Granular Dynamics Simulations using Charm++

PARVATHI MK

TEJ KUMAR

Outline

- The defining features of Charm++
- The physics of the problem
- Implementation
- Results
  - Simulation results
  - Scalability
  - Optimization of chare size

Charm++

A programming paradigm based on:

- Objects
- Overdecomposition
- Message
- Asynchrony
- Migratability
- Runtime system

Special Objects = Chares

- Overdecomposed entities: Chares
- Special functions: “Entry methods” are invoked by remote chares.
- Arguments to entry methods are packaged as a message.
- Chares can be multidimensional to reflect the underlying spatial decomposition.
- Communication via asynchronous method invocations
- Chares don’t belong to any particular PE; they are migrated by the runtime system in order to facilitate dynamic load balancing.

Chare Proxys and Sections

- A proxy is an ID that is used to recognize/locate a chare in a global sense.
- Utilized while invoking entry methods.
- A Proxy class is automatically generated for every chare.
- Chare section is a sub-group of chares associated with a given chare; mostly utilized for neighbor search and reductions.

Reduction Operations

- A feature similar to MPI_Reduction.
- Used to reduce local data scattered across a chare array to a single global value.
- Implemented by the chare member function '`contribute`' with the syntax :

```
void contribute(int  nBytes, const void *data, CkReduction:: reducerType
type);
```

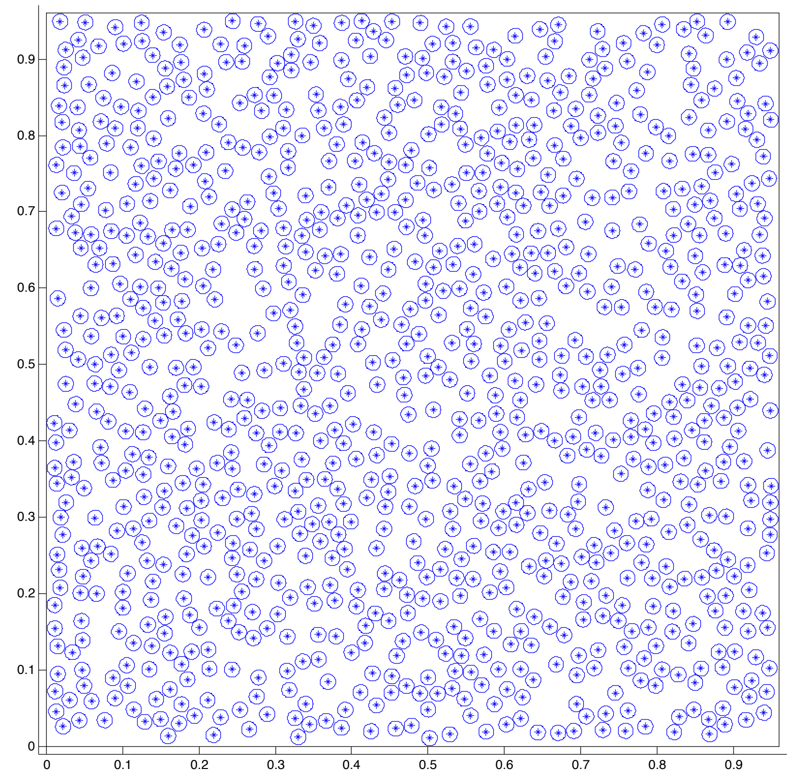
- The output of the reduction is handled using a callback object, 'reduction client'.
- Reduction client is just a function that is called once the reduced data is ready for access.

SDAG and ‘`when`’ Statements

- Chares are reactive entities; performs actions when invoked.
- Special construct : ‘`when`’
  - Specific actions performed ‘when’ a message is received.
  - Acts a ‘blocking recieve’.
  - Enforces synchronization of chares due to interdependence.
  - A ‘`when`’ must have a corresponding declaration of an entry method.

