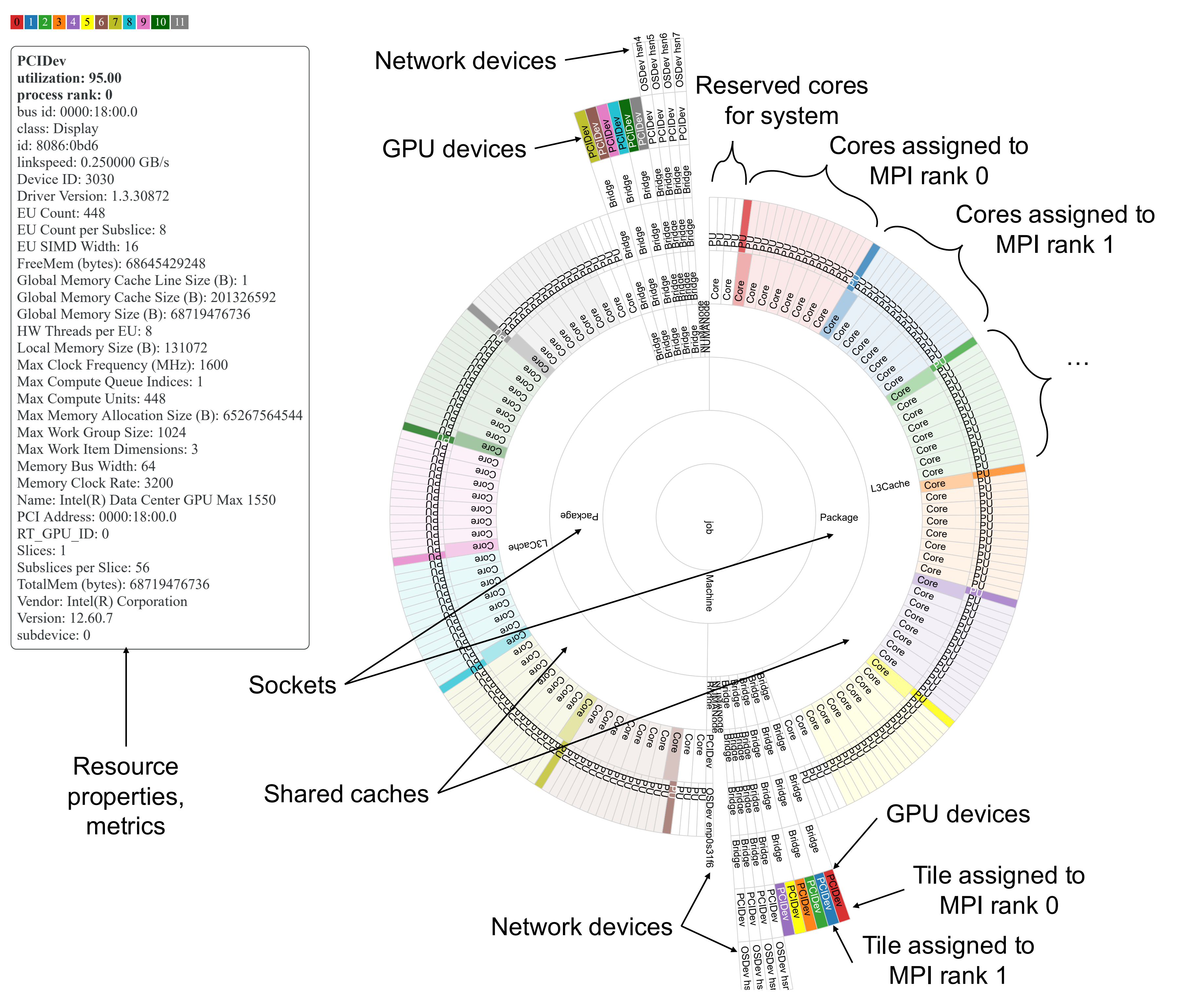
ZeroSum: User Space Utility for Monitoring Hardware and Software Resources for HPC

