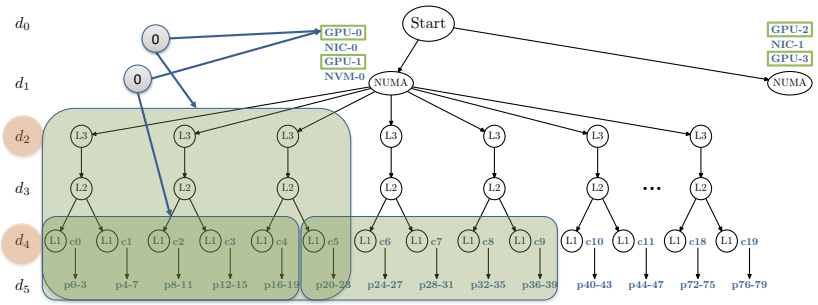
Mapping Applications to the Hardware Portably and Transparently

Edgar A. León
Livermore Computing

mpibind

FOSDEM, Brussels
2 February 2025



Prepared by LLNL under contract DE-AC52-07NA27344. LLNL-PRES-872166.

Lawrence Livermore National Laboratory

HPC users face complex computing architectures

Modern Systems	Growing Gap	Application Needs
Manycore	MPI OpenMP, Pthreads US DOE RAJA and Kokkos CUDA, HIP, OpenMP 5+, SYCL	Portability
Heterogeneous		Productivity
Multi-type Memory		Performance

AMD Instinct MI300A

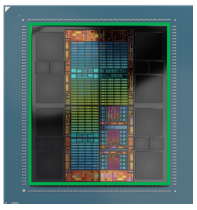


Image credit: AMD

AMD Instinct MI300X

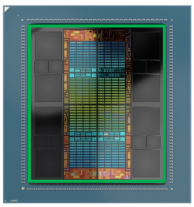
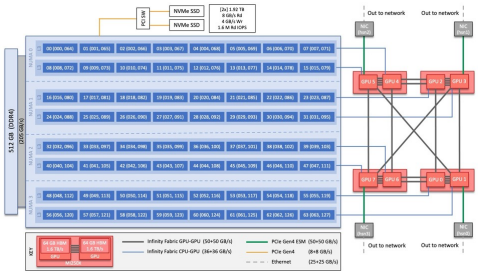


Image credit: AMD

AMD 3rd Gen EPYC CPU + AMD Instinct MI250X GPUs



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

NVIDIA GH200 Grace Hopper Superchip

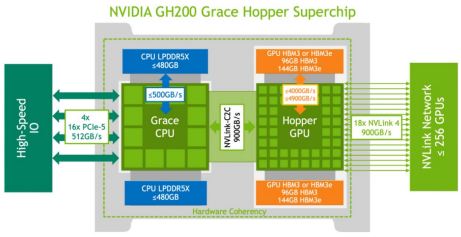
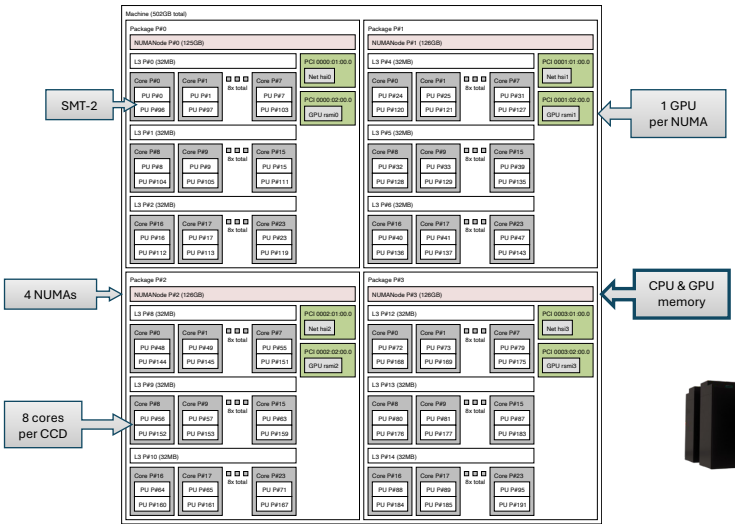
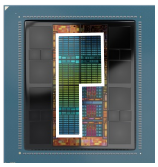


