Suricata
Extreme Performance Tuning

With Incredible Courage

SURICON

By

- Michal Purzynski (@MichalPurzynski)
  - Threat Management, Mozilla
- Peter Manev (@pevma)
  - Suricata Core Team
  - Lead QA and training instructor
  - Stamus Networks
  - Mobster evangelist

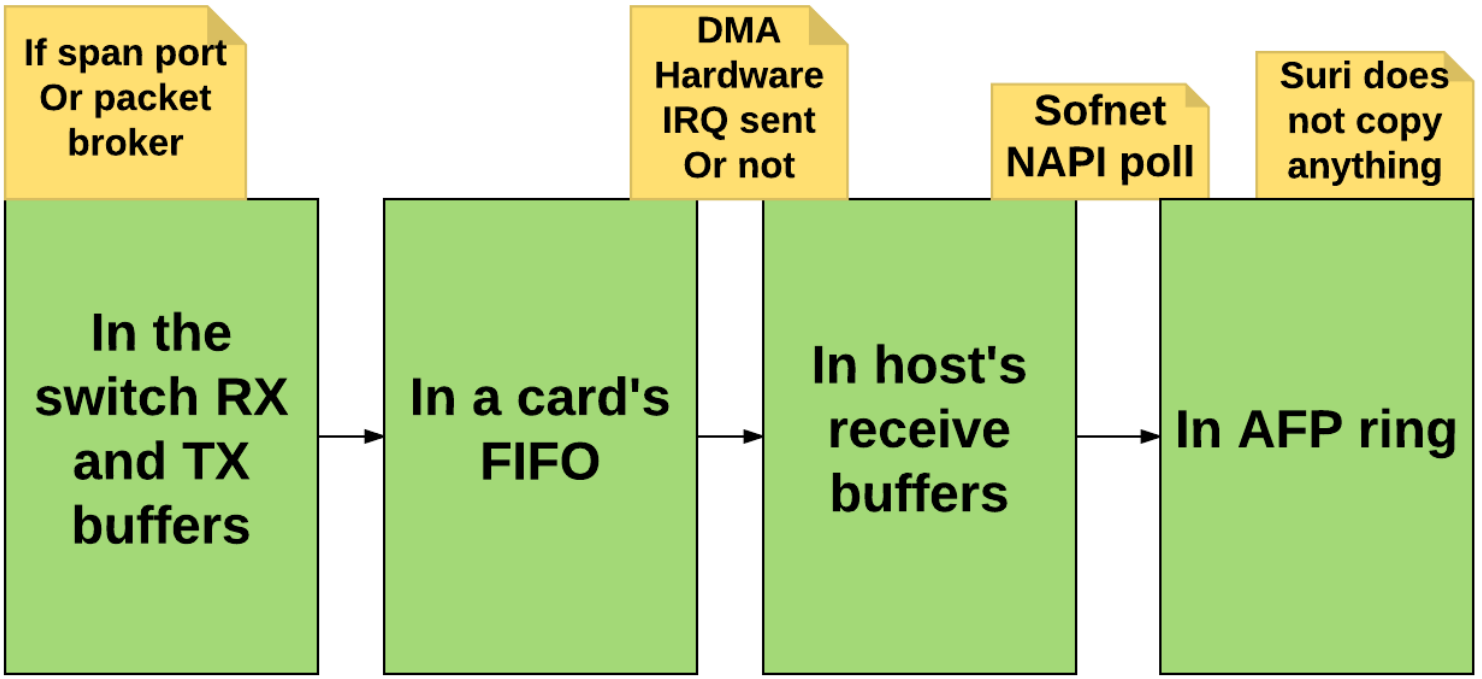
For...

