

RRDtool Caching Daemon

# Wie entgehe ich der I/O-Hölle

Sebastian „tokkee“ Harl  
<tokkee@debian.org>

Debian RRDtool Team

**FrOSCamp 2010**  
17. September 2010



## Was ist RRDCacheD?

- ▶ RRDCacheD = RRDtool Caching Daemon
- ▶ Gedacht für große Setups mit I/O-bezogenen Problemen

### Arbeitsweise

1. „Abfangen“ von RRD updates
2. Akkumulieren der Daten
3. Schreiben der gesammelten Daten



## Grundlagen: RRDtool Internas

Arbeitsweise von RRDCacheD

Verwendung von RRDCacheD

Integration von RRDCacheD

Ausblick



## Grundlagen: RRDtools Grundidee

- ▶ Aktuelle Daten sind im Detail am interessantesten
- ▶ Ältere Daten werden mit geringerer Auflösung benötigt (→ Überblick über längere Zeiträume)
- ▶ ⇒ Datenhaltung nach der Round-Robin Methode
- ▶ ⇒ Daten zunehmend verdichten (Konsolidierung)



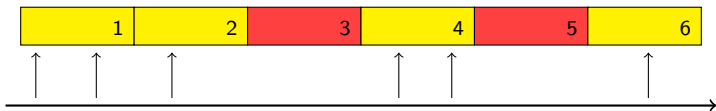
## RRD, RRA, PDP, CF, ... WTF?

Zur Wiederholung, zunächst ein paar Grundbegriffe ...

- ▶ RRD: Round Robin Database
- ▶ DS: Data Source
- ▶ RRA: Round Robin Archive
- ▶ PDP: Primary Data Point
- ▶ CF: Consolidation Function  
→ AVERAGE, MINIMUM, MAXIMUM, ...
- ▶ CDP: Consolidated Data Point



## Primary Data Points (PDP)

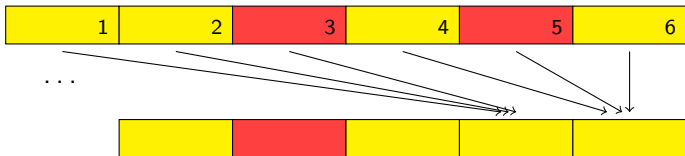


- ▶ Step: Interval der PDPs (Sekunden)
- ▶ Durchschnitt der Messwerte pro Step bilden einen PDP
- ▶ Heartbeat: maximaler Abstand zwischen zwei Messwerten, sonst „unknown“



# Grundlagen: Daten konsolidieren

## Round Robin Archive (RRA)



- ▶ Konsolidierungsfunktionen: AVERAGE, MIN, MAX, LAST
- ▶ xff (xfiles factor): maximale Anteil der unbekannt PDPs, sonst „unknown“
- ▶ Steps: Anzahl der PDPs
- ▶ Rows: Anzahl der CDPs



