

collectd in der großen weiten Welt

Anbindung des Datensammlers an Monitoringlösungen

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

teamix GmbH / collectd core team



OSMC 2011
29. November 2011



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

- collectd ist auf das effiziente Sammeln und Speichern von Performance-Daten ausgelegt
- übliche Monitoringlösungen können davon profitieren und sich auf das eigentliche Monitoring konzentrieren

Überblick über collectd

Überblick

Wichtige Eigenschaften

Plugin-Überblick

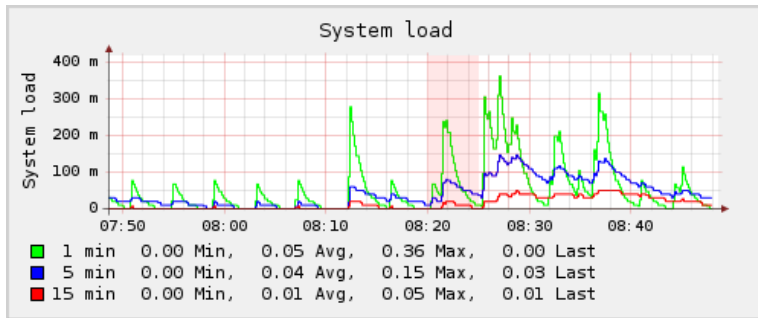
collectd anbinden und erweitern

Weitere Möglichkeiten von collectd

- **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)
- Mächtige Netzwerkunterstützung
- 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)
- Mächtige Netzwerkunterstützung
- **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)
- Mächtige Netzwerkunterstützung
- 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)
- Mächtige Netzwerkunterstützung
- 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	ntp
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

- 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 "..."

Graph "CPU utilization"

Instance "..." / 0"

Instance: "... / 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

Überblick über collectd

Plugin-Überblick

CPU, Speicher, Netzwerk-I/O

Netzwerk-Plugin

RRDtool-Plugin

Generische Plugins (Überblick)

Eigene Erweiterungen (Überblick)

collectd anbinden und erweitern

Weitere Möglichkeiten von collectd

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

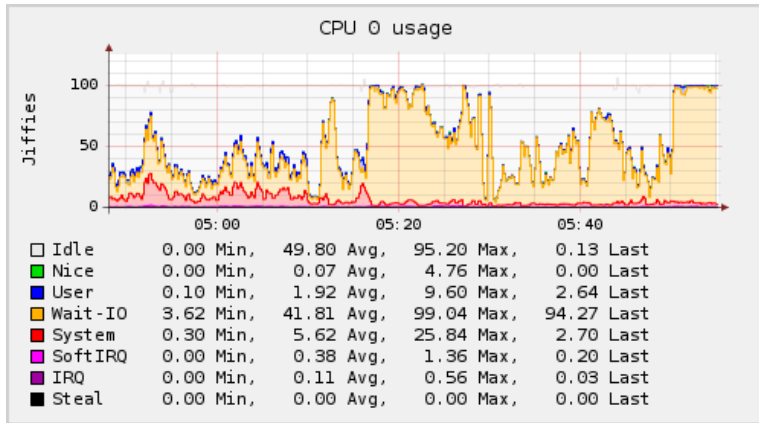
Synopsis

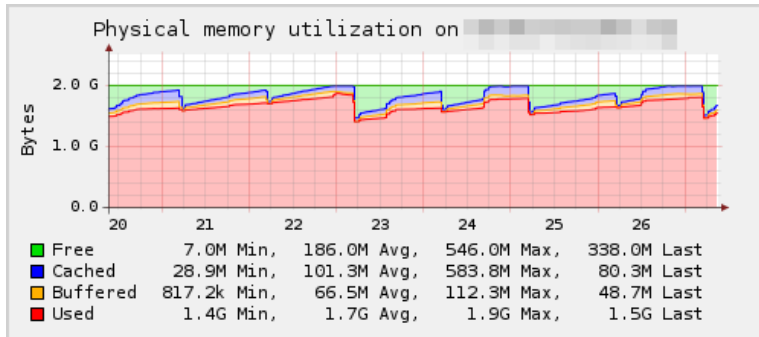
```
LoadPlugin "cpu"  
LoadPlugin "memory"  
LoadPlugin "interface"
```

