LA-UR-22-32229
Approved for public release; distribution is unlimited.

Title:	LANL CODE TEAMS' OUTBRIEFS FROM EL CAP HACKATHON
Author(s):	Pietarila Graham, Anna Mataleena; Haack, Jeffrey Robert; Long, Alex Roberts; Mauney, Christopher Michael; Holladay, Daniel Alphin; Aulwes, Rob Tuan; Edelmann, Philipp Valentin Ferdinand; Pietarila Graham, Jonathan David; Lakshmiranganatha, Sumathi; Matsekh, Anna M.; Hart, Nathan Henry; Zerr, Robert Joseph; Magee, Daniel J.
Intended for:	Report
Issued:	2022-11-21

Los Alamos
NATIONAL LABORATORY

Los Alamos National Laboratory, an affirmative action/equal opportunity employer, is operated by Triad National Security, LLC for the National Nuclear Security Administration of U.S. Department of Energy under contract 89233218CNA000001. By approving this article, the publisher recognizes that the U.S. Government retains nonexclusive, royalty-free license to publish or reproduce the published form of this contribution, or to allow others to do so, for U.S. Government purposes. Los Alamos National Laboratory requests that the publisher identify this article as work performed under the auspices of the U.S. Department of Energy. Los Alamos National Laboratory strongly supports academic freedom and a researcher's right to publish; as an institution, however, the Laboratory does not endorse the viewpoint of a publication or guarantee its technical correctness.

LANL CODE TEAMS' OUTBRIEFS FROM EL CAP HACKATHON

Capsaicin (Jeffrey Haack)
Jayenne (Alex Long)
xRage (Chris Mauney, Daniel Holladay, Rob Aulwes)
FleCSi (Philipp Edelmann, Jonathan Pietarila Graham, Sumathi Lakshmiranganatha, Anna Matsekh)
Partisn (Nathan Hart, Robert Zerr, Daniel Magee)

CoE (Anna Pietarila Graham)

10/04/2022 Sandia

Managed by Triad National Security, LLC, for the U.S. Department of Energy's NNSA.

El Cap COE Hackathon Outbrief : Capsaicin

- Goals:
  - Hipify cu files
  - Figure out how hip interfaces with our build system
  - Get up to speed with existing compiler issues
  - Run something in our code on an AMD GPU
- Accomplishments
  - Hipify cuda modules ✅
  - Find solution to tcmalloc compiler issue so we can actually test ✅?
  - Figure out how hip interfaces with our build system 🏗️
- Blockers encountered:
  - Sounds like cce15 will be great!
  - Access to LANL git servers from rzvernal would be handy
- Lessons Learned:
  - Make sure you have system access sooner than the day before hackathon...
- To Be Improved:
  - All good - looking forward to next event!

El Cap COE Hackathon Outbrief : Jayenne

- Goals:
  - Get the GPU version of Jayenne (develop) compiling with HIP
  - Get Initial performance data for stand alone driver and compare to V100
- Accomplishments:
  - Successfully integrated CCE + clang into our Cmake build system!
  - Found gaps in our host/device decorations (clang seems to be very careful)
  - GPU kernels are compiling with HIP + M. Rowan’s Random123!
- Blockers encountered:
  - Cmake picks the clang compiler in the ROCM module unless you tell it to use CCE (fine? Bug?)
  - Cmake cannot compile a simple HIP test program with CCE because it fails to find the “CLANGRT_BUILTINS” library (seems to fail in enable_language(HIP) path only), Alex will report bug, workaround points CMAKE directly to the library in rocm module path
  - Shared memory array does not allow default initialization of custom types (e.g. __shared__ Particle particles[64] not allowed), work around with reinterpret cast
- Lessons Learned:
  - Don’t declare compiler specific preprocessor variables (__NVCC__) no matter how convenient it may seem

El Cap COE Hackathon Outbrief : xRage

- Goals:
  - Understand profile outputs to determine how efficient we're using UVM
  - Explore other profiling tools
  - Resolve build issues
- Accomplishments:
  - Successfully ran miperf, rocprof, hpctoolkit profilers
  - EAP tests in Rage suite ran with the HIP backend
    - 164 passes, 16 diffs, 24 fails
- Blockers encountered:
  - Working through Spack buildouts, CMake versioning
  - Figuring out the correct set of HW counters for a particular performance metric
- Lessons Learned:
  - How to use hpctoolkit and miperf
  - Understanding profiling metrics
  - If you *think* your interactive allocation is close to ending prior to launching a profile that takes > 30 minutes, get a new allocation rather than losing your allocation 29 minutes into the run
- To Be Improved: Anything that could be improved for future hackathon events
  - "One-stop" documentation (various slides/tutorials/how-tos).

El Cap COE Hackathon Outbrief : xRage

wish list

- pre-defined groups of metrics for omniperf so that it reduces the # of replays
- better documentation of the metrics
- source-level metrics (like Intel Vtune)
- timeline trace that links back to source (which source is generating the copyToDevice?)
- GUI tool to help with srun bindings
- Print Kokkos parallel loop name
- Several current and upcoming development efforts rely on the relaxed constexpr flag for host/device markups. Not having this feature available could be a significant hurdle.

El Cap COE Hackathon Outbrief : FleCSI

- Goals:
  - Build and run MPI & Legion backends on GPU
  - Initial look at performance profiling
  - Stretch: full spack build recipe
- Accomplishments:
  - Ran toy poisson problem on both FleCSI backends
- Blockers encountered:
  - Cray MPICH & bool
  - List any problems that need to be resolved by HPE/AMD
  - Legion + Kokkos only runs on 1 GPU
  - Not building entire stack with spack
- Lessons Learned:
  - Build entire stack with rocmcc@5.3.0
- To Be Improved:
  - more docs, tutorials, ...

El Cap COE Hackathon Outbrief : PARTISN

- Goals:
  1. Run a kernel on an AMD GPU (we are a Fortran code with all GPU capabilities via CUDA Fortran, so this is not trivial) and assess performance versus NVIDIA GPU.
  2. Identify source of run-to-run variability for a single QA test.
  3. Determine long-term path forward for memory management (we explicitly manage memory on host vs. device using native CUDA Fortran routines).
- Accomplishments:
  1. We ran a kernel on an AMD GPU! Isolated kernel vs. kernel performance is favorable to AMD GPUs* vs. NVIDIA GPUs.
  2. Diff has only been observed to fail on Cray machines (Trinitite, RZNevada, RZVernal) regardless of compiler or MPI flavor. Source not identified.
  3. Long-term path forward seems to be unified memory approach.
- Blockers encountered: CCE bugs, potential issue on Cray machines, HIP memory management (e.g., hipMemPrefetchAsync not asynchronous)
- Lessons Learned: We have a simpler path forward than porting our entire CUDA Fortran code structure to C/HIP
- Lessons *not* Learned: While this model seems highly likely to work for a machine w/homogeneous memory, how will it compare for heterogeneous memory vs. explicit memory management?

Los Alamos NATIONAL LABORATORY