



**njmon is nmon but saving to JSON format
for modern performance stats tooling**

Slides + Video Replay + FAQ available at <http://ibm.biz/PowerVUG>



Nigel Griffiths - Advanced Technology Support, Europe, IBM
nag@uk.ibm.com
 @mr_nmon twitter
<http://nmon.sourceforge.net/njmon>
<https://www.youtube.com/user/nigelargriffiths>
<http://tinyurl.com/AIXpert>

<https://tinyurl.com/njmon>



- Version 16

This is Nigel's Personal Deck & Nigel's Opinions

Summary - nmon for the next generation

Nigel's Monitor (nmon) has come a long way in 2 decades.

In this session, Nigel reviews the state of nmon today both modes: online on-screen monitoring and graphing the stats later with various tools and includes a review of areas in which nmon struggles.

Then we look at the performance monitoring needs for the next few years and experiments with collecting JSON data and live web-based services to short and long-term graphing.

Bring your ideas and experience to the discussion.

njmon = nmon + JSON format + real-time push to a stats database + instant graphing of "all the stats you can eat" (AIX and Linux)

This njmon is a major overhaul of nmon for the next 10 years:

- Load more stats
- JSON format is self documenting, flexible and the performance stats format for many new tools
- Direct real-time loading of the JSON into modern open source time aware databases
- New age browser based graphing tools allow dynamic data choice and graph style per VM, per server or across the estate

All this will be covered and more including many demo's.

<https://www.youtube.com/user/nigelargriffiths>



nmon for Linux
Nigel Griffiths - 1 / 3




1. nmon for Linux Starter Pack
Nigel Griffiths
20:01
2. nmon for Linux Data Capture
Nigel Griffiths
14:56
3. nmonchart to graph your nmon data files
Nigel Griffiths
22:21

~60 minutes

Newbie starter playlists

nmon for AIX
Nigel Griffiths - 1 / 4



1. nmon Starter Pack Monitoring Online for current AIX versions
Nigel Griffiths
14:00
2. nmon Starter Pack for AIX Data Capture
Nigel Griffiths
14:37
3. nmon Starter Pack for AIX Analyser
Nigel Griffiths
9:47
4. nmonchart to graph your nmon data files
Nigel Griffiths
22:21

~60 minutes



nmon started about 22 years ago for benchmarking & reports
nmon became part of AIX

21st Nov 2008



nmon also works on Linux

Open sourced 27th July 2009



- C language is Open Source
 - <http://nmon.sourceforge.net>
- Different source code
 - Similar output to nmon for AIX
- Many Linux distros & versions

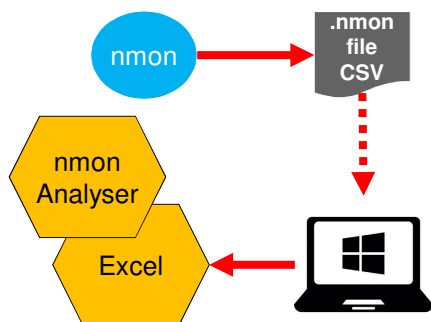
100% IBM Supported

Supported by Nigel Multiple Platform:

- POWER4 – 9 = ppc64
- x86_64 (Intel, AMD)
- ARM (Raspberry Pi)
- Z mainframe

nmon + analyser

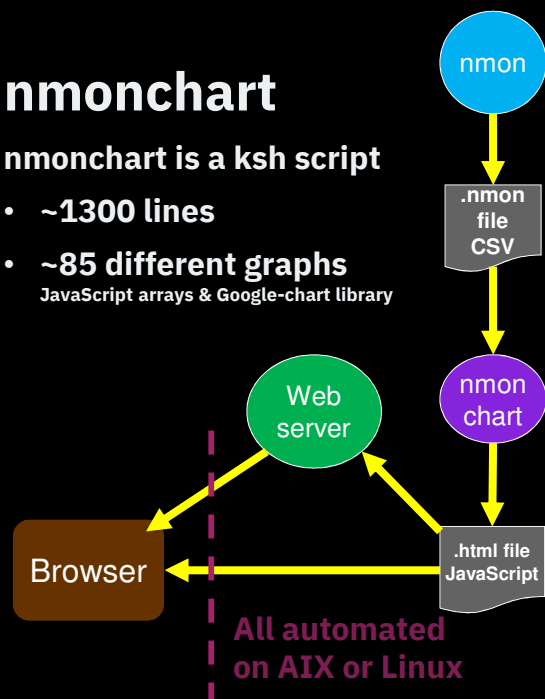
Move file to your laptop,
 start Excel → Analyser in
 Visual BASIC but largely a
 manual process



nmonchart

nmonchart is a ksh script

- ~1300 lines
- ~85 different graphs
 JavaScript arrays & Google-chart library



Online nmon

```

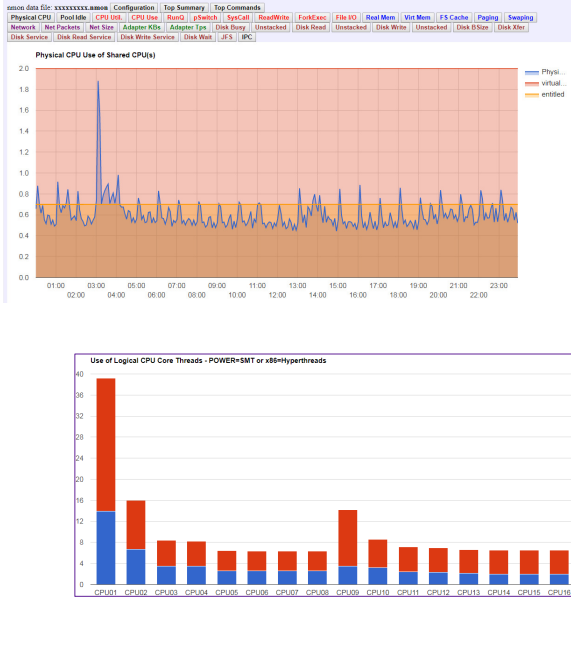
topas nmon-5-Top-by-I/O-use Host=brass6 Refresh=2 secs=15:54.13
CPU Utilization-Small-View EntitledCPU= 1.00 UsedCPU= 0.976
Logical CPU 0-----25-----50-----75-----100
CPU User% Sys% Wait% Idle%|
0 37.5 0.0 0.0 62.5|#####|
1 0.0 0.0 0.0 100.0|
2 0.0 0.0 0.0 100.0|
3 0.0 0.0 0.0 100.0|
4 0.0 0.0 0.0 100.0|
5 0.0 4.5 0.0 95.5|>>>
6 0.0 4.5 0.0 95.5|>>>
7 0.0 4.5 0.0 95.5|>>>
-----|-----|-----|-----|
EntitledCapacity/VirtualCPU +-----+
EC 21.8 0.1 0.0 15.8|#####|
VP 10.9 0.1 0.0 7.3|#####|
EC= 37.8% VP= 18.9% +---No Cap---|---Folded=1---|100% VP=2 CPU+
-----+-----+-----+-----+-----+-----+-----+
Memory Physical PageSpace | pages/sec In Out | FileSystemCache
% Used 93.9% 1.5% | to Paging Space 0.0 0.0 | (mmap) 0.8%
% Free 66.1% 98.5% | to File System 0.0 0.0 | Process 15.8%
MB Used 1387.0MB 7.6MB | Page Scans 0.0 0.0 | System 17.3%
MB Free 2708.2MB 504.4MB | Page Cycles 0.0 0.0 | Free 66.1%
Total(MB) 4096.0MB 512.0MB | Page Steals 0.0 0.0 | Total 100.0%
-----+-----+-----+-----+-----+-----+-----+
Min/Maxperm 108MB( 3%) 3247MB( 98%) <--% of RAM | memlimit 0.8%
Min/Maxfree 960 1088 Total Virtual 4.5GB | userlimit 90.0%
Min/Maxpghead 2 8 Accessed Virtual 1.3GB 29.0% | Pinned 30.9%
-----+-----+-----+-----+-----+-----+-----+
Disk-KBytes/second-(M=1024,N=1024*1024)
Disk Busy Read Write 0-----50-----75-----100
Name KB/s KB/s | | | |
hda#0 0% 0 0| | | |
Totals 0% 0 0+-----|-----|-----|-----|

```

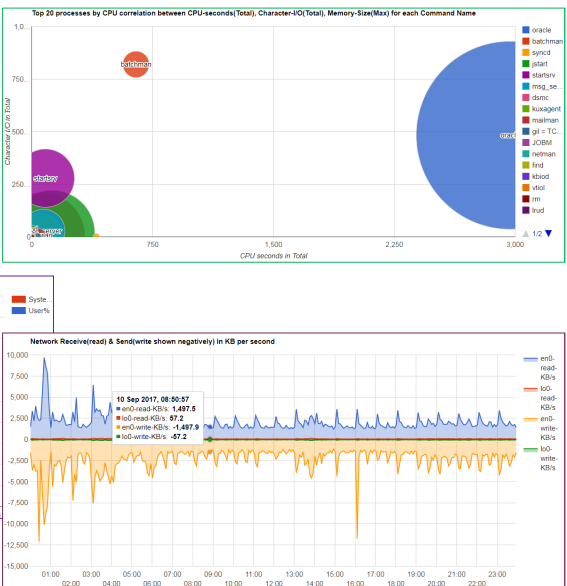
nmon Analyser



nmonchart



Example Graphs



That is enough
about good old “nmon”

nmon is not going away

Lets talk about **njmon**
The **J** is for JSON format

Odd facts:

Jason is the name of my son!

**My next project must be call nhmon
- my daughter is called Hayley!**

nmon
for AIX & Linux

Two modes

- 95% use for post collection graphing
- 5% online investigate Right-Now issue
- nmon analyser = Excel very 1990's
- nmonchart = ksh, new age & fast → .html
- nmon: very good at what it does
and nmon is not going away

Limitations = limited stats,
quirky CSV data format,
static graphs,
data storage management issue
- especially: if doing capacity planning,
no real-time graphing

njmon
for AIX & Linux

- One mode
- 10x the stats
- JSON format = very fast processing via Python
- JSON → Time-Series DB = data management
- Flexible Dynamic Real-time graphing
- njmonchart – in Python, first 24 graphs done
- Running njmon → JSON generated
- Inject in to the Time-Series database
 - Collected JSON file and inject it
 - ssh, data returned by a socket direct injection
 - Transit JSON to a collector daemon for injection
- Real-time browser based graphing engine
- Graphing templates=graphs + select the server
- Alerts – not tried it yet, email if over a threshold

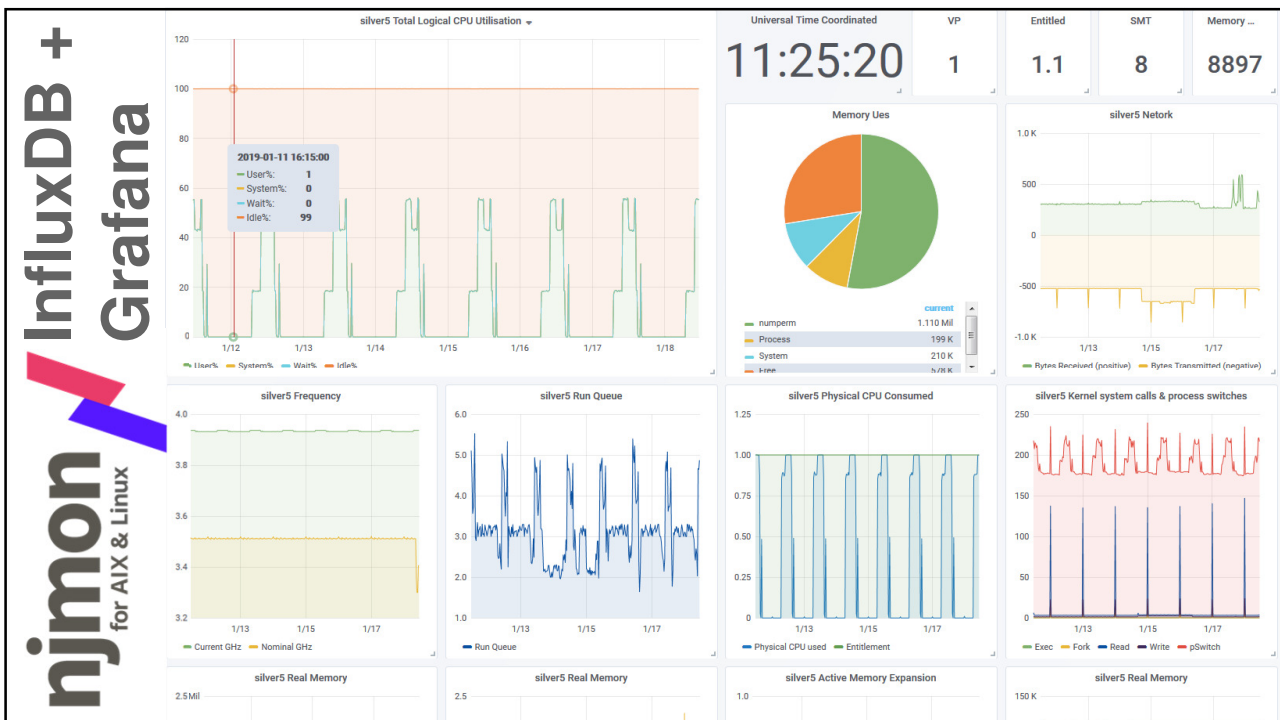
Demonstration 1

Intro to the GUI

Using for example Grafana

Reminder for Nigel

- <http://ultraviolet.aixncc.uk.ibm.com:3000>
- <http://9.137.62.10:3000>





JSON format: JavaScript Object Notation

```
{
  "labelA": "string",
  "labelB": 123456,
  "labelC": 987.654,
  "labelD": true
}
```

