Cricket FAQ

Sasha Mikheev

1. General Questions

1.1. What is cricket ?

Cricket is a high performance, extremely flexible system for monitoring trends in time-series data. Cricket was expressly developed to help network managers visualize and understand the traffic on their networks, but it can be used all kinds of other jobs, as well.

Cricket has two components, a collector and a grapher. The collector runs from cron every 5 minutes (or at a different rate, if you want), and stores data into a data structure managed by RRD Tool. Later, when you want to check on the data you have collected, you can use a web-based interface to view graphs of the data.

1.2. How does Cricket differ from MRTG ?

Short answer: in more ways than meet the eye. The most noticable are that Cricket has built in magic to look up and verify SNMP instances (MRTG only verifies instances). Cricket does away with creating the images every five minutes, which should allow your management station to monitor more datasources with fewer resources. Cricket also has built in threshold monitoring.

In practice, MRTG has involved since Cricket began, and other alternatives appeared. Choosing between network monitoring tools requires more decisions than this FAQ can deal with.

1.3. What is rrdtool ?

RRD is the Acronym for Round Robin Database. RRD is system to store and display time-series data (i.e. network bandwidth, machine-room temperature, server load average). It stores the data in a very compact way that will not expand over time, and it presents useful graphs by processing the data to enforce a certain data density. It can be used either via simple wrapper scripts (from shell or Perl) or via frontends that poll network devices and put a friendly user interface on it. Check out http://www.rrdtool.org for more information

1.4. How do I bill customers with Cricket ?

The main answer is: *Don't do it!* Why: It's not supported... it's not designed for this, nor is it tested for this. There's no warranty, so if you customers sue you, don't expect to sue the Cricket maintainers. It would be good to provide a list of alternative systems that can be trusted for billing purposes, or documentation from Cricket users who have successfully done so.

2. Installation and Configuration

2.1. How I install cricket on Unix ?

Follow steps described in “Installing Cricket for Complete Beginner”article which is available from http://cricket.sourceforge.net/support/doc/beginner.html

2.2. How I install cricket on NT ?

There are rumors about working NT installations. If you have Cricket running on NT please contribute.

2.3. Can I import my MRTG config files and data into Cricket ?

Short answer is *NO*.

Cricket configuration tree is completely incompatible with MRTG config file. Also MRTG stores measurements in text files while RRDtool uses binary files.

In theory it is possible to write a script which will convert MRTG config files into cricket subtree, parse MRTG data files and fill up relevant RRD data file. In practice, no one bothered to do this yet.

2.4. Is there a central repository of useful configurations

Check out http://www.certaintysolutions.com/tech-advice/cricket-contrib/

2.5. How I improve CGI performance on Unix ?

Cricket can be run under mod_perl module for Apache web server. Check out http://cricket.sourceforge.net/support/doc/modperl.html

2.6. How I add new device to Cricket

There are detailed instructions at http://cricket.sourceforge.net/support/doc/new-devices.html

3. Common Problems

3.1. FAQ: mega = 10^6 vs 2^20

While it is a common knowledge that byte has 8 bits many people think that Kilobyte has 1000 bytes and Megabyte has 1 000 000 bytes. This is wrong. Kilobyte has 1024 bytes and Megabyte has 1024 * 1024 = 1048576 bytes. This is

about 4.9% difference.

By default cricket uses powers of 10 to convert measurement data to SI units. This is OK for most things. However network traffic is measured in bits per second so cricket should use powers of 2 to scale this data. bytes tag tells cricket to use powers of 2 instead of powers of 10.

See definition of ifInOctets and ifOutOctets in the default config tree.

3.2. Why HP OpenView measurements differ from Cricket ?

HP OpenView assumes that mega = 10^6. See answer to previous question.

3.3. I see huge spikes on my graphs for Netapp filer. What is going on ?

Apparently, the Netapp stores counters internally as unsigned integers (Counter32 or just plain Counter in SNMP parlance), but sends them out as signed integers (INTEGER in SNMP parlance). Cricket collects its data using Perl, which (unlike C) will not fix the corruption caused by Netapp‘s goof, so it will blithely operate on the large negative values it retrieves and (not surprisingly) come up with wrong values.

Solution: add 2^31 to the collected value. Following code fragment donated by Matthew Stier <Matthew.Stier@fnc.fujitsu.com> illustrates it:

```
my($cpuusage) = "1.3.6.1.4.1.789.1.2.1.2.0";
my($netsent) = "1.3.6.1.4.1.789.1.2.2.2.0";
my($netrcvd) = "1.3.6.1.4.1.789.1.2.2.3.0";

my(@ret) = snmpUtils::get($snmp, $cpuusage, $netsent, $netrcvd);

print $ret[0] . "\n";
print 2147483648 + $ret[1] * 1024 . "\n";
print 2147483648 + $ret[2] * 1024 . "\n";
```

Matthew says: “ I’ve had discussions with Network Appliance, and they have added additional OID’s to the MIB to provide the same information, in the more common "Counter32" format. These OID’s are available in ONTAP 6.”

4. Miscellaneous questions

4.1. NT SNMP resources

From: "William Schwartz" <WSchwartz@apz-applied.com>

http://www.wtcs.org/snmp4tpc/ Is a great place for info on SNMP for NT. It is for MRTG specifically, but I think you can make the "conversion" in your mind. Its has a great tutorial for setting up SNMP properly on your boxes.

4.2. What are perl mode indentation setting for emacs ?

Try these:

```
# Local Variables:
# mode: perl
# indent-tabs-mode: nil
# tab-width: 4
# perl-indent-level: 4
# End:
```

4.3. How do I change my config tree to use the view dictionary entries introduced in 1.0.4? (added by jakeb 6/7/02)

This is best illustrated via example. Suppose this is your existing (pre-1.0.4) targettype dictionary and graph --default-- dictionary:

```
graph --default--
    default-ranges = "d"
    show-avg-max = false

targettype process
    ds = "usertime, kerneltime, utilization, workingsetsize,
       reads, writes, pagefaults"
    view = "processor: usertime kerneltime utilization,
            memory: workingsetsize,
            io: reads writes,
            swapping: pagefaults"
```

Here is the translation to the 1.0.4 syntax:

```
graph --default--
# datasource specific, remains in graph --default--
    show-avg-max = false

view --default--
# non-datasource specific, goes to view --default--
    default-ranges = "d"

targettype process
    ds = "usertime, kerneltime, utilization, workingsetsize,
       reads, writes, pagefaults"
    view = "processor, memory, io, swapping"

view processor
    elements = "usertime,kerneltime,utilization"

view memory
    elements = "workingsetsize"
```

```
view io
    elements = "reads,writes"

view swapping
    elements = "pagefaults"
```

And now for some advance features using the same example:

```
view processor
    elements = "usertime,kerneltime,utilization"
# assume these are on the percentage scale, force y-axis
    y-min = 0
    y-max = 100

view memory
    elements = "workingsetsize"
# assume aberrant behavior detection rras are define for this target
    holtwinters = true

view io
    elements = "reads,writes"
# choose a different font (RRDtool 1.1.x)
# Win32 Note:
# RRDTOOL seems to have difficulty if the font path contains a colon
# so times.ttf is in the working directory (crickethome)
    rrd-graph-args      = "--font DEFAULT:12:times.ttf"

view swapping
    elements = "pagefaults"
# change the default-range for this view
    default-ranges      = "d:w"
```