Scaling and Complexity: Spatial Decomposition MD

T. Gottschalk and E. Upchurch
Center for Advanced Computing Research
California Institute of Technlogy

I: The Molecular Dynamics (MD) Problem

The basic computational problem is the evaluation of the total force on a particle, written as a sum over pair-wise forces arising from all other particles in an ensemble:

$$\mathbf{F_j = ?_{i?j} F_{ij}}$$

The pairwise force Fij is provided by some dynamical model (e.g., described by a Lennard-Jones potential). It depends on the positions of the two particles involved and possibly on other state variables of the physics model.

The kinematic state of an individual particle at a time t is specified by the particle's position and velocity. The force equation gives the acceleration that is used to update the particle's state through some small time step ? **T**. ("Real" MD codes generally use more sophisticated integrators. This is a per-particle computational cost and does not affect the scaling discussions of this note.)

The essential simplifying assumption for MD models is limited range of the pairwise forces:

$$\mathbf{F_{ij} = 0, \quad |r_{ij}| > r_C}$$

The force cutoff r_C is a parameter of the model. Given this assumption, the total computational cost for a single update cycle is approximately

Where

1. N_{TOT} is the total number of particles
2. N_{NBD} is the (typical) number of particles in the force neighborhood of an individual particle
3. ? is the cost of integrating the equation of motion for an individual particle over the (small) time step.
4. ? is the cost of computing a single inter-particle force F_{ij}
5. ? is the cost of finding/enumerating particles in the neighborhood of the current particle of interest.

The coefficients ‘a’ and ‘b’ are fairly straightforward and could presumably be measures by profiles of single-processor executions of an actual code. The “finding” coefficient ‘d’ is a bit more complicated and will be discussed in more detail below.

II. Spatial Decomposition Algorithm: Qualitative Overview

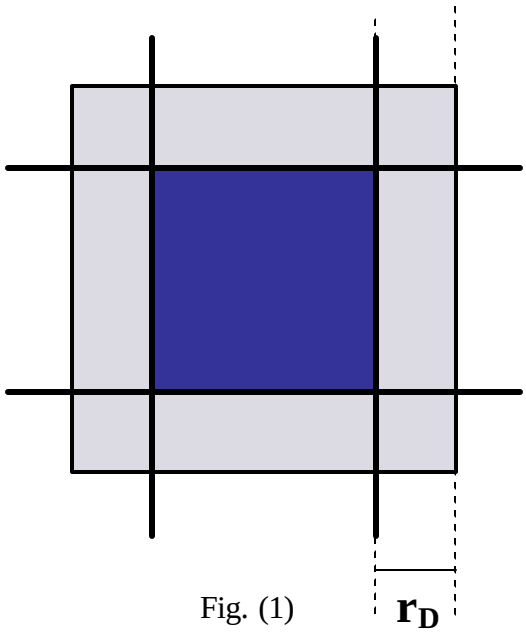

Fig. (1)

The Spatial Decomposition (SD) algorithm for parallel MD can be described as follows:

1. The physical volume is divided into a (regular) grid.
2. Each grid cell (e.g., the dark square above) is assigned to a processor, and a processor is responsible for performing the force calculations and state updates for all particles (nominally) within the cell.
3. Force computation requires state information for some particles owned by other processors – the lightly shaded area in the figure. These are acquired by a communications phase at the start of each computational step.
4. Particles will occasionally drift across processor boundaries. These processors remain the responsibility of the original parent processor during the basic (Communicate,Update) cycle outlines in steps 2 and 3. Reassignment of particles to processors according to the cell boundaries is done periodically but (far) less frequently than the basic update cycle.

The communications for the data sharing of Step 2 are straightforward and involve synchronized messaging within the grid.

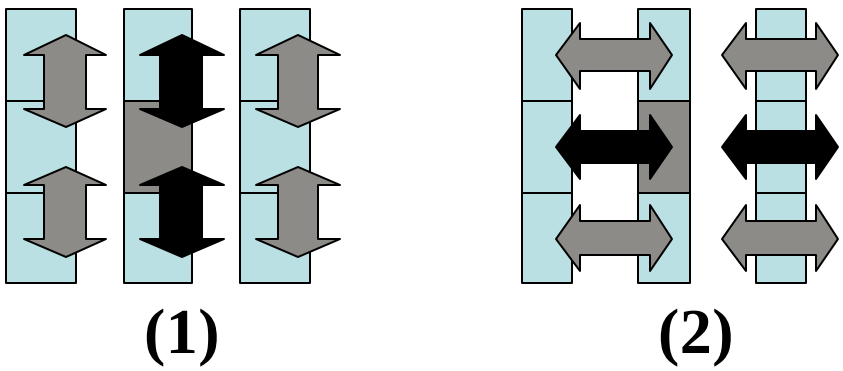

Fig. (2)

The communications phase is a number of pairwise data exchanges between (logically) neighboring processors. In terms of diagram, the steps are as follows.

