Upwatch User Guide

Ron Arts

Upwatch User Guide
by Ron Arts

Table of Contents

Preface

This book describes the client side installation and con guration of UpWatch.

Chapter 1. About UpWatch

1.1. History

UpWatch is born from the loins of Netland Internet Services BV, Amsterdam, The Netherlands. We are a hosting company which started in 1993 (when even Bill Gates knew nothing of the internet). We started doing managed hosting in 1995, and that's when we found out about monitoring. The hard way.

It became clear that customers can easily bring down their own server on impossible moments, and that it doesn't look very good if you both nd out about that the monday after. So we started doing SLA's and limit customer rights on their own server.

Initially we used Big Brother (bb4.com) for monitoring. This is an outstanding and useful package, and we have been using it for many years. But it has a few downsides. One is scalability. It does not scale well to hundreds of hosts. Also it has a geek-like look, we felt we couldn't give the URL to our customers. Third problem was integration with our backof ce.

At the same time yours truly was thinking about setting up a commercial service for monitoring servers remotely. All this culminated into UpWatch. So lets get straight to the ..

1.2. Features

This is the full list of relevant upwatch features:

- OS support: clients available for Linux, Windows, FreeBSD, Solaris
- GUI is multi-language enabled (uses gettext)
- GUI has mobile client support
- GUI is brandable, you can give it your own look & feel
- Generates realtime graphs from the database
- Noti cations by email or SMS.
- Clients for: HTTP GET, IMAP, MSSQL, MySQL, PING, POP3, PostgreSQL, SMTP, SNMP GET, TCP connect (any port)
- Local client detects: CPU load, loadavg, swap use, I/O use, memory use, and where supported hardware info like CPU temperature, fan speed and Power voltages. Also you can set it up to scan any log le using regular expressions you supply.
- Scalable: designed for monitoring tens of thousands of hosts
- Multi-tenanting: multiple companies can run monitoring services for network of multiple client-companies using the same backend+probe serverpark
- Extensive and complete documentation, partly generated from source
- Secure: run as ordinary user, developed with security in mind
- Fully opensource: GUI built on Apache/PHP, Backend on C/Perl, Database is MySQL. Uses GNU con gure.
- SuSE, RedHat and Fedora RPM's generated from sourcetree for easy installation

Chapter 2. Installation

2.1. Getting upwatch

The UpWatch clients can be download from www.upwatch.com. They are available in the following formats:

- as a tar.gz le, including sources for every supported platform
- SuSE, RedHat of Fedora .rpm les
- A windows installer

If you want, you can inspect the code for security issues.

2.2. Requirements

2.2.1. Run-time requirements

First ensure that the time/date on all hosts is correctly set.

Here's a list of everything we expect (I'll also list the version we use ourselves):

- glib2 >= 2.0.4
- xml2 (any version will do)
- libpcre 3.9.10
- libncurses 5.2
- libreadline 4.3

Delivered with upwatch are libstatgrab (0.10) (http://www.i-scream.org/libstatgrab/), to retrieve critical operating system values, xmbmon 2.03 (http://www.nt.phys.kyushu-u.ac.jp/shimizu/download/download.html) for harware statistics, and the State Threads Library (1,4) (http://state-threads.sourceforge.net/) for fast and ef cient multithreading.

2.2.2. Build requirements

You probably don't want to build upwatch yourself. Most likely you'll grab the RPM packages and issue rpm -Uvh upwatch*rpm. Then skip to Con guration.

But on the other hand: you can build the software yourself. Apart from the normal GNU compilation tools, and the development versions of the above mentioned packages, you'll need the following on your system to build upwatch:

- lynx
- autogen 5.3.6 (autogen.sourceforge.net)
- autogen needs libguile and umb-scheme
- RPM tools, if you want to build RPM's

If you run RedHat, Debian or SuSE, don't forget to install the *-devel packages if there are any.

2.3. Compiling upwatch

Just in case you really want to (or need to) compile upwatch yourself, it's pretty easy:

```
$ tar xzvf upwatch-x.x.tar.gz
$ cd upwatch-x.x
$ ./con gure
$ make
$ make install
```

Nothing to it... In case of problems, you're probably missing some library or header les, or they are in unexpected places. Look at the last parts of con g.log.

Chapter 3. Con guration

3.1. Con guration Files

First: it is extremely important your host clock is set correctly, and please use NTP to keep it up-to-date. If you don't, you will see very strange things happening, and what's more, if this gets into the database, it will be impossible to correct (apart from deleting all results alltogether).

The directory structure for con guration les is as follows. At the toplevel is upwatch.conf (usually residing in /etc). This le is read by all programs, and contains global parameters, and parameters you want to make globally known. At the same level is the directory upwatch.d. This contains con g les for every program. The upwatch.conf le looks like this:

```
# Upwatch con guration le
# contains defaults for all modules
# these can be overridden in /etc/upwatch.d/<module>.conf
#

realm neonova
debug 2
syslog no
stderr no

log le /var/log/upwatch/messages

spooldir /var/spool/upwatch
```

All values in this le can be overridden or augmented in program-speci c les. realm denotes the short name for the database this system belongs to, This usually corresponds to your company name. debug speaks for itself. Never set it higher than two. Zero suppresses all debugging except the most critical ones, 'debug 1' will output only warnings, 'debug 2' will send progress information to the log. syslog yes will enable logging to the system log, stderr is really not very useful because the commandline parameter -e 1 accomplishes the same for every program. The log le value denotes where logging will takes place. This le should be writable by the user upwatch. The same holds for the spooldir base directory. It should contain subdirectories of the maildir format (meaning: each having a new and tmp subdirectory).

