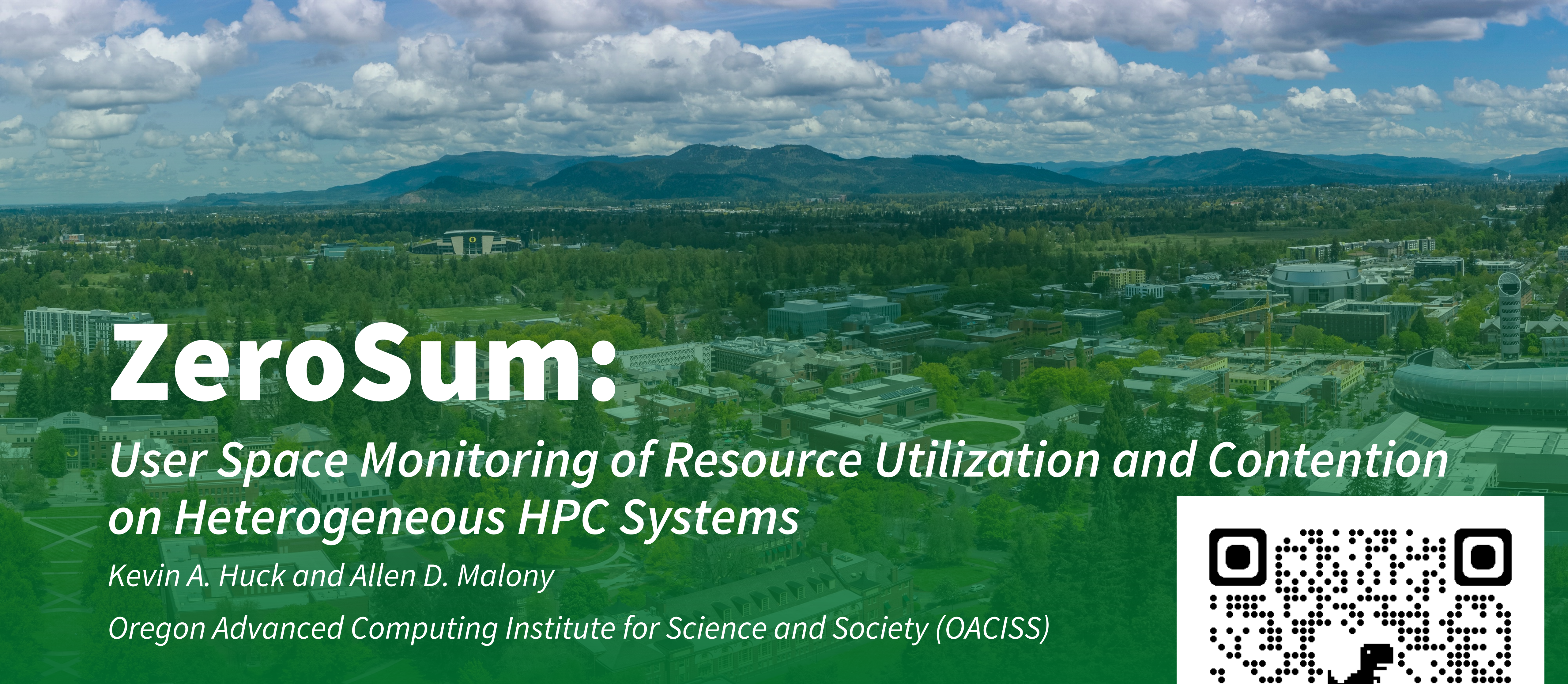


ZeroSum:

User Space Monitoring of Resource Utilization and Contention on Heterogeneous HPC Systems

Kevin A. Huck and Allen D. Malony

Oregon Advanced Computing Institute for Science and Society (OACISS)

UNIVERSITY OF OREGON

https://tinyurl.com/hust23-zerosum

Performance Analysis Perspective on Monitoring

3 classes of performance optimizations:

1. Algorithmic replacement (usually a high level of difficulty)
   - E.g. replace $O(n^2)$ with $O(n\log n)$
   - Can involve data structure changes, new dependencies, major rewrites
2. Code optimization (usually a medium difficulty)
   - Improve cache reuse, reduce stalls (branching, instructions, I/O, etc)
3. Optimized launch configuration (low difficulty)
   - Misconfiguration
   - Wrong assumptions from another system
   - Changes to system policies/defaults (e.g. reserved cores)
   - Low hanging fruit – but no purpose-built tool to assist (some ad-hoc solutions)
   - Need for a user space monitoring solution

Motivation: Why do users monitor at runtime?

- Sanity check / curiosity / impatience
- **Check for misconfiguration**
- **Check for efficient utilization**
- Confirmation of expected hardware / operating system behavior
- Identify cause of failure
- Adaptation / computational steering / feedback & control
- ~~Identify system failures~~ – out of scope

(Our foci today)

