Black Hole Simulation with CUDA

Accelerating HARM: A General Relativistic Magnetohydrodynamics Code

James Stevens
Praneet Sahgal
Shreyas Siravara
Bhargava Gopi Reddy

12/7/2014

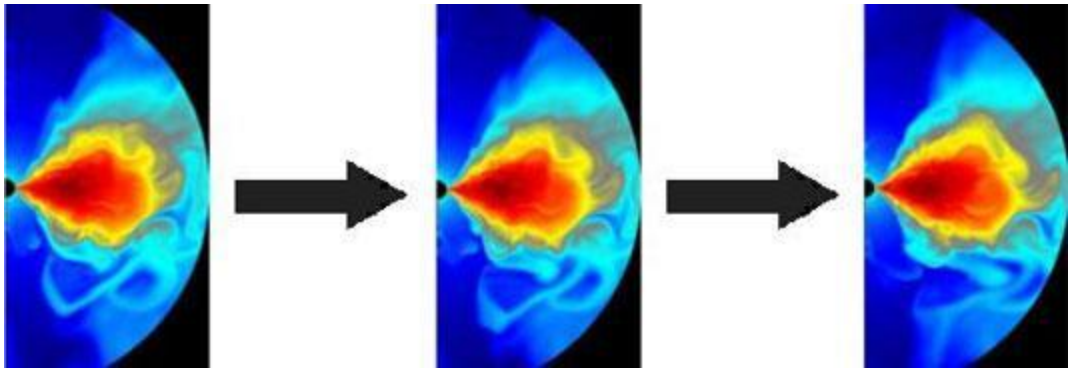
Problem Description

- Astrophysical Fluid Dynamics Group
  - In the Center for Theoretical Astrophysics at UIUC
  - Lead by Professor Charles F. Gammie
  - Numerical modeling
- Star formation
- Magnetohydrodynamics
- Accretion physics
  - Black hole simulation - HARM

Accretion disk
(Interstellar)

Problem Description

- HARM
  - 3-D general relativistic magnetohydrodynamics (GRMHD) code
  - Black hole simulation



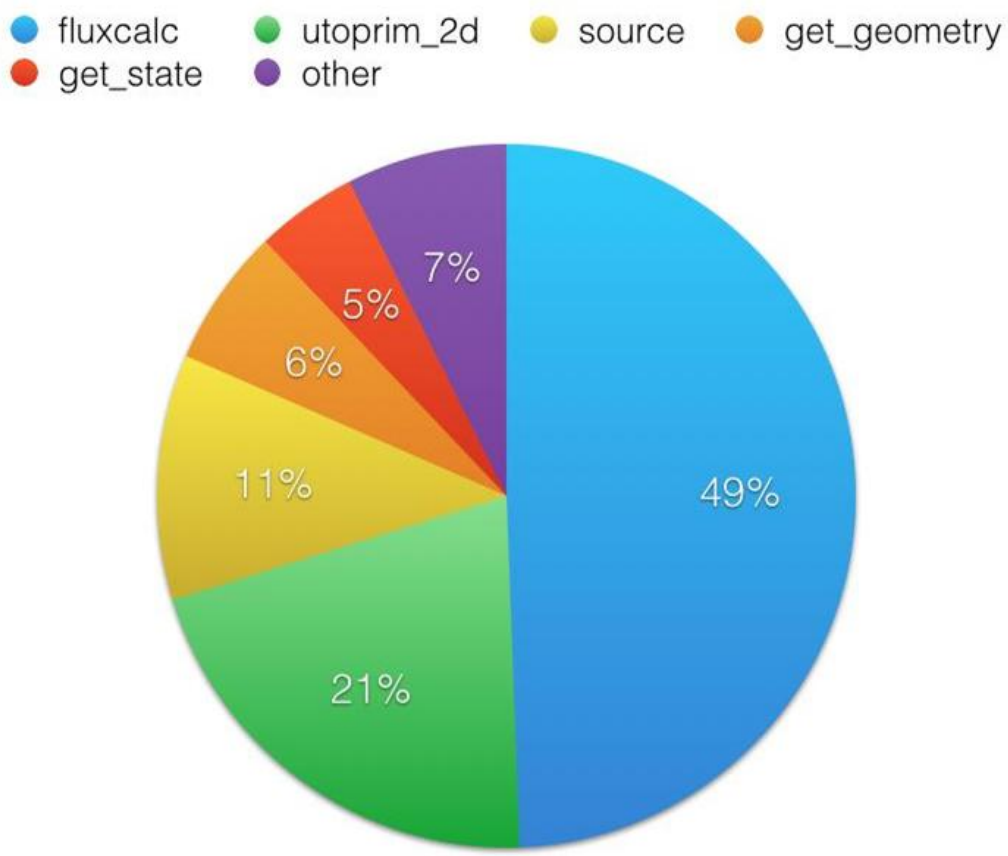
Problem Description

- HARM-2D
  - 2-D version of HARM
  - Fundamental tool for analysis before running HARM 3-D on supercomputer
    - Testing new physics or initial conditions
  - Predicts 3-D results well
    - Accretion disks exhibit strong axis-symmetry
  - Runs on desktop, but takes hours (runs overnight)
- Goal: accelerate HARM-2D
  - < 1 hour

