Performance Analysis and Capacity Planing

collectd – the system statistics collection daemon

Sebastian 'tokkee' Harl
<sh@teamix.net>

teamix GmbH / **collect**d core team

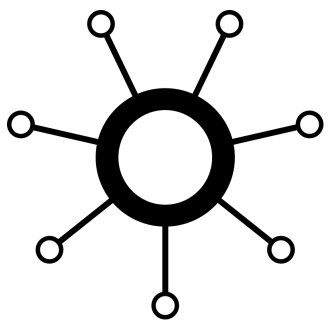
Libre Software Meeting 2012
July 10, 2012

Solid IT-Infrastructure
Location: Nuremberg, Munich, Frankfurt

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

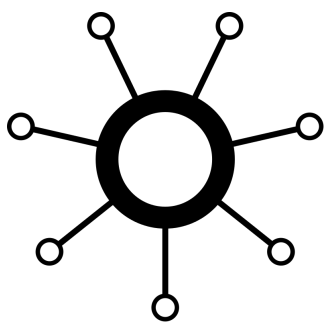
collectd overview

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



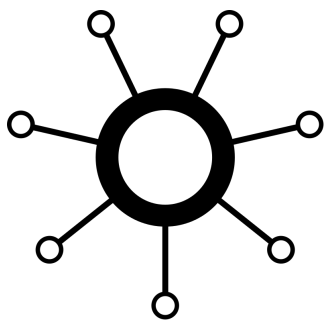
Main features

- 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)

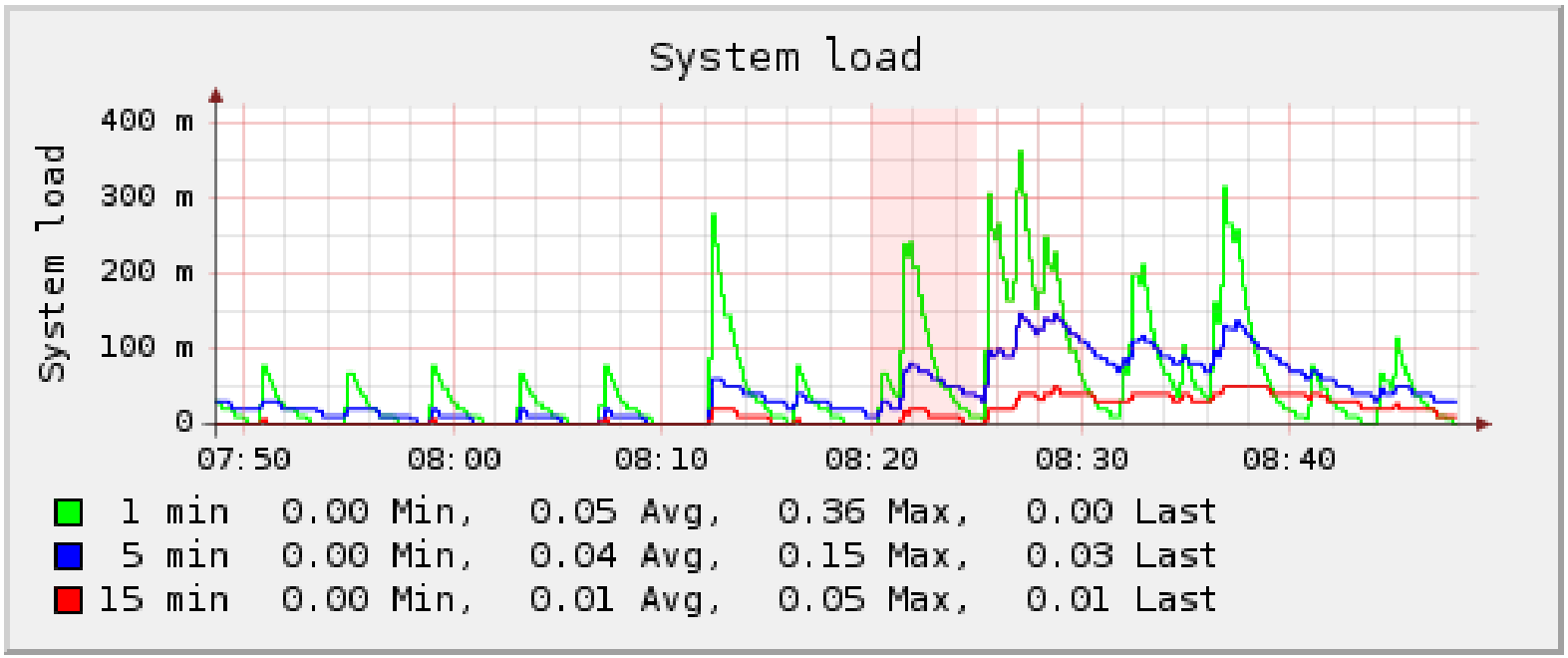


Main features

- 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)

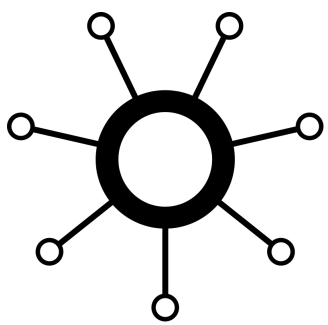


10 seconds resolution



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)

