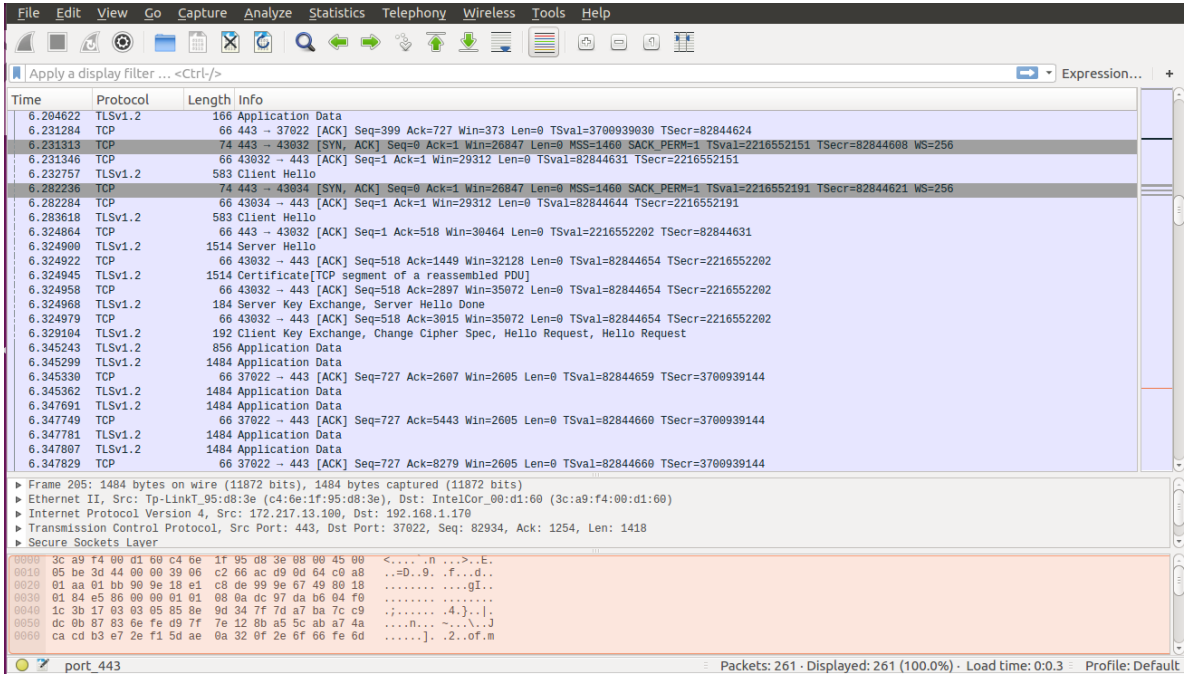
ntop Training

ntop TRAINING

Traffic Recording

Packet Analysis

- Dumping network traffic to disk and drilling down to the **packet level** is a well established technique for **troubleshooting**.



Show the PCAP or it didn't happen

- Large companies are often protected by firewalls and IDSs (Intrusion Detection Systems).
- However those security tools do not keep traffic history but just log **security events**.
- Thus it becomes difficult to provide evidence of an event.

Continuous Recording

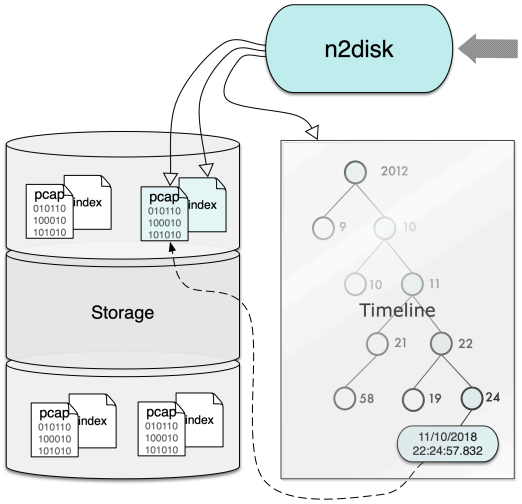
- In most cases it's not possible to **predict** when a network event occurs, **we need to record** traffic until the problem occurs.
- We need a "network VCR", recording Network traffic 24/7, similar to a closed-circuit camera.
- Continuous traffic recording provides a **window into network history**, that allows you to retrieve and analyse all the raw traffic in that period of time.

n2disk

- It relies on Open Source high-performance technologies for capturing and processing traffic, including our **PF_RING** framework, delivering line-rate packet capture up to 100 Gbit/s.
- It uses the de-facto standard **PCAP file format** to dump packets into files, so the resulting output can be easily integrated with existing third party and Open Source analysis tools like *ntopng*, *Wireshark, tcpdump, etc.*

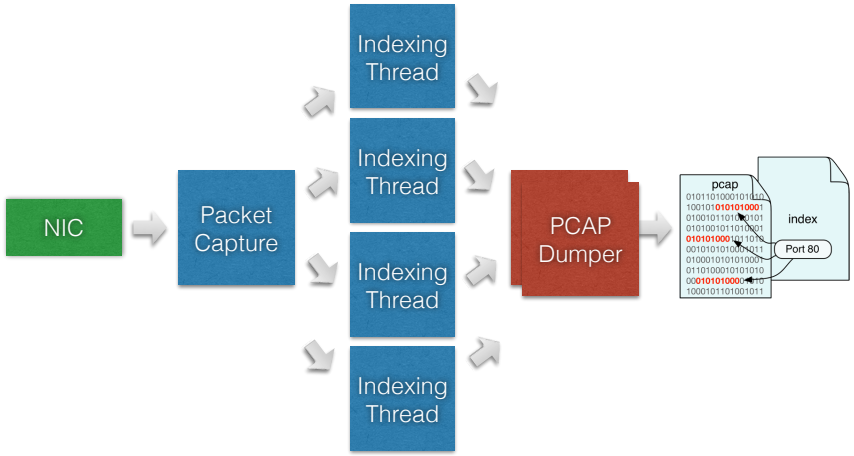
Indexing

- **Searching** for traffic matching IP addresses or sessions among tons of stored data might be challenging.
- n2disk indexes data **on-the-fly** while recording raw traffic, to give the flexibility to quickly retrive packets while the system is capturing at line-rate.
- Full packets are **stored, indexed and organized in a timeline** to enable on-demand retrieval, specifying **time** interval **and BPF** criteria to fully reconstruct past events.



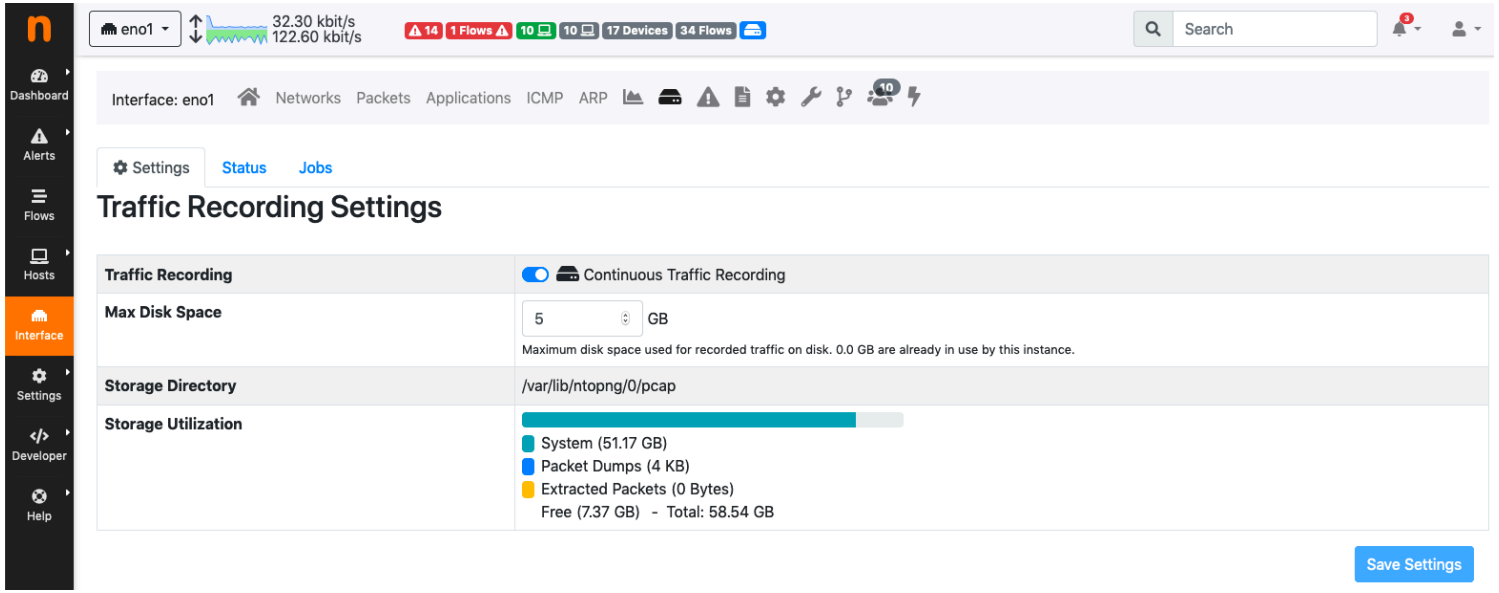
Technology

- n2disk has been designed to leverage on modern multicore architectures and scale up the performance
- Drawback: configuration at high speed may require some experience (e.g. threads to cores pinning)



