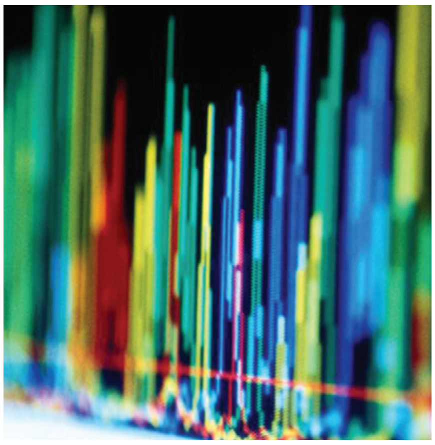
Cacti Web Front-end for RRDtool

Graphical Monitor



RRDtool uses scripts or SNMP to collect arbitrary data and creates histograms. But the tool is extremely difficult to configure – this is where, Cacti, a Web-based front-end can help admins keep an eye on their systems and monitor their current status. BY ACHIM SCHREPFER

Current operating data from a collection of machines is an important source of information for administrators, allowing them to monitor trouble-free operations and help make vital system development decisions. Trends are just as important as the current state. Admins will normally be interested in memory and disk usage, network traffic, and CPU load, or – depending on the application – the number of messages waiting in a queue, the number of server processes or the total number of Web hits per minute.

It is no problem for Linux to collect this data using cronjobs or scripting. However, GUI tools are required for management and evaluation tasks: RRDtool (see the "RRDtool" insert) is a powerful tool that collects and displays data – but it is extremely difficult to configure. Cacti (GPL licensed) is a big help in this case, providing a Web-based front-end for RRDtool (also GPL) to help system administrators to manage and visualize system data.

Round Robin Database

Cacti uses Round Robin databases to store all the relevant data and display this in freely configurable graphs. To collect data, Cacti regularly runs a cronjob to launch scripts and store their output. If the script collection supplied with the tool does not provide the functionality you need, you can of course write your own scripts and add them to the program. Single line shell scripts are normally all you need.

The Web-based GUI is also used to configure the graphs. There is even a function for organizing graphs in a tree view, and this could be a real advantage when monitoring multiple systems. Integrated SNMP support rounds off the Cacti package displaying the traffic from network interfaces on SNMP aware devices.

Cacti takes the work out of those daily chores, but it might take you a while to get used to the tool, as the sheer bulk of functions and forms is not immediately intuitive. Before you can view your first graph, you will need to complete fairly extensive configuration tasks. The following article shows an example of the steps required to monitor the number of arbitrary processes on a single machine, using a single data collection script.

Out of the box

After setting up the cronjob (see the "Installing Cacti" insert), the software immediately starts collecting data. Values for the individual data sources are stored in the corresponding Round Robin archives every five minutes. Ten minutes after setting up the cronjob, you should be able to click on the *Graphs* tab (at the top of the page) to display an overview of the default graphs (see Figure 1). Cacti needs at least two values to calculate an average.

The standard scripts, data sources and graphs cover quite a wide range of system data including CPU load, memory usage, and many other resources. These scripts are not rocket science, and mainly use existing system tools such as *uptime* reducing the output to a single figure in most cases.

It is quite simple to add more data sources to the system ranging from the number of DNS queries per second to the temperature and humidity at your chosen vacation spot. Scripts are required to return a numerical value when called, as Cacti needs to plot a graph. This level abstraction means that the system is equally at home in fields of application such as measuring and monitoring, or even collecting meteorological data.

But take care when programming, as the system will parse the complete output from your script, and wrapped lines can lead to faulty stats being displayed. Ensure that your Cacti scripts output only