Kevin Huck[1], Allen D. Malony – University of Oregon [1]*now at AMD*

UNIVERSITY OF OREGON

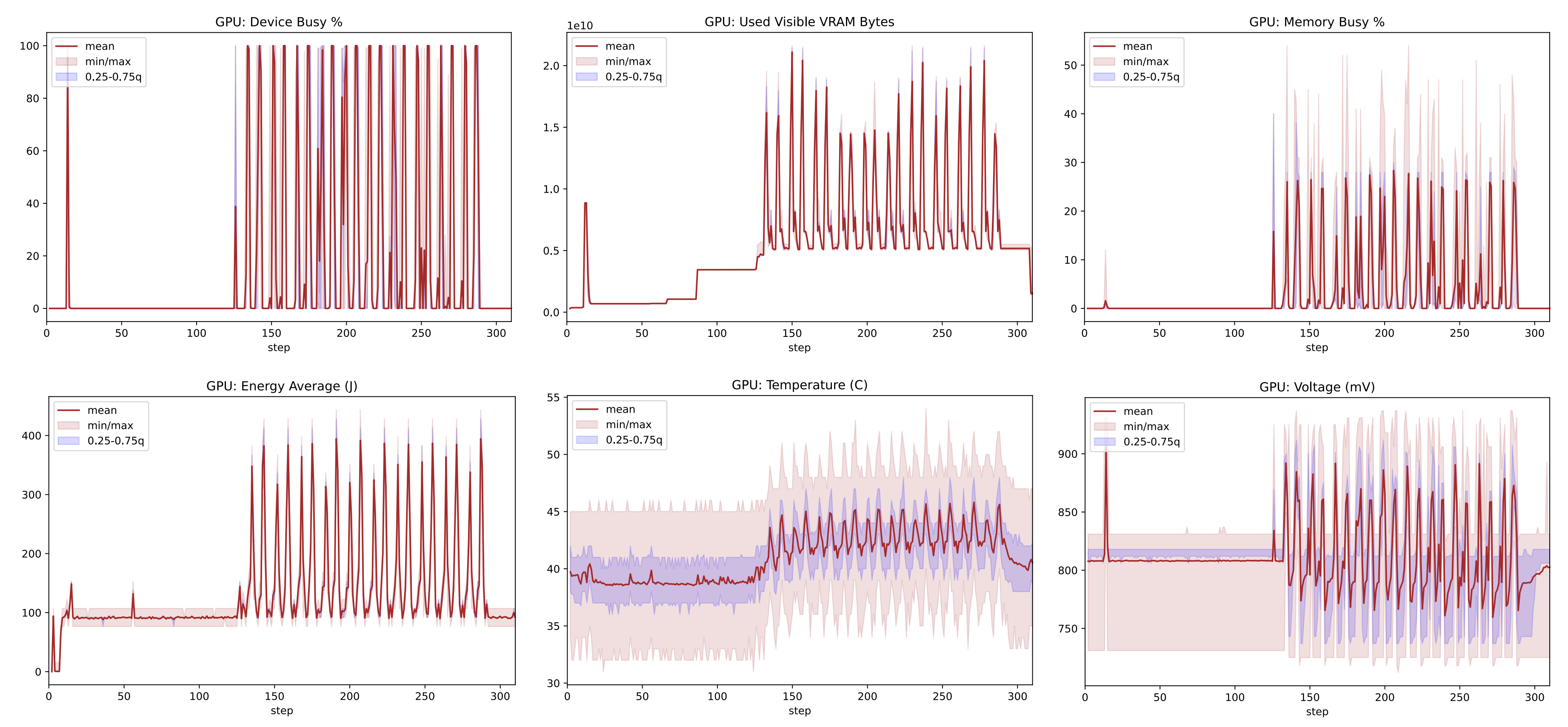

0 1 2 3 4 5 6 7 8 9 10 11
Network devices
GPU devices
Reserved cores for system
Cores assigned to MPI rank 0
Cores assigned to MPI rank 1
…
Sockets
Resource properties, metrics
Shared caches
GPU devices
Tile assigned to MPI rank 0
Tile assigned to MPI rank 1
Network devices
Package
L3Cache
Core
Bridge
PCIDev
OSDev
Machine
job

