Watch your elephants
PostgreSQL Performance Analysis using collectd

Sebastian „tokkee“ Harl <sh@teamix.net>

teamix GmbH / collectd core team

PGConf.EU 2012
October 24, 2012
Prague
**WARNING**: I’m not a database (performance) expert!

This talk is an overview about a tool that may be used for the purpose of performance analysis.
Solid IT-Infrastructure
Location: Nuremberg, Munich, Frankfurt
http://teamix.net/

Open-Source  Monitoring  Network
N-IX  NetApp  Juniper
Riverbed  VMWare  Trainings

Watch your elephants
collectd Overview

Overview
Main Features

Feature Overview

PostgreSQL Processes

Querying statistics from PostgreSQL
collectd overview

- **collectd** collects performance data of systems
- some (simple) examples:
  - CPU utilization
  - memory utilization
  - network traffic
- **collectd** collects and stores the performance data
- stored data is usually used to generate graphs
- → performance analysis, capacity planning
- not to be confused with *monitoring*
- Homepage: http://collectd.org/
Main features

- daemon
- free software (mostly GPL)
- portable (Linux, *BSD, Solaris, ...)
- scalable (OpenWrt, ..., Cluster / Cloud)
- sophisticated network support
- efficient (default resolution: 10 seconds)
- flexible architecture
- modular (more than 100 plugins in Version 5.1)
Main features

- daemon
- free software (mostly GPL)
- portable (Linux, *BSD, Solaris, ...)
- scalable (OpenWrt, ..., Cluster / Cloud)
- sophisticated network support
- efficient (default resolution: 10 seconds)
- flexible architecture
- modular (more than 100 plugins in Version 5.1)
10 seconds resolution

Watch your elephants
Main features

- daemon
- free software (mostly GPL)
- portable (Linux, *BSD, Solaris, . . .)
- scalable (OpenWrt, . . ., Cluster / Cloud)
- sophisticated network support
- efficient (default resolution: 10 seconds)
- flexible architecture
- modular (more than 100 plugins in version 5.1)
available plugins (version 5.1)

<table>
<thead>
<tr>
<th>amqp</th>
<th>apache</th>
<th>apcups</th>
<th>apple_sensors</th>
<th>ascent</th>
</tr>
</thead>
<tbody>
<tr>
<td>battery</td>
<td>bind</td>
<td>conntrack</td>
<td>contextswitch</td>
<td>cpu</td>
</tr>
<tr>
<td>cpufreq</td>
<td>csv</td>
<td>curl</td>
<td>curl_json</td>
<td>curl_xml</td>
</tr>
<tr>
<td>dbi</td>
<td>df</td>
<td>disk</td>
<td>dns</td>
<td>email</td>
</tr>
<tr>
<td>entropy</td>
<td>ethstat</td>
<td>exec</td>
<td>filecount</td>
<td>fscache</td>
</tr>
<tr>
<td>GenericJMX.java</td>
<td>ipvs</td>
<td>hddtemp</td>
<td>interface</td>
<td>ipmi</td>
</tr>
<tr>
<td>iptables</td>
<td>logfile</td>
<td>irq</td>
<td>java</td>
<td>libvirt</td>
</tr>
<tr>
<td>load</td>
<td>match_regexp</td>
<td>lpar</td>
<td>madwifi</td>
<td>match_empty_counter</td>
</tr>
<tr>
<td>match_hashed</td>
<td>memcached</td>
<td>match_timediff</td>
<td>match_value</td>
<td>mbmon</td>
</tr>
<tr>
<td>md</td>
<td>mysql</td>
<td>memory</td>
<td>netapp</td>
<td>modbus</td>
</tr>
<tr>
<td>Monitorus.pm</td>
<td>nginx</td>
<td>notify</td>
<td>netlink</td>
<td>notify_email</td>
</tr>
<tr>
<td>network</td>
<td>numa</td>
<td>olsrd</td>
<td>notify_email</td>
<td>onewire</td>
</tr>
<tr>
<td>ntpd</td>
<td>OpenVZ.pm</td>
<td>nut</td>
<td>orb</td>
<td>perl</td>
</tr>
<tr>
<td>openssl</td>
<td>postgresql</td>
<td>oracle</td>
<td>perl</td>
<td>protocols</td>
</tr>
<tr>
<td>ping</td>
<td>redis</td>
<td>powerdns</td>
<td>perl</td>
<td>pinba</td>
</tr>
<tr>
<td>python</td>
<td>serial</td>
<td>rrddcached</td>
<td>processes</td>
<td>protocols</td>
</tr>
<tr>
<td>sensors</td>
<td>tail</td>
<td>swap</td>
<td>protocols</td>
<td>rrdtool</td>
</tr>
<tr>
<td>sensing</td>
<td>target</td>
<td>target_notification</td>
<td>rrdtool</td>
<td>syslog</td>
</tr>
<tr>
<td>table</td>
<td>scale</td>
<td>tcpconns</td>
<td>target_replace</td>
<td>teampeak2</td>
</tr>
<tr>
<td>target_scale</td>
<td>tcs</td>
<td>threshold</td>
<td>unix</td>
<td>teampeak2</td>
</tr>
<tr>
<td>ted</td>
<td>thermal</td>
<td>threshold</td>
<td>systemic</td>
<td>teamspeak2</td>
</tr>
<tr>
<td>uptime</td>
<td>users</td>
<td>threshold</td>
<td>teamspeak2</td>
<td>vmem</td>
</tr>
<tr>
<td>vservcer</td>
<td>wireless</td>
<td>threshold</td>
<td>teamspeak2</td>
<td>write_mongodb</td>
</tr>
<tr>
<td>write_redis</td>
<td>xms</td>
<td>threshold</td>
<td>teamspeak2</td>
<td></td>
</tr>
<tr>
<td>xmms</td>
<td></td>
<td>threshold</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Watch your elephants
collectd setup