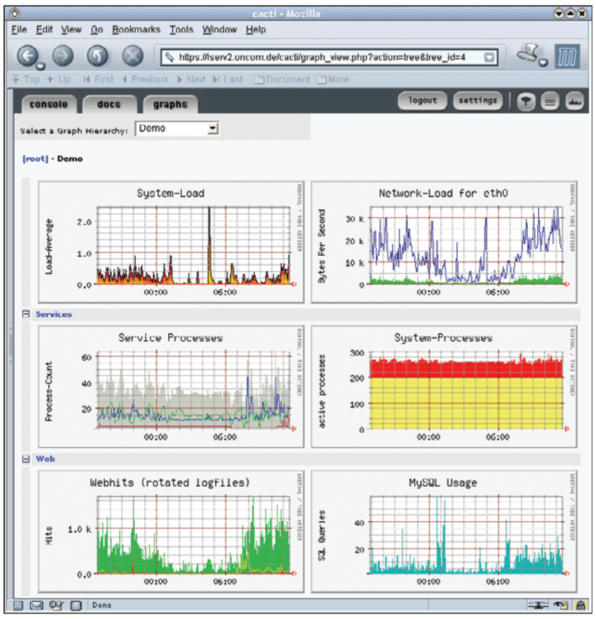
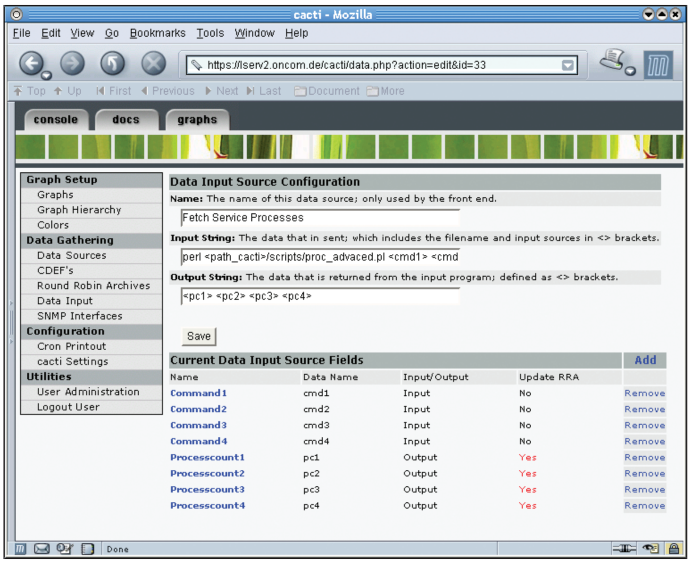


Figure 1: In the overview Cacti displays any graphs you have configured as thumbnails. Clicking a thumbnail opens the corresponding graph

numbers and filter out any wrapped lines.

You can use the Perl script in Listing 1 to enumerate processes. The script will reside in the *scripts* directory. It expects one or more command names as parameters and outputs the number of processes for this command in the same order. The following call enumerates the Apache and MySQL processes:

```
./proccount.pl /usr/↩
sbin/apache /usr/↩
sbin/mysqld
34 18
```



https://lserv2.oncom.de/cacti/data.php?action=edit&id=33

Data Input Source Configuration

Name: The name of this data source; only used by the front end.

Fetch Service Processes

Input String: The data that in sent; which includes the filename and input sources in <> brackets.

perl <path_cacti>/scripts/proc_advaced.pl <cmd1> <cmd

Output String: The data that is returned from the input program; defined as <> brackets.

<pc1> <pc2> <pc3> <pc4>

Current Data Input Source Fields

Name	Data Name	Input/Output	Update RRA	
Command1	cmd1	Input	No	Remove
Command2	cmd2	Input	No	Remove
Command3	cmd3	Input	No	Remove
Command4	cmd4	Input	No	Remove
Processcount1	pc1	Output	Yes	Remove
Processcount2	pc2	Output	Yes	Remove
Processcount3	pc3	Output	Yes	Remove
Processcount4	pc4	Output	Yes	Remove

Figure 2:*Data Input Source Configuration* allows you to specify script parameters and return values: a field must be created for each value

A total of 34 Apache and 18 MySQL processes are registered in the process list. These figures can vary and are thus ideal candidates for a time/value visualization using the Cacti tool.

Registering Scripts

Before you can use a new script, you have to register it first: To do so, select *Data Gathering* | *Data Input* (the box on the left margin) in the Cacti console (the *Console* tab at the top of the screen). The *Add* link (top right) calls the form for new scripts (see Figure 2). Cacti prompts your for an internal name; make sure that this name identifies the task performed by the script, such as *Count Processes*. The *Input String* expects the total command syntax for the script and wildcards for the arguments it needs. Wildcards must be enclosed in angled brackets and should contain only numbers and letters:

```
perl /var/www/cacti/scripts↩
/proccount.pl <cmd1> <cmd2> ↩
<cmd3> <cmd4>
```

Cacti will expect values for the wildcards, as it needs to insert them when calling the script. Cacti also needs to know the output format for the script – process numbers for each command, separated by space characters, in this case. This is configured in the *Output String* field, where wildcards are used to represent the output format:

```
<pc1> <pc2> <pc3> <pc4>
```

You can specify more than four arguments and return values, although most applications do not require an argument and can make do with a single return value (such as the CPU temperature). The form re-appears after saving your changes, but now the (currently empty) list *Current Data Input Source Fields* appears at the end.