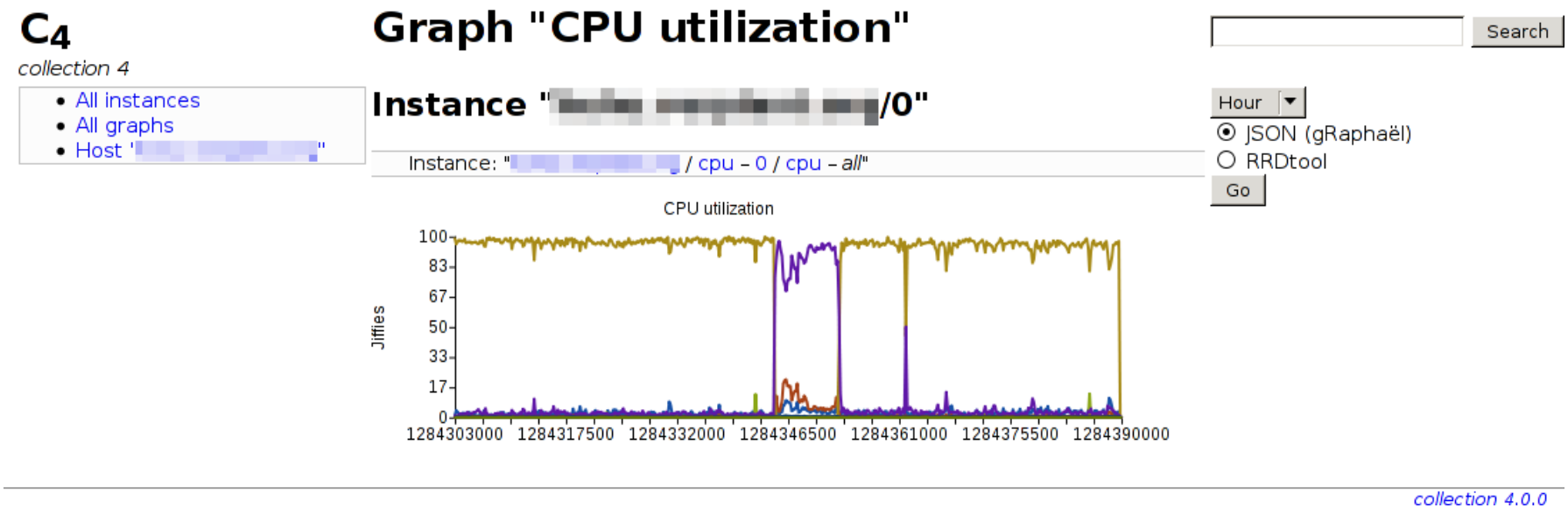
teamix
SOLIDE IT-INFRASTRUKTUR

amqp	apache	apcups	apple_sensors	ascent
battery	bind	conntrack	contextswitch	cpu
cpufreq	csv	curl	curl_json	curl_xml
dbi	df	disk	dns	email
entropy	ethstat	exec	filecount	fscache
GenericJMX.java	gmond	hddtemp	interface	ipmi
iptables	ipvs	irq	java	libvirt
load	logfile	lpar	madwifi	match_empty_counter
match_hashed	match_regex	match_timediff	match_value	mbmon
md	memcachec	memcached	memory	modbus
Monitorus.pm	multimeter	mysql	netapp	netlink
network	nfs	nginx	notify_desktop	notify_email
ntpd	numa	nut	olsrd	onewire
openvpn	OpenVZ.pm	oracle	perl	pinba
ping	postgresql	powerdns	processes	protocols
python	redis	routeros	rrdcached	rrdtool
sensors	serial	snmp	swap	syslog
table	tail	tape	target_notification	target_replace
target_scale	target_set	target_v5upgrade	tcpconns	teamspeak2
ted	thermal	threshold	tokyotyrant	unixsock
uptime	users	uuid	varnish	vmem
vserver	wireless	write_graphite	write_http	write_mongodb
write_redis	xmms	zfs_arc		

collectd setup

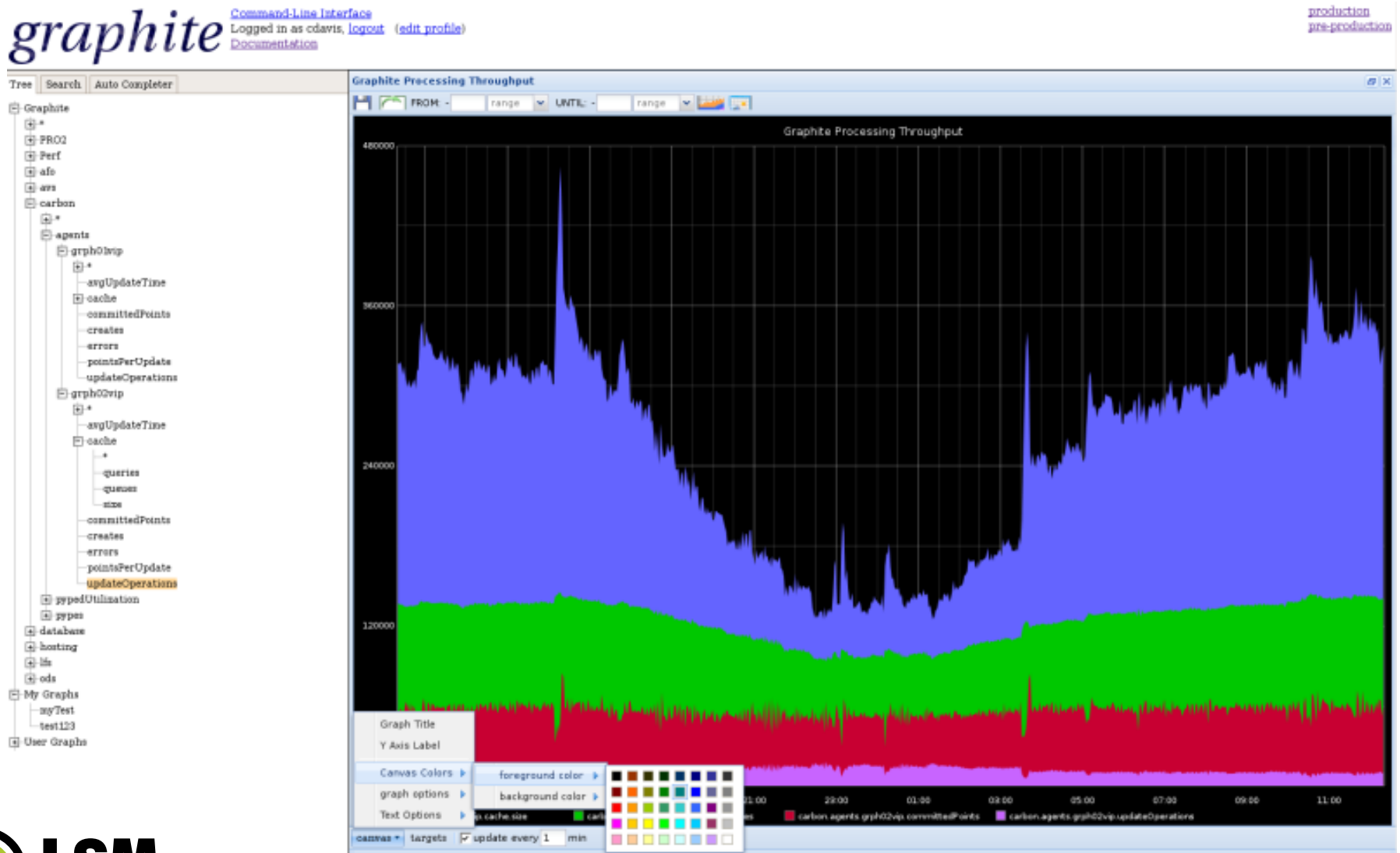
- daemon collects data locally ⇒ runs on every client system (exceptions: SNMP, databases, etc.)
- one or more central servers
- clients push their data to the central servers
- first steps: `install; select plugins; start daemon; enjoy` ;-)

data display: Collection 4 (C4)



- provide collected data through JSON
- different frontends possible
- efficiently handles large amounts of data
- flexible configuration of graphs

