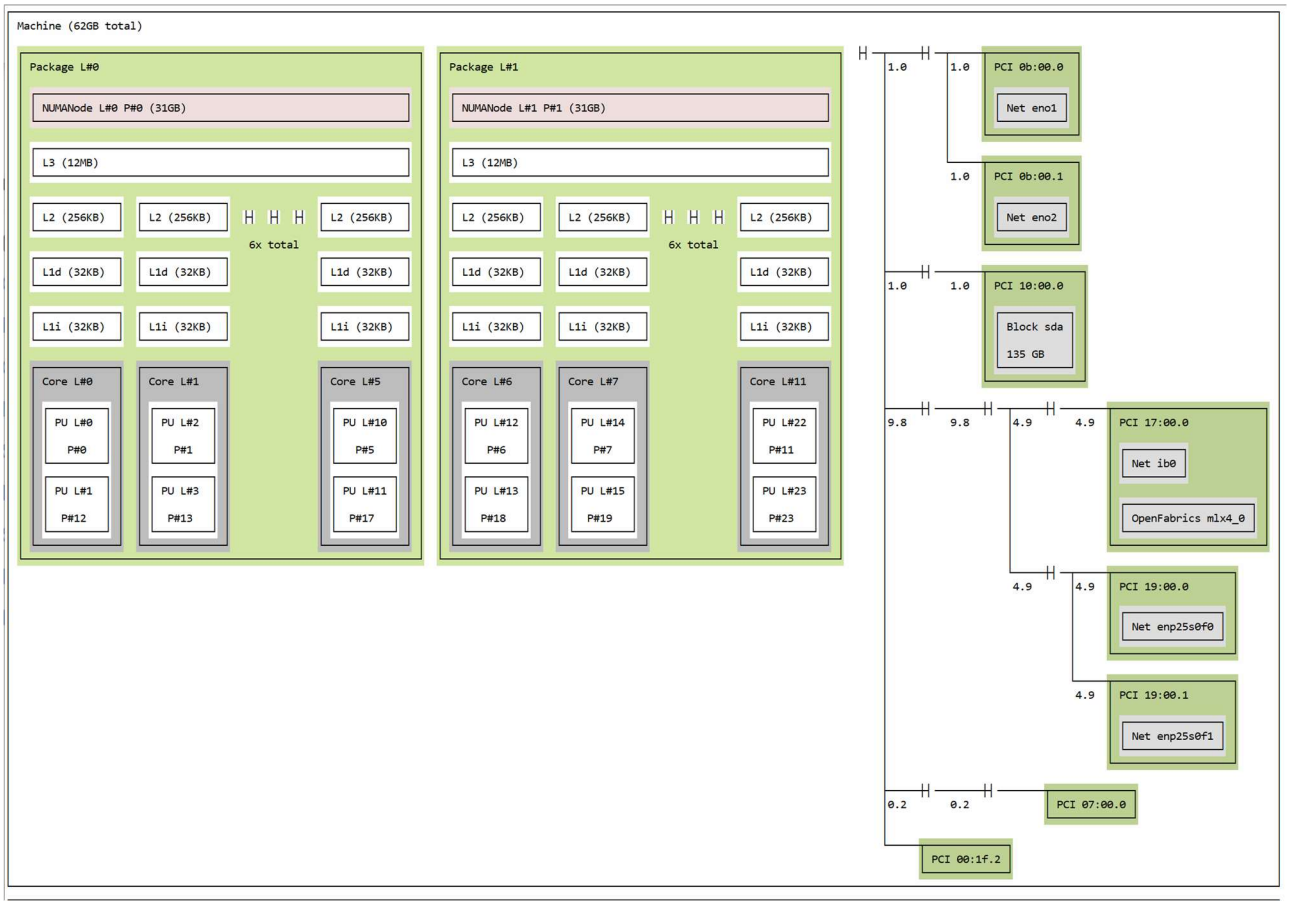
Assignment 2

Task1



- Can you retrieve the number of CPUs and cores from the output?
  - ➔ 12 Cores numbered from #0 to #11combined in the 2 nodes.

- Investigate the memory hierarchy information provided by lstopo.
  - How much memory (RAM) does the compute node offer?
    - ➔ The picture above is measured from the login node which offers 62GB RAM distributed over 2 Nodes. I tried the same for the computing nodes the numbers where 47GB distributed over the 2 Nodes.

  - What does the term `NUMANode` tell you about the memory? Why are there two `NUMANodes`?

➔ In multiple CPU architectures the RAM is usually split so that a portion of it is assigned to each core. If CPU0 needs data from CPU1's RAM, then it needs to "request" this data through CPU1.