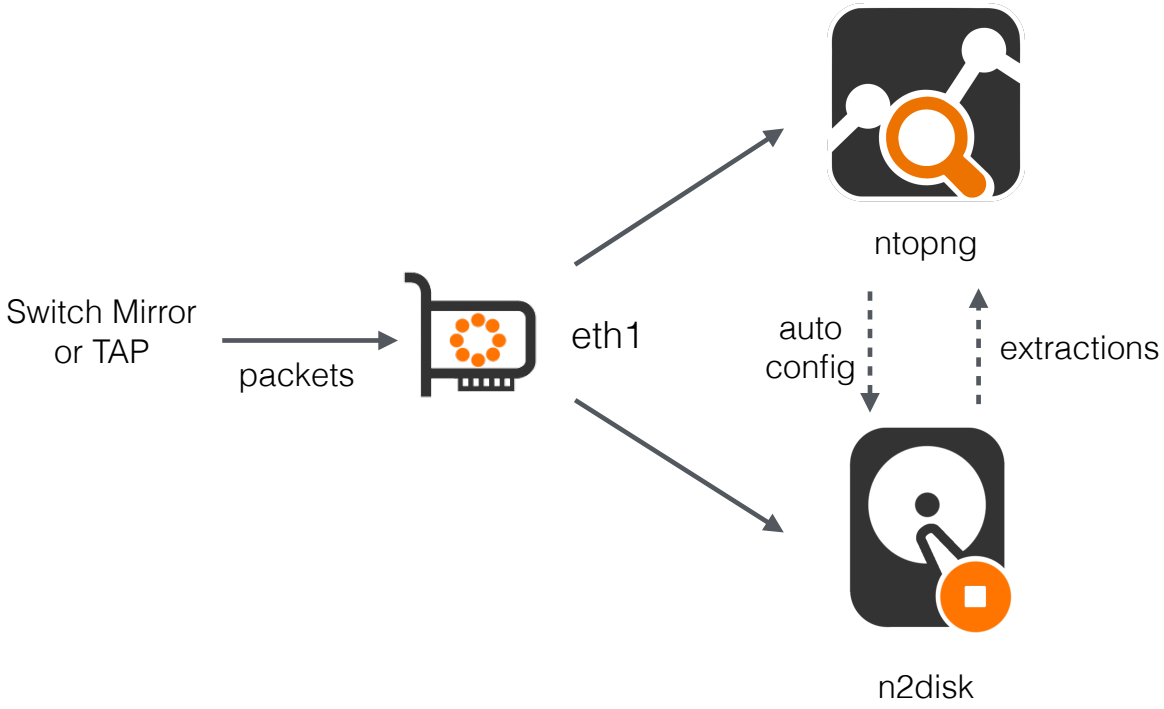
Simple Configuration (ntopng)

- Enable n2disk on the same interface used by ntopng
- Available when processing packets from an interface



ntop TRAINING

Simple Configuration (ntopng)



n2disk Storage Sizing (10 Gbit Link)

Max Data Rate	10 Gbps
Max Packet/s	14.88 Mpps
Data/s On Disk	1.2 GB/s
Data/h On Disk	4 TB/h
Data/day On Disk	100 TB/day

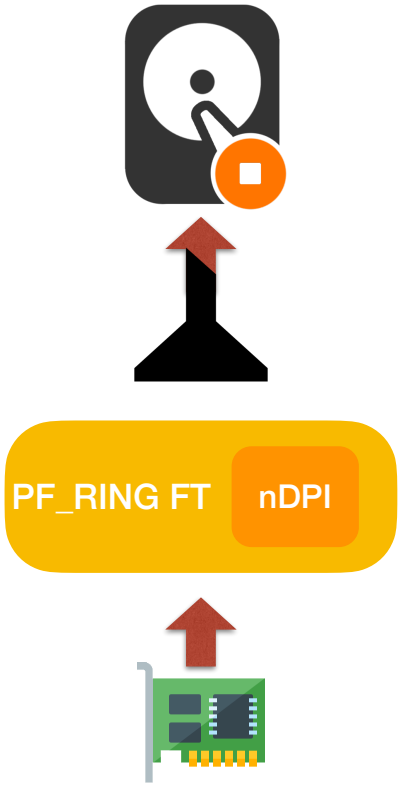
- consider some disk overhead for index and traffic extraction (around +10%)

Saving Space: Filtering

- Filtering can occur during or after capture:
  - **During** capture it allows traffic dumps to be reduced as **unwanted** traffic is **discarded**.
    Make sure the traffic portion you are dropping does not contain interesting packets!
  - **After** capture might require TBs of data to be scan in order to find the traffic we're interested in. Traffic indexing helps **accelerating** traffic **extraction**.
- The standard **BPF** syntax is supported both in capture filters and extraction filters.

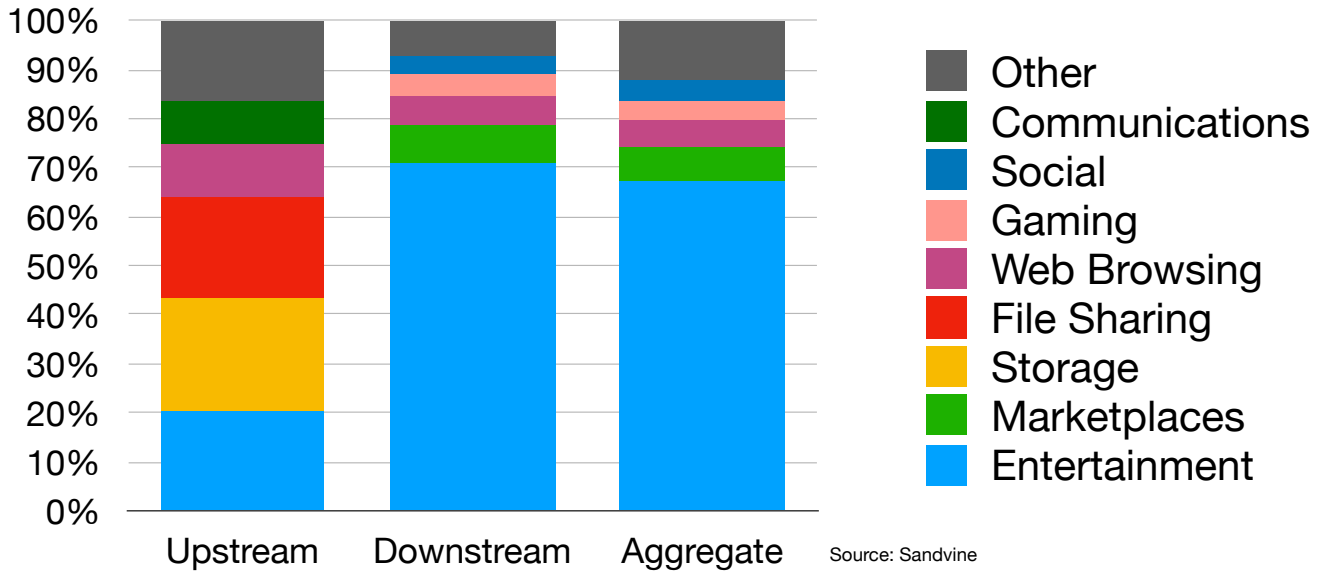
Saving Space: L7 Filtering

- Legacy BPF are usually not suitable to select the traffic to store.
- n2disk can leverage on the **PF_RING FT** classifier (based on **nDPI**) to filter L7 traffic.
- It is often useless to store encrypted traffic (besides the handshake), compressed traffic, multimedia traffic.
- Ability to discard or shunt.



Internet Traffic

- ~70% of Internet traffic during peak hours comes from video and music streaming.