Each wildcard in the input or output string needs an entry in this list to allow Cacti to work with the corresponding parameters and return values. Unfortunately, the system does not do this automatically; so admins can look forward to a round of clicking. Enter an intuitive name for the field in *Name*; this could be *Process number 1* for example. *Data Name* contains the exact name of the wildcard, enclosed in two angled brackets. The *Input/Output Field* pull-down is used to specify the parameter type, that is an input or output parameter. The checkmark in the *Field Used to Update RRA* checkbox tells Cacti to save the output

RRDtool

The RRD acronym means Round Robin Database. Authored by Tobi Oetiker, RRDtool is an extremely useful program that organizes data in RRDs. But if you select a sampling rate of one minute (that is one record per minute), the volume of data in an RRD would be immense, if you wanted to store a year's worth of data. This is why RRDtool selects a higher sampling rate for newer data than older.

This provides for a fixed database size with very quick access times. You can select arbitrary periods and rates in so-called RRAs (Round Robin Archives). The Cacti system provides four pre-configured RRAs by default, and the defaults are probably okay for most applications. RRDtool automatically overwrites older data when the database approaches its size limit.

Besides providing data storage and archive management facilities, RRDtool is also capable of visualizing any data stored in the archive files. generating highly-configurable graphs to do so. But you can also access individual records, or export the whole archive to an XML file. The RRDtool homepage [1] contains a number of tutorials on this topic. Of course, you can download the program here, and also access a good collection of links to other tools for RRDtool.

New Cacti Version

Ian Berry, Cacti's author, continues to enhance the system, and version 0.6.8a, which we used for the examples in this article, has now been replaced by version 0.8.1. Administrators working on large server farms can expect to benefit most from the new features.

The system now focuses on SNMP support with templates for graphs, data sources and hosts providing centralized configuration. This can save admins a lot of time when setting up a number of identical machines. SNMP data is collected by means of data Queries for processing. The GUI has grown – but despite its complex functionality, it is easy to use.

Cacti 0.6.8a is still available for downloading, and it has shown itself to be stable in the course of time. Production environments might prefer to install both versions in parallel. The new architecture of the 0.8.x series means that there is still some teething trouble to be ironed out. Newer Cacti versions also provide an import feature for migrating 0.6 databases to the new structure. This prevents data loss on updating.

parameter in a Round Robin archive (RRA).

After completing this procedure we needed a insert list with a total of eight fields below the form in our example. This script is now available within the Cacti tool, and ready for use with arbitrary data sources and various parameters.

The Data Source

A data source must be configured to allow Cacti to initiate the data collection script. The admin user defines a data source, at the same time specifying script arguments and data storage parameters.

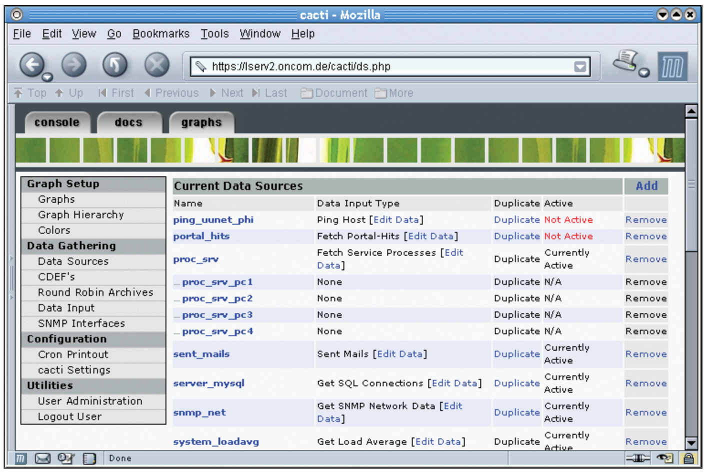
The *Data Gathering | Data Sources* menu item provides an overview of known data sources (see Figure 2). The *Add* link opens the form for new data sources. Data source names should comprise only letters, numbers and underscores as Cacti uses them to create filenames. In our example the data source is called *proccount*. Cacti fills the following fields with suitable values. The *Data Source Type* list field specifies the data type,