Underlying Physics and Assumptions

- Simulation is done in 2D
- Particles allowed move around in a rigid square box.
- Spherical Particles
  - Mass = 1E-1 kg
  - Radius = 1E-2 m
  - Max. Average Initial velocity ~ 0.5 m/s
- Time step = 1E-3 s
- Time stepping using Forward Euler



Force Modeling

- Interaction of particles modeled by using a spring-mass model
- Spring constant, k = 1E+5 N/m

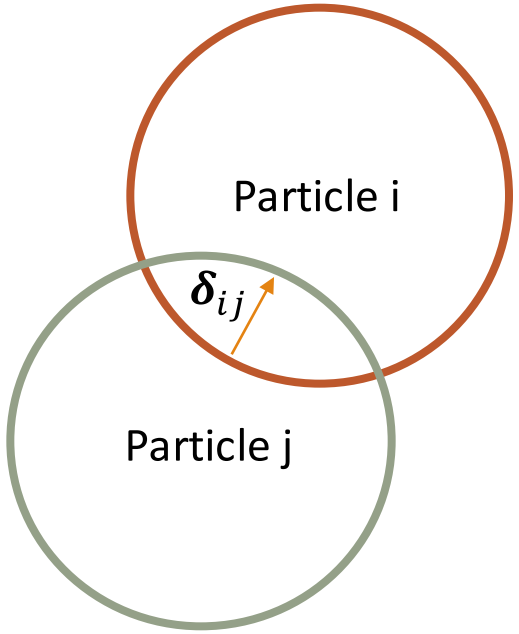
$$\boldsymbol{F_{ij}} = k\boldsymbol{\delta_{ij}}$$

Here F_{ij} is defined as the force on the ith particle due to the jth particle.

$$\boldsymbol{\delta}_{ij} = 2r\hat{\boldsymbol{n}}_{ij} - (\boldsymbol{R}_i - \boldsymbol{R}_j)$$

where R_i and R_j are the position vectors of the particles.

- No damping incorporated.
- Wall Interactions modeled as elastic collisions.



Implementation

- The convoluted programming model inhibited coding from scratch.
- Starting point was required which can guide in understanding and coding.
- LeanMD is a molecular dynamics simulation application written in Charm++.
- Granular Dynamics code is a modified LeanMD code:
  - 3D to 2D.
  - Periodic to wall boundary condition: This was challenging. Led to different no. of neighbours for different spatial chares(or subdomains).
  - Different Physics: Forces computed based on region of influence is replaced with contact detection and subsequent force computation based on amount of overlap.

Domain Decomposition

Spatial Chares

- Called Cells, identified by unique 2D indices.
- Square domain is divided using square spatial chares.
- These chares contain particle data which involves position and velocity.

Compute Chares

- Required for carrying out computation.
- Each of these chare contain indices of two spatial chares i.e. these are 4D chares.
- These two can be neighbouring spatial chare or the same chare.

Sections

- These are the groups of compute chares.
- All compute chares containing common spatial chare are grouped as a section.

Various files

Interface file – leanmd.ci

- Contains all the chare classes.
- Contains all the entry methods.
- Contains all the readonly variables.

Main.cc

- Contains the mainchare generally named Main.
- Program execution begins with mainchare.
- All the readonly variables are initialized here.
- Creates the spatial chares and the corresponding proxy.