Python dictionary:

```
{
  "labelA": "string",
  "labelB": 123456,
  "labelC": 987.654,
  "labelD": True
}
```

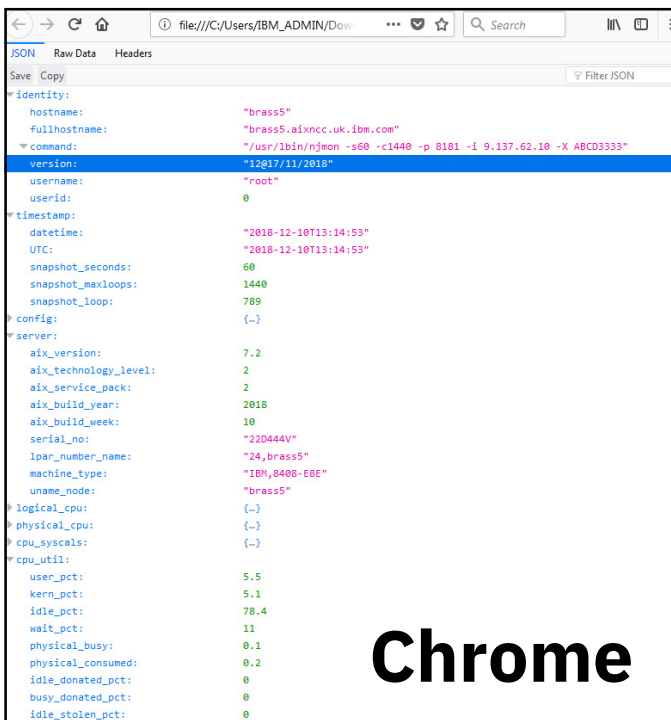
JSON format: JavaScript Object Notation

```
{
  "labelA": "string",
  "labelB": 123456,
  "labelC": 987.654,
  "labelD": true
},
{
  ...
}
```

Python dictionary:

```
D = {
  "labelA": "string",
  "labelB": 123456,
  "labelC": 987.654,
  "labelD": True
}

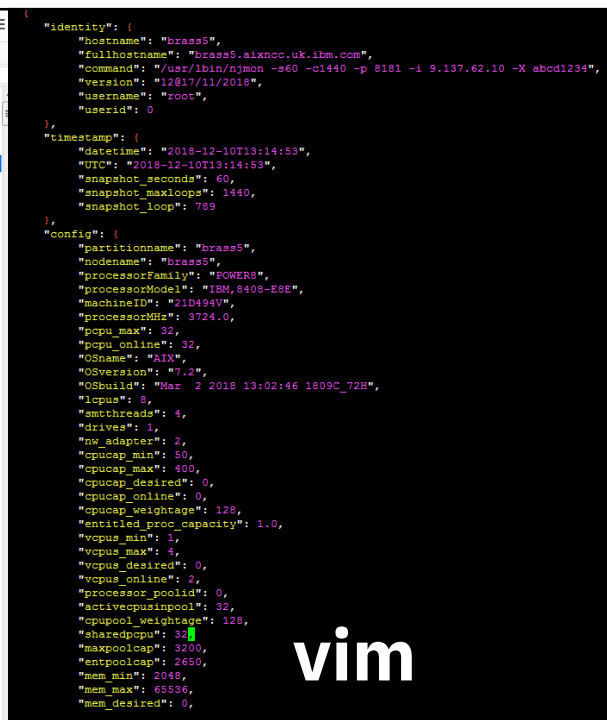
print( D["labelB"] )
123456
```



The screenshot shows a Chrome browser window displaying a JSON file. The file path is `file:///C:/Users/IBM_ADMIN/Down...`. The JSON data is expanded to show the following details:

```
{
  "identity": {
    "hostname": "brass5",
    "fullhostname": "brass5.aixncc.uk.ibm.com",
    "command": "/usr/sbin/njmon -s60 -c1440 -p 8181 -i 9.137.62.10 -X ABCD3333",
    "version": "12817/11/2018",
    "username": "root",
    "userid": 0
  },
  "timestamp": {
    "datetime": "2018-12-10T13:14:53",
    "UTC": "2018-12-10T13:14:53",
    "snapshot_seconds": 60,
    "snapshot_maxloops": 1440,
    "snapshot_loop": 789
  },
  "config": {
    "server": {
      "aix_version": 7.2,
      "aix_technology_level": 2,
      "aix_service_pack": 2,
      "aix_build_year": 2018,
      "aix_build_week": 10,
      "serial_no": "22D444V",
      "lpar_number_name": "24_brass5",
      "machine_type": "IBM,8408-E8E",
      "uname_node": "brass5"
    },
    "logical_cpu": {},
    "physical_cpu": {},
    "cpu_syscalls": {},
    "cpu_util": {
      "user_pct": 5.5,
      "kern_pct": 5.1,
      "idle_pct": 78.4,
      "wait_pct": 11,
      "physical_busy": 0.1,
      "physical_consumed": 0.2,
      "idle_donated_pct": 0,
      "busy_donated_pct": 0,
      "idle_stolen_pct": 0
    }
  }
}
```

Chrome



The screenshot shows the vim editor displaying the JSON data from the previous screenshot, formatted as a Python dictionary:

```
{
  "identity": {
    "hostname": "brass5",
    "fullhostname": "brass5.aixncc.uk.ibm.com",
    "command": "/usr/sbin/njmon -s60 -c1440 -p 8181 -i 9.137.62.10 -X abcd1234",
    "version": "12817/11/2018",
    "username": "root",
    "userid": 0
  },
  "timestamp": {
    "datetime": "2018-12-10T13:14:53",
    "UTC": "2018-12-10T13:14:53",
    "snapshot_seconds": 60,
    "snapshot_maxloops": 1440,
    "snapshot_loop": 789
  },
  "config": {
    "partitionname": "brass5",
    "nodename": "brass5",
    "processorFamily": "POWER8",
    "processorModel": "IBM,8408-E8E",
    "machineID": "21D494V",
    "processorMHz": 3724.0,
    "pcpu_max": 32,
    "pcpu_online": 32,
    "OSname": "AIX",
    "OSversion": "7.2",
    "OSbuild": "Mar 2 2018 13:02:46 1809C_72H",
    "lcpu": 8,
    "smthreads": 4,
    "drives": 1,
    "hw_adapter": 2,
    "cpu_cap_min": 50,
    "cpu_cap_max": 400,
    "cpu_cap_desired": 0,
    "cpu_cap_online": 0,
    "cpu_cap_weightage": 128,
    "emptied_proc_capacity": 1.0,
    "vcpu_min": 1,
    "vcpu_max": 4,
    "vcpu_desired": 0,
    "vcpu_online": 2,
    "processor_poolid": 0,
    "activecpupool": 32,
    "cpupool_weightage": 128,
    "sharedcpu": 32,
    "maxpoolcap": 3200,
    "entpoolcap": 2650,
    "mem_min": 2048,
    "mem_max": 65536,
    "mem_desired": 0,
  }
}
```

vim



njmon is generating JSON

Lots of tools that can handle JSON & graphs
- if you already have one – carry on

Tools I have investigated:

- InfluxDB and Grafana
- ELK stack = ElasticSearch, Logstash, Kibana
- Splunk = well known
- Prometheus = no hope!
- there are many more



These tools:

Data gathering agent(s)

All have a stats database

Many handle odd format log files + analysis

- think syslog, app & RDBMS log, error logs, apache log . . .

All have:

A) Free open source version

- Some optional support

B) Enterprise version

- Typically with Support, HA, Scaling, backup/restore

C) Cloud hosted Service offerings

The choice is yours – so are the costs!



I can't "boil the ocean" !!

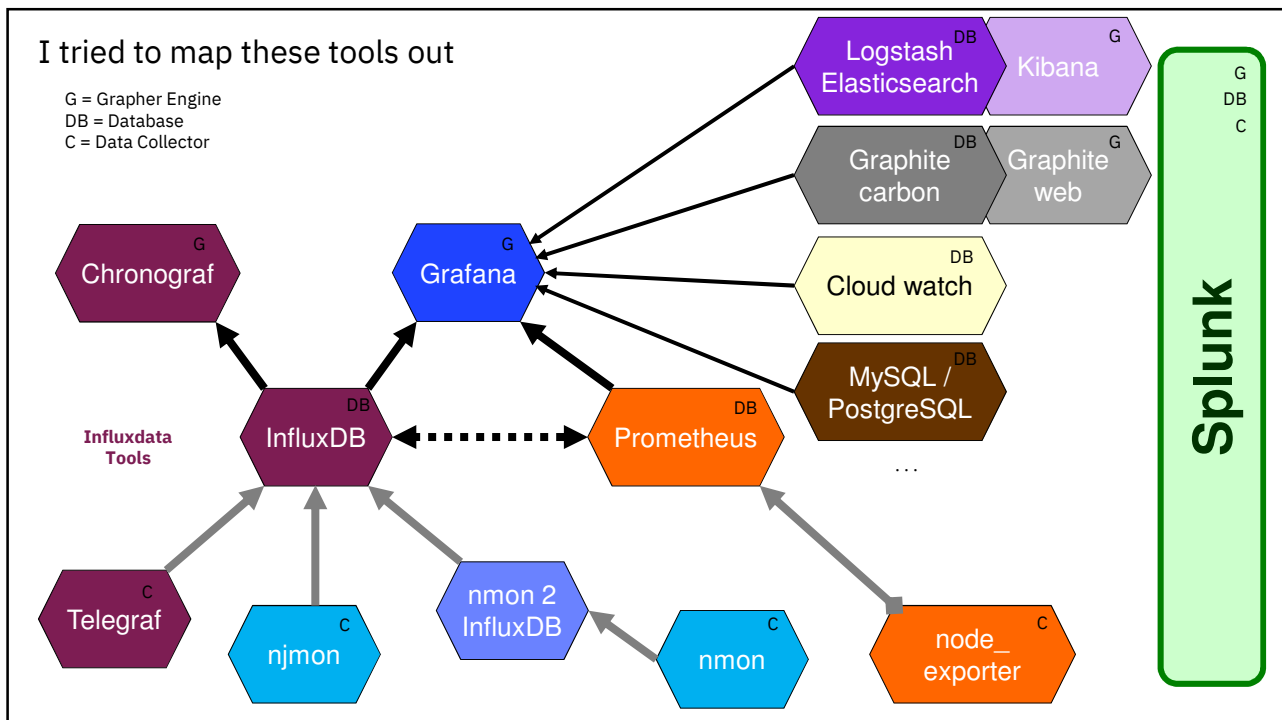
I can't install, manage, test, document
four different Time-Series Databases +
four graphing engines

So I chose one:

open source = cheap
easy to install and run
popular & highly rated on the web
recommended by IBM (MOP)
has Python module to insert stats

= InfluxDB + Grafana

Your criteria will be different





All these tool

Assume Linux on x86_64 or ARM (Raspberry Pi)

As a POWER guy Linux is OK but
Intel not acceptable ☹!
I have large POWER servers &
want to use them.

