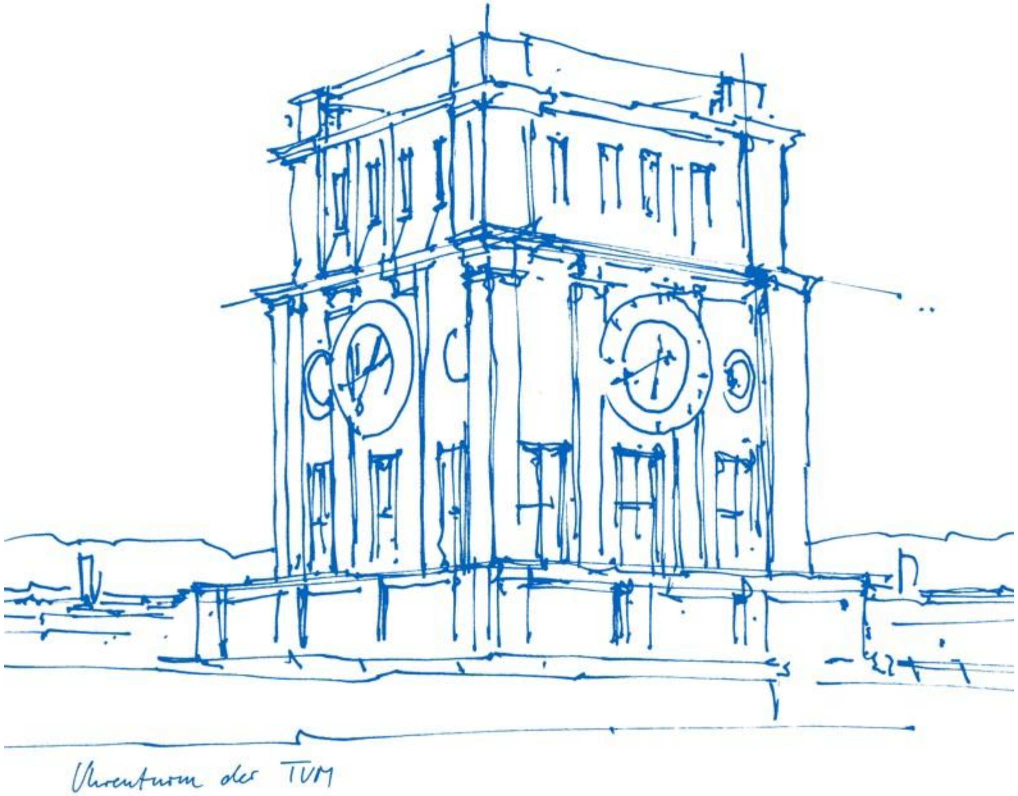
Department of Physics
Technical University of Munich

N-Body Simulations

(Seminar: Parallelization of Physics calculations on GPUs with CUDA)



Ashish Darekar
ashish.darekar@tum.de

Shubham Khatri
shubham.khatri@tum.de

Technische Universität München
Fakultät für Physik

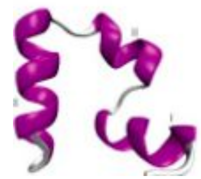
25th June, 2020

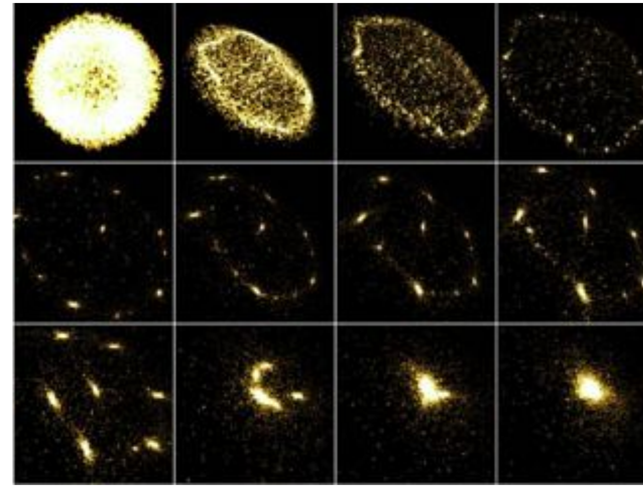
Overview

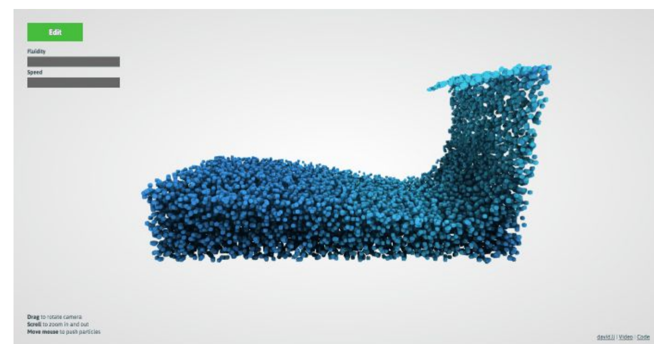
- Introduction: N-Body Simulations
- Particle-Particle(PP) Method - $O(N^2)$
  Features - Force - Time Integration - Initialization
- CPU Implementation
- Improvements step I : OpenMP Parallelization
- Improvements step II : CUDA Implementation
- Performance results :
  CPU: Serial and Serial with openMP
  GPU: Variation in Block size - Comparison of GPU Arch. - Benchmark
- Visualisation of results - Disc of particles
- Conclusion and Improvements

Introduction : N-body simulations

“Time evolution of a system of bodies in which each body continuously interacts with every other body”







Examples:

1. Astrophysical simulation
2. Protein folding
3. Coulomb forces exerted by the atoms in a molecule
4. Turbulent fluid flow simulation

Mathematically:

$$U(\mathbf{x}_0) = \sum_i F(\mathbf{x}_0, \mathbf{x}_i)$$

References:

https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda

Introduction : N-body simulations

Different Algorithms:

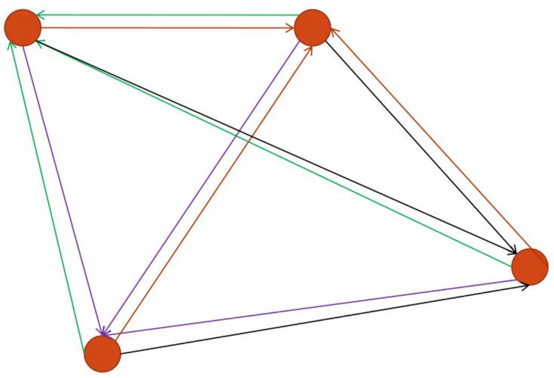
1. **The Particle-Particle (PP) method - $O(n^2)$**
2. The Barnes-Hut algorithm - O(n log n)
3. The Particle-Mesh (PM) method - O(n)
4. The Particle-Particle/Particle-Mesh algorithm (P3 M) - O(n)
5. Fast Multipole method (FMM) - O(n)
6. Other methods: Hybrid Methods, Self-Consistent Field (SCF) method, Symplectic method. etc.