- ABSOLUTE
- COUNTER
- DERIVE
- GAUGE

where *GAUGE* is the correct setting for the process counter, as the number of processes fluctuates within a specific range – about 0 through 300. Refer to the RRD Tutorial [2] for explanations of the other data types and how Cacti interprets them.

The *Data Input Type* field contains a selectable list of registered scripts. We will be using the new *Process Count* script in our example. The *Associated RRAs* selection box contains a list of defined Round Robin archives. There is nothing to stop you selecting all the archives and storing a lot of data. Cacti even allows you to define your own archives with different intervals.

The minimum and maximum value fields should be filled with plausible values for each application area. Finally, remember to check *Update .rrd File* and save all your changes.


https://lserv2.oncom.de/cacti/ds.php

console | docs | graphs

Graph Setup
- Graphs
- Graph Hierarchy
- Colors

Data Gathering
- Data Sources
- CDEF's
- Round Robin Archives
- Data Input
- SNMP Interfaces

Configuration
- Cron Printout
- cacti Settings

Utilities
- User Administration
- Logout User

Current Data Sources Add

Name	Data Input Type	Duplicate	Active	
ping_uunet_phi	Ping Host [Edit Data]	Duplicate	Not Active	Remove
portal_hits	Fetch Portal-Hits [Edit Data]	Duplicate	Not Active	Remove
proc_srv	Fetch Service Processes [Edit Data]	Duplicate	Currently Active	Remove
_ proc_srv_pc1	None	Duplicate	N/A	Remove
_ proc_srv_pc2	None	Duplicate	N/A	Remove
_ proc_srv_pc3	None	Duplicate	N/A	Remove
_ proc_srv_pc4	None	Duplicate	N/A	Remove
sent_mails	Sent Mails [Edit Data]	Duplicate	Currently Active	Remove
server_mysql	Get SQL Connections [Edit Data]	Duplicate	Currently Active	Remove
snmp_net	Get SNMP Network Data [Edit Data]	Duplicate	Currently Active	Remove
system_loadavg	Get Load Average [Edit Data]	Duplicate	Currently Active	Remove

Done

Figure 3: The *Current Data Sources* table lists the configured data sources. Cacti automatically generates sub-entries for scripts with multiple return values

The *Current Data Sources* table (Figure 3) now contains five new entries: one called *proccount* with four others attached – the number will depend on the output fields defined in *Data Input Sources*. The *Edit Data* link in the line with the main entry opens a form that finally allows you to pass arguments to the script we discussed previously.

You will need an argument like */usr/sbin/apache*, or similar, to ascertain the number of Apache processes. You can monitor up to four programs in this way. This approach – which may seem somewhat complicated initially – allows you to apply scripts you have registered to various data sources with various arguments.

Testing the Source

The *Configuration | Cron Printout* menu item allows you to test any scripts that you have registered and assigned as data sources. A table with script commands including all their arguments is initially shown. Clicking on the *Show Output* button tells Cacti to perform a trial run of all commands and display the return values, but without storing any data.

The next step takes us to the *Graph Setup | Graphs* menu, where an overview of previously configured graphs is shown. The *Add* link allows us to open the form for a new graph, and again Cacti prompts you first for a name. The new name is used to label the graph. You can usually use the defaults for all the other settings.

After completion and saving, you are returned to the overview. The *Edit Graph Items* link in each line takes you to the *Graph Items* overview, where you can add lines and other elements to the graph. You can freely define the number and type of elements, and leave Cacti to take care of the annoying details like the scale and labelling that would otherwise prove to be very time comsuming and quickly become a chore.

