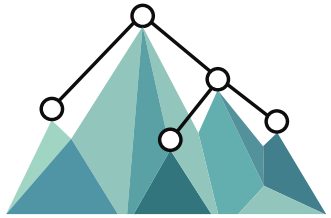
Sustainable Application Mapping for HPC Systems

Scalable Tools Workshop

28 July 2026



mpibind

Edgar A. León

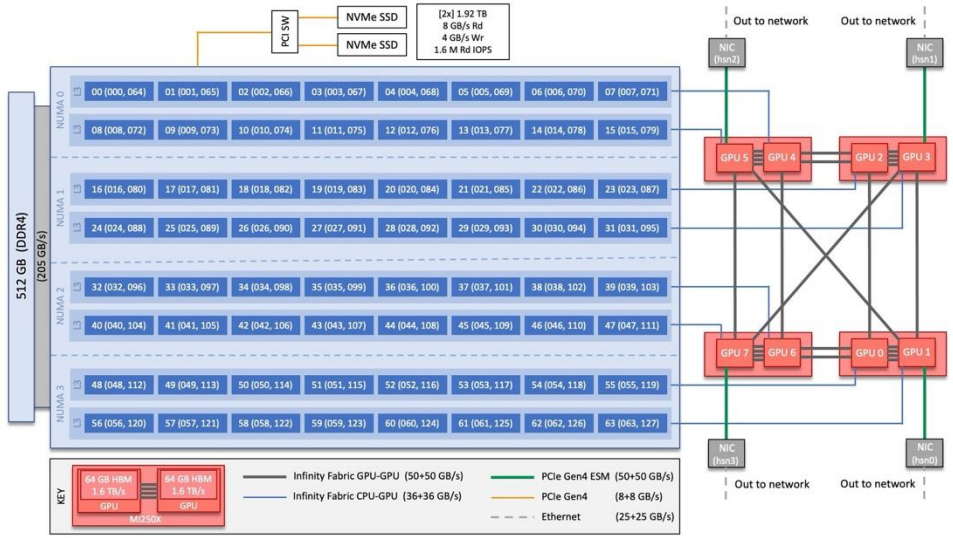
Livermore Computing

Prepared by LLNL under Contract DE-AC52-07NA27344.

Sustainable application performance and portability have become a central challenge

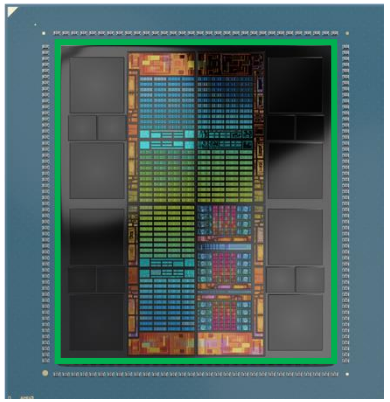
Sustainable application performance and portability have become a central challenge

AMD 3rd Gen EPYC CPU + AMD Instinct MI250X GPUs



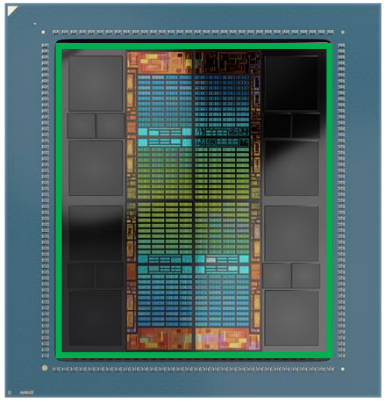
https://docs.olcf.ornl.gov/systems/crusher_quick_start_guide.html