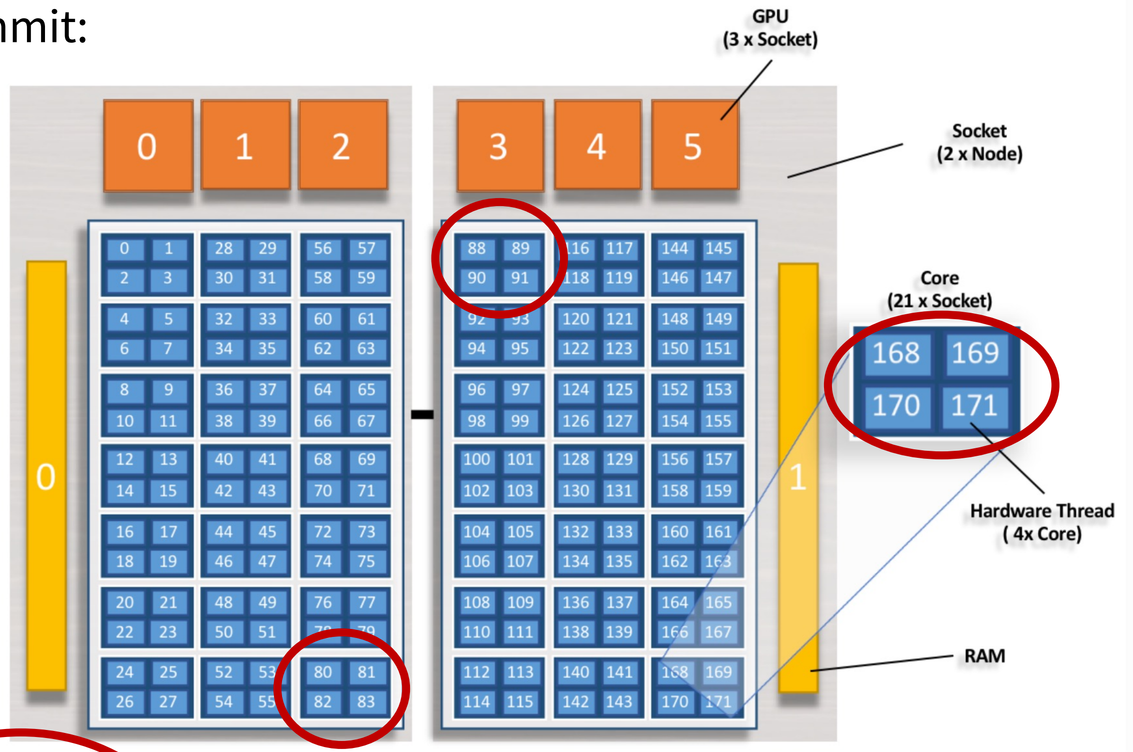
UNIVERSITY OF OREGON

(Mis)Configuration

- Process placement:
  - logical/physical mappings, resources assigned/constrained to each process
  - Reserved core(s) for system?
  - GPU mapping
- Thread placement: Socket, NUMA domain, core, thread (HWT)
- Undersubscribing: Wasted hardware, energy, time (under-utilization)
- Oversubscribing: Increased contention with no realized benefit (or worse, a penalty)
- Imbalances (see process placement, above)
  - What is the communication frequency/volume between pairwise ranks?
- Let’s admit it: Slurm/PBS/Alps/Torque/Flux are *complicated to use*
  - …especially when combined with MPI, OpenMP, GPUs, or other model settings

System documentation can be subtle/confusing, although accurate

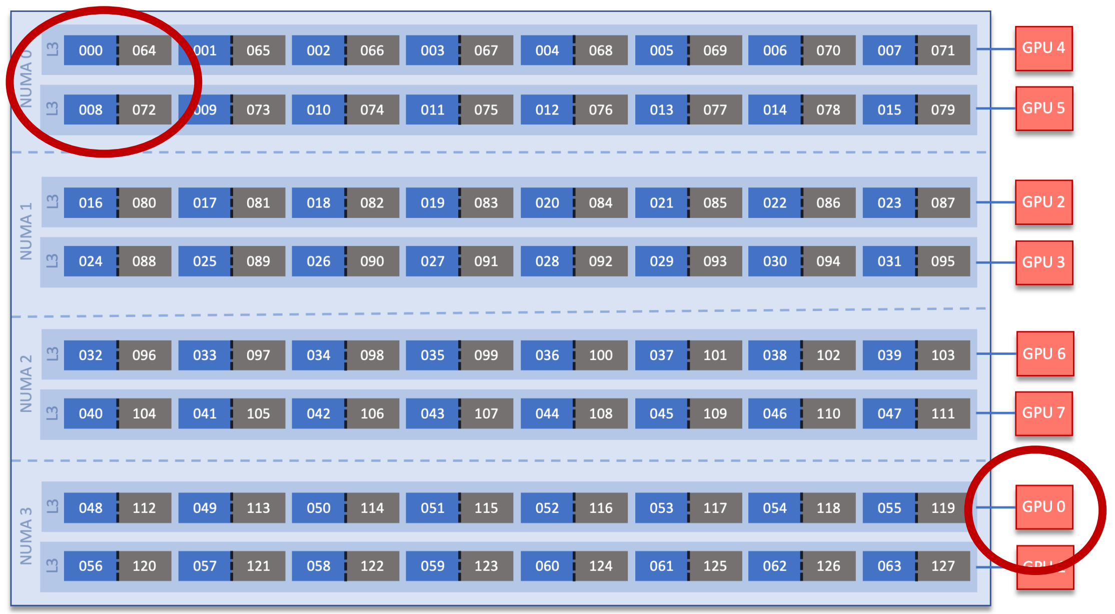
Summit:



- 1 node
- 2 sockets (grey)
- 42 physical cores* (dark blue)
- 168 hardware cores (light blue)
- 6 GPUs (orange)
- 2 Memory blocks (yellow)

***Core Isolation:** 1 core on each socket has been set aside for overhead and is not available for allocation through jsrun. The core has been omitted and is not shown in the above image.