The brother and sister mobsters

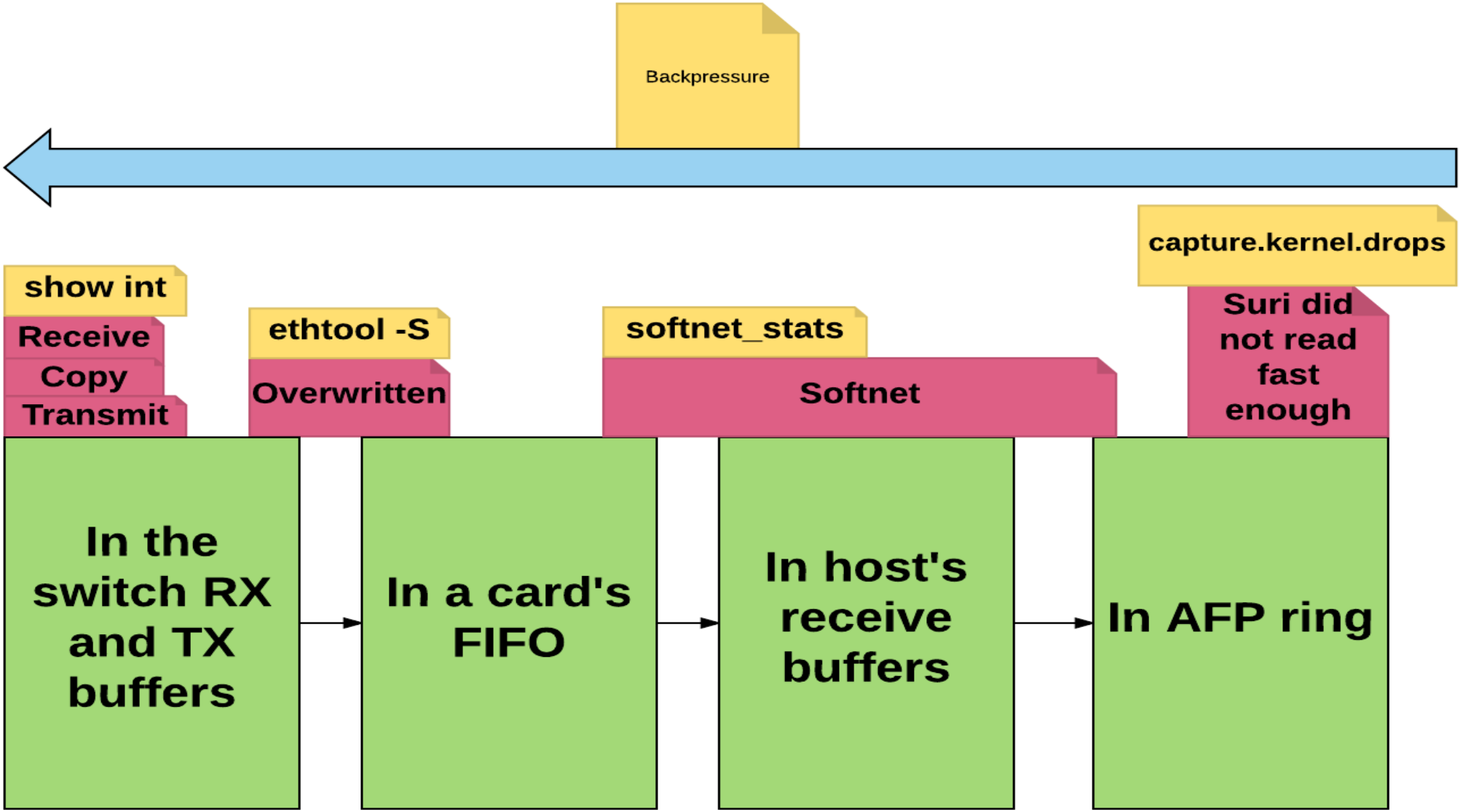
SURICON

Life of a packet...

...is full of dark alleys
where it can get lost...



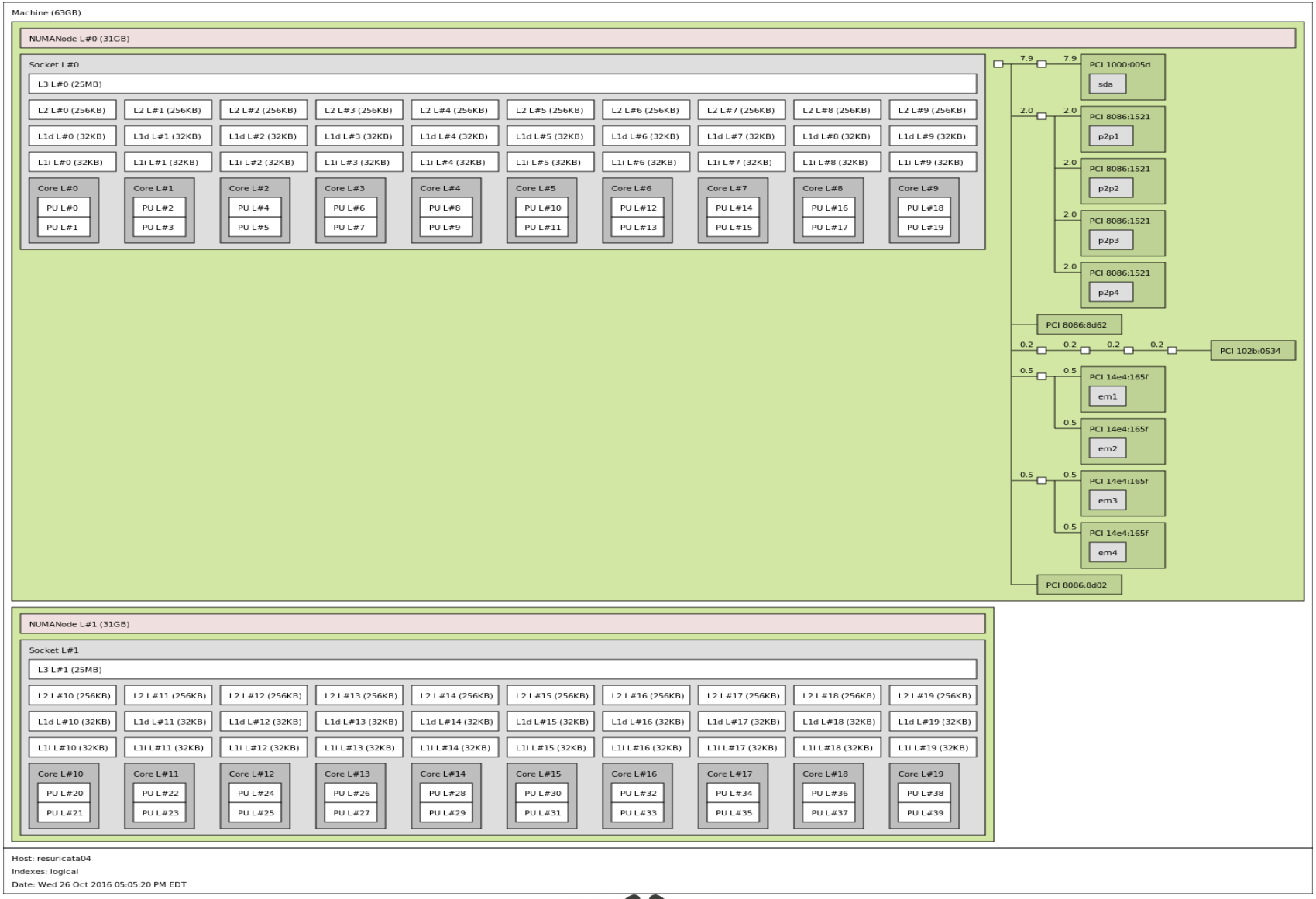
All your packets belong to Suricata..... if they make it