AMD Instinct MI300A



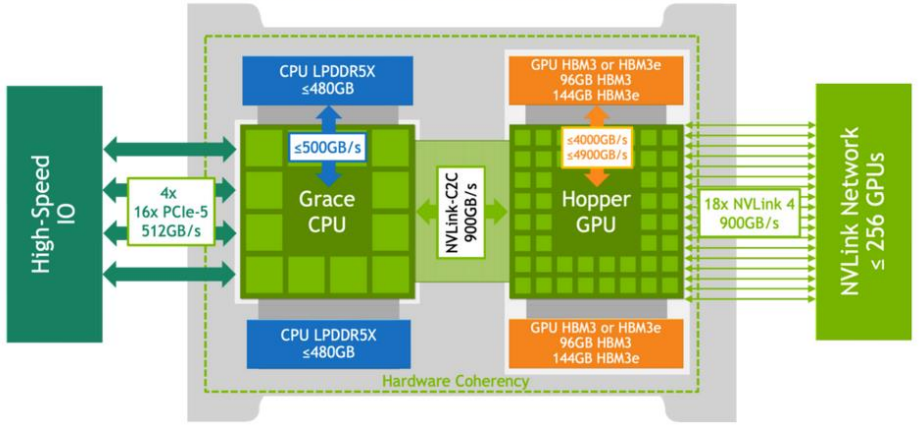
Source: AMD

AMD Instinct MI300X



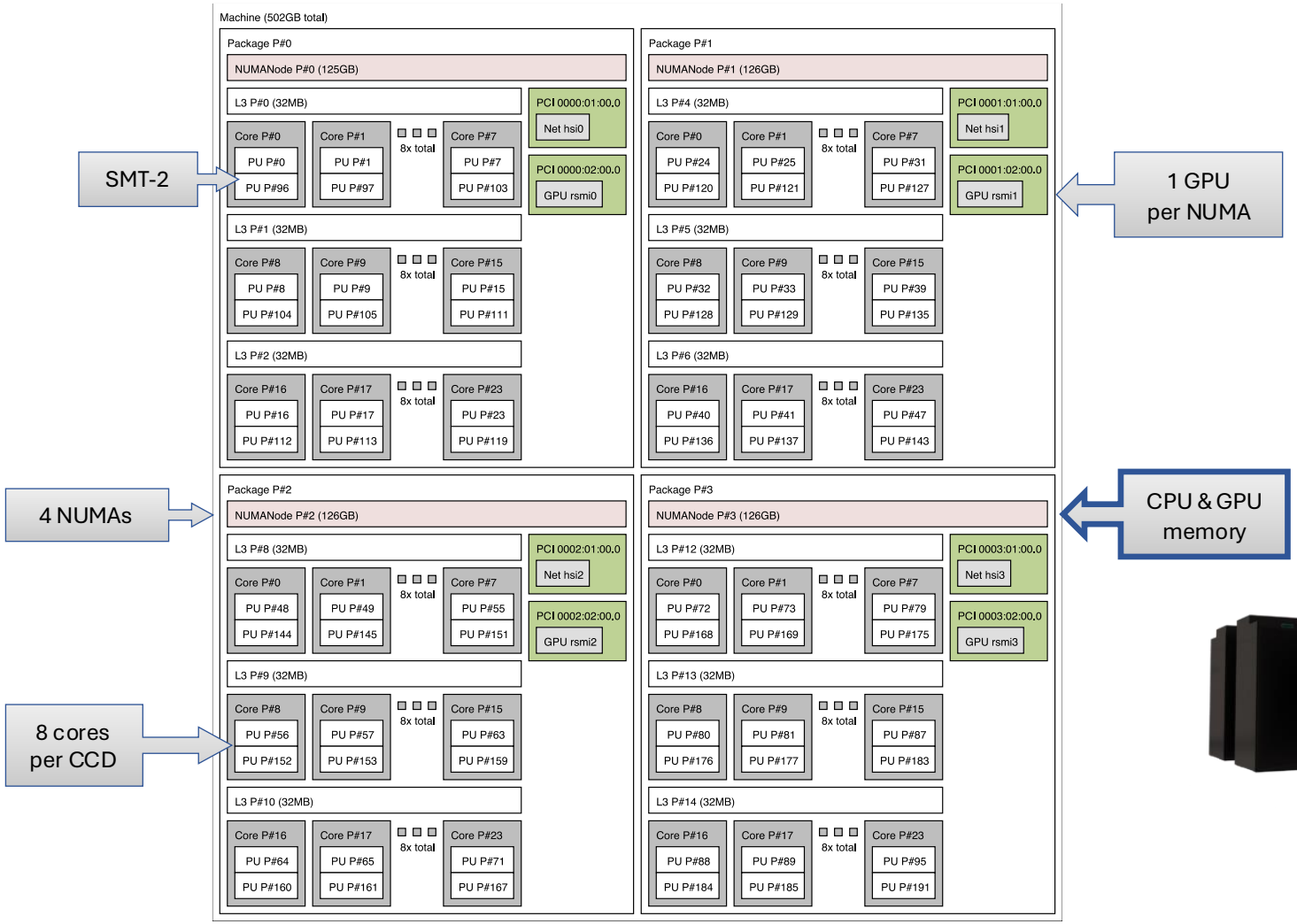
Source: AMD

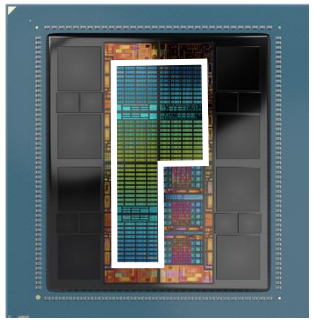
NVIDIA GH200 Grace Hopper Superchip



Source: NVIDIA

AMD MI300A: Heart of the 2nd fastest super in the world

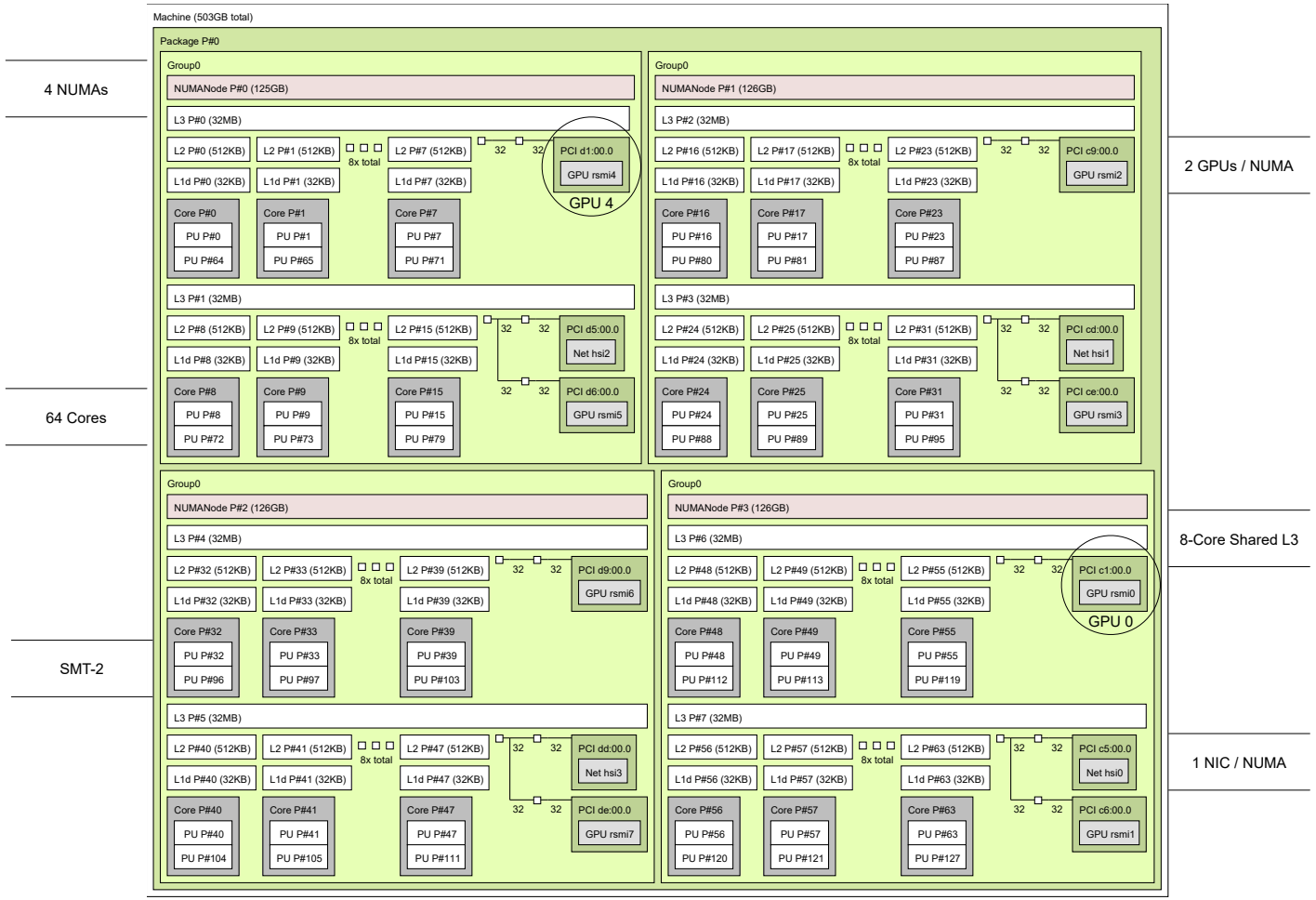




AMD Instinct MI300A APU

Each compute node has 4 MI300A APUs

Mapping applications to the machine is hard



Frontier's node architecture

What could go wrong?

- Multiple processes running on a single core
- Multiple GPU kernels sharing a GPU
- Processes launching kernels on non-local GPUs
- Threads accessing remote memory

Affinity using your favorite MPI library can be a nightmare

Intel MPI	I_MPI_PIN_DOMAIN	Core, sock, numa, node, cache
	I_MPI_PIN_ORDER	Range, scatter, compact, spread, bunch
MVAPICH2	MV2_CPU_BINDING_LEVEL	Core, socket, numanode
	MV2_CPU_BINDING_POLICY	Bunch, scatter, hybrid
	MV2_HYBRID_BINDING_POLICY	Bunch, scatter, linear, compact, spread
Open MPI	--bind-to, --rank-by	Slot, hwthread, core, cache, socket, numa, board
	--map-by	+ node, sequential, distance, ppr:n:unit:pe=n
IBM Spectrum MPI	-aff	Bandwidth, latency, cycle:unit

How do we make application mapping sustainable?