Frontier:

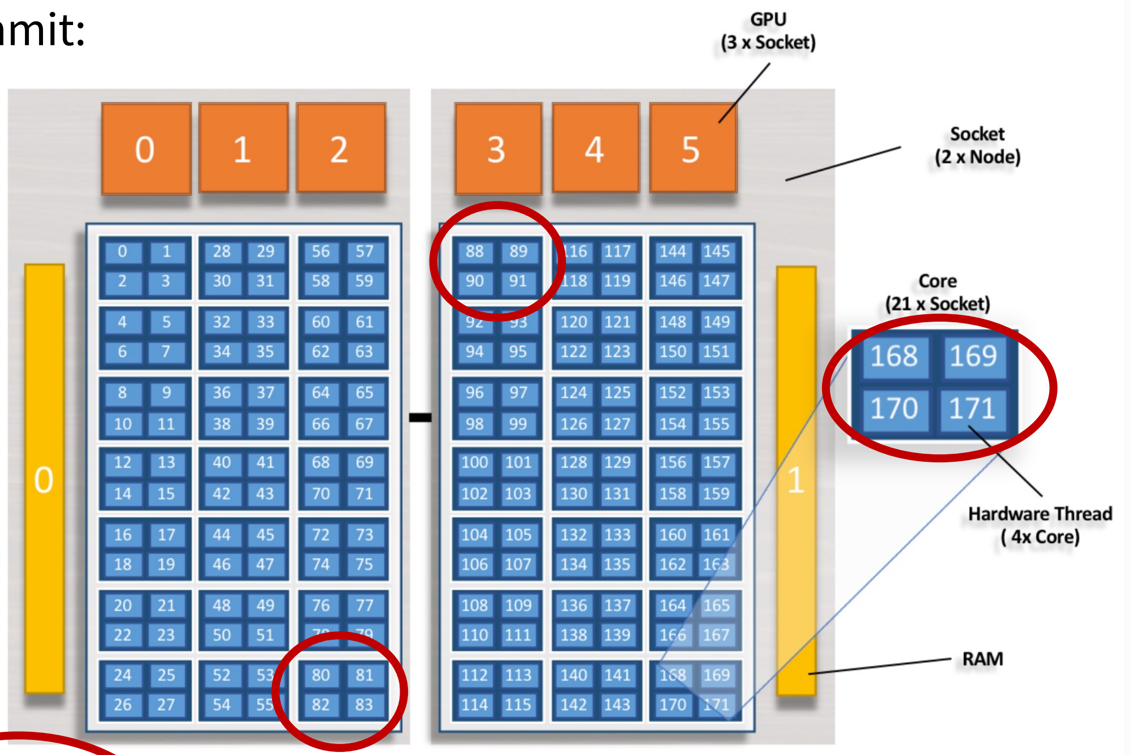


Core indexing, HWT indexing, GPU mapping not universal

Sources: https://docs.olcf.ornl.gov/systems/summit_user_guide.html , https://docs.olcf.ornl.gov/systems/frontier_user_guide.html

System documentation can be subtle/confusing, although accurate

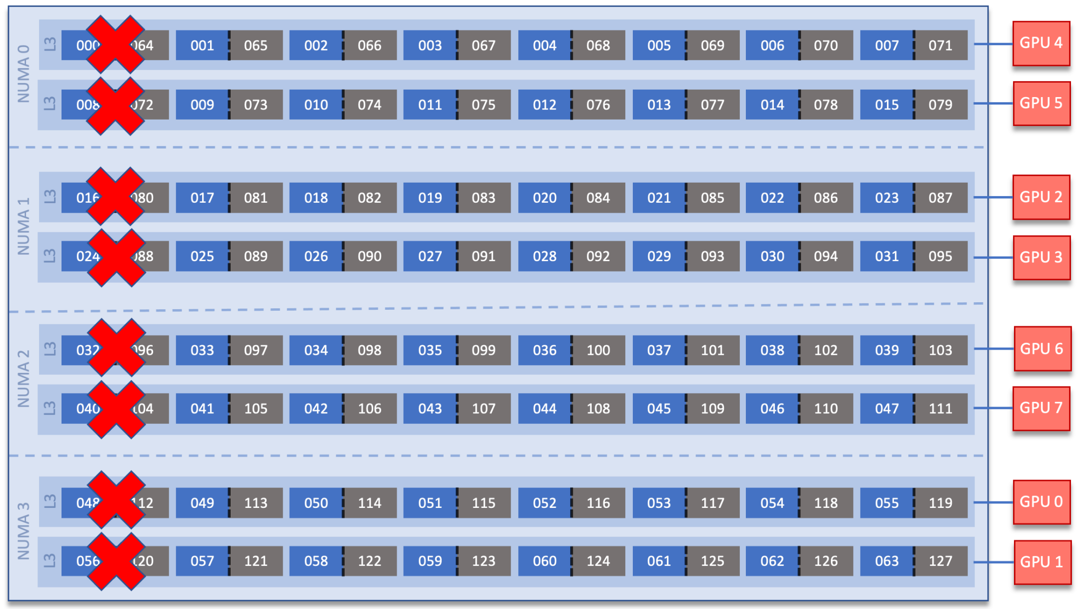
Summit:



- 1 node
- 2 sockets (grey)
- 42 physical cores* (dark blue)
- 168 hardware cores (light blue)
- 6 GPUs (orange)
- 2 Memory blocks (yellow)

***Core Isolation:** 1 core on each socket has been set aside for overhead and is not available for allocation through jsrun. The core has been omitted and is not shown in the above image.