data display: Graphite



collectd Overview

Feature Overview

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

Extending collectd

Extras

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

CPU, memory, network I/O

configuration synopsis

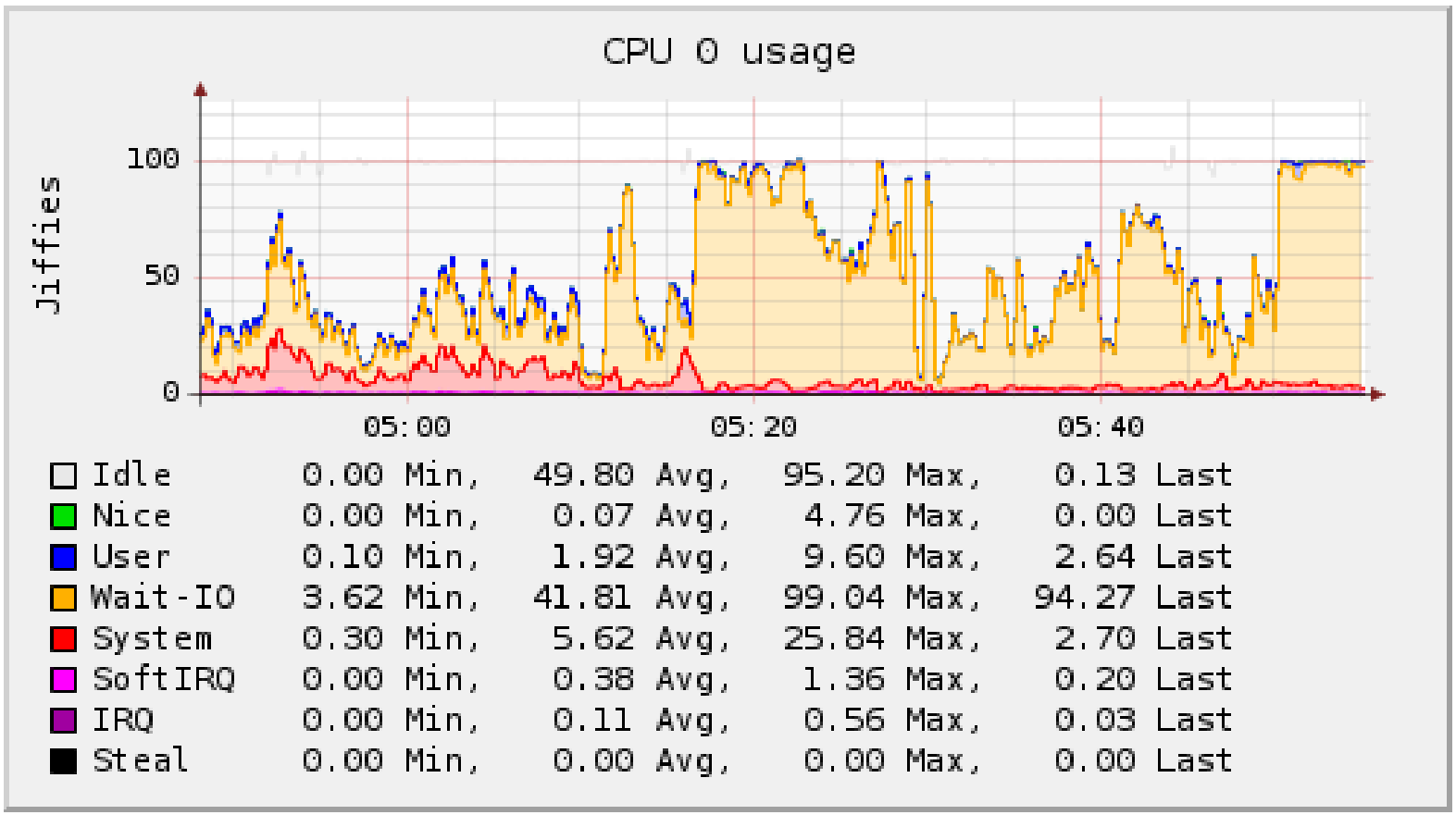
```
LoadPlugin "cpu"
LoadPlugin "memory"
LoadPlugin "interface"
```