- daemon collects data locally ⇒ runs on every client system (exceptions: SNMP, databases, etc.)
- one or more central servers
- clients push their data to the central servers
- first steps: install; select plugins; start daemon; enjoy ;-)
- provide collected data through JSON
- different frontends possible
- efficiently handles large amounts of data
- flexible configuration of graphs
Watch your elephants
collectd Overview

Feature Overview
- CPU, memory, network I/O
- Networking Support
- RRDtool Support
- Generic Plugins (Overview)

PostgreSQL Processes

Querying statistics from PostgreSQL
plugin overview

- specialized read plugins
  - CPU, memory, network interfaces, ...
- IO plugins
  - network plugin
  - RRDtool, RRDCacheD
  - Graphite
  - MongoDB, Redis
  - AMQP
- generic plugins
  - SNMP
  - tail
  - PostgreSQL
- filter-chains
configuration synopsis

LoadPlugin "cpu"
LoadPlugin "memory"
LoadPlugin "interface"
CPU, memory, network I/O

configuration synopsis

LoadPlugin "cpu"
LoadPlugin "memory"
LoadPlugin "interface"

<Plugin interface>
  Interface lo
  Interface sit0
  IgnoreSelected true
</Plugin>
CPU, memory, network I/O

Watch your elephants

Folie 16
CPU, memory, network I/O

Physical memory utilization on

- **Free**: 7.0M Min, 186.0M Avg, 546.0M Max, 338.0M Last
- **Cached**: 28.9M Min, 101.3M Avg, 583.8M Max, 80.3M Last
- **Buffered**: 817.2k Min, 66.5M Avg, 112.3M Max, 48.7M Last
- **Used**: 1.4G Min, 1.7G Avg, 1.9G Max, 1.5G Last

Watch your elephants
CPU, memory, network I/O

Watch your elephants
modes of operation

- send data ("client")
- receive data ("server")
- forward data ("proxy")
- Unicast ("point-to-point")
- Multicast ("point-to-group")
- IPv4 and IPv6

rule them all

Modes may be mixed arbitrarily.
LoadPlugin "network"

<Plugin "network">
    Server "collectd0.example.com"
    Server "collectd1.example.com"
    Server "ff18::efc0:4a42"
</Plugin>
network plugin: server / client

**synopsis: server**

LoadPlugin "network"

```xml
<Plugin "network">
  Listen "collectd0.example.com"
  Listen "ff18::efc0:4a42"
</Plugin>
```
network plugin: proxy

**synopsis: proxy**

LoadPlugin "network"

<Plugin "network">
  Listen "collectgw.extern.example.com"
  Server "collectd1.intern.example.com"
  Forward true
</Plugin>
**RRDtool plugin**

- writes data to RRD files **efficiently** → caching
- functionality now also available in RRDtool as stand-alone RRD Caching Daemon (RRDCacheD)

**synopsis**

```
LoadPlugin "rrdtool"

<Plugin "rrdtool">
  DataDir "/var/lib/collectd/rrd"
</Plugin>
```
RRDtool plugin (caching)

configuration synopsis

<Plugin "rrdtool">
  DataDir "/var/lib/collectd/rrd"
  CacheTimeout 3600  # 1 hour
  CacheFlush 86400  # 1 day
  WritesPerSecond 30
</Plugin>

- FLUSH command allows for graphing of current values
generic plugins (overview)

• idea: generic approaches rather than specialized solutions
• → user configuration determines behavior
• ⇒ new equipment does not require a new version of `collectd`
• examples: SNMP, tail, curl, DBI, PostgreSQL
collectd Overview