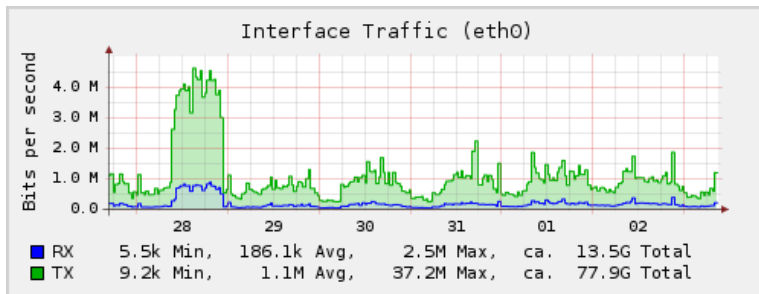

Synopsis

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

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





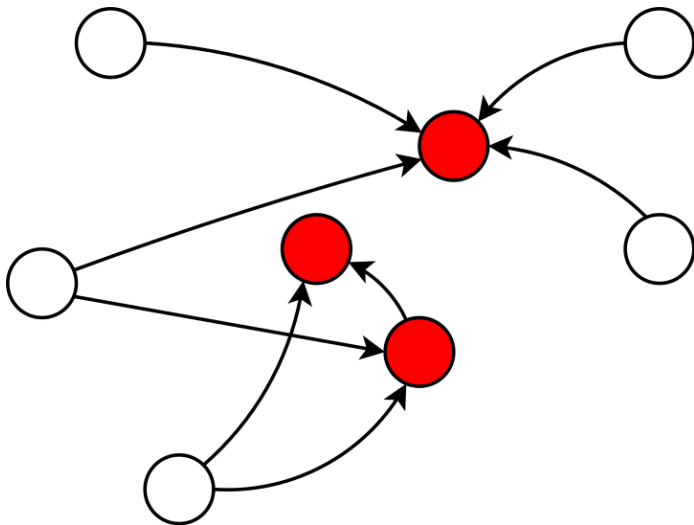


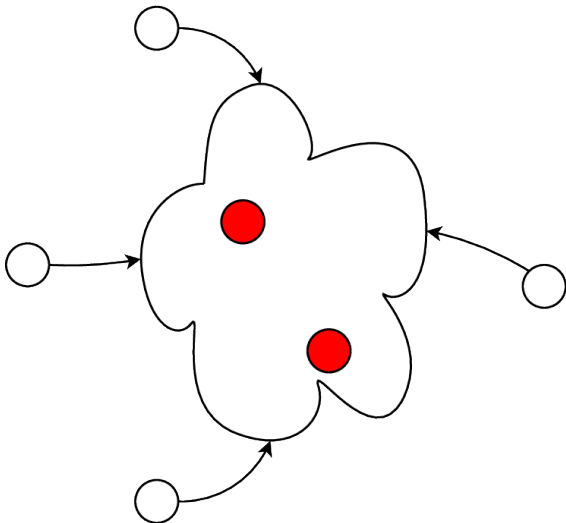
Betriebsarten

- 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

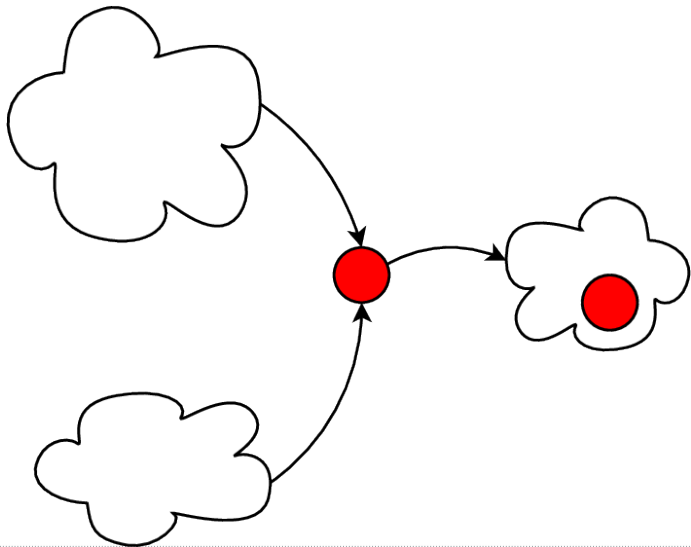
```
LoadPlugin "network"
```

```
<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

Überblick über collectd

Plugin-Überblick

collectd anbinden und erweitern

- UNIXSOCK Schnittstelle

- collectds Namensschema

- collectd-nagios

- collectd-nagios: Zukunftsmusik

- Eigene Erweiterungen

Weitere Möglichkeiten von collectd

UNIXSOCK Plugin

- öffnet einen UNIX-Domain-Socket zur Interaktion mit dem Daemon
- Text- und Zeilen-basiertes Protokoll (z. B. PUTVAL, FLUSH, LISTVAL, GETVAL)
- ermöglicht, u.a., die Abfrage von Werten
- collectd-nagios
- collectdctl (ab Version 5.0)
- `cussh.pl`: „*collectd UNIX socket shell*“