Frontier:

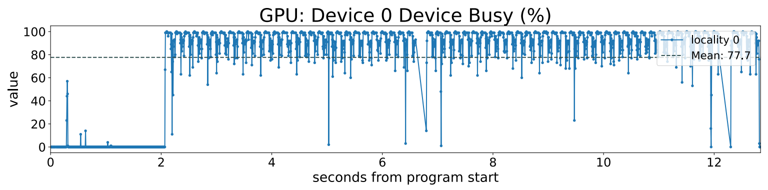


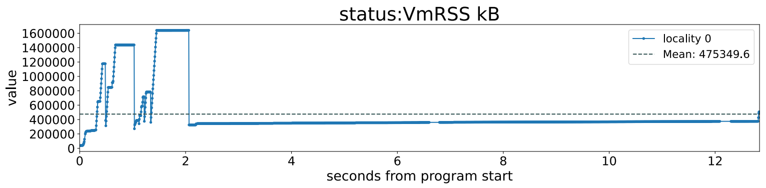
Modified default: system reserves 8 cores…but user controllable

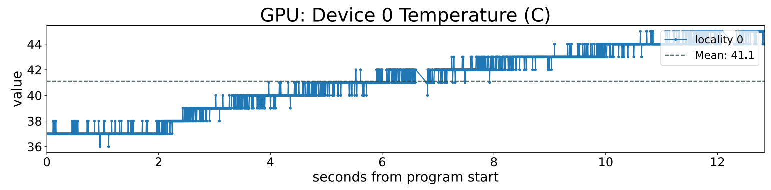
Sources: https://docs.olcf.ornl.gov/systems/summit_user_guide.html , https://docs.olcf.ornl.gov/systems/frontier_user_guide.html

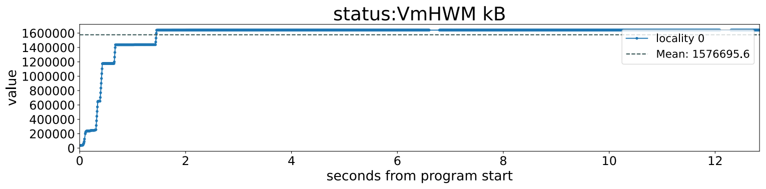
Utilization

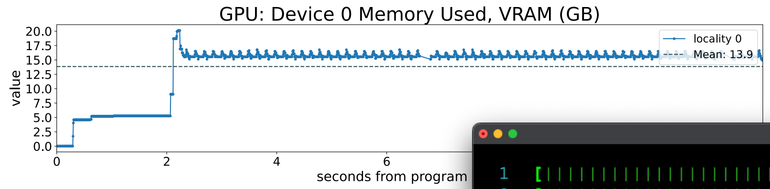
- Resource Monitor, Activity Monitor, menumeters, top, htop, NVML, ROCm-SMI, etc.
  - We LOVE these!
- Why can't we have that for HPC?
  - System admins do!
  - LDMS, Ganglia, Puppet Console, TACC Stats, JobStats, etc.
- …for the user?