mpibind

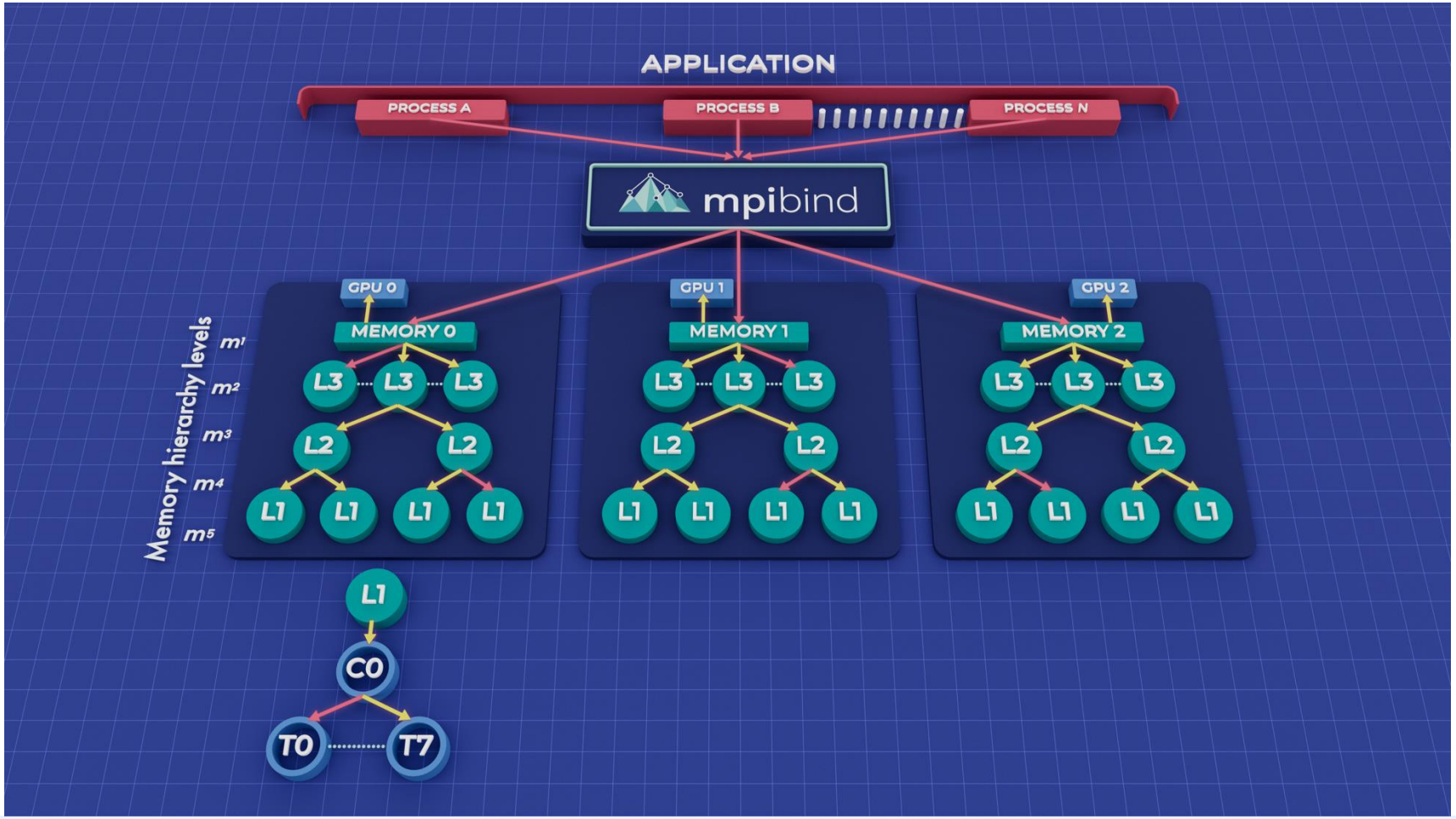
mpibind automatically maps applications to the hardware

- *Requires no code changes or machine-specific tuning*
- Contributes to software sustainability
  - Performance portability
  - Long-term maintainability
  - Productivity and accessibility

mpibind

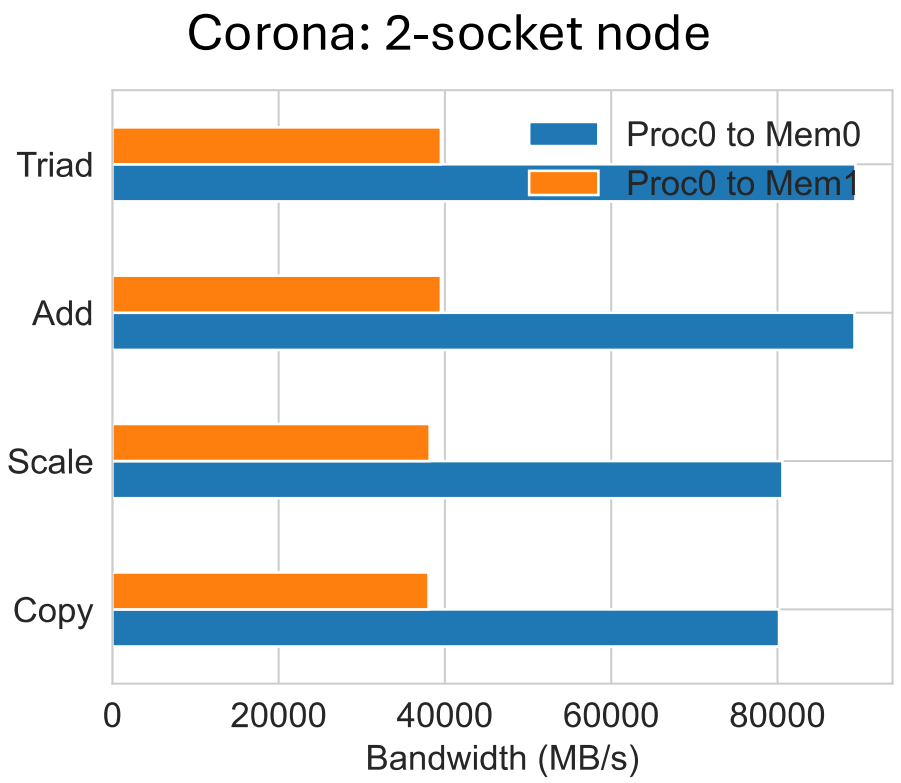
- Abstracts away hardware details
- Provides locality-aware placement
- Automatically computes mapping based on hardware topology
- Empowers broader community including those without architecture knowledge

mpibind calculates locality-aware placement based on runtime discovery of the system's topology



