RC 22059 (W0105-011) May 16, 2001
Computer Science

IBM Research Report

A Note on N-body Computation with Cutoffs

Marc Snir
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

LIMITED DISTRIBUTION NOTICE: This report has been submitted for publication outside of IBM and will probably be copyrighted if accepted for publication. It has been issued as a Research Report for early dissemination of its contents. In view of the transfer of copyright to the outside publisher, its distribution outside of IBM prior to publication should be limited to peer communications and specific requests. After outside publication, requests should be filled only by reprints or legally obtained copies of the article (e.g. , payment of royalties). Copies may be requested from IBM T. J. Watson Research Center , P. O. Box 218, Yorktown Heights, NY 10598 USA (email: reports@us.ibm.com). Some reports are available on the internet at http://domino.watson.ibm.com/library/CyberDig.nsf/home .

This page intentionally left blank.

A Note on N-body Computations with Cutoffs

Marc Snir
IBM Research

May 16, 2001

IBM Thomas J. Watson Research Center
Yorktown Heights, NY 10598-0218 [1]

Abstract

We provide a theoretical analysis of the communication requirements of parallel algorithms for molecular dynamic simulations. This paper contributes some improvements and a more precise analysis of communication, for previously used algorithms. It defines a new algorithm with improved communication. It analyzes it, both for a machine with a complete graph topology and for a machine with a mesh topology, and proves matching lower bounds on communication complexity.

1 Introduction

1.1 Outline of Paper

We consider in this paper parallel algorithms for molecular dynamic simulations that use finite difference methods. Such simulations compute the movement of interacting atoms at discrete time intervals by repeatedly computing the forces acting on atoms and updating accordingly the position and velocity coordinates of these atoms. These simulations are very important in biomolecular research, as they allow us to understand the behavior of biological systems at the mesoscale level. Molecular simulations are expected to become increasingly important in drug design and in various applications of material sciences. However, these simulations are very time consuming and may require massive parallelism to be performed in acceptable time [2]. Several parallel implementations of the finite difference method have been analyzed in the past, in some detail [12, 13]. The analysis has shown that communication requirements may be a significant impediment to the scalability of these algorithms.

This paper contributes a more precise analysis of communication. We focus on the computation of inter-molecular forces, which usually is the most time consuming part of a molecular dynamic simulation. The next section summarizes the usual setting for such simulations and motivates the definition in Section 1.3 on page 3 of the generic problem that we study in this paper, which we call "N-body computation with cutoff".

We proceed in Section 2 on page 6 with an introduction of the theoretical framework we use for algorithmic analysis. We show that the parallel algorithms for the N-body

[1]Current address: Computer Science Dept., UIUC, Urbana, IL 61801; *snir@cs.uiuc.edu*

computation with cutoff can be succinctly specified by defining how the computation of forces and of atom coordinates is distributed among processors, and by defining a broadcast and reduce communication pattern for atom positions and for forces, respectively. We also formally prove a "duality" result, showing that the same communication pattern can be used for the broadcast as for the reduce.

In Sections 3.1 on page 10 and 3.4 on page 15 we represent and analyze in this framework variants of the two main algorithms that have been used in the past: space decomposition and force decomposition. We improve previous bounds on communication by providing a tighter analysis, and also provide a more rigorous analysis on the use of randomization for load balancing. We show that one collective communication operation, namely shift on one dimension of a (virtual) mesh, is sufficient for implementing these algorithms.

We introduce in Section 4 on page 21 a new algorithm that further reduces communication. This new algorithm can also be implemented using only shifts. In Section 5 on page 23 we show that this new algorithm is optimal, by providing a matching lower bound on communication.

Finally, we further refine communication analysis in Section 6 on page 31 by considering the mapping of our new algorithm on a system with a mesh topology, such as Blue Gene [2]. We conclude in Section 7 on page 34.

1.2 Molecular Dynamics

We consider in this paper N-body computations, as used in molecular dynamics. A molecular system of n atoms is simulated by repeatedly computing the forces acting between atoms and updating the atom coordinates accordingly. Each atom a is associated with some constant parameters $\mathbf{p}(a)$ (atom type, mass, charge, etc.) and some varying coordinates. These include position $\mathbf{x}(a,t)$ and may include velocity $\dot{\mathbf{x}}(a,t)$. If $\mathbf{f}(a,t)$ is the force applied on atom a at time t, then position and velocity are updated as

$$\mathbf{x}(a,t+\Delta) = \mathbf{x}(a) + \Delta\dot{\mathbf{x}}(a,t),$$
$$\dot{\mathbf{x}}(a,t+\Delta) = \dot{\mathbf{x}}(a) + \Delta\ddot{\mathbf{x}}(a,t),$$

where

$$\ddot{\mathbf{x}}(a,t)\,\mathrm{mass}(a) = \mathbf{f}(a,t).$$

More typically, one eliminates velocity by using a Taylor expansion of position. The widely used Verlet algorithm [14, 1] has

$$\mathbf{x}(a,t+\Delta) = 2\mathbf{x}(a,t) - \mathbf{x}(a,t-\Delta) + \Delta^2\ddot{\mathbf{x}}(a,t).$$

In this case, the algorithm maintains, for each atom, the positions at two successive time intervals.