References:
Amara Grap's excellent web page on N-body algorithms, internet, http://www.amara.com/papers/nbody.html

The Particle-Particle (PP) method - $O(n^2)$

Particle-Particle Method - Features

- Brute force technique
- Evaluate all pair-wise interactions
- $O(n^2)$ computational complexity
- $O(n^2)$ memory requirement
- Most accurate values

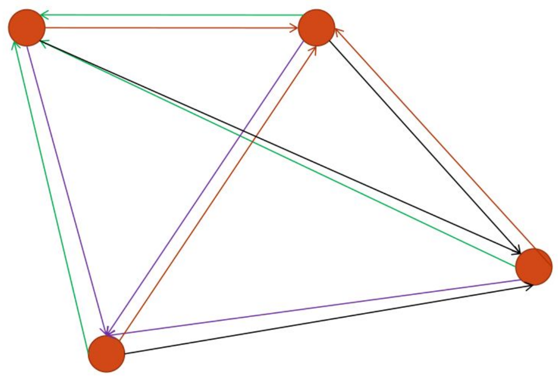


Simple N-Body scenario with 4 bodies

Particle-Particle Method - Force

- Force vector f_{ij} on body i due to j

$$\mathbf{f}_{ij} = G\frac{m_i m_j}{\left\|\mathbf{r}_{ij}\right\|^2} \cdot \frac{\mathbf{r}_{ij}}{\left\|\mathbf{r}_{ij}\right\|},$$



Simple N-Body scenario with 4 bodies

- Total Force on body i

$$\mathbf{F}_i = \sum_{\substack{1 \le j \le N \\ j \ne i}} \mathbf{f}_{ij} = Gm_i \cdot \sum_{\substack{1 \le j \le N \\ j \ne i}} \frac{m_j \mathbf{r}_{ij}}{\left\|\mathbf{r}_{ij}\right\|^3}.$$

- Addition of Softening factor:

$$\mathbf{F}_i \approx Gm_i \cdot \sum_{1 \le j \le N} \frac{m_j \mathbf{r}_{ij}}{\left(\left\|\mathbf{r}_{ij}\right\|^2 + \varepsilon^2\right)^{3/2}}.$$

Particle-Particle Method - Time Integration

- Acceleration:

$$\mathbf{a}_i \approx G \cdot \sum_{1 \le j \le N} \frac{m_j \mathbf{r}_{ij}}{\left(\left\|\mathbf{r}_{ij}\right\|^2 + \varepsilon^2\right)^{3/2}}.$$

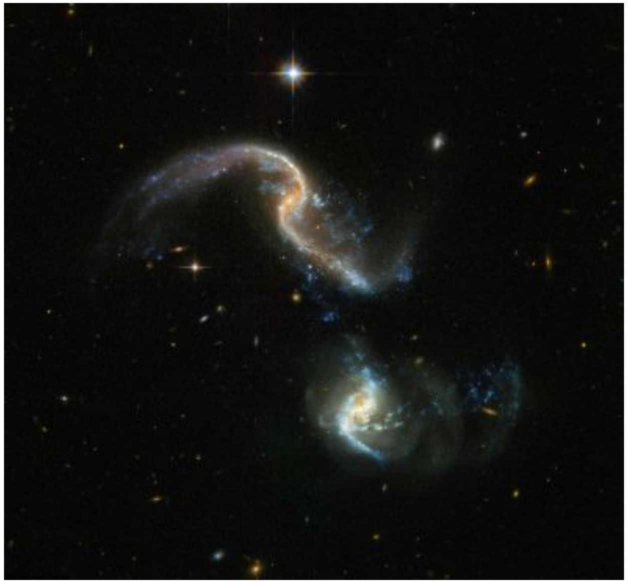
- Position and Velocity update - Euler Scheme:

$$\mathbf{v}(t + \Delta t) = \mathbf{v}(t) + \mathbf{a}(t)\Delta t$$

$$\mathbf{x}(t + \Delta t) = \mathbf{v}(t + \Delta t)\Delta t$$

Particle-Particle Method - Initialization

- Plummer model for spherical galaxy
- Simple disk galaxy : Disc of Particles
- Two galaxies: colliding disk galaxy