Various files contd…

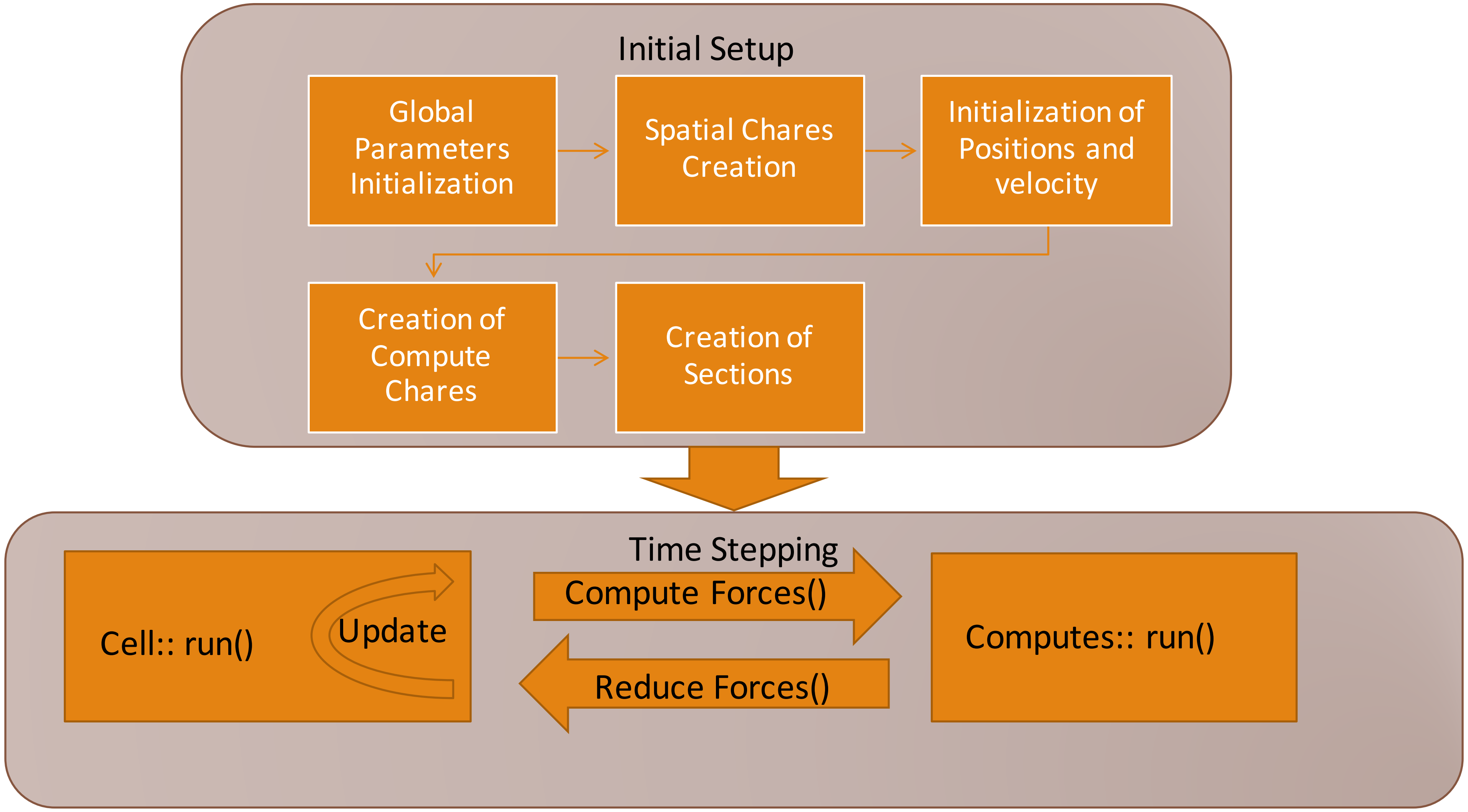
Cell.cc

- Contains most of the functions like for creating computes and initializing the cells.
- Contains functions responsible for property updating and migration of particles among chares.
- Sections of compute chares are created here and those sections are involved in force computation as discussed later.

Compute.cc

- Contains functions like interact and self-interact. .
- These functions call the force calculation functions.
- The calculated forces are reduced in interact and self-interact.

Program Flow

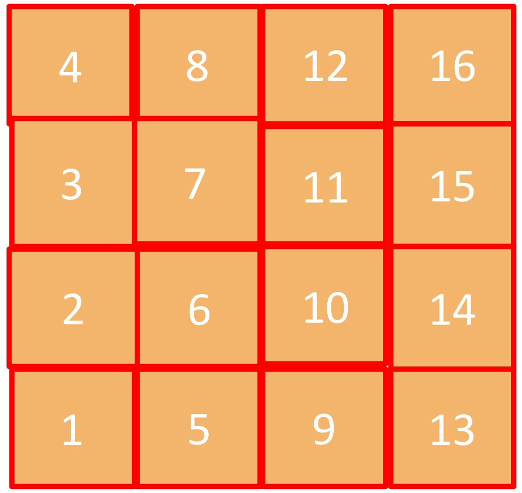


Compute chares and Section creation

Compute chares are responsible for carrying out collision detection and force computation.