```
-> | GETVAL "FQDN/load/load"  
<- | 3 Values found  
<- | shortterm=4.000000e-02  
<- | midterm=6.000000e-02  
<- | longterm=7.000000e-02
```

- jeder Datensatz hat einen eindeutigen Identifier
 - Hostname
 - Plugin Name
 - Plugin Instanz (optional)
 - Typ
 - Typ Instanz (optional)
- `hostname/plugin[-instanz]/typ[-instanz]`

- Beispiel: `localhost/cpu-0/cpu-idle`

- `cd /var/lib/collectd;`
`ls **/*.rrd`
- `collectdctl listval`

```
cpu-0/cpu-idle
cpu-0/cpu-interrupt
cpu-0/cpu-nice
cpu-0/cpu-softirq
cpu-0/cpu-steal
cpu-0/cpu-system
cpu-0/cpu-user
cpu-0/cpu-wait
interface-eth0/if_errors
interface-eth0/if_octets
interface-eth0/if_packets
load/load
memory/memory-buffered
memory/memory-cached
memory/memory-free
memory/memory-used
users/users
...
```

- Nagios-/Icinga-/Shinken-Plugin
- verbindet sich zum UNIX Socket von collectd
- fragt einen (WIP: mehrere) Datensatz ab
- überprüft übergebene Schwellwerte
- kehrt mit Nagios-kompatiblen Rückgabewert zurück

```
% collectd-nagios -h
```

```
Usage: collectd-nagios <-s socket> <-n value_spec> <-H hostname> [options]
```

Valid options are:

- s <socket> Path to collectd's UNIX-socket.
- n <v_spec> Value specification to get from collectd.
Format: 'plugin-instance/type-instance'
- d <ds> Select the DS to examine. May be repeated
to examine multiple
DSes. By default all DSes are used.
- g <consol> Method to use to consolidate several DSes.
See below for a list of valid arguments.
- H <host> Hostname to query the values for.
- c <range> Critical range
- w <range> Warning range


