LA-UR-23-24415

Approved for public release; distribution is unlimited.

Title:	El Capitan hackathon, April 2023: LANL code team outbriefs
Author(s):	Pietarila Graham, Anna Mataleena; Haack, Jeffrey Robert; Lakshmiranganatha, Sumathi; Pietarila Graham, Jonathan David; Pritchard, Howard Porter Jr.; Lee, Theresa Joyce; Henderson, Thomas Gerard Lowe; Hart, Nathan Henry; Saller, Thomas; Ferenbaugh, Charles Roger; Mauney, Christopher Michael; Fogerty, Shane Patrick
Intended for:	Report
Issued:	2023-04-26

El Capitan hackathon, April 2023: LANL code team outbriefs

Anna Pietarila Graham

Capsaicin: Jeffrey Haack

FleCSI: Sumathi Lakshmiranganatha, Jonathan Pietarila Graham

Pagosa: Howard Pritchard, Theresa Lee, Thomas Henderson

Partisn: Nathan Hart, Thomas Saller

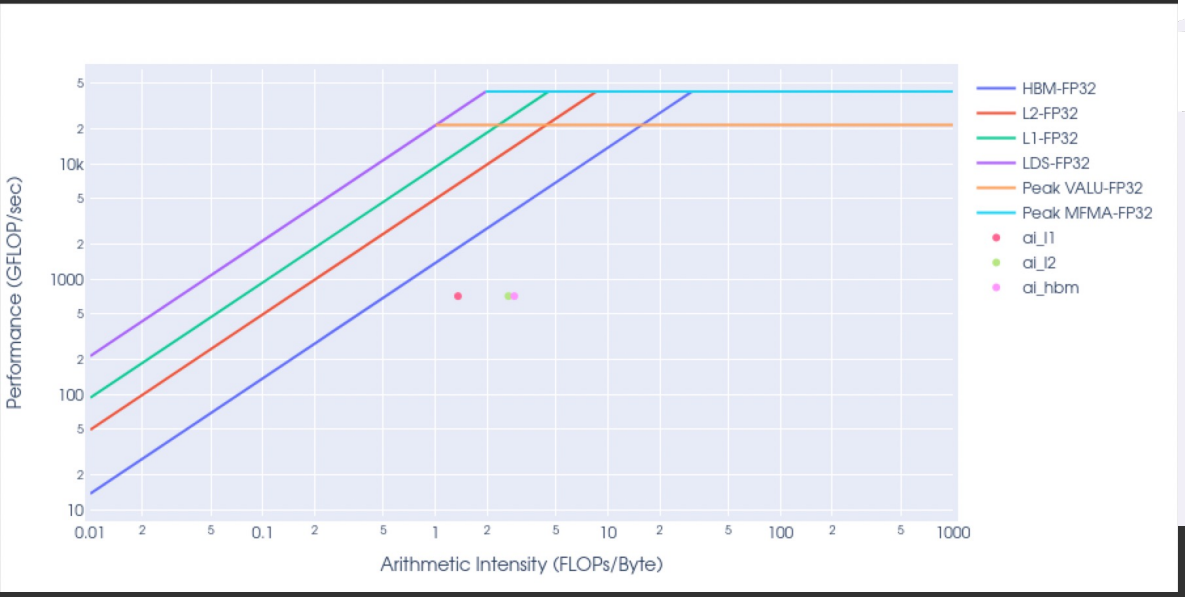
xRage: Charles Ferenbaugh, Christopher Mauney, Shane Fogerty

