RRDtool Caching Daemon

How to Escape the I/O Hell

Sebastian "tokkee" Harl
<tokkee@debian.org>

Debian RRDtool Team

OSMC 2010
October 6, 2010

About RRDCacheD?

- RRDCacheD = RRDtool Caching Daemon
- meant for **large** setups to solve I/O-related problems

In short:

1. "intercept" RRD `updates`
2. accumulate data
3. write accumulated data to disk at once

Background: RRDtool internals

RRDCacheD – behind the scenes

Using RRDCacheD

Integration of RRDCacheD

Future directions

Background: basic idea of RRDtool

- current values should be available with high resolution
- older values are increasingly less interesting ($\rightarrow$ overview over large time-spans)
- $\Rightarrow$ consolidate old data with increasing density
- $\Rightarrow$ keep data using the round-robin method

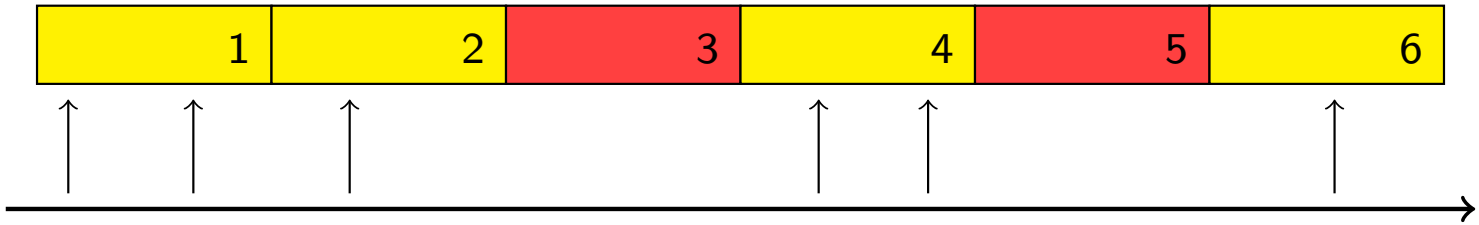
RRD, RRA, PDP, CF, ... WTF?

Quick overview over basic concepts of RRDtool:

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

Background: data storage

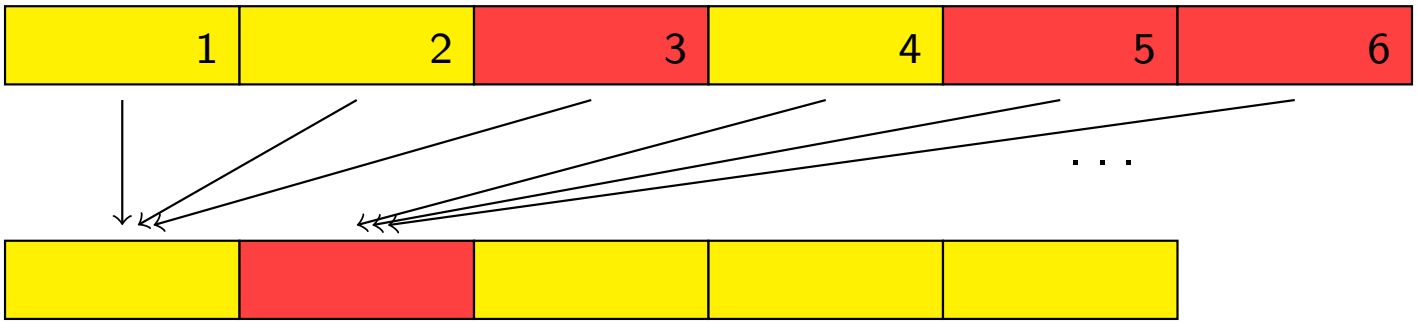
Primary Data Points (PDP)



- step: interval of an PDP (seconds)
- average value of all updates in one step form a PDP
- heartbeat: maximum amount of time that may pass before considering a value "unknown"

Background: consolidating data

Round Robin Archive (RRA)



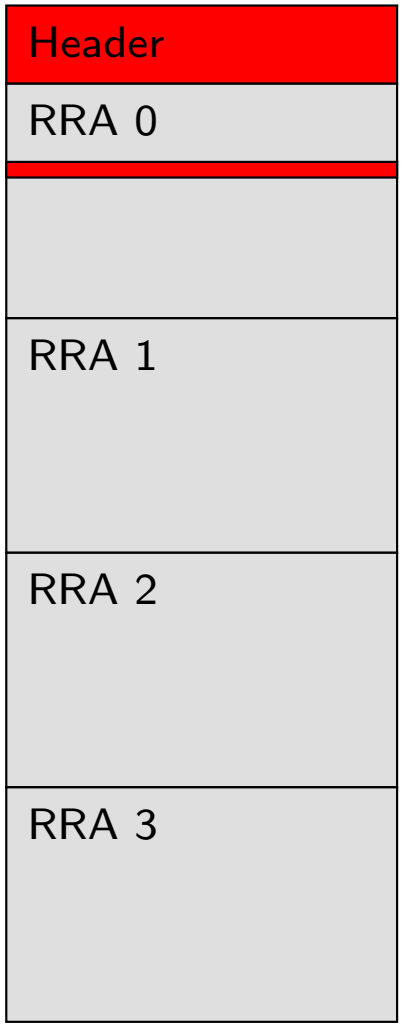
- consolidation functions: `AVERAGE`, `MIN`, `MAX`, `LAST`
- xff (xfiles factor): maximum amount of unknown PDPs allowed for a valid CDP
- steps: number of PDPs
- rows: number of CDPs

