

Wisse, was deine Elefanten machen

PostgreSQL Performance-Analyse mit collectd

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

teamix GmbH / collectd core team



PGConf.DE 2011
11. November 2011

- gegründet 2001
- Ursprünge: Open-Source und Netzwerke
 - Debian
 - Nagios
 - Schulungen
 - u.v.m.
- Heute auch:
 - NetApp
 - VMWare
 - Riverbed (WAN-Beschleunigung)
 - Juniper
 - N-IX (Nürnberger Internet-eXchange)

Was ist collectd?

Überblick

Wichtige Eigenschaften

Plugin-Überblick

PostgreSQL Prozesse

Statistiken von PostgreSQL abfragen

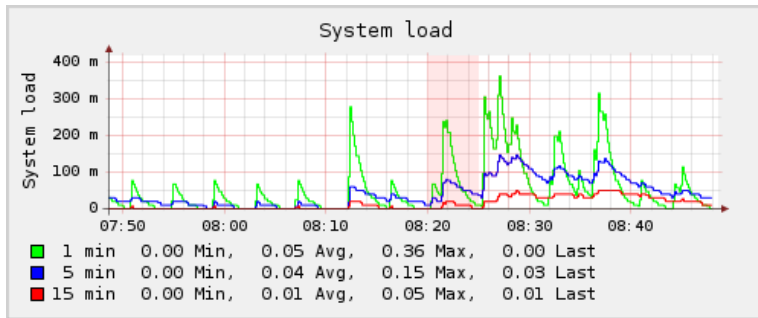
Performance-Daten in RDBMS speichern

Über den Tellerrand

- **collectd** sammelt Leistungsdaten von Rechnern
- Leistungsdaten sind zum Beispiel:
 - CPU-Auslastung
 - Speichernutzung
 - Netzwerkverkehr
- Daten werden erhoben, verarbeitet und gespeichert
- Häufig: Darstellung als Graphen
- → Performance-Analyse, Kapazitätsplanung
- Nicht verwechseln mit *Monitoring!*
- Homepage: <http://collectd.org/>

- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, ...)
- Skalierbar (OpenWrt, ..., Cluster / Cloud)
- Effizient (Default-Auflösung: 10 Sekunden)
- Modular (über 100 Plugins in Version 5.0)

- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, ...)
- Skalierbar (OpenWrt, ..., Cluster / Cloud)
- **Effizient** (Default-Auflösung: 10 Sekunden)
- Modular (über 100 Plugins in Version 5.0)



- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, ...)
- Skalierbar (OpenWrt, ..., Cluster / Cloud)
- Effizient (Default-Auflösung: 10 Sekunden)
- Modular (über 100 Plugins in Version 5.0)

- Daemon
- Freie Software (größtenteils GPLv2)
- Portierbar (Linux, *BSD, Solaris, ...)
- Skalierbar (OpenWrt, ..., Cluster / Cloud)
- Effizient (Default-Auflösung: 10 Sekunden)
- **Modular** (über 100 Plugins in Version 5.0)

Verfügbare Plugins (Auswahl)

apache	amqp	apcups	ascent	battery
bind	conntrack	contextswitch	cpu	cpufreq
csv	curl	curl_json	dbi	df
disk	dns	email	entropy	exec
filecount	fscache	GenericJMX	gmond	hddtemp
interface	ipmi	iptables	ipvs	irq
java	libvirt	load	logfile	madwifi
match_regex	mbmon	memcachec	memcached	memory
Monitorus	multimeter	mysql	netapp	netlink
network	nfs	nginx	notify_email	ntpd
nut	olsrd	onewire	openvpn	OpenVZ
oracle	perl	ping	postgresql	powerdns
processes	protocols	python	routeros	rrdcached
rrdtool	sensors	serial	snmp	swap
syslog	table	tail	tape	target_scale
tcpconns	teamspeak2	ted	thermal	tokyotyrant
unixsock	uptime	users	uuid	vmem
vserver	wireless	write_http	xmms	zfs_arc

- Aktuelle Version ist 5.0 (Release: März 2011)
- Pakete für diverse Distributionen vorhanden (Debian, RedHat, FreeBSD, OpenWrt, OpenSolaris [WIP], ...)
- Major-Version 4.x ist an einigen Stellen inkompatibel¹
→ v5upgrade Target
- Geschrieben in C
- Versionsverwaltung mit Git
→ `git://git.verplant.org/collectd.git`

