Parallel Implicit Time Integration for Particle-based Models on Graphics Clusters

Authors

Adrian SABOU
Dorian GORGAN

Technical University of Cluj-Napoca

Overview

- Introduction
- Objectives
- Particle-based models
- Implicit Numeric Integration
- Parallelization
- Parallel Implicit Integration
- Experiments
- Conclusions and Future Work

Introduction

- Particle-based models are used for simulating soft body dynamics
- **Soft body dynamics** is a field of computer graphics that focuses on visually realistic physical simulations of the motion and properties of deformable objects (or soft bodies)
- E.g.
  - Cloth simulation (Creating Garments)
  - Human tissue (Virtual Surgery)
  - Hair and Vegetation (Computer Games)

Introduction (contd.)

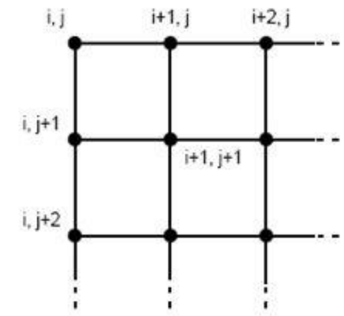
- Issues
  - Usually, real-time execution is required
  - Realistic simulation requires complex models (i.e. large numbers of particles)
  - Complex models have high computational cost

Objectives

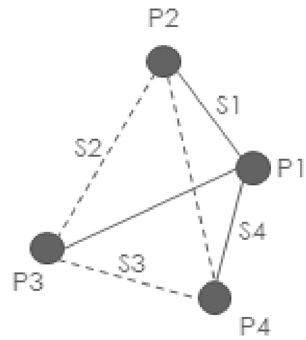
- Research methods for accelerating particle-based simulators
- Parallelize the computation requires to simulate particle-based models
- Run simulations on a graphics cluster
- Assess performances

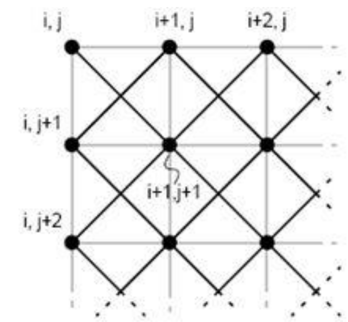
Mass-Spring Model

- Particle-based model

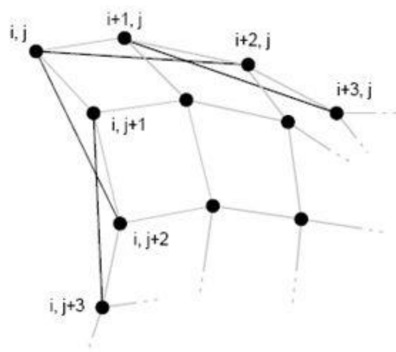


- Points of mass interconnected by springs
  - Structural springs
  - Shear springs
  - Bend springs





- For modeling volume, the points of mass can be organized in a tetrahedron



Mass-Spring Model (contd.)

- Behavior of mass-spring systems is dictated by Hooke's Law of Elasticity

$$\boldsymbol{F}_{\boldsymbol{el}} = -k\Delta\mathbf{x}$$

  - where *k* – constant of elasticity, Δx – elongation

- Force formula for particles *i* and *j*

$$\boldsymbol{F}_{i,j}^{el} = k \cdot \boldsymbol{x}_{i,j} \cdot \left(1 - \frac{l}{\left|\boldsymbol{x}_{i,j}\right|}\right)$$

  - where $\boldsymbol{x}_{i,j} = \boldsymbol{x}_i - \boldsymbol{x}_j$ and *l* is the rest length

- Damping

$$\boldsymbol{F}_{i,j}^{damp} = -\gamma\boldsymbol{v}_{i,j}$$

  - where $\boldsymbol{v}_{i,j} = \boldsymbol{v}_i - \boldsymbol{v}_j$

Implicit Numeric Integration

- The evolution of the system in time is obtained by integrating Newton's equations of motion

$$\boldsymbol{F} = m\ddot{\boldsymbol{x}}$$

- Implicit Euler integration

$$\begin{cases} \Delta\boldsymbol{x} = (\boldsymbol{v}_t + \Delta\boldsymbol{v}) \cdot \Delta t \\ \Delta\boldsymbol{v} = M^{-1} \cdot \boldsymbol{F}(\boldsymbol{x_t} + \Delta\boldsymbol{x}, \boldsymbol{v}_t + \Delta\boldsymbol{v}) \cdot \Delta t \end{cases}$$