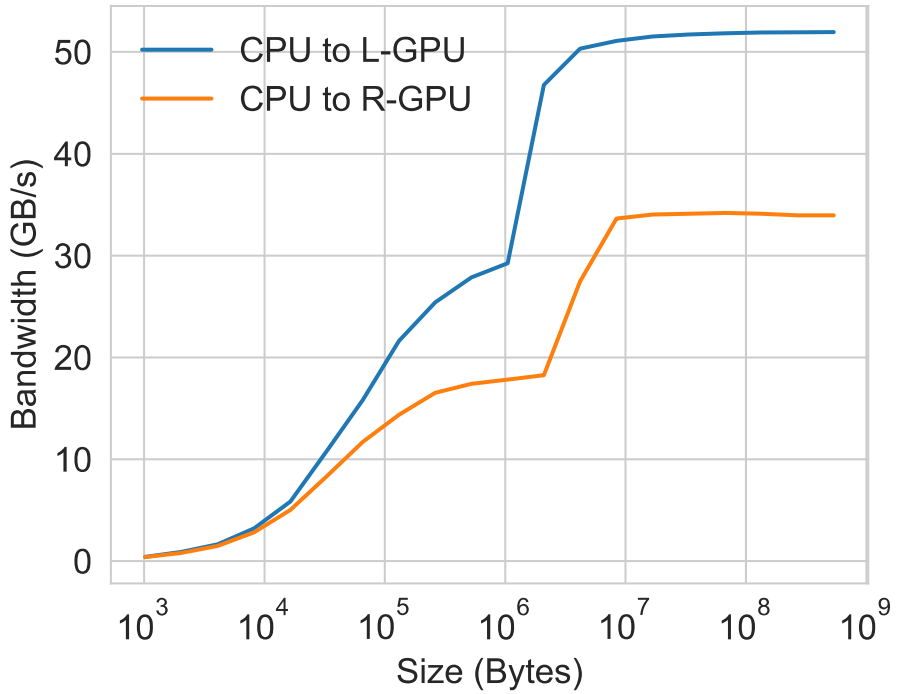
Locality matters: Local vs remote memory

- By default, mpibind maps each process to CPUs within a NUMA domain
- Leverage local memory
  - Spillover if necessary

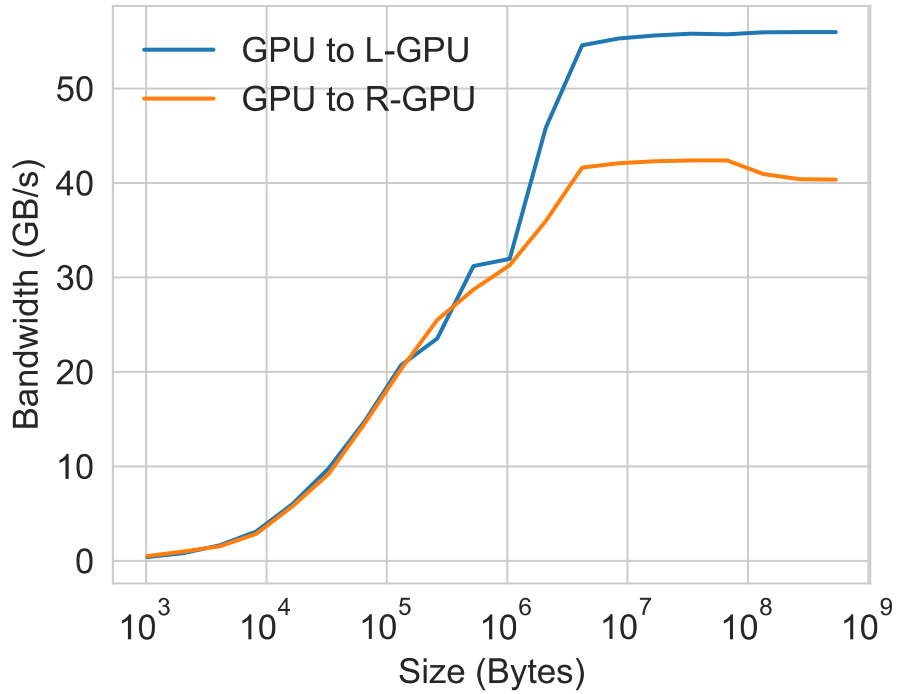


Locality matters: Local vs remote CPU-GPU pairs

Up to 35% penalty for remote CPU to GPU transfers



Up to 28% penalty for remote GPU to GPU transfers



mpibind mappings leverage local devices

mpibind mitigates complex launch configurations that degrade software maintainability