¹http://collectd.org/wiki/index.php/V4_to_v5_migration_guide

- Daemon läuft auf jedem Client (Ausnahme: SNMP o.ä.)
- üblicherweise: ein oder mehrere zentrale Server, die Werte von Clients empfangen (Push-Modell)
- First steps: `install; select plugins; start daemon; enjoy ;-)`

C4

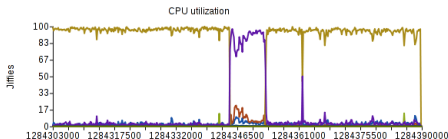
collection 4

- All instances
- All graphs
- Host "[redacted]"

Graph "CPU utilization"

Instance "[redacted] / 0"

Instance: "[redacted] / cpu - 0 / cpu - all"

 Search

Hour ▾

JSON (gRaphaël)

RRDtool

Go

collection 4.0.0

- Grundidee: Daten über, z. B., JSON zur Verfügung stellen
- verschiedene Frontends davor möglich
- effiziente Handhabung von vielen Datensätzen durch Caching
- flexible Konfiguration von Graphen

Was ist collectd?

Plugin-Überblick

CPU, Speicher, Netzwerk-I/O

Netzwerk-Plugin

RRDtool-Plugin (Überblick)

Generische Plugins (Überblick)

Eigene Erweiterungen (Überblick)

PostgreSQL Prozesse

Statistiken von PostgreSQL abfragen

Performance-Daten in RDBMS speichern

- Spezielle Lese-Plugins
 - CPU, Speicher, Netzwerk-Schnittstellen
- Schreib- bzw. IO-Plugins
 - Netzwerk-Plugin
 - RRDtool, RRDCacheD
- Generische Plugins
 - SNMP
 - tail
 - PostgreSQL

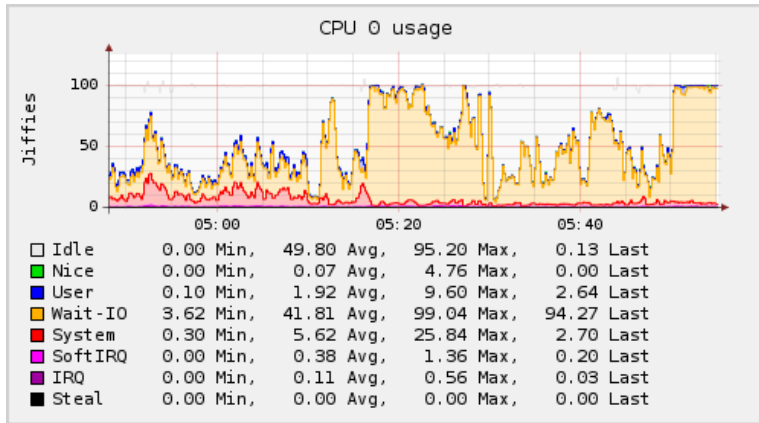
Synopsis

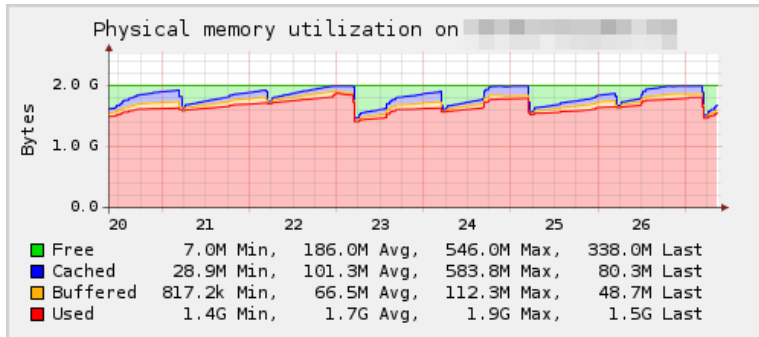
```
LoadPlugin "cpu"  
LoadPlugin "memory"  
LoadPlugin "interface"
```

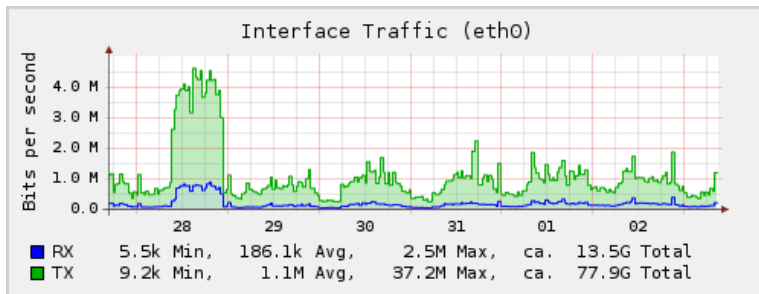

Synopsis

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

```
<Plugin interface>  
  Interface lo  
  Interface sit0  
  IgnoreSelected true  
</Plugin>
```