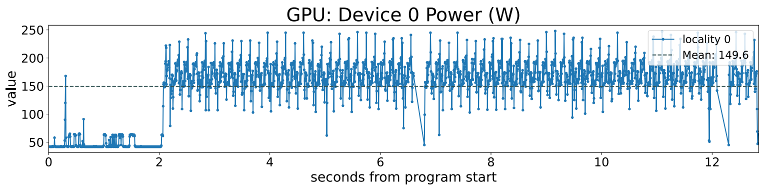


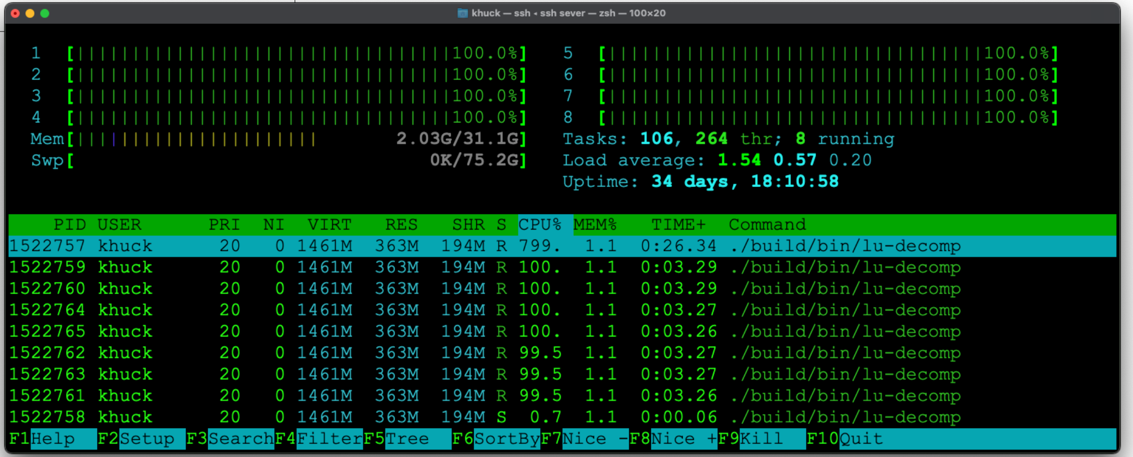












```
  1  [||||||||||||||||||||||||||||||||||100.0%]   5  [||||||||||||||||||||||||||||||||||100.0%]
  2  [||||||||||||||||||||||||||||||||||100.0%]   6  [||||||||||||||||||||||||||||||||||100.0%]
  3  [||||||||||||||||||||||||||||||||||100.0%]   7  [||||||||||||||||||||||||||||||||||100.0%]
  4  [||||||||||||||||||||||||||||||||||100.0%]   8  [||||||||||||||||||||||||||||||||||100.0%]
  Mem[|||||||||||||||||||||          2.03G/31.1G]   Tasks: 106, 264 thr; 8 running
  Swp[                                  0K/75.2G]   Load average: 1.54 0.57 0.20
                                                    Uptime: 34 days, 18:10:58

    PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
1522757 khuck      20   0 1461M  363M  194M R 799.  1.1  0:26.34 ./build/bin/lu-decomp
1522759 khuck      20   0 1461M  363M  194M R 100.  1.1  0:03.29 ./build/bin/lu-decomp
1522760 khuck      20   0 1461M  363M  194M R 100.  1.1  0:03.29 ./build/bin/lu-decomp
1522764 khuck      20   0 1461M  363M  194M R 100.  1.1  0:03.27 ./build/bin/lu-decomp
1522765 khuck      20   0 1461M  363M  194M R 100.  1.1  0:03.26 ./build/bin/lu-decomp
1522762 khuck      20   0 1461M  363M  194M R 99.5  1.1  0:03.27 ./build/bin/lu-decomp
1522763 khuck      20   0 1461M  363M  194M R 99.5  1.1  0:03.27 ./build/bin/lu-decomp
1522761 khuck      20   0 1461M  363M  194M R 99.5  1.1  0:03.26 ./build/bin/lu-decomp
1522758 khuck      20   0 1461M  363M  194M S  0.7  1.1  0:00.06 ./build/bin/lu-decomp
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Figures: time series monitor data from APEX performance measurement tool
https://github.com/UO-OACISS/apex/

UNIVERSITY OF OREGON

What (we think) users DON'T want:

Sources: https://www.space.com/22652-nasa-redesigns-mission-control.html

ZeroSum: User Space Monitoring of Resource Utilization and Contention

- Inspired by the *hello_jsub* program from Tom Papatheodore - https://code.ornl.gov/t4p/Hello_jsrun
- Name "ZeroSum" comes from:
  - the fixed number of resources available in an allocation
  - the need to periodically use *some* resources in order to monitor
- Available on GitHub: https://github.com/khuck/zerosum
- Monitors application threads (LWP), CPU hardware (HWT), Memory, and GPU hardware for all processes, all nodes in the allocation
- Prototype solution to two problems:
  - *Configuration optimization* – help understand new system/software
  - Ongoing usage for monitoring/adaptation

Proposed Functionality for ZeroSum

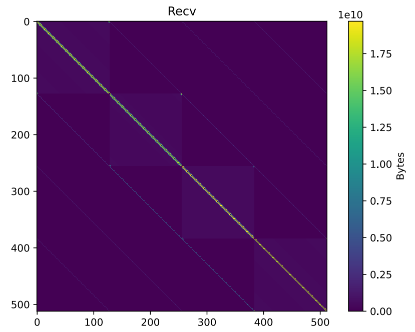
- ✓ **Detect the initial/changing configuration of the application**
- ▪ Evaluate the configuration to automatically detect misconfigurations
- ✓ Provide runtime feedback to the user that the program is progressing
- ✓ **Provide a report of how effectively the hardware was utilized**
- ✓ **Provide a report of how much contention was identified in the execution**
- ▪ Provide a way to export the observed data to other tools that can perform computational steering, if desired

✓ Implemented
▪ Future work

(Our foci today)

How to use ZeroSum

- Wrapper script(s) – `zerosum` and `zerosum-mpi`
  - Periodicity – default 1 second
  - HWT/Core for async thread – defaults to last core/HWT in affinity list for process
  - Detailed output – true/false, default false
  - Verbose output – true/false, default false
  - Heartbeat – true/false, default false
  - Register signal handler – true/false, default false
- Preloads the library, wraps `__libc_start_main` or creates global static constructor/destructor functions
- MPI implementation includes wrappers for `MPI_Recv/Irecv`, `MPI_Sendrecv`, `MPI_Send/Bsend/Isend/Rsend/Ssend` to capture P2P frequency, volume



Configuration Detection

- Query `/proc/[self|pid]/status` to get the allowable cores
- Query `/proc/meminfo` to get total memory available
- Query MPI rank, size, hostname
- If available, use hwloc to query cpu topology (`lstopo`)
- Asynchronous background thread is started, it periodically queries:
  - `/proc/[self|pid]/status` to get process thread count, memory usage
  - `/proc/[self|pid]/task` directory to get all thread IDs
  - For each thread, query the affinity list for that thread (it may change!), utilization, state, core/HWT it's running on, context switches
  - `/proc/stat` to query core utilization of all cores
- OMP-Tools (v5.0+) callback used to identify OpenMP threads at creation
- NVML/ROCm-SMI/SYCL libraries used to query GPU(s)*

UNIVERSITY OF OREGON

Utilization Report

- Rank 0 writes a summary report to the screen
- All ranks write a report to a log file – including full time series data as CSV
- All observed threads (LWP) are reported – user/system/idle, context switches, affinity list
- All assigned cores (HWT) are reported – user/system/idle
- All assigned GPUs are reported – stats captured with NVML or ROCm-SMI, SYCL has limited support (memory only)

Example Output – miniQMC (OpenMP target offload) on Frontier (OLCF)

```
Duration of execution: 210.878 s

Process Summary:
MPI 000 - PID 51334 - Node frontier09085 - CPUs allowed: [1,2,3,4,5,6,7]