- Linearization

$$\boldsymbol{F}(\boldsymbol{x_t} + \Delta\boldsymbol{x}, \boldsymbol{v}_t + \Delta\boldsymbol{v}) = \boldsymbol{F}(\boldsymbol{x_t}, \boldsymbol{v}_t) + \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{x}} \cdot \Delta\boldsymbol{x} + \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{v}} \cdot \Delta\boldsymbol{v}$$

- Final discretization

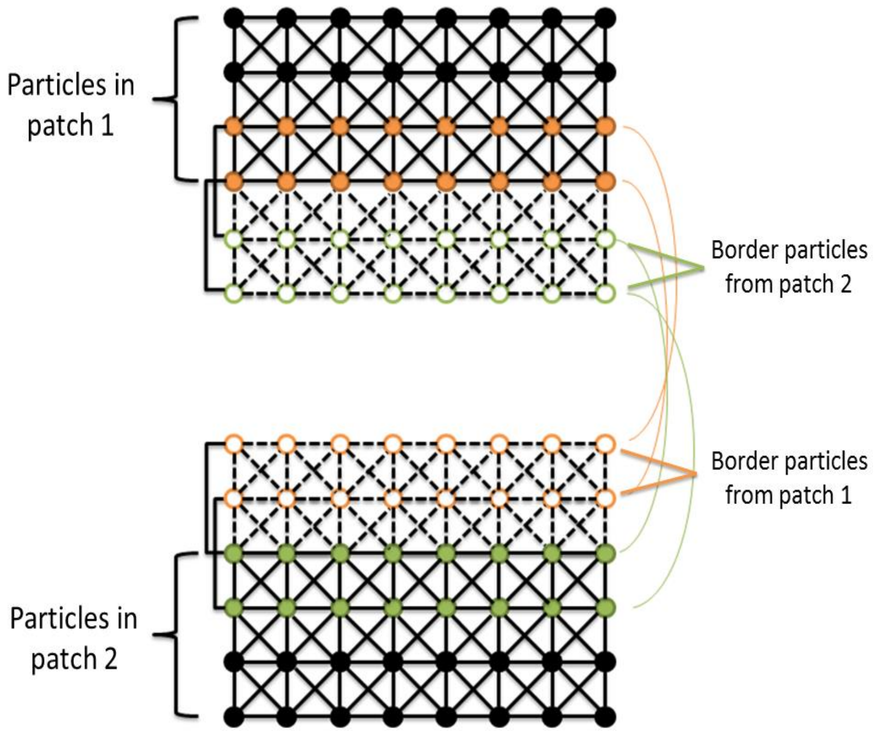
$$\left(I - M^{-1} \cdot \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{v}} \cdot \Delta t - \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{x}} \cdot \Delta t^2\right) \cdot \Delta\boldsymbol{v} = M^{-1} \cdot \left(\boldsymbol{F}(\boldsymbol{x_t}, \boldsymbol{v}_t) + \frac{\partial\boldsymbol{F}}{\partial\boldsymbol{x}} \cdot \boldsymbol{v}_t \cdot \Delta t\right) \cdot \Delta t$$

Achieving Parallelization

- Computation process for each particle is independent of other particles
- Each particle can be handled by a different thread
- Taking advantage of the power of the GPU (SIMD - Single Instruction Multiple Data)
- OpenCL – newly emerging standard
  - Allows parallel programming of heterogeneous systems
  - Interoperability with OpenGL through Vertex Buffer Arrays (VBOs)

