In several talks at suricon we've seen that the best performance happens when the NIC and suricata are on the same NUMA node, and that Suricata should be limited to this node.

Even in a multi-NIC scenario, Suricata will likely not perform well when running on multiple nodes at once, as global data structures like the flow table are then accessed/updated over the interconnects a lot.

Evaluate what strategies exist.

Reading material:
https://www.akkadia.org/drepper/cpumemory.pdf
https://stackoverflow.com/a/47714514/2756873

Several possible subtasks come to mind:

1. making configuration easier: take NUMA into account when configuring CPU affinity. Currently a list of CPUs has to be provided, which can be tedious and error prone. libnuma could help with identifying the CPUs belong to a node.

2. assign memory to specific nodes: the default allocation behaviour (at least on Linux) seems to already be that the allocating thread allocates memory in its own node. For packets we already do this correctly, with packet pools initialized per thread, in the thread. But for example the flow spare queue is global and the flows in it are initially alloc'd from the main thread, and later updated from the flow manager. This means these flows will likely be unbalanced and lean towards one node more than others. Creating per thread flow spare queues could be one way to address this. Similarly for other 'pools' like stream segments, sessions, etc.

3. duplicate data structures per node. Not sure yet if this a good strategy, but the idea is that something like the flow table or detect engine would have a copy per node to guarantee locality. In a properly functioning flow table this should be clean, as the flows should stay on the same thread (=CPU). For the detection engine this will pretty much duplicate memory use for the detection engine. Unless loading is done in parallel, start up time would also increase.
Assignee changed from OISF Dev to Victor Julien

#6 - 11/23/2019 10:55 PM - Andreas Herz
Do we also have some more insights how this does affect the management threads for example? If we can at least move those to a different node to keep the other cpu cores free for the heavy tasks?

#7 - 11/24/2019 06:33 AM - Victor Julien
They would probably have to run on the same node as where the traffic is and where the memory for that traffic is owned to avoid accessing locks over the interconnects.

#8 - 11/07/2020 07:42 AM - Victor Julien
- Related to Task #3695: research: libhwloc for better autoconfiguration added