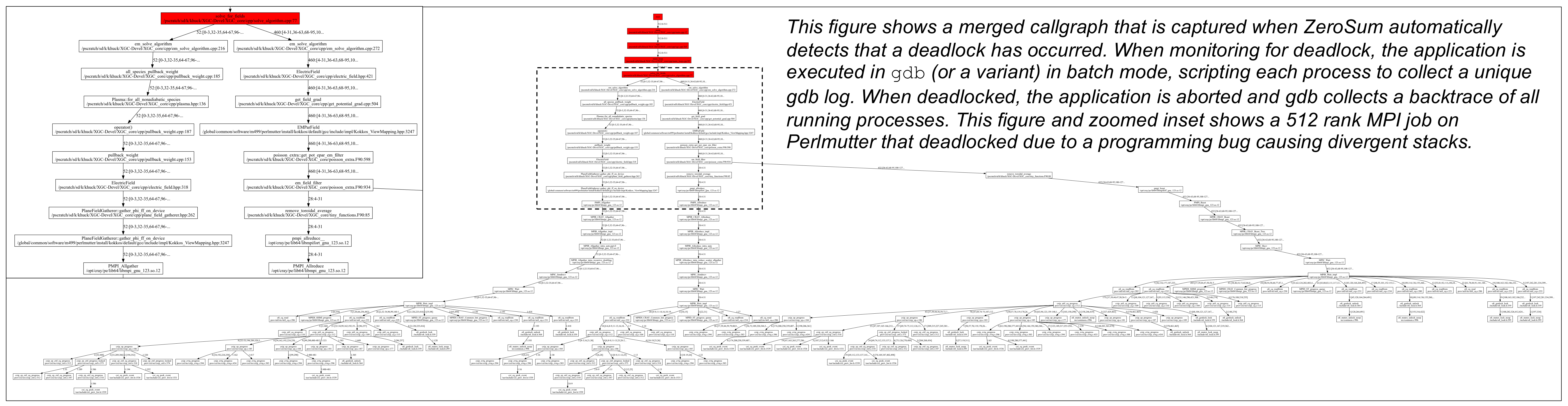
PCIDev
utilization: 95.00
process rank: 0
bus id: 0000:18:00.0
class: Display
id: 8086:0bd6
linkspeed: 0.250000 GB/s
Device ID: 3030
Driver Version: 1.3.30872
EU Count: 448
EU Count per Subslice: 8
EU SIMD Width: 16
FreeMem (bytes): 68645429248
Global Memory Cache Line Size (B): 1
Global Memory Cache Size (B): 201326592
Global Memory Size (B): 68719476736
HW Threads per EU: 8
Local Memory Size (B): 131072
Max Clock Frequency (MHz): 1600
Max Compute Queue Indices: 1
Max Compute Units: 448
Max Memory Allocation Size (B): 65267564544
Max Work Group Size: 1024
Max Work Item Dimensions: 3
Memory Bus Width: 64
Memory Clock Rate: 3200
Name: Intel(R) Data Center GPU Max 1550
PCI Address: 0000:18:00.0
RT_GPU_ID: 0
Slices: 1
Subslices per Slice: 56
TotalMem (bytes): 68719476736
Vendor: Intel(R) Corporation
Version: 12.60.7
subdevice: 0

Above: ZeroSum uses the hwloc library to capture the hierarchical topology of the nodes in the allocation and all available properties for each resource as a JSON tree. The tree is post-processed with Python scripts and merged with telemetry data from each process and compute node monitored by ZeroSum. The data is mapped to the resource topology and rendered as an interactive sunburst plot and presented to the user as a single HTML page. Users can quickly visualize which resources were mapped to each process executing on each node.

High Performance Computing (HPC) systems are large, heterogeneous, sophisticated – and are therefore so *complicated* that they are difficult to use efficiently. HPC users are allocated finite compute time on systems and yet have no portable utility to confirm that they are *effectively utilizing* the allocation at their disposal.
ZeroSum is a user space library that is launched within the process space of the HPC application. For each application process, it will monitor the application threads, MPI communication, and the hardware resources assigned to them – including CPU cores and/or hardware threads, memory usage and GPU utilization. Supported systems include all Linux operating systems, as well as GPUs from NVIDIA (using the NVML library), AMD (using the ROCm-SMI library) and Intel (using the SYCL API). Host side monitoring utilizes the virtual `/proc` filesystem and therefore is portable to all Linux systems. When integrated with the hwloc library, visualizations of utilization data can be generated from included Python post-processing scripts. *Automatic deadlock detection* is available, and ZeroSum will generate call stacks from all ranks, merge them, and visualize the resulting merged call stacks to help diagnose where expected behavior diverged (similar to STAT/Cray-STAT). Monitoring overhead is less than 0.5%.
Future Work: Plans for enhancements include monitoring of network and filesystem devices, automated statistical analysis of monitoring data, and in situ monitoring support through asynchronous aggregation services.

