PoWA Documentation

Release 1.2.1

Dalibo

Jun 26, 2017

Contents

Warning: This is the documentation for PoWA 1.x series. If you're looking for the new (2.x series), please visit the latest documentation If you are using PostgreSQL >= 9.4, you should use the 2.x series.

PoWA is an extension designed to historize informations given by the *pg_stat_statements extension*. It provides sql SRF (Set Returning Functions) to gather useful information on a specified time interval. If possible (verify with pg_test_timing), also activate track_io_timing in postgresql.conf.

PoWA requires PostgreSQL 9.3 or more.

Connecting on the GUI requires a PostgreSQL user with SUPERUSER and LOGIN privileges.

CHAPTER 1

PostgreSQL Workload Analyzer detailled installation guide

Read *the introduction* for further details about PoWA.

PoWA requires **PostgreSQL 9.3 or more**. **The contrib must also be available**.

The following documentation describes the detailed installation steps to install PoWA.

Download PoWA from the website

```
wget https://github.com/dalibo/powa/archive/REL_1_2_1.zip
```

Unpack the downloaded file

```
cd /usr/src
unzip powa-REL_1_2_1.zip
```

Compile and install the software

Before proceeding, be sure to have a compiler installed and the appropriate PostgreSQL development packages. Something like

```
apt-get install postgresql-server-dev-9.3
```

or

```
yum install postgresql93-devel
```

Then:

```
cd /usr/src/powa-REL_1_2_1
make
```

If everything goes fine, you will have this kind of output :

```
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-
↪labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -
↪fexcess-precision=standard -g -fpic -I. -I. -I/home/thomas/postgresql/postgresql-9.
↪3.4/include/server -I/home/thomas/postgresql/postgresql-9.3.4/include/internal -D_
↪GNU_SOURCE -I/usr/include/libxml2   -c -o powa.o powa.c
gcc -Wall -Wmissing-prototypes -Wpointer-arith -Wdeclaration-after-statement -Wendif-
↪labels -Wmissing-format-attribute -Wformat-security -fno-strict-aliasing -fwrapv -
↪fexcess-precision=standard -g -fpic -L/home/thomas/postgresql/postgresql-9.3.4/lib -
↪Wl,--as-needed -Wl,-rpath,'/home/thomas/postgresql/postgresql-9.3.4/lib',--enable-
↪new-dtags  -shared -o powa.so powa.o
```

Install the software :

This step has to be made with the user that has installed PostgreSQL. If you have used a package, it will be certainly be root. If so:

```
sudo make install
```

Else, sudo into the user that owns your PostgreSQL executables, and

```
make install
```

It should output something like the following :

```
/bin/mkdir -p '/usr/pgsql-9.3/share/extension'
/bin/mkdir -p '/usr/pgsql-9.3/share/extension'
/bin/mkdir -p '/usr/pgsql-9.3/lib'
/bin/mkdir -p '/usr/pgsql-9.3/share/doc/extension'
/usr/bin/install -c -m 644 ./powa.control '/usr/pgsql-9.3/share/extension/'
/usr/bin/install -c -m 644 ./powa--1.1.sql ./powa--1.2.sql ./powa--1.1--1.2.sql  '/
↪usr/pgsql-9.3/share/extension/'
/usr/bin/install -c -m 755  powa.so '/usr/pgsql-9.3/postgresql-9.3.4/lib/'
/usr/bin/install -c -m 644 ./README.md '/usr/pgsql-9.3/share/doc/extension/'
```

Create a PoWA database and create required extensions

Note: if you are upgrading from a previous PoWA release, please consult the upgrading section at the end of this file.