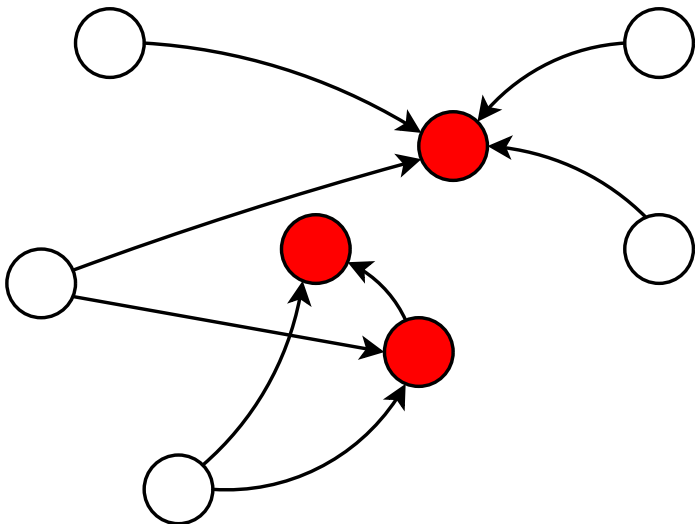


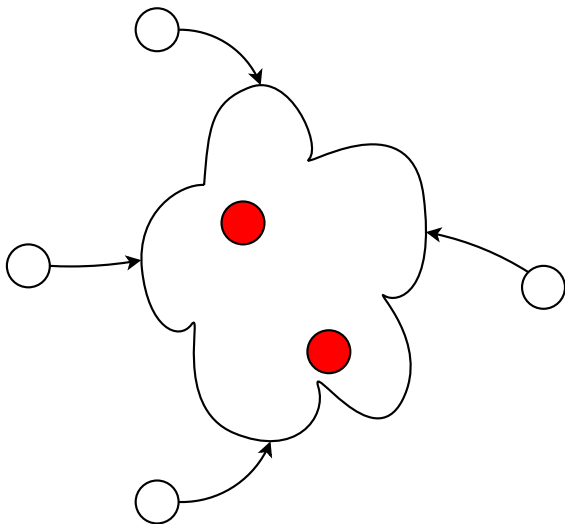
Betriebsarten

- Daten versenden („Client“)
- Daten empfangen („Server“)
- Weiterleiten („Proxy“)
- Unicast („Punkt-zu-Punkt“)
- Multicast („Punkt-zu-Gruppe“)
- IPv4 und IPv6

Ein Daemon für alles

Rolle des Daemon hängt von der Konfiguration ab.





Synopsis: Client

```
LoadPlugin "network"
```

```
<Plugin "network">
```

```
  Server "collectd0.musterfirma.de"
```

```
  Server "collectd1.musterfirma.de"
```

```
  Server "ff18::efc0:4a42"
```

```
</Plugin>
```


Synopsis: Server

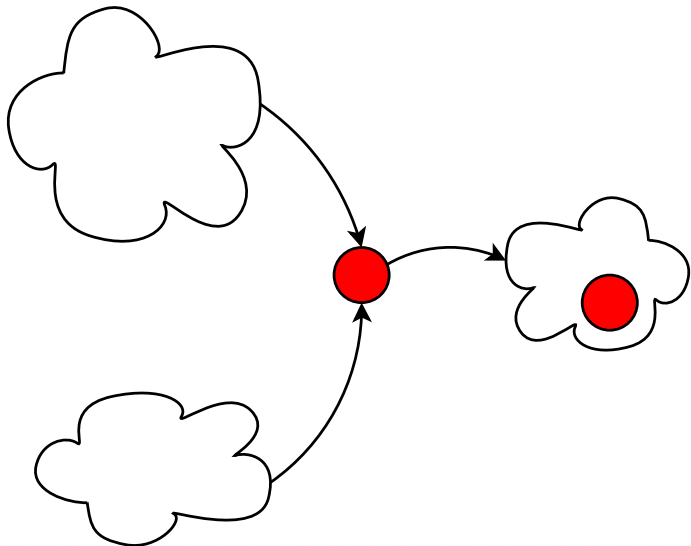
```
LoadPlugin "network"
```

```
<Plugin "network">
```

```
  Listen "collectd0.musterfirma.de"
```

```
  Listen "ff18::efc0:4a42"
```

```
</Plugin>
```