Particle-Particle Method - Initialization

- Mass: Heavy Central Mass
- Center Particle Position

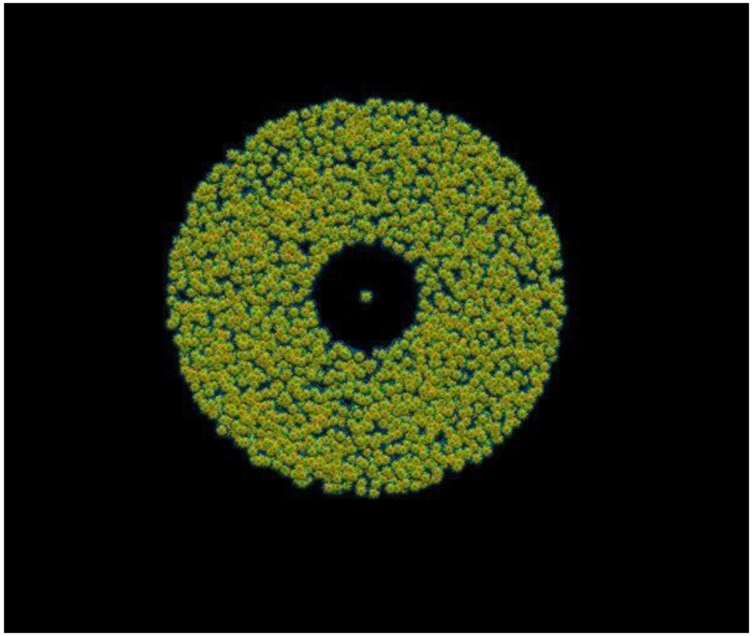
$$P_C = \{0, 0, 0\}$$

- Disc Particle Position

$$P_{D_i} = \{R_i \cdot \cos\theta_{rand}, R_i \cdot \sin\theta_{rand}, T_i\}$$

$$T_i \in [T_{min}, T_{max}]$$

- Velocity of Particles

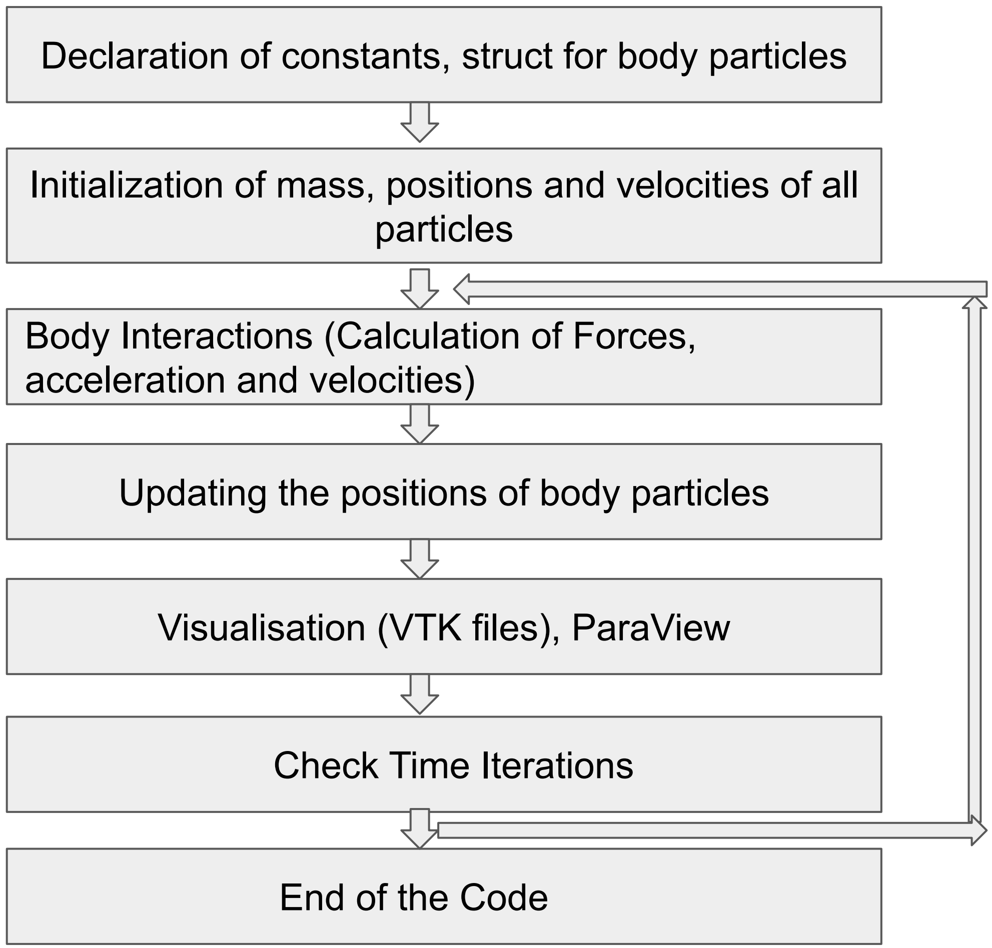


N-Body scenario with 4096 particles
(Disc of particles)

Serial Implementation

CPU Implementation

Algorithm



OpenMP Implementation

Step I: Parallelization - OpenMP

- Position update step independent.
- Position and Velocity Initialization Independent.
- Parallelize independent blocks.
- SIMD and Static schedule for performance optimization.
- Fan in reductions to increase parallelization

CUDA Implementation

Step II: Parallelization - CUDA

- Same data dependency
- Semantically same implementation
- Hypothetical Parallelism of O (N^2) is possible - N *x* N Grid
  - Memory and Bandwidth Restriction
- Two kernels:
  - Particle Initialization
  - Position Update

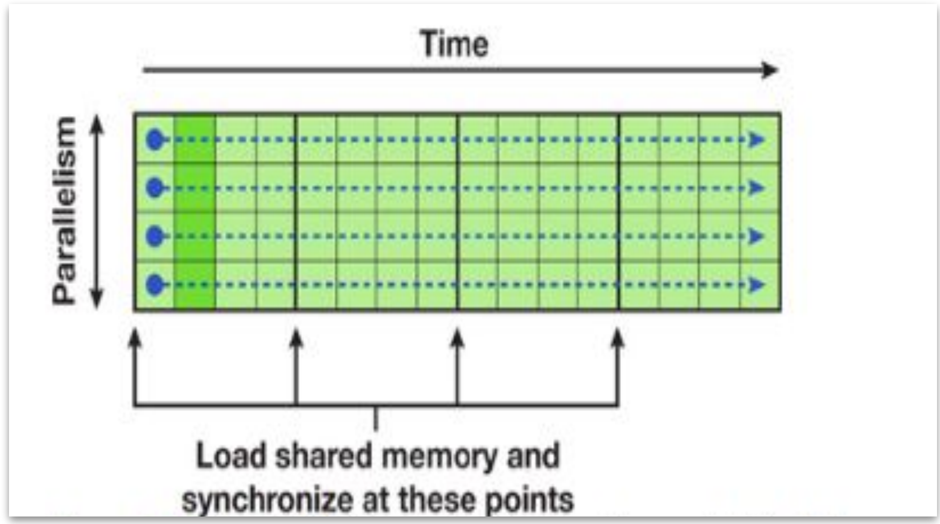
Advantages:

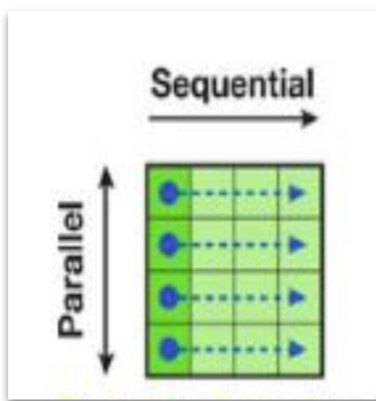
- Initialisation on GPU - saving memcpy(HostToDevice)
- Highly parallelized initialization and computation

