Fast Adaptive Insecure Monitoring: the monitoring system that should never have been invented, that is fun

jul

2024-07-07

- ./tmp/bin/ash.txt
- ./tmp/bin/fdflush.txt
- ./tmp/bin/sh.txt
- ./tmp/etc/ca-certificates/update.d/certhash.txt
- ./tmp/sbin/fbsplash.txt
- ./tmp/usr/bin/c_rehash.txt
- ./tmp/usr/local/lib/python3.8/config-3.8-x86_64-linux-gnu/install-sh.txt
- ./tmp/usr/local/lib/python3.8/venv/scripts/posix/activate.csh.txt
- ./tmp/usr/local/lib/python3.8/venv/scripts/posix/activate.fish.txt
- ./tmp/usr/share/terminfo/m/macintosh.txt
- ./tmp/usr/share/terminfo/x/xwsh.txt

% Fast Adaptive Insecure Monitoring: the monitoring system that should never have been invented, that is fun % jul % 2024-07-07

Intro

This project is an implementation of such a way of thinking distributed system. For sake of education I took the most compact language for the task : *bash*

We are gonne realize on this principle a Fast Adaptative Insecure Monitoring system.

FAIM is designed as a funny experiment of doing a munin clone (doing less) in bash only that is specialized in high speed (~1 seconde / measure) distributed measuring system without a centralized collector.

No broker, no Zmq, no webrtc, no QUIC, no rabbitMQ are used for transport but ... BROADCAST UDP.

Hence, well, this toy is fondamentally insecure and can hardly be ciphered in its current form. But, it enables a category of software that are both educational for doing your own tool AND for deploying an adhoc measuring system.

Read full documentation here

```
{.graphviz digraph finite_state_machine {     rankdir=LR;     size="8,5"
node [shape = doublecircle]; LR_0 LR_3 LR_4 LR_8;     node [shape
= circle];     LR_0 -> LR_2 [ label = "SS(B)" ];     LR_0 -> LR_1
[ label = "SS(S)" ];     LR_1 -> LR_3 [ label = "S($end)" ];
LR_2 -> LR_6 [ label = "SS(b)" ];     LR_2 -> LR_5 [ label =
"SS(a)" ];     LR_2 -> LR_4 [ label = "S(A)" ];     LR_5 -> LR_7
[ label = "S(b)" ];     LR_5 -> LR_5 [ label = "S(a)" ];     LR_6
-> LR_6 [ label = "S(b)" ];     LR_6 -> LR_5 [ label = "S(a)"
];     LR_7 -> LR_8 [ label = "S(b)" ];     LR_7 -> LR_5 [ label
= "S(a)" ];     LR_8 -> LR_6 [ label = "S(b)" ];     LR_8 -> LR_5
[ label = "S(a)" ]; }
```

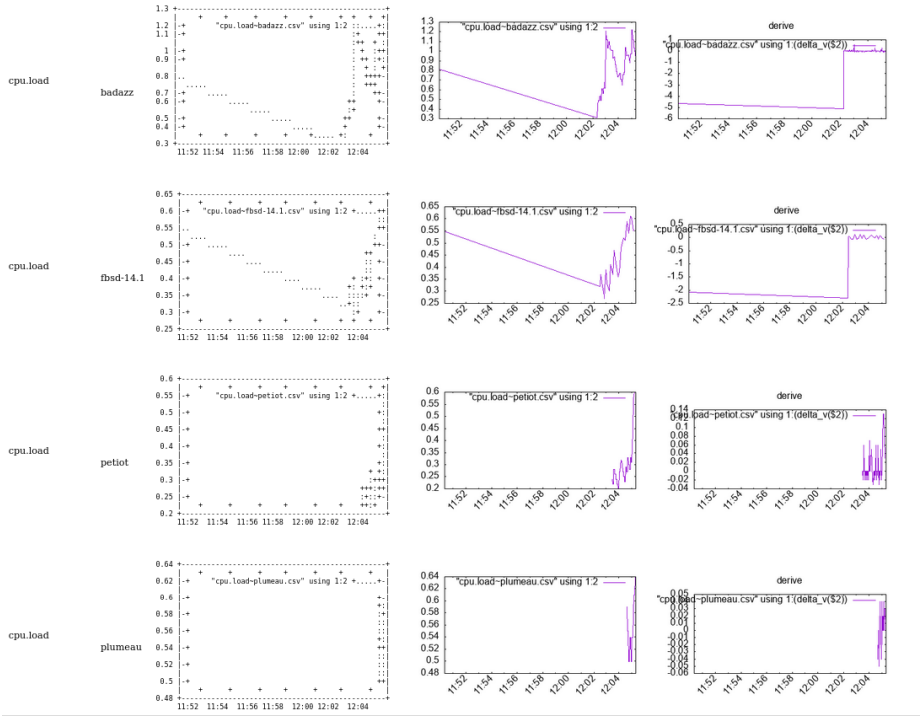

Figure 1: example

Quickstart

Requirements

Perl, python3, gnuplot (gnuplot-lite maybe enough if 1Gb dependency rebukes you), bash, socat, and whatever the plugins have dependencies upon.