updates animated

RRD `file.rrd`
step = 300 s

RRA 0: 5 minutes-average
RRA 1: 10 minutes-average
RRA 2: 60 minutes-average
RRA 3: 60 minutes-maximum

current time: 45 minutes

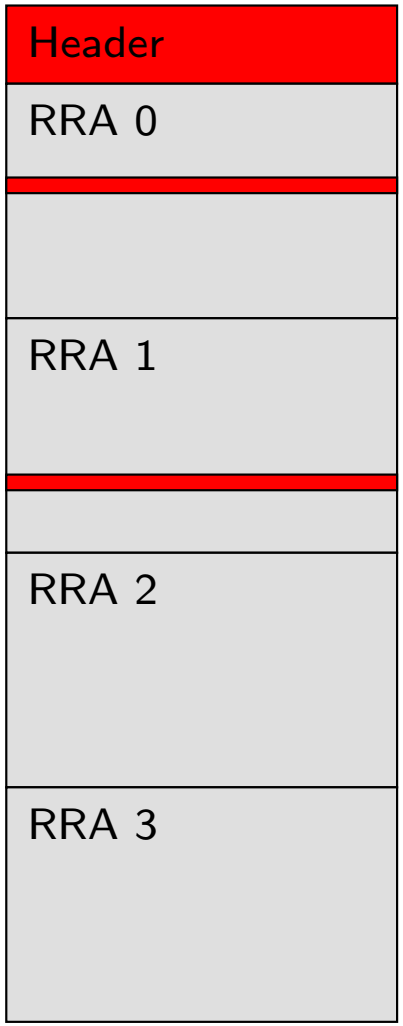


updates animated

RRD `file.rrd`
step = 300 s

RRA 0: 5 minutes-average
RRA 1: 10 minutes-average
RRA 2: 60 minutes-average
RRA 3: 60 minutes-maximum

current time: 50 minutes

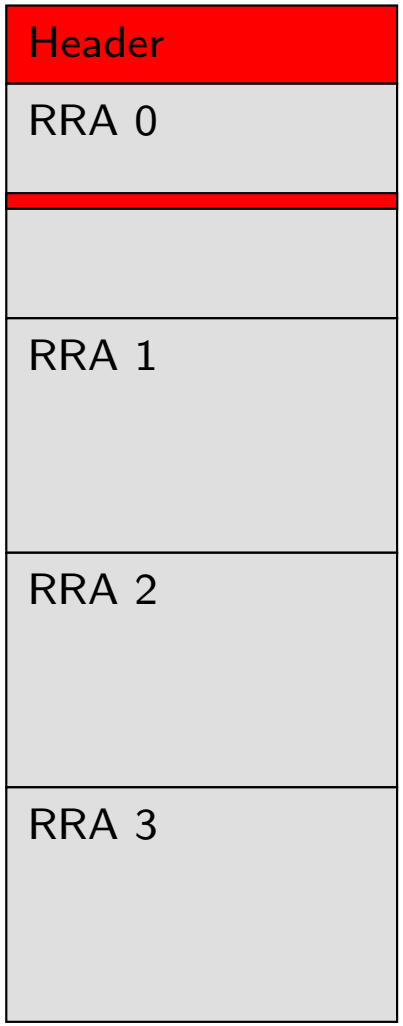


updates animated

RRD `file.rrd`
step = 300 s

RRA 0: 5 minutes-average
RRA 1: 10 minutes-average
RRA 2: 60 minutes-average
RRA 3: 60 minutes-maximum

current time: 55 minutes

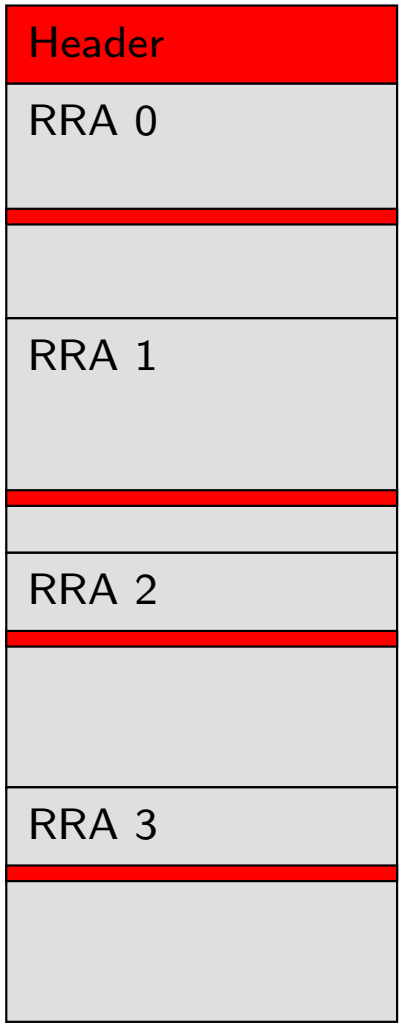


updates animated

RRD `file.rrd`
step = 300 s

RRA 0: 5 minutes-average
RRA 1: 10 minutes-average
RRA 2: 60 minutes-average
RRA 3: 60 minutes-maximum

current time: 60 minutes



updates in detail

`update` algorithm (schematic):

1. read headers
2. foreach RRA to be updated:
   - `seek`
   - read data
   - `seek`
   - write back modified data
3. `seek`
4. read headers
5. `seek`
6. write headers

updates in 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$<\314\0@\303k\327.8\316\363?", 16) = 16
_llseek(4, 53248, [53248], SEEK_SET)     = 0
read(4, "\0\0\370\377\0\0\0\0\0\0\370\377\0\0\0\0\0\0"..., 4096) = 4096
_llseek(4, -3372, [53972], SEEK_CUR)     = 0
write(4, "\2557Q$<\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 without `mmap`, else those details are hidden)