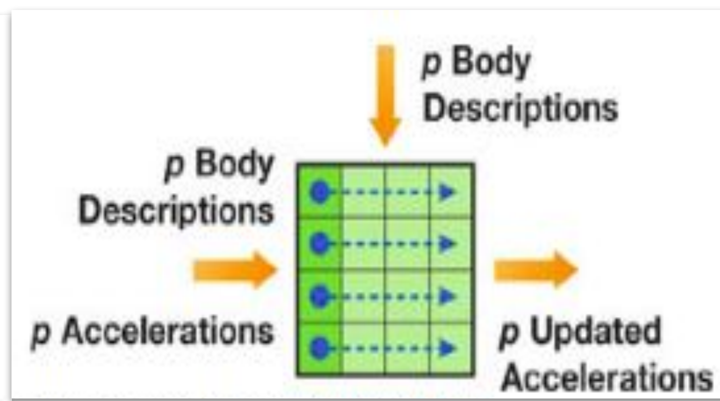
Step II: Parallelization - CUDA

Further consideration:

- Tiling
  - Computational Tile - p x p
  - Sequential execution of tiles.
  - Better cache use



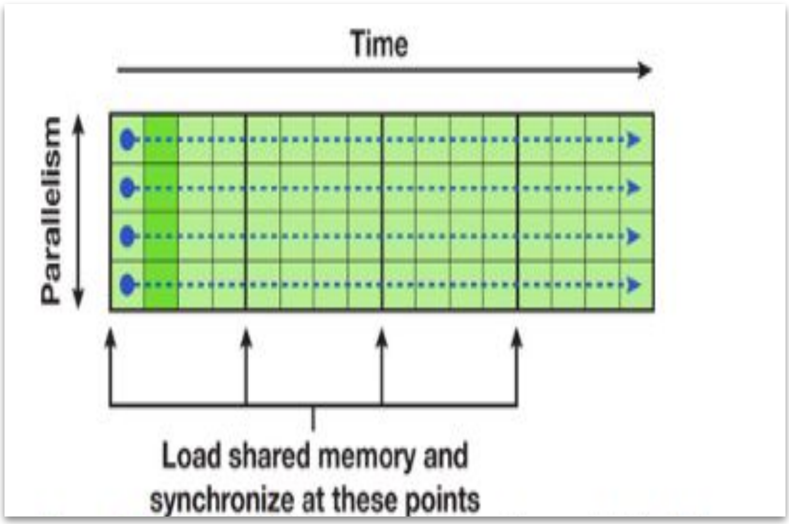


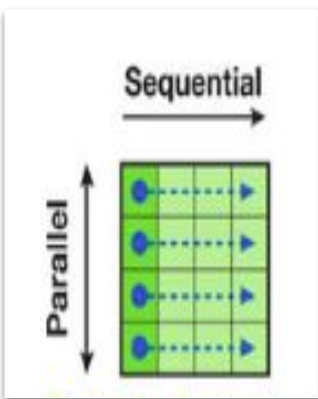


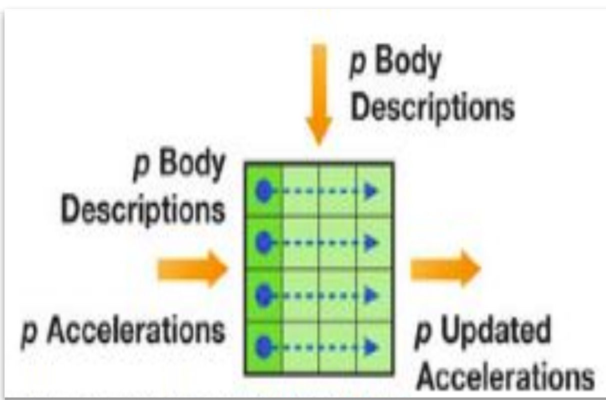
References:

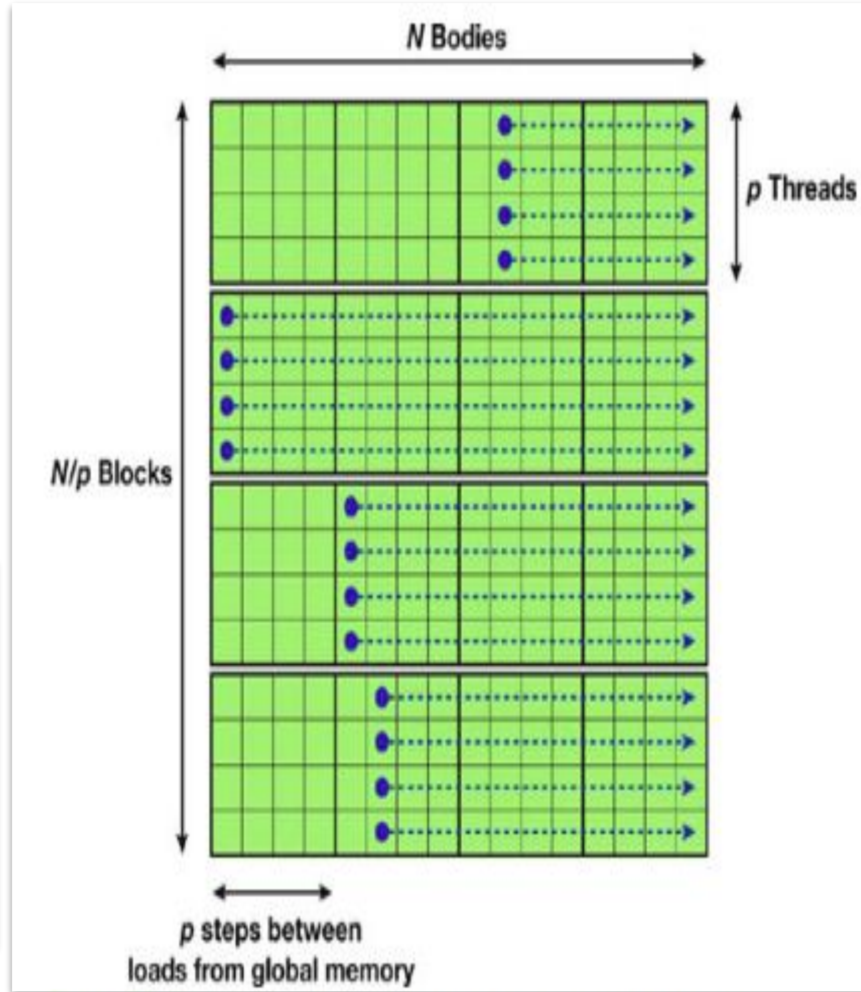
https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda

Step II: Parallelization - CUDA









References:

https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda

Step II: Parallelization - CUDA

Further consideration:

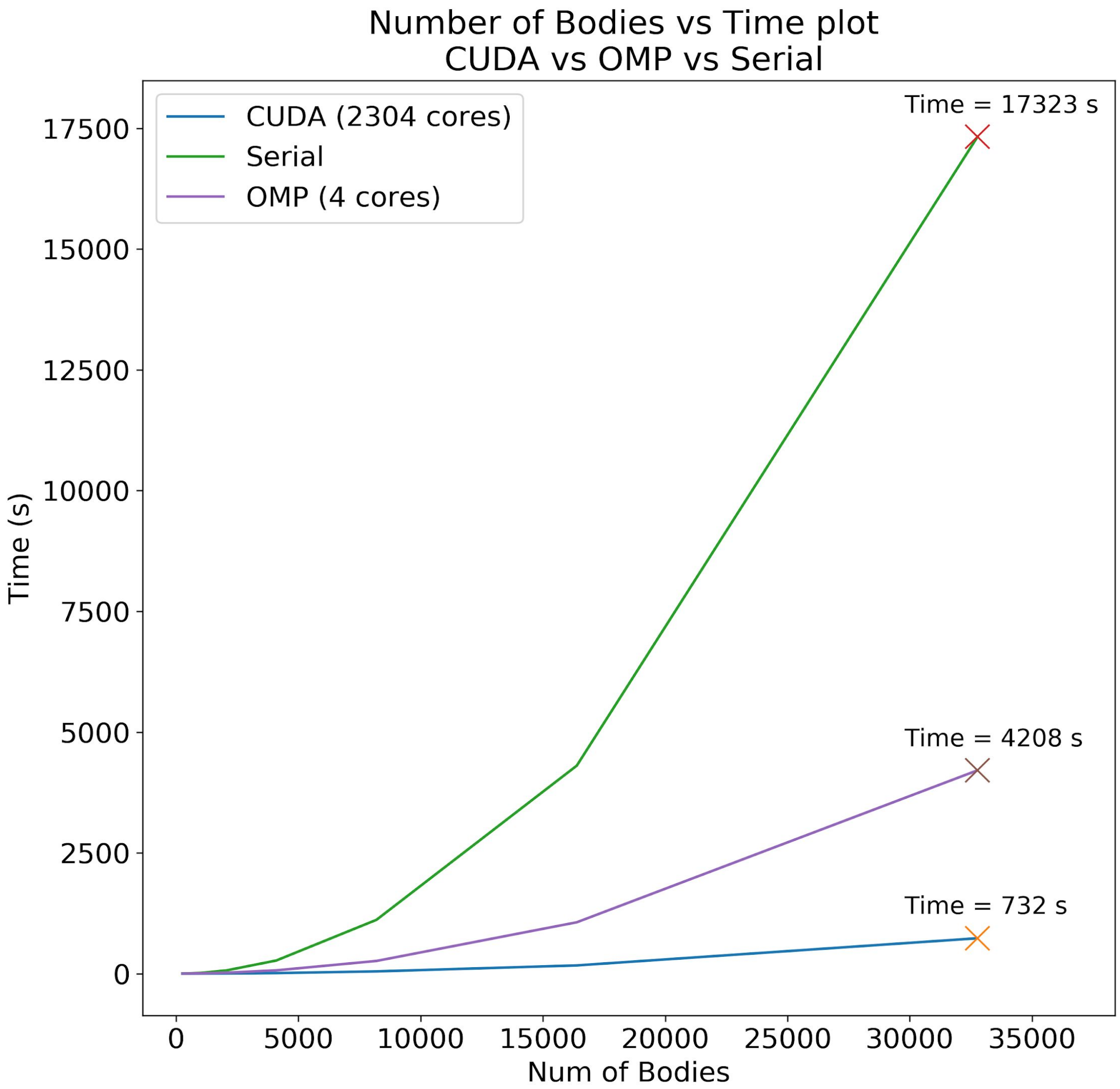
- Usage of float4
  - Coalesced memory access
  - Memory Alignment for better caching
- Loop Unrolling
  - Better thread scheduling.
  - Reduced loop control overhead.

References:

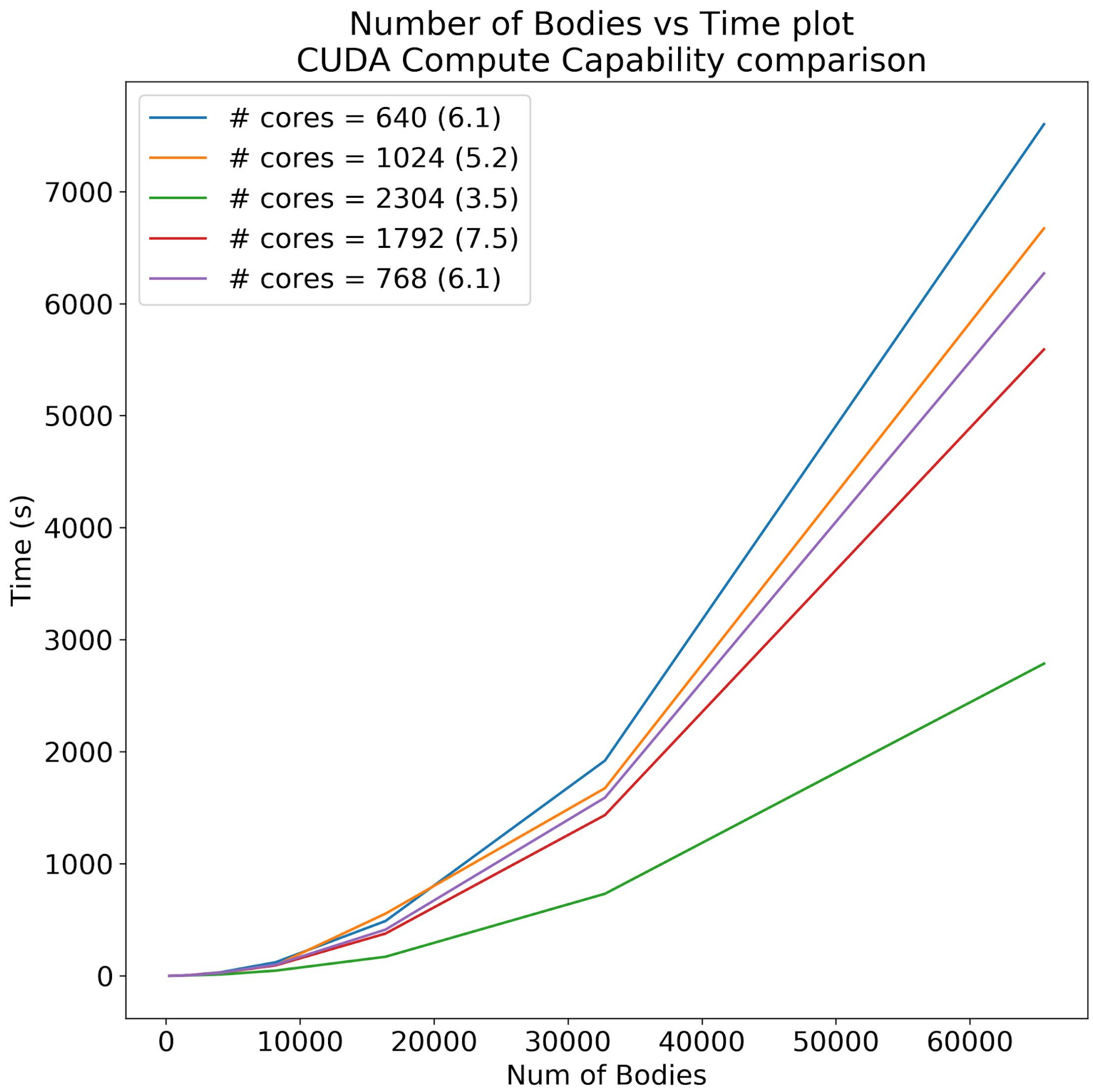
https://developer.nvidia.com/blog/how-access-global-memory-efficiently-cuda-c-kernels/