Synopsis: Proxy

```
LoadPlugin "network"
```

```
<Plugin "network">
```

```
  Listen "collectgw.extern.musterfirma.de"
```

```
  Server "collectd1.intern.musterfirma.de"
```

```
  Forward true
```

```
</Plugin>
```

Authentifizierung / Verschlüsselung

- (seit Version 4.7.0)
- Authentifizierung via HMAC-SHA-256
- Verschlüsselung mit AES-256 (OFB)

Authentifizierung / Verschlüsselung

		Client		
		Nichts	Sign	Encrypt
Server	Nichts	akzeptiert	akzeptiert	nicht möglich
	AuthFile	akzeptiert	akzeptiert	akzeptiert
	Sign	nicht akzeptiert	akzeptiert	akzeptiert
	Encrypt	nicht akzeptiert	nicht akzeptiert	akzeptiert

- Schreibt Daten effizient in RRD-Dateien → Caching
- Funktionalität nun in RRDtool als RRD Caching Daemon verfügbar

Synopsis

```
LoadPlugin "rrdtool"
```

```
<Plugin "rrdtool">
```

```
  DataDir "/var/lib/collectd/rrd"
```

```
</Plugin>
```

Konfiguration

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

- FLUSH ermöglicht dennoch die graphische Darstellung von aktuellen Daten

- Idee: Generische Ansätze, statt Speziallösungen
- → Benutzerkonfiguration bestimmt das Verhalten
- ⇒ Unterstützung für neue Geräte braucht i.d.R. keine neue Version von **collectd**
- Beispiele: SNMP, tail, curl, DBI, PostgreSQL

- **collectd** API: C, Perl, Python, Java
- Externe Programme mittels unixsock- oder exec-Plugin

Was ist collectd?

Plugin-Überblick

PostgreSQL Prozesse

Statistiken von PostgreSQL abfragen

Performance-Daten in RDBMS speichern

Über den Tellerrand

```
% ps ax | grep postgres
20177 ?  S    0:05 /usr/lib/postgresql/8.3/bin/postgres
        -D /var/lib/postgresql/8.3/main
        -c config_file=/etc/postgresql/8.3/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
```

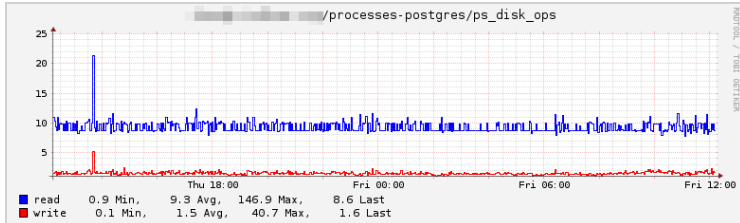
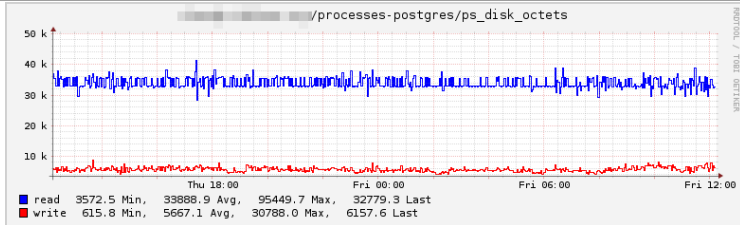
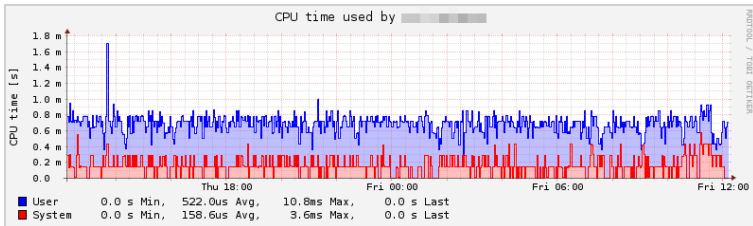
- Behandlung einer Client-Verbindung:
postgres: user database host activity

- Das processes Plugin kann div. Informationen über einzelne Prozesse (oder Gruppen)
 - RSS- und VM-Größe
 - User- und System-Zeit
 - Anzahl Page-Faults
 - Schätzwerte zum I/O
- Auswahl entweder an Hand des Prozessnamens oder Regex auf Kommandozeile

collectd.conf

```
<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"
  # Datenbankverbindungen durch Benutzer 'user'
  ProcessMatch pg_user_mail "postgres:.user"
  # Datenbankverbindungen auf DB 'mail'
  ProcessMatch pg_db_mail "postgres:.[A-Za-z0-9]+.mail"
</Plugin>
```

(vor 5.0.1 durften Regexen kein Whitespace enthalten)



Was ist collectd?