Feature Overview

**PostgreSQL Processes**

Querying statistics from PostgreSQL
PostgreSQL Processes

% ps ax | grep postgres
20177 ? S 0:05 /usr/lib/postgresql/9.1/bin/postgres
   -D /var/lib/postgresql/9.1/main
   -c config_file=/etc/postgresql/9.1/main/postgresql.conf
20183 ? Ss 0:09 postgres: writer process
20184 ? Ss 0:05 postgres: wal writer process
20185 ? Ss 0:04 postgres: autovacuum launcher process
20186 ? Ss 0:13 postgres: stats collector process
20312 ? Ss 2:04 postgres: collectd mail 127.0.0.1(33027) idle

• processes handling client connections:
  postgres: user database host activity
The processes plugin collects various information about (groups of) processes:
- RSS and VM size
- user and system time
- number of page-faults
- I/O estimates

Processes are selected either by process name or by regex of its command line (/proc/cmdline on Linux)
<Plugin "processes">
    ProcessMatch pg_writer "postgres:.writer.process"
    ProcessMatch pg_wal_writer "postgres:.wal.writer.process"
    ProcessMatch pg_autovacuum "postgres:.\*autovacuum"
    ProcessMatch pg_stats collector \ 
        "postgres:.stats.collector.process"
    # database connections by 'user'
    ProcessMatch pg_user_mail "postgres:.user"
    # database connections to database 'mail'
    ProcessMatch pg_db_mail "postgres:.[A-Za-z0-9]+.mail"
</Plugin>

(versions before 5.0.1 did not support whitespace in regexes)
collectd Overview

Feature Overview

PostgreSQL Processes

Querying statistics from PostgreSQL
  The PostgreSQL statistics collector
  The collectd postgresql plugin
The PostgreSQL statistics collector

postgresql.conf

# currently running command
track_activities = on
# access to tables and indices
track_counts = on
# user-defined functions
track_functions = none  # none, pl, all

Storing statistics on, for example, flash storage:

stats_temp_directory = '/mnt/flash/pg_stat_tmp'
Collecting statistics

- server processes submit statistics before going idle
- collector generates report each `PGSTAT_STAT_INTERVAL` milliseconds
- during each transaction, a snapshot of the report will be used → see `pg_stat_clear_snapshot()`
**Querying statistics**

- some predefined views
  - `pg_stat_bgwriter`
  - `pg_stat_database`
  - `pg_stat_all_indexes`
  - `pg_statio_all_tables`
  - many more (see table 27.1 in the documentation)

- also there are various functions to query single values

---

Watch your elephants
Querying statistics: example

```
sh=# select datname, numbackends, xact_commit, xact_rollback
sh=# from pg_stat_database;

<table>
<thead>
<tr>
<th>datname</th>
<th>numbackends</th>
<th>xact_commit</th>
<th>xact_rollback</th>
</tr>
</thead>
<tbody>
<tr>
<td>template1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>template0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>postgres</td>
<td>0</td>
<td>611</td>
<td>0</td>
</tr>
<tr>
<td>sh</td>
<td>1</td>
<td>661</td>
<td>12</td>
</tr>
</tbody>
</table>
```

(4 rows)
The **collectd** **postgresql plugin**

- generic plugin which collects arbitrary (numeric) values from a database
- by default, queries various values from the statistics collector
- configuration has two parts
  - SQL query and specifications how to interpret the values
  - database connection plus queries assigned to it
The collectd naming scheme

- each data-set uses a unique identifier
  - hostname
  - plugin name
  - plugin instance (optional)
  - type
  - type instance (optional)
- hostname/plugin[-instance]/type[-instance]
- the type specifies how collectd is supposed to handle the data-set (cf. RRDtool’s data-source types)
- any type needs to be pre-defined (types.db(5))

- example: server1.ex.com/cpu-0/cpu-idle
collectd.conf

<Plugin postgresql>
  <Query disk_usage>
    Statement "SELECT pg_database_size($1) AS size;"
    Param database
    
    <Result>
      Type pg_db_size
      ValuesFrom "size"
    </Result>
  </Query>
</Plugin>
The PostgreSQL plugin: DB Definition

```
<Plugin postgresql>
  <Database mail>
    Host "db.ex.com"
    User "user"
    Password "secret"
    Query disk_usage
    Query disk_io
  </Database>
</Plugin>
```
Thanks for your attention!

Any questions?
Contact:
Sebastian „tokkee“ Harl
teamix GmbH, Nürnberg
<sh@teamix.net>

<collectd@verplant.org> — irc.freenode.net/#collectd — http://identi.ca/collectd