1. Processors send all particles within the interaction of a horizontal boundary to the other processor at that boundary, at the same time accepting particles from that processor.
2. The "vertical" sharing in step (1) is then repeated in the other physical dimensions.

During the second/horizontal sharing, a processor will generally send some particles it received during the preceding vertical sharing. This is the mechanism for acquiring relevant data from the "diagonal neighbors".

III. Complications and Simplifications

Ignoring the periodic, lower frequency reassignments of ownership of particles that drift across cell/processor boundaries, the basic update cycle for any one processor has two parts:

1. **Communications**: Retrieve current positions of "boundary" particles assigned to neighboring processors. Send current state of boundary particles known by this processor to neighbors
2. **Computation**: Perform the force evaluation and state update calculation for all particles owned by the processor.

The amount of communications depends on the relative magnitudes of the force range (r_S) and the width (d) of a physical grid cell assigned to a given processor. If $d < r_S$, then the current positions must be exchanged across multiple hops in the communications scheme of Fig. (2). In the other cases, we can approximate

$$\mathbf{N_{COMM}} = ? \ \mathbf{N_{TOT}}$$

For some scale factor ?,

? = Fraction of local particles interesting across a single boundary.

The analysis here makes this assumption, ignoring the more complex $d < r_S$ case.

The low frequency rearrangement of particles across cell boundaries will also typically involve some (smaller) fraction of the local particles. It is during this lower frequency exchange that Plimpton recommends reconstruction of the the data structures used for efficient near neighbor searches in the force computation loop. For now, the scaling behaviors and expectations for this low-frequency particle migration and search tree reconstruction are ignored.

IV. Parameterized Model: Performance and Scaling

The activity of an individual processor for a single computational cycle can thus be modeled by a simple "time line", as shown below.

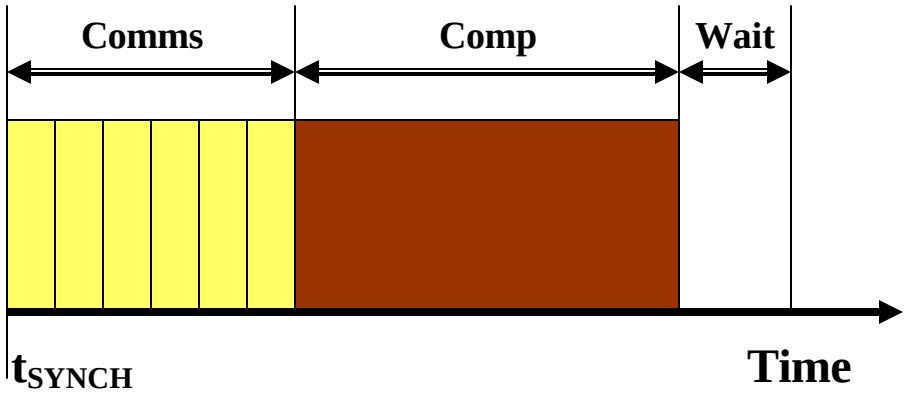

Fig. (3)

The activities and expected costs/times for these components are as follows:

1. **Communication**

   In each of three dimensions and two directions per dimension, the processor exchanges data with its neighbor. The amount of data exchanged is

   $$\textbf{Data} = ? * N_{LOC} * (\text{Individual Datum Size})$$

   A typical datum size would be three doubles for position and one int for particle ID. This gives the size of the message. Actual communications costs will depend on the location of the logically adjacent processor within the communications network.

2. **Computation**

   As described above in Section I, the cost/time for the computational phase can be written as

   $$\textbf{Cost} = N_{LOC} (? + ? N_{LOC})$$

   Where, for simplicity, the data structure maintenance cost (?) has been ignored.

3. **Synchronization/Waiting**

   The pairwise data exchanges of Fig.(2) are synchronized. This will introduce various communications delays that have been collectively lumped into a single Wait Time before the start of the next simulation step.

In the above,

$$\mathbf{N_{LOC} = N_{TOT}/N_P}$$

Is the "local" particle count – the number of particles out of N_{TOT} total particles owned by one of N_P total processors. The N_{NBD} "force neighborhood" count from Section I has been estimated as some fraction of the Local count – essentially an assumption of approximately uniform particle densities across the system.

The overall scaling behavior will clearly depend on which of the parameters N_{TOT}, N_{LOC}, N_P are held fixed.

This provides a simple three parameter model for approximating the Spatial Decomposition MD algorithm in a Speedes-based simulation. The various points on the time axes of Fig. (3) are the discrete events for the simulation. The communications message size estimates the total byte count for each message in terms of one parameter (?) and the Computation cost is a simple two-parameter representation.