## updates animiert

RRD file.rrd

Step = 300 s

RRA 0: 5 Minuten-Durchschnitt

RRA 1: 10 Minuten-Durchschnitt

RRA 2: 60 Minuten-Durchschnitt

RRA 3: 60 Minuten-Maximum

Zeit: 45 Minuten

Header
RRA 0
RRA 1
RRA 2
RRA 3





## updates animiert

RRD file.rrd

Step = 300 s

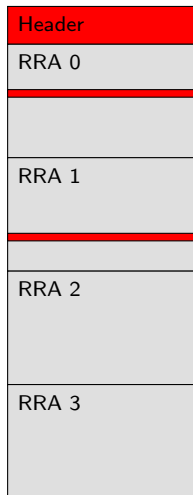
RRA 0: 5 Minuten-Durchschnitt

RRA 1: 10 Minuten-Durchschnitt

RRA 2: 60 Minuten-Durchschnitt

RRA 3: 60 Minuten-Maximum

Zeit: 50 Minuten



## updates animiert

RRD file.rrd

Step = 300 s

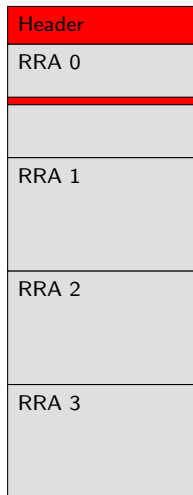
RRA 0: 5 Minuten-Durchschnitt

RRA 1: 10 Minuten-Durchschnitt

RRA 2: 60 Minuten-Durchschnitt

RRA 3: 60 Minuten-Maximum

Zeit: 55 Minuten



## updates animiert

RRD file.rrd

Step = 300 s

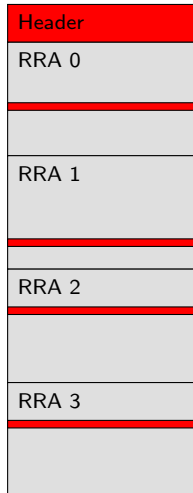
RRA 0: 5 Minuten-Durchschnitt

RRA 1: 10 Minuten-Durchschnitt

RRA 2: 60 Minuten-Durchschnitt

RRA 3: 60 Minuten-Maximum

Zeit: 60 Minuten



update Algorithmus (schematisch):

1. Lesen der Header
2. foreach RRA, das aktualisiert wird:
  - ▶ seek
  - ▶ Lesen der Daten
  - ▶ seek
  - ▶ Schreiben der aktualisierten Daten
3. seek
4. Lesen der Header
5. seek
6. Schreiben der Header



## updates im Detail

```
open("file.rrd", O_RDWR) = 4
read(4, "RRD\0000001\0\0\0\0/%\300\307C+\37[\2\0\0\0"... , 4096) = 4096
_llseek(4, 0, [4096], SEEK_CUR) = 0
_llseek(4, 4096, [4096], SEEK_SET) = 0
_llseek(4, 4096, [4096], SEEK_SET) = 0
_llseek(4, -1324, [2772], SEEK_CUR) = 0
write(4, "\2557Q$\<\314\0@\303k\327.8\316\363?", 16) = 16
_llseek(4, 53248, [53248], SEEK_SET) = 0
read(4, "\0\0\370\377\0\0\0\0\0\0\370\377\0\0\0\0\0"... , 4096) = 4096
_llseek(4, -3372, [53972], SEEK_CUR) = 0
write(4, "\2557Q$\<\314\0@\303k\327.8\316\363?", 16) = 16
_llseek(4, 0, [0], SEEK_SET) = 0
read(4, "RRD\0000001\0\0\0\0/%\300\307C+\37[\2\0\0\0"... , 4096) = 4096
_llseek(4, -2880, [1216], SEEK_CUR) = 0
write(4, "t \370E832936\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0\0"... , 1540) = 1540
close(4)
```

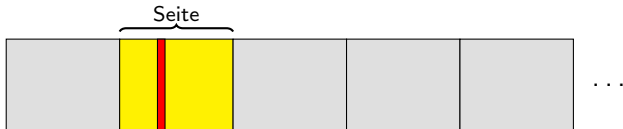
(RRDtool ohne mmap, aber prinzipiell mit identisch)



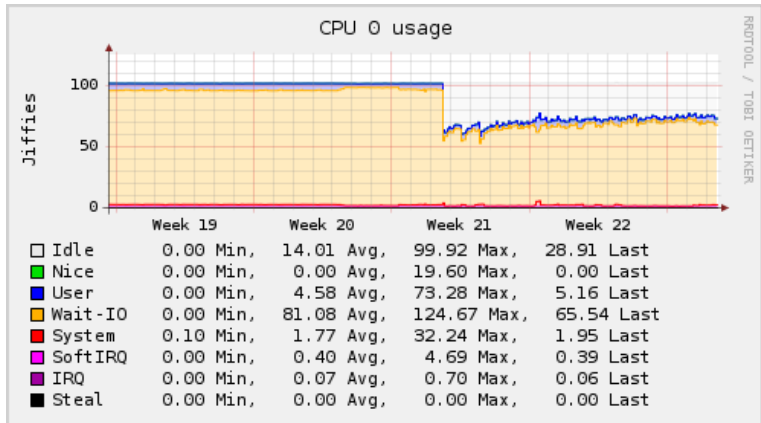
## Problem: Festplatten I/O

**Problem:** In großen Setups kommen Platten schnell an ihr Limit

- ▶ Update schreibt i.d.R. ein Wert („double“, 8 Bytes)
- ▶ Konsolidierung ändert wenige Werte verteilt über die Datei
- ▶ **Aber:** Lesen von / Schreiben auf Platte geschieht in Einheiten der Seitengröße (engl.: page size), z. B. 4096 Bytes
- ▶ Zusätzlich: viele Seeks notwendig