The force applied on an atom a is the vector sum of the pairwise interactions of that atom a with all other atoms b in the system. Those pairwise interactions can take several forms, as described in [1]. These include *intra-molecular* forces that occur between atoms that are close-by in the same molecule and *inter-molecular* forces that occur between any pair of atoms. The number of intra-molecular interactions is linear in the number of atoms. On the

other hand, the number of inter-molecular interactions is, in principle, quadratic. Thus, for large system size n, most of the computation time is spent on inter-molecular forces. This holds true even if various cutoff methods are used to reduce the number of inter-molecular forces that have to be computed. Furthermore, the computation of intra-molecular forces require little communication as it typically involves atoms held at the same processor, or at near-by processors. On the other hand, the computation of inter-molecular forces has little intrinsic locality and may require expensive, global communication. Therefore, we focus in this paper on the computation of inter-molecular forces.

There are two types of inter-molecular forces: *Lennard-Jones* forces and *Coulomb* forces. The magnitude of the Lennard-Jones force between two atoms a and b is given by a term of the form

$$U_{LJ} = \frac{A_{ab}}{r_{ab}^{12}} - \frac{B_{ab}}{r_{ab}^{6}}$$

where r_{ab} is the distance between the two atoms and A_{ab} and B_{ab} are constants that depend on the atom types; $A_{ab} = A_{ba}$ and $B_{ab} = B_{ba}$. The magnitude of the Coulomb force is computed by a term of the form

$$U_C = \frac{q_a q_b}{\epsilon r_{ab}}$$

where q_a and q_b are the atom charges. The Lennard-Jones forces decay rapidly with distance so that, usually, a *cutoff* distance is used, and forces are computed only for atoms that are within this cutoff. Coulomb forces, on the other hand, decay slowly, so that one need to account for all pairwise interactions. However, it is common to simulate a system that consists of a periodic lattice (i.e., a rectangular cell that is replicated infinitely in all three dimensions). In such a case, on can use the *Ewald* method [7, 1] where the computation of the Coulomb forces is divided into two fast converging sums: a *real-space* part, where interactions are computed directly only for atoms within a cutoff distance, and a *k-space* part, where long-range interactions are computed in a transform space. By suitably balancing the two computations, one can reduce the asymptotic complexity to $O(n^{1.5})$. The use of a periodic system not only simplifies the computation of Coulomb forces; it also leads to a more realistic simulated system, as it avoids the "non-physical" water-vacuum layer one has at the boundary of the simulated cell in an aperiodic system.

The asymptotic complexity of the Coulomb force computation can be further reduced to $O(n)$ by using a fast multipole algorithm [9, 10]. However, it is not clear that fast multipole algorithms are faster, in practice, for system sizes and machine sizes of interest [4].

1.3 General Problem Formulation

We assume that the simulation uses a cubic cell $\mathbf{D}$ of dimensions $d \times d \times d$, replicated infinitely in all three dimensions. The results can be easily extended to rectangular cells of bounded aspect ratio, or to an acyclic system. This cell contains a set $\mathcal{A} = \{a_1, \ldots, a_n\}$ of n atoms. We denote by $\mathbf{x}(a) = (x_1(a), x_2(a), x_3(a))$ the Cartesian coordinates of atom a. The atom location varies with time. We omit here and below the implicit time argument t from our notation.

We use

$$\mathrm{D}(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{3}(x_i - y_i)^2\right)^{1/2}$$

for the Euclidian distance in $\mathbb{R}^3$.