The Grand Plan

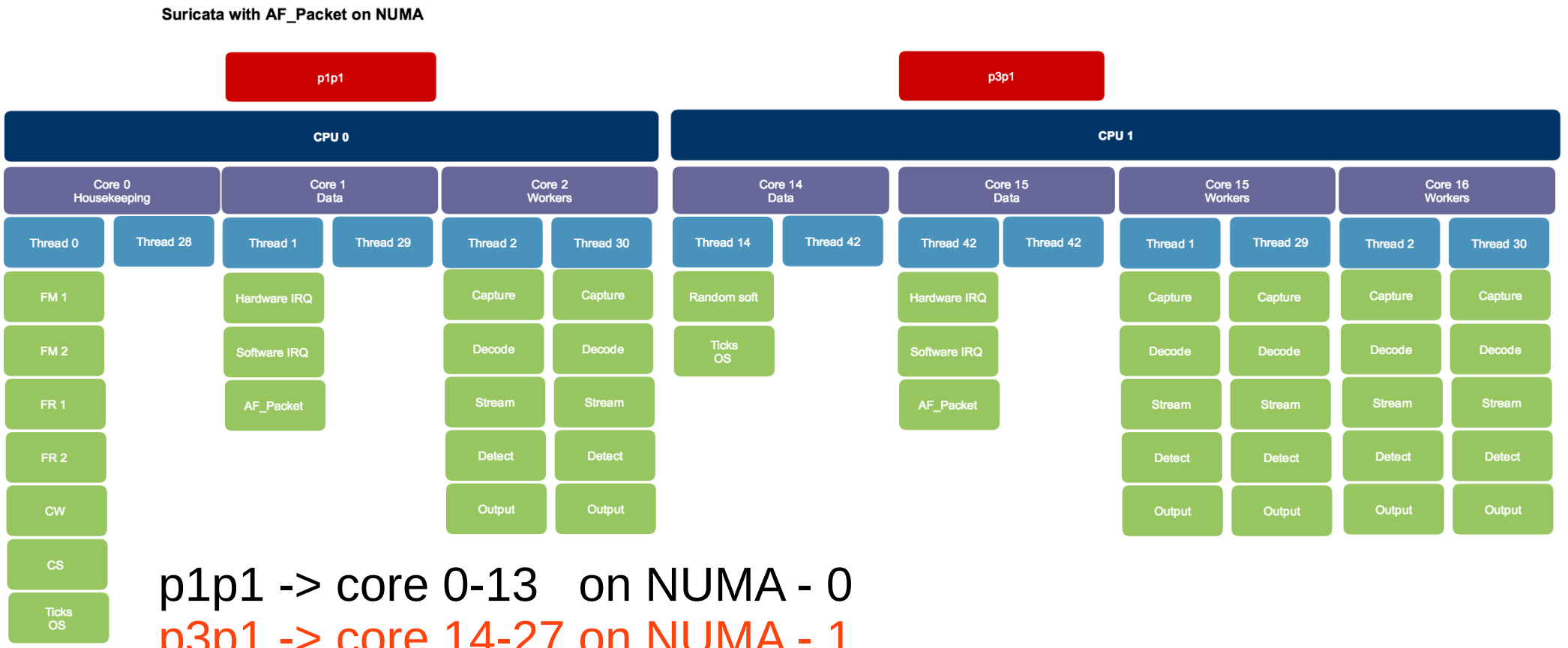
Use a local CPU cache L3 as a data bus :-)