LWP (thread) Summary:
LWP 51334: Main,OpenMP - stime: 12.48, utime: 63.94, nv_ctx:   4, ctx: 365488, CPUs: [1]
LWP 51343:     ZeroSum - stime:  0.15, utime:  0.26, nv_ctx:   9, ctx:    679, CPUs: [7]
LWP 51374:       Other - stime:  0.00, utime:  0.00, nv_ctx:   0, ctx:      6, CPUs:
    [1-7,9-15,17-23,25-31,33-39,41-47,49-55,57-63,65-71,73-79,81-87,89-95,97-103,
     105-111,113-119,121-127]
LWP 51384:      OpenMP - stime: 12.60, utime: 64.00, nv_ctx:   3, ctx: 365742, CPUs: [3]
LWP 51385:      OpenMP - stime: 12.63, utime: 64.27, nv_ctx:   2, ctx: 352574, CPUs: [5]
LWP 51386:      OpenMP - stime: 12.74, utime: 63.76, nv_ctx: 473, ctx: 368585, CPUs: [7]
```

Process Summary

LWP (thread) Summary

Note: times are % of total

UNIVERSITY OF OREGON

Example Output – miniQMC (OpenMP target offload) on Frontier (OLCF)

HWT (core/thread) Summary

GPU Summary

Note: times are % of total

```
Hardware Summary:
CPU 001 - idle:  22.70, system:  12.42, user:  64.52
CPU 002 - idle:  99.82, system:   0.00, user:   0.00
CPU 003 - idle:  23.08, system:  12.60, user:  63.97
CPU 004 - idle:  99.83, system:   0.00, user:   0.00
CPU 005 - idle:  22.79, system:  12.62, user:  64.23
CPU 006 - idle:  99.83, system:   0.00, user:   0.00
CPU 007 - idle:  22.94, system:  12.89, user:  63.81