First, connect to PostgreSQL as administrator :

```
bash-4.1$ psql
psql (9.3.5)
Type "help" for help.
postgres=# create database powa;
CREATE DATABASE
postgres=# \c powa
You are now connected to database "powa" as user "postgres".
powa=# create extension pg_stat_statements ;
CREATE EXTENSION
powa=# create extension btree_gist ;
CREATE EXTENSION
```

```
powa=# create extension powa;
CREATE EXTENSION
powa=# \dt
                        List of relations
  Schema |                Name                 | Type  |  Owner
---------+---------------------------------+--------+----------
  public | powa_functions                  | table | postgres
  public | powa_last_aggregation           | table | postgres
  public | powa_last_purge                 | table | postgres
  public | powa_statements                 | table | postgres
  public | powa_statements_history         | table | postgres
  public | powa_statements_history_current | table | postgres
(6 rows)
```

Modify the configuration files

In *postgresql.conf*:

Change the *shared_preload_libraries* appropriately :

```
shared_preload_libraries = 'powa,pg_stat_statements'# (change requires restart)
```

If possible (check with pg_test_timing), activate track_io_timing on your instance, in postgresql.conf :

```
track_io_timing = on
```

Other GUC variables are available. Read *the main documentation* for further details.

In *pg_hba.conf*:

Add an entry if needed for the PostgreSQL user(s) that need to connect on the GUI. For instance, assuming a *local connection* on database *powa*, allowing any user:

host powa all 127.0.0.1/32 md5

Restart PostgreSQL

As root, run the following command :

```
service postgresql-9.3 restart
```

PostgreSQL should output the following messages in the log files :

```
2014-07-25 03:48:20 IST LOG:  registering background worker "powa"
2014-07-25 03:48:20 IST LOG:  loaded library "powa"
2014-07-25 03:48:20 IST LOG:  loaded library "pg_stat_statements"
```

Upgrading from a previous version of PoWA

If you already have an older PoWA installation, you can simply upgrade PoWA with the following steps :

First, connect to PostgreSQL as administrator and update the extension :

```
bash-4.1$ psql powa
psql (9.3.5)
Type "help" for help.
powa=# ALTER EXTENSION powa UPDATE ;
ALTER EXTENSION
```

Next, you will need to restart PostgreSQL in order to take account of the updated background worker. As root, run the following command :

```
service postgresql-9.3 restart
```

Finally, adapt the ui/powa.conf file to suit the new format. For instance,

- if coming from powa 1.1

```
"database" : {
    "dsn"      : "dbi:Pg:database=powa;host=127.0.0.1;port=5432",
    "options"  : {
        "AutoCommit" : 0,
        "pg_enable_utf8" : 1
    }
},
```

must be changed to

```
"servers" : {
    "main" : {
        "dbname"    : "powa",
        "host"      : "127.0.0.1",
        "port"      : "5432"
    }
},
```

- if coming from powa 1.2

```
"database" : {
    "dbname"   : "powa",
    "host"     : "127.0.0.1",
    "port"     : "5432",
    "options"  : {
        "AutoCommit" : 0,
        "pg_enable_utf8" : 1
    }
},
```

must be changed to

```
"servers" : {
    "main" : {
        "dbname"    : "powa",
        "host"      : "127.0.0.1",
        "port"      : "5432"
    }
},
```

Set-up the UI

Read *the ui documentation* for details.

CHAPTER 2

PostgreSQL Workload Analyzer User Interface

Overview

You can run the POWA User Interface in various ways : as Perl webservice (Morbo), as a CGI with Apache or with Nginx (as a reverse proxy in front of Hypnotoad).

But first let's talk about safety:

/!WARNING /!

You need to be careful about the security of your PostgreSQL server when installing POWA

We designed POWA so that the user interface will only communicate with PostgreSQL via prepared statements. This will prevent the risk of SQL injection.

However to connect to the POWA User Interface, you will use the login and password of a postgeSQL superuser. See *the main documentation* for more details. If you don't protect your communications, an attacker placed between the GUI and PostgreSQL, or between you and the GUI, could gain superuser rights to your database server.

Therefore we **strongly** recommend the following precautions :

- Read the Great PostgreSQL Documentation
- Check your `pg_hba.conf` file
- Do not allow users to access POWA from the Internet
- Do not allow users to access PostgreSQL from the Internet
- Run POWA on a HTTPS server and disable HTTP access
- Use SSL to protect the connection between the GUI and PostgreSQL
- Reject unprotected connections between the GUI and PostgreSQL (`hostnossl .... reject`)
- Check your `pg_hba.conf` file again

Prerequisites

The versions showed have been tested, it may work with older versions

- Perl 5.10
- Mojolicious 4.75 and later
- Perl DBI and DBD-Pg modules
- PostgreSQL 9.3
- A CGI/Perl webserver

Browser compatibility

PoWA is designed to respect standards and should work on most standard compliant browser.