```
# adduser nagios collectd
# su - nagios

% collectd-nagios -s /var/run/collectd-unixsock \
                  -H FQDN -n load/load
OKAY: 0 critical, 0 warning, 3 okay |
  shortterm=0.120000;;;
  midterm=0.130000;;;
  longterm=0.180000;;;
```

(Umbrüche nur auf den Folien)

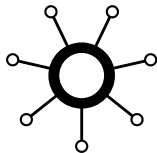
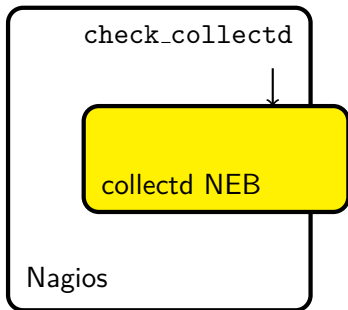
```
% collectd-nagios -s /var/run/collectd-unixsock \  
                -H FQDN -n df/df-root \  
                -d free -d used -g percentage  
OKAY: 31.289281 percent |  
    free=46196220000.000000;;;;  
    used=101446100000.000000;;;;
```

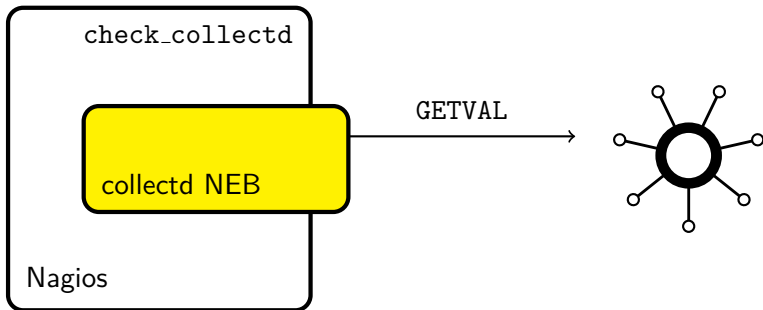
(Umbrüche nur auf den Folien)

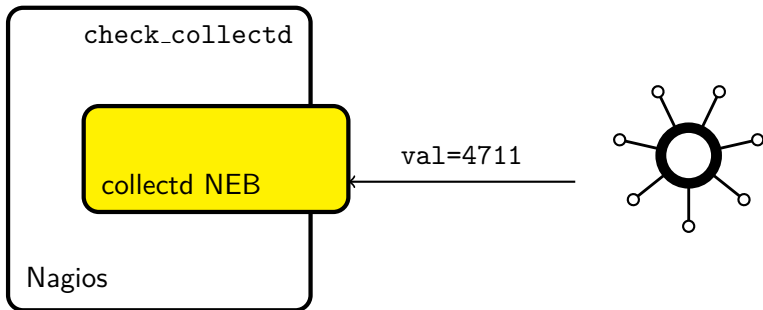
```
define command {  
    command_name check_collectd  
    command_line /.../collectd-nagios  
                -s /.../collectd-unixsock  
                -H $HOSTNAME$ -n $ARG1$  
                -d $ARG2$  
                -w $ARG3$ -c $ARG4$  
}
```

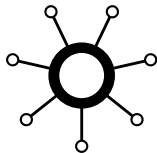
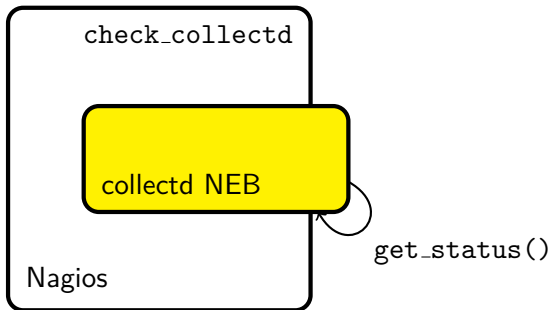
```
define command {
    command_name check_collectd_percentage
    command_line /.../collectd-nagios
                -s /.../collectd-unixsock
                -H $HOSTNAME$ -n $ARG1$
                -g percentage
                -d $ARG2$ -d $ARG3$
                -w $ARG4$ -c $ARG5$
}
