Maciej Czekaj

Data Science
→ FreeBSD
→ ARM64

Who am I?

Maciej Czekaj, PhD

- ARMv7/ARMv8 embedded software
  - FreeBSD
  - Linux
  - Marvell Armada, ThunderX, Octeon
- Dataplane networking (Telecom/Security)
  - DPDK
    - First ARMv8 40GB/s Ethernet driver
  - ODP
    - Port for ThunderX & Octeon

Lead S/W engineer @ Semihalf

- TCP/IP stacks
- FPGA
  - Kornik 100G Ethernet Smart NIC
- Clang compiler
  - Xtensa HIFI support
- Comp. Sci. PhD
  - AGH University, Kraków,Poland
  - Hardware acceleration of traffic classifiers for high throughput Ethernet

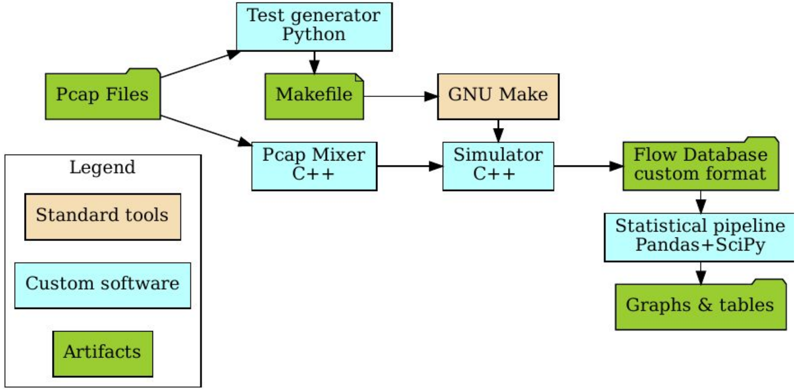
Agenda

- The task
    - Data Science experiment
- The platform
    - ThunderX2
- The stack
    - Data Science software stack on FreeBSD
- The execution
    - Large-scale simulations controlled by simple means
- The aftermath
    - What worked well, what needs to be improved

The task

Massive simulation experiment

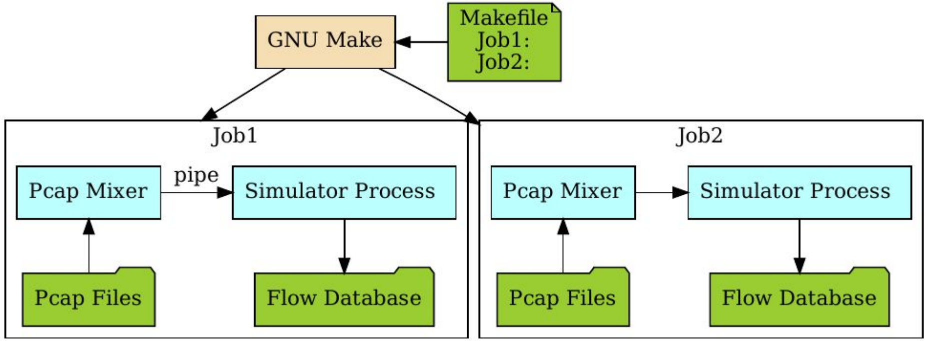
- Network device simulation
- Input: pcap files
- Output
  - Event trace
  - Statistics
- Technology:
  - Custom simulator in C++
  - Statistical s/w in Python
- Originally on Linux Desktop



The scale

Massive simulation experiment

- ~ 200 input files spanning 100 GB
- ~ 200 GB of output data
  - Custom binary format
- ~ 1000 simulation experiments
- 1 experiment takes up to:
  - 1 hour on 1 CPU core
  - 30 GB of RAM
- How to make it scale?
  - Simplify I/O
  - Coarse-grained parallelism



The platform

Workload characteristics

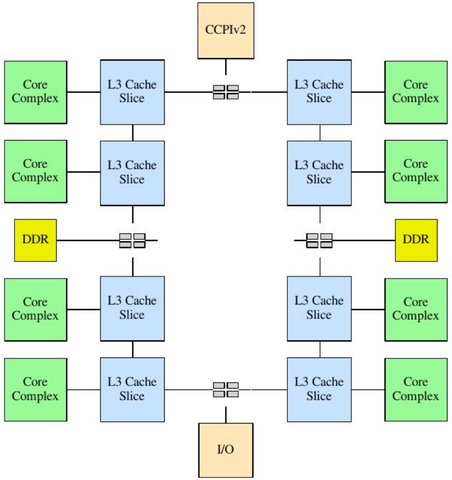
- 1 Core Bottleneck: memory latency
  - Large data structures: hash tables, trees
- System bottleneck: memory bandwidth