CPU, memory, network I/O

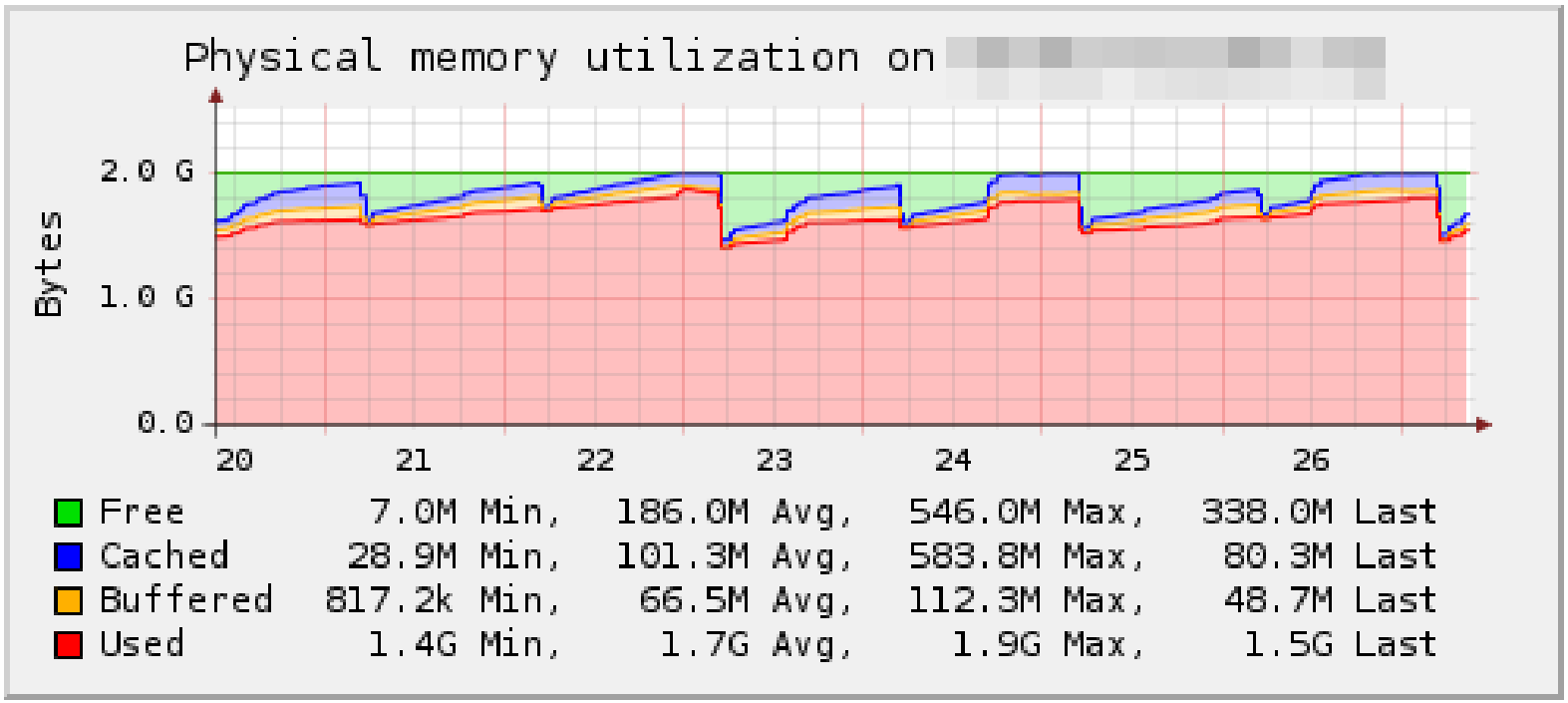
configuration synopsis

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

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

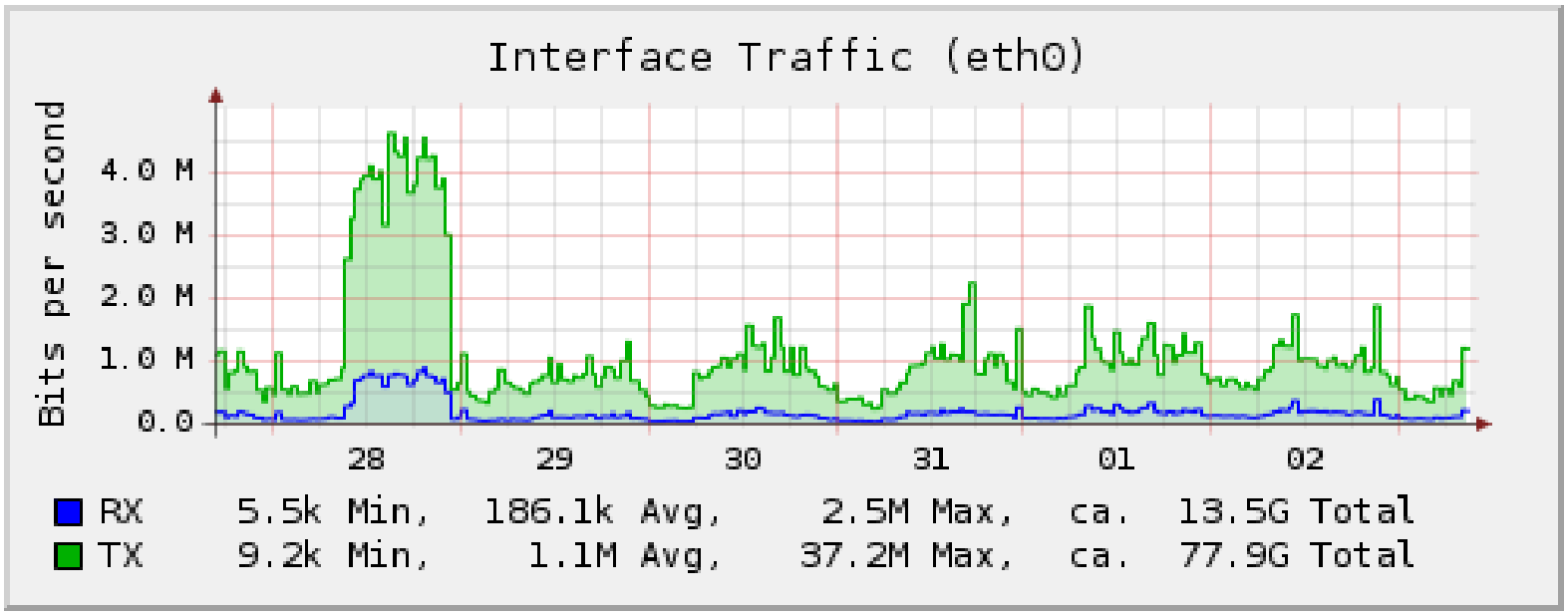
CPU, memory, network I/O



CPU, memory, network I/O



CPU, memory, network I/O



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.

synopsis: client

```
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"

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

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

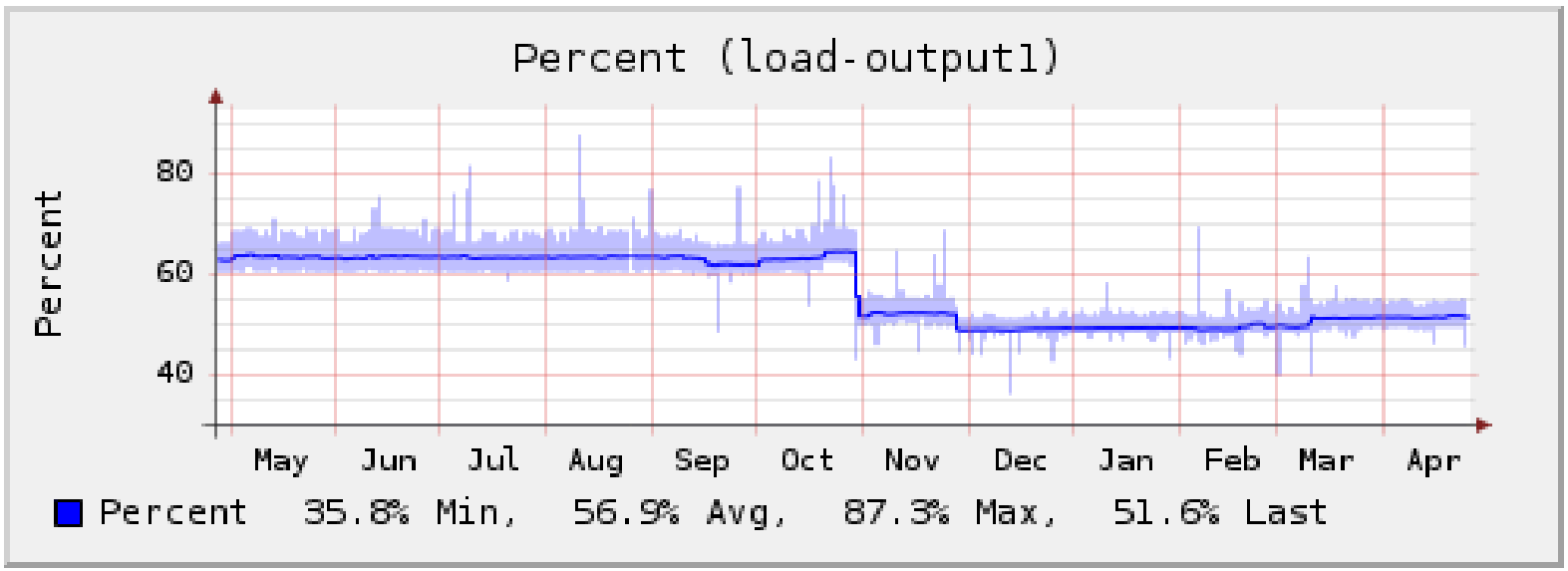
```
<Plugin "rrdtool">
  DataDir "/var/lib/collectd/rrd"

  CacheTimeout 3600   # 1 hour
  CacheFlush 86400    # 1 day

  WritesPerSecond 30
</Plugin>
```

- FLUSH command allows for graphing of current values

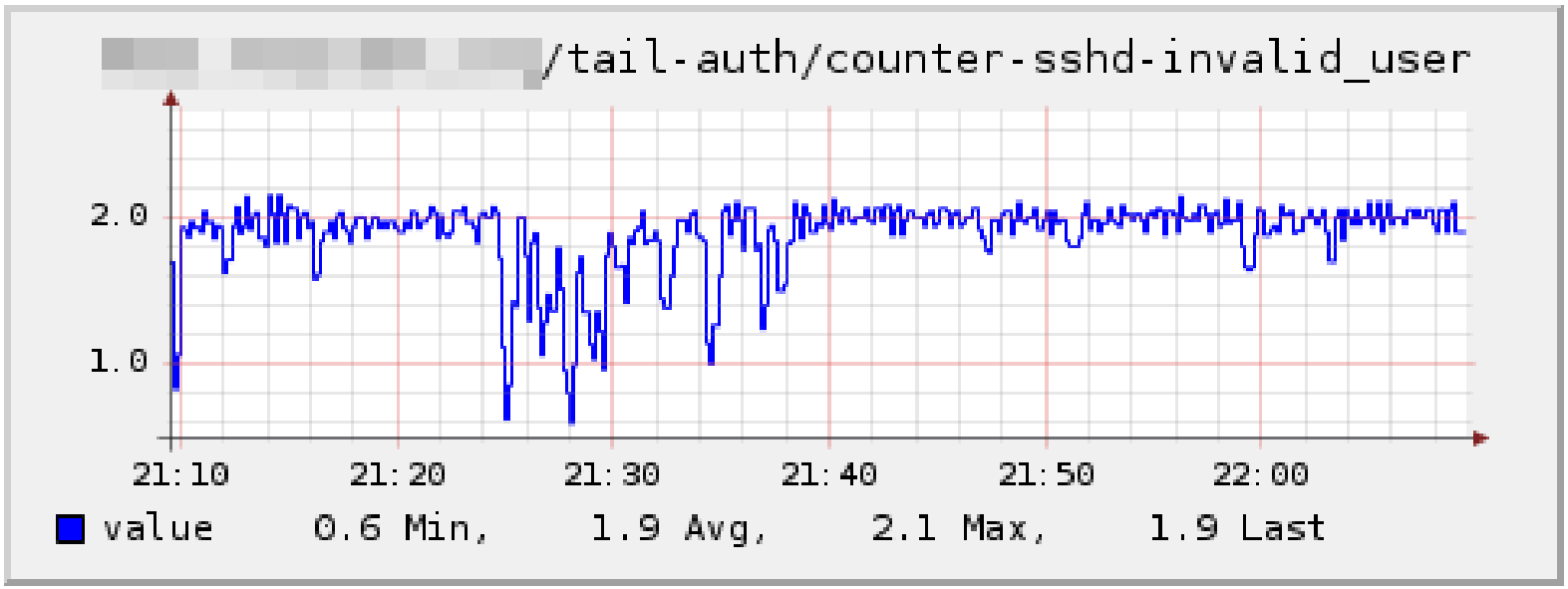
generic plugins (overview)

- idea: generic approaches rather than specialized solutions
- $\rightarrow$ user configuration determines behavior
- $\Rightarrow$ new equipment does not require a new version of **collect**d
- examples: SNMP, tail, curl, DBI, PostgreSQL

SNMP plugin: UPS load



tail plugin: SSH brute-force attack



custom extensions (overview)

- **collect**d API: C, Perl, Python, Java
- external programs may use unixsock or exec plugins

UNIXSOCK plugin

- opens a UNIX socket
- text and line based protocol
  (e. g. `PUTVAL`, `FLUSH`, `LISTVAL`, `GETVAL`)
- query and submit values
- `collectd-nagios`
- `collectdctl` (since version 5.0)
- `cussh.pl`: *“collectd UNIX socket shell”*

UNIXSOCK plugin: example

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

-> | PUTVAL FQDN/users/users 1341851406:42
<- | 0 Success: 1 value has been dispatched.
```