Non-uniform memory access (NUMA) is a computer memory design used in multiprocessing, where the memory access time depends on the memory location relative to the processor. Under NUMA, a processor can access its own local memory faster than non-local memory (memory local to another processor or memory shared between processors). The benefits of NUMA are limited to particular workloads, notably on servers where the data is often associated strongly with certain tasks or users.

- What interesting information can you retrieve about the caches of the system?

➔ L#i = Instruction Cache, L#d = Data Cache. L1 = a Level 1 cache.

(# being the number)

So each core has one instruction and data cache apart from a Level 2 cache. About cache levels:

L1 Cache (Level 1 Cache):

- L1 cache is the smallest and fastest cache, located closest to the CPU cores.
- It is usually split into separate instruction cache (L1i) and data cache (L1d).
- L1 cache has the lowest latency and is used to store frequently accessed data and instructions.

L2 Cache (Level 2 Cache):

- L2 cache is larger but slower than L1 cache.
- It sits between the L1 cache and the main memory (RAM) in the cache hierarchy.
- L2 cache serves as a secondary cache and helps to reduce the average memory access time by storing additional copies of frequently accessed data.

L3 Cache (Level 3 Cache):

- L3 cache is the largest and slowest cache in the hierarchy.
- It is shared among multiple CPU cores within a processor or across different CPU cores in a multi-processor system.
- L3 cache helps improve overall system performance by providing a larger cache space for storing frequently accessed data that may not fit in the smaller L1 and L2 caches.

- Is there anything else you can find out from the output?

  You can see the PCI `hierarchy from the documentation:`

  - The PCI hierarchy is not drawn as a set of included boxes but rather as a tree of bridges (that may actually be switches) with links between them. The tree starts with a small square on the left for the hostbridge or root complex. It ends with PCI device boxes on the right. Intermediate PCI bridges/switches may appear as additional small squares in the middle.
  - PCI devices on the right of the tree are boxes containing their PCI bus ID (such as 00:02.3). They may also contain sub-boxes for OS device objects such as a network interface eth0 or a CUDA GPU cuda0.
  - When there is a single link (horizontal line) on the right of a PCI bridge, it means that a single device or bridge is connected on the secondary PCI bus behind that bridge. When there is a vertical line, it means that multiple devices and/or bridges are connected to the same secondary PCI bus.
  - The datarate of a PCI link may be written (in GB/s) right below its drawn line (if the operating system and/or libraries are able to report that information). This datarate is the currently configured speed of the entire PCI link (sum of the bandwidth of all PCI lanes in that link). It may change during execution since some devices are able to slow their PCI links down when idle.

- Based on your observations, how many threads could you utilize at maximum when parallelizing a program with OpenMP on this system?

  ➔ Not knowing the limitations of OMP I can say by only looking at the output of lstopo, that each core utilizes 2 processor units, which each can carry out computations separately i.e. threads.

  So the number of threads is about double the amount of cores -> 24

- Compare your observations to the LCC3 documentation and CPU documentation. Do your observations match the documentation?

  ➔ The number of cores is the same since 2 cpus are used in each computing node, which also checks out when looking at the amount of NUMAnodes. Other than that the RAM checks out as well when looking at the compute nodes 24 GB as written in the documentation (It's not exactly the same but my guess its rounded or lstopo is not 100% exact).

Task 2

Speedup achieved by a parallel algorithm is defined as the ratio of the time required by the best sequential algorithm to solve a problem, T(1), to the time required by parallel algorithm using p processors to solve the same problem, T(p).

Amdahl's law:

$$S(p) = \frac{t_s}{ft_s + (1-f)t_s/p} = \frac{p}{1 + (p-1)f}$$

S(p) speedup

ts total execution time

fts (time for serializable section)

(1-f)ts time for parallel sections

p number of processors

f time spent unparallelized

Amdahl's law is often used in parallel computing to predict the theoretical speedup when using multiple processors. Important is that the overall performance improvement is limited by its unparallelizable parts.

- Compute the theoretical speedup of a program that spends 10% of its time in unparallelizable, sequential regions for 6 cores and for a hypothetically unlimited number of cores.

$$S(6) = \frac{6}{1 + (6-1) * 0.1} = 4$$

$$S(\infty) = \frac{1}{0.1} = 10$$

- Compute the theoretical speedup of a program that spends 20% of its time in unparallelizable, sequential regions for 6 cores and for a hypothetically unlimited number of cores.

$$S(6) = \frac{6}{1 + (6 - 1) * 0.2} = 3$$

$$S(\infty) = \frac{1}{0.2} = 5$$

- Given an algorithm of time complexity O(n^3). How large (in %) can the unparallelizable, sequential region be at most, such that a speedup of 10 can be achieved using 64 cores?

$$10 = \frac{64}{1 + (64 - 1) * f} => f = 0.09 = 9\%$$