# Problem: Festplatten I/O



Quelle: [http://collectd.org/wiki/index.php/Inside\\_the\\_RRDtool\\_plugin](http://collectd.org/wiki/index.php/Inside_the_RRDtool_plugin)



## Lösungsansätze

- ▶ Klassisch: `tmpfs` und regelmäßige „manuelle“ Synchronisation  
⇒ unflexibel, fehleranfällig, sync von vielen unnötigen Daten
- ▶ Seit RRDtool 1.3: `madvise/fadvise`  
→ kein read-ahead; alte Daten weg; wichtige Daten behalten  
⇒ weiterhin lesen/schreiben von ganzen Blöcken pro update
- ▶ ⇒ updates zusammenfassen und ganze Blöcke schreiben  
→ `RRDCacheD` (seit Version 1.4, Oktober 2009)





Grundlagen: RRDtool Internas

Arbeitsweise von RRDCacheD

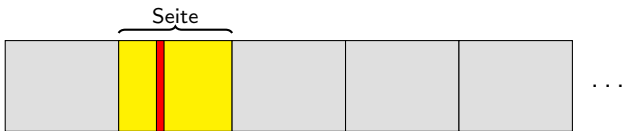
Verwendung von RRDCacheD

Integration von RRDCacheD

Ausblick



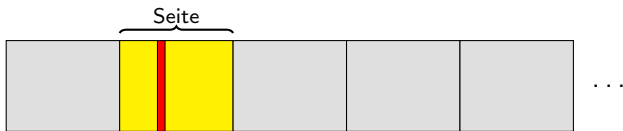
## RRDCacheD: Grundidee



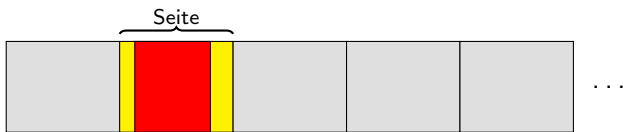
⇒ 1 Seite lesen (z.B. 4096 Bytes); 8 Bytes ändern; 1 Seite schreiben hat nahezu den gleichen Aufwand, wie:



## RRDCacheD: Grundidee



⇒ 1 Seite lesen (z.B. 4096 Bytes); 8 Bytes ändern; 1 Seite schreiben hat nahezu den gleichen Aufwand, wie:

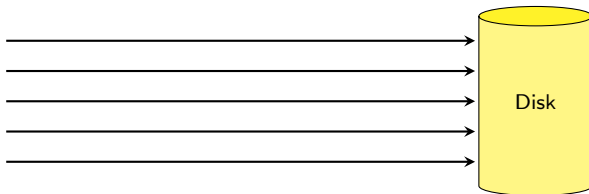


... 1 Seite lesen; viele Daten ändern; 1 Seite schreiben

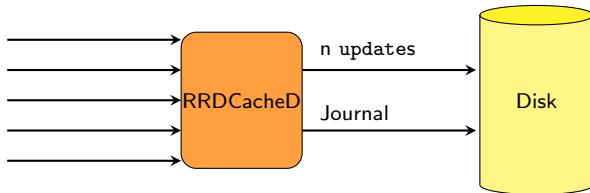
⇒  $4096 / 8 = 512$  mal mehr updates ohne Overhead machbar



## RRDCacheD: Grundidee



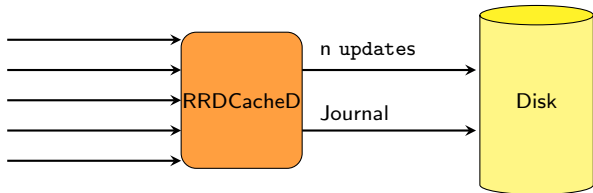
## RRDCacheD: Grundidee



- ▶ „Abfangen“ von updates
- ▶ Ansammeln von Werten
- ▶ Nach Timeout: gesammelte Werte in RRD schreiben