Perl plugin

overview

- integrates a Perl interpreter
  (similar to Apache's `mod_perl`)
- compiles Perl code only once
- exports internal API
  ($\rightarrow$ full flexibility available)

also available

- Java and Python

exec plugin

overview

- executes arbitrary programs / scripts
- parses STDOUT of the program
- respawns process upon termination
  (cf. `init`)

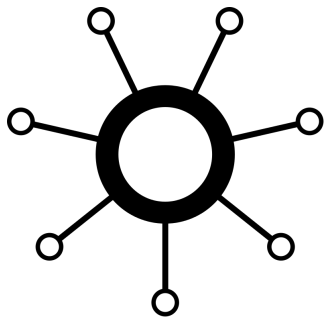
exec plugin: example

```
#! /bin/bash
INTVL=${COLLECTD_INTERVAL:-10}
CHOST="${COLLECTD_HOSTNAME:-localhost}"
IDENT="$CHOST/magic/magic_level"
while sleep $INTVL
do
   V=‘magic --level‘
   echo "PUTVAL \"$IDENT\" interval=$INTVL ‘date +%s‘:$V"
done
```

Future

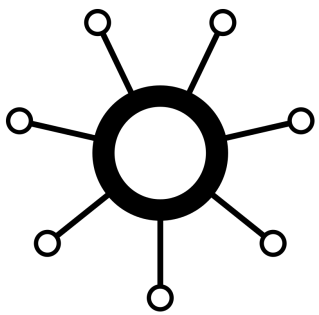
- aggregation of arbitrary values
- per-plugin interval settings
- Nagios support (pushing information to Nagios)
- store data in RMDBSs
- ...

`http://collectd.org/wiki/index.php/Roadmap`



Thank you for your attention!

Questions?



Contact:
Sebastian “tokkee” Harl
teamix GmbH, Nuremberg
<sh@teamix.net>

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

SNMP plugin

overview

- queries network equipment via SNMP
- *generic:* not specialized for any particular hardware
- queries multiple systems in parallel

configuration

- “Data” Block
- “Host” block

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

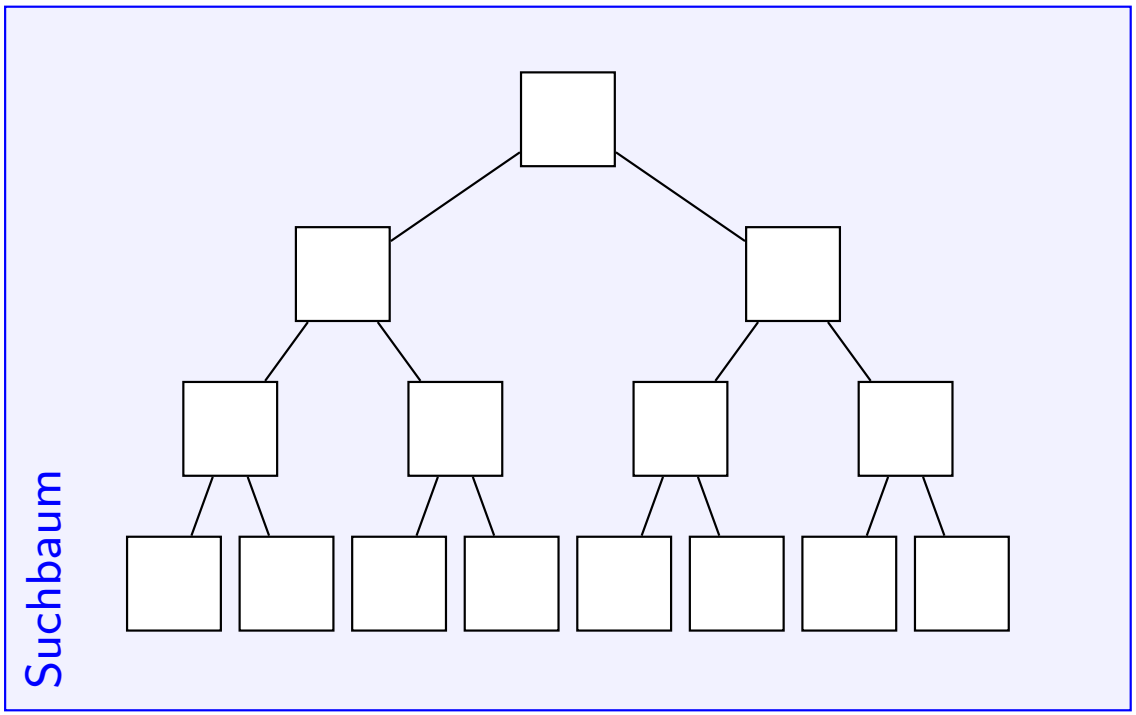
```
<Plugin "snmp">
  <Host "switch0.intern.musterfirma.de">
    Address "10.0.42.2"
    Version 1
    Community "public"
    Collect "ifmib_if_octets64"
    Interval 60
  </Host>
</Plugin>
```