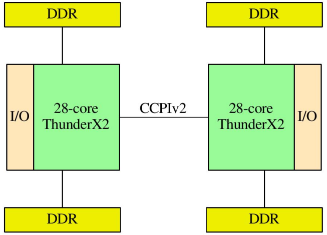
Wish list

- Lots of RAM
- Many CPUs
- Large L3 cache

ThunderX2

- 2 x 28-core ARMv8.2
- 4-way SMT (turned off)
- 2 x 28 MiB L3$
- L3 cross section B/W: 6TB/s
- 8 - channel DRAM controller
- DRAM B/W: 200 GB/s
- CCPIv2 B/W: 600 Gb/s
- FreeBSD 12.2
- ZFS SSD pool





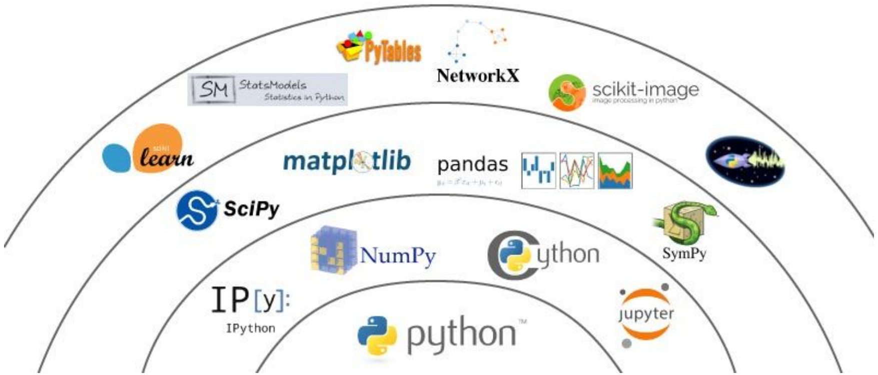
Semihalf

The stack

Python data science toolkit

- Numpy
- Scipy
- Scikit-learn
- Matplotlib
- Pandas
- Jupyter Notebook
- Dependencies…
  - 10s of packages
  - Each requires specific version

➔ https://morioh.com/p/a42cb68ff2b5

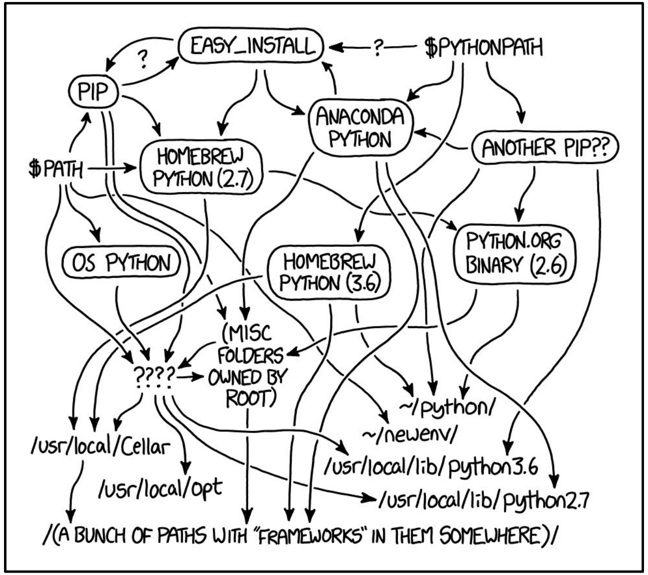


Deployment dilemma

Deployment options

- OS package manager (FreeBSD ports)
  - bad idea (e.g. wrong Python version)
- pip - Python package manager
- pip + virtualenv ("jail" for Python)
- Anaconda - Python distro
  - only Linux & Windows & MacOS :(
- Containers?

➔ https://xkcd.com/1987



MY PYTHON ENVIRONMENT HAS BECOME SO DEGRADED THAT MY LAPTOP HAS BEEN DECLARED A SUPERFUND SITE.