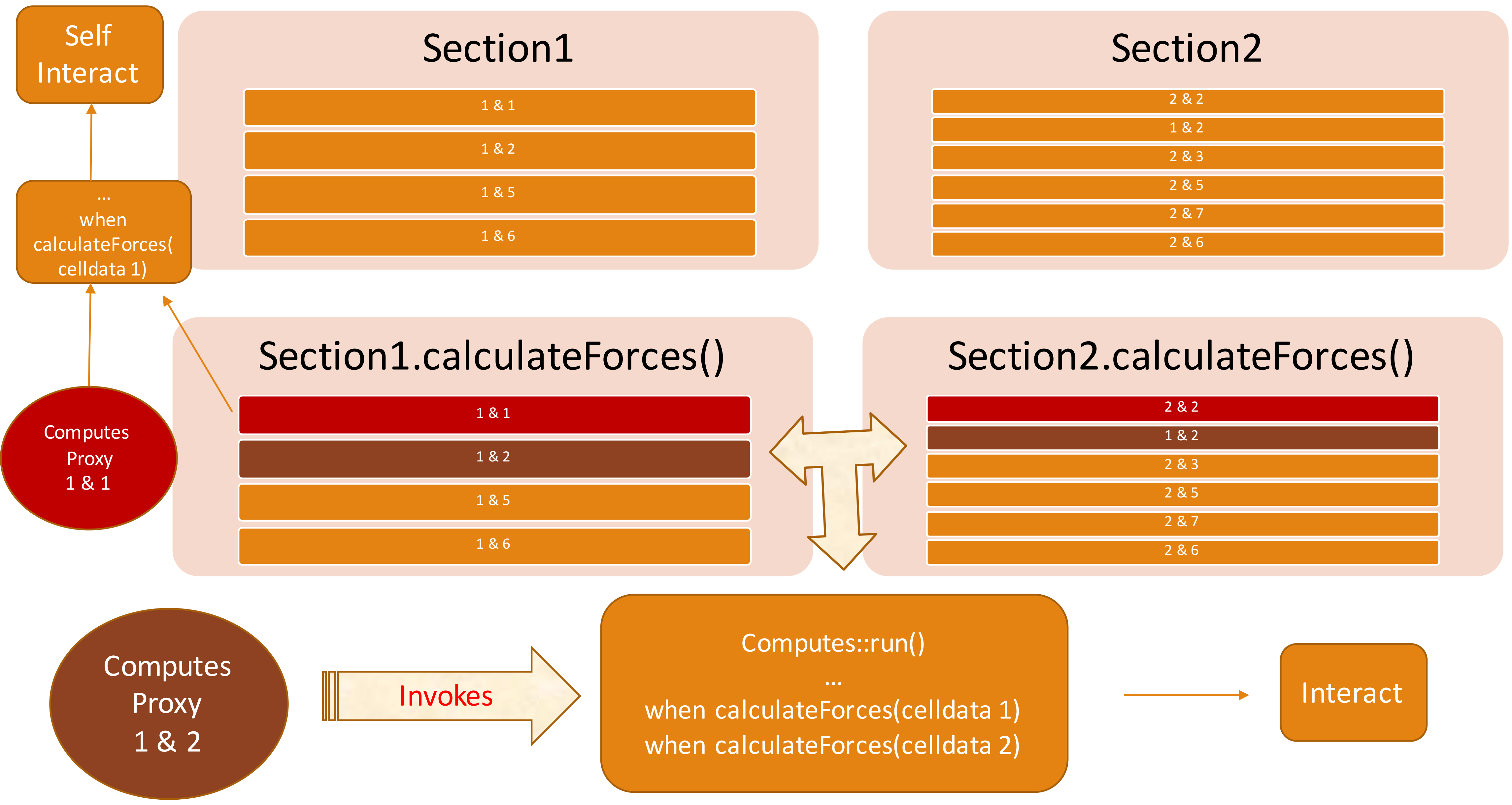
They are of two types based on purpose:

- Within a spatial chare
- Between two neighbouring chares.
  - Total no. of these chares is R*(C-1) + C*(R-1) + 2*(R-1)*(C-1)