CPU socket
NIC location and NUMA relation



SURICON

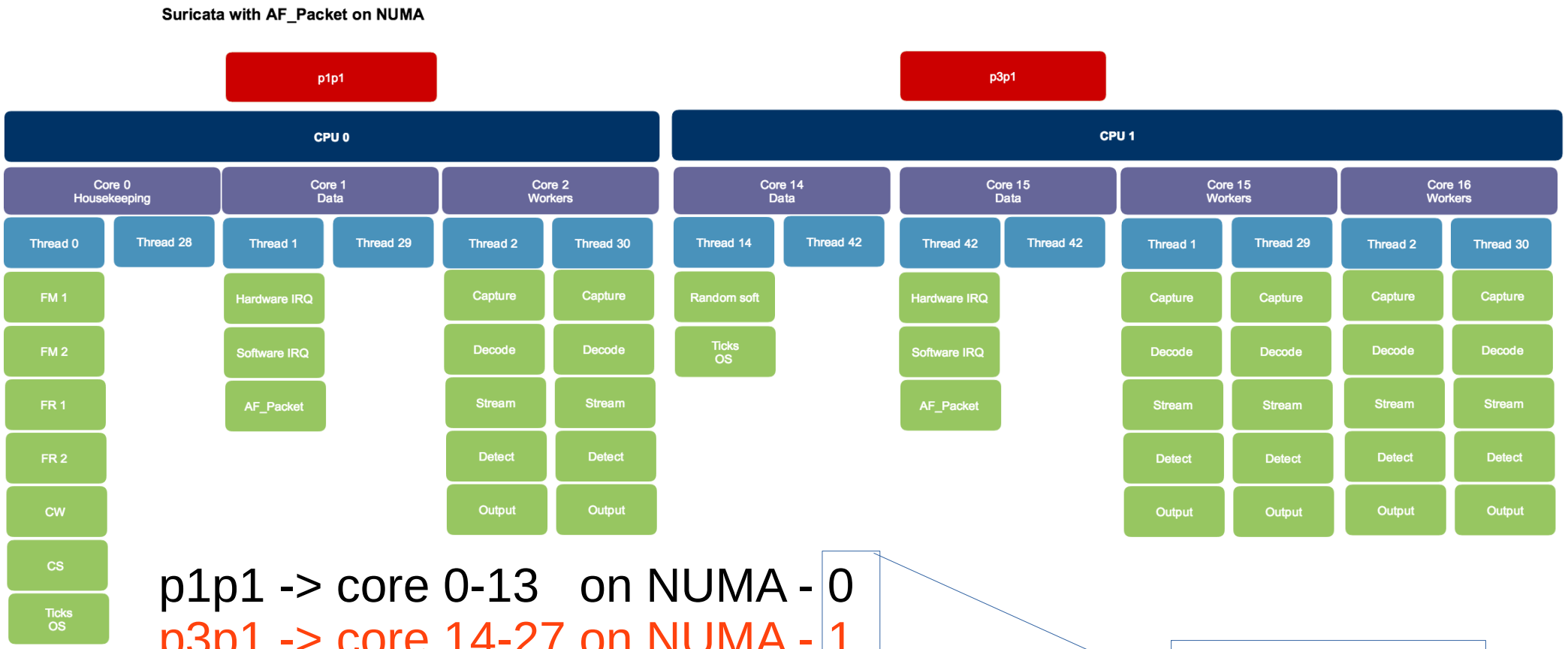
CPU socket
NIC location and NUMA relation (HT)



p1p1 -> core 0-13 on NUMA - 0
p3p1 -> core 14-27 on NUMA - 1
p1p1 -> core 28-41 on NUMA - 0
p3p1 -> core 42-55 on NUMA - 1

SURICON

CPU socket
NIC location and NUMA relation (HT)



p1p1 -> core 0-13 on NUMA - 0
p3p1 -> core 14-27 on NUMA - 1
p1p1 -> core 28-41 on NUMA - 0
p3p1 -> core 42-55 on NUMA - 1