Deployment dilemma

Choice: Python + virtualenv

- Pros
  - It worked! (eventually)
- Cons
  - Source packages
    - Python packages are often C wrappers
    - Two compilers: Clang + GCC
    - Tweaks due to Linux/x86 assumptions
  - Not totally isolated
    - Some packages provided by ports
    - System upgrade breaks dependencies

Conclusions:

- Jail is a must
- Keep your dev env in a container
- Upgrade when you must
- Keep the backup on the old containter
- Don't chase the latest API (unless you must)

Porting C++

Ingredients

- POSIX compliance
  - minor API tweaks
- Clang compliance
  - compiler warnings
- Performance differences
  - C++ iostream is slow
  - … but it is not designed to be fast!

Conclusions:

- Standard compliance pays off
- Portability = code quality
  - Found few bugs in the process
  - Eliminated undefined behavior
- Porting was simpler that installing the Python stack!
- Use low-level I/O
  - `fread/fwrite`
  - Design C++ structures as C structures (POD)

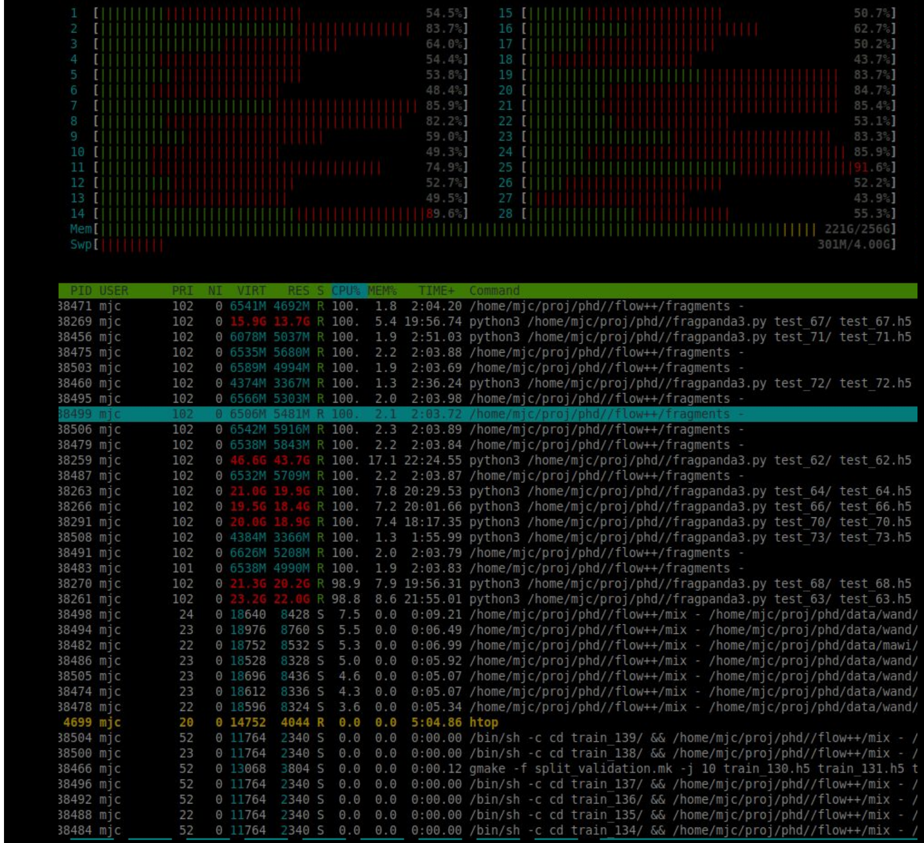
What about ARM64?

ARM64 becomes "invisible"

- Portable code = no issues
- Frictionless recompilation
- Little-endian helps
  - Similar ABI
- FreeBSD/ARM64 is "invisible" too!
  - True Tier 1 platform!
  - Just works