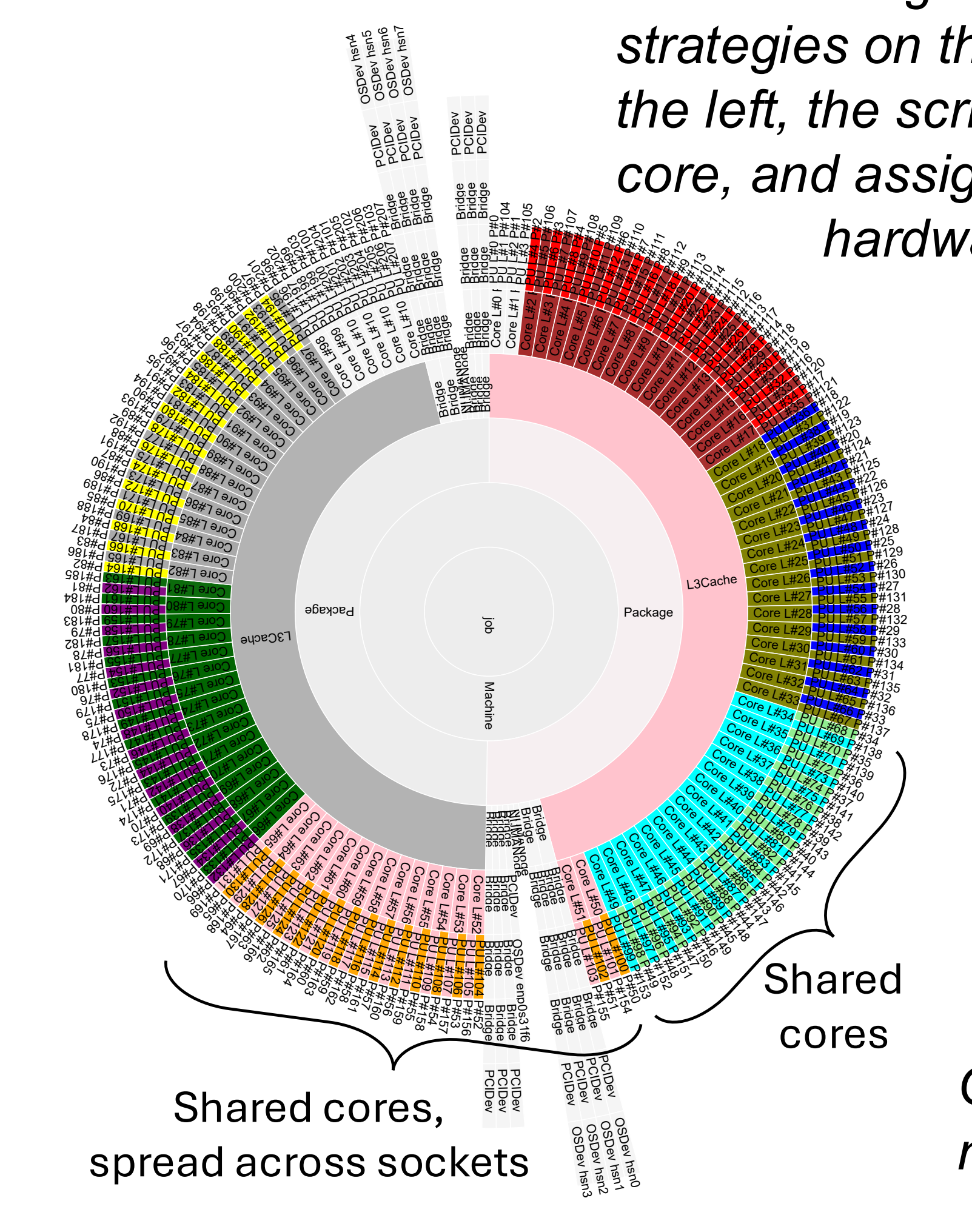
Below: Telemetry data is captured as time series data and can be post-processed to visualize changes over time – and visualize variability between threads, ranks or nodes. These sparklines show a collection of GPU monitoring data from a 64 rank MPI job utilizing one AMD MI250X GCD per rank.