Kernel – 3.x / 4.x with HT

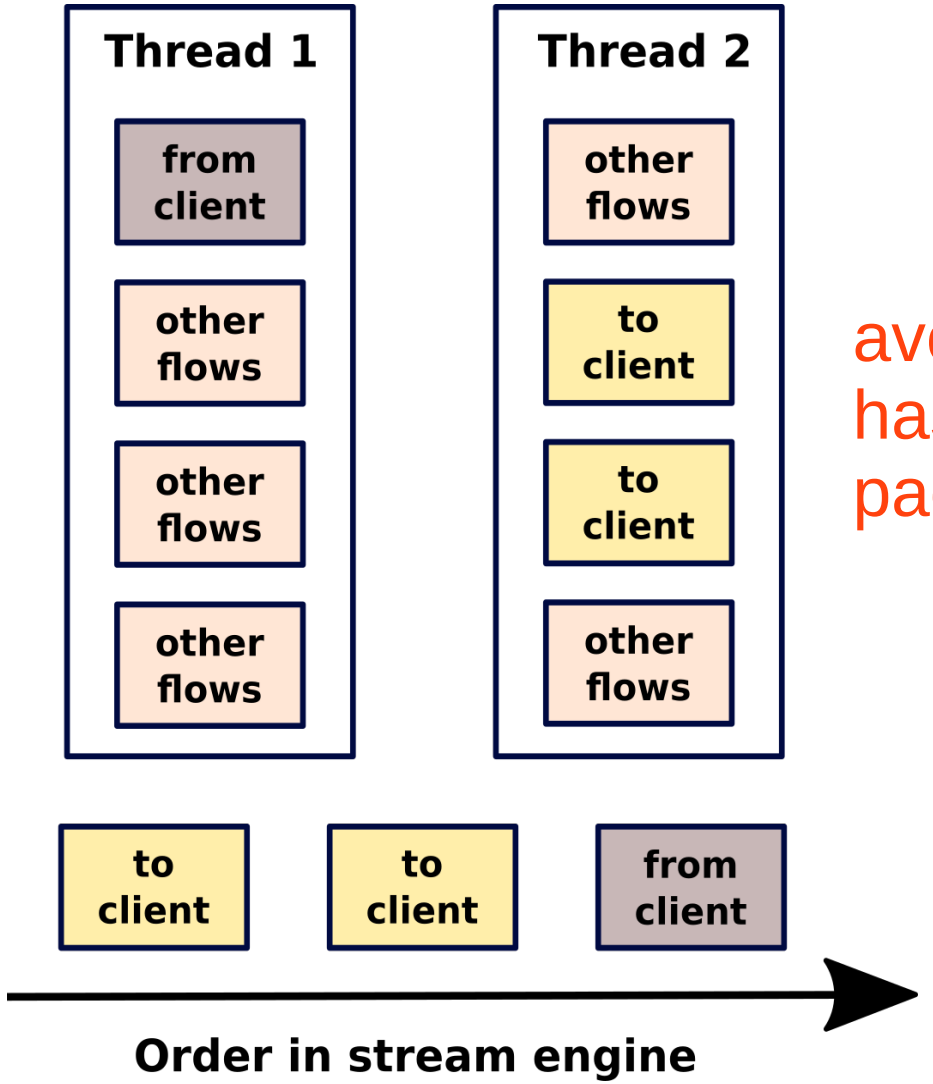
SURICON

Server

- One card per NUMA slot.
  - Or latency will kill you (™)
- Haswell, for best results
  - Independent P-states and other goodness
- C-states enabled. Turbo mode rocks IDS.
  - Linux will overwrite anyway
- L2 prefetchers disabled
  - Your card will send packets to L3 for you (DCA)
  - L2 prefetching impacts L3 and trashes it
- Snoop early, snoop frequently and never at home

NIC

Keep # of RSS queues == 1



avoid asymmetric hashing impact -> packet reordering

NIC indicators

- rx_missed_errors
  - packets overwritten in card’s FIFO
- rx_no_dma_resources / rx_no_buffer_count
  - means we are dropping packets because we don't have any free descriptors and cannot alloc new.
  - when using RSS=1 - “rx_no_dma_resources” does not increase counter

Use ethtool -S <interface> to discover magic

Use ethtool -g <interface> make ring smaller (yes)

The Cache consideration

- CPU load cache misses and latency
  - Execution WAITS till the data is fetched from RAM or other cache
  - MESIF (cache and memory coherence protocol) plays for Suri
- Happens when
  - Other software trashes caches (Redis/ES/DB...)
  - Threads are “jumping” between CPU sockets
  - NIC's ring descriptors are too big
  - HW prefetchers are enabled “wild wild west style”

The NUMA consideration

- MESIF plays for Intel
- Remote data access has a huge delay
- Remote node cache L3 access latency +- cache L3 miss latency
- Latency, not a cache misses are a problem for NUMA

Local L3 -+ 20ns - Remote L3 - >80ns
Local RAM - 96ns, remote 140ns

Cache thrashing effect in (bad)action

```
Performance counter stats for 'CPU(s) 0-39':

   25,845,771,051      LLC-loads                                                   [50.00%]
    7,214,035,111      LLC-load-misses           #   27.91% of all LL-cache hits   [50.00%]
    3,768,432,384      LLC-stores                                                  [50.00%]
   12,306,901,314      LLC-prefetches                                              [50.00%]

     60.001104971 seconds time elapsed
```

"A cache miss is a failed attempt to read or write a piece of data in the cache, which results in a main memory access with much longer latency."

Cache thrashing effect in (bad)action

CPU can not access RAM directly

```
Performance counter stats for 'CPU(s) 0-39':

   25,845,771,051      LLC-loads                                                   [50.00%]
    7,214,035,111      LLC-load-misses           #   27.91% of all LL-cache hits   [50.00%]
    3,768,432,384      LLC-stores                                                  [50.00%]
   12,306,901,314      LLC-prefetches                                              [50.00%]

     60.001104971 seconds time elapsed
```

If system is under pressure and packets are not in CPU L3 local cache = load_misses

SURICON

Which usually results in... (watch out for)

- capture.kernel_drops
- tcp.reassembly_gap
- Flow timeouts reached
- A small number of CPUs being pegged to 100%
- Drops in place no one looks for except us :-)
  - ….and you now :)

So we looked at our set up (20Gbps)

- Suricata 3.2dev (using AF-PACKETv2/3)
- Kernel 4.4.0-38-generic #57~14.04.1-Ubuntu
- 21332 rules from ET Pro
- 128GB RAM, 8 DIMMS, 4 per socket.
- 2x Intel(R) Xeon(R) CPU E5-2697 v3 @ 2.60GHz - 28 cores total, HT enabled and used for 56 hardware threads.
- 2x dual port X520 AKA 82599 And X710 - one port on each card used. Cards installed into separate NUMA nodes.

Dug in research...

- Kernel code, ixgbe, i40e (and long nights)
- Suricata CPU affinity
- Intel e1000-devel mailing lists
- af-packet v2 and af-packet v3 for Suricata
- X520 and x710 NIC testing
- Suricata's new bypass feature
- 1 RSS queues considerations

After some time digging in...

We introduced the CPU affinity move....