Launch 2 MPI processes per GPU using local resources

JSM

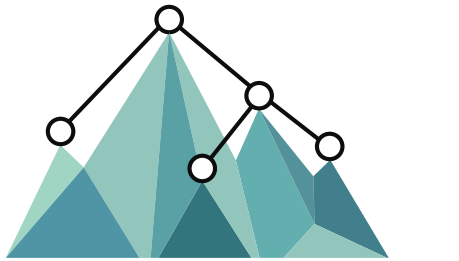
```
jsrun -a 2 -c 10 -g 1 -r 4 -d packed -b packed:5 <prog>
```

mpibind

```
srun -n8 <prog>
srun -n8 -mpibind=on <prog>
```

CORAL node
- 2 NUMAs
- 40 Cores
- 4 GPUs

Required	Number of tasks
Optional	Number of threads
	Greedy (true/false)
	GPU optimized (true/false)
	SMT (1—num. HW threads per core)
	Restricted topology (MEM/CPU)



mpibind

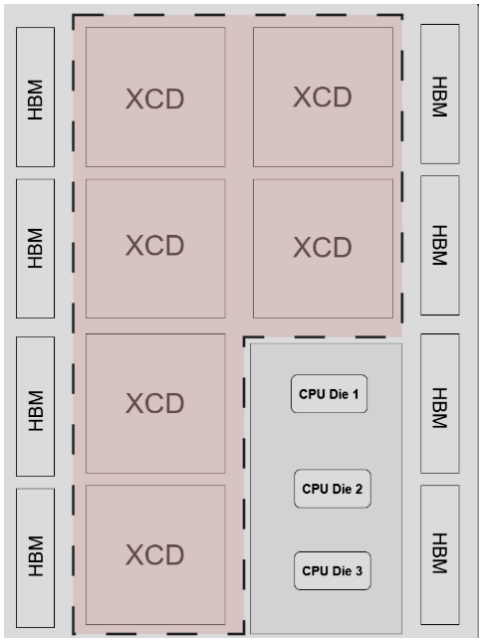
mpibind abstracts away hardware details while optimizing placement regardless of architecture

Launch 1 process per GPU

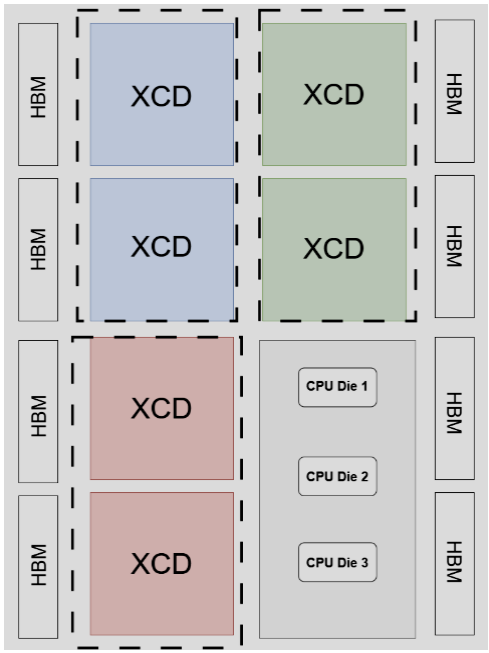
flux run –ntasks-per-node=**1***4 –amd-gpumode=**spx** <app>

flux run –ntasks-per-node=**3***4 –amd-gpumode=**tpx** <app>

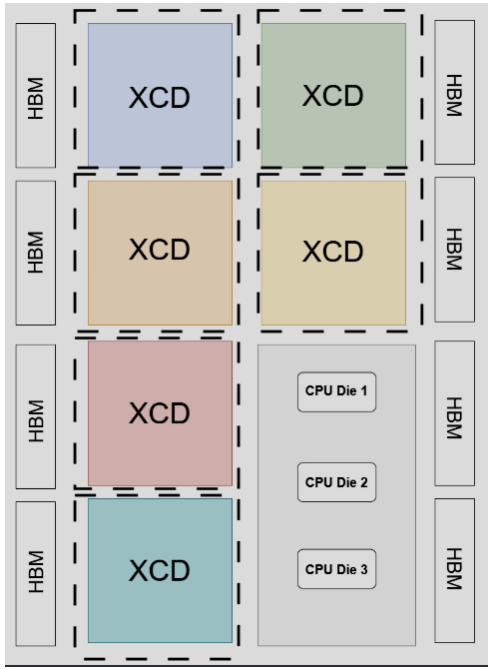
flux run –ntasks-per-node=**6***4 –amd-gpumode=**cpx** <app>