If you are using an old version of Microsoft Internet Explorer, you might not see some graphs. We recommend that you either :

- upgrade to a version (at least IE9)
- or switch to Mozilla Firefox.

Install

Install powa extension and configure it as seen it main README.md file.

Install other prerequisites: Mojolicious is available on CPAN and sometimes packages, for example the package in Debian is *libmojolicious-perl*

If the needed version is not available anymore on your distribution, you can download Mojolicious 4.75 here.

As you are then not using a package, you may not want to install Mojolicious globally on your system. So here is how to install it locally (let's say you installed powa in /path/to/powa):

> perl Makefile.PL PREFIX=/path/to/powa/mojo make install

Check that make tells you the files have been copied to /path/to/powa/mojo.

Now, you'll just have to tell perl that there is an extension in /path/to/powa/mojo (we'll see that with the morbo command below).

Copy *powa.conf-dist* to *powa.conf* and edit it.

If you have multiple PostgreSQL servers with PoWA installed, you can configure them in the *powa.conf* file, in the **servers** section. Each entry is of the form **"name": { info... }**, and must be coma separated.

For instance, if you have a production server listening on 10.0.0.1, port 5432 and a development server listening on 10.0.0.2, port 5433, the **servers** section should look like :

```
...
"servers" : {
    "production" : {
        "dbname"    : "powa",
        "host"      : "10.0.0.1",
        "port"      : "5432"
    },
    "development" : {
        "dbname"    : "powa",
        "host"      : "10.0.0.2",
        "port"      : "5433"
    }
},
...
```

You can also optionally configure a user and a password for each server. If credentials are found in the config file for the selected server, they will override what a user could provide on the login page, and of course the fields will become optionnal on the login page.

For instance, if you have a “test” server with must use the username “dba” and the password “testing”, the **powa.conf** file will look like :

```
...
"servers" : {
    "test" : {
    "development" : {
        "dbname"    : "powa",
        "host"      : "10.0.0.3",
        "port"      : "5433",
        "username" : "dba",
        "password" : "testing"
    }
}
...
```

CAREFUL: If you use this feature, it’s strongly advised to rely on an external security method, such as what is built-in on most of the http servers.

CAREFUL: If upgrading from PoWA 1.1 or PoWA 1.2, you need to change the format of the database section. See INSTALL.md in PoWA main directory for more details.

Run With Morbo

To quickly run the UI, do not activate *rewrite* in the config (this is Apache rewrite rules when run as a CGI) and start the morbo webserver inside the source directory:

> morbo script/powa

If you have installed Mojolicious locally, you’ll have to do this command instead (the paths may vary depending on where you run this command from):

> PERL5LIB=/path/to/powa/mojo/share/perl5/site_perl mojo/bin/site_perl/morbo ui/script/powa

Of course, putting PERL5LIB and PATH in your .bashrc file wouldn’t be a bad idea...

It will output what is printed to STDOUT/STDOUT in the code in the term. The web pages are available on http://localhost:3000/

Run With Apache

To run the UI with Apache, here is an example using CGI:

```
<VirtualHost *:80>
    ServerAdmin webmaster@example.com
    ServerName powa.example.com
    DocumentRoot /var/www/powa/public/

    <Directory /var/www/powa/public/>
        AllowOverride None
        Order allow,deny
        allow from all
        IndexIgnore *

        RewriteEngine On
        RewriteBase /
        RewriteRule ^$ powa.cgi [L]
        RewriteCond %{REQUEST_FILENAME} !-f
        RewriteCond %{REQUEST_FILENAME} !-d
        RewriteRule ^(.*)$ powa.cgi/$1 [L]
    </Directory>

    ScriptAlias /powa.cgi /var/www/powa/script/powa
    <Directory /var/www/powa/script/>
        AddHandler cgi-script .cgi
        Options +ExecCGI
        AllowOverride None
        Order allow,deny
        allow from all
        SetEnv MOJO_MODE production
        SetEnv MOJO_MAX_MESSAGE_SIZE 4294967296
    </Directory>

    ErrorLog ${APACHE_LOG_DIR}/powa.log
    # Possible values include: debug, info, notice, warn, error, crit,
    # alert, emerg.
    LogLevel warn

    CustomLog ${APACHE_LOG_DIR}/powa.log combined
</VirtualHost>
```