Listing 1: Enumerating Processes

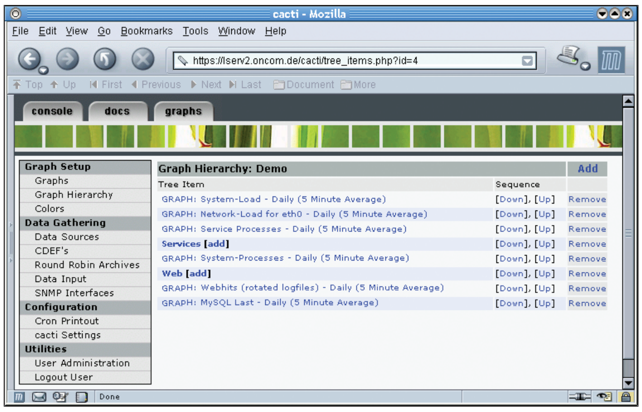
```
#!/usr/bin/perl -w
#
# proccount.pl - enumerates the
number of processes
# for arbitrary commands
(supplied as parameters).
# Current command list:
#   "ps -eo cmd|awk '{print ➥
$1}'|sort|uniq"
use strict;
# cancel if no arguments exist
if (!@ARGV) { exit 1; }
my %command_counter;
# initialize a counter for each
command
foreach (@ARGV) {
    $command_counter{$_} = 0;
}
# Go through process list and
increment
# command count if necessary
foreach (split(/\n/,`ps -eo ➥
cmd|awk '{print \$1}'`)) {
    if (
exists($command_counter{$_}) ) {
        $command_counter{$_}++;
    }
}
# Write the values to an array
# observing the correct order
my @values;
foreach (@ARGV) {
    push @values, $command_➥
counter{$_};
}
# Output
print join(' ',@values);
```

Adding a Line

The *Add* link in the *Graph Items* overview creates a new element for the graph. The *Data Source* list field links the new element with a data source. The data source for the first return value in our sample script is called *proccount_pc1*. This is followed by a color selection which you can modify in the *Graph Setup* | *Colors* menu, if required.

It makes sense to start off with *LINE1* as your original *Graph Item Type* – this is a single pixel line. The default setting is fine for all the other parameters. You can add a legend in the *Text Format* field. The field also accepts wildcards in angled brackets: Cacti will replace the *<pc1>* string in the graph with the current value of this variable.

After saving your changes, you are returned to the element overview. The Link *Turn Graph Preview On* link enables a preview of the graphs. If the data source has been enabled for a while, you should see some values in the graph. If there is no data available, Cacti will not create a preview, even though you enable the preview option.



Graph Hierarchy: Demo | Add

Tree Item	Sequence	
GRAPH: System-Load - Daily (5 Minute Average)	[Down], [Up]	Remove
GRAPH: Network-Load for eth0 - Daily (5 Minute Average)	[Down], [Up]	Remove
GRAPH: Service Processes - Daily (5 Minute Average)	[Down], [Up]	Remove
Services [add]	[Down], [Up]	Remove
GRAPH: System-Processes - Daily (5 Minute Average)	[Down], [Up]	Remove
Web [add]	[Down], [Up]	Remove
GRAPH: Webhits (rotated logfiles) - Daily (5 Minute Average)	[Down], [Up]	Remove
GRAPH: MySQL Last - Daily (5 Minute Average)	[Down], [Up]	Remove

Figure 4: Cacti supports any number of graph hierarchies. You can configure them separately to provide customized views

You can now add more elements to the graph; a line with custom settings for each process being monitored. The system also allows you to stack lines (Item Type: *STACK*). You can also use CDEF (Calculation with DEFs) functions to perform calculations on the values for each line and thus integrate totals. A comprehensive CDEF Tutorial is available from [3].

New Graphs, Please

The graph created in our example, will not show up in the overview in the *Graphs* tab at present. This is because you have to add the graph to the hierarchy. The *Graph Setup* | *Graph Hierarchy* (see Figure 4) allows you to create any number of trees (hierarchies). If you are monitoring multiple servers from a central host, you will come to appreciate this setup in time.

Cacti defines a single hierarchy by default. Clicking on *Edit Current Graph Hierarchy* opens a tree structure containing the graphs for the current hierarchy. The *Add* link creates a new entry at the top level. To add a new graph, simply select the *This Item is a Graph* option and then select the required graph from the list.

The graph we just added to the hierarchy now appears at the end of the list and can be moved using the *up* and *down* links (see Figure 4). To insert a subdivision, simply select the *This Item is a Heading* option and type a descriptive text (such as *System data* or *Network data*). This tells Cacti to create a folder which in turn can store elements (graphs or headings).