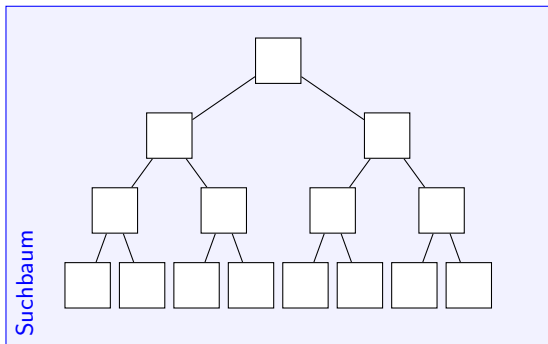
## RRDCacheD: Grundidee



- ▶ „Abfangen“ von updates
- ▶ Ansammeln von Werten
- ▶ Nach Timeout: gesammelte Werte in RRD schreiben
- ▶ Journal vermindert Datenverlust bei Absturz
- ▶ `flush` schreibt (einzelne) Werte sofort



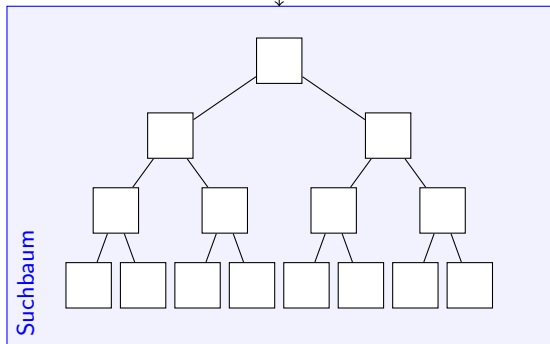
# RRDCacheD: Umsetzung



Graphik © Florian „octo“ Forster



# RRDCacheD: Umsetzung

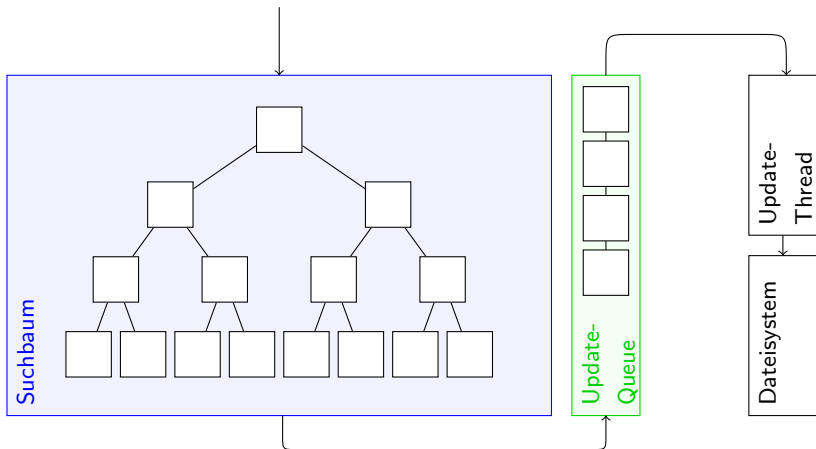


Graphik © Florian „octo“ Forster





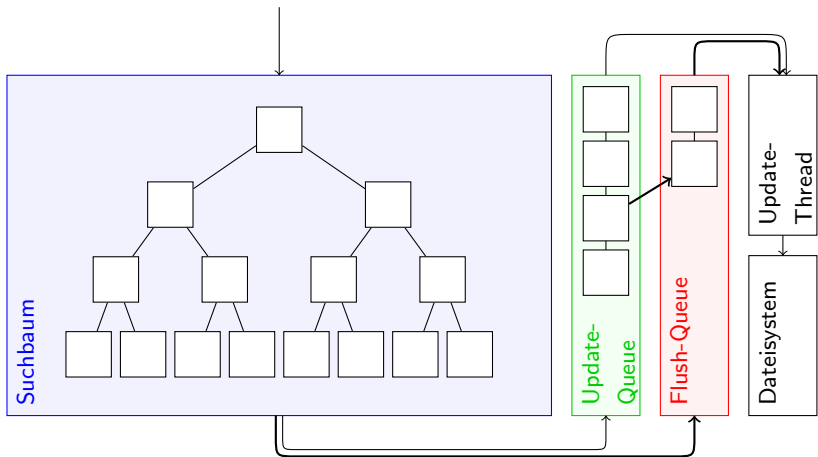
# RRDCacheD: Umsetzung



Graphik © Florian „octo“ Forster



# RRDCacheD: Umsetzung



Graphik © Florian „octo“ Forster



## Server

- ▶ Der eigentliche Caching Daemon
- ▶ Kommunikation über Sockets:  
UNIX Domain oder TCP (IPv4 / IPv6)  
→ Kommunikation mittels Text-basiertem Protokoll