This needs fixing:

- InfluxDB compiled OK on Linux on POWER
- needed to first compile Golang (go)
- Grafana - Ran out of time for today, soon

FUD Reduction

Very easy install on
x86_64
or Raspberry Pi B

The 6 minute Install challenge:

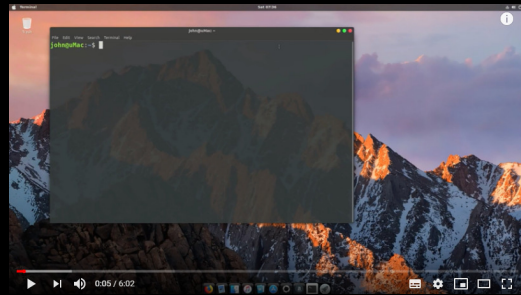
Tips For IT Pros

Nice YouTube Video that does it in well under in 6 minutes

<https://www.youtube.com/watch?v=hJSBgZlWwmE>

I do not recommend taking the Ubuntu Repository InfluxDB 1.1 but rather use the latest 1.7.3+ from influxdata.com

Install InfluxDB + Grafana on Ubuntu 18.04 on x86_64



InfluxDB download details

<https://portal.influxdata.com/downloads/> →

Prep Ubuntu 18.04 on x86_64

-- Get the timezone & date correct

-- Update Ubuntu

sudo apt update ; sudo apt upgrade

Get email working for Grafana

-- Install

\$ wget https://dl.influxdata.com/influxdb/releases/influxdb_1.7.3_amd64.deb

\$ sudo dpkg -i influxdb_1.7.3_amd64.deb

-- Setup a database for the njmon data

\$ influx

> create database njmon

> show databases

> exit



2 minutes

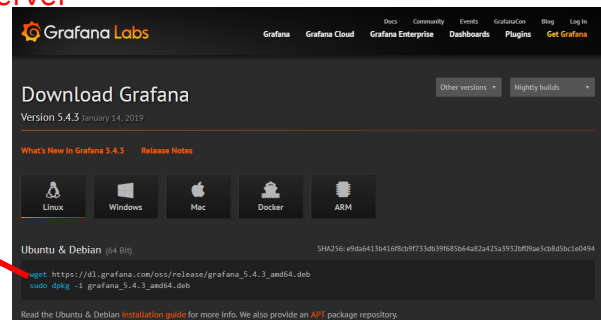


Assuming InfluxDB & Grafana on the same server

Grafana

<https://grafana.com/grafana/download>

```
$ wget https://dl.grafana.com/oss/\
release/grafana_5.4.3_amd64.deb
$ sudo dpkg -i grafana_5.4.3_amd64.deb
```



Config file `/etc/grafana/grafana.ini` → all the defaults worked for me

-- Set Grafana to start on reboots

```
$ sudo /bin/systemctl daemon-reload
```

```
$ sudo /bin/systemctl enable grafana-server
```

-- Start now

```
$ sudo /bin/systemctl start grafana-server
```

Next connect Grafana to your data source InfluxDB → next slide

1 minute



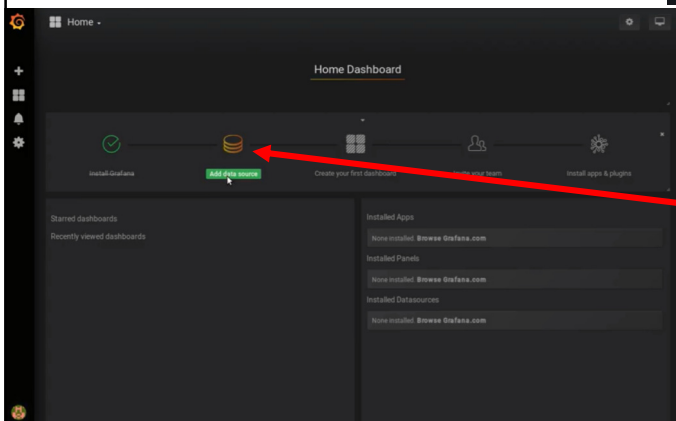
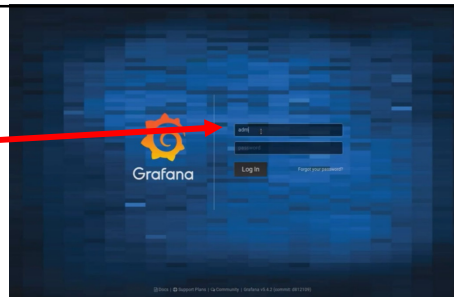
Assuming hostname is myserver.com

<https://myserver.com:3000>

Default user is admin / admin

It will demand you change password

I am using SECRET



← Grafana home page

Tasks

- Install Grafana DONE
- Add Data Source **[click now]**
- Create Dashboard
- Invite users
- Install apps & plugins

InfluxDB is our data source

1 minutes

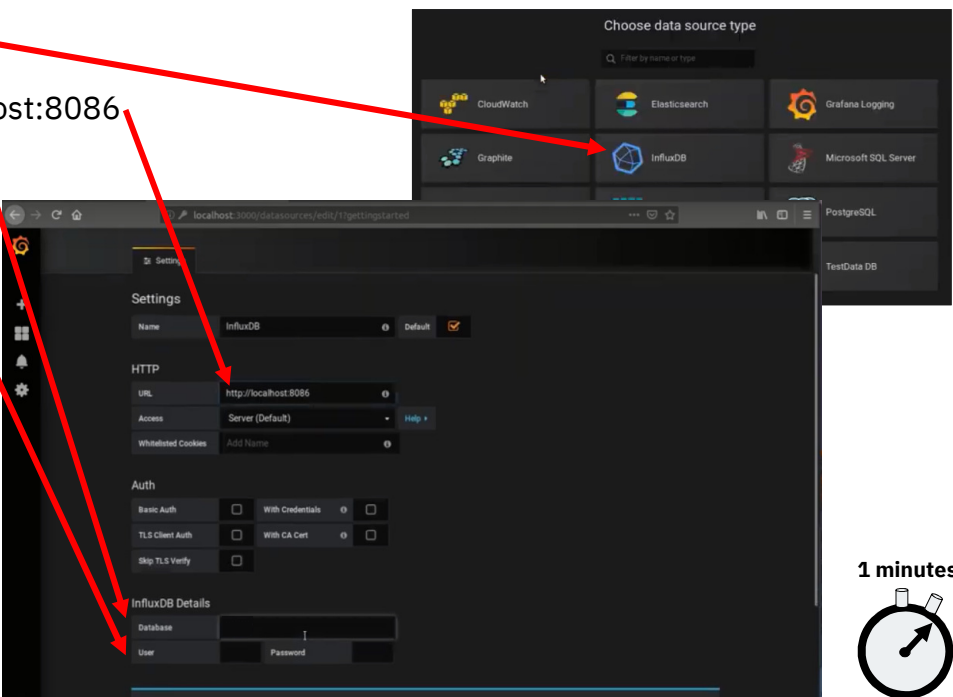



Select InfluxDB

URL = http://localhost:8086
 Database = njmon
 User admin
 Password SECRET

Scroll down to Save & Test

It will pop a confirmation that it is logged into InfluxDB



1 minutes 

Now you are ready to create a dashboard of graphs

BUT hold on you need some data!!

Enter njmon 😊

Recommended minimum: A whole day's worth of stats

njmon CLI	<pre>njmon -s 60 -c 1440 >x.json</pre>	Redirect mode
	<pre>-s seconds (default 60)</pre>	
	<pre>-c count of the number of captures (default don't end)</pre>	
	<pre>njmon -s 60 -c 1440 -m /home/perf -f</pre>	Save to file mode
	<pre>-m directory for output</pre>	
	<pre>-f generate a filename for output: hostname_dateTtime.json</pre>	
	<pre>Pipe straight to injector in to Time-Series DB*</pre>	Local pipe to DB mode
	<pre>njmon -s 60 -c 1440 injector.py</pre>	
	<pre>From the Time-Series DB* server to the target</pre>	Remote ssh to DB mode
	<pre>ssh nigel@myserver /lbin/njmon -s 300 -c 288 /lbin/injector.py</pre>	
		Send to collector mode
	<pre>Start the collector on the Time-Series DB* node once:</pre>	
	<pre>njmon_collector -p 8181 -d /home/janet -X Beetlejuice -i -c injector42.py</pre>	
	<pre>Send the data on the collector</pre>	
	<pre>njmon -s 300 -c 288 -i collectorhost.com -p 8181 -X Beetlejuice</pre>	
	<p><small>* InfluxDB or other time-series database with Python class library</small></p>	

Under the covers

- njmon for Linux
- Simple C code – small binary
 - simpler than nmon
- “p” functions generate JSON
 - Buffered output
- Typical Linux stats from all over the place!!
 - /proc & others
 - Every file a different format
 - + command like lscpu etc.
- To be added process level stats

Under the covers

- njmon for AIX
- Simple C code – small binary
- “p” functions to generate JSON
 - Buffered output
- Most data from the excellent AIX libpertsstat library – clean
- Includes 100’s of extra stats on the VIOS for virtual disks/nets
- Has process stats



Alternative: to simply graph a njmon .json file

Release on Monday 21st Jan

- It is a prototype ~ 400 lines of code (12 lines per graph)
- Can probably reduce that to 100 lines
- Add in the fancy nmonchart graphs

Allows bulk graphing 100's of njmon files

- Could build a website around it

Use for one-off investigation of a few VM or whole Server



Quick way to graph a single .json file to a webpage

njmonchart.py ← written in Python3

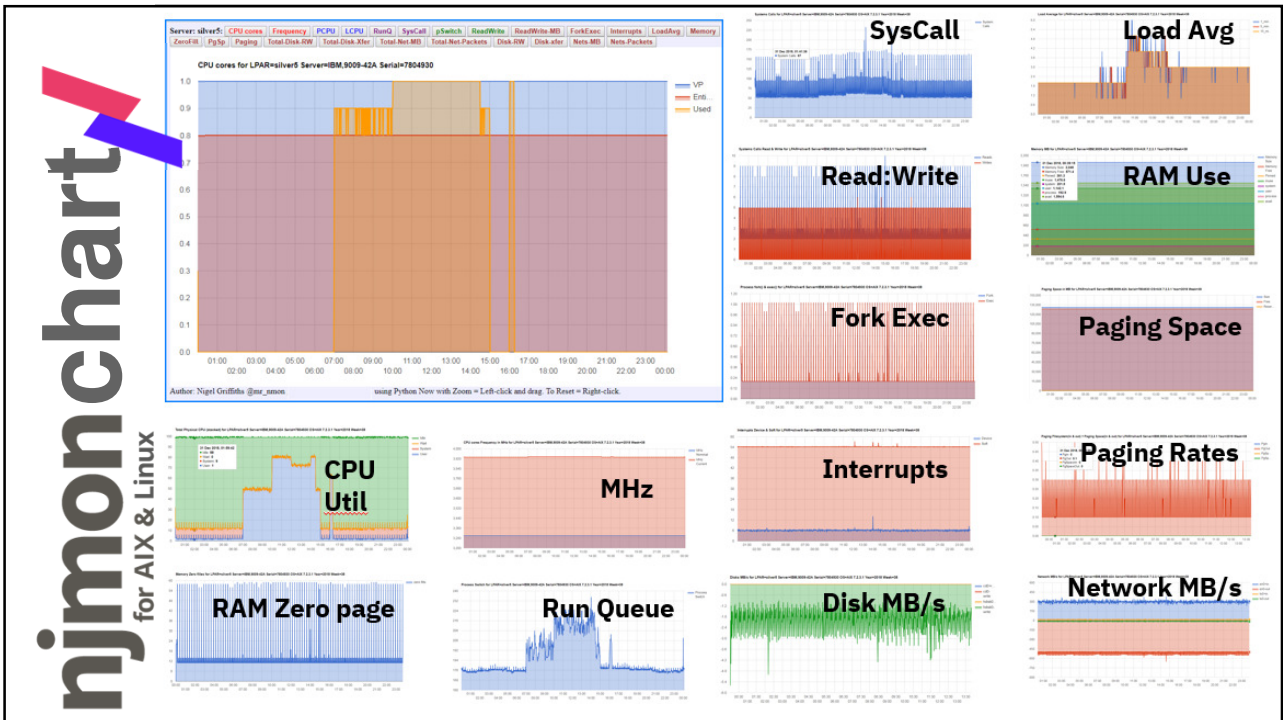
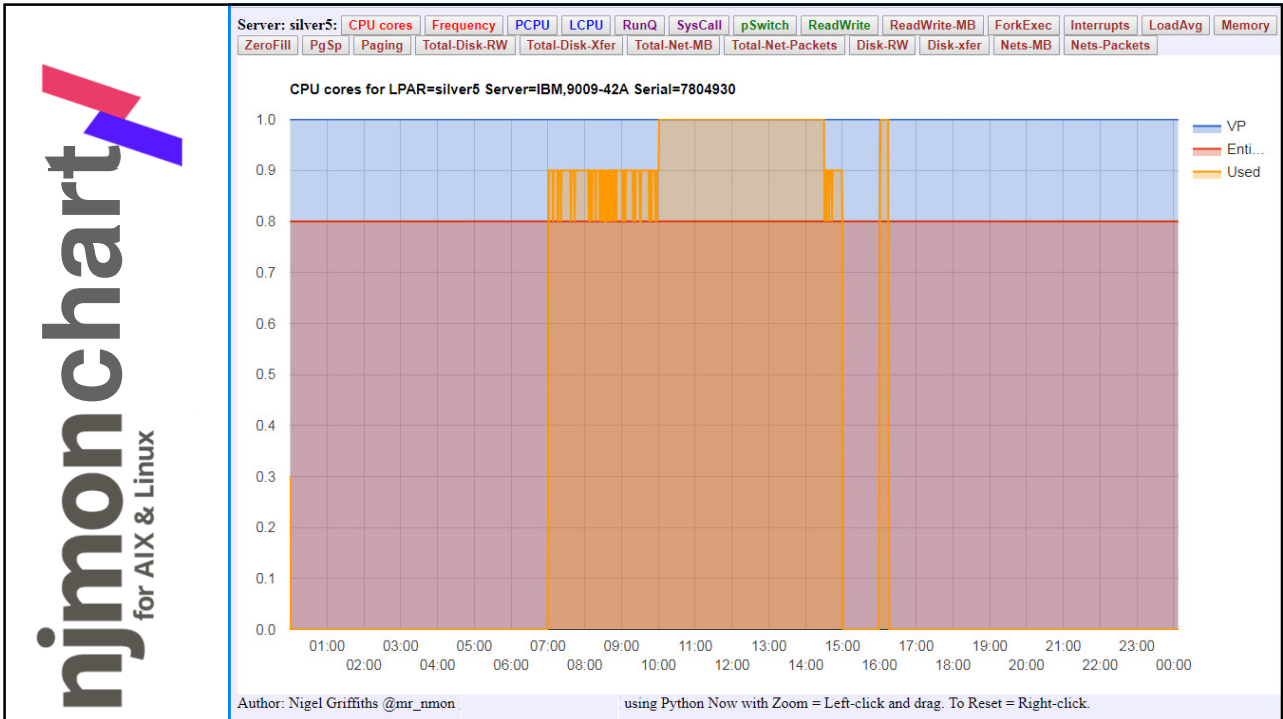
- Requires nchart.py class module (supplied too)
- \$ njmonchart myfile.json myfile.html