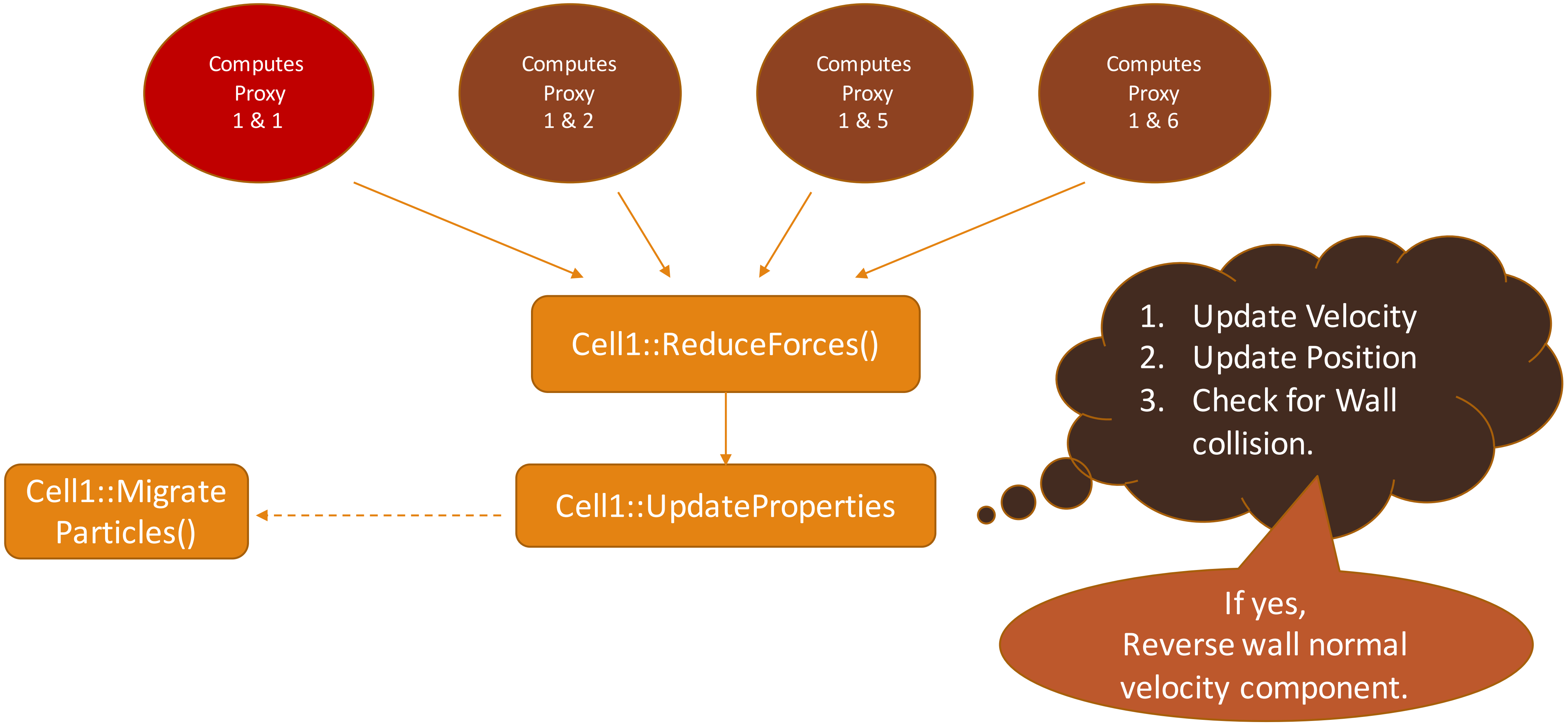


Section

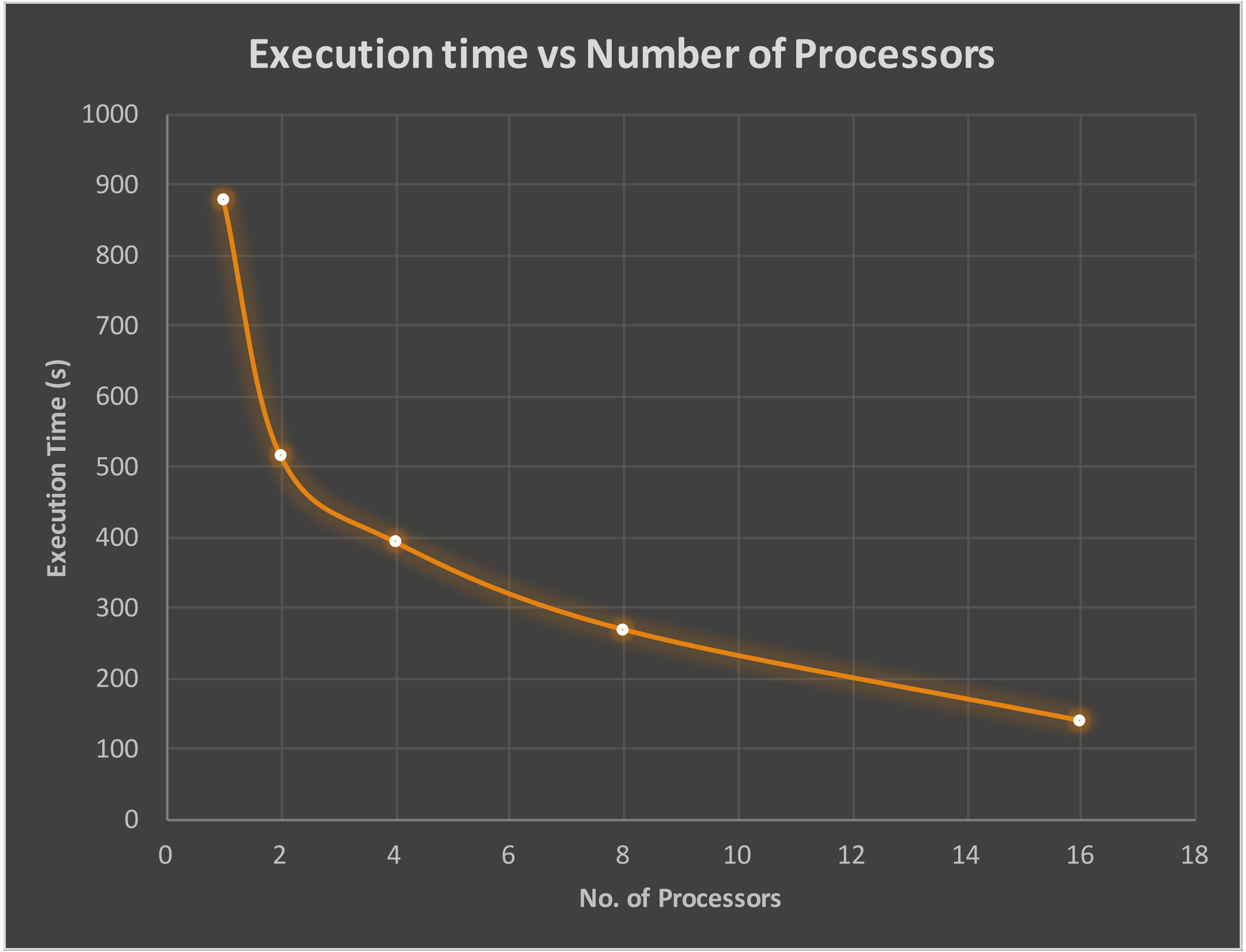
- is created corresponding to each spatial chare.
- contains list of all the neighbouring chares including self.
- created based on Computes List which is generated along with computes.


Self Interact
…
when calculateForces(celldata 1)
Computes Proxy 1 & 1
Section1
1 & 1
1 & 2
1 & 5
1 & 6
Section2
2 & 2
1 & 2
2 & 3
2 & 5
2 & 7
2 & 6
Section1.calculateForces()
1 & 1
1 & 2
1 & 5
1 & 6
Section2.calculateForces()
2 & 2
1 & 2
2 & 3
2 & 5
2 & 7
2 & 6
Computes Proxy 1 & 2
Invokes
Computes::run()
…
when calculateForces(celldata 1)
when calculateForces(celldata 2)
Interact