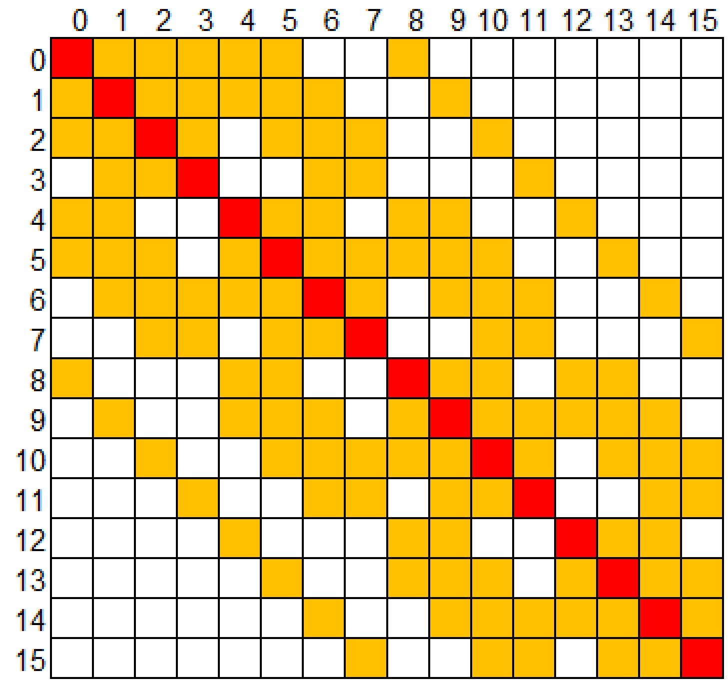
Achieving Parallelization (contd.)

- Taking it further…
- Dividing the model among more computers: GPU cluster
  - N computers – N patches
  - Assign each patch to one node
  - Synchronize common regions at each simulation step



Parallel Implicit Integration

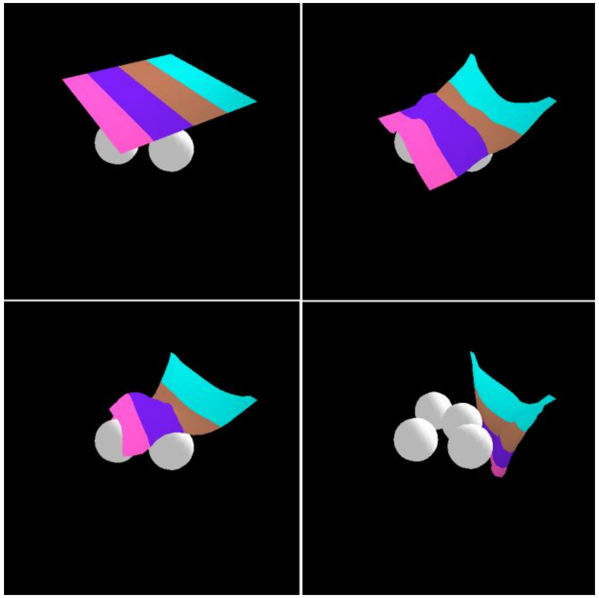
- Parallel Conjugate Gradient (Ax=B)
  - ViennaCL
    - Sparse Matrix-Vector Multiplication (SPMV)
    - Compressed Sparse Row (CSR)
      - A, IA, JA
      - IA, JA - precomputed



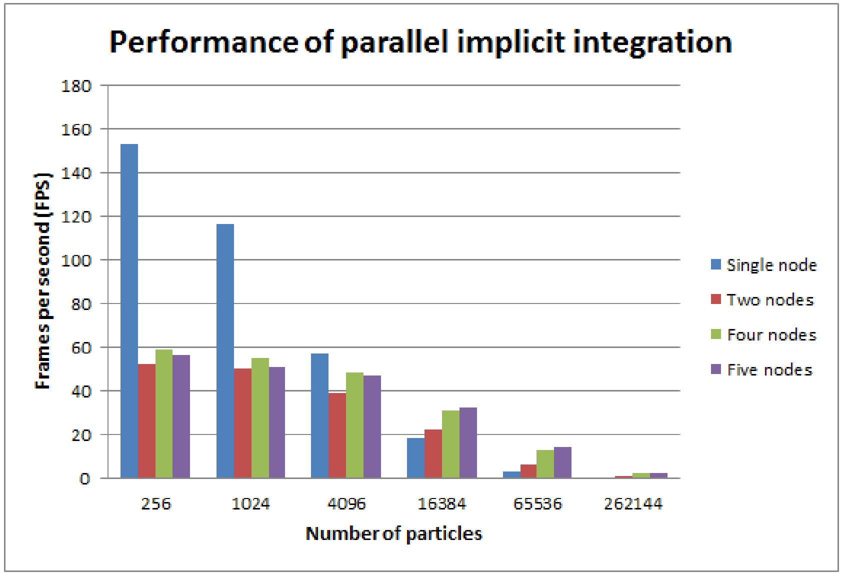
- Parallel Global Matrix Update (A)
  - Each row handled by a different thread
  - Eg. – structure of A for a 4x4 grid of particles