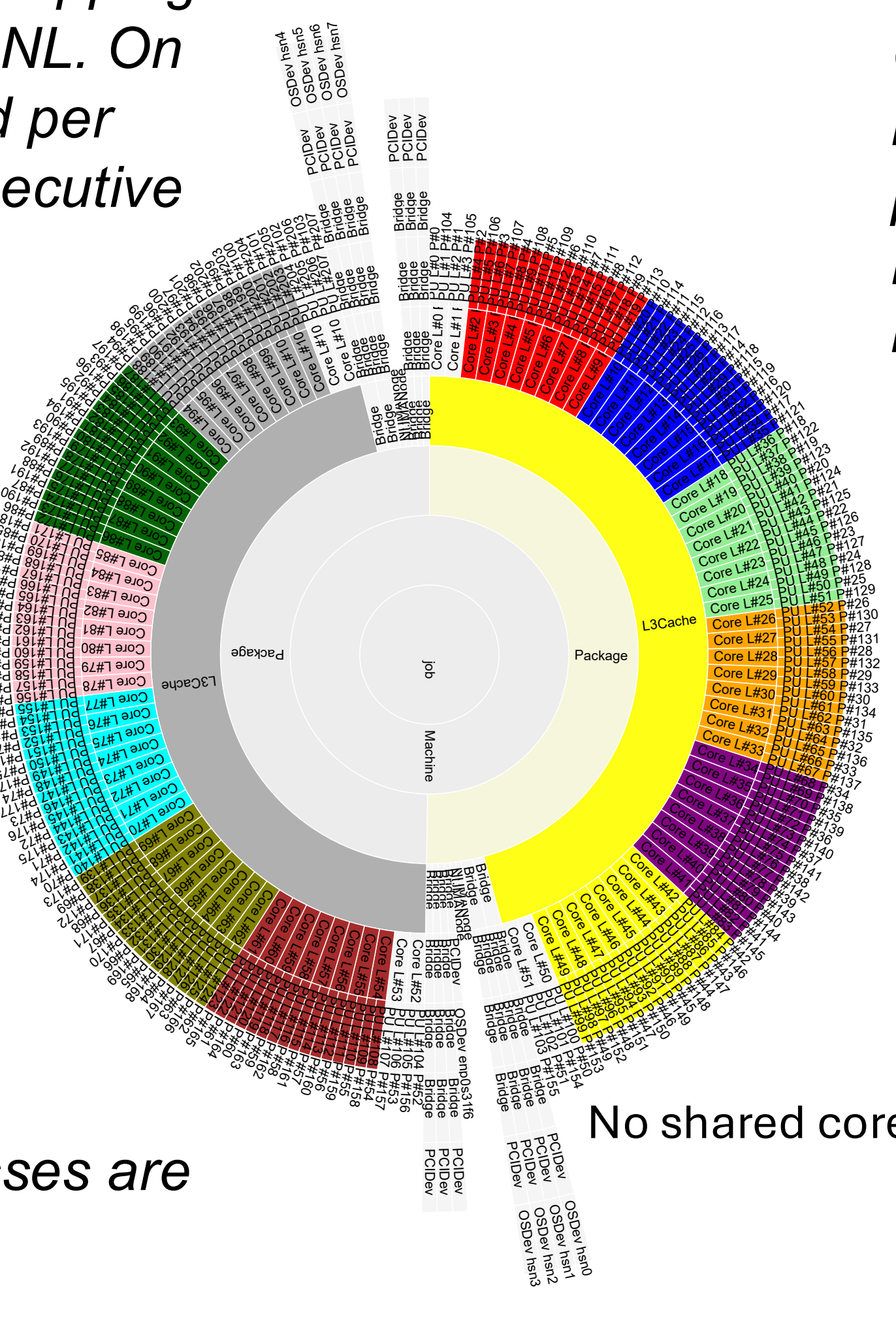


This figure shows a merged callgraph that is captured when ZeroSum automatically detects that a deadlock has occurred. When monitoring for deadlock, the application is executed in `gdb` (or a variant) in batch mode, scripting each process to collect a unique gdb log. When deadlocked, the application is aborted and gdb collects a backtrace of all running processes. This figure and zoomed inset shows a 512 rank MPI job on Perlmutter that deadlocked due to a programming bug causing divergent stacks.

These two figures show two process mapping strategies on the Sunspot system at ANL. On the left, the script assumed one thread per core, and assigned processes to consecutive hardware thread IDs.