Covers first 24 charts/graphs

And it's fast:

- JSON file with 1440 data point =41 MB generates the web page (.html) of graphs in 1.03 seconds

1. CPU cores,
2. Frequency,
3. PCPU,
4. LCPU,
5. Run Queue,
6. System Call,
7. Process Switch,
8. Read Write,
9. Read Write-MB,
10. Fork Exec,
11. Interrupts,
12. Load Avg,
13. Memory,
14. Memory Zero-Fill,
15. Page Space,
16. Paging,
17. Total-Disk-RW,
18. Total-Disk-Xfer,
19. Total-Net-MB,
20. Total-Net-Packets,
21. Disk-RW,
22. Disk-xfer,
23. Nets-MB,
24. Nets-Packets



Demonstration 2

Example: njmonchart output

Get the code from

<http://nmon.sourceforge.net/pmwiki.php?n=Site.Njmon>

Hint to Nigel: Tools_NEW/njmonchart/silver5-2018-12-31.html

Back to the main plot

Time-Series databases

and

Real-time graphing

What does the injector do?

- Logon to InfluxDB or Splunk
 - Loads the JSON into a dictionary
- Adds each snapshot and all its data to a measure dictionary
Data agnostic it just sends the data it finds in the file = flexible
 - Writes it to the Time-Series DB
 - Go to 1

```

from influxdb import InfluxDBClient ← load library
dbname = 'njmon'
client = InfluxDBClient('localhost', 8086, 'nigel',
                        'passw0rd', dbname)

entry = [] # empty list
fields = {} # empty dictionary

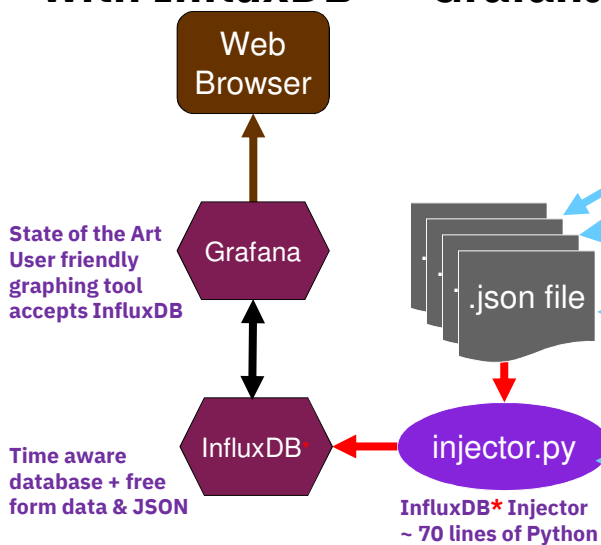
for sample in jsondata ← loop through the dictionary
    host = sample['hostname']
    dt = sample["datetime"]
    fields = sample['cpu_util'] (includes usr, sys, wait + idle)

    measure = {'measurement': 'CPU_util',
              'tags': {'host': host },
              'time': dt,
              'fields': fields }

    entry.append(measure) ← add to list
client.write_points(entry) ← send to InfluxDB

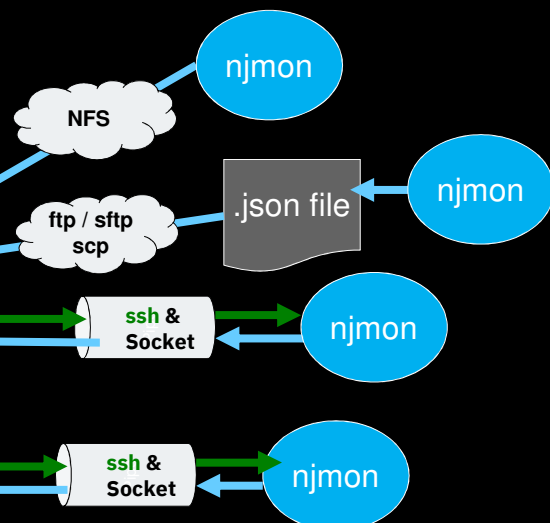
```

njmon infrastructure with InfluxDB* + Grafana



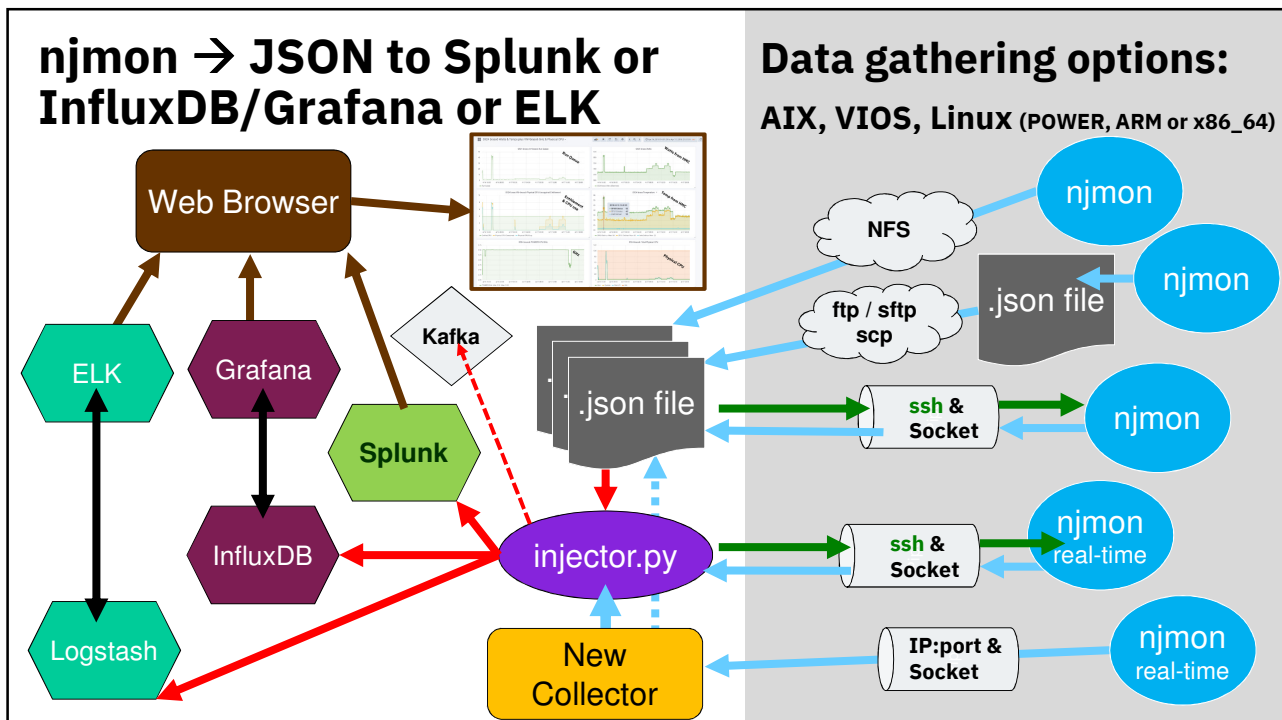
* InfluxDB or other time-series database with Python class library

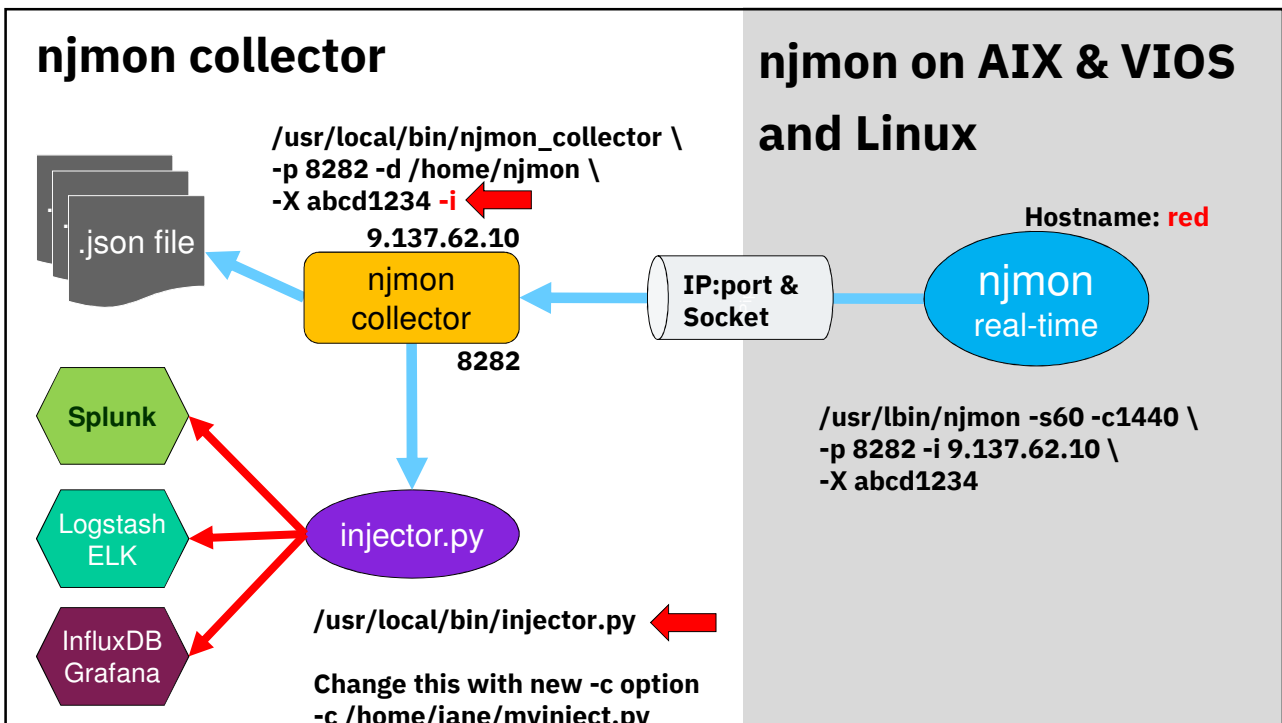
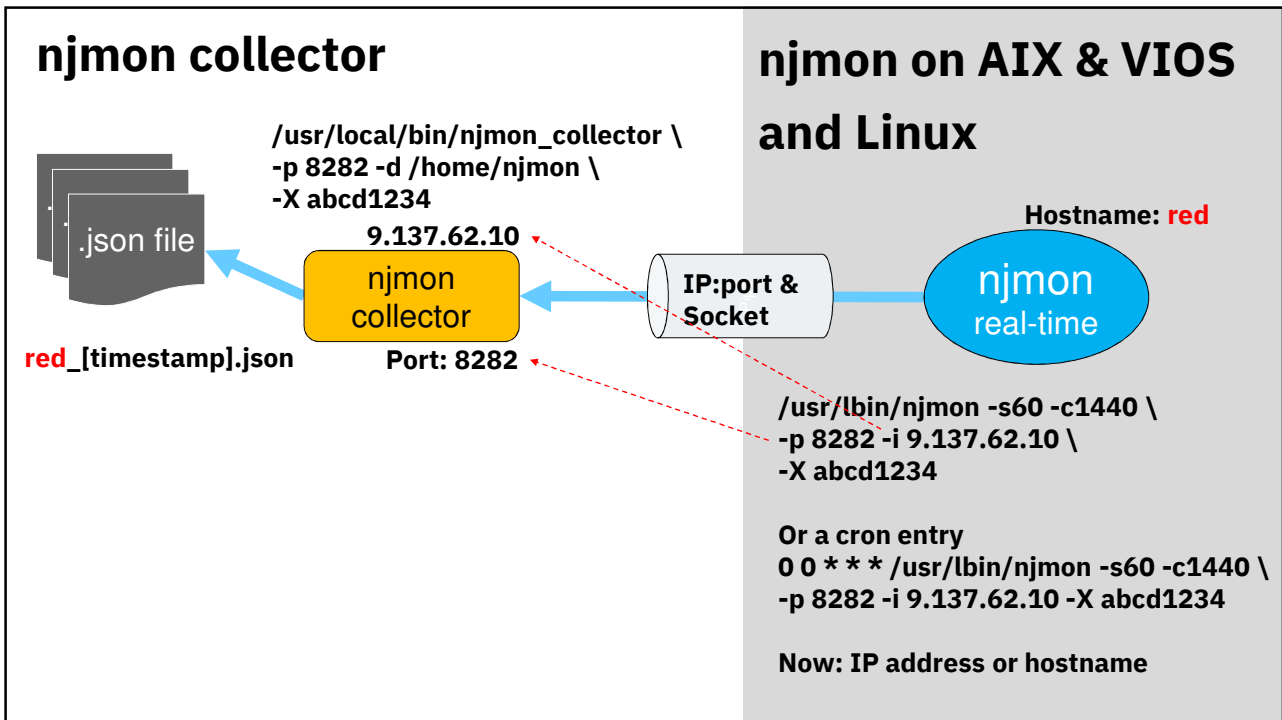
Remote data gathering options