Run with Nginx

If you want ot use Nginx, the best solution is probably to run Hypnotoad behind a reverse proxy:

More details here : http://mojolicio.us/perldoc/Mojolicious/Guides/Cookbook#Nginx

CHAPTER 3

Installation

For a detailed installation procedure, please read *the installation guide*.

Optionally, you can create a dedicated user for PoWA. For instance, connected on PostgreSQL : *CREATE USER powa SUPERUSER ENCRYPTED PASSWORD 'mypassword'* (don't forget to change the password).

- make install in the main directory
- Make sure you have installed and configured *pg_stat_statements*
- create a dedicated database (powa for instance)
- create extension powa in this databse
- add "powa" in the *shared_preload_libraries* in postgresql.conf (you should already have configured "*pg_stat_statements*")
- configure GUC in postgresql.conf (see the §Configuration below)
- configure connections in pg_hba.conf to allow connection from the server that will run the GUI
- restart instance

Upgrade from previous version:

- make install in the main directory
- restart your PostgreSQL engine to use the new powa library
- ALTER EXTENSION powa UPDATE; – This will take some time, a lot of things are rewritten as the schema is upgraded
- If you have deadlock messages, it means that the powa extension is trying to update data, while your update is doing conflicting operations. To solve this, put powa.frequency=-1 to deactivate powa temporarily, then do the extension update, and put powa.frequency back to what it was before. Don't forget to reload your configuration each time.

CHAPTER 4

Configuration:

Here are the configuration parameters (GUC) available:

- *powa.frequency* : Defines the frequency of the snapshots. Minimum 5s. You can use the usual postgresql time abbreviations. If not specified, the unit is seconds. Defaults to 5 minutes. Setting it to -1 will disable powa (powa will still start, but it won't collect anything anymore, and wont connect to the database).
- *powa.retention* : Automatically purge data older than that. If not specified, the unit is minutes. Defaults to 1 day.
- *powa.database* : Defines the database of the workload repository. Defaults to powa.
- *powa.coalesce* : Defines the amount of records to group together in the table.

The more you coalesce, the more PostgreSQL can compress. But the more it has to uncompact when queried. Defaults to 100.

If you can afford it, put a rather high work_mem for the database powa. It will help, as the queries used to display the ui are doing lots of sampling, implying lots of sorts.

We use this: *ALTER DATABASE powa SET work_mem TO '256MB';*

It's only used for the duration of the queries anyway, this is not statically allocated memory.

CHAPTER 5

Reset the stats:

SELECT powa_stats_reset(); (in the powa database of course)

CHAPTER 6

Impact on performances

Using POWA will have a small negative impact on your PostgreSQL server performances. It is hard to evaluate precisely this impact but we can analyze it in 3 parts :

- First of all, you need to activate the *pg_stat_statements* module. This module itself may slow down your instance, but some benchmarks show that the impact is not that big.
- Second, the POWA collector should have a very low impact, but of course that depends on the frequency at which you collect data. If you do it every 5 seconds, you'll definitely see something. At 5 minutes, the impact should be minimal.
- And finally the POWA GUI will have an impact too if you run it on the PostgreSQL instance, but it really depends on many user will have access to it.

All in all, we strongly feel that the performance impact of POWA is nothing compared to being in the dark and not knowing what is running on your database. And in most cases the impact is lower than setting `log_min_duration_statement = 0`.

See our own benchmark for more details: POWA vs The Badger