```

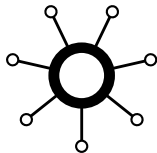
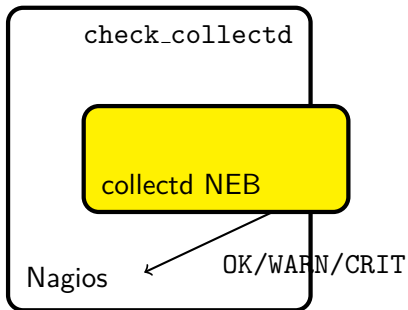
```
check_collectd_percentage!df/df-root!free!used!20:!10:  
check_collectd!load/load!midterm!5!10  
check_collectd!users/users!users!50!70  
check_collectd!postgresql-webapp/users-loggedin!users!...  
check_collectd!tail-postfix/gauge-connections_in!value!...
```



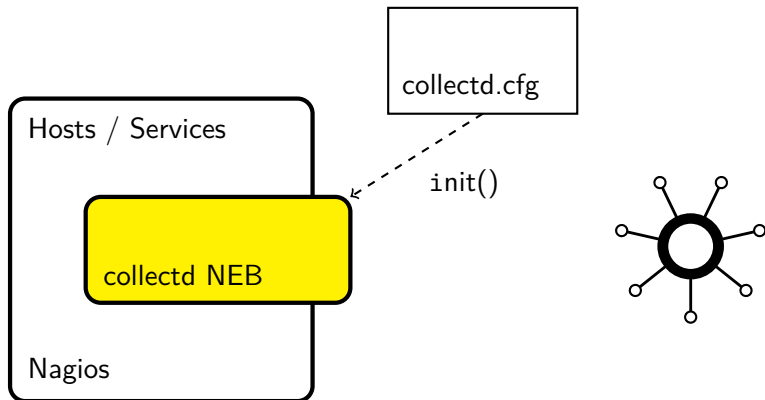




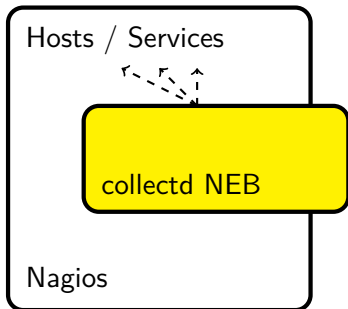
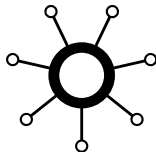


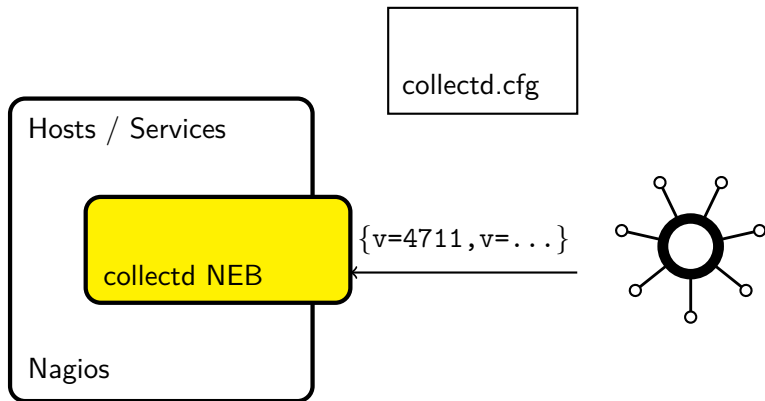


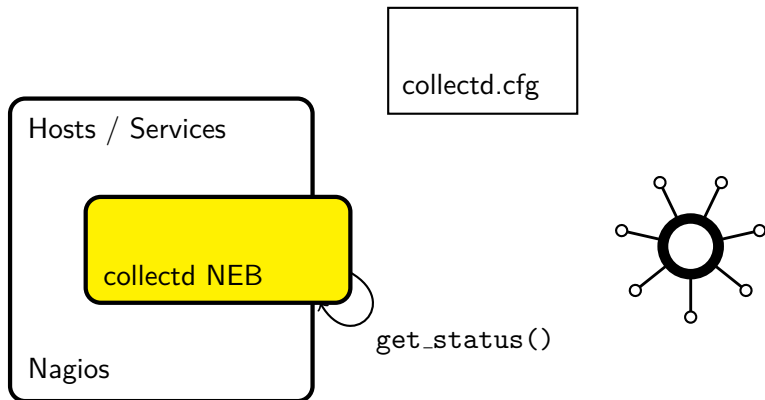
- „aktiver“ Event Broker: „embedded collectd-nagios“
- Erkennen von collectd Checks (konf. command name, custom variable, o.ä.)
- Abfrage von collectd über UNIX Socket
- Auswerten der Daten
- active check result an Nagios übergeben



collectd.cfg







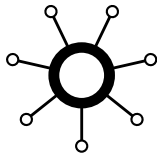
collectd.cfg

Hosts / Services

collectd NEB

Nagios

OK/WARN/CRIT

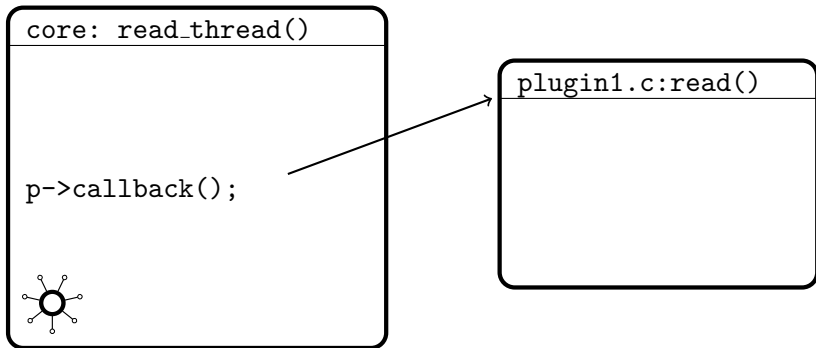


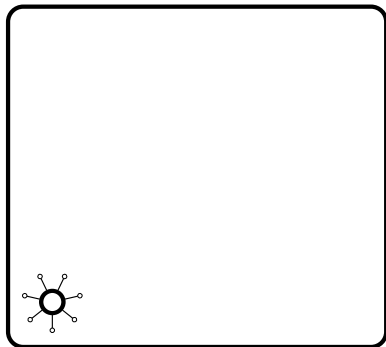
- „passiver“ Event Broker: collectd Netzwerk Client
- Konfiguration von gewünschten Werten und Schwellwerten in eigener Konfigurationsdatei (`collectd.cfg`)
- Erzeugung von dazugehörigen Hosts und Services
- Server für collectds Netzwerk-Plugin
- Auswerten der gewünschten Werte
- active/passive check results an Nagios übergeben