On the average client two processes will be running continuously: uw_sysstat and uw_send The rst program collects info on your system, and writes an XML le, the second sends it to the central database. upwatch.d will contain the les uw_sysstat.conf and uw_send.conf. Lets rst look at uw_send. It looks like this:

```
# where to send to
host cms-db.of ce.netland.nl
port 1985
uwuser 20010631
uwpasswd SaSNF8bu
debug 2
threads 1
# where to read from
input uw_send
```

Host and port refer to the central database location. You need a user and uwpasswd to log into that. The debug setting determines the amount of logging that the program does, threads how many les to send at the same time, and nally, the input tells uw_send where to nd its XML les to send out. Pretty straightforward.

The uw_sysstat.conf le is comparable:

```
serverid 381
output uw_send
hwstats on
errlog syslog /var/log/messages
errlog maillog /var/log/maillog
```

The serverid is the numerical id of this server in that particular database. Without output it would not know where to store the XML result le. hwstats determines (where supported) if uw_sysstat will try to talk to the motherboard to get hardware values like temperature and fanspeed (run mbmon rst to test if this is supported) and the errlog parameters tell uw_sysstat which les to monitor and their format.

3.2. Trouble shooting

3.2.1. uw_send crashes immediately when it is started

Most probably you have an libxml2 with compiled-in thread support on your system. libxml2 should not be compiled with thread support, so you'll have to replace it. You can check this with:

```
# ldd /usr/lib/libxml2.so
```

The output should not contain any threading libraries.

3.2.2. Hardware Monitoring problems

Hardware monitoring does not work for all hardware. There is a list of supported hardware in the ReadMe le in the xmbmon subdirectory. Here are solutions for some common problems.

3.2.2.1. ASUS Pentium4 motherboards

Some of the ASUS Pen4 motherboard with ICH2(82801BA) and ICH4(82801DB) chipsets switch off the SMBus PCI device. Therefore, one has to enable it explicitly in order to make hardware-monitoring possible.

If "mbmon/xmbmon" does not search the SMBuses, it's not the fault of "mbmon/xmbmon", but the results of ASUS's BIOS setting. For users who experienced this, try the following:

- Check the chipset. ICH2/ICH4 chipset is identi ed by the following PCI con guration data:

  pci-device ID --- ICH2: vendorID = 0x8086, chipID = 0x2440
  ICH4: vendorID = 0x8086, chipID = 0x24C0

pci-device ID --- ICH2(PM): vendorID = 0x8086, chipID = 0x2443
ICH4(PM): vendorID = 0x8086, chipID = 0x24C3

(PM means Power Management Controller, which provides SMBus access.)

By using the command:

pciconf -l (FreeBSD)

pcitweak -l (Linux)

list up the pci-devices recognized by OS, and try to nd whether your chipset is ICH2 or ICH4. If you nd ICH2 or ICH4 in the list, while do not nd ICH2(PM) or ICH4(PM), then you are one of sufferers of this problem. If not, your problem is of different kind, sorry.

- Enable the SMBus access. You have to enable the ICH2(PM) or ICH4(PM) by turning off the bits number 8 and 3 (counting from 0) in the word data of the LPC register of ICH2 or ICH4 at 0xF2. First, nd the value of the actual data:

  # pciconf -r -h [selector of ICH2 or ICH4] 0xf2 (FreeBSD)

  # setpci -d 8086:2440/24c0 f2.w (Linux).

  (bit) 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
  word data at 0xF2 x x x x x x x * x x x x * x x x

  Next, after calculating the byte by VAL = VAL & 0xFEF7, set this VAL to ICH2 or ICH4 by the following command:

  # pciconf -w -h [selector of ICH2 or ICH4] 0xf2 VAL (FreeBSD)

  # setpci -d 8086:2440/24c0 f2.w=VAL (Linux).

- Check ICH2(PM) or ICH4(PM) appears. List up the pci-device again, and con rm that now ICH2(PM) or ICH4(PM) are there. Then, try "mbmon/xmbmon" again.

  Reference: ./prog/hotplug/README.p4b of the lm_sensors package downloadable from http://www.lm-sensors.nu/

3.2.2.2. OpenBSD problems

>From: "andreas guetl"
>To:
>Subject: x for mbmon on openbsd
>Date: Fri, 5 Sep 2003 10:35:37 +0200

hello!

i tried various progs for monitoring cpu-temp on openbsd boxes
but none of them worked. then i found the solution on
http://archives.neohapsis.com/archives/openbsd/2001-02/2550.html
and id simply did the trick, not only for healthd (which gave me
wrong values) but also for mbmon which works very ne.

