N-Body Simulation GPU Implementation Report

Chin-Hui Chu
r12944041

1. Implementation Details

1.1 Parallelism Strategy

Multi-level parallelism:

- **CPU-Level**: 2 OpenMP threads (hw5.cpp:461-611)
  - GPU 0: Problem 1 (minimum distance)
  - GPU 1: Problem 2 (collision detection + checkpointing)
  - Run simultaneously
- **GPU-Level**:
  - Kernel parallelism: one thread per body
  - Dynamic block size (256 threads, hw5.cpp:433-450)
  - 4 HIP streams per GPU for Problem 3 (hw5.cpp:619)
- **Problem 3**: Both GPUs test devices concurrently
  - Interleaved distribution: GPU 0→devices 0,2,4,...; GPU 1→devices 1,3,5,...
  - 8 concurrent device tests (2 GPUs × 4 streams)

1.2 Optimization Techniques

Kernel Fusion:

- Forces + velocities (hw5.cpp:117-147): eliminates intermediate acceleration writes
- Positions + distance tracking (hw5.cpp:273-295): reduces kernel launch overhead

Memory Optimizations:

- Shared memory tiling with padding (`tile_size + 1`) to avoid bank conflicts (hw5.cpp:149-230)
- Structure-of-arrays layout for coalesced access

Control Flow:

- Warp divergence elimination via masking instead of branches (hw5.cpp:199-217)
- Loop unrolling (`#pragma unroll 8` , hw5.cpp:195)

Algorithmic:

- Checkpointing (hw5.cpp:583-595): save state when missiles hit devices
- Atomic operations for min distance tracking (hw5.cpp:245-253)
- Stream concurrency: overlap memory transfers and computation (hw5.cpp:619-760)

1.3 Two-GPU Management

Workload:

```
GPU 0: Problem 1 | Problem 3 (devices 0,2,4,...)
GPU 1: Problem 2 | Problem 3 (devices 1,3,5,...)
```

- OpenMP critical sections for result updates (hw5.cpp:601-607, 762-768)
- No inter-GPU synchronization needed

2. Scaling to 4 GPUs

Strategy: Maximize Problem 3 Throughput

Workload Distribution:

```
GPU 0: Problem 1          → Then Problem 3 (devices 0,4,8,...)
GPU 1: Problem 2          → Then Problem 3 (devices 1,5,9,...)
GPU 2: Idle (or assist P1) → Problem 3 (devices 2,6,10,...)
GPU 3: Idle (or assist P1) → Problem 3 (devices 3,7,11,...)
```

Implementation:

- Use 4 OpenMP threads for Problem 3
- Round-robin device distribution: GPU i tests devices `i, i+4, i+8, ...`
- Increase `NUM_STREAMS` to 8-16 per GPU
- **Expected speedup:** ~2× for Problem 3

Key Code Change:

```
#pragma omp parallel num_threads(4)  // Change from 2 to 4
{
    int gpu_id = omp_get_thread_num();
```

```
    // Distribute: for (idx = gpu_id; idx < devices.size(); idx += 4)
}
```

Potential Bottlenecks:

- PCIe bandwidth if GPUs on different switches
- Small problem sizes may not saturate 4 GPUs

3. Testing 5 Gravity Devices

Answer: No, independent simulations are NOT necessary.

Checkpoint-Based Approach (hw5.cpp:537-597, 613-769):

Phase 1: Create Checkpoints

- Run 1 simulation for 200,000 steps
- When missile hits device i at time t_i, save checkpoint (positions, velocities, masses)
- Result: 5 checkpoints

Phase 2: Test Devices Concurrently

- For each device: restore from checkpoint, set device mass to 0, continue simulation
- 2 GPUs × 4 streams = **8 concurrent tests** → all 5 devices tested simultaneously

Efficiency:

- **Naive**: 5 × 200,000 steps = 1,000,000 total steps
- **Checkpoint**: 200,000 + Σ(200,000 - t_i) ≈ 300,000-400,000 steps (**2.5-3× faster**)
- **Memory**: 5 checkpoints × 56n bytes = 280 KB for n=100 (negligible)
- **Wall-clock time**: Dominated by longest partial simulation (with 8-way parallelism)

When independent simulations would be needed:

- Different initial conditions per test (not the case here)
- Testing device combinations (problem only tests single devices)
- Non-deterministic simulations (not applicable)