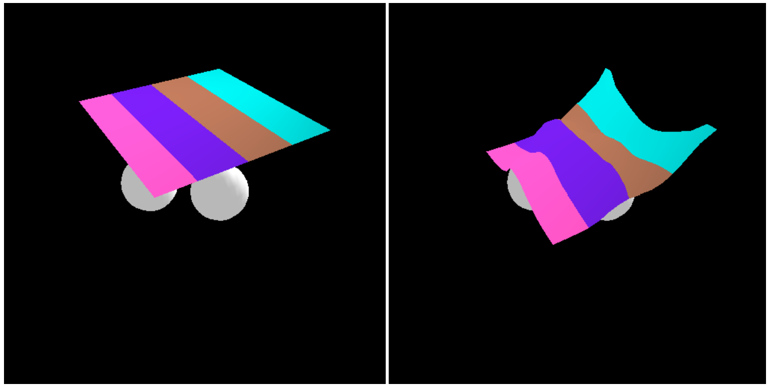
Experiments – Parallel Matrix Update

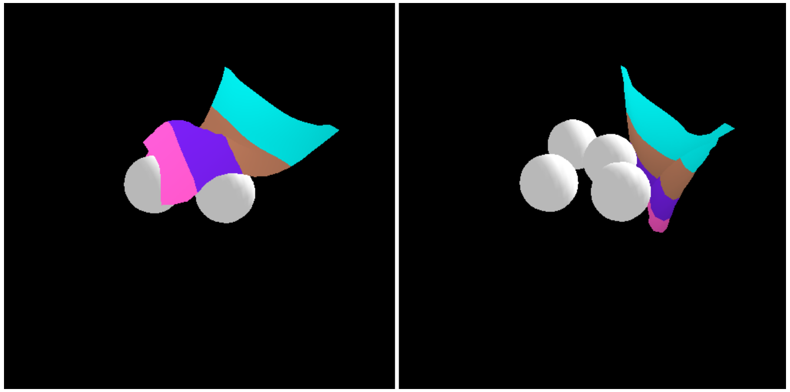
Nb. of particles	Sequential time (s)	Parallel time (s)
256	0.193	0.000038
1024	0.914	0.000038
4096	4.57	0.000038
16384	36	0.000038
65536	164	0.000038

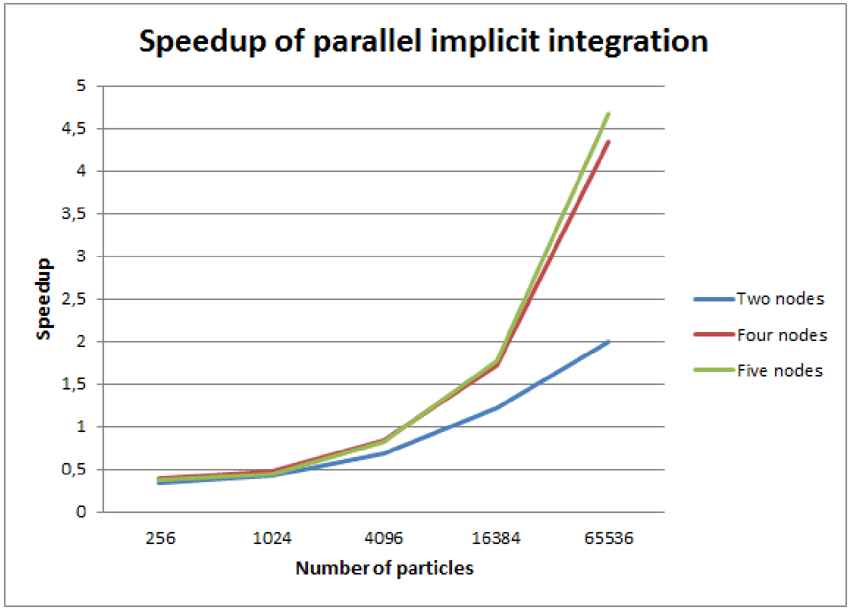


Experiments – Parallel Matrix Update (contd.)









Conclusions and Future Work

- Conclusions
  - GPU – an efficient way to accelerate mass-spring models simulations
  - Performance can be further increased by distributed computing – GPU clusters
  - Efficient update method for the global matrix A
- Future Work
  - Experiment on more performant GPU cluster architectures

Thank you for your attention!

Questions or Comments?

Authors

Adrian SABOU
Dorian GORGAN

Technical University of Cluj-Napoca