RRDCacheD plugin

overview

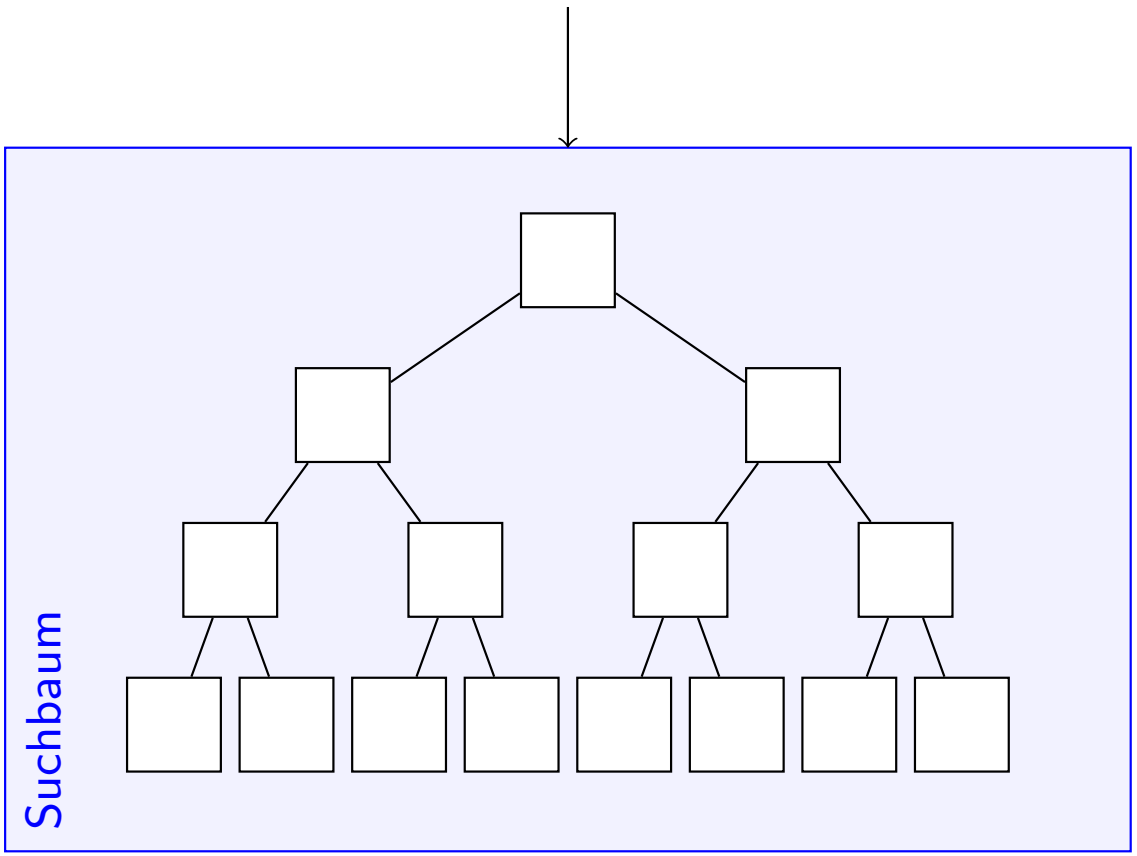
- generalizes caching idea of the rrdtool plugin
- standalone daemon
- integrated into RRDtool (available since version 1.4)
- extended feature set, e. g. journal
- benefit: restart **collect**d without losing cache