## Client

- ▶ Teil von `librrd`
- ▶ Verbinden zum Daemon
- ▶ Abstraktion des Netzwerk-Protokolls



## RRDCacheD: Parameter/Features

Benutzergruppe des UNIX Sockets

```
man rrdcached; # ;-)
rrdcached \
  -s rrdcached \
  -m 0660 \
  -l unix:/var/run/rrdcached.sock \
  -P FLUSH \
  -l 10.42.23.1 \
  -j /var/lib/rrdcached/journal/ -F \
  -b /var/lib/rrdcached/db/ -B \
  -w 3600 \
  -z 3600 \
  -f 86400
```



## RRDCacheD: Parameter/Features

### Dateirechte des UNIX Sockets

```
man rrdcached; # ;-)  
rrdcached \  
-s rrdcached \  
-m 0660 \  
-l unix:/var/run/rrdcached.sock \  
-P FLUSH \  
-l 10.42.23.1 \  
-j /var/lib/rrdcached/journal/ -F \  
-b /var/lib/rrdcached/db/ -B \  
-w 3600 \  
-z 3600 \  
-f 86400
```



## RRDCacheD: Parameter/Features

UNIX Socket

```
man rrdcached; # ;-)  
rrdcached \  
-s rrdcached \  
-m 0660 \  
-l unix:/var/run/rrdcached.sock \  
-P FLUSH \  
-l 10.42.23.1 \  
-j /var/lib/rrdcached/journal/ -F \  
-b /var/lib/rrdcached/db/ -B \  
-w 3600 \  
-z 3600 \  
-f 86400
```



## RRDCacheD: Parameter/Features

### Einschränkungen für folgende Sockets

```
man rrdcached; # ;-)  
rrdcached \  
-s rrdcached \  
-m 0660 \  
-l unix:/var/run/rrdcached.sock \  
-P FLUSH \  
-l 10.42.23.1 \  
-j /var/lib/rrdcached/journal/ -F \  
-b /var/lib/rrdcached/db/ -B \  
-w 3600 \  
-z 3600 \  
-f 86400
```



## RRDCacheD: Parameter/Features

TCP IPv4 Socket

```
man rrdcached; # ;-)
```

```
rrdcached \  
-s rrdcached \  
-m 0660 \  
-l unix:/var/run/rrdcached.sock \  
-P FLUSH \  
-l 10.42.23.1 \  
-j /var/lib/rrdcached/journal/ -F \  
-b /var/lib/rrdcached/db/ -B \  
-w 3600 \  
-z 3600 \  
-f 86400
```





## RRDCacheD: Parameter/Features

### Journal Einstellungen

```
man rrdcached; # ;-)
rrdcached \
  -s rrdcached \
  -m 0660 \
  -l unix:/var/run/rrdcached.sock \
  -P FLUSH \
  -l 10.42.23.1 \
  -j /var/lib/rrdcached/journal/ -F \
  -b /var/lib/rrdcached/db/ -B \
  -w 3600 \
  -z 3600 \
  -f 86400
```



## RRDCacheD: Parameter/Features

### Basisverzeichnis Einstellungen

```
man rrdcached; # ;-)  
rrdcached \  
-s rrdcached \  
-m 0660 \  
-l unix:/var/run/rrdcached.sock \  
-P FLUSH \  
-l 10.42.23.1 \  
-j /var/lib/rrdcached/journal/ -F \  
-b /var/lib/rrdcached/db/ -B \  
-w 3600 \  
-z 3600 \  
-f 86400
```



## RRDCacheD: Parameter/Features

Timeout

```
man rrdcached; # ;-)
rrdcached \
  -s rrdcached \
  -m 0660 \
  -l unix:/var/run/rrdcached.sock \
  -P FLUSH \
  -l 10.42.23.1 \
  -j /var/lib/rrdcached/journal/ -F \
  -b /var/lib/rrdcached/db/ -B \
  -w 3600 \
  -z 3600 \
  -f 86400
```