Plugin-Überblick

PostgreSQL Prozesse

Statistiken von PostgreSQL abfragen

Der PostgreSQL Statistik-Sammler

Das postgresql Plugin von collectd

Performance-Daten in RDBMS speichern

Über den Tellerrand

postgresql.conf

```
# Aktuell ausgeführtes Kommando
track_activities = on
# Tabellen- und Index-Zugriff
track_counts = on
# Benutzerdefinierte Funktionen
track_functions = none # none, pl, all
```

Ablegen der Statistiken, z.B. auf Flash-Speicher:

```
stats_temp_directory = '/mnt/flash/pg_stat_tmp'
```


- Server Prozesse übermitteln Statistiken vor idle
- Reports werden vom Sammler minimal alle `PGSTAT_STAT_INTERVAL` Millisekunden erstellt
- Während einer Transaktion wird ein Snapshot des Reports verwendet
→ siehe `pg_stat_clear_snapshot()`

- Einige vordefinierte Views
 - `pg_stat_bgwriter`
 - `pg_stat_database`
 - `pg_stat_all_indexes`
 - `pg_statio_all_tables`
 - u.v.m. (Tabelle 27.1 in Doku)
- alternativ: Funktionen zur Abfrage der einzelnen Werte

- Generisches Plugin, welches beliebige (numerische) Werte über SQL abfragen kann
- Standardmäßig werden diverse Werte vom Statistik-Sammler abgefragt
- Konfiguration besteht aus zwei Teilen:
 - SQL-Queries mit Spezifikation zur Interpretation der Werte
 - Datenbankverbindungen

- jeder Datensatz hat einen eindeutigen Identifier
 - Hostname
 - Plugin Name
 - Plugin Instanz (optional)
 - Typ
 - Typ Instanz (optional)
- `hostname/plugin[-instanz]/typ[-instanz]`
- Der Typ definiert, wie ein Datum interpretiert werden soll (angelehnt an RRDtools Datasource-Typen)
- Typen müssen vordefiniert sein (`types.db(5)`)

- Beispiel: `server1.bsp.de/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>
```

collectd.conf