```
core: read_thread()
```

```
p = get_next_plugin();  
sleep_until_due(p);
```







```
plugin1.c:read()  
  
...  
dispatch_values(v);
```

```
core: dispatch_values()
```

```
...  
plugin_write(v);
```



```
plugin1.c:read()
```

```
...  
dispatch_values(v);
```



```
core: dispatch_values()
```

```
...
```

```
plugin_write(v);
```



```
plugin1_thread()
```

```
plugin2.c:write()
```

```
...
```

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');
```

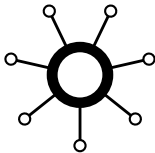

Allgemeines

- 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
```

Vielen Dank für die Aufmerksamkeit!

Gibt es Fragen?



Kontakt:

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

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

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)

Überblick über collectd

Plugin-Überblick

collectd anbinden und erweitern

Weitere Möglichkeiten von collectd

- SNMP-Plugin

- tail-Plugin

- processes-Plugin am Beispiel PostgreSQL

- PostgreSQL-Plugin

- RRDCacheD-Plugin

- Eigene Erweiterungen

- Über den Tellerrand

Allgemeines

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

Konfiguration

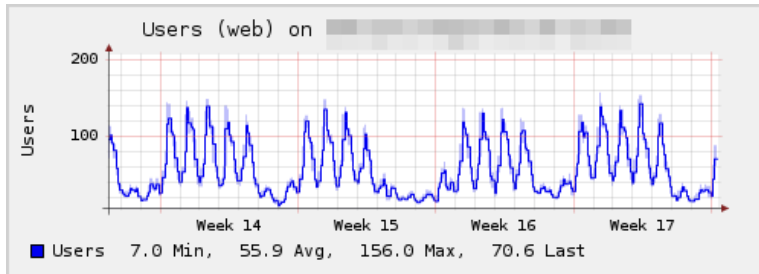
- „Data“-Blöcke
- „Host“-Blöcke

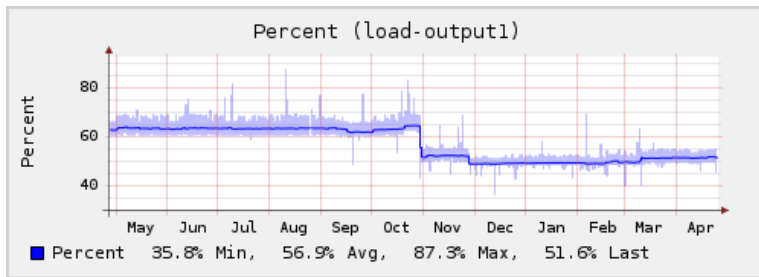
Synopsis: Data-Block

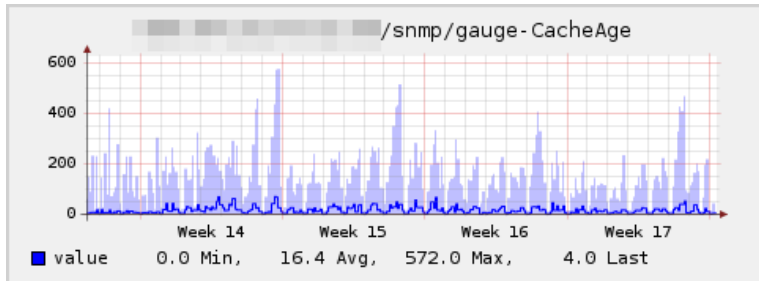
```
<Plugin "snmp">
  <Data "ifmib_if_octets64">
    Type "if_octets"
    Table true
    Instance "IF-MIB::ifName"
    Values "IF-MIB::ifHCInOctets" \
          "IF-MIB::ifHCOutOctets"
  </Data>
</Plugin>
```

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>
```