The *add* links next to the folder names create new elements within folders. Folders can be opened or closed within the graph view, and this is a big advantage

Installing Cacti

RRDtool is a pre-requisite for Cacti. The RRDtool program package is included by default in many distributions. If this is not true of your distribution, you can download the sources from [4]. Follow the typical steps to compile and install:

```
./configure
make
make install
```

This will install RRDtool below */usr/local/rrdtool*-Version. Cacti also assumes the availability of Web server with PHP and MySQL support. After downloading the current version from [5], unpack the tarball in a PHP-capable directory, such as */var/www/*. You then need a MySQL database with the Cacti tables:

```
mysqladmin create cacti
mysql cacti < cacti.sql
```

The *cacti.sql* file is also located in the main directory of the Cacti distribution. The next step is to create a MySQL user:

```
mysql -p --user=root mysql
mysql> GRANT ALL ON cacti.*
  TO Cacti-User@localhost
  IDENTIFIED BY 'Password';
mysql> flush privileges;
mysql> exit
```

You also need to add the database details (hostname, database name, username and password) to *include/config.php*.

Unfortunately, the command script that is run as a cronjob every five minutes is written in PHP, and assumes that you have installed the PHP binary. The following entry in */etc/crontab* will launch the cronjob every five minutes:

```
*/5 * * * * Cacti-User php ⮐
/var/www/cacti/cmd.php >
/dev/null 2>&1
```

This assumes the existence of the Cacti-User on your system. As an alternative, you could use a program like *wget* to launch the script via the local Web server, thus avoiding the extra cost of installing the PHP binary; but this would run the script with the privileges of the Web server. Wget requires the following cron table entry:

```
*/5 * * * * wget -O -
http://localhost/⮐
cacti/cmd.php > /dev/null 2>&1
```

The third option is a Perl script available from [6]; the script assumes the role of *cmd.php*. Depending on the variant you choose, you may need to assign write privileges to the user account that runs the command script to the *rra/* and *log/* directories.

Finally, the admin should ensure that the *register_globals* and *register_argc_argv* options are enabled in *php.ini*. This is the default for older PHP versions, but newer versions disable this setting for security reasons – unfortunately, Cacti cannot do without these options.

Cacti should be ready for work now, you can use your Web browser to open *http://localhost/cacti/index.php*. When you log on for the first time, your username and password will default to *admin* – Cacti will immediately prompt you to enter a new password before displaying the main screen.

in terms of readability, as you will not normally need the whole range of data on view at the same time.

MRTG Deluxe

If your toolbox includes MRTG (Multi Router Traffic Grapher, [7]), you can consider replacing it with Cacti. The *Data Gathering | SNMP Interfaces* menu allows you to incorporate network interfaces as data sources, in a similar way to MRTG, and to create histograms to match.

The *Add* link opens a form that expects a host name and an SNMP community string. Cacti will then search for network interfaces on the specified device and finally display an entry in the *Current SNMP Hosts* table. Admins can use *View Gathered Interfaces* to check for interfaces that Cacti has discovered at the specified address. To create a graph, you can then simply click on *Make Graph* in the line that interests you, and finally store the form after completing your changes.