```
<Plugin postgresql>
  <Database mail>
    Host "db.bsp.de"
    User "user"
    Password "geheim"
    Query disk_usage
    Query disk_io
  </Database>
</Plugin>
```

Was ist collectd?

Plugin-Überblick

PostgreSQL Prozesse

Statistiken von PostgreSQL abfragen

Performance-Daten in RDBMS speichern

Über den Tellerrand

- Traditionell: Speichern in RRDtool (konstanter Speicherverbrauch, Konsolidierung, schlechte Skalierbarkeit)
- Speicherung in RDBMS ermöglicht komplexe Auswertung und bessere Skalierung
- Probleme:
 - Struktur der Datenbank (Komplexität vs. unnötige Redundanz vs. Performance)
 - Behandlung von alten Daten (Partitionierung?)

Was ist collectd?

Plugin-Überblick

PostgreSQL Prozesse

Statistiken von PostgreSQL abfragen

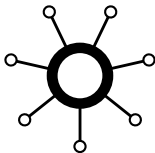
Performance-Daten in RDBMS speichern

Über den Tellerrand

- `collectd-nagios`
Fragt Daten via `unixsock`-Plugin ab und erzeugt Nagios-kompatible Ausgabe
- `exec-nagios.px`
Perl-Skript welches *Nagios*-Plugins ausführt (→ *exec-Plugin*)
- `exec-munin.px`
Perl-Skript welches *Munin*-Plugins ausführt (→ *exec-Plugin*)
- `gmond-Plugin`
Empfängt und verarbeitet *Ganglia* Multicast-Pakete

Vielen Dank für die Aufmerksamkeit!

Gibt es Fragen?



<https://www.postgresql.eu/events/feedback/pgconfde2011/>

Kontakt:

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

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

Artikel zum Thema **collectd**:

[http://linuxtechnicalreview.de/Vorschau/\(show\)/Themen/
Monitoring/Performance-Analyse-mit-Collectd](http://linuxtechnicalreview.de/Vorschau/(show)/Themen/Monitoring/Performance-Analyse-mit-Collectd)

Optional

Allgemeines

- Fragt Netzwerk-Zubehör via SNMP ab
- *Generisch*: Nicht für ein gestimmtes Gerät geschrieben
- Mehrere Geräte werden parallel abgefragt

Konfiguration

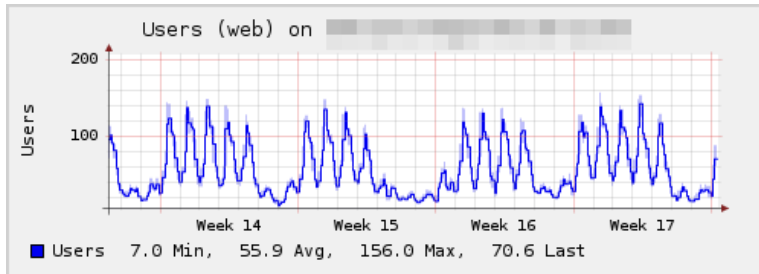
- „Data“-Blöcke
- „Host“-Blöcke

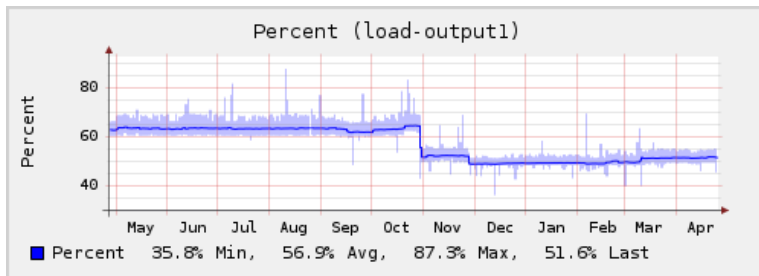
Synopsis: Data-Block

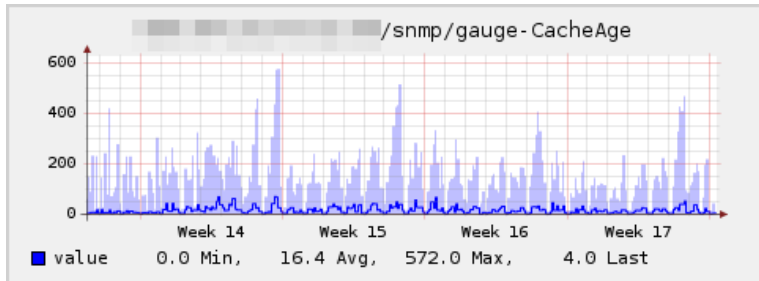
```
<Plugin "snmp">
  <Data "ifmib_if_octets64">
    Type "if_octets"
    Table true
    Instance "IF-MIB::ifName"
    Values "IF-MIB::ifHCInOctets" \
          "IF-MIB::ifHCOutOctets"
  </Data>
</Plugin>
```

Synopsis: Host-Block