Image credit: NVIDIA

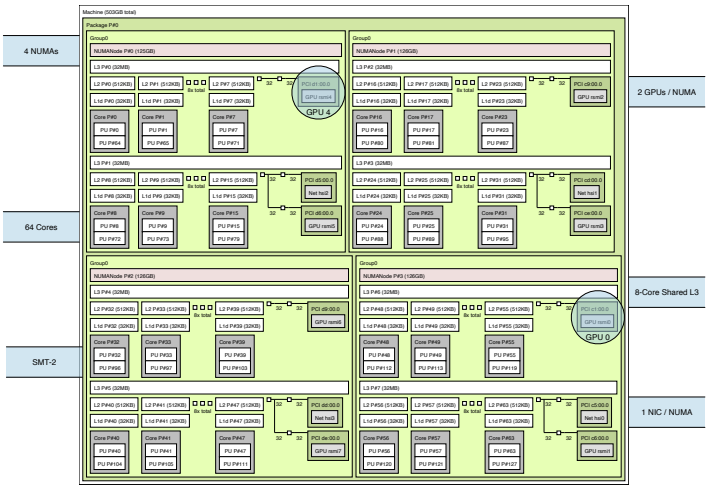
AMD MI300A: Heart of the top super in the world





AMD Instinct MI300A APU

Mapping applications to the machine is hard



- What could go wrong?
  - Multiple threads running on a single core
  - Multiple GPU kernels sharing a GPU
  - Tasks launching kernels on non-local GPUs
  - Threads accessing remote memory
  - Etc.

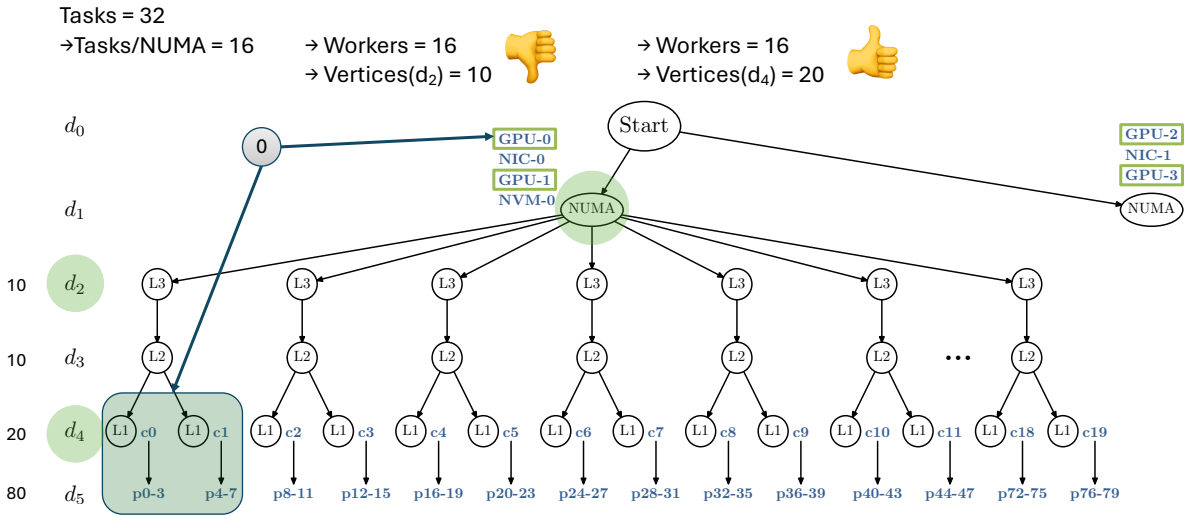
mpibind is a memory-first, user-friendly algorithm

- Design principles
  - Memory-system driven
  - Locality based
  - Portable across architectures

mpibind

- Properties
  1. Provides a simple interface
  2. Requires minimal user input
  3. Provides cross-system portability
  4. Minimizes rem. memory accesses
  5. Maximizes cache per worker
  6. Leverages compute/mem locality
  7. Enables system-noise mitigation
  8. Enables job co-scheduling

