OpenMP experiences with thornado

J. Austin Harris (ORNL)
M. Paul Laiu (ORNL)
Ran Chu (UTK)
Eirik Endeve (ORNL)
thornado
(toolkit for high-order neutrino rad-hydro)

- ExaStar proxy application for spectral neutrino transport in stellar astrophysics simulations
  - Designed to be incorporated into AMR-based codes (e.g., Flash-X)
  - Focus on single MPI rank+GPU performance

- Tabulated nuclear equation of state and neutrino opacity tables from WeakLib

- GPU port of original CPU code via compiler directives and linear algebra libraries

- Discontinuous Galerkin (DG) method for phase-space discretization
  - 1 AMR block $\approx 8^3$ spatial elements = 4096 nodes
  - 16 energy elements = 32 nodes
thornado time integration

- Positive, diffusion accurate, two-stage implicit-explicit (PD-IMEX) time integration
  - Explicit neutrino advection operator
  - Implicit neutrino collision operator
    - Nonlinear solver
    - Tabulated microphysics
  - Numerical limiters to maintain realizability and conservation

Enter from Flash-X

1. Explicit Update (Advection)
2. Apply Limiters
3. Implicit Update (Collisions)
4. Apply Limiters
5. Explicit Update (Advection)
6. Apply Limiters
7. Implicit Update (Collisions)
8. Apply Limiters

Exit to Flash-X
OpenMP features

• Ubiquitous use of:
  – target teams distribute parallel do simd collapse
  – target enter data, target exit data
  – target data use_device_ptr
  – declare target

• In the future:
  – metadirective (for conditional GPU execution at runtime)
  – interop (for asynchronous OpenMP and linear algebra interoperability)
  – collapse of non-rectangular loop nest (for triangular loops)
GPU Strategy

Compiler Directives

```c
#include "THORANADO.HOMP"
!OMP TARGET TEAMS DIRECTIVE PARALLEL DO SIMD COLLAPSE(7)
#define defined(THORANADO,HOMP)
!OMP PARALLEL LOOP GANG ASYNC( async.flag ) IF( do_gpu ) THEN
#define defined(WEAKLIB.HOMP)
!ACC LOOP VECTOR PRIVATE( i0, j0, i, j )
#define defined(WEAKLIB.OCC)
!ACC LOOP VECTOR PRIVATE( i0, j0, i, j )
#define defined(WEAKLIB.OCC)
!ACC LOOP VECTOR PRIVATE( i0, j0, i, j )
#define defined(WEAKLIB.OCC)
!ACC LOOP VECTOR PRIVATE( i0, j0, i, j )
#define defined(WEAKLIB.OCC)
!ACC LOOP VECTOR PRIVATE( i0, j0, i, j )
```
Example #1
LogInterpolateSingleVariable_2D2D_CustomAligned

• ~40% of runtime in this kernel
• 2D interp. of opacity table for each spatial point and (E1,E2) pair
• Called multiple times per nonlinear solver iteration in implicit step
Example #2
ApplyPositivityLimiter_TwoMoment

- ~20% of runtime in this routine
- Check that each point is a realizable physical solution
- Called 5 times per call to thornado
- Contains bisection iteration
- Very low arithmetic intensity
Example #3
Computelncrement_Divergence_X{1,2,3}

- ~10% of runtime in these routines
- Computes IMEX explicit update
- Called twice per call to thornado
- Healthy mix of linear algebra libraries and compiler directives
- Arrays permuted for better striding
Example #4
SolveMatterEquations_FP_Coupled

- High level driver routine for non-linear solver
- Composed of many kernels, including opacity interpolations (Example #1)
- Called twice per call to thornado
- Only need to calculate for non-converged points:
  - With MASK array
  - Packing/unpacking (scatter/gather)
Compiler status

• Can thornado compile and run with ____
  – NVIDIA HPC SDK (Summit/Ascent/CoriGPU)?
    • Yes (Preferred development compiler for OpenMP and OpenACC)
  – IBM XL(Summit/Ascent)?
    • Yes (but very long compile times)
  – Intel oneAPI (Iris/Arcticus)?
    • Somewhat (internal compiler errors); collaboration with Intel and ANL COE
    • Can compile and test stand-alone kernels
  – CCE (Crusher/Birch)?
    • Somewhat (internal compiler errors)
    • Can compile and run small number of tests
Streaming Sine Wave Test Problem

- Verification problem to test explicit operator
  - Run on single V100 and 7 P9 cores
- Evolve one "block" with **thornado**
  - Energy discretization: 32 points
  - Spatial discretization: $8^3$ points
- Performance data from Summit
  - PGI 19.9 (CPU and OpenACC)
  - XL 16.1.1-5 (OpenMP OL)
- ~7x speedup relative to threaded CPU code
Relaxation Test Problem

- Verification problem to test implicit operator
  - Run on single V100 and 7 P9 cores
- Evolve one “block” with *thornado*
  - Energy discretization: 32 points
  - Spatial discretization: $8^3$ points

- Performance data from Summit
  - NVHPC 21.3 (CPU and OpenACC)
  - XL 16.1.1-10 (OpenMP OL)
- ~20x speedup relative to threaded CPU code
V100 Roofline Analysis

- Generated hierarchical roofline model of bottleneck kernels in $O(v/c)$ Relaxation benchmark

- Key takeaways
  - thornado is memory (HBM) bound
  - Most kernels are near the roofline (good!)
  - L1 and L2 cache reuse is so-so
  - Need to think about how to increase arithmetic intensity (e.g., change vector length?)

- Need analysis on other systems for comparison
Final Thoughts

• Write “portable” CPU code (e.g., tight loop nests, libraries when possible)  
  – Will also be faster for CPU

• Challenging to port iteration kernels due to register pressure (e.g.,  
  Newton-Raphson, Bisection)  
  – Need to pack points and break up iteration into separate, smaller kernels

• Managing asynchronous kernels between libraries and directives is a challenge

• Need more compilers with robust Fortran+OpenMP 5.x support (worry about performance later)

• Looking forward to more comparisons with upcoming systems