SPX: Single partition of 6 XCDs



TPX: Three partitions of 2 XCDs each



CPX: Six partitions of 1 XCD each

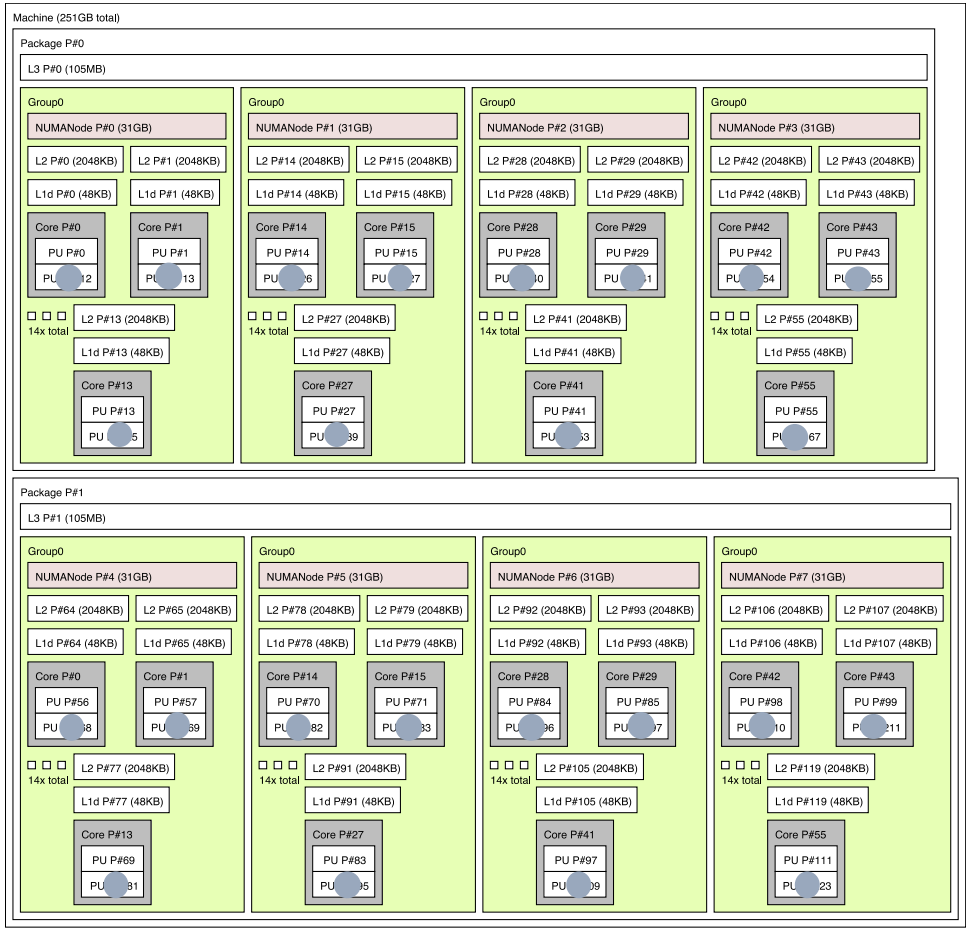
One mapping algorithm for all systems → Portability

- Mapping algorithm relies on abstract memory-compute tree
  - Portable hardware locality (hwloc)
- Mapping algorithm is separate from applying affinity

Mapping Algorithm	Resource Manger Plugin
{Process} → {CPUs, GPUs, Thread mapping}	Use job info as input to mpibind
	Get mpibind mapping
	Bind tasks and set environment variables

mpibind enables mitigation of system noise

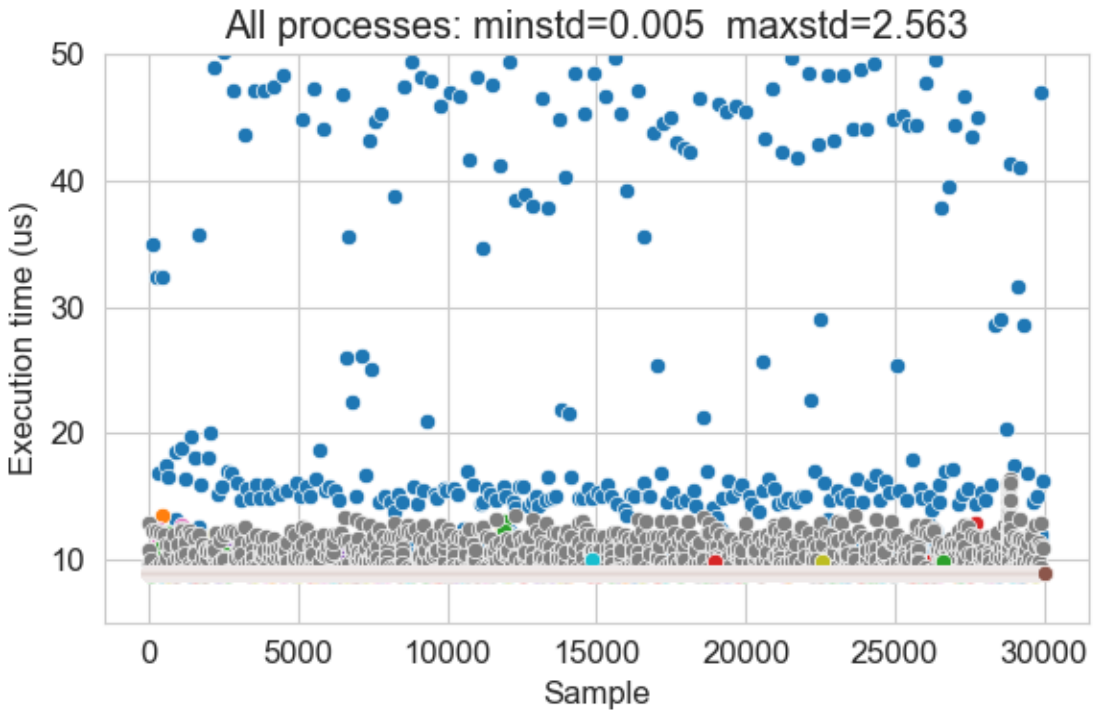
- CPU specialization
  - Application CPUs
  - System CPUs
- Noise mitigation
  - Place app threads on AppCPUs
  - Place system services on SysCPUs
- mpibind controls app placement
  - `MPIBIND_RESTRICT=<cpu|mem>`
  - Apps can still use all resources