Performance Results:

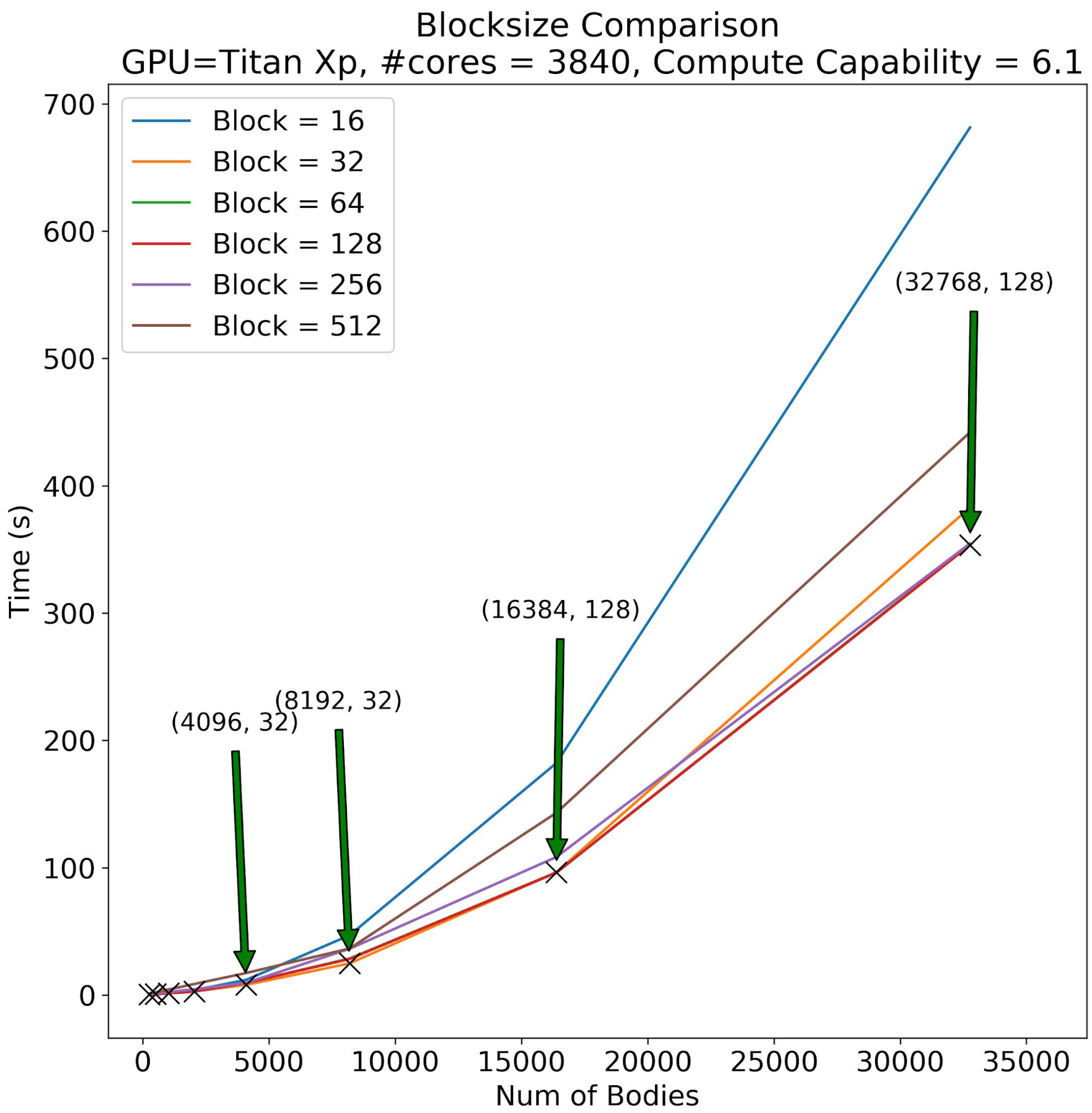
Sequential vs OpenMP vs CUDA



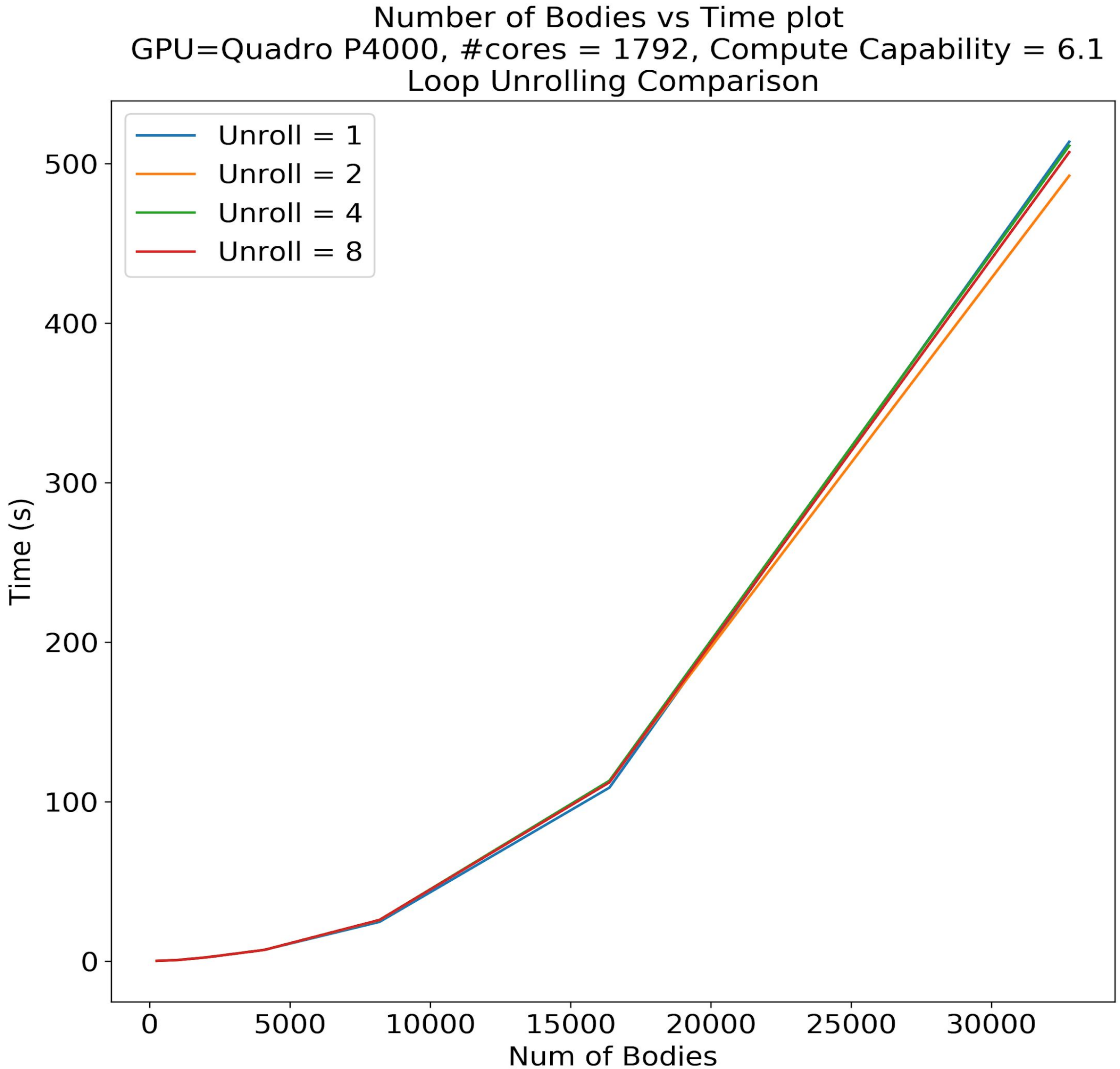
CUDA - Architecture Comparison