The execution

- Principal guideline: no more software!
- Try standard tools
- 1 experiment takes:
  - GNU Make job
  - 2 processes joined by a pipe
  - Followed by a python script
  - Up to 30GB of RAM!
- Issues with Make job server
  - -j option not flexible enough
  - CPU load is a bad metric
  - No build system monitors RAM pressure



```
  1  [||||||||||||||||||||||||||||                 54.5%]   15 [|||||||||||||||||||||||||||                  50.7%]
  2  [|||||||||||||||||||||||||||||||||||||||||||  83.7%]   16 [||||||||||||||||||||||||||||||||             62.7%]
  3  [|||||||||||||||||||||||||||||||||            64.0%]   17 [||||||||||||||||||||||||||                   50.2%]
  4  [||||||||||||||||||||||||||||                 54.4%]   18 [|||||||||||||||||||||||                      43.7%]
  5  [||||||||||||||||||||||||||||                 53.8%]   19 [|||||||||||||||||||||||||||||||||||||||||||  83.7%]
  6  [|||||||||||||||||||||||||                    48.4%]   20 [|||||||||||||||||||||||||||||||||||||||||||  84.7%]
  7  [|||||||||||||||||||||||||||||||||||||||||||| 85.9%]   21 [|||||||||||||||||||||||||||||||||||||||||||  85.4%]
  8  [||||||||||||||||||||||||||||||||||||||||||   82.2%]   22 [||||||||||||||||||||||||||||                 53.1%]
  9  [||||||||||||||||||||||||||||||               59.0%]   23 [||||||||||||||||||||||||||||||||||||||||||   83.3%]
  10 [|||||||||||||||||||||||||                    49.3%]   24 [|||||||||||||||||||||||||||||||||||||||||||| 85.9%]
  11 [|||||||||||||||||||||||||||||||||||||||      74.9%]   25 [|||||||||||||||||||||||||||||||||||||||||||||91.6%]
  12 [|||||||||||||||||||||||||||                  52.7%]   26 [|||||||||||||||||||||||||||                  52.2%]
  13 [|||||||||||||||||||||||||                    49.5%]   27 [||||||||||||||||||||||                       43.9%]
  14 [|||||||||||||||||||||||||||||||||||||||||||||89.6%]   28 [||||||||||||||||||||||||||||                 55.3%]
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||||221G/256G]
  Swp[|||||||||                                                                                 301M/4.00G]

  PID USER       PRI  NI  VIRT   RES S CPU% MEM%   TIME+  Command
38471 mjc        102   0 6541M 4692M R 100.  1.8  2:04.20 /home/mjc/proj/phd//flow++/fragments -
38269 mjc        102   0 15.9G 13.7G R 100.  5.4 19:56.74 python3 /home/mjc/proj/phd//fragpanda3.py test_67/ test_67.h5
38456 mjc        102   0 6078M 5037M R 100.  1.9  2:51.03 python3 /home/mjc/proj/phd//fragpanda3.py test_71/ test_71.h5
38475 mjc        102   0 6535M 5680M R 100.  2.2  2:03.88 /home/mjc/proj/phd//flow++/fragments -
38503 mjc        102   0 6589M 4994M R 100.  1.9  2:03.69 /home/mjc/proj/phd//flow++/fragments -
38460 mjc        102   0 4374M 3367M R 100.  1.3  2:36.24 python3 /home/mjc/proj/phd//fragpanda3.py test_72/ test_72.h5
38495 mjc        102   0 6566M 5303M R 100.  2.0  2:03.98 /home/mjc/proj/phd//flow++/fragments -
38499 mjc        102   0 6506M 5481M R 100.  2.1  2:03.72 /home/mjc/proj/phd//flow++/fragments -