Source: Sandvine

L7 Filtering Configuration

- A configuration file containing filtering rules can be provided to n2disk with the --l7-filter-conf <conf file> option. Example:

```
[shunt]
TLS = 10

[filter]
YouTube = discard
Netflix = discard
Spotify = discard
```

- A full list of available L7 protocols and categories is available with *ndpiReader -H*

Saving Space: Slicing

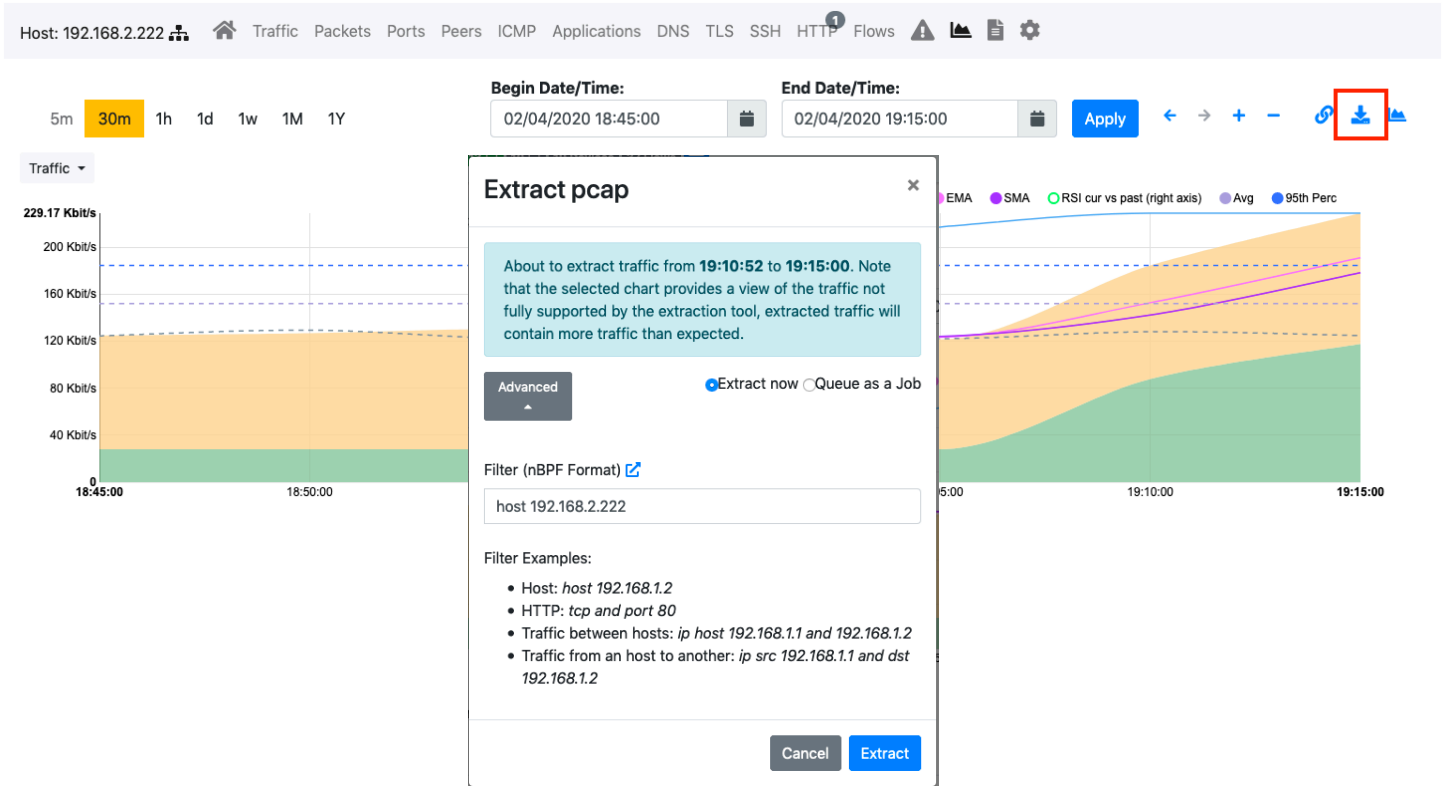
- Packet filtering can be rude, slicing can be a better option sometimes if we are interested in **headers only**.
- Packet slicing is the ability to reduce packet size by cutting them dynamically at specific sizes (e.g. up to the IP, or up to the TCP/UDP header).
- As network packets are in average ~512 bytes, this practice can help saving space (headers are usually <60 bytes)

Saving Space: PCAP Compression

- Packet compression can help depending on traffic type:
  - Internet traffic is already compressed (JPEG, MP3)
  - LAN traffic is often uncompressed (SQL, FTP…)
- You can save ~5% on Internet, and > 50% on LAN.
- Requires quite some CPU

Traffic Extraction

- ntopng GUI
  - Drill down from charts, alerts, historical data, ...



Traffic Extraction

- ntopng REST API

```
curl -u admin:password
"http://127.0.0.1:3000/lua/rest/get/pcap/live_extraction.lua?
ifid=0&epoch_begin=1746607618&epoch_end=1746610618&
bpf_filter=not+host+192.168.1.1" -o output.pcap
```

- Command-line

```
npcapextract -t /var/lib/ntopng/1/pcap
 -b “2025-04-21 16:00:00”
 -e “2025-04-21 16:05:00”
 -f “host 192.168.1.1 and port 80”
 -o output.pcap
```

ntop TRAINING

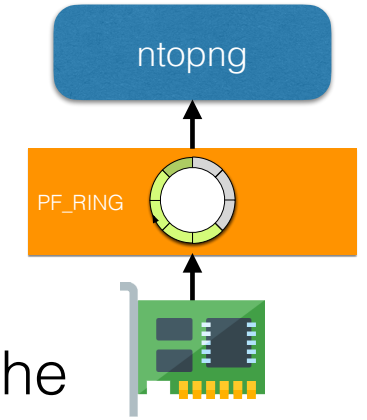
Performance & Tuning

n2disk

ntop TRAINING

Packet Capture

- PF_RING is a packet capture framework for improving the performance of network monitoring applications, accelerating packet capture.
- A commodity adapter, using standard drivers, can deliver 1-5 Gbps
- PF_RING offers on commodity hardware the ability to receive up to 100 Gbps with ZC (Zero-Copy) accelerated drivers



Multi-Vendor Support

- PF_RING natively supports many vendors (1/10/40/100 Gbit)

- PF_RING-based applications transparently select the module by means of the interface name:
  - `pfcount -i eth1` `[Standard Linux adapter]`
  - `pfcount -i zc:eth1` `[ZC for Intel]`
  - `pfcount -i mlx:mlx5_0` `[ZC for Mellanox/NVIDIA]`
  - `pfcount -i nt:1` `[Napatech]`
  - `pfcount -i fbcard:0:a:0` `[Silicom/Fiberblaze]`

PF_RING Service

- PF_RING is automatically loaded when installing pfring-dkms and can be controlled as a standard systemd service
- Check the service status:

```
$ systemctl status pf_ring
```

pfcount

- Packet capture with PF_RING. Print traffic statistics (add -v 1 to print all packets)

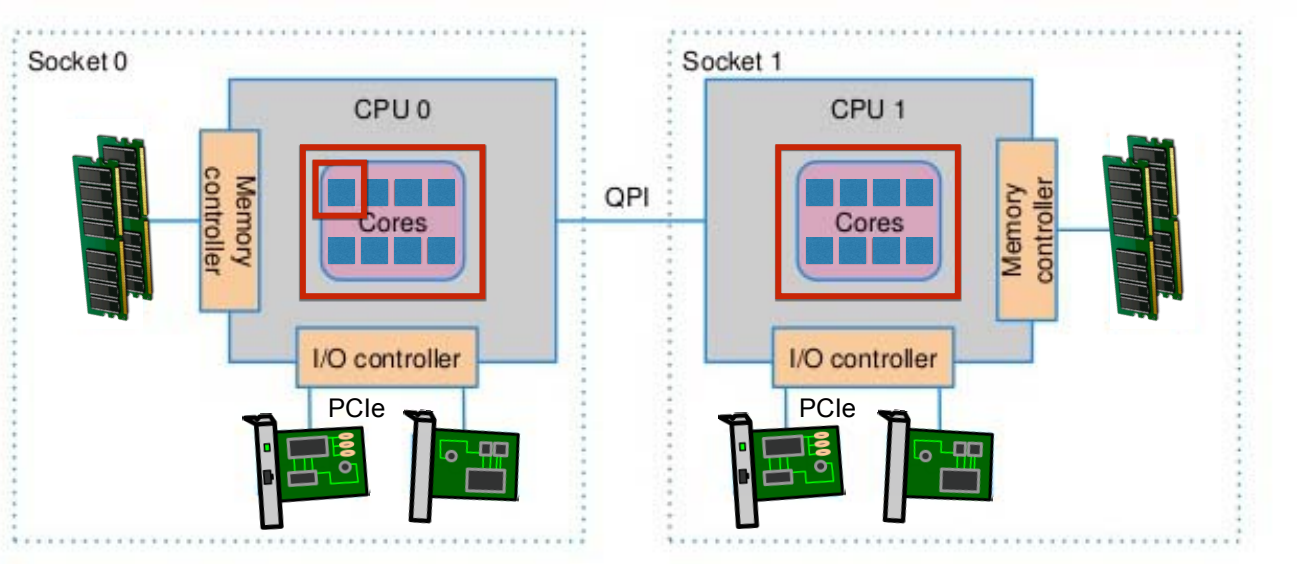
```
$ sudo pfcount -i eth1
Using PF_RING v.9.1.0
Capturing from eth1 [mac: 0C:C4:7A:CC:C4:4A][if_index: 3]
# Device RX channels: 4
# Polling threads:    1
Dumping statistics on /proc/net/pf_ring/stats/3358408-eno1.3
=========================
Absolute Stats: [87 pkts total][0 pkts dropped][0.0% dropped]
[87 pkts rcvd][19'009 bytes rcvd][29.00 pkt/sec][0.05 Mbit/sec]
=========================
Actual Stats: [54 pkts rcvd][1'000.09 ms][54.00 pps][0.00 Gbps]
=========================

(Ctrl + C to stop it)
```