Follow the memory system and leverage locality



1. Provides a simple interface &
2. Requires minimal user input

Coral Node
2 NUMAs,
40 cores, 4 GPUs

JSM

```
# 8 MPI tasks across all cores and all GPUs on a CORAL node
jsrun -a 2 -c 10 -g 1 -r 4 -d packed -b packed:5 <prog>
```

mpibind

```
# 8 MPI tasks across all cores and all GPUs on a CORAL node
srun -n8 <prog>
srun -n8 -mpibind=on <prog>
```

Required	Number of tasks
Optional	Number of threads
	Greedy (true or false)
	GPU optimized (true or false)
	SMT (1 to num. HW threads per core)
	Restricted topology (Mem or CPU)

mpibind

3. Provides portability across systems

- Counterexample

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
OpenMPI	--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

8

3. Provides portability across systems

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

C Interface	Slurm and Flux Plugins
{Task} → {CPUs, GPUs, Thread mapping}	Use job info as input to mpibind
	Get mpibind mapping
	Bind tasks
	Set environment variables

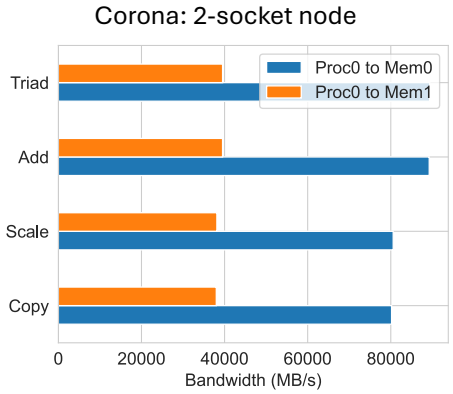
mpibind

slurm workload manager

flux

4. Minimizes remote memory accesses

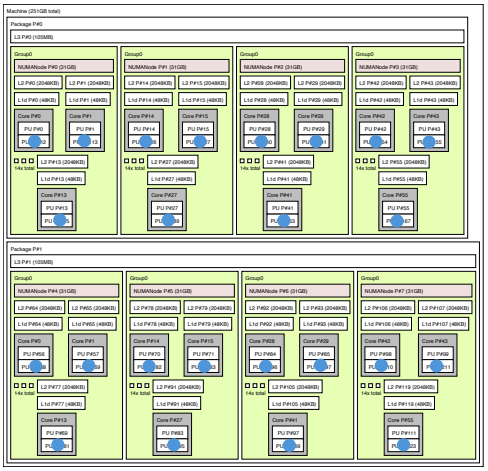
- Each task restricted to CPUs within a NUMA domain
- Leverage local memory
  - Spillover if necessary



10

7. Enables system noise mitigation

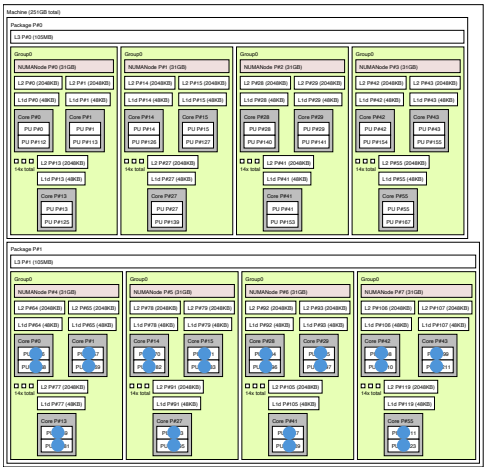
- CPU specialization
  - Application CPUs
  - System CPUs
- Noise mitigation
  - Place app threads on AppCPUs
  - Place system services on SysCPUs
- mpibind controls app placement
  - MPIBIND_RESTRICT=<cpu|mem>
  - **Apps can still use all resources**



MPIBIND_RESTRICT=112-223

7. Enables system noise mitigation

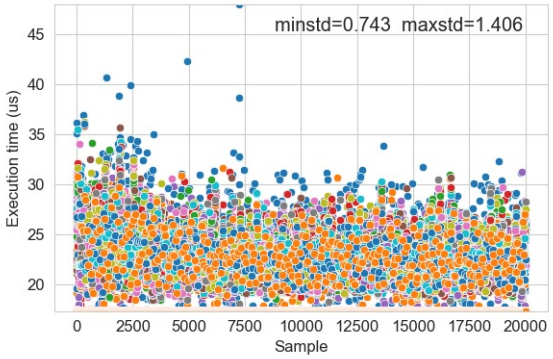
- CPU specialization
  - Application CPUs
  - System CPUs
- Noise mitigation
  - Place app threads on AppCPUs
  - Place system services on SysCPUs
- mpibind controls app placement
  - MPIBIND_RESTRICT=<cpu|mem>
  - **Apps can still use all resources**