Force Computation



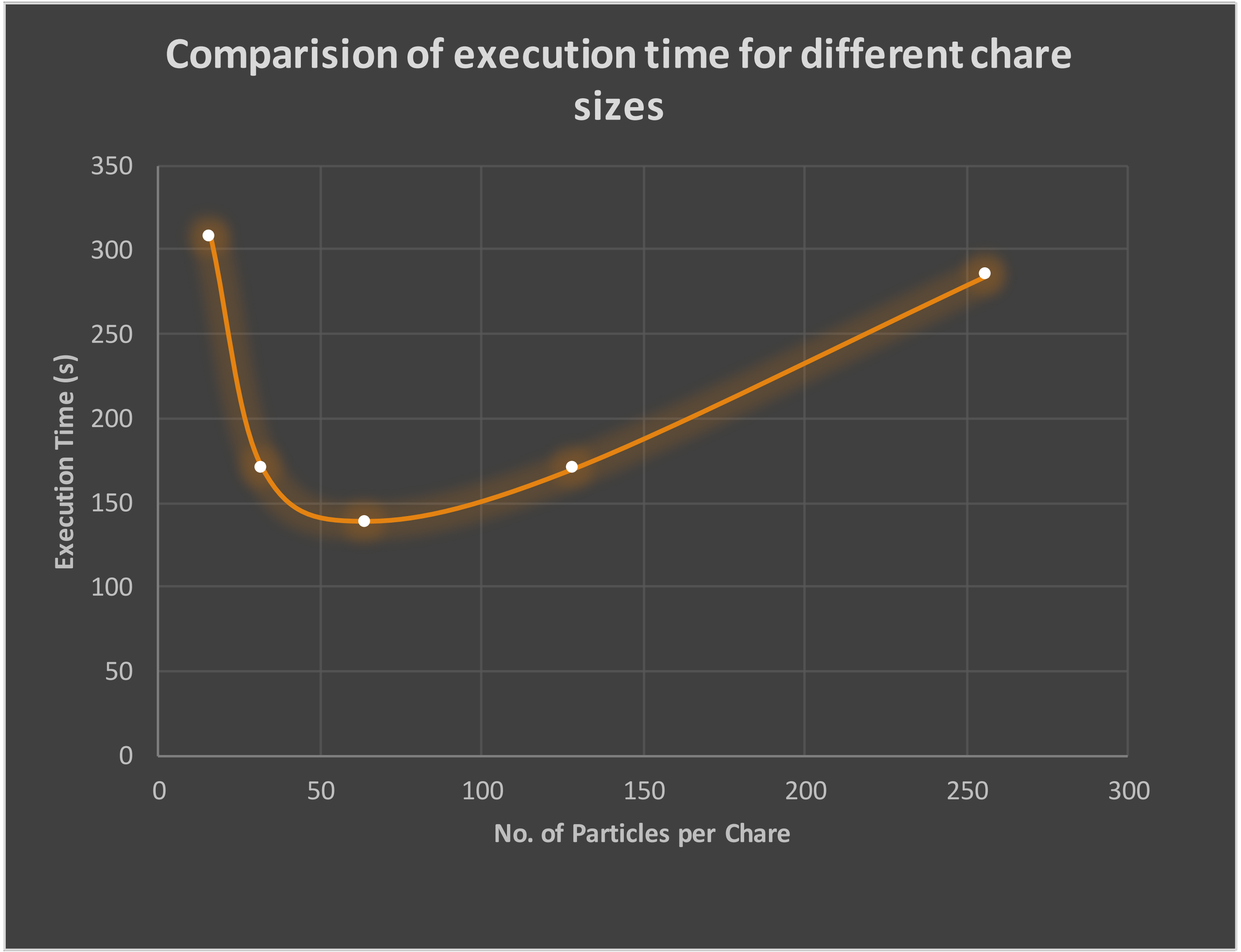
Scaling Analysis



Execution Configuration

- Time steps = 2000
- Total number of particles =262144
- Number of chares = 64 x 64

Optimization of Chare Size



Execution Configuration

- Time steps = 2000
- Total number of particles =262144
- Number of Processors = 16
- Optimum Chare size = 8 x 8

Observations

- The execution time for granular dynamics simulation of a given number of particles is dependent upon the chare size chosen as well as number of processors.
- The chare size has an optimum value which in our case turns out to be 8 X 8 in the limited number of trial runs.
- The initial higher times could be due to memory latency which is hidden by increasing the number of computations within a processor.
- The increase afterwards could be due to the increased execution time for collision detection.
- The trend with change in number of processors is very intuitive i.e. the execution time decreases exponentially with increase in number of processors.

References

- www.charm.cs.illinois.edu/tutorial/
- http://charmplusplus.org/
- Dan’s presentation material on Charm++.
- **A Discrete Numerical Model for Granular Assemblies**, P. A Cundall, O.D.L Strack, *Geotechnique,* Vol 29 (1), **1979**

Acknowledgements

We would like to thank Felipe Barragan for his earnestness in helping us understand the LeanMD code and Charm++. The discussions were invaluable.

Thank You.