```
<Plugin "snmp">  
  <Host "switch0.intern.musterfirma.de">  
    Address "10.0.42.2"  
    Version 1  
    Community "public"  
    Collect "ifmib_if_octets64"  
    Interval 60  
  </Host>  
</Plugin>
```



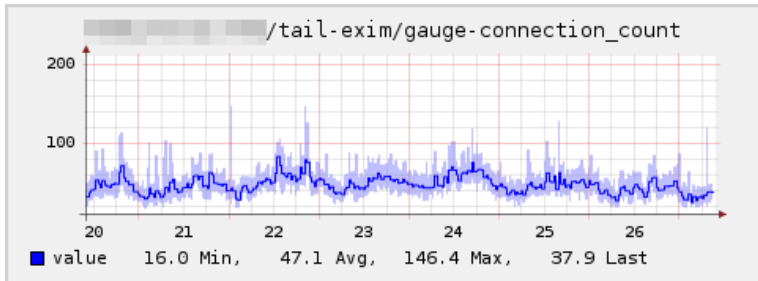


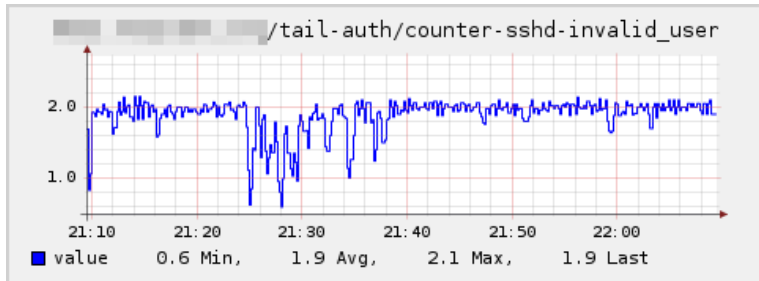
Allgemeines

- Verfolgt Log-Dateien
- Extrahiert Werte oder zählt Ereignisse
- Selektion der Zeilen / Werte mit regulären Ausdrücken
- Verwendbar für MTAs, Web-Server, ...

Konfiguration

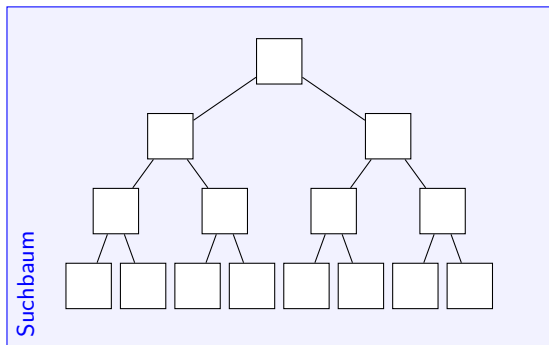
```
<Plugin "tail">
  <File "/var/log/exim4/mainlog">
    Instance "exim"
    <Match>
      Regex "S=([1-9][0-9]*)"
      DStype "CounterAdd"
      Type "ipt_bytes"
      Instance "total"
    </Match>
  </File>
</Plugin>
```



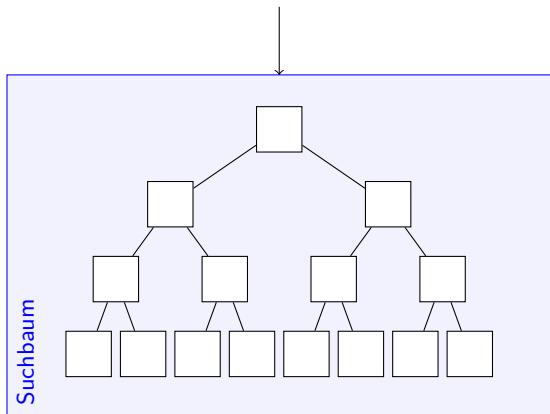


Allgemeines

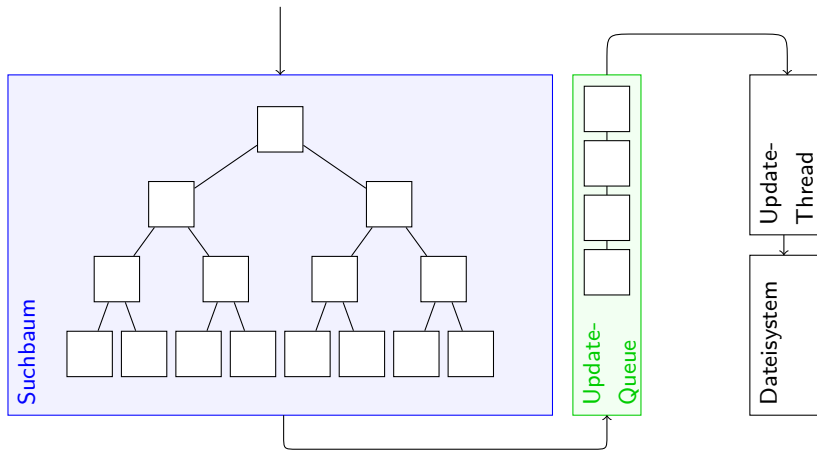
- Update-Prinzip des RRDtool-Plugins
- Eigenständiger Daemon
- Integration in RRDtool 1.4
- Weitere Funktionen, z. B. Journaling
- Vorteil: Neustart von **collectd** ohne Cache-Verlust



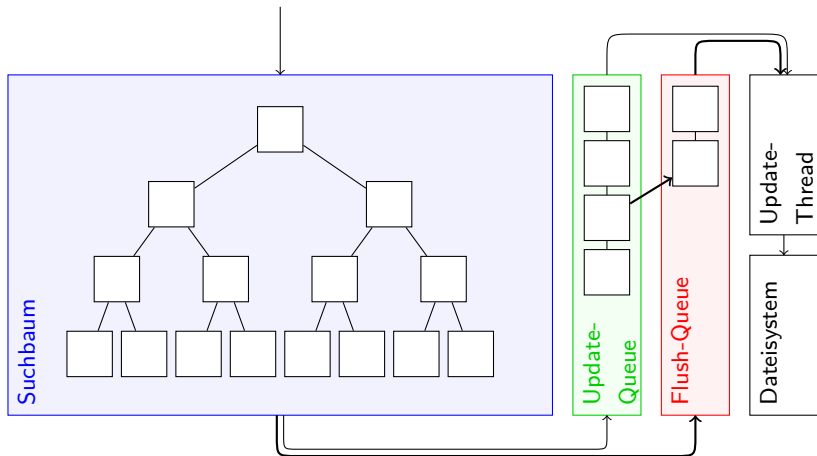
Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster

Allgemeines

- Integriert einen Perl-Interpreter
(vergleichbar zu Apaches `mod_perl`)
- Instanziierung und Syntax-Analyse nur einmal
- Exportiert die API
(→ nicht nur Lese-Plugins möglich)