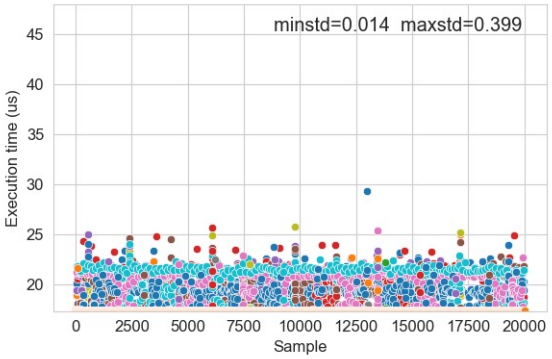
MPIBIND_RESTRICT_TYPE=mem
MPIBIND_RESTRICT=4-7

7. Enables system noise mitigation studies

AppCPUs: Top hardware threads



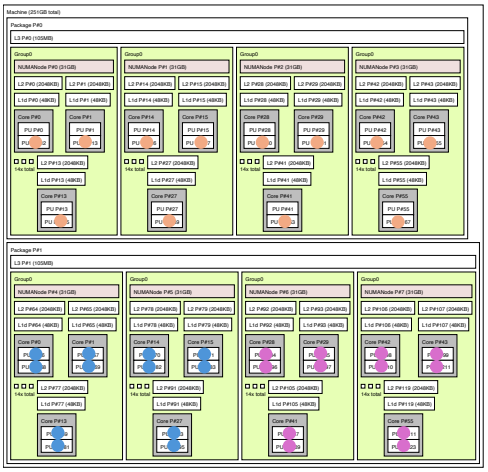
AppCPUs: Bottom hardware threads*



*Additional kernel settings applied

8. Enables arbitrary placement of co-located, concurrent jobs

- Job 1 on Mem 4-5
  - `MPIBIND_RESTRICT_TYPE=mem`
  - `MPIBIND_RESTRICT=4-5`
  - `srun -n4 -overlap -mpibind=on prog1 &`
- Job 2 on Mem 6-7
  - `MPIBIND_RESTRICT_TYPE=mem`
  - `MPIBIND_RESTRICT=6-7`
  - `srun -n14 -overlap -mpibind=on prog2 &`
- Job 3 on bottom CPUs of Mem 0-3
  - `MPIBIND_RESTRICT_TYPE=cpu`
  - `MPIBIND_RESTRICT=112-167`
  - `srun -n4 -overlap -mpibind=on prog3 &`



LLNL using mpibind in production since 2015

- Open source
  - MIT license
  - Written in C
  - Depends on hwloc
- Slurm SPANK plugin

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

- Building mpibind
  - GNU autotools

    bootstrap
    configure
    make
    make install

  - Spack

    spack install mpibind+rocm
    spack install mpibind+cuda

https://github.com/LLNL/mpibind

18

mpibind is an excellent initial policy, but…

- Not suited for apps with dynamically changing mappings
  - Static policy set at job start
- Not intended for benchmarking
  - Performance of remote memory, remote GPUs, etc.
- Does not replace 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

- Tutorials
  - Flux affinity, Slurm affinity, mpibind
    https://github.com/LLNL/mpibind/tree/master/tutorials
- Articles

SC 2020	TOSS-2020: A commodity software stack for HPC
MEMSYS 2018	Achieving transparency mapping parallel applications: A memory hierarchy affair
GTC 2018	Mapping MPI+X applications to multi-GPU architectures: A performance-portable approach
MEMSYS 2017	Mpibind: A memory-centric affinity algorithm for hybrid applications
IPDPS 2016	System Noise Revisited: Enabling Application Scalability and Reproducibility with SMT

mpibind helps users make better use of modern supercomputers

- Performance
  - Leverages local memory and local devices
  - Helps mitigate system noise
  - Maximizes cache per worker
- Productivity
  - Provides a simple interface
  - Requires minimal user input
- Portability and vendor-neutrality
  - Same algorithm across system architectures, MPI libraries, and resource managers

mpibind