GPU 0 - (metric: min  avg  max)
    Clock Frequency, GLX (MHz): 800.000000 1614.691943 1700.000000
    Clock Frequency, SOC (MHz): 1090.000000 1090.000000 1090.000000
    Device Busy %: 0.000000 14.616114 52.000000
    Energy Average (J): 0.000000 8.328571 10.000000
    GFX Activity: 0.000000 17223.704762 38443.000000
    GFX Activity %: 0.000000 13.706161 41.000000
    Memory Activity: 0.000000 623.623810 1536.000000
    Memory Busy %: 0.000000 0.355450 3.000000
    Memory Controller Activity: 0.000000 0.303318 2.000000
    Power Average (W): 90.000000 126.483412 138.000000
    Temperature (C): 35.000000 37.909953 39.000000
    UVD|VCN Activity: 0.000000 0.000000 0.000000
    Used GTT Bytes: 11624448.000000 11624448.000000 11624448.000000
    Used VRAM Bytes: 15044608.000000 4743346651.601895 4839596032.000000
    Used Visible VRAM Bytes: 15044608.000000 4743346884.549763 4839596032.000000
    Voltage (mV): 806.000000 891.848341 906.000000
```

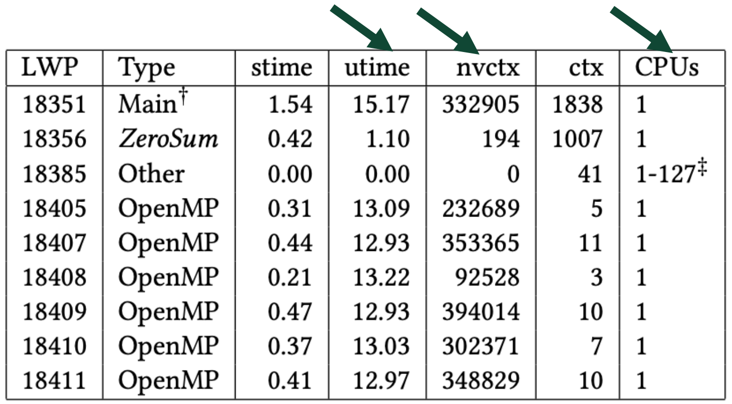
UNIVERSITY OF OREGON

Contention Detection Support

- (Non) voluntary context switches
- Minor/major page faults, pages swapped
- System time
- Comparing affinity lists – across threads *and* across processes
- Memory consumption
- GPU memory consumption
- Currently a manual process
- Not automated yet – but *should* be straightforward…

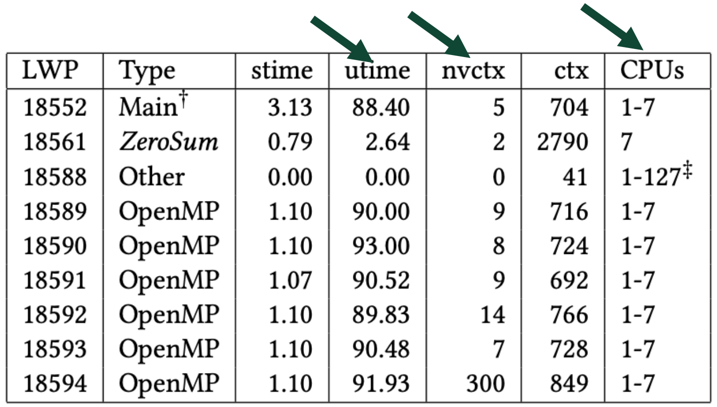
Evaluation / Example usage

MPI+OpenMP version of miniQMC on Frontier, 8 processes, 7 threads (64 cores, 1 thread per core, 8 cores reserved)



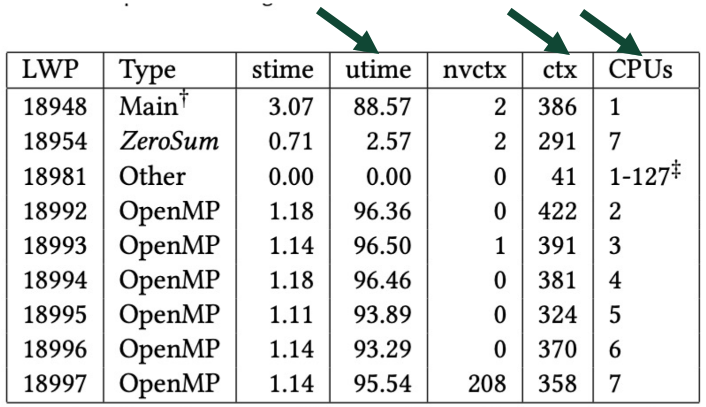
LWP	Type	stime	utime	nvctx	ctx	CPUs
18351	Main[†]	1.54	15.17	332905	1838	1
18356	*ZeroSum*	0.42	1.10	194	1007	1
18385	Other	0.00	0.00	0	41	1-127[‡]
18405	OpenMP	0.31	13.09	232689	5	1
18407	OpenMP	0.44	12.93	353365	11	1
18408	OpenMP	0.21	13.22	92528	3	1
18409	OpenMP	0.47	12.93	394014	10	1
18410	OpenMP	0.37	13.03	302371	7	1
18411	OpenMP	0.41	12.97	348829	10	1

Table 1: Frontier results, default configuration. †indicates that the main thread is also an OpenMP thread. ‡indicates that the first core of each L3 region was set aside for system processes, not all threads in the sequence 1-127 are allowed but summarized for brevity in the table (see LWP 51274 in Listing 2).



LWP	Type	stime	utime	nvctx	ctx	CPUs
18552	Main[†]	3.13	88.40	5	704	1-7
18561	*ZeroSum*	0.79	2.64	2	2790	7
18588	Other	0.00	0.00	0	41	1-127[‡]
18589	OpenMP	1.10	90.00	9	716	1-7
18590	OpenMP	1.10	93.00	8	724	1-7
18591	OpenMP	1.07	90.52	9	692	1-7
18592	OpenMP	1.10	89.83	14	766	1-7
18593	OpenMP	1.10	90.48	7	728	1-7
18594	OpenMP	1.10	91.93	300	849	1-7

Table 2: Frontier results, configuration requesting 7 cores per process. †indicates that the main thread is also an OpenMP thread. ‡indicates that the first core of each L3 region was set aside for system processes, not all threads in the sequence 1-127 are allowed but summarized for brevity in the table (see LWP 51274 in Listing 2).



LWP	Type	stime	utime	nvctx	ctx	CPUs
18948	Main[†]	3.07	88.57	2	386	1
18954	*ZeroSum*	0.71	2.57	2	291	7
18981	Other	0.00	0.00	0	41	1-127[‡]
18992	OpenMP	1.18	96.36	0	422	2
18993	OpenMP	1.14	96.50	1	391	3
18994	OpenMP	1.18	96.46	0	381	4
18995	OpenMP	1.11	93.89	0	324	5
18996	OpenMP	1.14	93.29	0	370	6
18997	OpenMP	1.14	95.54	208	358	7

Table 3: Frontier results, configuration requesting 7 cores per process and binding OpenMP threads to cores. †indicates that the main thread is also an OpenMP thread. ‡indicates that the first core of each L3 region was set aside for system processes, not all threads in the sequence 1-127 are allowed but summarized for brevity in the table (see LWP 51274 in Listing 2).