## RRDCacheD: Parameter/Features

Verzögerung

```
man rrdcached; # ;-)  
rrdcached \  
-s rrdcached \  
-m 0660 \  
-l unix:/var/run/rrdcached.sock \  
-P FLUSH \  
-l 10.42.23.1 \  
-j /var/lib/rrdcached/journal/ -F \  
-b /var/lib/rrdcached/db/ -B \  
-w 3600 \  
-z 3600 \  
-f 86400
```



## RRDCacheD: Parameter/Features

„garbage collection“

```
man rrdcached; # ;-)
rrdcached \
  -s rrdcached \
  -m 0660 \
  -l unix:/var/run/rrdcached.sock \
  -P FLUSH \
  -l 10.42.23.1 \
  -j /var/lib/rrdcached/journal/ -F \
  -b /var/lib/rrdcached/db/ -B \
  -w 3600 \
  -z 3600 \
  -f 86400
```



## RRDCacheD: Basisverzeichnis

Basisverzeichnis: `/var/lib/rrdcached/db/`  
(Standard: `/tmp/!`)

Kommandozeile	verwendete Datei
<code>foo.rrd</code>	<code>/var/lib/rrdcached/db/foo.rrd</code>
<code>foo/bar.rrd</code>	<code>/var/lib/rrdcached/db/foo/bar.rrd</code>
<code>/tmp/foo.rrd</code>	<code>/tmp/foo.rrd</code> mit <code>-B</code> : nicht akzeptiert



Grundlagen: RRDtool Internas

Arbeitsweise von RRDCacheD

Verwendung von RRDCacheD

Integration von RRDCacheD

Ausblick



## Verfügbare Kommandos

- ▶ dump\*
- ▶ fetch\*
- ▶ flush
- ▶ graph\*
- ▶ graphv\*
- ▶ info\*
- ▶ last\*
- ▶ lastupdate\*
- ▶ update
- ▶ xport\*





## Verwendung von RRDCacheD

- ▶ `--daemon` Kommandozeilen Option
- ▶ `RRDCACHED_ADDRESS` Umgebungsvariable  
⇒ erlaubt vollkommen transparente Integration



## Verwendung von RRDCacheD

- ▶ `--daemon` Kommandozeilen Option
- ▶ `RRDCACHED_ADDRESS` Umgebungsvariable  
⇒ erlaubt vollkommen transparente Integration
- ▶ Achtung:
  - ▶ Basisverzeichnis Einstellungen
  - ▶ `rrdupdates --template` nicht unterstützt
  - ▶ Ausreichend RAM verfügbar haben!



## Verwendung von RRDCacheD

- ▶ `--daemon` Kommandozeilen Option
- ▶ `RRDCACHED_ADDRESS` Umgebungsvariable  
⇒ erlaubt vollkommen transparente Integration
- ▶ Achtung:
  - ▶ Basisverzeichnis Einstellungen
  - ▶ `rrdupdates --template` nicht unterstützt
  - ▶ **Ausreichend RAM verfügbar haben!**



## C API

```
int status;  
  
char *values [] = { "1234567890:42", "1234567900:23" };  
int values_num = 2;  
  
/* daemon_address == NULL => verwende RRDCACHED_ADDRESS */  
status = rrdc_connect(daemon_address);  
if (status) {  
    fprintf(stderr, "ERROR: %s\n", rrd_get_error());  
    exit(1);  
}  
  
status = rrdc_update("file.rrd", values_num, values);  
if (status) /* ... */;
```



## Perl „API“

```
use RRDs;  
  
RRDs::update("file.rrd", "--daemon", $daemon_address ,  
             "1234567890:42", "1234567900:23" );  
my $err = RRDs::error;  
die "ERROR: $err" if $err;
```



# Netzwerk-Protokoll

- ▶ Text- und Zeilen-basiert
- ▶ Kommandos bestehen aus dem Befehlsnamen und ggf. Parametern  
z.B. `FLUSH foo.rrd`
- ▶ Antwort des Servers besteht aus Status-Code und kurzer Nachricht
  - ▶ Status  $< 0$ : Fehler (Beschreibung folgt in gleicher Zeile)
  - ▶ Status  $\geq 0$ : Erfolg; Status entspricht Anzahl weiterer Zeilenz.B. `0 Success`



