hwloc-ls gives us a nice view into the system. What the NUMA nodes are, which devices are connected to each node. Also what the cpu id's are for the nodes.

Example output:

```sh
data hwlloc-ls
Machine (63GB total)

NUMANode L#0 (P#0 31GB)
  Package L#0 + L3 L#0 (30MB)
    L2 L#0 (256KB) + L1d L#0 (32KB) + L1i L#0 (32KB) + Core L#0
      PU L#0 (P#0)
      PU L#1 (P#24)
...
    L2 L#11 (256KB) + L1d L#11 (32KB) + L1i L#11 (32KB) + Core L#11
      PU L#22 (P#11)
      PU L#23 (P#35)
  HostBridge L#0
  PCIBridge
    PCI 1000:0086
      Block(Disk) L#0 "sda"
  PCIBridge
    PCI 19ee:4000
      Net L#1 "ens1np0"
      Net L#2 "ens1np1"
  PCIBridge
    PCI 8086:1d6b
    PCI 8086:1502
      Net L#3 "eno1"
  PCIBridge
    PCI 8086:10d3
      Net L#4 "enp1s0"
  PCIBridge
    PCI 10de:128b
      GPU L#5 "renderD128"
      GPU L#6 "controlD64"
      GPU L#7 "card0"
  PCI 8086:2826
NUMANode L#1 (P#1 31GB)
  Package L#1 + L3 L#1 (30MB)
    L2 L#12 (256KB) + L1d L#12 (32KB) + L1i L#12 (32KB) + Core L#12
      PU L#12 (P#12)
      PU L#25 (P#36)
...
    L2 L#23 (256KB) + L1d L#23 (32KB) + L1i L#23 (32KB) + Core L#23
```
There are 4 NICs in this machine: 2 Dual port Netronome cards (ens3np* on NUMA node 1, ens1np* on node 0. Built-in NICs enp1s0 and eno1 also on node 0).

We could use this info in properly setting up CPU affinity for Suricata.

I'm assuming that libhwloc exposes this info in way that Suricata would use it.

**Goals:**
- review hwloc availability and versions for our 'tier 1' and 'tier 2' supported OS', distros.
- create a PoC where configure detects and enables libhwloc and prints the NUMA node for the interface Suricata intends to use (single iface is ok for the PoC)
- determine if the lib is suitable for the autoconfig goal

**Bigger picture:**
- idea is to allow a option to suri like --numa-from-nic (name TBD) that would take the numa node for the nic, then set cpu affinity and thread counts to only use that numa node.
- in multi-nic capture, setup threads incl affinity according to numa config
- if possible, detect and warn on misconfiguration by numactl (e.g. nic is on numa node 0, threads are forced on node 1)
- simplify manual configuration. E.g. instead of cpu: [ 0, 2, 4, 6, 8, 16, 18, 20, 22 ] something like numa: [ 0 ]

**Related issues:**
Related to Task #3318: Research: NUMA awareness

---

**History**

**#1 - 05/12/2020 08:58 AM - Victor Julien**
- Status changed from New to Assigned
- Assignee set to Shivani Bhardwaj
- Target version set to 6.0.0beta1

**#2 - 06/01/2020 02:29 PM - Victor Julien**
- Target version changed from 6.0.0beta1 to 7.0rc1

**#3 - 11/07/2020 07:42 AM - Victor Julien**
- Related to Task #3318: Research: NUMA awareness added

**#4 - 03/11/2021 08:23 AM - Shivani Bhardwaj**

As of May 2020, on hwloc v2.2.0, there were following findings based on the goals defined for this task.

**Available components**
- Linux: official component for discovering CPU, memory and I/O devices in linux. It discovers PCI devices without the help of external libraries such as libpciaccess but requires the pci component for adding vendor/device names to PCI objects. It also discovers many kinds of linux specific OR devices.
- Aix, darwin, freebsd, NetBSD, Solaris, Windows: Each officially supported OS has its own native component which is statically built when supported and which is used by default.

A lot more available on https://www-lb.open-mpi.org/projects/hwloc/doc/v2.0.1/a00324.php#plugins_list

**Integration with Suricata**
- On Linux, it seems to work. There is an elaborate API provided by Hwloc that can be used to access all nodes of the topology.
- The PoC checks for hwloc library’s presence on the system if configured with --enable-hwloc option
- Looks for the one and only interface that Suricata is currently using
- Looks for NUMA nodes attached to that interface and prints out "FOUND THE NUMA node"

Code for the topology on my then system can be found here: [https://github.com/inashivb/suricata/tree/hwloc-poc/v1](https://github.com/inashivb/suricata/tree/hwloc-poc/v1)

Victor took a look at this and modified some parts to make it work on the topology of his system. The relevant conversation was:
So what I did was very generic I think: find the NIC and walk back until we find the package. That then knows the numa id

yeah but if its the machine as was in my case there's nothing to walk back to. I don't know if there can be any more topology structures than these

not even a machine or package?

Machine is the root so we walk down from there

I think the reverse makes more sense. Use the search func to find the pci id, then walk backwards towards the parents

Maybe we can just:

```
cat /sys/class/net/enp8s0/device/numa_node
```

instead...

Hmm not sure why I get -1 there

I don't get it, on another box I see

```
HostBridge L#0
 PCI 144d:a801
   Block(Disk) L#0 "sdb"
 PCIBridge
 PCI 10de:1c03
   GPU L#1 "renderD128"
   GPU L#2 "card0"
 PCI 8086:2827
 PCI 8086:15a0
 Net L#3 "eth0"
```

this I want everywhere