38506 mjc        102   0 6542M 5916M R 100.  2.3  2:03.89 /home/mjc/proj/phd//flow++/fragments -
38479 mjc        102   0 6538M 5843M R 100.  2.2  2:03.84 /home/mjc/proj/phd//flow++/fragments -
38259 mjc        102   0 46.6G 43.7G R 100. 17.1 22:24.55 python3 /home/mjc/proj/phd//fragpanda3.py test_62/ test_62.h5
38487 mjc        102   0 6532M 5709M R 100.  2.2  2:03.87 /home/mjc/proj/phd//flow++/fragments -
38263 mjc        102   0 21.0G 19.9G R 100.  7.8 20:29.53 python3 /home/mjc/proj/phd//fragpanda3.py test_64/ test_64.h5
38266 mjc        102   0 19.5G 18.4G R 100.  7.2 20:01.66 python3 /home/mjc/proj/phd//fragpanda3.py test_66/ test_66.h5
38291 mjc        102   0 20.0G 18.9G R 100.  7.4 18:17.35 python3 /home/mjc/proj/phd//fragpanda3.py test_70/ test_70.h5
38508 mjc        102   0 4384M 3366M R 100.  1.3  1:55.99 python3 /home/mjc/proj/phd//fragpanda3.py test_73/ test_73.h5
38491 mjc        102   0 6626M 5208M R 100.  2.0  2:03.79 /home/mjc/proj/phd//flow++/fragments -
38483 mjc        101   0 6538M 4990M R 100.  1.9  2:03.83 /home/mjc/proj/phd//flow++/fragments -
38270 mjc        102   0 21.3G 20.2G R 98.9  7.9 19:56.31 python3 /home/mjc/proj/phd//fragpanda3.py test_68/ test_68.h5
38261 mjc        102   0 23.2G 22.0G R 98.8  8.6 21:55.01 python3 /home/mjc/proj/phd//fragpanda3.py test_63/ test_63.h5
38498 mjc         24   0 18640  8428 S  7.5  0.0  0:09.21 /home/mjc/proj/phd//flow++/mix - /home/mjc/proj/phd/data/wand/
38494 mjc         23   0 18976  8760 S  5.5  0.0  0:06.49 /home/mjc/proj/phd//flow++/mix - /home/mjc/proj/phd/data/wand/
38482 mjc         22   0 18752  8532 S  5.3  0.0  0:06.99 /home/mjc/proj/phd//flow++/mix - /home/mjc/proj/phd/data/mawi/
38486 mjc         23   0 18528  8328 S  5.0  0.0  0:05.92 /home/mjc/proj/phd//flow++/mix - /home/mjc/proj/phd/data/wand/
38505 mjc         23   0 18696  8436 S  4.6  0.0  0:05.07 /home/mjc/proj/phd//flow++/mix - /home/mjc/proj/phd/data/wand/
38474 mjc         23   0 18612  8336 S  4.3  0.0  0:05.07 /home/mjc/proj/phd//flow++/mix - /home/mjc/proj/phd/data/wand/
38478 mjc         22   0 18596  8324 S  3.6  0.0  0:05.34 /home/mjc/proj/phd//flow++/mix - /home/mjc/proj/phd/data/wand/
 4699 mjc         20   0 14752  4044 R  0.0  0.0  5:04.86 htop
38504 mjc         52   0 11764  2340 S  0.0  0.0  0:00.00 /bin/sh -c cd train_139/ && /home/mjc/proj/phd//flow++/mix - /
38500 mjc         23   0 11764  2340 S  0.0  0.0  0:00.00 /bin/sh -c cd train_138/ && /home/mjc/proj/phd//flow++/mix - /
38466 mjc         52   0 13068  3804 S  0.0  0.0  0:00.12 gmake -f split_validation.mk -j 10 train_130.h5 train_131.h5 t
38496 mjc         52   0 11764  2340 S  0.0  0.0  0:00.00 /bin/sh -c cd train_137/ && /home/mjc/proj/phd//flow++/mix - /
38492 mjc         52   0 11764  2340 S  0.0  0.0  0:00.00 /bin/sh -c cd train_136/ && /home/mjc/proj/phd//flow++/mix - /
38488 mjc         22   0 11764  2340 S  0.0  0.0  0:00.00 /bin/sh -c cd train_135/ && /home/mjc/proj/phd//flow++/mix - /
38484 mjc         52   0 11764  2340 S  0.0  0.0  0:00.00 /bin/sh -c cd train_134/ && /home/mjc/proj/phd//flow++/mix - /
```