- Step one - isolate cores and pin IRQs
- Step two - configure Suricata cpu affinity
- Step three - local bypass

Step one - isolate cores

- Cores are for you, not for a scheduler
- Steal them, leave one per node for housekeeping
- Single work type per dedicated core = no ticking
- Less userspace->kernel transitions, less TLB and cache trashing

```
nsm16 ➜  ~  cat /proc/cmdline
BOOT_IMAGE=/vmlinuz-4.4.0-45-generic.efi.signed root=UUID=dedcba7d-1909-4797-bd57-663a423a6a2f ro
processor.max_cstate=3 intel_idle.max_cstate=3 selinux=0 apparmor=0 mce=ignore_ce nohz_full=1-13,1
5-55 isolcpus=1-13,15-55 rcu_nocbs=1-13,15-55
```

Step one - set all IRQs away from Suricata thread workers

- Have an IRQ-->core per node
- If not enough, use RPS but never split processing

```
Terminal — ssh nsm16.private.scl3.mozilla.com — zsh
Last login: Fri Oct 28 12:56:49 on ttys009
Identity added: /Users/mpurzynski/.ssh/id_rsa_moco (/Users/mpurzynski/.ssh/id_rsa_moco)
Michals-Mac-mini ➜ ~  ssh nsm16.private.scl3.mozilla.com
Last login: Fri Oct 28 10:56:54 2016 from 10-22-248-146.vpn.scl3.mozilla.com
nsm16 ➜ ~/i40e-1.5.23/scripts  sudo -s
nsm16 ➜ ~/i40e-1.5.23/scripts  ./set_irq_affinity 1 p1p1
IFACE CORE MASK -> FILE
=======================
p1p1 1 2 -> /proc/irq/132/smp_affinity
nsm16 ➜ ~/i40e-1.5.23/scripts  ./set_irq_affinity 15 p3p1
IFACE CORE MASK -> FILE
=======================
p3p1 15 8000 -> /proc/irq/284/smp_affinity
nsm16 ➜ ~/i40e-1.5.23/scripts  cat /proc/interrupts | egrep 'p1p1|p3p1'
 132:     122248  348871981          0          0          0          0          0          0          0          0
      0          0          0          0          0          0          0          0          0          0          0
        0          0          0          0          0          0          0          0          0          0          0
          0          0          0          0          0          0          0          0          0          0
 0          0          0          0          0          0          0          0          0          0          0
   0          0          0   PCI-MSI 4194305-edge      i40e-p1p1-TxRx-0
 284:      16143          0          0          0          0          0          0          0          0          0
      0          0          0          0          1  348770707          0          0          0          0          0
        0          0          0          0          0          0          0          0          0          0          0
          0          0          0          0          0          0          0          0          0          0
 0          0          0          0          0          0          0          0          0          0          0
   0          0          0   PCI-MSI 68157441-edge      i40e-p3p1-TxRx-0
nsm16 ➜ ~/i40e-1.5.23/scripts
```

Step two – Suricata cpu affinity

```
set-cpu-affinity: yes
# Tune cpu affinity of suricata threads. Each family of threads can be bound
# on specific CPUs.
cpu-affinity:
  - management-cpu-set:
      cpu: [ 0,28,14,42 ]  # include only these cpus in affinity settings
      mode: "balanced"
      prio:
        default: "low"
  - detect-cpu-set:
      # NUMA and Hyper-threading example on kenel 4.x
      # NUMA order -> 0/1/0/1
      # (2x14 cpus -56 total with HT)
      # 2 x NICs. 1 - p1p1 and 1 - p3p1
      # p1p1 -> 3-13 on NUMA-0 / p3p1 ->17-27 on NUMA-1 /
      # p1p1 -> 31-41 on NUMA-0 / p3p1 -> 45-55 on NUMA-1
      cpu: ["3-13","17-27","31-41","45-55"]
      mode: "exclusive" # run detect threads in these cpus
      # Use explicitely 3 threads and don't compute number by using
      # detect-thread-ratio variable:
      # threads: 3
      prio:
        default: "high"
```

SURICON

Step two – Suricata cpu affinity

```
- interface: p1p1
  threads: 11
  cluster-id: 99
  use-mmap: yes
  tpacket-v3: yes
  ring-size: 400000
  block-size: 393216
  #buffer-size: 1048576
  ##buffer-size: 262144
  cluster-type: cluster_flow
- interface: p3p1
  threads: 11
  cluster-id: 98
  use-mmap: yes
  tpacket-v3: yes
  ring-size: 400000
  block-size: 393216
  #buffer-size: 1048576
  ##buffer-size: 262144
```