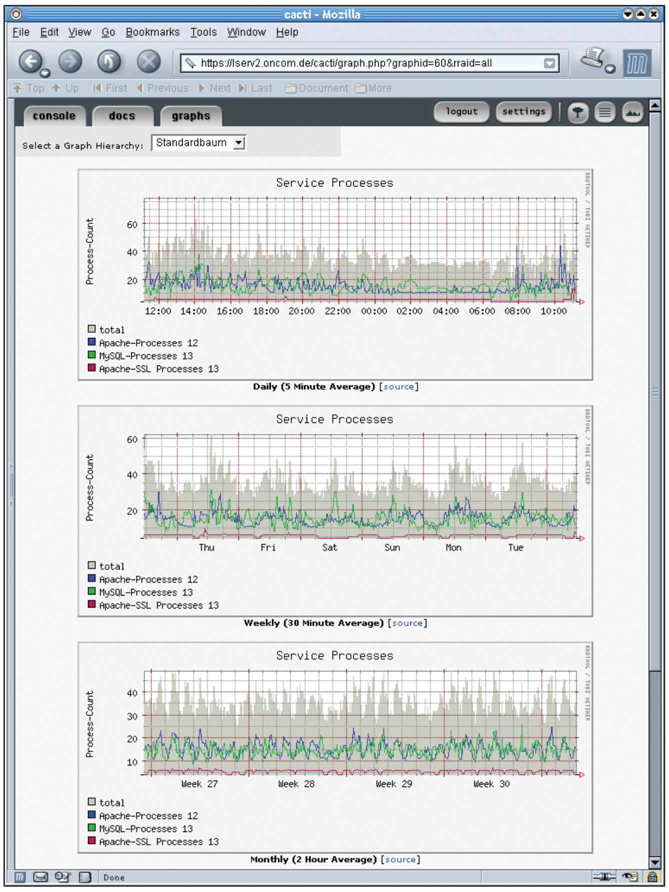

cacti - Mozilla
File Edit View Go Bookmarks Tools Window Help
https://lserv2.oncom.de/cacti/graph.php?graphid=60&rraid=all
console docs graphs
logout settings
Select a Graph Hierarchy: Standardbaum
Service Processes
Process-Count
total
Apache-Processes 12
MySQL-Processes 13
Apache-SSL Processes 13
Daily (5 Minute Average) [source]
Weekly (30 Minute Average) [source]
Monthly (2 Hour Average) [source]
Done

Figure 5: The detailed view displays four graphs with different sampling rates by default. This allows you to monitor current processes over an extended period of time

The *Make all Graphs* link (at the top) is useful in environments with a lot of hosts, as it creates graphs for all the interfaces Cacti has discovered simultaneously. You can then add the graphs to the hierarchy as previously described.

These functions assume that the SNMP tools *snmpget* and *snmpwalk* have been installed on the machine. You can use the *Cacti Settings* menu to modify the paths to these programs.

User Management

You can use the *Utilities | User Administration* item to create user accounts and define privileges. There are even access control facilities that allow you to specify which user has access from what IP address. Privileges define the functions a user can access, for example, whether a user is permitted to create new data sources or set up graphs. Also, the admin user can hide individual graphs or complete hierarchies from specific users.

The system offers enough leeway to provide shared access to data within smaller teams. An Internet Service Provider could use these features to provide its customers access to Cacti graphs for a hosted server. Users with appropriate privileges could even change the graph settings to meet their individual requirements. The admin user can allow users to change the thumbnail size, the number of graphs per line and the default hierarchy.

The *Configuration | Cacti Settings* allows you to configure the paths to the various programs and for logging. The admin user can enable the export function in this menu. This prevents the system from creating new graphs on client access; instead it waits for a script to run, and stores the graphs as files on disk. This function is useful if Cacti is accessed by a larger number of users.

Cacti enables the export function if a path is defined in the *HTML Export Path* field. However, creating a complete set of graphs every five minutes is quite demanding on a system, so you might like to use the *Export Every x Times* option to modify this setting. A value of 12 in this field would tell Cacti to create graphs every 12·5 = 60 minutes.

Conclusion

Staying logged in on every server, and making sure everything is working okay, is a kind of hobby for some administrators. Another species, the console fans, loves rooting through logfiles and creating statistics from them. Any admins who use tools like Cacti and RRDtool can take things easy. Line graphs plotted over the course of a few minutes, hours, or even days provide an at-a-glance overview of the operative status of their machines.

Although Cacti may be difficult to handle originally, it does provide a convenient and well-thought-out GUI for visualizing statistical data against a time axis. ■

THE AUTHOR

Achim Schrepfer develops CMS and database solutions and manages Linux Web servers for a local Internet Service Provider.

INFO

[1] RRDtool Homepage: *http://www.rrdtool.org/*

[2] RRD Tutorial: *http://www.rrdtool.org/tutorial/rrdtutorial.html*

[3] CDEF Tutorial: *http://www.rrdtool.org/tutorial/cdeftutorial.html*

[4] Download RRDtool: *http://www.rrdtool.org/download.html*

[5] Download Cacti: *http://www.raxnet.net/downloads.php*

[6] Alternative script for *cmd.php*: *http://www.raxnet.net/products/cacti/additional_scripts.php*

[7] MRTG: *http://www.mrtg.org*