CPU Cores

- CPU pinning of a process/thread to a core is important to isolate processing and improve performance.

```
# nprobe -i eth1 --cpu-affinity 0
```

- In most cases dedicating a physical core to each thread is the best choice for optimal performance.

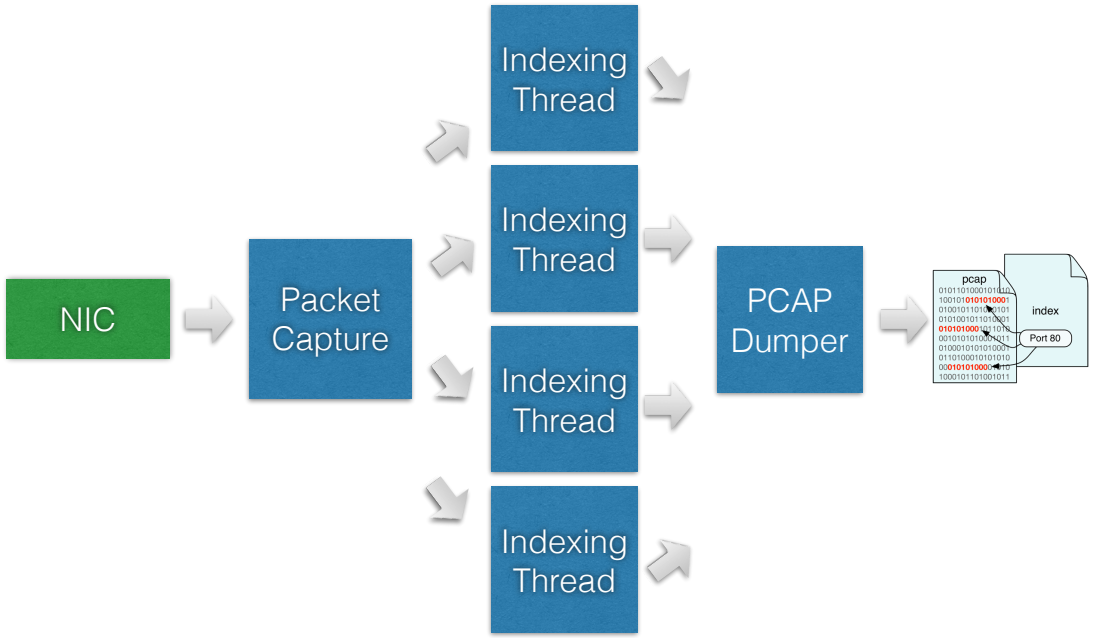


/proc/cpuinfo

- Check the CPU model and cores

```
$ cat /proc/cpuinfo | grep "model name" | head -n 1
model name : Intel(R) Xeon(R) E-2136 CPU @ 3.30GHz

$ cat /proc/cpuinfo | grep processor
processor  : 0
processor  : 1
processor  : 2
processor  : 3
processor  : 4
processor  : 5
```

n2disk Architecture (Threads)



n2disk Hardware Sizing: CPU/RAM

	<1 Gbit	10 Gbit	40 Gbit	100 Gbit
CPU	1 core (any)	Intel Xeon 4 cores 3Ghz	Intel Xeon 8 cores 3 Ghz	Intel Xeon 12 cores 3 Ghz
RAM	<1 GB	8 GB	32 GB	64 GB *

* all supported memory channels populated

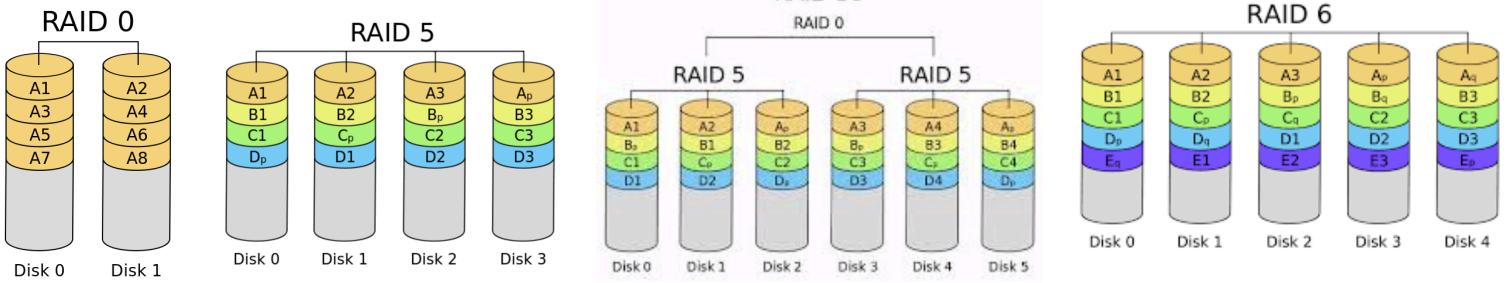
ntop TRAINING

Drives Performance

Drive Type	Random IOPS	Sustained Sequential I/O	
SAS/SATA 7,200RPM	70 – 175	100 – 230 MB/s	
SAS 10,000RPM	275 – 300	125 – 200 MB/s	
SAS 15,000RPM	350 – 450	125 – 200 MB/s	1-2 Gbps
2.5" Solid State (SSD)	15,000 – 100,000	110 – 500 MB/s	1-5 Gbps
PCI-E Solid State (SSD) / NVMe	70,000 – 625,000	1,100 – 3,200 MB/s	10-30 Gbps

RAID

- RAID is a good option for increasing disk bandwidth
- 8-10 HDD drives for 10 Gbit when using RAID 0, more drives are required with parity (e.g. RAID 5/50/6)



RAID Controller

- Minimum specs at 10+ Gbit:
  - Onboard cache: 1-2 GB
  - Stripe size: 1MB
  - Write policy: Always Write Back (Write Through writes data to disk on every write event, better to flush it on disk in large blocks, the controller knows when it's better to do it!)

How Many Drives - 10 Gbps

SAS/SATA	8-10
SSD	3-5
NVMe	1

Simulation Mode

- n2disk provides a simulation mode (no adapter or traffic required) to check the actual storage throughput and max dump speed simulating traffic at max speed.
- Command example:

```
n2disk -o /storage -p 2000 -b 8000 -c 1 -w 2 -v 1 -C 8192 -e 1
```

HDD vs SSD

- SATA/SAS 10k/15k RPM HDD drives are a good compromise in terms of price/number
- SSDs should be used when we need to read and write simultaneously to avoid seeking issues
- HDDs/SSDs are an option up to 40 Gbps (a common RAID controller can usually handle ~40 Gbps of write throughput)