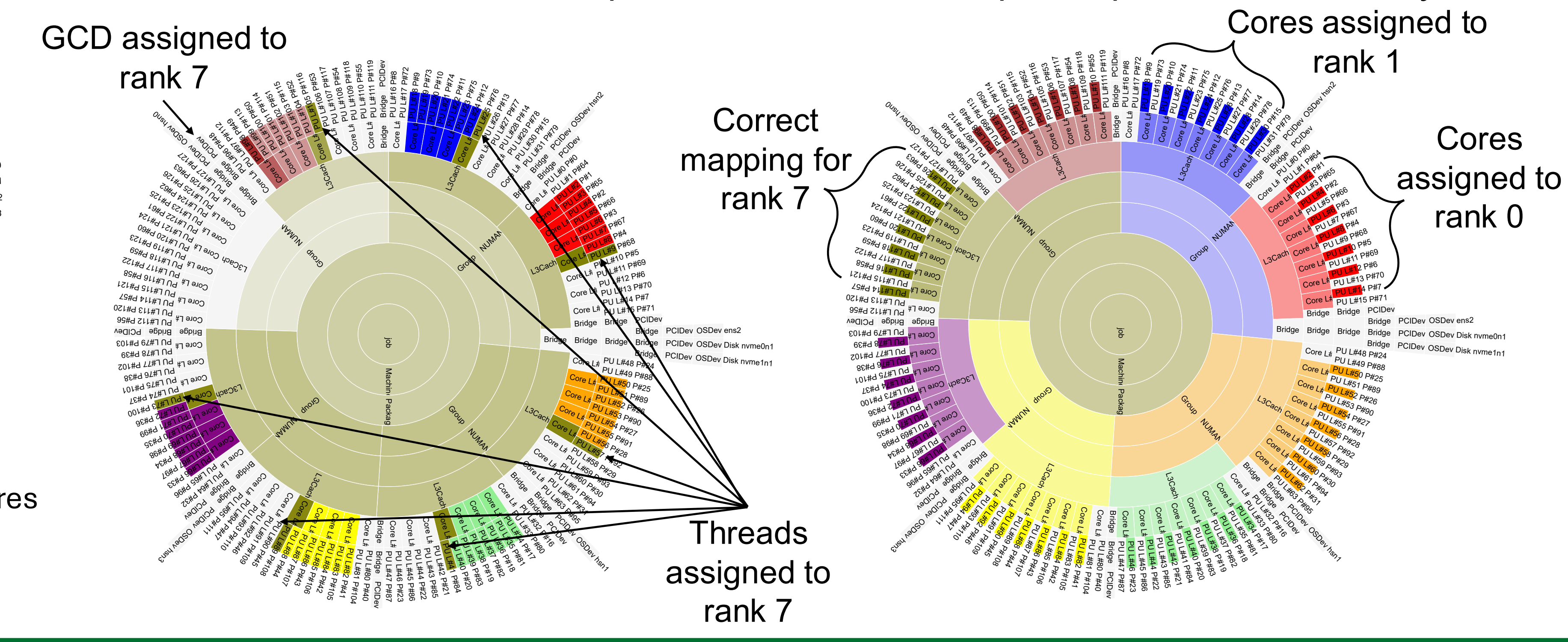


This caused cores to be shared between processes. In addition, two processes were spread across sockets, and some threads were not in the same L3 as their assigned GPU tiles.



On the right, the processes are mapped correctly.

These two figures show a possible slurm misconfiguration on the Frontier system at ORNL. On the left, eight processes of seven threads each are mapped to the eight L3 cache regions of the node using two hardware threads per core, and one AMD MI250X GCD per process. Unfortunately, one of the processes is spread across seven of the eight cache regions, none of which are physically located with its GCD. On the right, the application is launched with one hardware thread per core, and slurm maps the processes correctly.



This work was supported by the Scientific Discovery through Advanced Computing (SciDAC) program funded by U.S. Department of Energy, Office of Science, Advanced Scientific Computing Research (ASCR) under contract DE-SC0021299. This research was also supported by the Exascale Computing Project (17-SC-20-SC), a collaborative effort of the U.S. Department of Energy Office of Science and the National Nuclear Security Administration. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. This research used resources of the Argonne Leadership Computing Facility, a U.S. Department of Energy (DOE) Office of Science user facility at Argonne National Laboratory and is based on research supported by the U.S. DOE Office of Science-Advanced Scientific Computing Research Program, under Contract No. DE-AC02-06CH11357. This research used resources of the National Energy Research Scientific Computing Center (NERSC), a Department of Energy Office of Science User Facility using NERSC award FES-ERCAPm499.

Argonne NATIONAL LABORATORY
FRONTIER
OAK RIDGE National Laboratory | LEADERSHIP COMPUTING FACILITY
NERSC
Perlmutter
U.S. DEPARTMENT OF ENERGY | Office of Science