FreeBSD to the rescue

- Embrace the failure
  - Let the OOM killer handle it
  - Make job server would cancel the job
  - Partial artifacts would be deleted
  - ZFS maintains integrity
- OOM killer's strategy:
  - Select the process with the most pages
    - swap & active
    - private & vnodes
  - Controversial but successful!
- see `vm_pageout_oom()` in `vm_pageout.c`

`From vm_pageout.c:`

```
/ * After one round of OOM terror,
recall our vote.
```

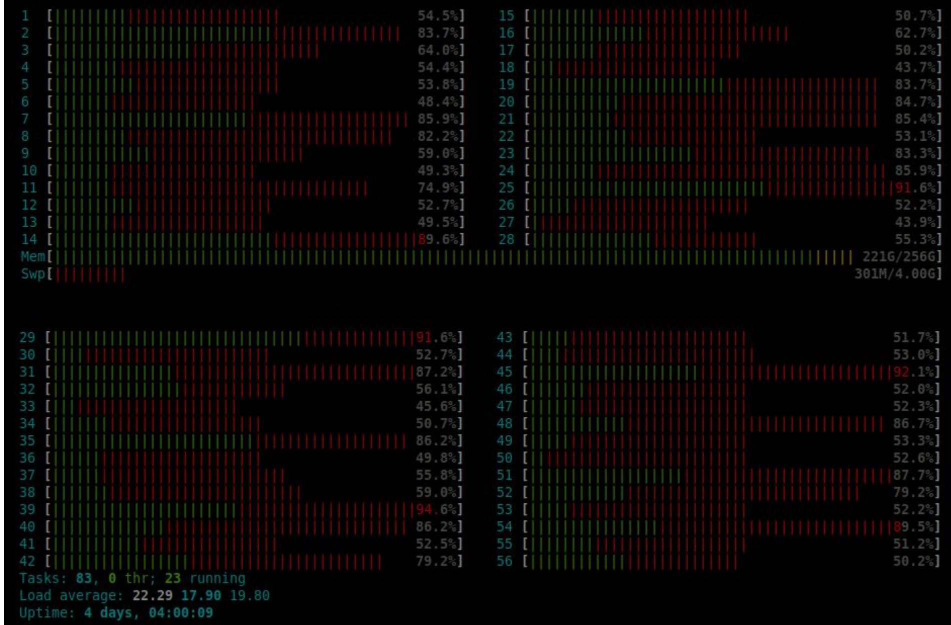
- Brutal, but effective
  - System is responsive
  - No page thrashing
  - SSH sessions can be opened
- Blocked by
  - `protect`
  - `procctl()`

The final run

A week-long experiment:

- No surprises ! (good)
- Let the Make job server do the job
- Occasional supervision & re-spin of the failed tasks
- Scientific article:

  Czekaj, M.; Jamro, E.; Wiatr, K. Estimating the Memory Consumption of a Hardware IP Defragmentation Block. *Electronics* **2021**, *10*, 2015. https://doi.org/10.3390/electronics10162015



```
1  [   54.5%]   15 [   50.7%]
2  [   83.7%]   16 [   62.7%]
3  [   64.0%]   17 [   50.2%]
4  [   54.4%]   18 [   43.7%]
5  [   53.8%]   19 [   83.7%]
6  [   48.4%]   20 [   84.7%]
7  [   85.9%]   21 [   85.4%]
8  [   82.2%]   22 [   53.1%]
9  [   59.0%]   23 [   83.3%]
10 [   49.3%]   24 [   85.9%]
11 [   74.9%]   25 [   91.6%]
12 [   52.7%]   26 [   52.2%]
13 [   49.5%]   27 [   43.9%]
14 [   89.6%]   28 [   55.3%]
Mem[                    221G/256G]
Swp[                   301M/4.00G]

29 [   91.6%]   43 [   51.7%]
30 [   52.7%]   44 [   53.0%]
31 [   87.2%]   45 [   92.1%]
32 [   56.1%]   46 [   52.0%]
33 [   45.6%]   47 [   52.3%]
34 [   50.7%]   48 [   86.7%]
35 [   86.2%]   49 [   53.3%]
36 [   49.8%]   50 [   52.6%]
37 [   55.8%]   51 [   87.7%]
38 [   59.0%]   52 [   79.2%]
39 [   94.6%]   53 [   52.2%]
40 [   86.2%]   54 [   89.5%]
41 [   52.5%]   55 [   51.2%]
42 [   79.2%]   56 [   50.2%]
Tasks: 83, 0 thr; 23 running
Load average: 22.29 17.90 19.80
Uptime: 4 days, 04:00:09
```

The aftermath

The Good:

- A FreeBSD success story
- ARM64 is a premier Tier 1 platform
- A decade long effort!
- Reliable platform for scientific research
- Simple tools do the job

The Bad:

- Complex s/w ecosystem needs containers

The ugly:

- “If you work for a living, why do you kill yourself working?” **– Tuco, The Ugly.**
- Lack of binary packages for python
  - Anaconda not interested for now
- Need an image shop for Data Science

Q&A