N-body memory layout exploration

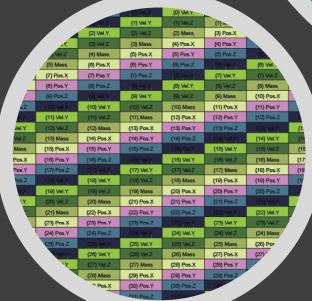
Bernhard Manfred Gruber CERN, CASUS

N-body simulations

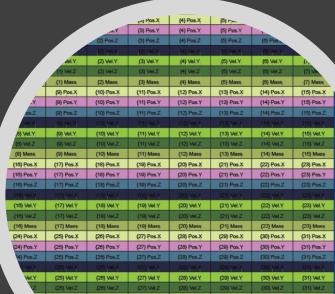
- Used to simulate interaction of many bodies
 - Galaxies, stars, atoms, electrons, proteins, ...
 - Gravity, electromagnetic forces, van der Waals forces, ...
- https://www.youtube.com/results?search_query=nbody+si mulation
- N² computational complexity
 - There are faster methods and acceleration data structures
 - E.g. using Octrees and approx. groups of bodies
 - But they still use the N² base algorithm inside

Objective

- Given 3 n-body CPU implementations
 - ~430 LOCs C++
 - with different memory layouts
 - AoS (Array of Structs)
 - SoA (Structs of Arrays)
 - AoSoA (Array of Structs of Arrays)
- Find the fastest possible CUDA implementation
 - by finding the fastest memory layout for global and shared memory
 - without functional changes



25.X	{1} Pos.X	{2} Pos.X	{3} Pos.X	{4} Pos.X	(5) Pos.X	{6} Pos.X	{7} Pos.X
Pos.X	{9} Pos.X	{10} Pos.X	(11) Pos.X	(12) Pos.X	{13} Pos.X	(14) Pos.X	(15) Pos.X
6) Pos.X	{17} Pos.X	{18} Pos.X	(19) Pos.X	(20) Pos.X	{21} Pos.X	(22) Pos.X	(23) Pos.X
(24) Pos.X	{25} Pos.X	{26} Pos.X	{27} Pos.X	(28) Pos.X	{29} Pos.X	(30) Pos.X	(31) Pos.X
{0} Pos.Y	{1} Pos.Y	{2} Pos.Y	{3} Pos.Y	{4} Pos.Y	(5) Pos.Y	(6) Pos.Y	{7} Pos.Y
{8} Pos.Y	{9} Pos.Y	{10} Pos.Y	{11} Pos.Y	{12} Pos.Y	{13} Pos.Y	{14} Pos.Y	{15} Pos.Y
{16} Pos.Y	{17} Pos.Y	{18} Pos.Y	(19) Pos.Y	(20) Pos.Y	{21} Pos.Y	{22} Pos.Y	{23} Pos.Y
{24} Pos.Y	{25} Pos.Y	{26} Pos.Y	{27} Pos.Y	(28) Pos.Y	{29} Pos.Y	(30) Pos.Y	(31) Pos.Y
{0} Pos.Z	{1} Pos.Z	{2} Pos.Z	{3} Pos.Z	{4} Pos.Z	(5) Pos.Z	(8) Pos.Z	{7} Pos.Z
(8) Pos.Z	{9} Pos.Z	{10} Pos.Z	{11} Pos.Z	(12) Pos Z	{13} Pos.Z	{14} Pos.Z	{15} Pos.Z
(16) Pos.Z	{17} Pos.Z	(18) Pos.Z	(19) Pos.Z	{20} Pos.Z	(21) Pos.Z	{22} Pos.Z	(23) Pos.Z
{24} Pos.Z	{25} Pos.Z	{26} Pos.Z	{27} Pos.Z	(28) Pos.Z	{29} Pos.Z	(30) Pos.Z	(31) Pos.Z
(8) VIII X							
(10) Vet.X							
241 (VEX)							
Vel.Y	{1} Vel.Y	{2} Vel.Y	{3} Vel.Y	{4} Vel.Y	(5) Vel.Y	(6) Vel.Y	{7} Vel.Y
fel.Y	{9} Vel.Y	{10} Vel.Y	{11} Vel.Y	{12} Vel.Y	{13} Vel.Y	{14} Vel.Y	{15} Vel.Y
(X	{17} VeLY	{18} Vel.Y	{19} Vel.Y	{20} Vel.Y	{21} Vel.Y	{22} Vel.Y	{23} Vel.Y
	{25} Vel.Y	{26} Vel.Y	{27} Vel.Y	(28) Vel.Y	(29) Vel.Y	{30} Vel.Y	(31) Vel.Y
	{1} Vel.Z	{2} Vel.Z	(3) Vel.Z	(4) Vel Z	(5) Vel.Z	(6) Vel.Z	(7) Vel.Z
	(9) Vel.Z	(10) Vel.Z	{11} Vel.Z	(12) Vel.Z	(13) Vel.Z	{14} Vel.Z	(15) Vel.Z
	7 Vel.Z	(18) Vel.Z	{19} Vel.Z	{20} Vel.Z	(21) Vel.Z	{22} Vel.Z	(23) Vel.Z
	7	(26) Vel.Z	{27} Vel.Z	{28} Vel.Z	{29} Vel.Z	{30} Vel.Z	{31} Vel.7
		{2} Mass	{3} Mass	{4} Mass	(5) Mass	(6) Mass	177
		વ્ય Mass	{11} Mass	{12} Mass	{13} Mass	{14} Mass	
10)			(19) Mass	(20) Mass	{21} Mass	{22} **	



Tasks

- 1. Start by cloning: https://github.com/bernhardmgruber/nbody-memory-layout-student-project
- 2. Compile/run sample CPU code. Compare runtime of different memory layouts
- 3. Inspect disassembly and understand CPU vectorization and access pattern: https://godbolt.org/z/ligo3z
- 4. Write CUDA versions with various memory layouts in global memory and compare
- 5. Use a profiler to find explanations for the runtime differences
- 6. Write a faster CUDA variant using shared memory for caching
- 7. Find the best memory layout combination for global/shared memory