RRDCacheD plugin



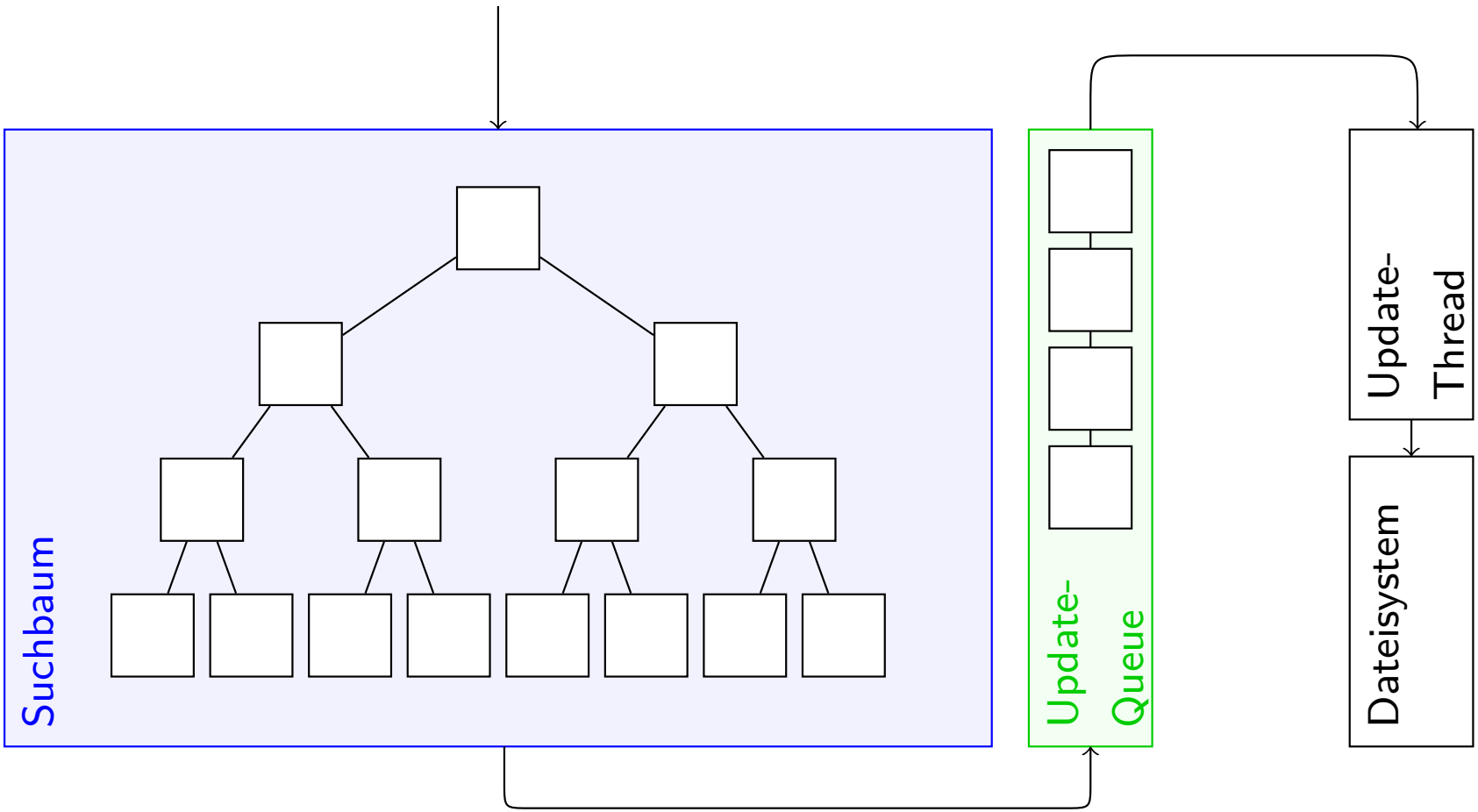
graphic © Florian “octo” Forster

RRDCacheD plugin



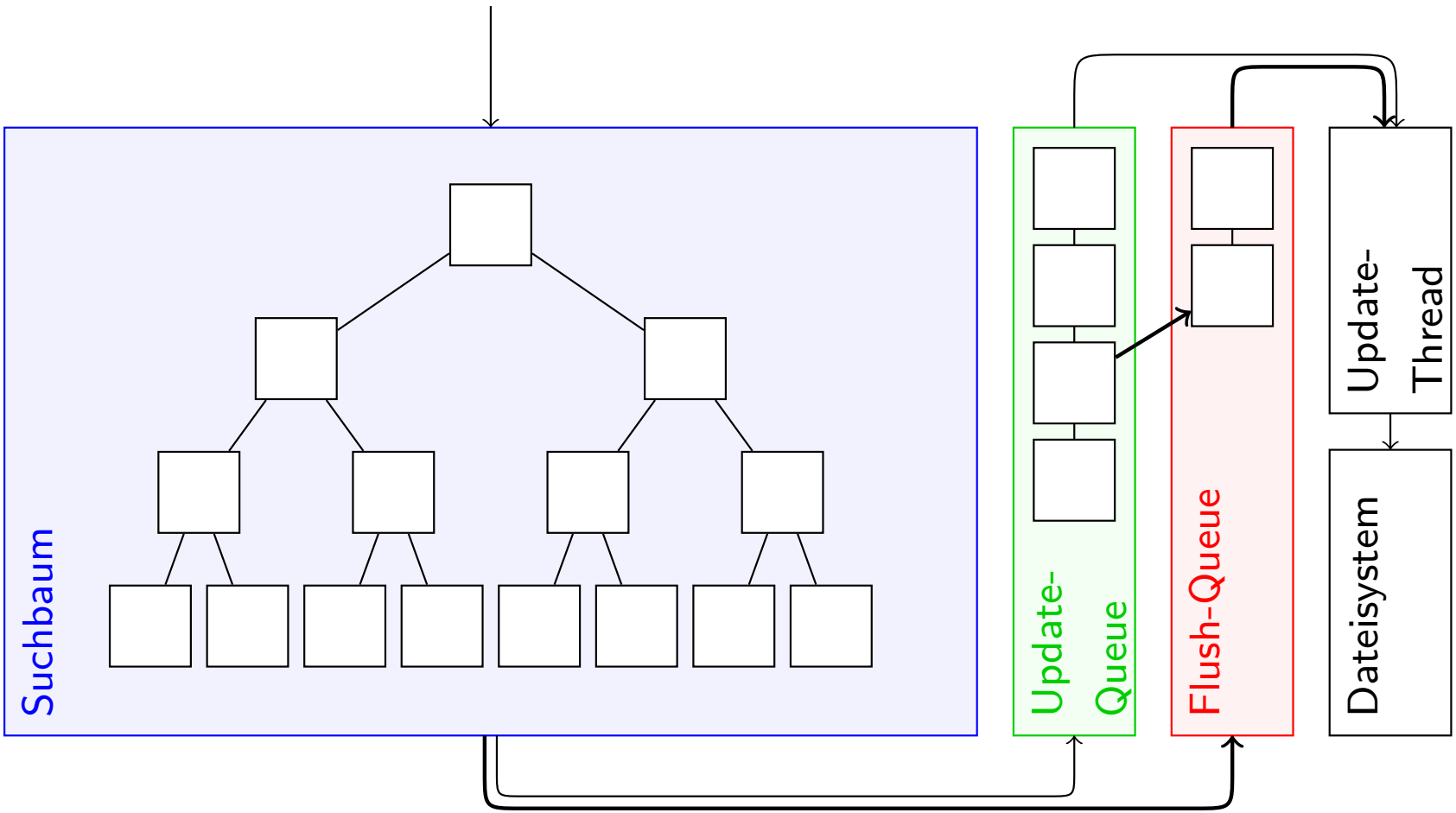
graphic © Florian “octo” Forster

RRDCacheD plugin



graphic © Florian “octo” Forster

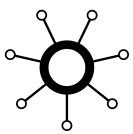
RRDCacheD plugin

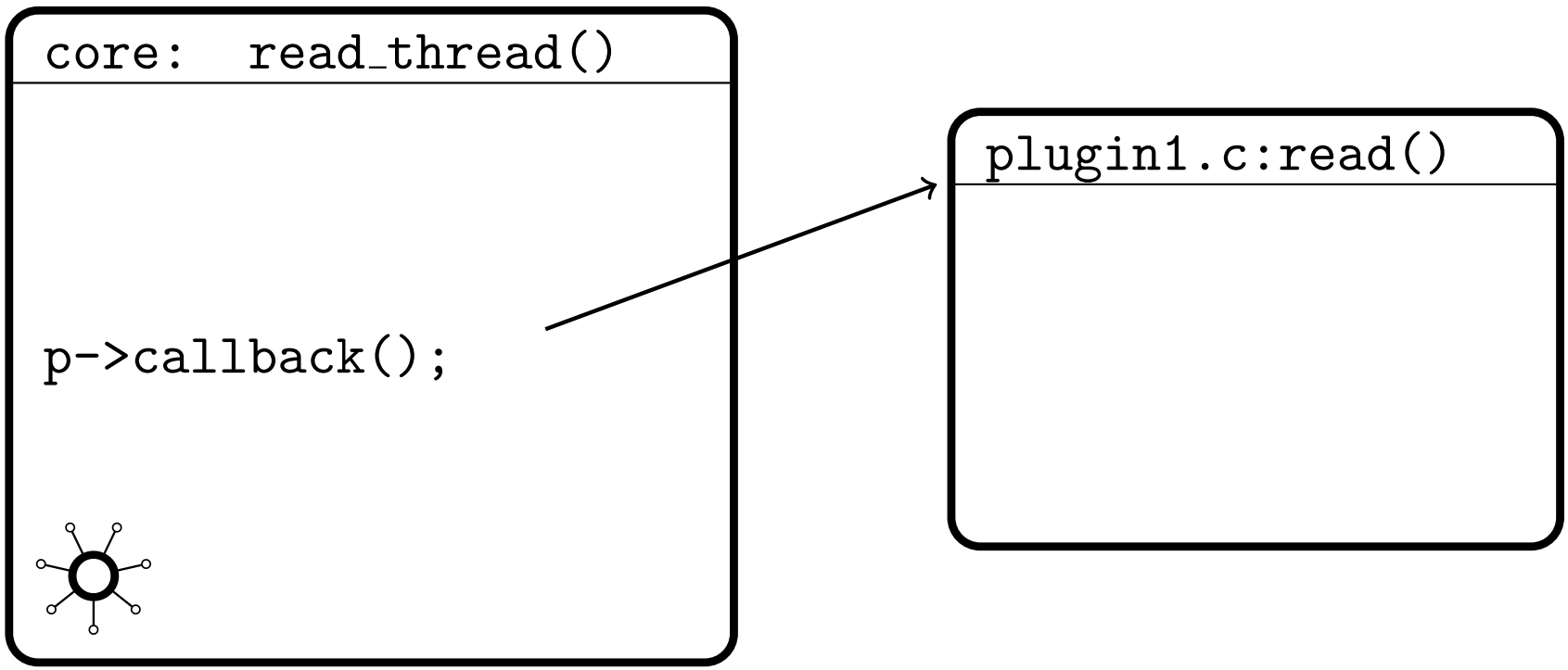


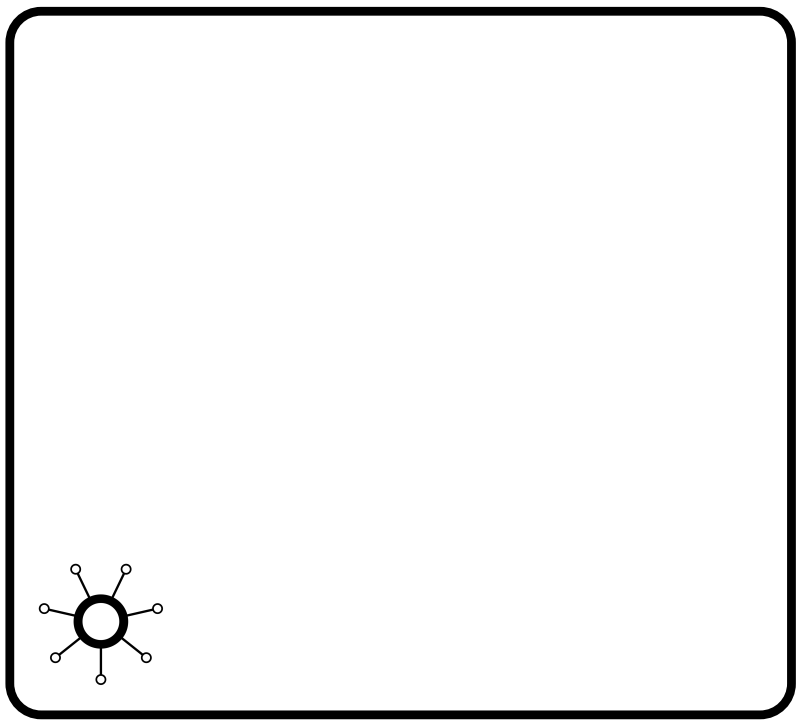
graphic © Florian “octo” Forster

core: read_thread()