1.) enable the option APERTURE in the kernel
enabled by default but disabled in any selfmade kernel, who needs
x on openbsd ;)

2.) uncomment machdep.allowaperture in /etc/sysctl.conf and change
the value from 2 to 1

after rebooting with the new kernel the monitoring works really
 ne.

thanks for your ne program!!

(but what has the agp-aperture to do with monitoring hw?)

so long,
andy

3.3. Monitoring speci c log les

uw_sysstat is special in that it allows you to scan every (line-oriented) log le you want. It uses regular expressions to set a yellow or red state. It works as follows:

On startup it reads /etc/upwatch.d/uw_sysstat.conf , and searches for log le statements. Say it encounters the statement:

```
log le errlog /var/log/messages
```

what it does is it reads all les in the directory /etc/upwatch.d/uw_sysstat.d/errlog (except rmacros.txt and macros.txt). These les should contain regular expressions pre xed by one of the keywords green, yellow, or red. Next uw_sysstat starts scanning /var/log/messages . It reads a line from the log le and the following happens:

- Check against the red list. If match found, ag red condition, and send the offending line to the upwatch server
- Check line against the yellow list. If matches, ag yellow and send to server
- Check agains green list. If it matches, ignore this line and go the next line in the log le. If the current line does not match any of the green list, ag yellow, and send line to server

The regular expressions may (for readability) contain macros, they should be entered in /etc/upwatch.d/uw_sysstat.d/syslog/macros.txt .

You can easily add a directory of youw own, containing regular expressions for you own log les. In fact upwatch includes a handy utilitychklog to help you create regular expression lists. Here is an example how to do it.

Suppose you plan to scan the log le for the imaginary 'timtim' navigational system. It resides in /var/log/timtim.log .

- First create the directory:

  ```
  # cd /etc/upwatch.d/uw_sysstat.d
  # mkdir timtim
  # cp syslog/rmacros.txt timtim
  # cp syslog/general timtim
  # chown -R root:upwatch timtim
  # chmod 770 timtim
  # chmod 660 timtim/*
  ```

- Next look atrmacros.txt and tailor it to accomodate speci cs for the timtim log le. It might for example contain entries for zipcodes, or latitudes/longitudes for which you would like to create macros.
- Next step: extract regular expressions from an example log le:

  ```
  # chklog -t timtim -r /var/log/timtim.log | sort -u > /tmp/timtim
  ```

- edit this le. Maybe it will contain dupes, you should try to keep the number of regular expressions low. If you are satis ed you can try copying it to /etc/upwatch.d/uw_sysstat/timtim using any lename, and usechklog to test it:

  ```
  # chklog -t timtim -m /var/log/timtim.log
  ```

  Now you should only see the lines you want to be reported by uw_sysstat. Repeat steps until you are satis ed

- Finally tell uw_sysstat that you want it to start scanning by addingdir log timtim /var/log/timtim.log to its con guration le. That's it. Occasionally you may see loglines showing up as yellow, but they should in fact be ignored. You can always add regular expressions for such lines. You don't need to restauw_sysstat, it will notice the le was modi ed.

3.4. Maintenance and Upgrades

The RPM les contain logrotate.d les to rotate the upwatch log les, other platforms should supply their own.

If the uw_senddaemon for some reason is not running, probing results will stack up in the queue. They will be sent out whenuw_sendis restarted.

Upgrades with yum or apt-get are automatic. There are repositories available, please ask our helpdesk for the proper addresses.

Chapter 4. Utilities

4.1. mbmon

On x86 architectures and some operating systems hardware readouts may be obtained. We use xmbmon (http://www.nt.phys.kyushu-u.ac.jp/shimizu/download/download.html) to get these values, so wherever that's supported, we can do it. But before switching on hwstats in uw_sysstat, use mbmon to test if your setup is supported. By the way, if mbmon is not on your system, its de nitely not supported. anyway, also look if the values given by mbmon are meaningful.

Each program has a manual page that documents options. Every long commandine option can also be entered in a con guration le.

4.2. chklog

This utility makes it simpler to create your own set of regular expressions for a particular log le you want to be monitored by uw_sysstat. It can both scan a test log le and output regular expressions, as scan a log le and outputs lines that are suspicious, and to which uw_sysstat should repond by agging a yellow or red condition. See Monitoring speci c log les for the procedure.

4.3. uwq

You can show all queues on the cureent system with uwq. We regularly use watch uwq to monitor queue status.

4.4. uwregexp

While creating a regular expression for an application speci c log le, sometimes you get problems creating a regular expressions that ts really well. This is where uwregexp comes in handy. You feed it the line to match, and the log le type, can it gives you a prompt where you can try various regular expressions. Use up and down arrows the circle through previous expressions.

4.5. uwsaidar

this is a general handy tool, a bit like top, it gives you an ongoing system status screen with CPU, memory, and I/O usage and vaious other parameters. It actually comes straight out of the libstatgrab (http://www.i-scream.org/libstatgrab/) library, that accompanies the Upwatch client.

Index