NVMe

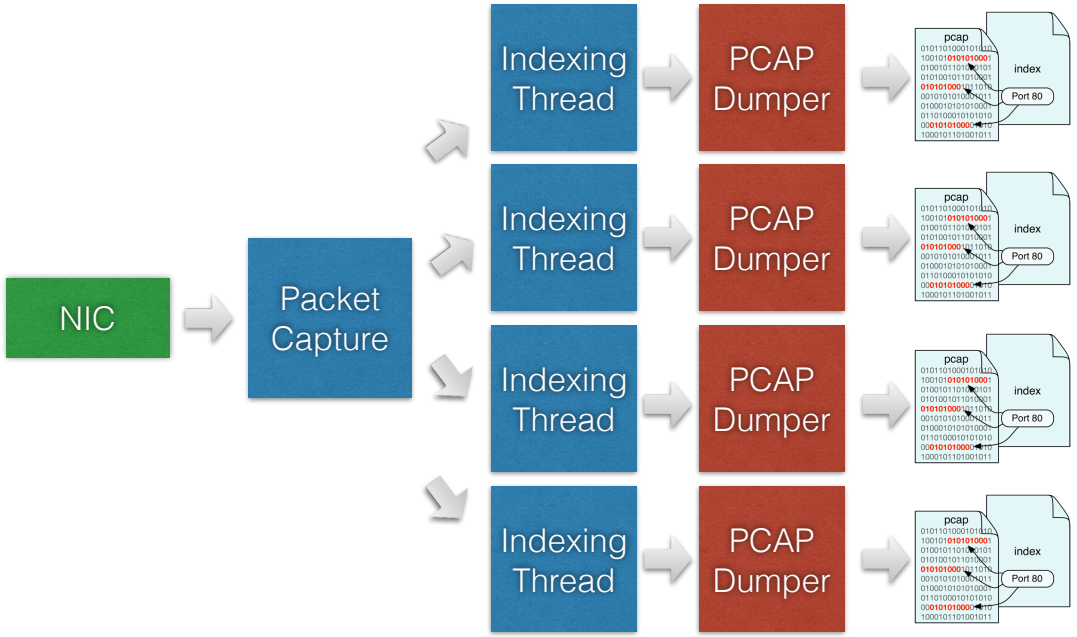
- NVMe drives are SSDs directly connected to the PCIe bus.
- NVMe are lightfast (~20-30 Gbps per disk).
- No need of a RAID controller (actually a standard SATA/SAS controller cannot drive them)
- Mandatory at 100 Gbps
- Choose write-intensive models!

How Many Drives - 100 Gbps

SAS/SATA	-
SSD	-
NVMe	6/8+

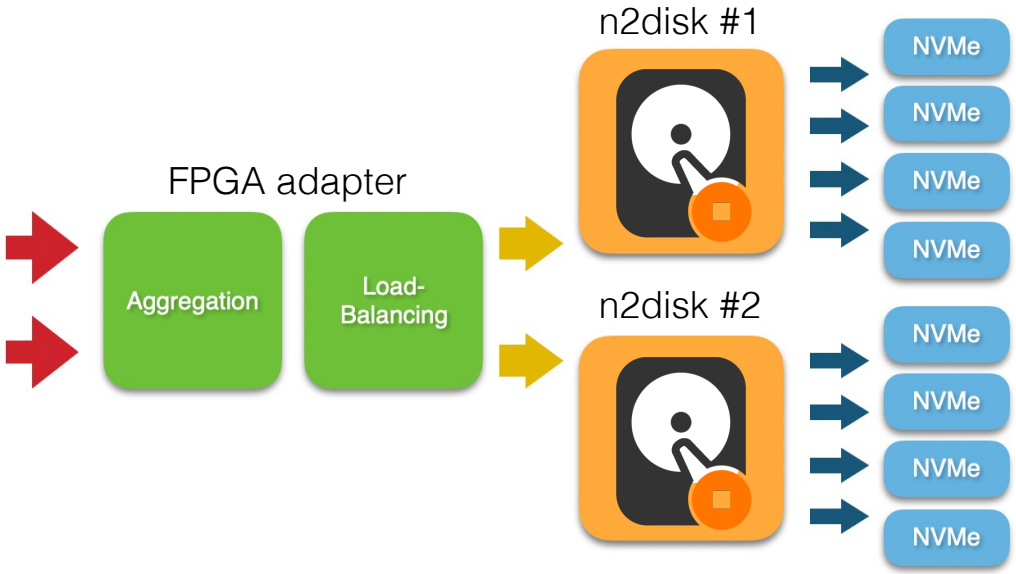
NVMe RAID Emulation

- Multithreaded parallel dump support in n2disk can write in parallel to multiple NVMe disks, emulating a RAID 0



100 Gbps Recording

- Load-balancing to 2 streams
- 2x n2disk instances, able to handle 50 Gbps each
- 8x total NVMe disks



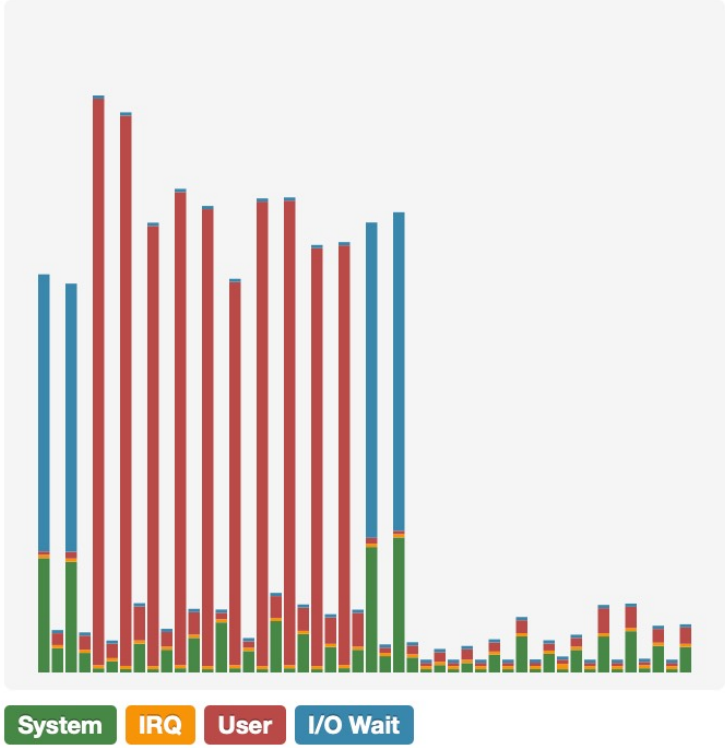
ntop TRAINING

Sample Configuration (50 Gbps)

```
# Capture interface
--interface=nt:stream0
# Storages (NVMe disks 1, 2, 3, 4)
--dump-directory=/storage1
--dump-directory=/storage2
--dump-directory=/storage3
--dump-directory=/storage4
# Disk space limit
--disk-limit=80%
# Max PCAP file size
--max-file-len=2048
# In-memory buffer size
--buffer-len=16384
# Chunk size
-C=16384
# Index and timeline
--timeline-dir=/storage1
--index
# Capture thread core affinity
--reader-cpu-affinity=0
# Indexing threads core affinity
--indexer-cpu-affinity=4,6,8,10
# Writer thread core affinity
--writer-cpu-affinity=22,22,22,22
```

CPU Load at 100 Gbps

- CPU cores utilization capturing, indexing and dumping 100 Gbps wire-rate (64-byte packets).



Which NIC?

- 1 Gbit
  - Any adapter
- 10 Gbit
  - Intel (no hw ts), NVIDIA/Mellanox (hw ts)
- 2x 10 Gbit aggregated
  - Napatech, Fiberblaze
- 40/100 Gbit
  - Napatech, Fiberblaze (both with segment mode, hw ts)
  - NVIDIA/Mellanox (many streams/cores required)

Performance & Tuning

ntopng / nProbe

ntop TRAINING

ntopng Hardware Sizing: Network Size

	Small Network	Medium Network	Large Network
Traffic	< 100Mbps	100Mbps - 1Gbps	Above 1Gbps
Active Hosts	Hundreds	Thousands	Hundreds of thousands
Active Flows	Thousands	Hundreds of thousands	Millions

ntop TRAINING

ntopng Hardware Sizing: CPU/RAM

	Small Network	Medium Network	Large Network
CPU	2+ cores	4+ cores	8+ cores
RAM	2+ GB	4+ GB	16+ GB