Starting the probe

`./start.sh`

Starts the probe. It will emit (see API of start for network parameters) on the UDP broadcast address in ASCII excactly what `bin/writer.sh` emits on stdout.

To stop the probe simply type

`./stop.sh`

Starting the collect of data

`LURKER=1 ./start.sh`

Will start the probe AND the data collector. The data collector can also be caught in a standalone mode with `./bin/launch_lurker.sh` or `./bin/launch_lurker.py`.

if you go in ./data/ you will see both csv where data are stored accumulating and the making of the html resume.

Just open ./data/index.html to view the graphs.

You can erase the content of data at any moment, everything will reconstruct itself.

What the lurker sees from the broadcast is log into ./log/journal.txt.

CSV files and html can be reconstructed by typing

```
cat log/journal.log | bin/lurker
bin/mkhtml.sh
```

mkhtml is the bash equivalent of PHP or using jinja in python : dynamic html generation.

I seriously advise to install tcpdump, and remember that `tcpdump -A [-i interface] -s0 udp and port 6666` can be a serious life saviour while troubleshooting.

State Machine of the system

See diag.dot

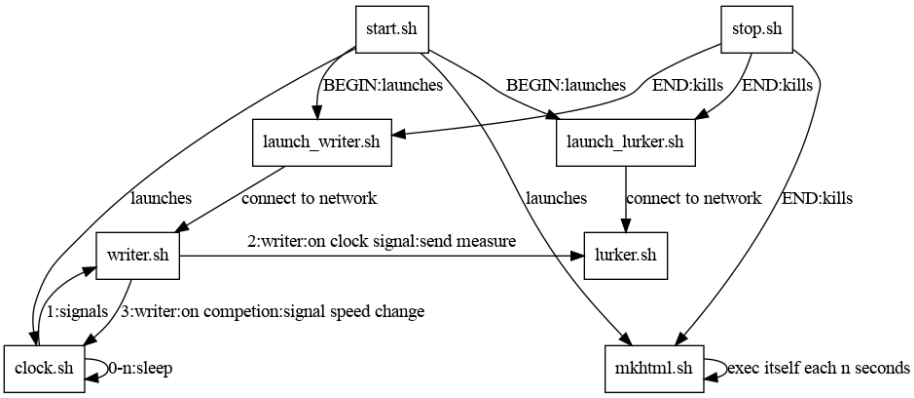

Figure 2: diag

Agent Oriented Programming

1. EverythingIsAnAgent
2. Agent communicate by sending and receiving messages (the way they prefer as long as they send messages).
3. Agent have their own memory and autonomy
4. Every Agent is an instance of an artifact (which then as to be accounted as an agent).
5. The Agent is accountable for maintaining its consistency as a state/transition agent
6. The topology is more important than the code.
7. each state machine is on a plane for which an uncoupled state machine lays.
8. violation of uncoupling between layers is bad so it has to be handled with care.

Documentation of each scripts

API of each components.

./bin/asci_plot.sh.txt

./bin/basic_plot.sh.txt

./bin/clock.sh.txt

./bin/launch_lurker.sh.txt

NAME launch_lurker.sh

SYNOPSIS Make data collector available for listening to the probes

```
[TICK=2] [BROADCAST=192.168.1.255] [RANGE=24] [PORT=6666] ./launch_writer.sh
```

OPTIONS For explanation of options see <file:../start.sh.html>

./bin/launch_writer.sh.txt

NAME launch_writer.sh

SYNOPSIS Make writer emit on BROADCAST/RANGE ono port PORT

```
[TICK=2] [BROADCAST=192.168.1.255] [RANGE=24] [PORT=6666] ./launch_writer.sh
```

OPTIONS For explanation of options see <file:../start.sh.html>

./bin/lurker.sh.txt

NAME lurker.sh

SYNOPSIS Collector of data

```
./lurker.sh
```

```
Can be used as
```

```
while [ 1 ]; do writer.sh | lurker.sh; sleep 30; done
```

```
To collect data emitted locally about the machine.
```

```
Results are written in ../data
```

./bin/mkhtml.sh.txt

NAME mkhtml

```
HTML maker
```

SYNOPSIS Generator of HTML output from data collected in ../data

```
[DAEMON=] [SINCE=3600] mkhtml.sh
```

```
Can be used as
```

```
./mkhtml.sh
```

```
to generate the web page in ../data
```

OPTIONS DAEMON This code will run permanently waking itself up to update the web page.

```
SINCE
    The window span time you are interested in in seconds from NOW
```

./bin/plot_histo_g.sh.txt

./bin/plot_histo.sh.txt

./bin/plot_rrd2.sh.txt

./bin/writer.sh.txt

NAME writer.sh

SYNOPSIS Emitter of data

```
[TICK=2] ./writer.sh
```

OPTIONS For explanation of options see <file:../start.sh.html>

```
If TICK is set then writer will assume it is to be launched in
conjunction with <file:./clock.sh.html> and do nothing until clock.sh
sends a signal to it to write data.
```

./mkdoc.sh.txt

NAME mkdoc.sh

SYNOPSIS Generates the doc. Requires pandoc for markdown to html conversion

```
./mkdoc.sh
```

./plugin/cpu.txt

./plugin/ibm_acpi_fan.txt

NAME acpi_ibm - Munin plugin to monitor the fan speed returned by ACPI probe.