p1p1 -> core 0-13 on NUMA - 0

p3p1 -> core 14-27 on NUMA - 1

p1p1 -> core 28-41 on NUMA - 0

p3p1 -> core 42-55 on NUMA - 1

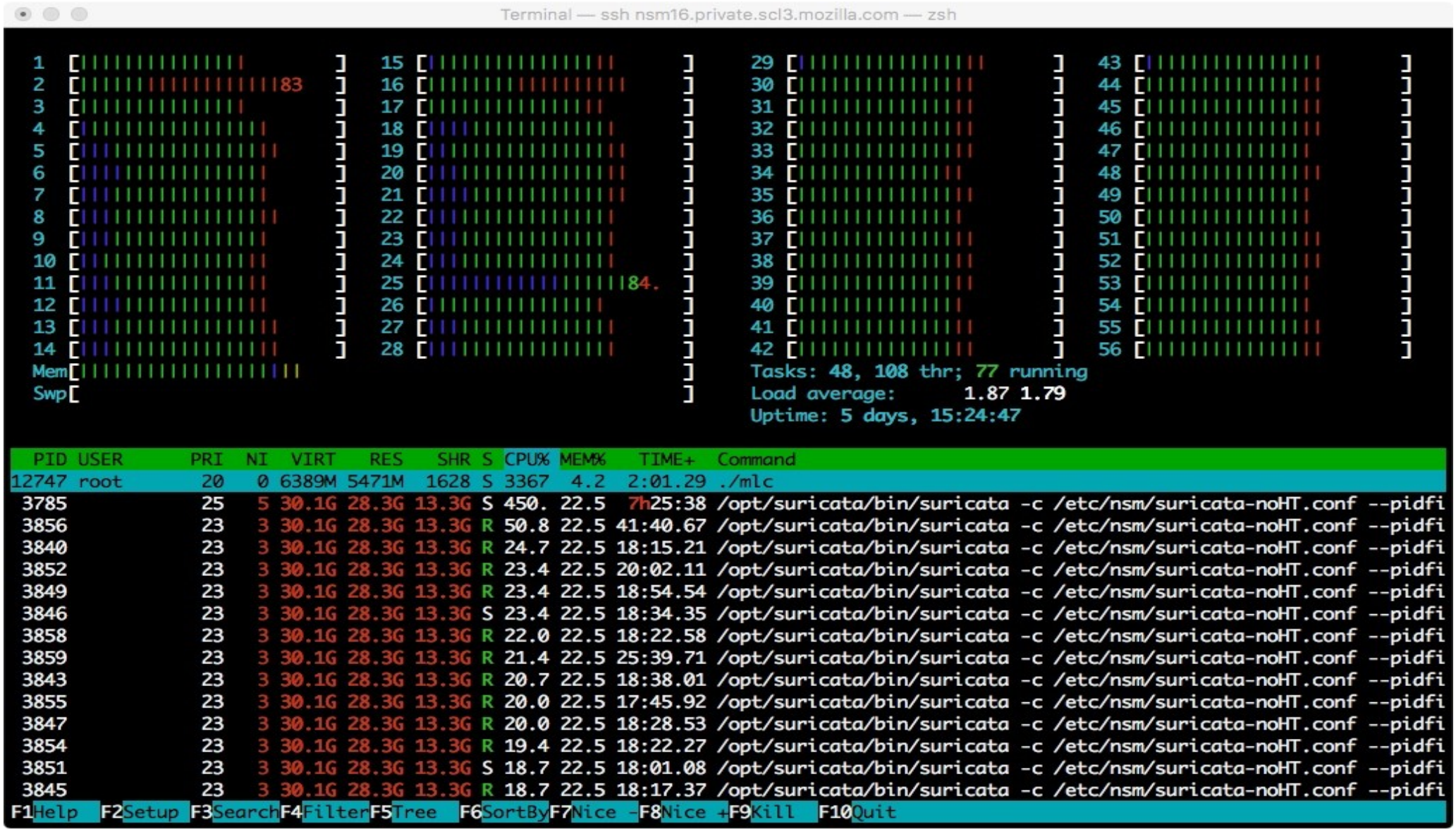
```
- interface: p1p1
  threads: 11
  cluster-id: 99
  use-mmap: yes
  tpacket-v3: yes
  ring-size: 400000
  block-size: 393216
  #buffer-size: 1048576
  ##buffer-size: 262144
  cluster-type: cluster_flow
- interface: p3p1
  threads: 11
  cluster-id: 98
  use-mmap: yes
  tpacket-v3: yes
  ring-size: 400000
  block-size: 393216
  #buffer-size: 1048576
  ##buffer-size: 262144
```