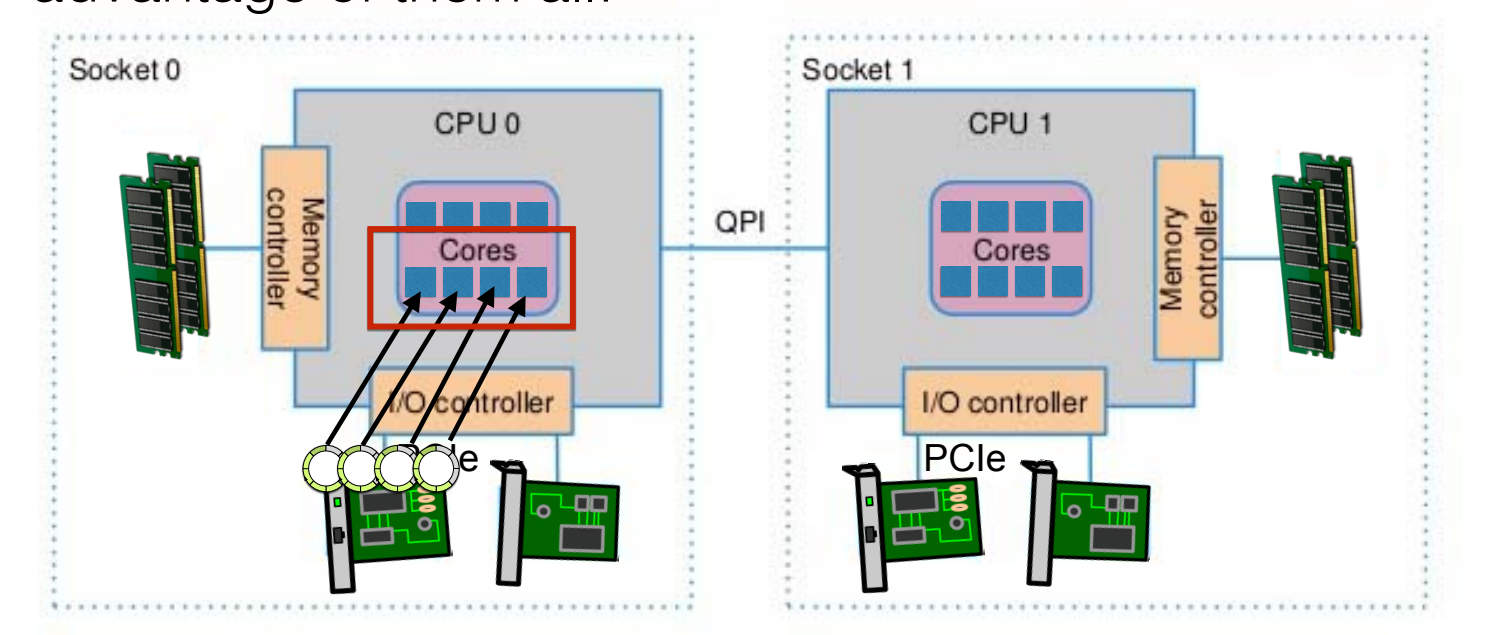
ntop TRAINING

Host and Flow Cache (ntopng)

- ntopng keeps active Flows and Hosts in a cache in memory
- Flows/hosts cashes have a preconfigured maximum size
- A red badge in the ntopng GUI indicates that the cache size should be increased
- It is possible to increase the size with -x and -X options
  - It is recommended to use values much higher than the actual number of hosts and flows (at least 2x)
  - Example: -x 100000 -X 200000

Load Balancing

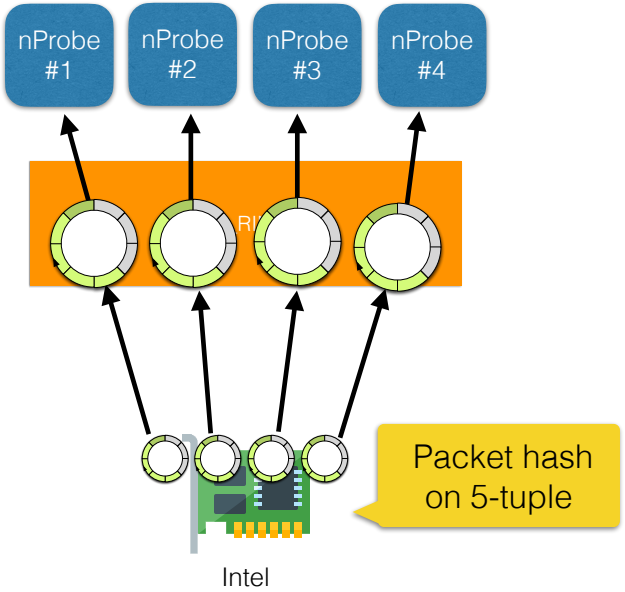
- Many cores usually available: we should take advantage of them all!



- Split the traffic to multiple queues, and process it with multiple threads/processes on many cores.

RSS

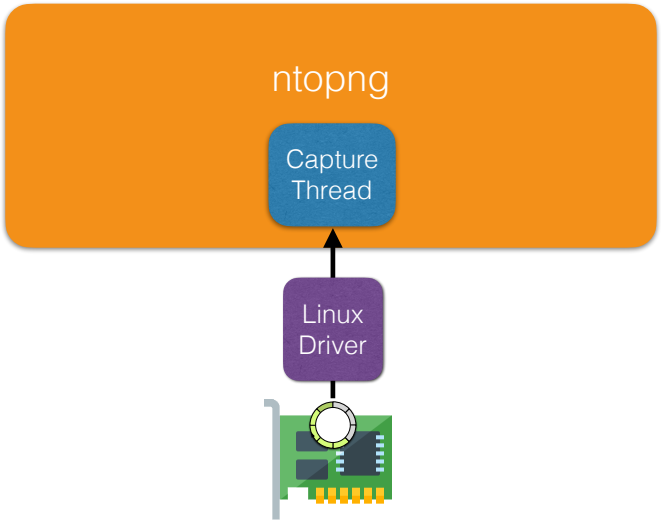
- RSS is a hardware technology that distributes the load across multiple queues keeping flow coherency.
- This technology is supported by ZC driver for Intel and NVIDIA (as well as FPGAs).
- Capture performance can scale with the number of RSS queues!



Example: ntopng

- Standard Linux driver:

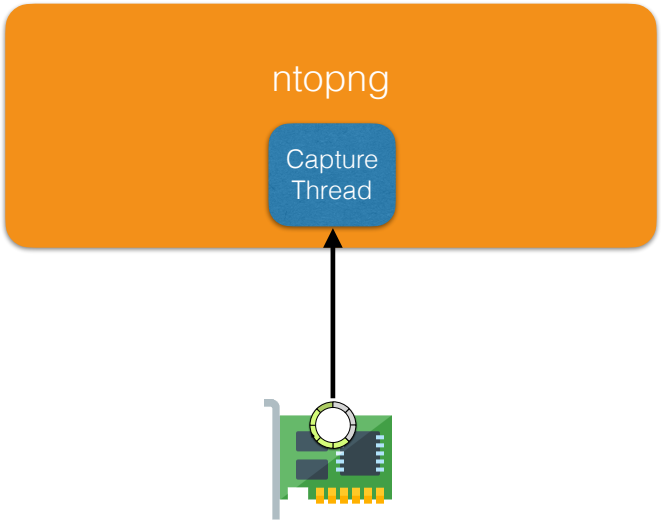
```
ntopng -i eth1
```



Example: ntopng

- PF_RING ZC driver:

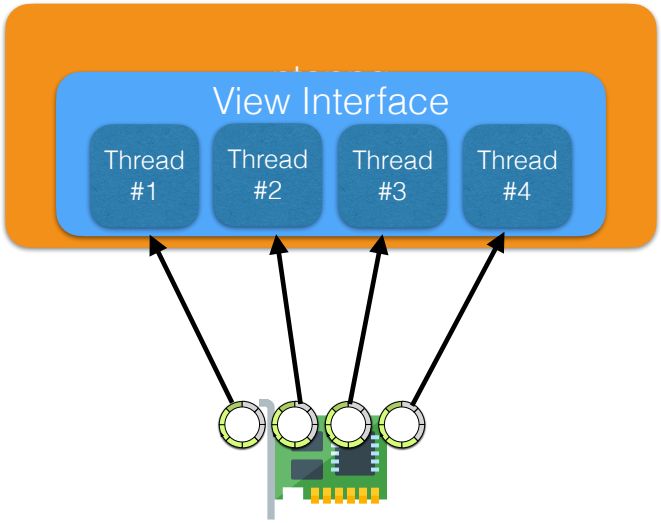
```
ntopng -i zc:eth1
```



Example: ntopng

- ntopng supports multiple interfaces (or multiple queues), performance can scale capturing from multiple RSS queues using multiple threads

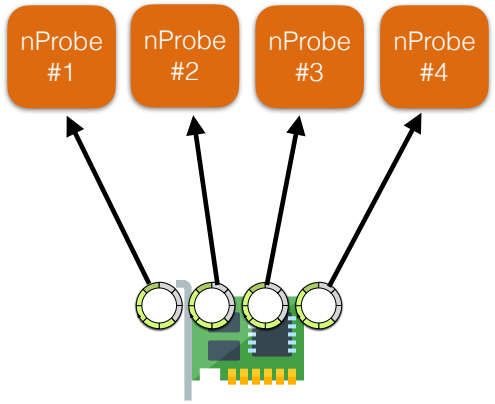
```
ntopng -i zc:eth1@0 -i zc:eth1@1 -i view:all
```



