LUMI architecture

Rasmus Kronberg | Running GROMACS efficiently on LUMI workshop 2024

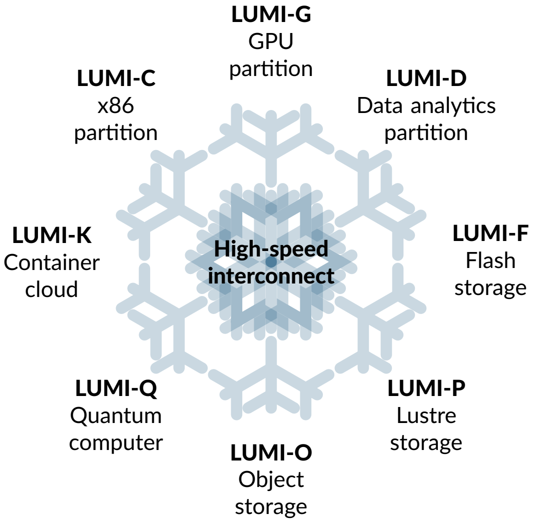
Motivation – Why do I need to know this?

I only want to run some program (GROMACS), why do I need to know about the system architecture?

1. A supercomputer like LUMI is **not** a large version of your personal laptop, but an expensive research instrument shared by hundreds of simultaneous users
2. Reserved resources (CPU, GPU, memory) cannot be accessed by others, so it is important to ensure those are utilized as efficiently as possible
3. Efficiency does not come automatically: besides application-specific details (problem size and algorithms), the job usually needs to be mapped properly on the hardware as well

LUMI is a large GPU supercomputer

- **LUMI-G:** 2978 GPU nodes with 4 AMD MI250X GPUs each
- **LUMI-C:** 2048 CPU nodes with 2 64-core AMD “Milan” CPUs each
- **LUMI-D:** Data analytics partition with large memory nodes and visualization GPUs (Nvidia A40)
- 118 PB storage space in total (**LUMI-P**, **LUMI-F**, **LUMI-O**)
- 4 login nodes and web interface
- HPE Cray Slingshot 11 interconnect



LUMI-G
GPU
partition

LUMI-C
x86
partition

LUMI-D
Data analytics
partition

LUMI-K
Container
cloud

High-speed interconnect

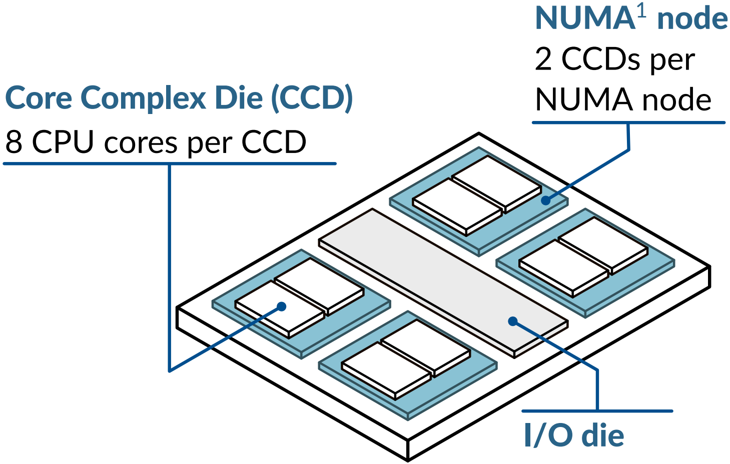
LUMI-F
Flash
storage

LUMI-Q
Quantum
computer

LUMI-P
Lustre
storage

LUMI-O
Object
storage

LUMI-C: The AMD EPYC “Milan” CPU



Core Complex Die (CCD)

8 CPU cores per CCD

NUMA[1] node

2 CCDs per NUMA node

I/O die

[1] NUMA = Non-uniform memory access

LUMI-C node

- A LUMI-C compute node has 2 sockets for connecting both CPUs
- Each node is linked to the 200 Gb/s Slingshot network via one of the sockets
- There's a strong hierarchy within a node:

Layer of hierarchy		per
1	2 threads	core
2	8 cores	CCD
3	2 CCDs	NUMA domain
4	4 NUMA domains	CPU (socket)
5	2 sockets	node

Data transfer delay increases

Distance from a core to memory affects performance

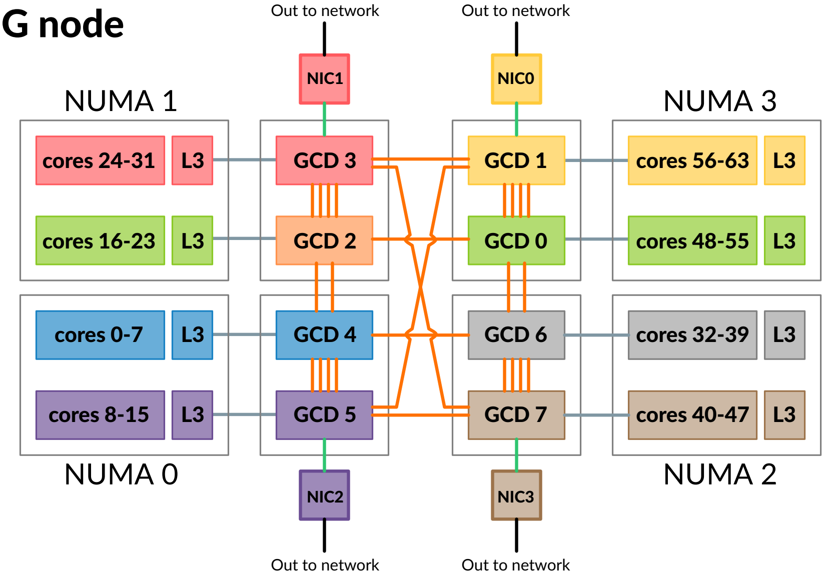
- Penalty for accessing memory in another NUMA domain in the same socket is minor (20%)
- …but accessing memory attached to another socket is a lot slower (320%)

		NUMA nodes CPU 1				NUMA nodes CPU 2			
		0	1	2	3	4	5	6	7
NUMA nodes CPU 1	0	10	12	12	12	32	32	32	32
	1	12	10	12	12	32	32	32	32
	2	12	12	10	12	32	32	32	32
	3	12	12	12	10	32	32	32	32
NUMA nodes CPU 2	4	32	32	32	32	10	12	12	12
	5	32	32	32	32	12	10	12	12
	6	32	32	32	32	12	12	10	12
	7	32	32	32	32	12	12	12	10

LUMI-G node

- Each LUMI-G node contains **4 AMD MI250X GPUs** and one **64-core AMD EPYC “Trento” CPU**
- The MI250X GPUs are *multi-chip modules* (MCM) with **2 graphics compute dies (GCDs)**
    - **Note!** From a software perspective, the GCDs act as individual GPUs, meaning that a *LUMI-G compute node can be considered to have 8 GPUs*
- The LUMI-G nodes have a very particular CPU–GPU linking
    - Binding a GCD to the right CCD is important for optimal performance
    - More on this later…

LUMI-G node


Out to network
Out to network
NIC1
NIC0
NUMA 1
NUMA 3
cores 24-31
L3
GCD 3
GCD 1
cores 56-63
L3
cores 16-23
L3
GCD 2
GCD 0
cores 48-55
L3
cores 0-7
L3
GCD 4
GCD 6
cores 32-39
L3
cores 8-15
L3
GCD 5
GCD 7
cores 40-47
L3
NUMA 0
NUMA 2
NIC2
NIC3
Out to network
Out to network

Take-home messages

Understanding the architecture is important for getting the best performance out of your application

- With LUMI-G, the most important point to remember is that each of the 8 GCDs in a node has a preferred CCD to work with
- Should be accounted for when mapping processes and threads on a GPU node

More details at:

- `docs.lumi-supercomputer.eu`
- `lumi-supercomputer.github.io/LUMI-training-materials/`

LUMI assembled