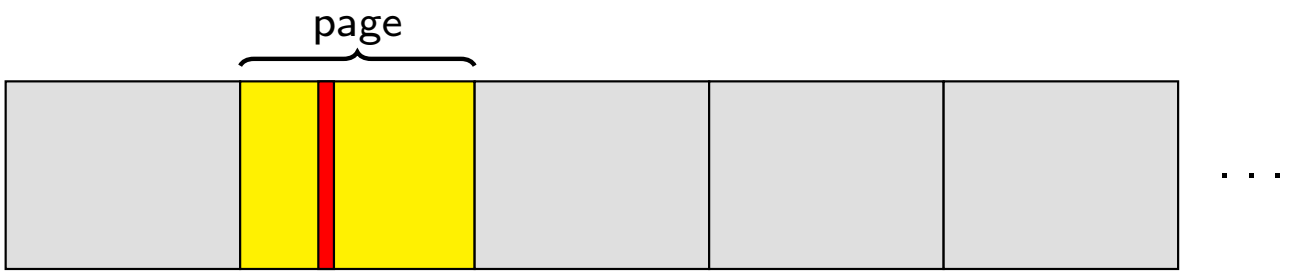
Problem: disk I/O

Problem: in large setups, hard-disks soon hit their limits

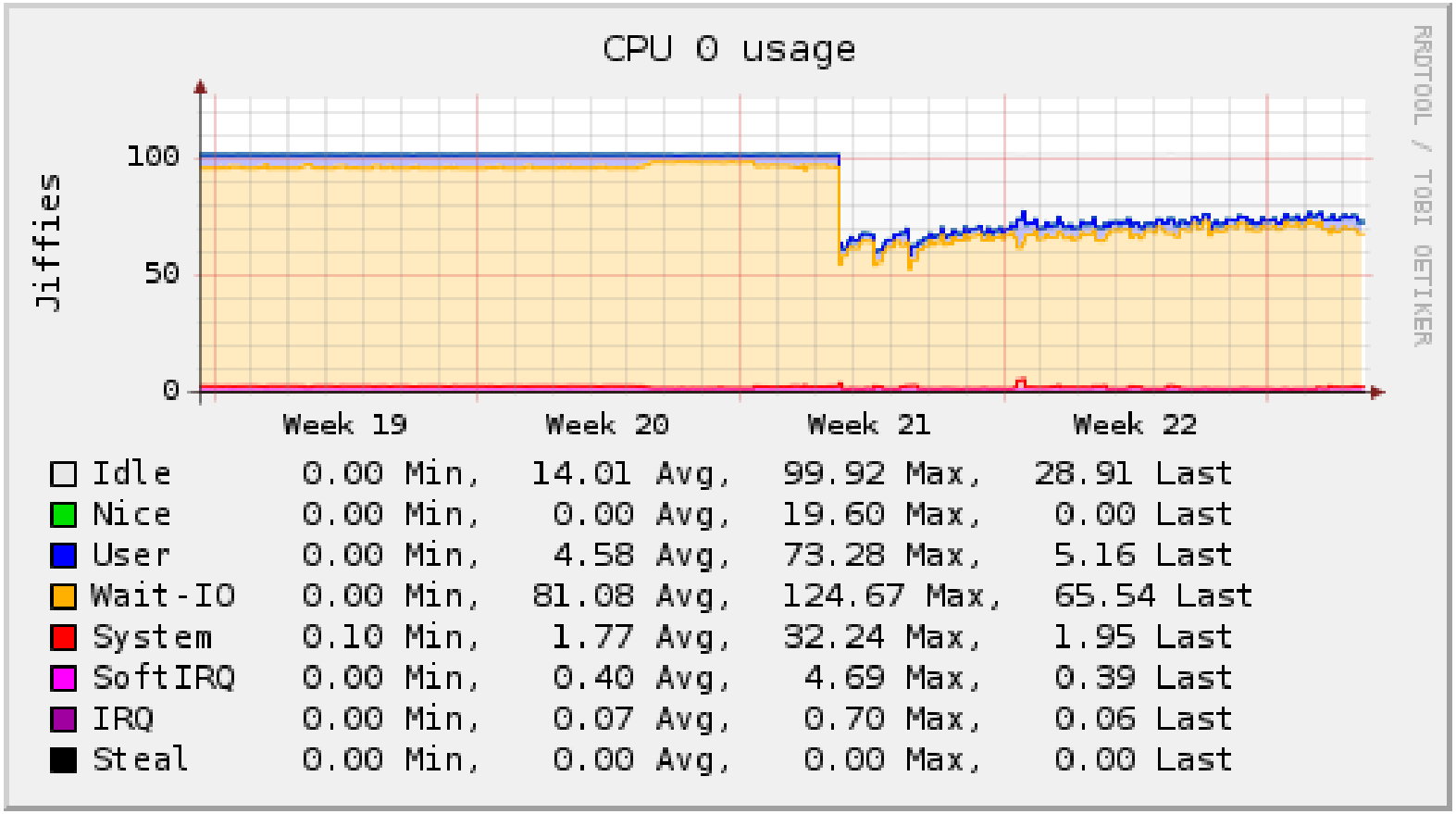
Problem: disk I/O

Problem: in large setups, hard-disks soon hit their limits

- `update` usually modifies a single value ("double", 8 bytes)
- consolidation modifies multiple CDPs spread over the file
- **but:** disk I/O is done in multiples of a page size (usually 4096 bytes)
- in addition: lots of seeks necessary
- disks are optimized for sequential access