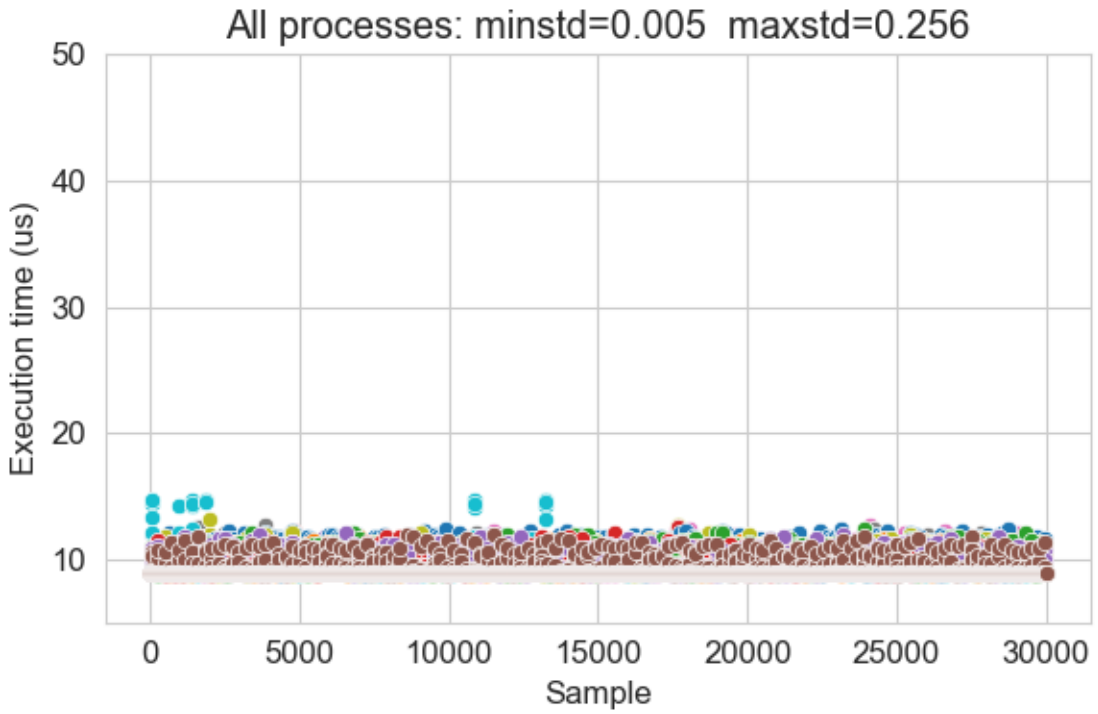
`MPIBIND_RESTRICT=112-223`

mpibind enables system noise mitigation studies

Keeping CPU zero out of Application CPUs pays off



AppCPUs: Low-numbered HT of every core



AppCPUs: High-numbered HT of every core

León et al. 2025. Breaking the System Noise Barrier at Exascale. SC25. https://doi.org/10.1145/3712285.3759793

mpibind is used in production across systems ranging from clusters to the world's fastest supercomputer

- Open source
  - MIT license
  - Written in C
  - Depends on hwloc

- Using mpibind
  - Slurm SPANK plugin

    $ grep mpibind /etc/slurm/plugstack.conf
    required /usr/lib64/mpibind/mpibind.so default_off

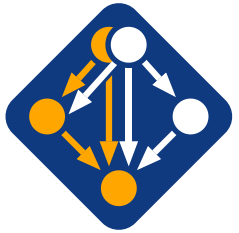
  - Flux plugin

- Building mpibind
  - GNU autotools

    bootstrap
    configure
    make
    make install

  - Spack

    spack install mpibind+rocm
    spack install mpibind+cuda



mpibind has limitations

- Not suited for applications with dynamically changing mappings
  - Static policy set at job start
- Not intended for benchmarking
  - Performance of remote memory, remote GPUs, etc.
- Not replacing custom mappings
  - Resource manager's affinity masks, etc.

Advanced use cases will be covered by Quo Vadis
https://github.com/hpc/quo-vadis

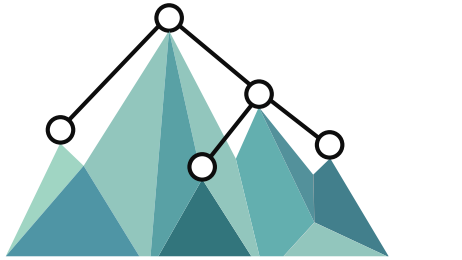
Documentation is available

https://github.com/LLNL/mpibind

SC 2025	Breaking the system noise barrier at Exascale
SC 2020	TOSS-2020: A commodity software stack for HPC
MEMSYS 2018	Achieving transparency mapping parallel applications: A memory hierarchy affair
MEMSYS 2017	mpibind: A memory-centric affinity algorithm
IPDPS 2016	System noise revisited: Enabling application scalability and reproducibility with SMT

mpibind contributes to software sustainability

- Performance portability
  - Abstracts away hardware details
  - Provides locality-aware mappings
- Long-term maintainability
  - Provides automatic and deterministic mappings
  - Software workflows become easier to maintain, share, and reproduce
- Productivity and accessibility
  - Eliminates the need for manual mapping and architecture knowledge
  - Empowers a broader community of scientists to focus on their science



mpibind