CUDA - Block Size Comparison



CUDA - Loop Unrolling



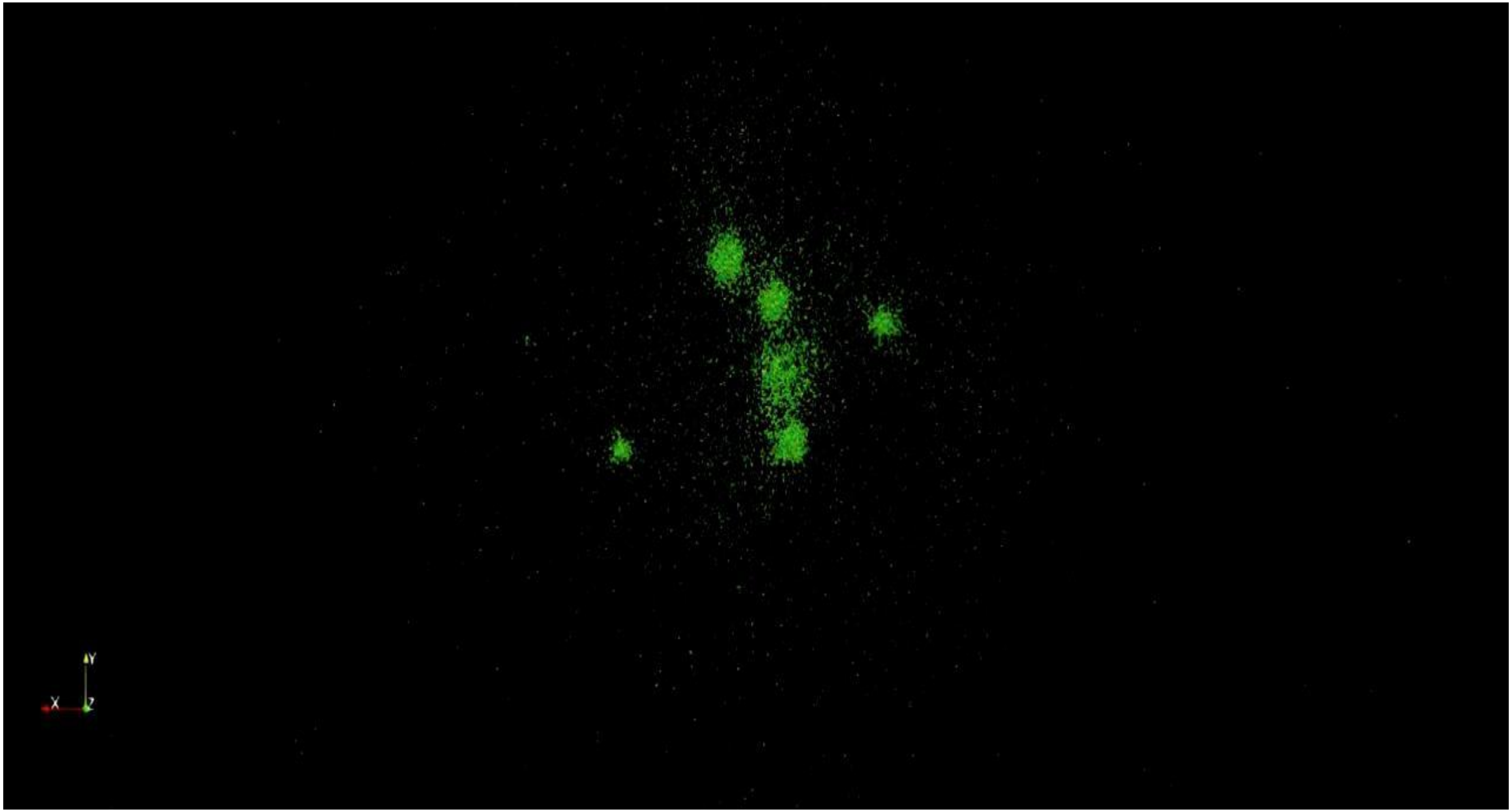
Benchmarking with NVIDIA Code

Hardware and parameters :