Problem: disk I/O



source: http://collectd.org/wiki/index.php/Inside_the_RRDtool_plugin

Solutions so far

- "ancient days": `tmpfs` and manual synchronization

  $\Rightarrow$ inflexible, error-prone, requires sync of lots of data

- RRDtool 1.3: `madvise`/`fadvise`
  $\rightarrow$ no read-ahead; optimize file-system caching

  $\Rightarrow$ still requires reading/writing of whole pages on `update`

- $\Rightarrow$ accumulate `updates` and write multiple, sequential updates at once

  $\rightarrow$ RRDCacheD (since RRDtool 1.4, October 2009)

Background: RRDtool internals

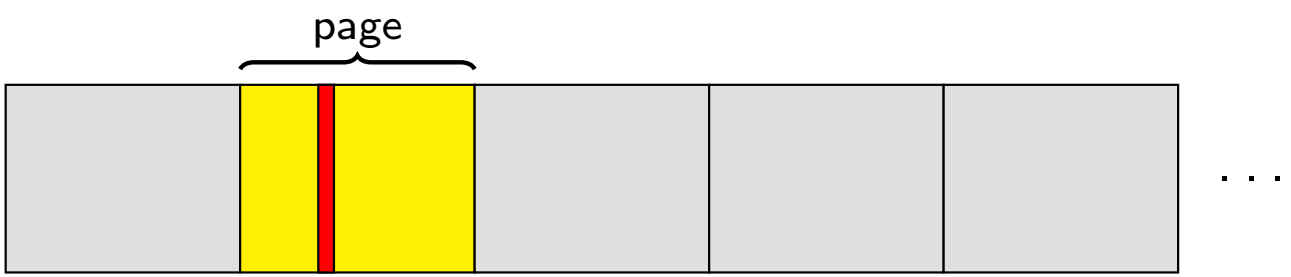
RRDCacheD – behind the scenes

Using RRDCacheD

Integration of RRDCacheD

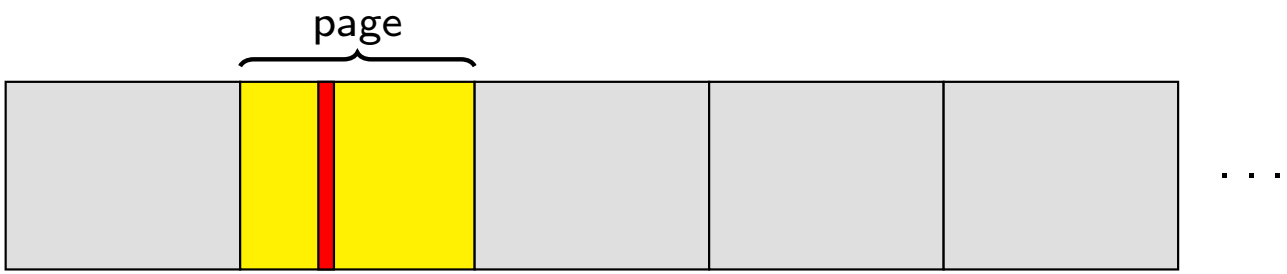
Future directions

RRDCacheD: basics

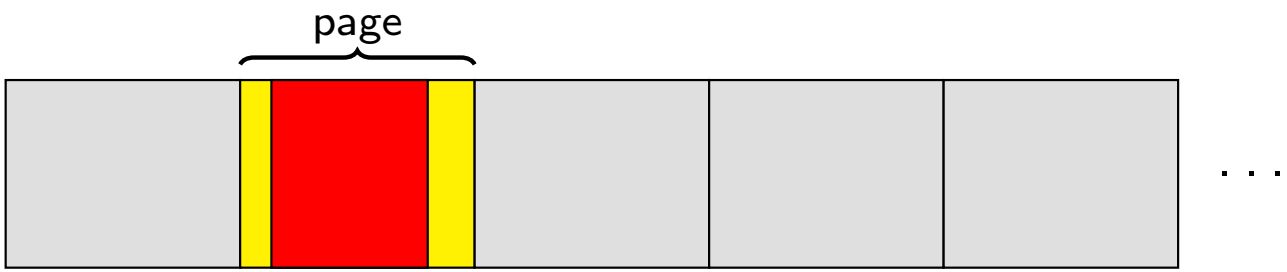


⇒ reading 1 page (e. g., 4096 Bytes); changing 8 bytes; writing back 1 page basically has the same overhead as:

RRDCacheD: basics



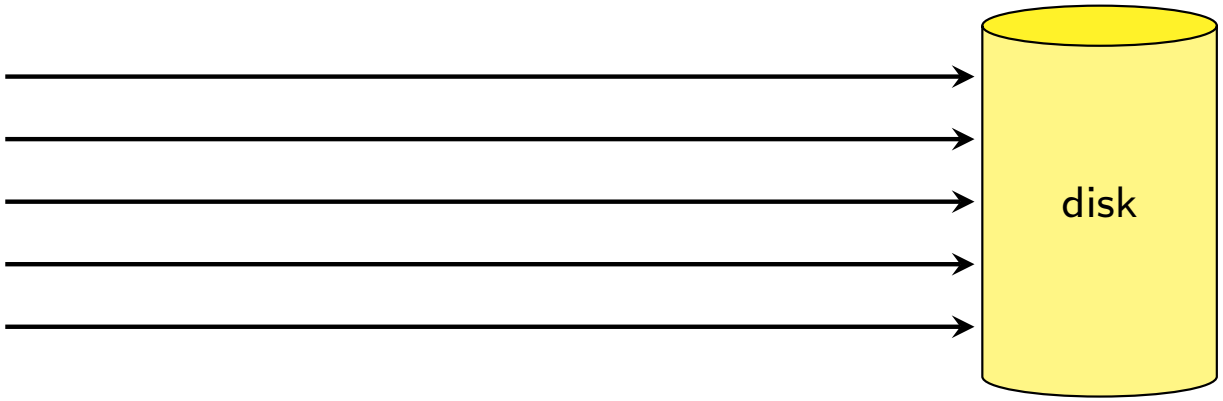
⇒ reading 1 page (e. g., 4096 Bytes); changing 8 bytes; writing back 1 page basically has the same overhead as:



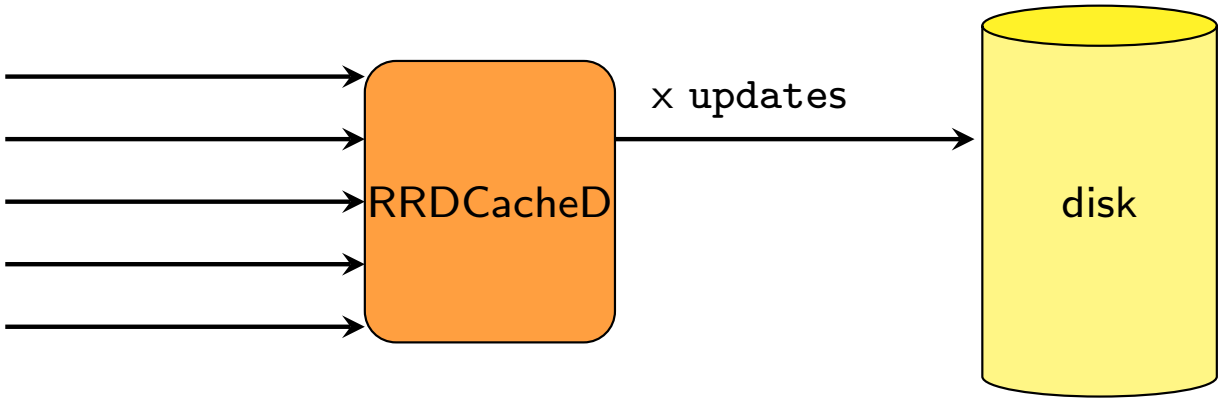
. . . reading 1 page; changing multiple values; writing back 1 page

⇒ 4096 / 8 = 512 times more updates possible (in theory)

RRDCacheD: basics

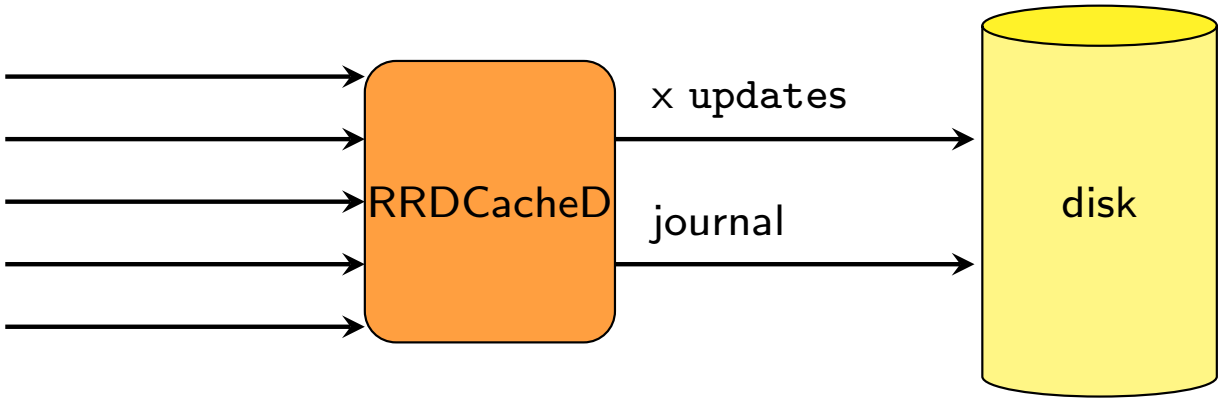


RRDCacheD: basics



- "intercept" RRD `updates`
- accumulate values
- on timeout: write accumulated values to disk