Example: nProbe

- nProbe performance can scale running multiple instances, one per RSS queue

```
nprobe -i zc:eth1@0 --cpu-affinity 0 --zmq tcp://..
nprobe -i zc:eth1@1 --cpu-affinity 1 --zmq tcp://..
nprobe -i zc:eth1@2 --cpu-affinity 2 --zmq tcp://..
nprobe -i zc:eth1@3 --cpu-affinity 3 --zmq tcp://..
```



ntop TRAINING

RSS Configuration

- Set the number of RSS queues (ZC drivers):

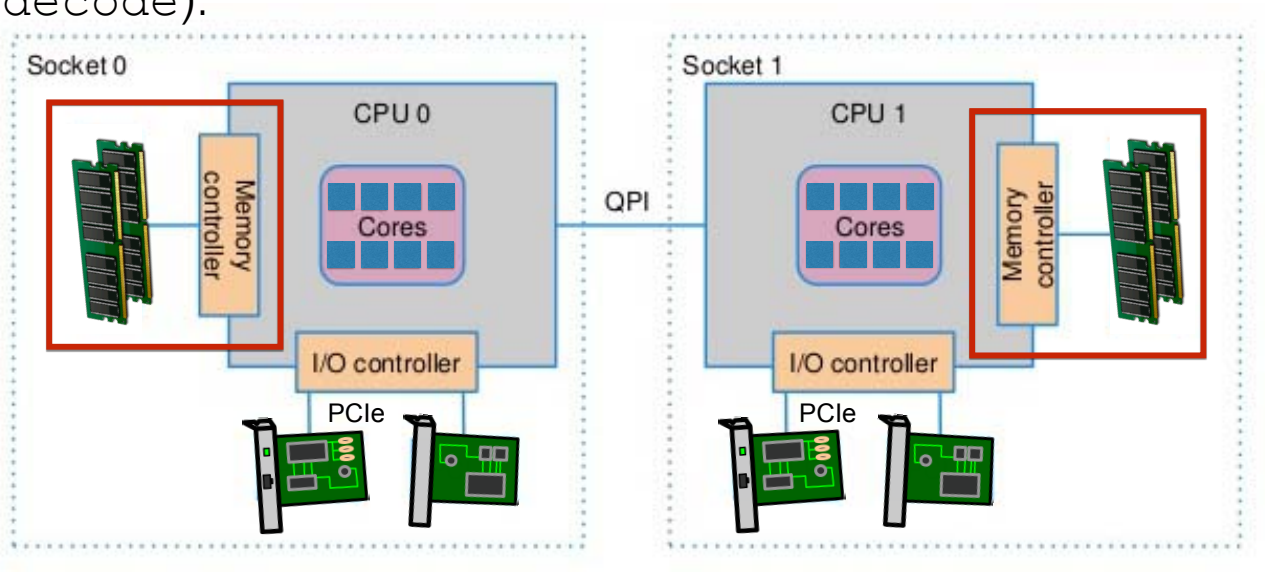
```
# pf_ringcfg --configure-driver ixgbe --rss-queues 4
```

- Check the number of RSS queues:

```
# cat /proc/net/pf_ring/dev/eth1/info | grep Queues
TX Queues:     4
RX Queues:     4
```

Memory Channels

- Multi-channel memory increases data transfer rate between memory and memory controller.
- Check how many channels your CPU supports and use at least as many memory modules as the number of channels (check `dmidecode`).

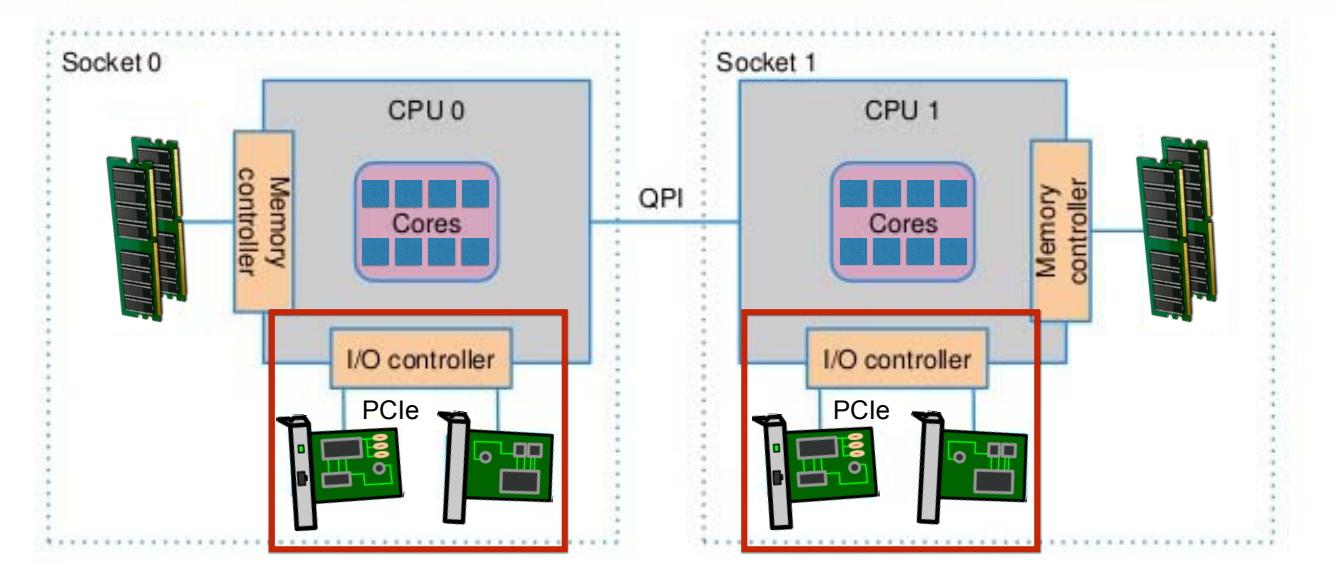


dmidecode

```
$ sudo dmidecode | grep “Speed\|Locator"

   Locator: DIMMA1
   Bank Locator: P0_Node0_Channel0_Dimm0
   Speed: Unknown
   Configured Memory Speed: Unknown

   Locator: DIMMA2
   Bank Locator: P0_Node0_Channel0_Dimm1
   Speed: 2667 MT/s
   Configured Memory Speed: 2667 MT/s

   Locator: DIMMB1
   Bank Locator: P0_Node0_Channel1_Dimm0
   Speed: Unknown
   Configured Memory Speed: Unknown

   Locator: DIMMB2
   Bank Locator: P0_Node0_Channel1_Dimm1
   Speed: 2667 MT/s
   Configured Memory Speed: 2667 MT/s
```

PCIe

- Each node has its dedicated PCIe lanes.
- Plug the Network Card (and the RAID Controller) to the right slot reading the motherboard manual.