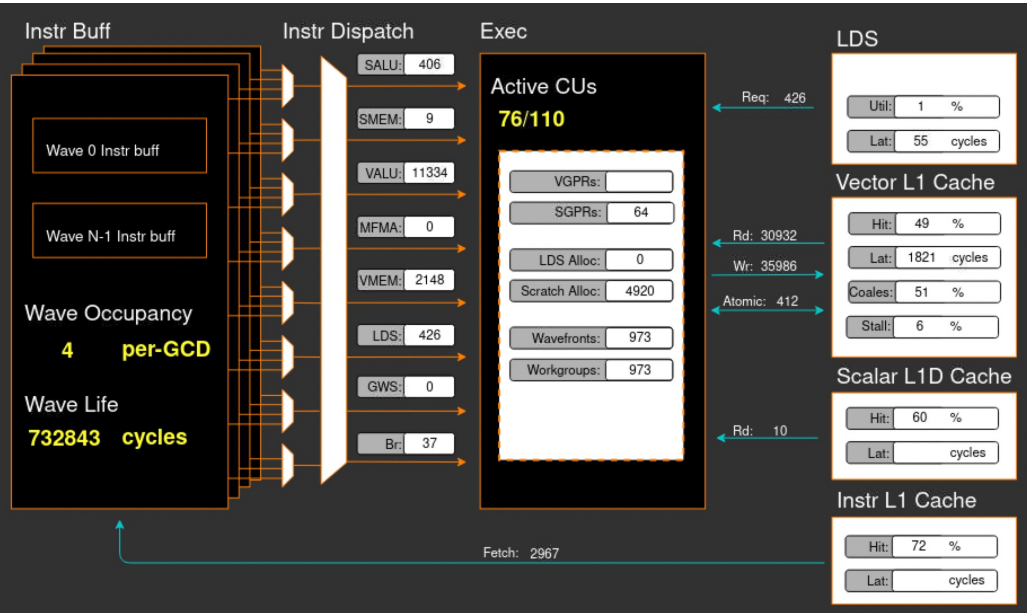
El Cap COE Hackathon: Capsaicin

- Goals:
  - What did you plan to do?
    - Make sure hip-enabled branch can still build with newer env changes from upstream
    - We would like to work through an AMD optimization report for 3D sweep kernel
    - Get a sense of comparison with V100 results (though these mostly 2D)
- Accomplishments:
  - Ran omniperf for differently sized problems
  - Got rooflines
  - Learned to ‘translate’ lingo from omniperf results, related to findings from V100s
  - Experimented with launch bounds, tinkering with memory footprint
- Blockers encountered:
  - Omniperf a little slow for large sized problem. Or I could just size problems better.
- Lessons Learned:
  - We might want kernel fission not kernel fusion for performance in 3D
    - And it might help in 2D too?
- To Be Improved: Anything that could be improved for future hackathon events
  - It would have been helpful if we had widescreen monitors in the room
    - (this was already in the template, but still true!)

El Cap COE Hackathon: Capsaicin

Empirical Roofline Analysis (FP32/FP64)



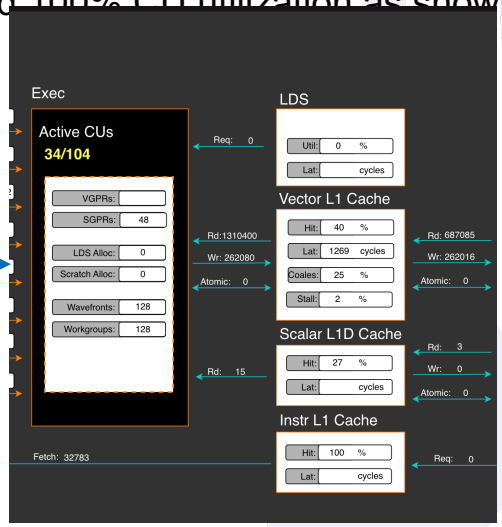


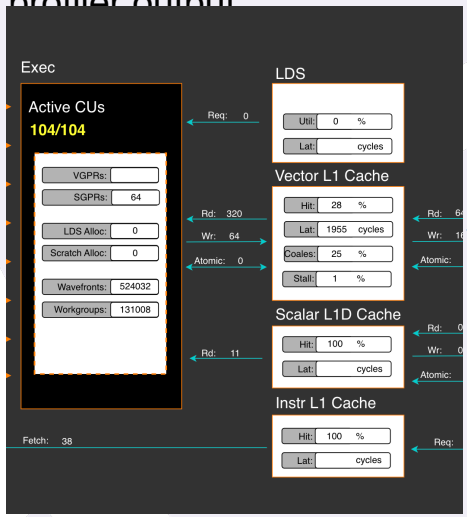

```
; Kernel info:
; codeLenInByte = 192
; NumSgprs: 52
; NumVgprs: 256
; NumAgprs: 256
; TotalNumVgprs: 512
; ScratchSize: 4936
; MemoryBound: 0
; FloatMode: 240
; IeeeMode: 1
; LDSByteSize: 0 bytes/workgroup (compile time only)
; SGPRBlocks: 6
; VGPRBlocks: 63
; NumSGPRsForWavesPerEU: 52
; NumVGPRsForWavesPerEU: 512
; AccumOffset: 256
```

El Cap COE Hackathon: FleCSI

- Goals:
  - Profile the toy application (Poisson) and understand the profiler output
  - Improve performance on AMD GPUs with better cache and CUs utilization
  - Improve scaling results for multi-gpu runs
- Accomplishments:
  - Change the loop structure and used Kokkos MDRange to exploit more parallelism and achieved 100% CU utilization as shown by profiler output