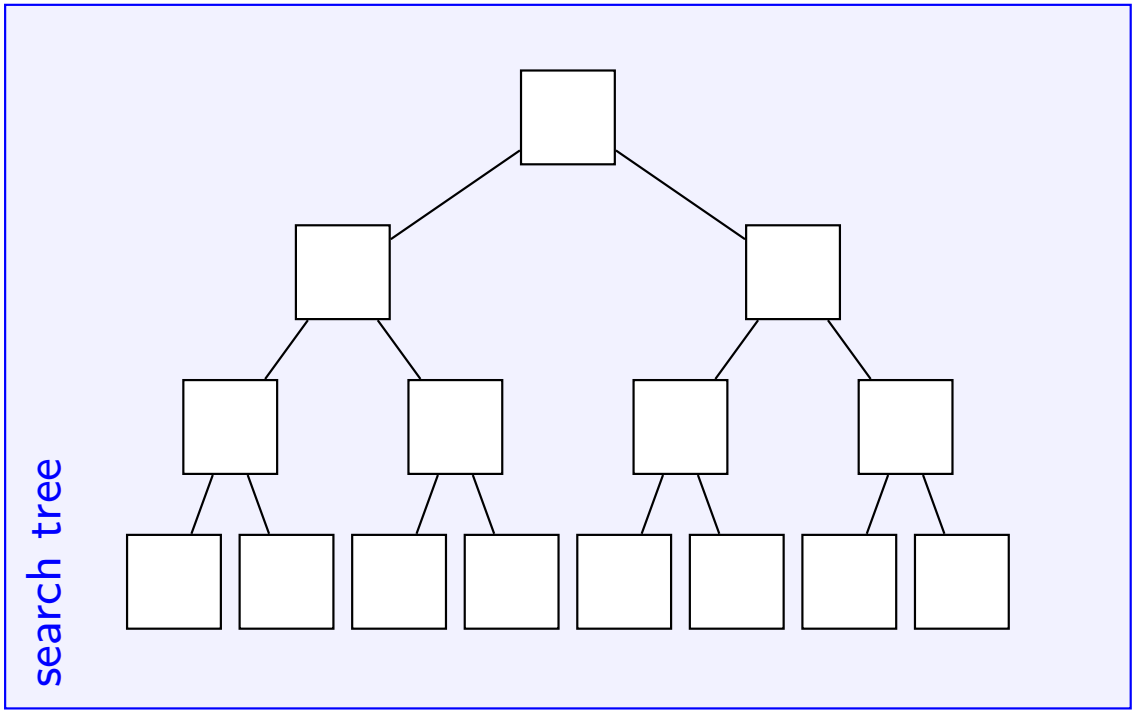
RRDCacheD: basics



- "intercept" RRD `updates`
- accumulate values
- on timeout: write accumulated values to disk

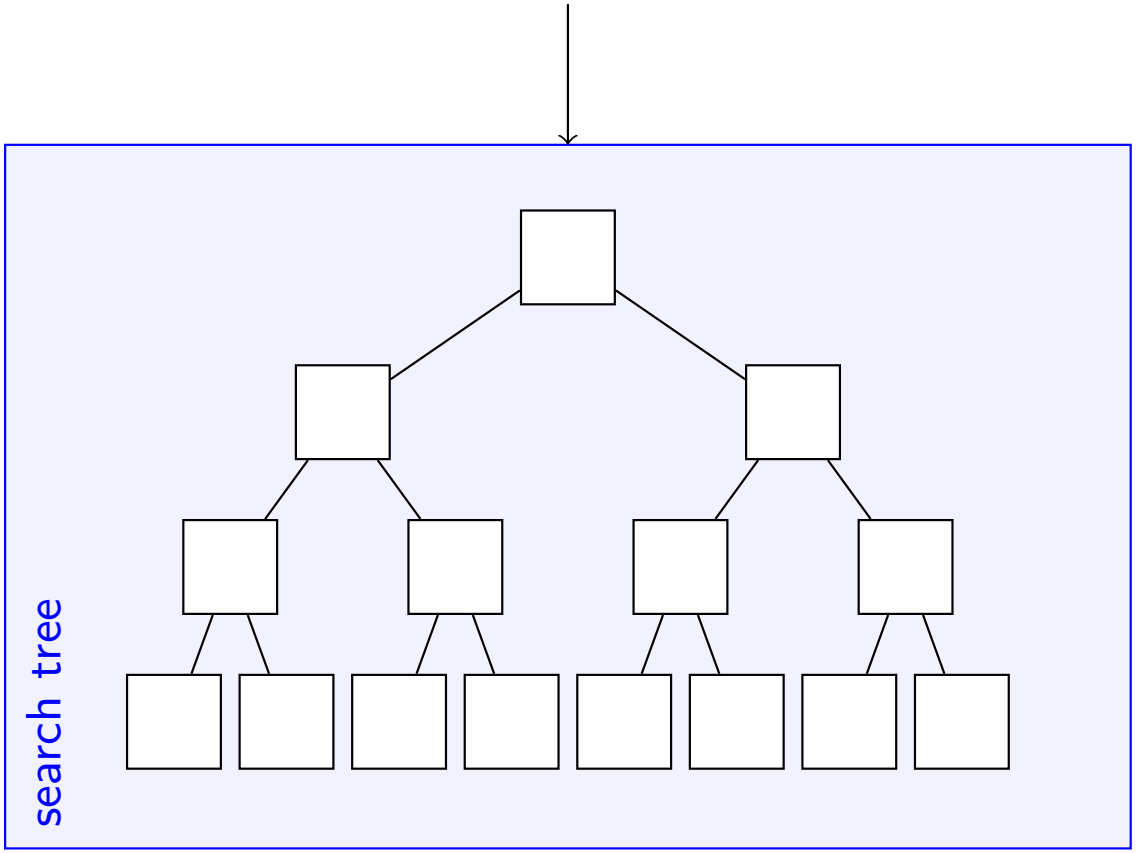
- journal allows recovery after crash
- `flush` writes back single value on request