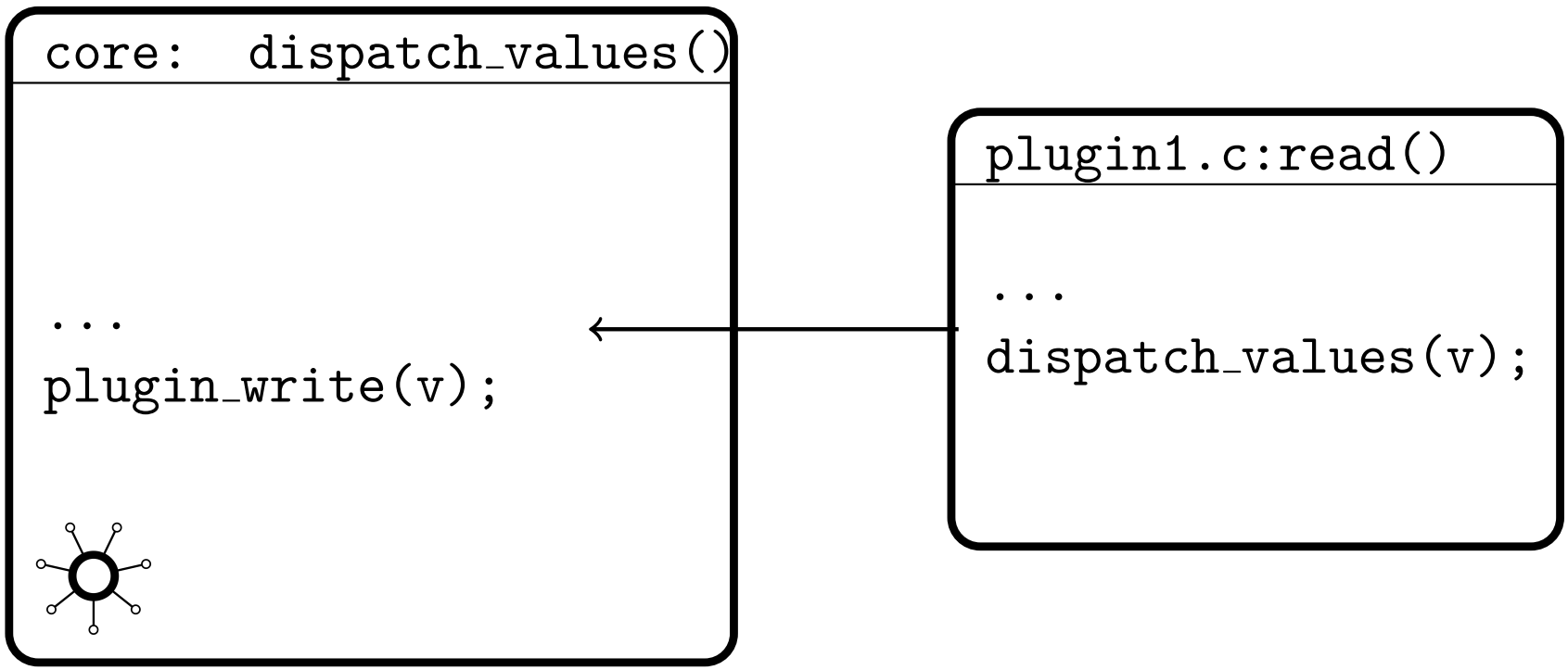
```
p = get_next_plugin();
sleep_until_due(p);
```



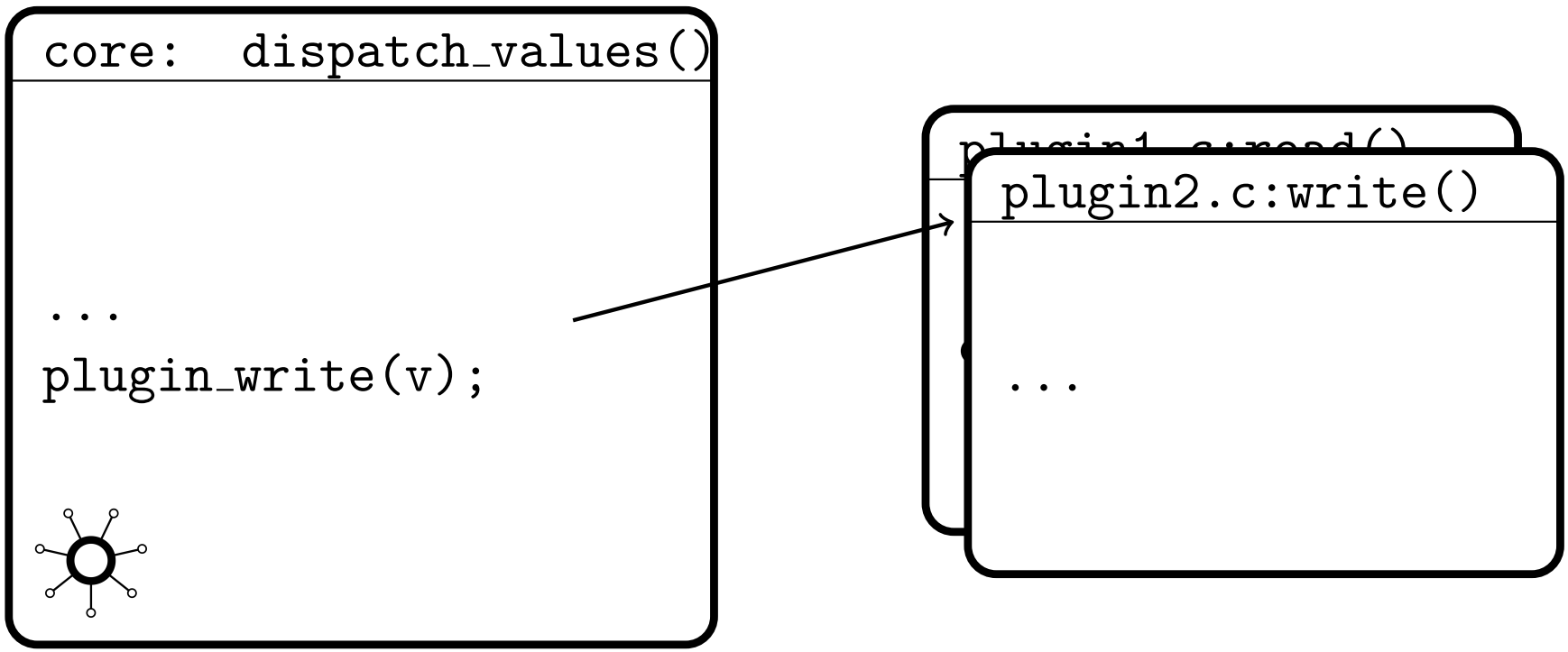




```
plugin1.c:read()

...
dispatch_values(v);
```



collectd internals



Perl plugin: example

```
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');
```