SURICON

Step three – local bypass

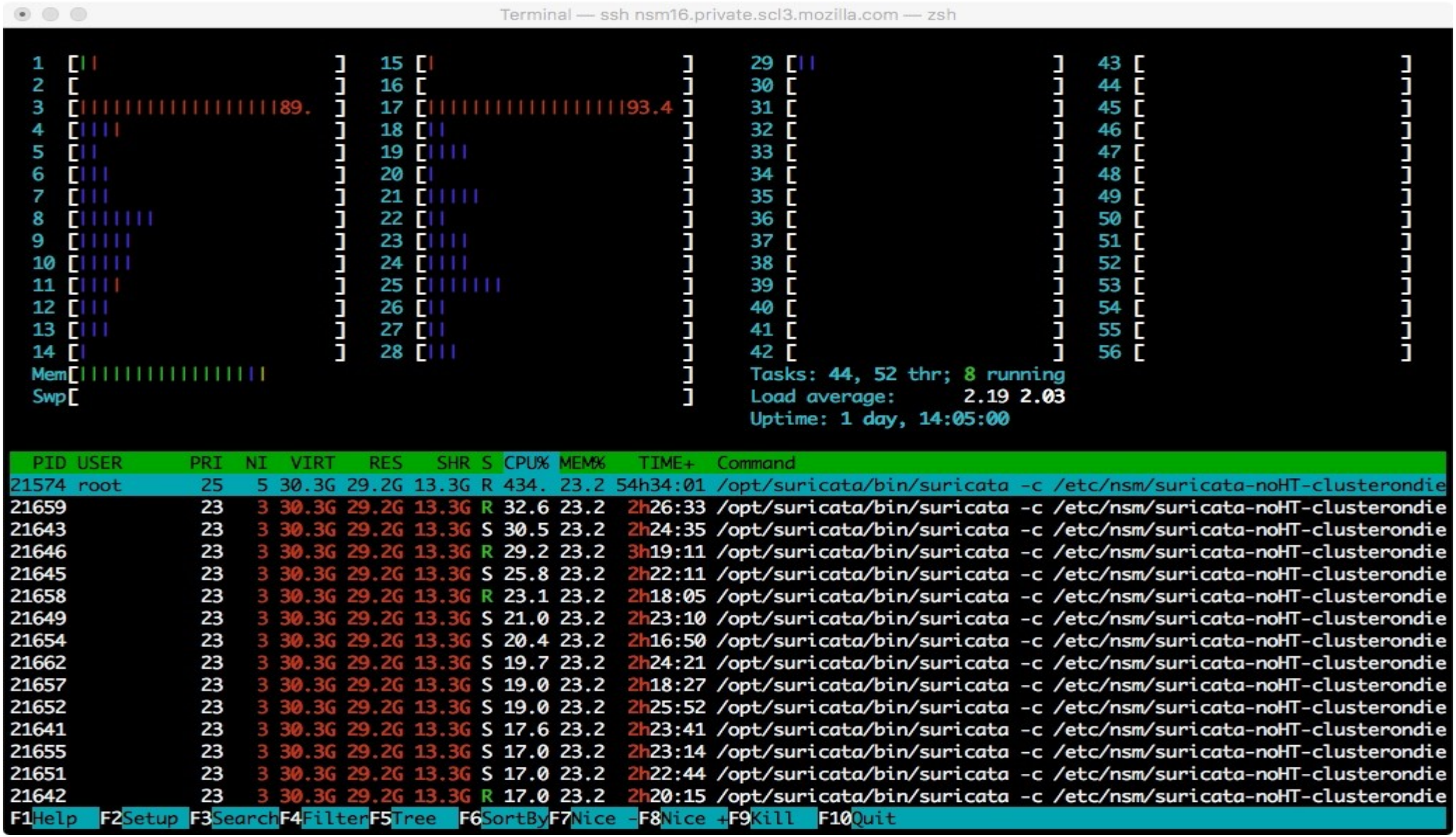
- Local bypass
  - If the corresponding flow is local bypassed then it simply skips all streaming, detection and output and the packet goes directly out in IDS mode and to verdict in IPS mode.
  - In suricata.yaml
    - Set by “stream.bypass: yes”
    - Adjusted by “stream.reassembly.depth”

Performance Before



SURICON

Performance after



SURICON

Cache misses Before and After

```
perf stat -e LLC-loads,LLC-load-misses,LLC-stores,LLC-prefetches -C 1

 Performance counter stats for 'CPU(s) 0':

      1939117135   LLC-loads                                     (66.67%)
       289305806   LLC-load-misses     #   14.92% of all LL-cache hits     (66.67%)
       356341541   LLC-stores                                    (66.67%)
 <not supported>   LLC-prefetches

      114.342786481 seconds time elapsed
```

Core handling kernel. Ring descriptor size 512 buffers. Each buffer is 2048 bytes in size.

```
perf stat -e LLC-loads,LLC-load-misses,LLC-stores,LLC-prefetches -C 1 sleep 60

 Performance counter stats for 'CPU(s) 1':

       659135009   LLC-loads                                     (66.67%)
         1372230   LLC-load-misses     #    0.21% of all LL-cache hits     (66.67%)
       124004599   LLC-stores                                    (66.67%)
 <not supported>   LLC-prefetches

      60.001419120 seconds time elapsed
```

SURICON

Suricata packet drops after

0.00137%

capture.kernel_packets	Total	18887981562
capture.kernel_drops	Total	260649
decoder.pkts	Total	18888406881

On 20Gbps

Mobster myths

“AFP is slow, you must use <a new kernel bypass>” - is it? :-)

“Linux cannot deal with interrupts (use BSD)” - we have NAPI from > 16 years

“Use RSS/RPS to load balance” - and get packets reordering and missing events

“Disable HT it hurts performance” - actually, the opposite

“Make <buffer> huge” - and it won’t fit into L3

Lessons Learned

- Make no assumptions - verify everything
- Understand the packet travel critical path
- Understand what your counters _really_ mean
- NUMA is awesome - know how to use it
- We run out of traffic....

  .
- Everything can be undone with a bunch of badly written rules

Things to watch out for

- rx_missed_errors (NIC/ethtool)
- rx_dma_failed /rx_no_buffer_count (NIC/ethtool)
- Switchport
- Cache misses
- Correct traffic mirroring (esp vlan tags)

- Suricata

- Memcaps
- Reassembly gaps
- Flow emergency
- Decoder invalid
- Make sure MTU is same across

Detailed info to come up

- Research paper
  - Containing all the details of this research
  - Commands and scripts
  - Trouble shooting advice and guidance
  - Perf Indicators
- Potential inclusion in Suricata's docs/repo (PR)
- Some other interesting experiments....

Thanks to

- Mozilla (time, traffic, hardware)
- Intel - Alexander Duyck
- Eric Leblond (@Regit – Suri cpu affinity)
- Daniel Borkmann (netsniff-ng, AFP)
- And Dave Miller for AFP :-)
- SuriCon 2016 !!