RRDCacheD: implementation



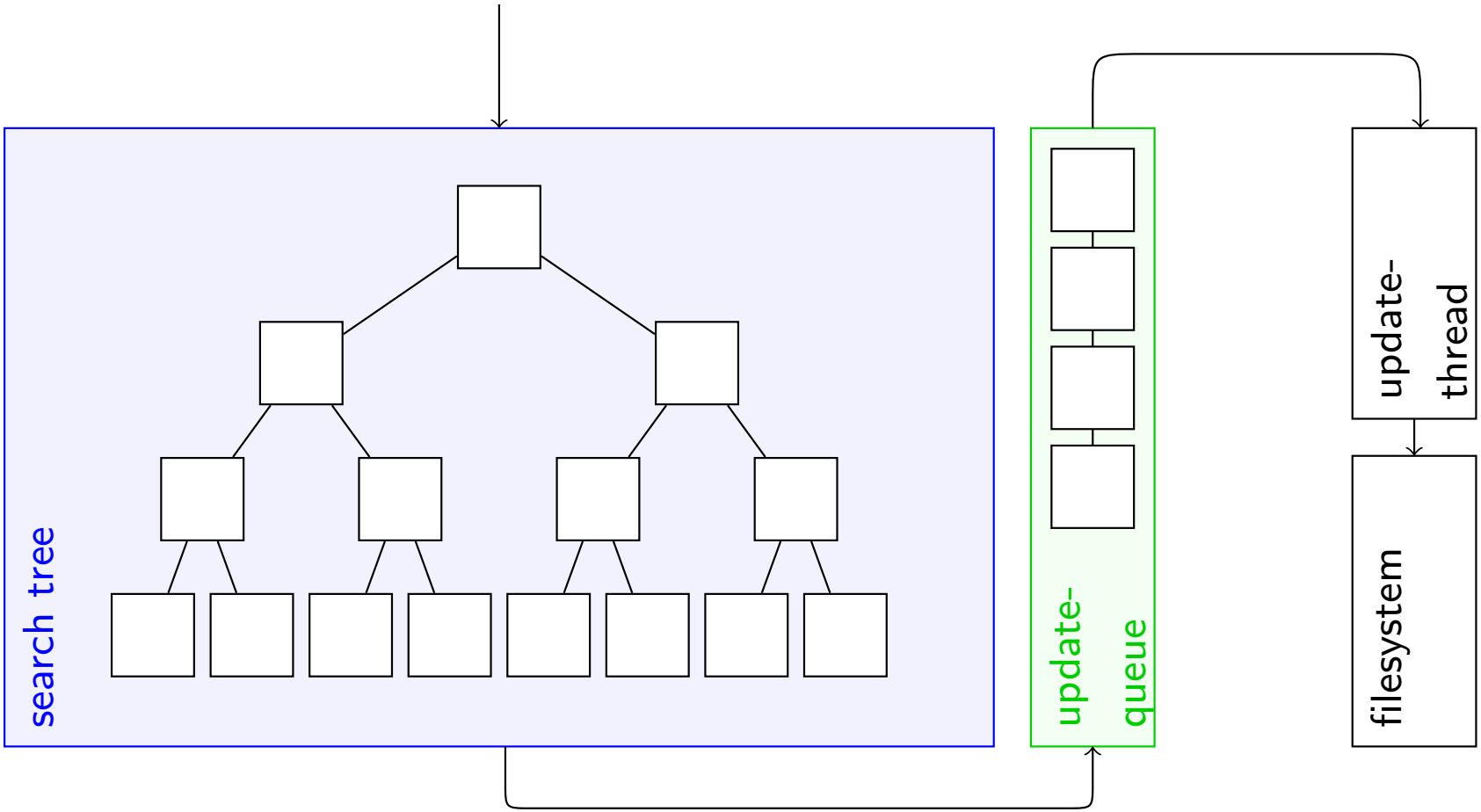
graphic © Florian "octo" Forster

RRDCacheD: implementation



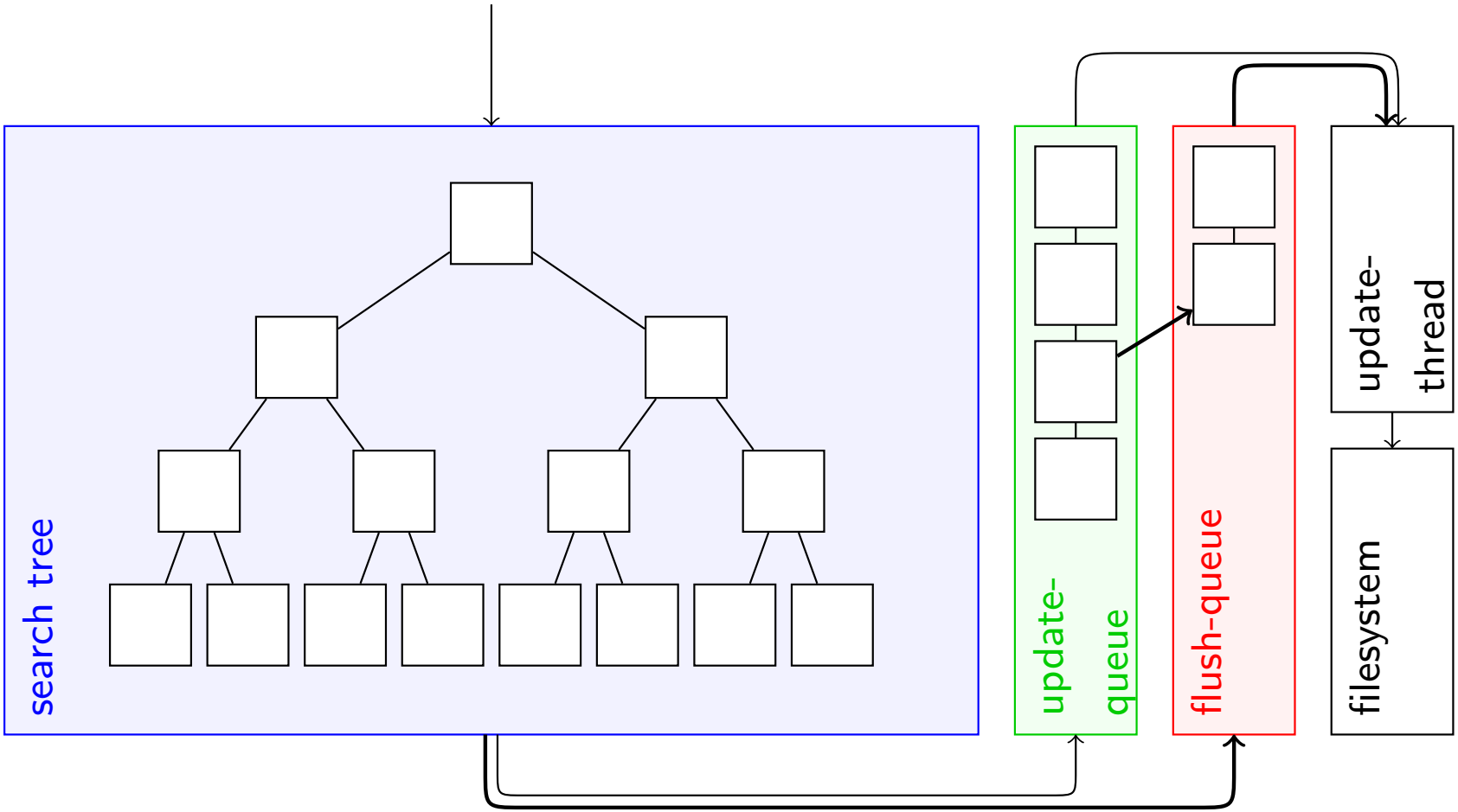
graphic © Florian "octo" Forster

RRDCacheD: implementation



graphic © Florian "octo" Forster

RRDCacheD: implementation



graphic © Florian "octo" Forster

RRDCacheD: implementation

Server

- the actual caching daemon
- access to the daemon using sockets:
  UNIX Domain or TCP (IPv4 / IPv6)
  $\rightarrow$ communication using a text-based protocol

Client

- implemented in `librrd`
- connection handling
- abstraction of the network protocol

RRDCacheD: parameters/features

owner of the 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: parameters/features

file permissions of the 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: parameters/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: parameters/features

access control for 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: parameters/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: parameters/features

journal settings

```
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: parameters/features

base directory settings

```
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: parameters/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: parameters/features

delay

```
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: parameters/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: base directory

recommendation: `/var/lib/rrdcached/db/`
(default: `/tmp/`!)

command line	file system
`foo.rrd`	`/var/lib/rrdcached/db/foo.rrd`
`foo/bar.rrd`	`/var/lib/rrdcached/db/foo/bar.rrd`
`/tmp/foo.rrd`	`/tmp/foo.rrd` rejected when using `-B`

Background: RRDtool internals

RRDCacheD – behind the scenes

Using RRDCacheD

Integration of RRDCacheD

Future directions

Available commands

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

* no “native” implementation (yet?)

Using RRDCacheD

- `--daemon` command line option
- `RRDCACHED_ADDRESS` environment variable
  $\Rightarrow$ fully transparent integration

Using RRDCacheD

- `--daemon` command line option
- `RRDCACHED_ADDRESS` environment variable
  $\Rightarrow$ fully transparent integration

- Caveats:
  - base directory settings
  - `rrdupdate`'s `--template` not supported
  - sufficient RAM needs to be available!

Using RRDCacheD

- `--daemon` command line option
- `RRDCACHED_ADDRESS` environment variable
  $\Rightarrow$ fully transparent integration

- Caveats:
  - base directory settings
  - `rrdupdate`'s `--template` not supported
  - sufficient RAM needs to be available!

