pg_stats_reporter shows data in the time resolution of 1 minutes,
##    so tweak the timestamps of the snapshots to fit the limitaion.

psql postgres -c 'update statsrepo.snapshot set time = now() - ((select max(snapid) from
statsrepo.snapshot) * 60)::text::interval + (snapid * 60)::text::interval;'
```

NTT

7.2. Creating offline report

Offline report consists of a collection of html and related files. The following steps will give you an offline report

```
# Creating offline report for the whole period where snapshots populate.
pg_stats_reporter -O myreport
```

Thank you

Thank you for your kind attention.

ご静聴ありがとうございました