Before –only outer loop parallelization





With Kokkos MDRange Policy

  - Reduced the kernel execution time from 7s to 1s with the above modification on MI250
- Next steps to do…
  - Need to compare the results with NVIDIA GPU performance and FleCSI's mdiota view for collapsing the loop
  - Perform scaling studies with the changes
  - Use omnitrace

Los Alamos NATIONAL LABORATORY Sandia National Laboratories

El Cap COE Hackathon: FleCSI

- Lessons Learned:
  - How to use omniperf on Tioga
  - Need to pass all the command line arguments as a script for omniperf on Tioga
- Blockers
  - Issues building spack packages with gcc 12.2.0 on Tioga
- To be improved
  - More documentation, tutorial examples..

El Cap COE Hackathon: Pagosa

- Goals:
  - Use/get familiar with rocgdb, rocprof (rocm tools)
  - Run PERF_Shaped_Charge_25mat offload on 1 node with 1,2,4,8 ranks on rzvernal and rzansel
  - Run PERF_Shaped_Charge_25mat without offload on 1 node with 1,2,4,8 ranks on rzvernal
  - Compare results
  - Identify opportunities to improve offload performance
- Accomplishments:
  - Able to profile and trace with rocprof (indicates >95% of time step time spent in data movement to-from GPU)
  - Adding openmp data directives to prevent some transfers. Ran into known bug, workaround seems to work
  - Learned to set CRAY_ACC_DEBUG environment variables and look at the output
- Blockers encountered:
  - Did not get Perftools working
  - Hit a CCE compiler bug that prevented us from using !$omp declare target
- Lessons Learned:
  - Rocm utilities may work better for us than Perf utilities
  - Avoid using !$omp declare target for module variables for the time being
- To Be Improved
  - Better access to electrical outlets
- Wishlist
  - More time to work things out

El Cap COE Hackathon: PARTISN

- Goals:
  - Stand up CCE-driven HIP capabilities on EAS3
  - Begin a preliminary performance study between CCE-HIP, ROCm-HIP, and XL-CUDA
  - Get other team members started porting kernels
- Accomplishments:
  - CCE seems to be working
  - We've used omniperf to generate preliminary performance reports with CCE- and ROCm-driven HIP
  - A complete newbie to the porting process has ported 4 kernels from CUDA Fortran to CUDA C – next step is porting to HIP
- Blockers encountered:
  - Seeing run-to-run variability in our QAs – this was a known issue, but it may be making profiling difficult
  - Potential bug in profiling tool?
  - A deficiency in our build system that caused errors when using atomics with CUDA C was identified and fixed
- Lessons Learned:
  - It looks like we're integer instruction-bound (for the 2 kernels we analyzed) – not surprising given we are porting from Fortran where we utilize up to 5D arrays to a flat C implementation
  - We have very few calls to L2 or HBM – mostly in L1 and LDS (for that problem)

El Cap COE Hackathon: PARTISN

- To Be Improved:
  - We had no issues with VPN or power cables, but the former did indirectly hamper us when working with vendors
  - WiFi issues aside, the location was nice for commuting to/from Los Alamos and Santa Fe
- Wishlist
  - None

El Cap COE Hackathon: xRage

- Goals:
  - Main: Initial performance analysis of current build + unsplit_triplepoint test
  - Secondary: Debugging on other test cases
  - Secondary: New build with upgraded CCE (14.0.3 -> 15.0.1), ROCM (5.2.3 -> 5.4.3)
- Accomplishments:
  - Ran rocprof/omniperf on a suite of test problems, using a selection of parallel run configs. (unsplit hydro is only part that is offloaded; tested simple unsplit inputs)
  - Made a full Spack buildout with upgraded config - working through build/run issues
- Blockers encountered:
  - Spack build issues for various compiler configs (documented for Srinath/HPE)
  - Data visualization/comparison (mostly network issues); omniperf crash for some benchmark runs (talked to AMD staff, seems a common issue).
- Lessons Learned:
  - COE-provided Spack configs may not work for us, or may need tweaking
  - Code is only partially on GPU, and what is there can be improved.
- To Be Improved: Anything that could be improved for future hackathon events
  - It would have been helpful if we had widescreen monitors in the room
  - Strong, reliable network connection

El Cap COE Hackathon: xRage

- Overview profile from rocprof, then zoomed in on the actual GPU computation