1. Machine: GeForce GTX 1050 (640 Cores)
2. Compute Capability: 6.1
3. Problem Size: 5120
4. Iterations : 10

	GFlops / s	**Runtime (ms)**
Our Implementation (25 Flops / interaction)	51	129.59
Nvidia (20 Flops / interaction)	1181	4.438

Visualization Results : One Particle at center

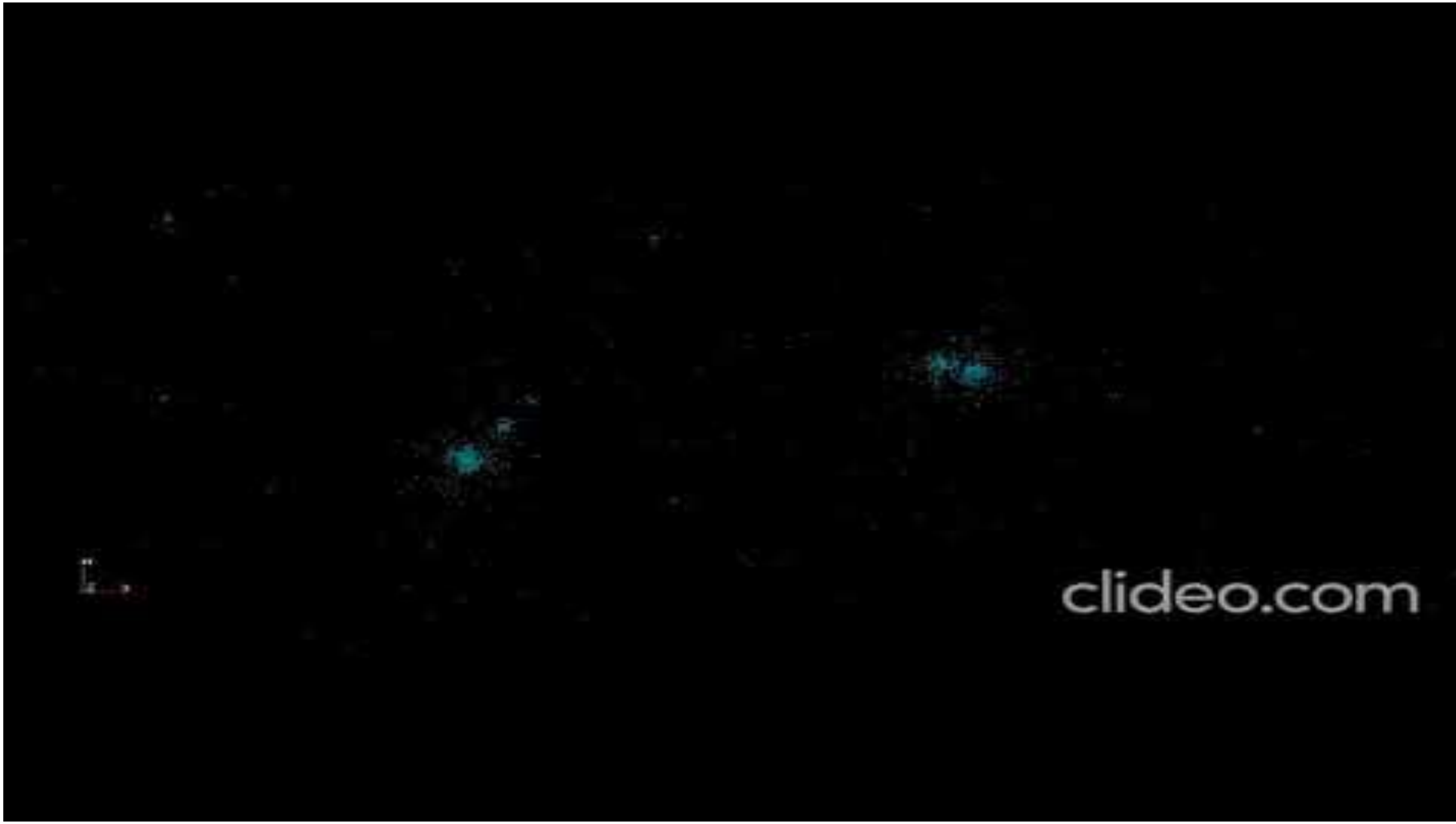


ParaView

https://www.youtube.com/watch?v=_4M3n7vfURM

Disc of particles - 1 particle case -
16384 Particles, 20K Iterations

Visualization Results : Two Particles at center



https://youtu.be/tEhh-Lp2biE

Disc of particles - 2 particles case - 32768 Particles, 20K Iterations

ParaView

Conclusion and Improvements

- PP method has computational complexity of O (N^2)
- Easily parallelizable using openMP, MPI or CUDA
- Performance gains through:
  - Tiling strategy (Block Size)
  - Data types - float3 and float4
  - Loop unroll
- Tree Methods like Barnes-Hut and fast Multipole Method can give better results. (O(nlogn))

References

[1] CUDA N-Body simulatoins:

https://developer.nvidia.com/gpugems/gpugems3/part-v-physics-simulation/chapter-31-fast-n-body-simulation-cuda

[2] Amara Grap's excellent web page on N-body algorithms, internet,

http://www.amara.com/papers/nbody.html

[3] Implementation of kernel

https://stackoverflow.com/questions/18501081/generating-random-number-within-cuda-kernel-in-a-varying-range

https://developer.nvidia.com/blog/easy-introduction-cuda-c-and-c/

[4] Programming Guide CUDA - NVIDIA

https://docs.nvidia.com/cuda/cuda-c-programming-guide/index.html

https://developer.nvidia.com/blog/how-access-global-memory-efficiently-cuda-c-kernels/

[5] CUDA Basics

https://www.nvidia.de/docs/IO/116711/sc11-cuda-c-basics.pdf

Questions?