APPLICABLE SYSTEMS FreeBSD systems with ACPI support. man acpi_ibm(4)

CONFIGURATION add ibm_acpi in loader.conf

USAGE Link this plugin to @@CONFDIR@@/plugins/ and restart the munin-node.

INTERPRETATION The plugin shows the fans' speeds.

MAGIC MARKERS #%# family=auto #%# capabilities=autoconf

BUGS None known.

VERSION v1.1 - 2024-03-24

AUTHOR Julien Tayon (julien@tayon.net)

LICENSE GPLv2

./plugin/ibm_acpi.txt

NAME acpii_ibm - Munin plugin to monitor the temperature in different ACPI Thermal zones.

APPLICABLE SYSTEMS FreeBSD systems with ACPI support. man acpi_ibm(4)

CONFIGURATION add ibm_acpi in loader.conf

USAGE Link this plugin to @@CONFDIR@@/plugins/ and restart the munin-node.

INTERPRETATION The plugin shows the temperature from the different thermal zones.

MAGIC MARKERS #%# family=auto #%# capabilities=autoconf

BUGS None known.

VERSION v1.1 - 2024-03-24

AUTHOR Julien Tayon (julien@tayon.net)

LICENSE GPLv2

./plugin/irq.txt

NAME interrupts - list number of interrupts since boot (linux) or the interrupt rate per interrupt

CONFIGURATION No configuration

AUTHOR Idea and base from Ragnar Wisløff.

LICENSE GPLv2

MAGIC MARKERS #%# family=auto #%# capabilities=autoconf

./plugin/open_files.txt

NAME open_files - Plugin to monitor the number of open files in the system

CONFIGURATION No configuration

AUTHOR Unknown author

LICENSE GPLv2

MAGIC MARKERS #%# family=auto #%# capabilities=autoconf

./plugin/processes.txt

NAME processes - Plugin to monitor processes and process states.

ABOUT This plugin requires munin-server version 1.2.5 or 1.3.3 (or higher).

```
This plugin is backwards compatible with the old processes-plugins found
on SunOS, Linux and *BSD (i.e. the history is preserved).

All fields have colours associated with them which reflect the type of
process (sleeping/idle = blue, running = green, stopped/zombie/dead =
red, etc.)
```

CONFIGURATION No configuration for this plugin.

AUTHOR Copyright (C) 2006 Lars Strand

LICENSE GNU General Public License, version 2

MAGIC MARKERS #%# family=auto #%# capabilities=autoconf

./plugin/stat.txt

NAME interrupts - Plugin to monitor the number of interrupts and context switches on a system.

CONFIGURATION No configuration

AUTHOR Idea and base from Ragnar Wisløff.

LICENSE GPLv2

MAGIC MARKERS #%# family=auto #%# capabilities=autoconf

./plugin/tcp.txt

NAME tcp - Plugin to monitor IPV4/6 TCP socket status on a Linux host.

LICENSE GPLv2

./pubsub.sh.txt

./start.sh.txt

NAME start.sh

DESCRIPTION Launches the networked apparatus of measures. It is the reciprocal function of stop.sh

SYNOPSIS All arguments are passed by environment variables

```
[HOST=0.0.0.0] [PORT=6666] [TICK=2] [LURKER=] [BROADCAST=192.168.1.255] [RANGE=24] [SINC
```

OPTIONS TICK TICK is the initial clock given to the system. It will however converge to its computed value.

```
LURKER
    When LURKER is set, the data collecting agent is launched and
    process all probes sent on the given broadcast address

BROADCAST
    UDP BROADCAST address to use

RANGE
    Range in the form [0-32] to specify the BROADCAST range.

    Ex: 24 will specify $BROADCAST/24

SINCE
    Argument given to the html generator to know how much seconds since
    NOW must be shown in the graph.
```

./stop.sh.txt

NAME stop.sh

DESCRIPTION stops all agent launched by start

OPTIONS None

./tmp/bin/ash.txt

./tmp/bin/fdflush.txt

./tmp/bin/sh.txt

./tmp/etc/ca-certificates/update.d/certhash.txt

./tmp/sbin/fbsplash.txt

./tmp/usr/bin/c_rehash.txt

./tmp/usr/local/lib/python3.8/config-3.8-x86_64-linux-gnu/install-sh.txt

./tmp/usr/local/lib/python3.8/venv/scripts/posix/activate.csh.txt

./tmp/usr/local/lib/python3.8/venv/scripts/posix/activate.fish.txt

./tmp/usr/share/terminfo/m/macintosh.txt

./tmp/usr/share/terminfo/x/xwsh.txt