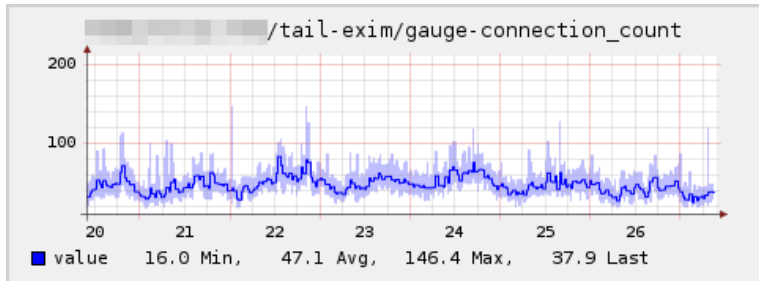


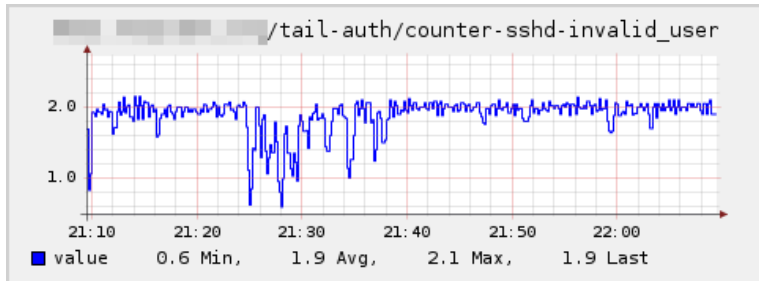
Allgemeines

- 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>
```





```
% 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
```

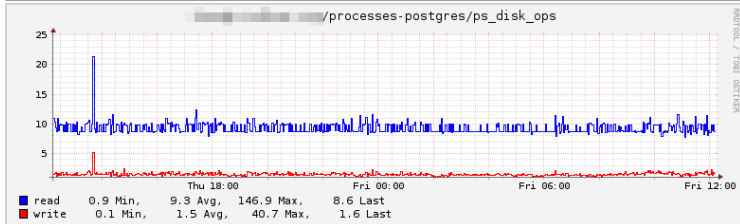
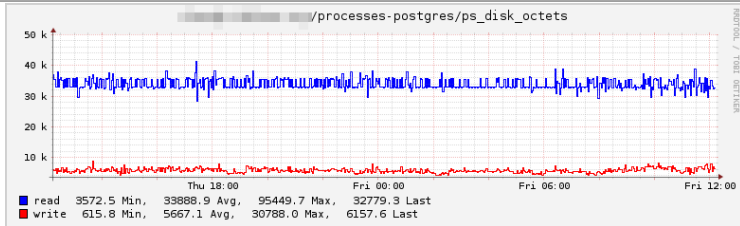
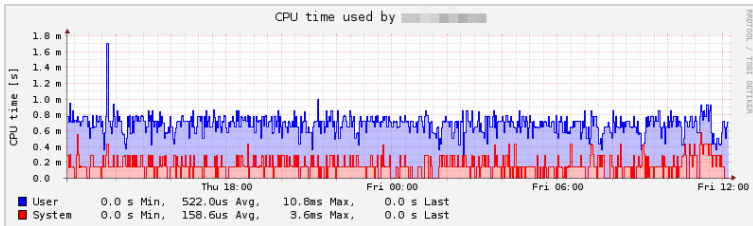
- Behandlung einer Client-Verbindung:
postgres: user database host activity

- 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)



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

collectd.conf

```
<Plugin postgresql>
  <Query disk_usage>
    Statement "SELECT pg_database_size($1) AS size;"
    Param database