By the way:
njmon JSON files
compress very well

32 to 1





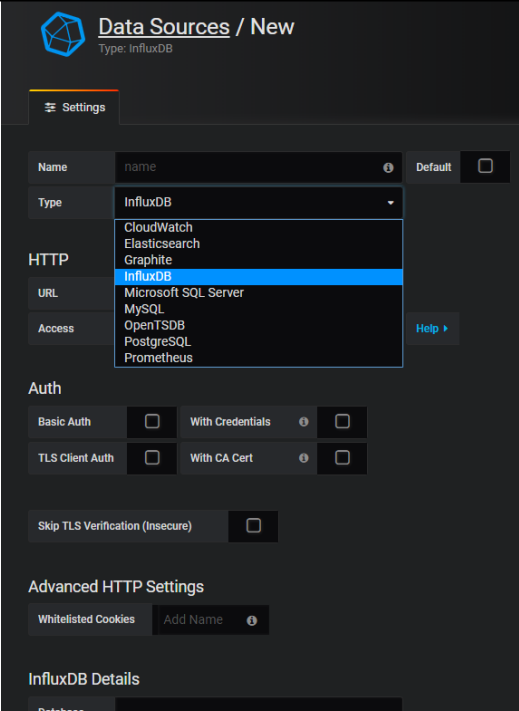
Demonstration 3

Graphing with Grafana

Hint to Nigel:

<http://ultraviolet.aixncc.uk.ibm.com:3000>

<http://9.137.62.10:3000>



Data Sources / New
Type: InfluxDB

Settings

Name: name Default

Type: InfluxDB

HTTP

URL:

Access:

Auth

Basic Auth: With Credentials:

TLS Client Auth: With CA Cert:

Skip TLS Verification (Insecure):

Advanced HTTP Settings

Whitelisted Cookies: Add Name

InfluxDB Details

Database:

User: Password:

Min time interval: 10s

Save & Test Back

Connecting Grafana to a data source Here InfluxDB

Advanced HTTP Settings

Whitelisted Cookies: Add Name

InfluxDB Details

Database:

User: Password:

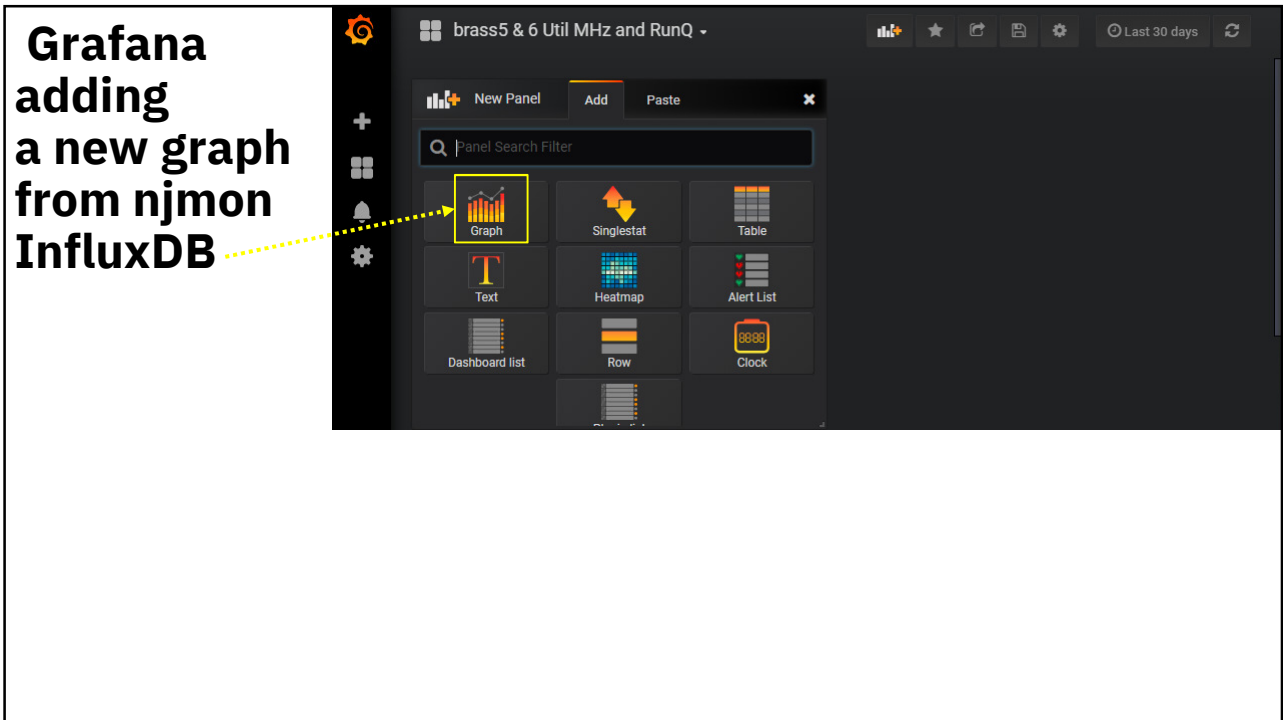
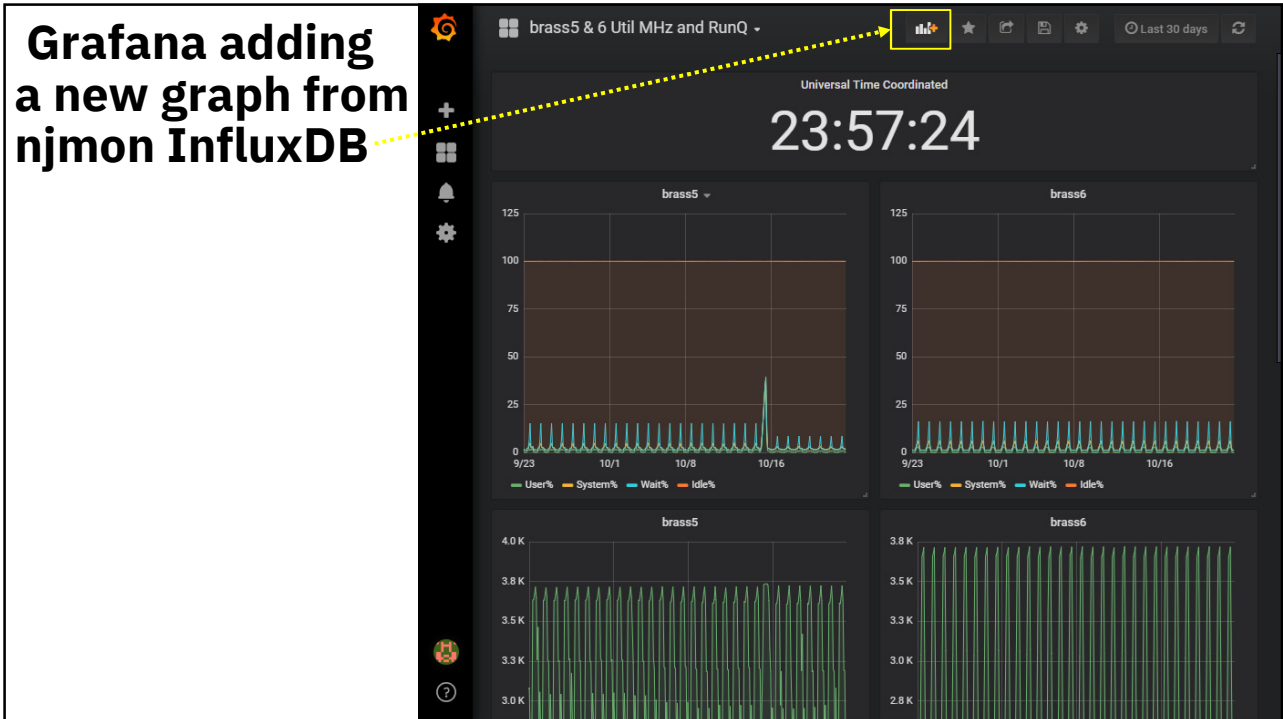
Database Access

Setting the database for this datasource does not deny access to other databases. The InfluxDB query syntax allows switching the database in the query. For example: `SHOW MEASUREMENTS ON _internal` or `SELECT * FROM "_internal".."database" LIMIT 10`

To support data isolation and security, make sure appropriate permissions are configured in InfluxDB.

Min time interval: 10s

Save & Test Back



Grafana adding a new graph from njmon InfluxDB data

Data in InfluxDB

Category

Host:brass5

Stats

The screenshot shows the Grafana query editor with the following configuration:

- Data Source:** njmon
- FROM:** default
- WHERE:** host = brass5
- SELECT:** field(runqueue) mean()
- GROUP BY:** time(\$interval) fill(null)
- FORMAT AS:** Time series
- Alias:** Run Queue

Grafana adding a New Title

Graph already running



Grafana adding a New Title



Grafana Legend And Display

The top screenshot shows the "Legend" configuration panel for the "brass5 Run Queue" graph. The "Legend" tab is highlighted with a yellow box. The panel is divided into three sections:

Options	Values	Hide series
Show	Min	With only nulls
<input checked="" type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
As Table	Max	With only zeros
<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
To the right	Avg	
<input type="checkbox"/>	<input type="checkbox"/>	
	Current	
	<input type="checkbox"/>	
	Total	
	<input type="checkbox"/>	
	Decimals	
	auto	

The bottom screenshot shows the "Display" configuration panel for the same graph. The "Display" tab is highlighted with a yellow box. The panel is divided into several sections:

Draw options	Draw Modes	Mode Options
Series overrides (0)	Bars	Fill
	<input type="checkbox"/>	1
Thresholds (0)	Lines	Line Width
	<input checked="" type="checkbox"/>	1
	Points	Staircase
	<input type="checkbox"/>	<input type="checkbox"/>
		Point Radius
		5

