Applications of GPUs to Binary Black Hole Evolutions

John Silberholz (josilber@umd.edu)

Department of Computer Science & Department of Math & Center for Scientific Computing And Mathematical Modeling, University of Maryland

CUDA Center for Excellence


Supported by NVIDIA, NASA, and NSF
Motivation

- Gravitational wave detectors
  - High sensitivity devices
  - High noise-to-useful-data ratio
  - Must understand how binary black holes (BBHs) interact to find them in the data
- Motivates research in the BBH phase space
Approach

• Current techniques
  • Full General Relativity
  • Post-Newtonian Approximations
    – Coupled ODEs solved w/ Monte Carlo simulation

• Goals
  • Begin studying BBH phase space
    – 7-dimensional space: Spin Magnitudes (2), Normalized Spin Directions (4), Mass ratio (1)
  • Implement PN equations quickly on the GPU
  • Use GPU cluster for largest GPU-based physics computation ever (that we know of)
GPU Implementation

- We have many initial configurations
- Each must to evolved to the final configuration
- Easy to implement in parallel
  - Each inspiral executed in parallel
  - Configs: CPU → GPU at inspiral execution
  - Kernel calls from CPU code
  - State maintained on GPU
  - Final state: GPU → CPU at inspiral termination
Pseudocode

// compute right hand side (rhs) of ODE
__global__ void ode_rhs(state)

// integrate the ODEs
void integrate(initial_state_and_params) {
    allocate_gpu_storage(...);
    while (all_omega>all_omega_final) {
        ode_rhs<<<nBlocks_rhs,nThreads>>>(...);
        checkCUDAError("first ode_rhs call");
        interm_1<<<nBlocks_interm,nThreads>>>(...);
        ... // more intermediate values & rhs calls
        adjust_timesteps(...);
    }
    transfer_from_gpu_to_host(...);
}
Memory Access

- State information stored contiguously in global memory
- Each RHS call, load state into thread local memory
- At end of RHS call, store back in global memory
- Due to locality, all accesses coalesced
Precision

- CPU code in double precision
  - Low performance hit
  - Validated against other physics codes
  - We can use this code to validate any GPU code
- GPU code in single precision
  - Double precision performance losses too significant
  - (At least with current GPUs)
Validation

- Each GPU run:
  - Run all inspirals on GPU
  - In separate CPU thread, run a fraction of the inspirals in double precision
- After the run:
  - Validate the GPU results against the CPU results
- Results:
  - (Small) error acceptable for our purposes
Performance

[Graph showing the relationship between speed (inspirals/ms) and the number of inspirals (N).]
Performance

- GPU code on NVIDIA Tesla S1070 (1035 Gflop, single precision / GPU)
- CPU code on Intel Xeon E5410 (~5 Gflop, double precision / core)
- Theoretical maximum speedup of 200x
Performance: Speedup

- Run many instances of same inspiral parameters
  - Increases reproducibility
  - Removes random number generation overhead
  - No divergent warps on GPU (but less than 10% improvement)

- Speedup of 50x
  - One GPU vs. one CPU core
Some results

- Equal mass, maximally spinning black holes
- High correlation between initial and final dot product
Some results

- Uneven mass, low spin
- We find antialignment between the initial and final dot product
Limitations

- Each run is expensive
  - 200 million inspirals per 2-D parameter config.
- We need many configurations
  - Even with 30 points sampled in each dimension, we need 900 configurations
  - In reality, we need many more
- For small runs, we use 2 Tesla S1070 units
  - ~1.7 billion inspirals / day
  - > 100 days in small example above
  - For large runs, lifetimes
Solution

- We use many GPUs
- We use Lincoln cluster
  - Operated by NCSA
  - Has 96 Tesla units and 192 8-core scalar units
  - Decreases short run to ~2 days
  - Difference between doing runs and not doing runs
Using MPI

• One node acts as a lead
  • Keeps track of outstanding jobs
  • Instructs nodes of ID of next job
  • Reassign long-running jobs
• Other nodes actually do computations
  • Calculate configs from job ID
  • Do the run
  • Save results to tape storage
  • Save an indicator file to tape storage
• Instruct lead that job is done; ask for more work
Getting the data

- Anonymous ssh, (HPN) scp from our machines
- Every 10 seconds, look for indicator files
  - When found, remove indicator and get data with scp
- scp failures
  - Common, because many jobs finish at once
- Increased scp robustness
  - Get and store md5 checksum
  - Only 5 scp requests at once; rerun failures
Larger runs

• Many runs needed for full analysis:
  • \(~80\) trillion inspirals
  • \(~2.8\) PB in raw data
  • \(~40\) days of full cluster-time

• Will give best-ever understanding of BBH phase space

• Will be largest GPU-based physics computation ever (to our knowledge)
Conclusions

- Phase space results useful
  - Predict behavior given current configuration
  - Already, much new knowledge
- GPUs speed up BBH computations
  - 50x speedup obtained
- GPU clusters important
  - Provide critical scaling
  - Difficulties in getting data