```
export OMP_NUM_THREADS=7
srun -n8 zerosum-mpi miniqmc
```

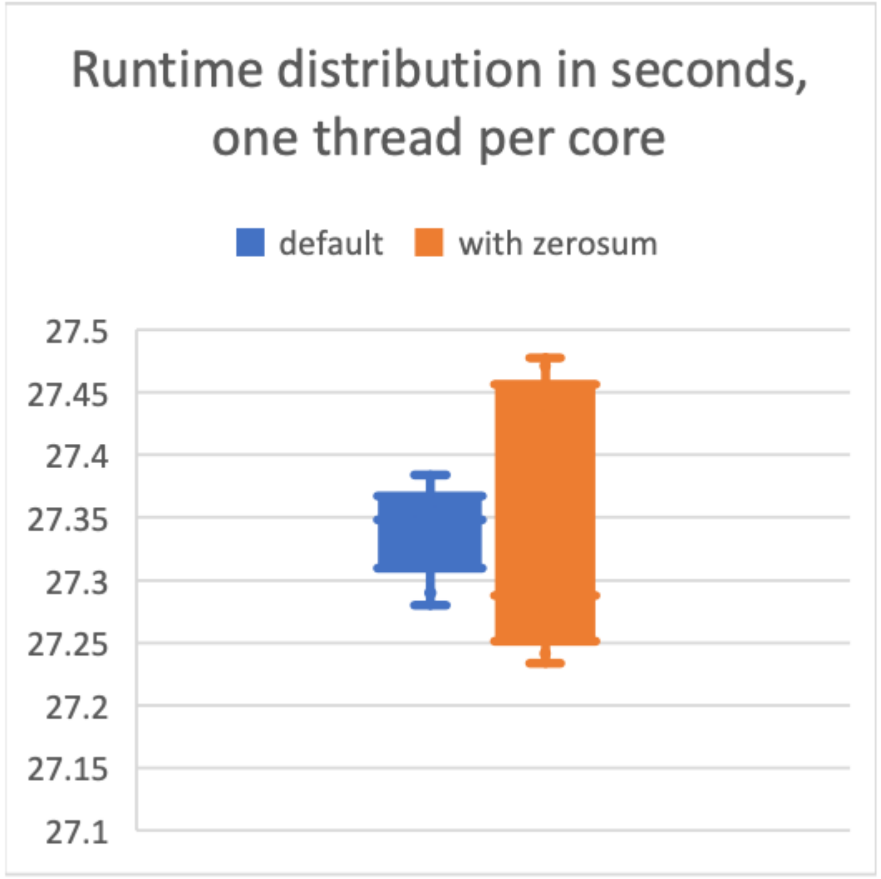
```
export OMP_NUM_THREADS=7
srun -n8 -c7 zerosum-mpi miniqmc
```

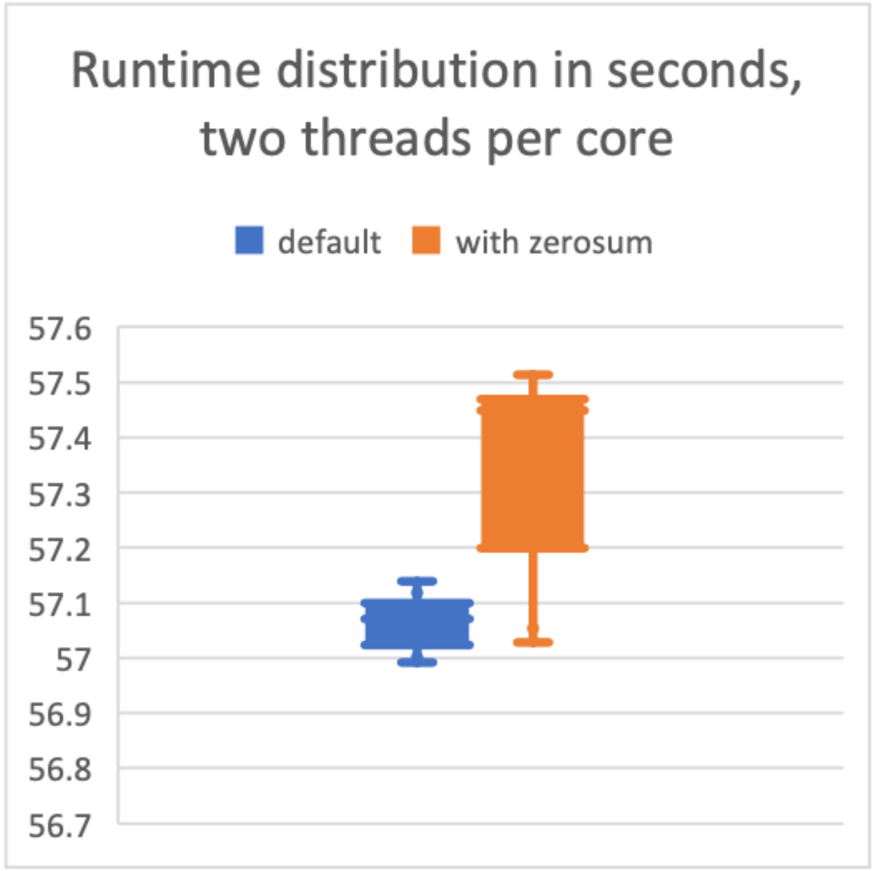
```
export OMP_NUM_THREADS=7
export OMP_PROC_BIND=spread
export OMP_PLACES=cores
srun -n8 -c7 zerosum-mpi miniqmc
```

UNIVERSITY OF OREGON

Note: stime/utime reported as % of total time

Overhead – less than 0.5% in resource constrained example





miniQMC time distributions executed 10 times using one OpenMP thread per core (left). In this comparison, the distribution of times with ZeroSum is noisier, but there is no significant observation of measurable overhead. The right figure shows the time distributions using two OpenMP threads per core. In this comparison, the distribution of times with ZeroSum is both noisier and longer tailed, and does show an observation of overhead, averaging about 0.2752 seconds, or 0.5%.

UNIVERSITY OF OREGON

Conclusions, Future Work

- ZeroSum meets a need – it addresses the *configuration optimization* problem
  - Future development driven by user requests
  - https://github.com/khuck/zerosum (will move to UO-OACISS group eventually)
- Need automated misconfiguration/contention detection, deadlock detection(?)
- Output data could be better – use ADIOS2 BP5? HDF5?
  - Depends on analysis needs/wants
  - Current log file “format” means analysis process is manual/ad hoc for now
- Streaming data to Mochi (SOMA) or other service (LDMS)?
  - Enables robust monitoring approach
- Integration with performance tools (TAU, APEX, etc)
  - Analysis of application performance data in context of system monitoring data
- Input for automated feedback/control (APEX, Argobots, SOMA, application, etc.)

Acknowledgements

Parts of this research was supported by the Exascale Computing Project (17-SC-20-SC), a joint project of the U.S. Department of Energy's Office of Science and National Nuclear Security Administration, responsible for delivering a capable exascale ecosystem, including software, applications, and hardware technology, to support the nation's exascale computing imperative.

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.

U.S. DEPARTMENT OF ENERGY | Office of Science

RAPIDS

ECP EXASCALE COMPUTING PROJECT

Thanks! Questions?

UNIVERSITY OF OREGON