

HW5 Report: N-Body Simulation GPU Acceleration

1. Implementation

Parallelism Strategy

Thread-level parallelism: Each GPU thread handles one body. All bodies compute their accelerations in parallel, then update positions synchronously using `__syncthreads()`.

Task-level parallelism (Problem 3): Testing different gravity devices are independent tasks. We distribute these across:

- 2 GPUs (via OpenMP threads)
- 4 HIP streams per GPU (concurrent kernel execution)

This gives us 8 concurrent device simulations.

Optimization Techniques

Technique	Benefit
Mega-kernel	Eliminates 200k kernel launches → 1 launch per simulation
Shared memory	Body positions cached in fast on-chip memory (reduce global memory bandwidth)
HIP streams	Overlap memory copies with kernel execution
Early exit	Break simulation loop once collision is detected

2-GPU Resource Management

```
#pragma omp parallel num_threads(2)
{
    int gpu_id = omp_get_thread_num();
    hipSetDevice(gpu_id);

    // Each GPU gets its own:
    // - Device memory allocations
    // - 4 HIP streams
    // - Subset of devices to test (interleaved: GPU0→0,2,4,...
    GPU1→1,3,5,...)
}
```

Each OpenMP thread binds to one GPU and manages its resources independently. Results are merged on CPU after all GPUs finish.

2. Scaling to 4 GPUs

Two simple changes:

1. **Increase OpenMP threads:** `num_threads(4)` and set `num_gpus = 4`
2. **Keep interleaved distribution:** GPU_i tests devices {i, i+4, i+8, ...}

Expected speedup: ~2x over 2 GPUs (linear scaling since device tests are independent).

Additional optimization: Could also increase streams per GPU if GPU utilization is low, but 4 streams is likely sufficient.

3. Do We Need 5 Independent Simulations?

Yes, currently we do — but there's room for optimization.

Why 5 simulations? Each device destruction changes the gravitational dynamics differently. The asteroid's trajectory diverges based on which device is destroyed and when.

Potential optimization — Shared prefix computation:

- All simulations are identical until the first missile hits any device
- We could simulate once up to `min(hit_step)` across all devices, then branch
- However, hit times depend on device positions, which change during simulation
- This makes prefix sharing complex to implement correctly

Practical answer: For this homework's scale (≤ 1024 bodies, 200k steps), running 5 independent simulations with 8-way parallelism (2 GPUs \times 4 streams) is fast enough. The complexity of prefix optimization isn't worth it.