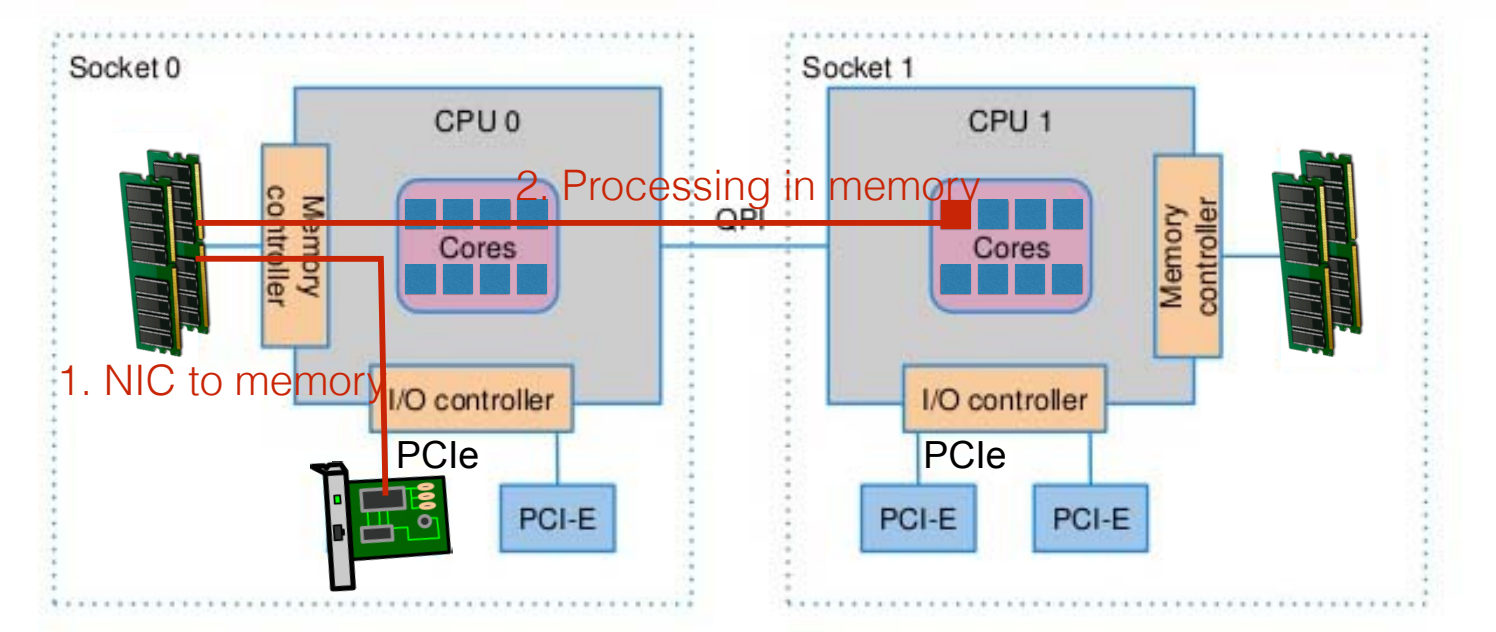
lstopo

- Detect the hardware configuration (CPU cores, NUMA nodes and connected devices, …)

```
$ lstopo
Machine (23GB total)
  Package L#0
    NUMANode L#0 (P#0 23GB)
    L3 L#0 (8192KB)
      L2 L#0 (256KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0
        PU L#0 (P#0)
        PU L#1 (P#4)
      L2 L#1 (256KB) + L1d L#1 (32KB) + L1i L#1 (32KB) + Core L#1
        PU L#2 (P#1)
        PU L#3 (P#5)
      L2 L#2 (256KB) + L1d L#2 (32KB) + L1i L#2 (32KB) + Core L#2
        PU L#4 (P#2)
        PU L#5 (P#6)
      L2 L#3 (256KB) + L1d L#3 (32KB) + L1i L#3 (32KB) + Core L#3
        PU L#6 (P#3)
        PU L#7 (P#7)
  HostBridge
    PCIBridge
      PCI 02:00.0 (Ethernet)
        Net "enp2s0f0"
      PCI 02:00.1 (Ethernet)
        Net "enp2s0f1"
    PCI 00:19.0 (Ethernet)
      Net "eno2"
  Package L#1
    NUMANode L#1 (P#0 23GB)
    …
```

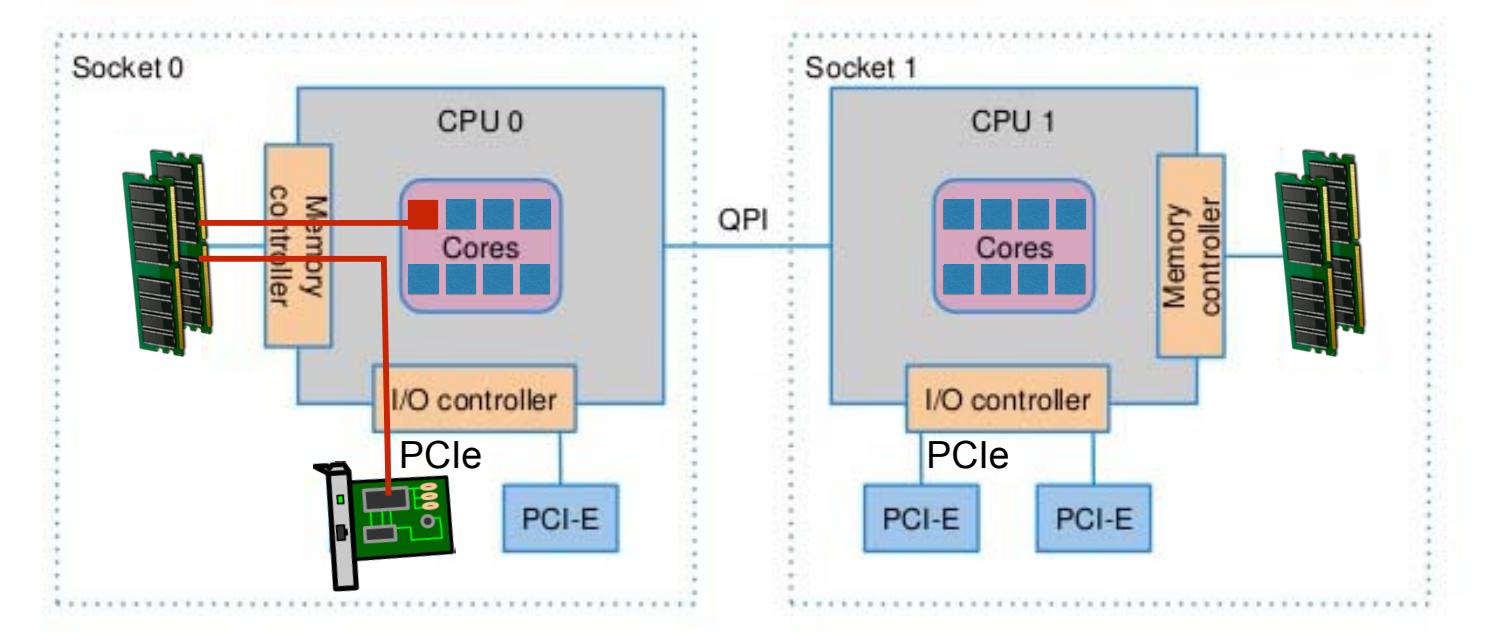
Example: Wrong Configuration

```
nprobe -i zc:enp2s0f0 --cpu-affinity 8
```



Example: Correct Configuration

```
nprobe -i zc:enp2s0f0 --cpu-affinity 0
```



Questions?

Managing Licenses

Dynamic Environments & K8s

ntop TRAINING

System ID

- Licenses are based on the System ID, a unique system identifier based on hardware configuration
  - Example: ntopng -V
- The System ID may change when replacing a Network adapter or when migrating the software to a new appliance (e.g. in case of hardware failures)
  - Migrate the license at https://shop.ntop.org/recover_licenses.php

Virtual Machines

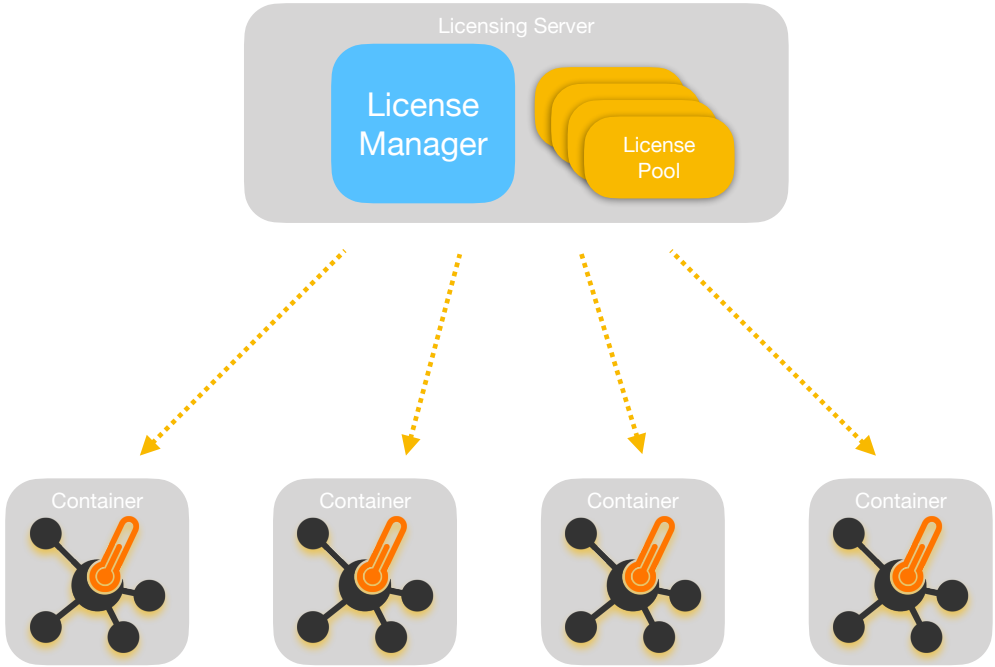
- The System ID is based on the hardware configuration (virtualized hardware in case of VMs)
- The System ID is migration-resistant
  - Don't worry about changes to the underlying hardware configuration when using live migration

Containers

- If you are running a container (e.g. Docker)
  - Use the license generated for the host system, rather than generating a license for each container
    - The container requires visibility on the host hardware configuration, including network devices: configuring the container to use the host network namespace (in Docker use *–network=host*)
  - Or use the License Manager...

Dynamic Environments & K8s

- LM dynamically associates licenses to instances



ntop TRAINING