```
package Collectd::Plugin::Magic;
use Collectd qw( :all );
sub magic_read
{
    my $vl = { plugin => 'magic',
               values => [Magic->getCurrentLevel ()] };
    plugin_dispatch_values ('magic_level', $vl);
}
plugin_register (TYPE_READ, 'magic', 'magic_read');
```

Allgemeines

- Öffnet einen UNIX-Domain-Socket
- Kennt mehrere Befehle
(z. B. PUTVAL, FLUSH, LISTVAL)
- Interaktion mit externen Programmen möglich
- `collectdctl` (ab Version 5.0, eta: dieses Jahr ;-))
- `cussh.pl`: „*collectd UNIX socket shell*“

```
-> | PUTVAL "testhost/magic/magic_level" \  
    interval=10 1179574444:42  
<- | 0 Success
```


Allgemeines

- Führt Programme aus
- Liest von deren Standard-Ausgabe
- Können über längere Perioden laufen
(vgl. `init`)

```
#!/bin/sh
INTVL=${COLLECTD_INTERVAL:-10}
CHOST="${COLLECTD_HOSTNAME:-localhost}"
IDENT="$CHOST/magic/magic_level"
while sleep $INTVL
do
    VALUE='magic --level'
    echo "PUTVAL \"$IDENT\" interval=$INTVL N:$VALUE"
done
```

Allgemeines

- Integriert eine „Java Virtual Maschine“ (JVM)
- Exportiert die API
(→ nicht nur Lese-Plugins möglich)
- Prinzipielle Ähnlichkeit zum Perl-Plugin

```
import org.collectd.api.Collectd;
import org.collectd.api.CollectdReadInterface;
public class MagicPlugin
    implements CollectdReadInterface
{
    public int read ();    /* Callback-Funktion */
    public MagicPlugin (); /* Konstruktor */
}
```

```
public int read ()
{
    ValueList vl = new ValueList ();
    vl.setHost ("testhost");
    vl.setPlugin ("magic");
    vl.setType ("magic_level");
    vl.addValue (Magic.getCurrentLevel ());
    return (Collectd.dispatchValues (vl));
}
```

```
public MagicPlugin ()
{
    /* Callback-Funktion anmelden */
    Collectd.registerRead ("MagicPlugin", this);
}
```

- **collectd** API nutzen
C, Perl, Python und Java möglich

- **collectd** API nutzen
C, Perl, Python und Java möglich
- Externe Programme erweitern
unixsock-Plugin ermöglicht Kommunikation

- **collectd** API nutzen
C, Perl, Python und Java möglich
- Externe Programme erweitern
unixsock-Plugin ermöglicht Kommunikation
- Eigenes Programm / Skript schreiben
→ exec-Plugin

- `snmp-probe-host.px`
Erzeugt semi-automatisch `<Host />`-Blöcke für das SNMP-Plugin
- `jcollected`
Java-Implementierung des Netzwerk-Protokolls (\rightarrow *JMX*)
- `kcollected`
KDE-Programm zur Near-Realtime-Anzeige von Graphen
- Perl, Ruby, Python Module und C-Bibliothek für die Kommunikation mit dem `unixsock`-Plugin verfügbar