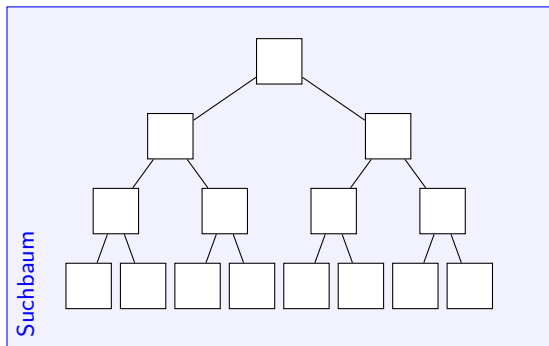
    <Result>
      Type pg_db_size
      ValuesFrom "size"
    </Result>
  </Query>
</Plugin>
```


collectd.conf

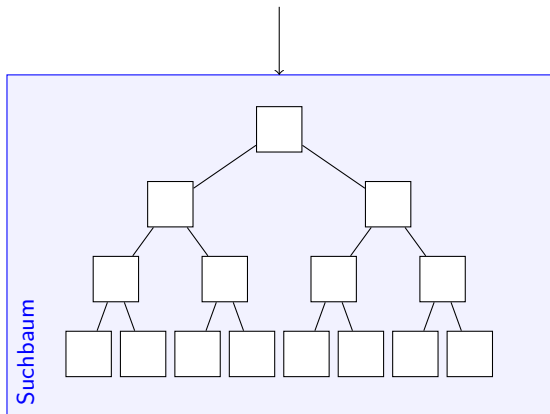
```
<Plugin postgresql>
  <Database mail>
    Host "db.bsp.de"
    User "user"
    Password "geheim"
    Query disk_usage
    Query disk_io
  </Database>
</Plugin>
```

Allgemeines

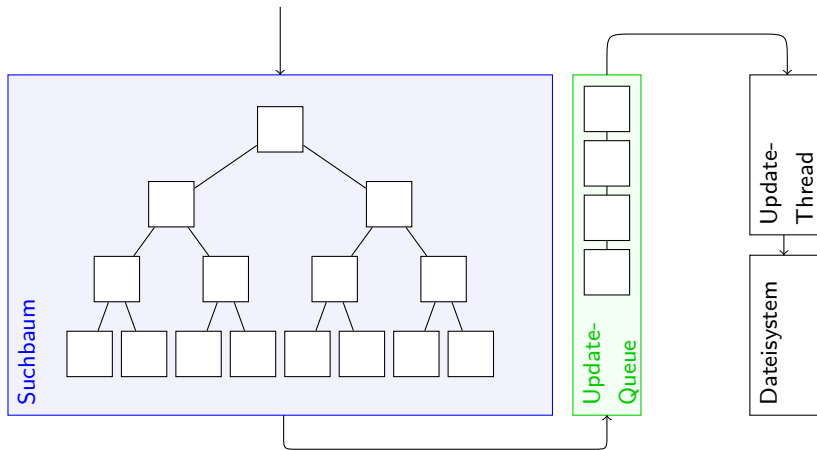
- Update-Prinzip des RRDtool-Plugins
- Eigenständiger Daemon
- Integration in RRDtool 1.4
- Weitere Funktionen, z. B. Journaling
- Vorteil: Neustart von **collectd** ohne Cache-Verlust



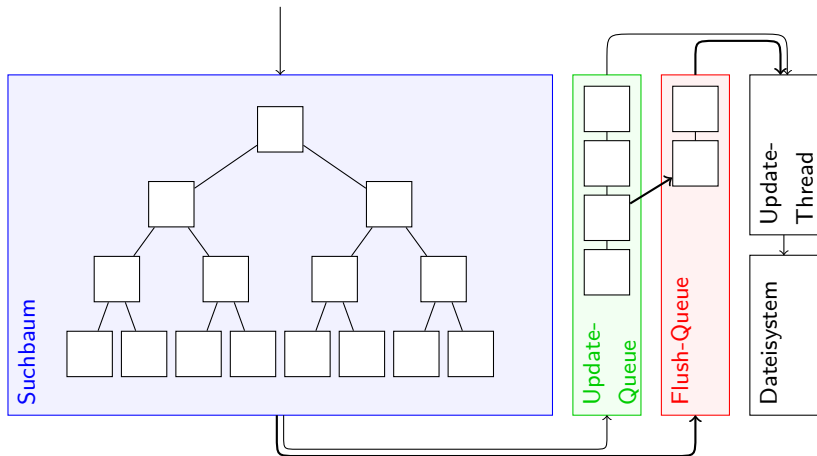
Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster



Graphik © Florian „octo“ Forster

Allgemeines

- Integriert 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