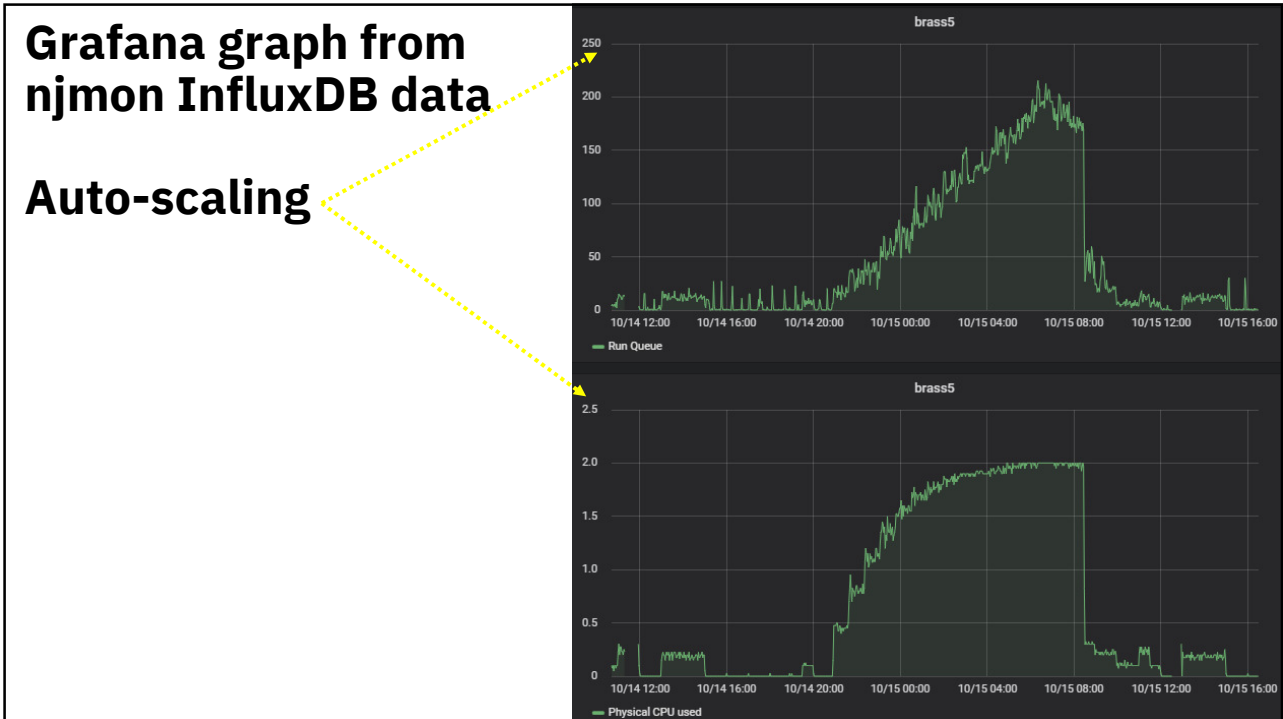
Hover tooltip	Stacking & Null value
Mode	Stack
All series	<input type="checkbox"/>
Sort order	Null value
None	null

Grafana the graph



Grafana graph from njmon InfluxDB data





njmon for AIX & Linux

InfluxDB 1.7 + Grafana 5.3

Added two Grafana plugins

The image shows a grid of Grafana dashboard plugins. The plugins are arranged in three rows and three columns. The first row contains 'Graph', 'Singlestat', and 'Table'. The second row contains 'Text', 'Heatmap', and 'Alert List'. The third row contains 'Dashboard list', 'Row', and 'Clock'. Below the grid, there are two green arrows: one pointing to the 'Pie Chart' plugin (which is located below the 'Dashboard list' and 'Row' plugins) and another pointing to the 'Clock' plugin.

njmon
for AIX & Linux

Grafana 5.3

1 Settings then Variables

2

3 "host" is one of the TAGs added by the Injector

4 Now you have a host selector

5 when you set: host = \$HOST

Variables > Edit

General

Name HOST Type Query

Label host Hide

Query Options

Data source njmon14 Refresh Never

Query SHOW TAG VALUES WITH KEY = "host"

Regex /[a-z]*/

Sort Alphabetical (asc)

Selection Options

Multi-value

Include All option

Value groups/tags (Experimental feature)

Enabled

Preview of values

IBM-Southbank blue brass5 brass6 pi.aixncc.uk.ibm.com silver1
silver2.aixncc.uk.ibm.com silver3 silver4 silver5 sky ultraviolet

Update

Graph

Data Source njmon14

FROM default total_logical_cpu WHERE host = \$HOST

SELECT field (user) mean ()

GROUP BY time (\$__interval) fill (null)

FORMATTED AS

ALIAS BY User%



Other stats:

- nextract - HMC REST API for perf stats
 - Temperature
 - Some Perf stats at LPAR & Server level
 - SSP stats
- Raspberry Pi computer room temperatures
 - see AIXpert Blog
- nmon2json (still data problem & non real-time)

Other Wacko ideas:

- njmon Hack-a-thon at TechU's
- AI investigation of njmon data via Python
- nmon = nmon + njmon



Browse to

<https://tinyurl.com/njmon>

= <http://nmon.sourceforge.net/pmwiki.php?n=Site.Njmon>

Download:

- njmon for AIX, VIOS and Linux
- njmonchart.py (currently AIX only)
- njmon collector
- njmon injector.py for InfluxDB & Splunk
- Grafana Dashboard Template

Decide your DB & Graph tools!
Or use InfluxDB + Grafana for a prototype

Add njmon to your virtual machines & VIOS

→ Start with snapshots once a minute (or 30 seconds)

The screenshot shows the njmon website interface. On the left is a sidebar with the following links:

- nmon page
- Screen shots
- Download Binaries
- Source Code
- Documentation
- nmon FAQ
- njmon
- Save to JSON format
- Other tools
- nmonchart
- TOPASchart
- nmon2json Format
- nweb Tiny Webservier
- nmonmerge
- Join nmon files

The main content area features the njmon logo and the heading: **njmon is nmon but saving to JSON format - for modern performance stats tooling**. Below this is a 'Briefly' section with the following text:

njmon is like nmon but collects a lot more performance and configuration data and outputs in JSON format ready for immediate uploading to a performance stats database for near real-time graphing by online graphing tools. There is a version of njmon for AIX and another njmon for Linux.

- **njmon for AIX** uses the AIX libperfstat to extract the performance stats
 - Roughly: 550 stats for AIX, an additional 55 for the Virtual I/O Serve and a further 35, if running a VIOS Shared Storage pool.
 - Further VIOS stats are to be expected soon.
- **njmon for Linux** the performance stats come mostly from /proc filesystem and system calls.
 - Roughly: 130 stats - possible future stats include: Fibre-Channel SAN, GPFS, GPU and Process stats.
 - Ideas welcome, if you know where the stats can be found on Linux.

Below the text is a diagram titled 'Why njmon?' with several thought bubbles:

- Can't change nmon file format much! Due to other tools.
- nmon limited # of tuning stats, njmon 100's
- 1000's of nmon file creates a data management issue
- njmon near real-time injection to a stats DB
- JSON processing is easy as it is a Python dictionary. Load fast too.
- Graph with dynamic online tools (browser) & merge with other stats
- JSON=self documenting & handles extra new data on the fly

At the bottom of the page, there is a note: *I will be adding more content and explanations here - let me know what is not well explained.*



Compiling:

njmon for Linux, AIX, VIOS (2.2 & 7.1)
& njmon_collector

1 Download the code = 3 files in total

2 Install GCC GNU C compiler - 2 minutes

- Linux = easy & now simple on AIX

3 \$ gcc -O4 njmon_aix.c -o njmon

- takes 1 second -O4 → maximum optimisation

4 VIOS: compile on AIX6.1.9 or AIX 7.2.3

- add the command line options: -D VIOS -D SSP

injector.py & njmonchart.py need Python3

- Linux easy to install Python

- Python now available on AIX (not tested that yet)



Report errors, fixes, new stuff, ideas:

- duff data

- your enhancements to njmonchart

- injector for ELK

- Grafana Dashboards for Linux

to Nigel via **email** or

DeveloperWorks: Performance Tools Forum

Add other data sources:

- HMC "nextract" – see AIXpert Blog

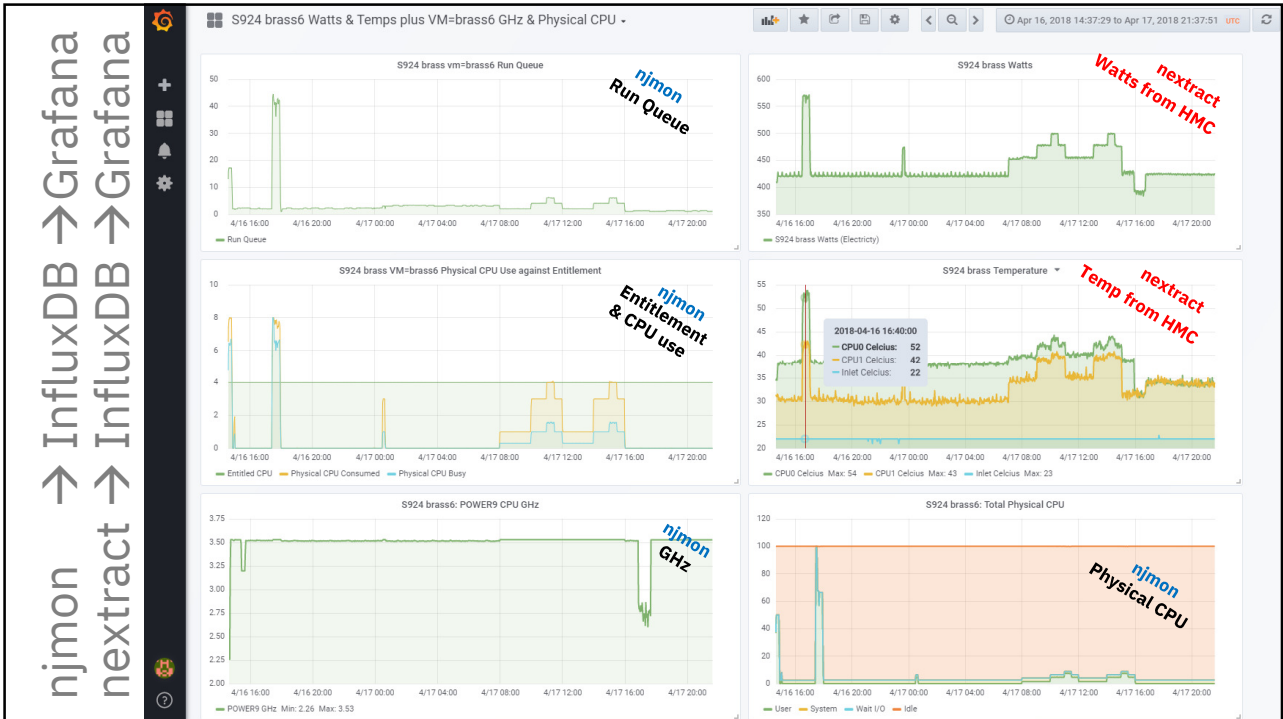
→ temp C, Watts, LPAR & Server stats

- Raspberry Pi temperatures

- Your application stats like:

transactions/sec, online users,

web hits/sec, batch start+stop

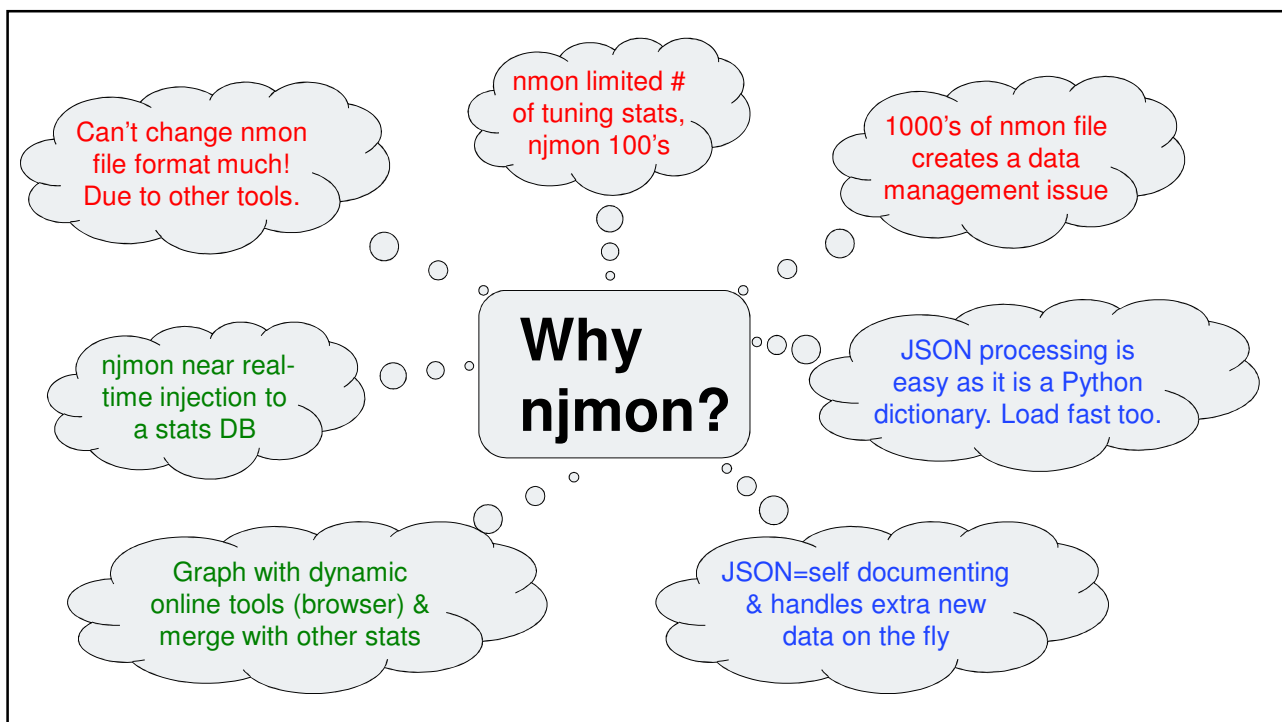


Questions



njmon is nmon but saving to JSON format for modern performance stats tooling ++

Backup slides



nextract

Temp & Watts

Extraction for

POWER8 S822/S824 + E850

Also good on

POWER9 S922/S924 + E950

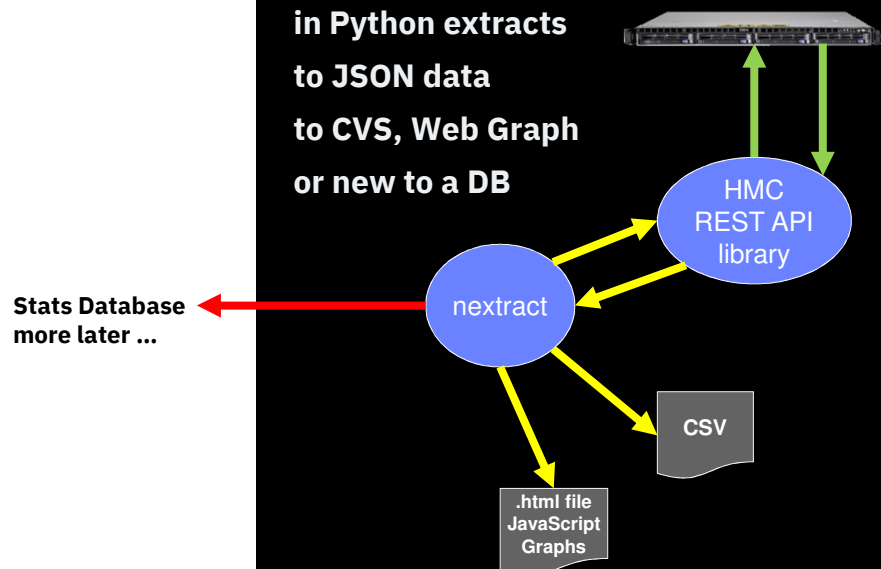
The HMC REST API is developer hostile = unusable from the docs

- So now hidden in a Python library
- Now extracting the stats is 10 - 20 simple lines of Python
- + 20 lines of Python to save as
- For CSV files or direct to graphs or direct in to online tools: InfluxDB
- Also covers Server or VM performance stats

nextract

HMC REST API
in Python extracts
to JSON data
to CVS, Web Graph
or new to a DB

Stats Database
more later ...



nextract

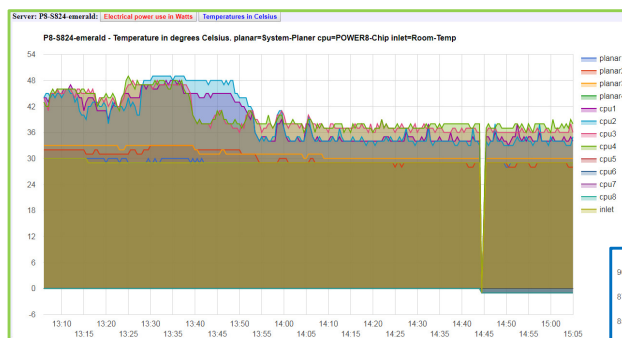
HMC Rest API Examples

Get the Python library to extract the stats from the HMC REST API + format the output from:

Nigel AIXpert Blog Entry
POWER8/9 Watts, Temp, SSP I/O & Server/LPAR stats from HMC REST API - Version 3

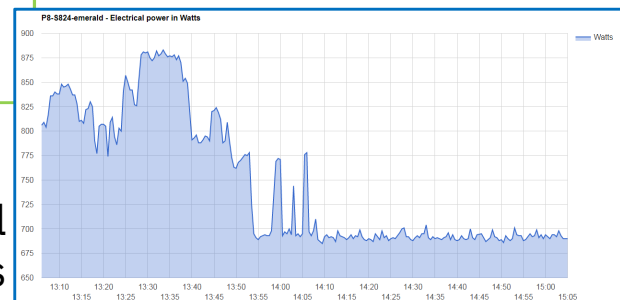
<http://www.nigel-aixpert.com/2018/01/10/hmc-rest-api-version-3/>

Extracted HMC REST API Energy data as Google-chart graphs

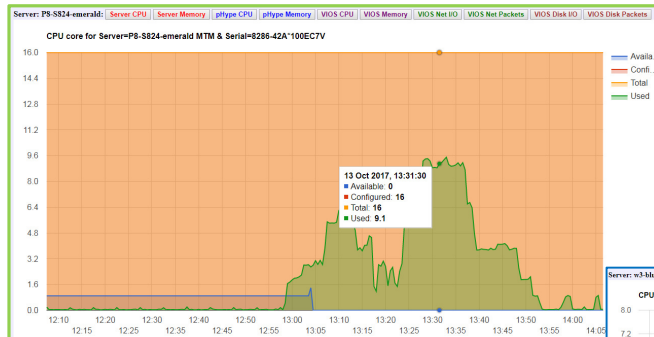


Temperature Celsius

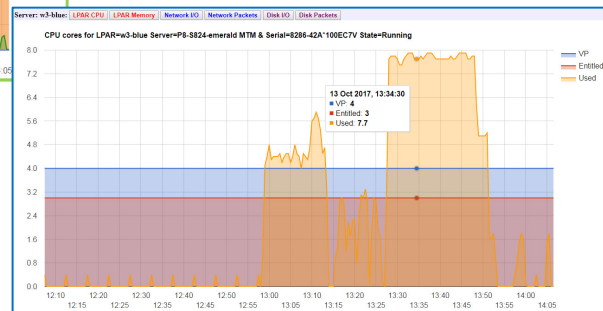
Electrical
Watts



Extracted HMC REST API Perf. Stats. as Google-chart graphs



Server level



LPAR level

STOP PRESS:
nmon for Linux

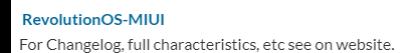
SourceForge
Project of the Week
17th Dec 2018

1 of 9 projects of the week

Projects of the Week, December 17, 2018

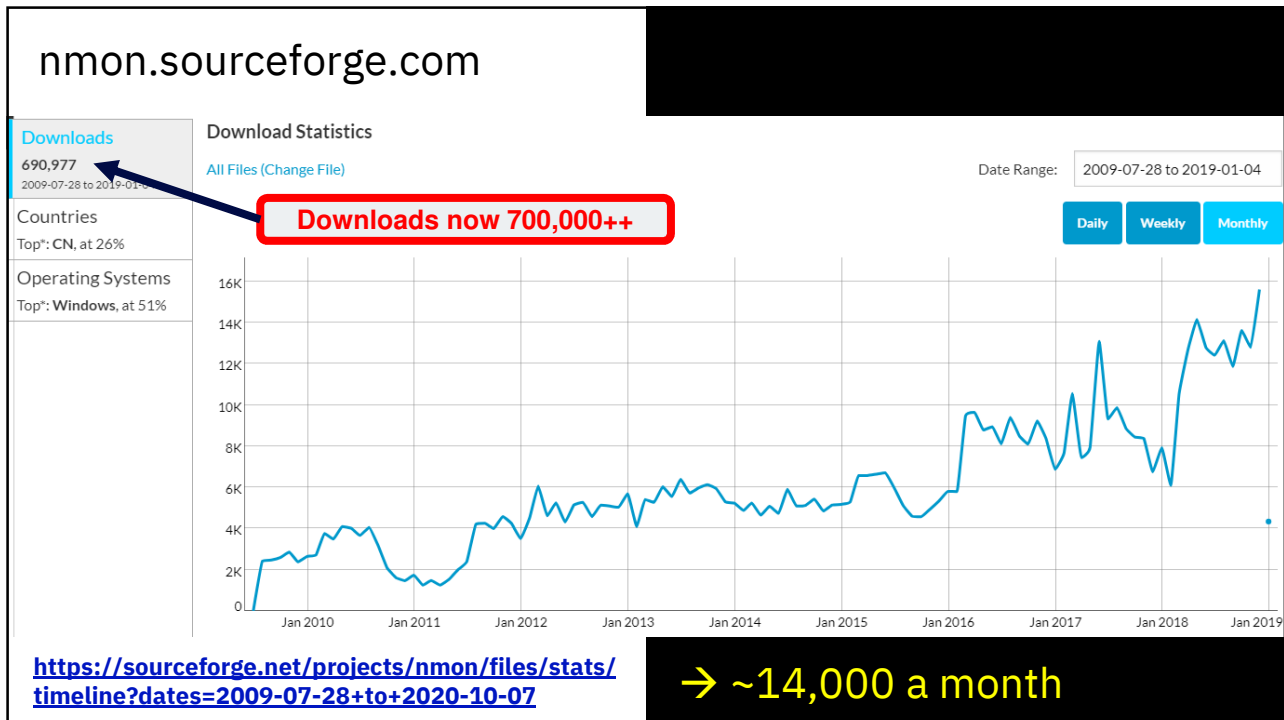
By Community Team December 17th, 2018

Here are the featured projects for the week, which appear on the front page of SourceForge.net:



nmon for Linux

nmon is short for Nigel's Performance Monitor. It either shows you the stats on-screen updating once a second or saves the data to a CSV file for later analysis and graphing. For details see the homepage ==> <http://nmon.sourceforge.net> Please use the latest version.



Lets talk about **nmon**

nmon for AIX – “old school”

22 years ago
 - uni-processor
 - 66 MHz
 - 64 MB RAM
 - 30 GB disks

Had to be ruthless in
 - code efficiency C
 - data compactness CSV
 - minimum disk I/O
 - only tunable stats
 - spreadsheet size limits
 - graphs are Excel Visual Basic (ugh)
 -- until nmonchart took over

If you have many servers then
 you have data organisation issues

nmon for AIX and Linux “next generation”

- CPUs x 200,1000 faster,
- RAM x1 million larger
- Network x 4000 rate
- Disks cache and SSD – 500,000 times larger
- nmon file format = quirky!

Time to rethink

1. Use industry standard format
2. Capture data centrally over a network
3. Use new tools for data handling
4. Use new tools for LIVE graphing

industry standard format

Long hard search

- XML – ugly and verbose

After self taught myself Python

I learnt

JSON format is a Python dictionary

- Python quickly loads 10MB JSON file
- Then native language syntax to access the data

How cool is that!!

njmon for AIX

For AIX uses **libperfstat** C library
 see `man libperfstat`
 or `vi /usr/include/libperfstat.h`
 or find the worked example code

Status quirky but useable for expert
 C programmer

Vast quantity of perf stats

Current njmon extracts
~600 stats per snap shot

(if you have many disks, nets then that grows rapidly)
 libperfstat gives us the current MHz



njmon for Linux

For Linux – oh dear! Some in **/proc**
 Stats are scattered across text files

Following nmon for Linux code to see what
 is available & where and units

Status=not pretty. Perf stats is not a Linux
 strength but we can get by (due to nmon)

Currently njmon extracts

~200 stats per snap shot

(if you have many disks, nets then that grows rapidly)



njmon syntax

`njmon -s seconds -c count`
`-m directory`
`-f`

Lines of C code

- AIX = 3000
- Linux = 2000 (process stats soon &
 fc stats,
 maybe GPU & GPFS)

`-s seconds` = time between snapshots
`-c count` = number of snapshots
`-m directory` = changes to this directory
 as it starts (good for crontab)
`-f` saves to a file(s):
`<hostname>_date_time.json` and
`<hostname>_date_time.err` for errors

Defaults are: `-s 60 -c forever` and output
 to standard out (screen) so you can
 directly pipe into down stream tools in
 near real-time

Don't reinvent the wheel

Use new tools for
- data handling
- graphing

Save over the network ... live

Save everything to a stats DB
 (open source = cheap)

Use web base dynamic graphing
 tools pulling live data from the
 DB

POWER Servers + PowerVM with AIX or Linux

Set the "Allow Performance Data Collection" flag on all the LPARs

For example: Shared CPU Pool Idle stats

Run this script on the HMC, it takes ~4 seconds per LPAR

From script guru Gareth Coates

```
lssyscfg -r sys -F name | while read M
do
  lssyscfg -r lpar -m $M -F lpar_id | while read L
  do
    echo chsyscfg -m $M -r lpar -i "lpar_id=${L},allow_perf_collection=1"
    chsyscfg -m $M -r lpar -i "lpar_id=${L},allow_perf_collection=1"
  done
done
```



InfluxDB open source
distributed time-series database

<https://www.influxdata.com/>

Download from: <https://portal.influxdata.com/downloads>
- influxdb_1.6.0_amd64.deb

Also need: python3-influxdb 4.1.1-2- Python library for
direct data uploads



The leading open
source software for time
series analytics

<https://grafana.com/>

Download via wget

<https://grafana.com/grafana/download>

- grafana 5.2.2

Once installed go to

<http://<hostname>:3000/>



InfluxDB + Grafana

Cloud based service

<https://corlysis.com/>

- But low price
- Starts at zero €\$£
- Hobbyist €2.49 / month

Also available as a service

- Not tried this myself
- Zero install
- I have no connections with Corlysis what-so-ever
- If you try, let me know how it goes

Any there other online tools?