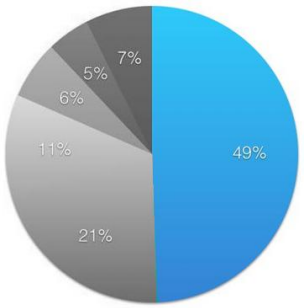
Profiling

- Perf
  - Linux tool for performance analysis
  - Supports hardware performance counters
  - Used to find initial targets for GPU acceleration

Profiling Results

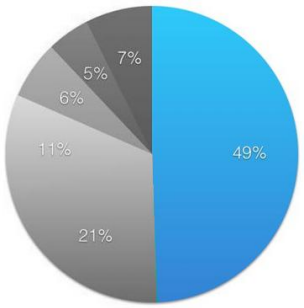


Serial Solution



- fluxcalc()
  - Calculates the amount of mass that moves across zone boundaries
  - Uses conservation of
    - rest-mass: $\frac{1}{\sqrt{-g}}\partial_\mu(\sqrt{-g}\,\rho u^\mu) = 0$
    - energy-momentum: $T^\mu_{v;\mu} = 0$
    - magnetic flux: $\partial_t\left(\sqrt{-g}\,B^i\right) = -\,\partial_j(\sqrt{-g}\left(b^j u^i - b^i u^j\right))$

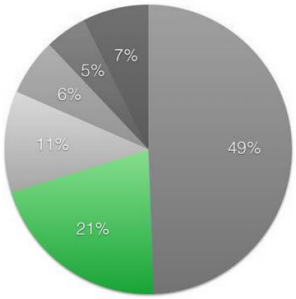
Serial Code



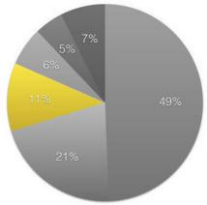
- fluxcalc()
  - Main flux calculation (double for-loop)
    - Includes loops within outer loops
  - Rescale operation (two double for-loops)
    - Also called in main flux calculation
  - Slope calculation (double for-loop)

Other Functions

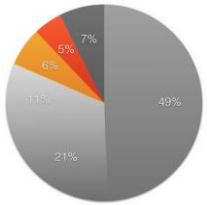
- UtoPrim_2d()
  - Utoprim_newbody ~ 18%
    - General Newton Raphson ~ 9.23%
      - Iterative algorithm
    - raise_g ~ 2.95%
    - Other ~1% each (mhd_calc, func_vsq, etc.)



- source()
  - A 4x4 loop
  - Multiple function calls for each dimension



- get_state and get_geometry
  - Can be accelerated like in fluxcalc()



Code Evaluation Summary

- Current work:
  - fluxcalc – easier to parallelize
- Future work: everything else
  - Fine grained function calls and small loops
  - Non-trivial to parallelize
  - Parallelize Newton method

HARM Parallelization Issues

- Multiple files (~30)
  - Complex source code (~8600 lines)
- Data Movement
  - Identifying the required arrays and scalars
    - Numerous variables/globals
    - Non-trivial due to complex code
- Loop in fluxcalc – but nested function calls

Parallelization Strategy

- Parallelize 2-D loops in fluxcalc()
  - Loops iterate over entire domain
  - Iterations are independent
  - 1 thread/iteration, 1 kernel/loop
  - 3 kernels, 2 device functions
  - Largest kernel: each thread performs 912+ operations

fluxcalc() serial

```
for(domain y-dim) {
 for(domain x-dim) {
  // 2 function calls
}}

// loop 2 (slope calc)

for(domain y-dim) {
 for(domain x-dim) {
  // 11 function calls
  // 912+ operations
}}

// repeat loop 1
```

fluxcalc() parallel

```
// copy data h->d
kernel1<<<>>>()
// copy data h->d

kernel2<<<>>>()
// copy data async h<-d
// copy data async h->d
// sync

kernel3<<<>>>()
// copy data async h<-d
// copy data async h->d
// sync

kernel1<<<>>>()
// copy data d<-h
```