C API

```c
int status;

char *values[] = { "1234567890:42", "1234567900:23" };
int   values_num = 2;

/* daemon_address == NULL => use $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”

```perl
use RRDs;

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

Network Protocol

- text and line based
- lines contain a command name and optional parameters
  e. g., `FLUSH foo.rrd`
- response includes status code and message
  - status < 0: error condition (details included on same line)
  - status ≥ 0: success; number of further lines returned

  e. g., `0 Success`

Network protocol: BATCH

- batch processing of multiple commands
- reduce number of calls to `read` and `write`
  $\rightarrow$ optimization for **really** large setups with high data rates

```
->:  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 <error message for 1. command>
<-:  12 <error message for 12. command>
```

Security considerations

- RRDCacheD allows to limit acceptable commands per socket
- **all** data arriving at a socket will be accepted
- network traffic is unencrypted
- daemon should run as unprivileged user

- $\Rightarrow$ admin is responsible for security
  e. g., dedicated (V)LAN, VPN, etc.

Background: RRDtool internals

RRDCacheD – behind the scenes

Using RRDCacheD

Integration of RRDCacheD

Future directions

Integration in **collectd**

- `rrdcached` Plugin

  `http://collectd.org/wiki/index.php/Plugin:RRDCacheD`
- since version 4.6 (02/2009)

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

Integration in Nagios

- PNP4Nagios supports RRDCached since 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 other 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`
- Munin:
  - `http://munin-monitoring.org/ticket/444`
- openNMS:
  - `http://www.opennms.org/wiki/Rrdcached`

Background: RRDtool internals

RRDCacheD – behind the scenes

Using RRDCacheD

Integration of RRDCacheD

Future directions

Future directions

- "Native" support for `fetch` und `graph`
  First steps:
  `DEF:v1=path/to/example.rrd:value:AVERAGE:daemon=r2d2.example.org`
  → graphing of distributed data (in `trunk`)
- encryption
- real authentication
- redundant/high availability setups

Future directions

- "Native" support for `fetch` und `graph`
  First steps:
  `DEF:v1=path/to/example.rrd:value:AVERAGE:daemon=r2d2.example.org`
  → graphing of distributed data (in `trunk`)
- encryption
- real authentication
- redundant/high availability setups

- It's OpenSource! ... send patches! ;-)

How to Escape the I/O Hell

Thanks for your attention!

Any questions?

contact:
Sebastian “tokkee” Harl
<tokkee@debian.org>