NUMA-Aware Update

Given we're often stuck with a full drive or want to start transferring data to collaborators while observations are on-going, I had a look into the NUMA setup of UCC1 to see if we could try keep recording processing on one NUMA node and offloading on another node.

Jargon

NUMA: Non-uniform memory access -- consider that we have dual CPU machines, these are split into 2 NUMA nodes to reflect the increase in latency between accessing something in the cache or memory attached to one of the CPUs from another.

The System Overview

top.png

Each of the REALRA nodes has two NUMA nodes, each with different components attached to them via the PCIe bus. We can see that node #0 has

- The fibre connections (eno* devices)
- The Tesla V100 (card0)

While node #1 has

- The storage devices (sd*)
- The infiniband networking card (ib*)

This is almost ideal for what we want to achieve, as we can restrict the recoridng processes to node #0 (with the fibre card having DMA to node #0 as a result, but writing to the drives at node #1) while other processes can execute access the disks on node #1.

As a result, I modified the normal `generic_ucc1.sh` script to constrain Olaf's recorder to NUMA node 0 using `numactl`,

```
numactl -m 0 /home/obs/Joe/Record_B1508/$recording_program --ports 16130 ... &
numactl -m 0 /home/obs/Joe/Record_B1508/$recording_program --ports 16131 ... &
numactl -m 0 /home/obs/Joe/Record_B1508/$recording_program --ports 16132 ... &
numactl -m 0 /home/obs/Joe/Record_B1508/$recording_program --ports 16133 ... *
```

This mean the recorder will allocate memory and only perform processing on NUMA node 0. This include any child processes used by zstd for compression, though we could modify the soruce to force that process to stay on NUMA node #1 if we want it to have direct, priority access to the disks.

Some other kernel parameters were also changed for this test, namely the size of the UDP buffer queue and the maximum size. These were increased from 26kb (~1 packet/port) and 1,000 respectively to ~250MB (~0.1 seconds of data/port) and ~500k packets, though the increase in queue length was likely not needed as the kernel never reported a value higher than 200.

With these changes, two obsevrations were performed while an rsync command was moving data from UCC1to UCC3 across the infiniband network, while being forced onto node #1. Packet loss was minimal during the first observation, with the worst port losing 2,105 packets across a 7 hour observation (0.000650% packet loss). The transfer finished a few hours in to the second observation, with the worst port losing 1,726 packets across the 6.5 hours (0.000542% packet loss).

While these values are low, by plotting the packet loss during the second observation it is extremely clear when the transfer stopped occurring:

packetloss.png

However, with the reduced packet loss levels (2,000 packets corresponds to 0.16 seconds) this setup should allow us to perfrm some amount of data transfer off UCC1. However I am unsure as to the effect of remote transfers as these will likely traverse the fibre line which will likely elevate the packet loss even further, though I'll say it's worth looking in to the loss rates during one of the upcoming local modes.

Revision #5
Created 7 August 2020 16:36:15 by David
Updated 7 August 2020 17:16:13 by David