Parallel Optimizations

- Long thin blocks for memory coalescence
  - N x 1 blocks, not square
  - ~10% reduction in kernel execution time
- Allocated GPU arrays once in init()
  - 2.5% reduction in total GPU execution time
  - May not be possible with larger runs
- Reduce computation by in-lining some GPU functions
  - Since some functions are general purpose, sometimes do more work than necessary
  - Could be optimized on CPU as well

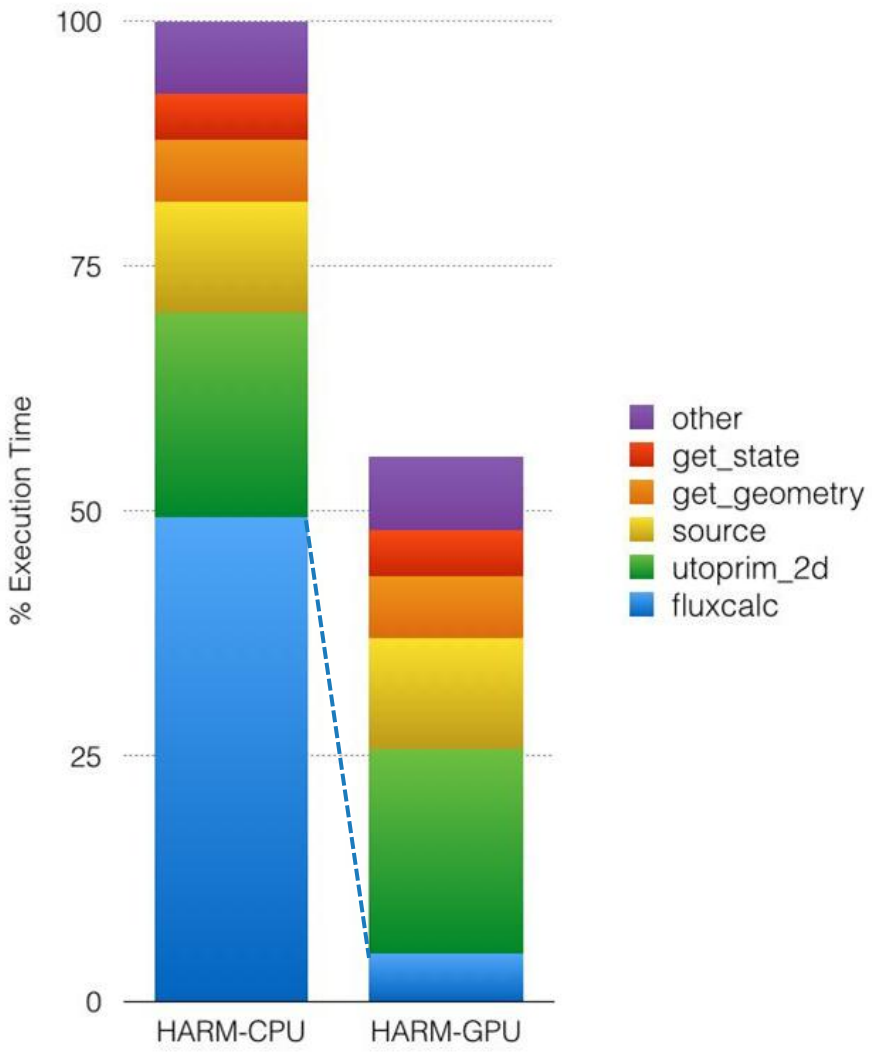
Parallel Optimizations Cont.

- Blocksize is a multiple of warp size (32)
- Constant variables in constant memory
- Minimize host-device memory transfers
  - Multiple kernels re-use the same data
  - Copy un-changing arrays once at beginning

Evaluation

Performance on GEM cluster

- ~12x speedup in fluxcalc
  - ~14x speedup in parallelized portions of fluxcalc
- ~44% reduction in total HARM runtime



Related Work

- Nvidia paper: Adaptive Mesh Astrophysical Fluid Simulations on GPU (2009)
  - Reported 10x speedup on entire application
  - NON-relativistic, HARM is a relativistic MHD code
    - Primitive variable evaluation is more expensive in HARM, so more runtime is spent outside of fluxcalc()
- Research at University of Tübingen
  - Relativistic MHD, simulating neutron star magnetic fields
    - 10x speedup double precision, 50 single precision
      - (quad core?)

Conclusion

- Achieved 14x speedup in parallelized code
  - Including all overhead and memory copies
- ~44% total HARM runtime reduction
- HARM is worth investigating further
  - Expect at least 10x overall speedup in HARM-2d
  - HARM-3d most likely amenable to GPU acceleration as well

Any Questions?

12/7/2014