Yes there are ...

1. Splunk
 2. ELK (Elastic Search + Logstash)
 3. Apache Kafka
- and ...

Hang on! I only have 24 hours a day & njmon is not my day job!

splunk >

About Splunk

- Mostly open source
- Run your own (on prem)
- Cloud service €\$€
 - Time limited free cloud access
- Enterprise scale support €\$€
- Function rich so fairly complex with many Apps & plug-ins

Work the Way Your Data Works



Real-Time

Splunk gives you the real-time answers you need to meet customer expectations and business goals.



Machine Data

Use Splunk to connect your machine data and gain insights into opportunities and risks for your business.



Scale

Splunk scales to meet modern data needs — embrace the complexity, get the answers.



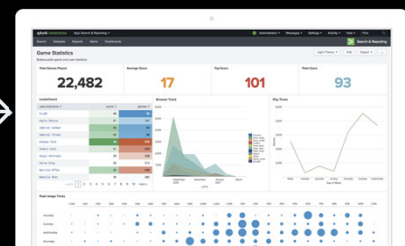
AI and Machine Learning

Leverage artificial intelligence (AI) powered by machine learning for actionable and predictive insights.

Splunk.com

- Think “log file scrapper” + tools
i.e. lots more than just Perf data
- **search** engine to get at the data
 - data **graphed**
 - njmon JSON = data source
 - Has a python-client to inject

- Random Graph →



splunk>

- Function rich so fairly complex
- 1 day to setup on Linux but couldn't work out how to add data
- Splunk Support just said → You can use our Cloud splunk for your njmon testing & it worked in 10 minutes
- Got njmon loading in 30 minutes
- Found data in Search tool
- Not yet graphed any data 😊

I know some ex-IBMer at Splunk

Probably I need training + time

- Setup & operate
- The JSON data analysis
- We have a working Injector via splunk_http_event_collector

splunk>

Rich Search Function with regular expressions and analytic functions

Graph

JSON hierarch

njmon JSON data

Time period

index="n00blab" source="http:njmon"

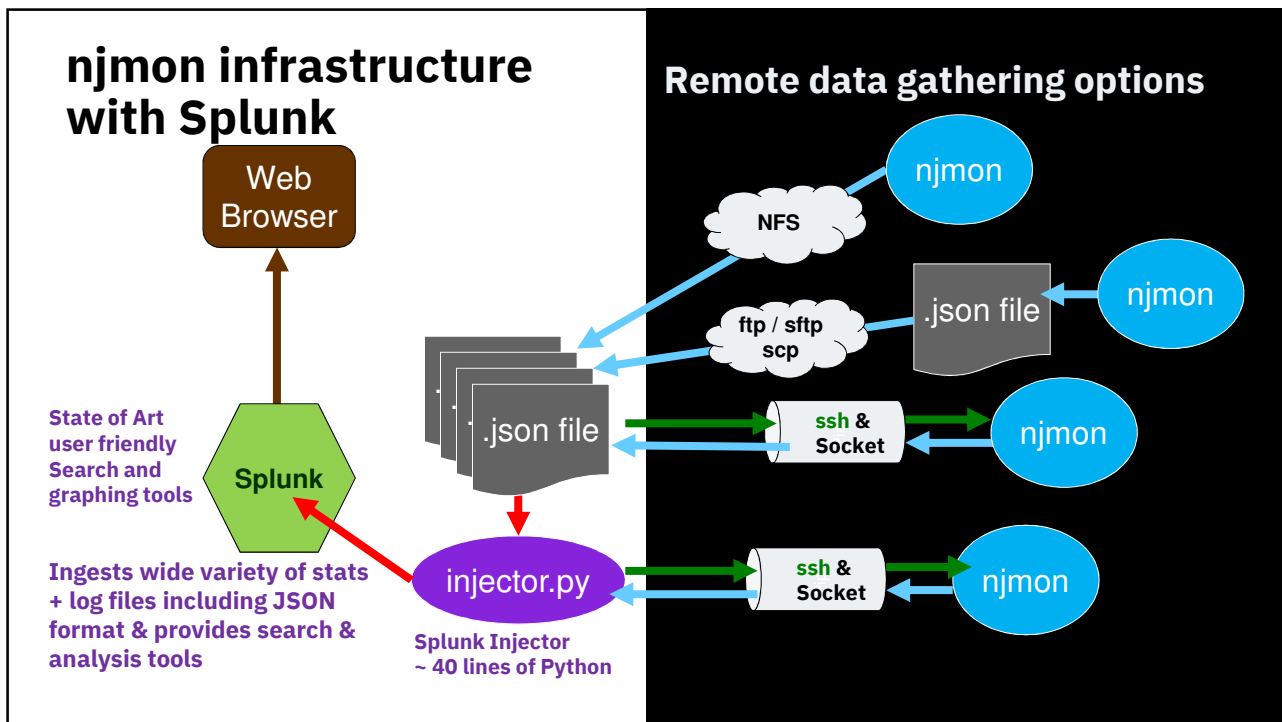
2 events (before 31/08/2018 15:25:32.000) No Event Sampling

Events (2) Patterns Statistics Visualization

Time 31/08/2018 15:25:14.081

Event

```
config: { [-]
OSBuild: Mar  2 2018 13:02:46 1889C_72H
OSName: AIX
OSVersion: 7.2
activecpuspinpool: 12
ame_targetmemexpfactor: 0
ame_targetmemexpsize: 0
ams_hyppgsz: 0
ams_mempoolid: 0
ams_totiomem: 0
cpucap_desired: 0
cpucap_max: 400
cpucap_min: 50
cpucap_online: 0
cpucap_weightage: 128
cpupool_weightage: 128
drives: 1
entitled_proc_capacity: 1
entpoolcap: 1000
expanded_mem_desired: 0
expanded_mem_max: 65536
expanded_mem_min: 16384
expanded_mem_online: 32768
lcpus: 8
machineID: 13601FX
maxpoolcap: 1200
```



Calling all Splunk expert or those willing to learn:

Can you **HELP!**



Three main parts to ELK

- **E**lastic Search - database
- **L**ogstash – data grabber
- **K**ibana – graphs
- Now they are adding more tools!

Open Source or Cloud service
or run your own + support

- 14 day free online trial

<https://www.elastic.co>

A quick look around & it looks nice
Has deployment tools for
Puppet, Chef & Ansible

Need to set up a ELK and develop a
Python Injector

It has a suitable Python Module
here <https://pypi.org/project/python-logstash/>



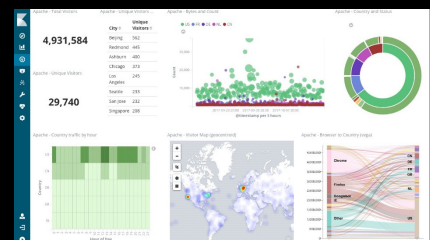
Hmmm!

Starting to look like a trend here !!!

<https://www.elastic.co>

Not tried it yet myself

Grabbed
random
a Graph →



Annoyingly:

All 3 assume Linux on x86_64

I would prefer Linux on POWER

- This can be tried & tested
- Found docker images for ppc64

Volunteers ???



Twice in the last week asked about Kafka from Apache !

Have we an expert in the room?

It has a python client module so we have a good chance ...

Kafka

For streaming **massive** data volumes

PUBLISH & SUBSCRIBE

Read and write streams of data like a messaging system.

PROCESS

Write scalable stream processing applications that react to events in real-time.

STORE

Store streams of data safely in a distributed, replicated, fault-tolerant cluster.

Can you get njmon for AIX njmon for Linux + JSON injectors ?

By the way: There is a nmon file
format to JSON formatter available

Already released to open source
at

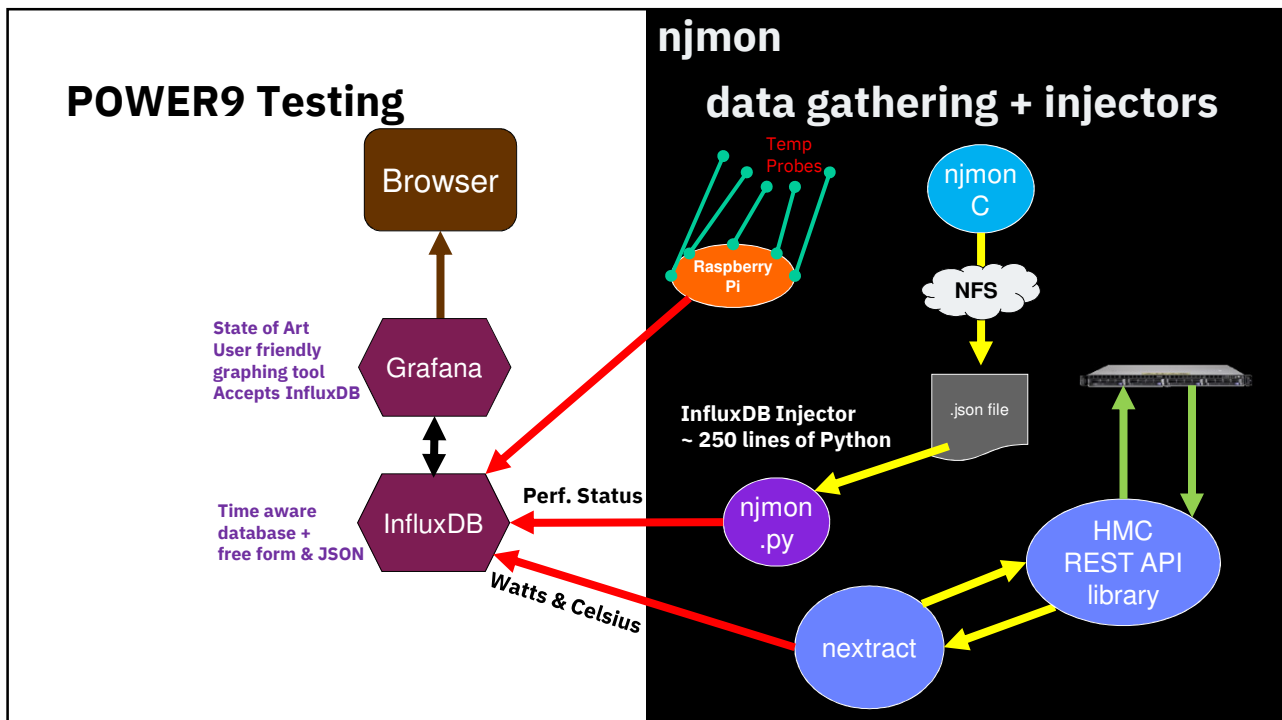
<https://tinyurl.com/njmon>

Binaries compiled for

- AIX 6 (VIOS2+) + AIX 7 (VIOS3)
- Linux Ubuntu 18.04 x86_64 & ppc64
- Simple compile for others

Proactive support from . . .

nag@uk.ibm.com



Times are changing!

Summary:

nmonchart

- ksh, fast, automated, beautiful graphs!

nmon not going away

- On screen or data capture
- Stable down stream infrastructure
- Very popular

njmon for new age online tooling

- Pretty simple C code
- Near real-time, stream, flexible, Python's JSON parser
- nextract for temp & Watts
- AIX vast array of perfstat stats
- Linux nmon & njmon code synergy
- Help needed for down stream tools