## Netzwerk-Protokoll: BATCH

- ▶ „Block-Abarbeitung“ von vielen Kommandos
- ▶ Spart read und write Aufrufe  
→ für sehr große Setups mit sehr großen Datenraten

```
->: BATCH
<-: 0 Go ahead. End with dot '.' on its own line.
->: UPDATE x.rrd 1223661439:1:2:3
->: UPDATE y.rrd 1223661440:3:4:5
->: ...
->: .
<-: 2 Errors
<-: 1 <Fehlermeldung für das 1. Kommando>
<-: 12 <Fehlermeldung für das 12. Kommando>
```



## Sicherheit

- ▶ Bislang nur Einschränkung von erlaubten Kommandos möglich
- ▶ **Alles**, was an einem Socket ankommt, wird angenommen
- ▶ Netzwerk-Verkehr ist unverschlüsselt
- ▶ Daemon sollte mit eingeschränkten Benutzerrechten laufen
  
- ▶ ⇒ Admin muss sich um geeignete Abschottung kümmern, z.B. dediziertes (V)LAN, VPN, o.ä.





Grundlagen: RRDtool Internas

Arbeitsweise von RRDCacheD

Verwendung von RRDCacheD

Integration von RRDCacheD

Ausblick



## Integration in collectd

- ▶ rrdcached Plugin  
<http://collectd.org/wiki/index.php/Plugin:RRDCacheD>
- ▶ seit Version 4.6 (02/2009)

```
LoadPlugin rrdcached
<Plugin rrdcached>
    DaemonAddress "unix:/var/run/rrdcached.sock"
</Plugin>
```



## Integration in Nagios

- ▶ PNP4Nagios unterstützt RRDCached seit Version 0.4.11
- ▶ <http://www.pnp4nagios.org/pnp/rrdcached>
- ▶ <http://www.semintelligent.com/blog/articles/40/nagios-performance-tuning-early-lessons-learned-lessons-shared-part-45-scalable-performance-data-graphing>



## Integration in PNP4Nagios

process\_perfdata.cfg

```
RRD_DAEMON_OPTS = 127.0.0.1:8888
```

config\_local.php

```
$conf['RRD_DAEMON_OPTS'] = '127.0.0.1:8888';
```



## Integration in andere Software

- ▶ Cacti:
  - ▶ <http://binaryfury.wann.net/2010/01/rrdcached-and-cacti-not-quite-there-yet/>
  - ▶ <https://lists.oetiker.ch/pipermail/rrd-developers/2010-March/003664.html>
- ▶ Ganglia:
  - ▶ [http://sourceforge.net/apps/trac/ganglia/wiki/rrdcached\\_integration](http://sourceforge.net/apps/trac/ganglia/wiki/rrdcached_integration)
- ▶ Munin:
  - ▶ <http://munin-monitoring.org/ticket/444>
- ▶ openNMS:
  - ▶ <http://www.opennms.org/wiki/Rrdcached>



Grundlagen: RRDtool Internas

Arbeitsweise von RRDCacheD

Verwendung von RRDCacheD

Integration von RRDCacheD

Ausblick



- ▶ Echte Unterstützung für `fetch` und `graph`  
`DEF:v1=path/to/example.rrd:value:AVERAGE:daemon=r2d2.example.org`  
→ Graph von verteilten Daten (in trunk)
- ▶ Verschlüsselung
- ▶ Echte Authentifizierung
- ▶ Redundante Setups?
- ▶ Drosselung (bessere Alternative zu `-z?`)



- ▶ Echte Unterstützung für `fetch` und `graph`  
`DEF:v1=path/to/example.rrd:value:AVERAGE:daemon=r2d2.example.org`  
→ Graph von verteilten Daten (in trunk)
- ▶ Verschlüsselung
- ▶ Echte Authentifizierung
- ▶ Redundante Setups?
- ▶ Drosselung (bessere Alternative zu `-z?`)
- ▶ It's OpenSource! ... send patches! ;-)





## Wie entgehe ich der I/O-Hölle

Vielen Dank für die Aufmerksamkeit!

Gibt es Fragen?

Kontakt:  
Sebastian „tokkee“ Harl  
<tokkee@debian.org>