A cutoff distance c is used. We assume that $c < d/2$. This implies that, for any two atoms a and b, there is at most one copy of b that is within the cutoff distance from a. Indeed, assume that atom a interacts with the copy of atom b at location $\mathbf{y}$, so that $\mathrm{D}(\mathbf{x}(a),\mathbf{y}) < c$. The other copies of atom b are at locations

$$\mathbf{y}' \;=\; \mathbf{y} + d\sum_{1=1}^{3} n_i \mathbf{e}_i,$$

for integer coefficients (n_1, n_2, n_3), not all zero. But

$$\mathrm{D}(\mathbf{y}', \mathbf{x}(a)) \;\geq\; \mathrm{D}(\mathbf{y}', \mathbf{y}) - \mathrm{D}(\mathbf{x}, \mathbf{y}) \;>\; d - c \;>\; c.$$

We define the *cyclic distance* between two points $\mathbf{x}$ and $\mathbf{y}$ to be

$$\tilde{\mathrm{D}}(\mathbf{x},\mathbf{y}) = \left(\sum_{i=1}^{3}\left((x_i - y_i) \bmod d\right)^2\right)^{1/2}$$

and define

$$\tilde{\mathrm{D}}(a,b) = \tilde{\mathrm{D}}(\mathbf{x}(a), \mathbf{x}(y)).$$

$\tilde{\mathrm{D}}(a,b)$ is the shortest Euclidian distance between a copy of atom a and a copy of atom b. Also, we denote by $\mathbf{n}(a,b)$ a normal vector that points from $\mathbf{x}(a)$ to the location of the copy of atom b that is nearest to $\mathbf{x}(a)$.

We say that two atoms a and b *interact* if $\tilde{\mathrm{D}}(a,b) <$c and denote by $\mathcal{E} = \{ab \,:\, \tilde{\mathrm{D}}(a,b) < c\}$ the set of interacting atom pairs.

This paper analyzes the following generic problem:

N-body problem with cutoff: Given a n atoms $a_1, \ldots, a_n$ in a cubic cell $\mathbf{D}$ of dimension d, and a cutoff $c < d/2$, the algorithm maintains for each atom a a small tuple of fixed parameters $\mathbf{p}(a)$ and a small tuple of variable coordinates $\mathbf{x}(a), \mathbf{x}'(a), \ldots$. These include atom position $\mathbf{x}(a)$. The algorithm repeatedly performs the following:

Force computation: for each pair of atoms $(a,b) \in \mathcal{E}$ compute the force

$$\mathbf{f}_{ab} \;=\; \mathbf{f}(\mathbf{p}(a), \mathbf{p}(b), \mathbf{x}(a), \mathbf{x}(b)).$$

Force reduction: for each atom a compute the force

$$\mathbf{f}_a = \sum_{(b,a)\in\mathcal{E}} \mathbf{f}_{ba}$$

Atom update: For each atom a apply an update to the atom coordinates

$$\mathbf{x}(a), \mathbf{x}'(a), \ldots = \mathbf{U}(\mathbf{x}(a), \mathbf{x}'(a), \ldots, \mathbf{f}_a)$$

We make the following assumptions:

- The functions $\mathbf{f}()$ and $\mathbf{U}()$ can be computed in (low) constant time.
- $\mathbf{f}(\mathbf{p}(a), \mathbf{p}(b), \mathbf{x}(a), \mathbf{x}(b)) = f(\mathbf{p}(a), \mathbf{p}(b), \tilde{\mathrm{D}}(a,b)\mathbf{n}(b,a))$. The force that atom a exercises on atom b is in the direction from $\mathbf{x}(b)$ to the nearest instance of a and has a magnitude that depends only on the cyclic distance between the atoms, and their constant parameters.
- $\mathbf{f}(\mathbf{p}(a), \mathbf{p}(b), \mathbf{x}(a), \mathbf{x}(b)) = -\mathbf{f}(\mathbf{p}(b), \mathbf{p}(a), \mathbf{x}(b), \mathbf{x}(a))$. The force that atom a exercises on atom b has the same magnitude and a reverse direction that the force that atom b exercises on atom a.

The formulation given here for the generic N-body problem with cutoff encompasses the molecular dynamic computations described in Section 1.2 on page 2, with one caveat: intra-molecular forces need be computed as well, and included in the sum that yields the force $\mathbf{f}(a)$. This omission does not affect the lower bounds. In practice, this omission does not affect the upper bounds as well, as explained in Section 1.2 on page 2, since the computation of intra-molecular forces adds little overhead.

Note that the formulation used here does not encompass fast multipole algorithms.

An important feature of molecular dynamic simulations (as distinct from gravitational simulations) is that density is roughly constant. To simplify discussion, we shall make the following assumption:

Assumption 1 (constant density assumption)
Let $\rho = n/d^3$ be the average density of the system. Then there exist constants δ and ϵ, such that any cube or ball of volume $V \geq \delta$ contains at least $\rho(1-\epsilon)V$ and at most $\rho(1+\epsilon)V$ atoms.

We shall assume that $c^3 \gg \delta$, so that the system is homogeneous at the scale of the cutoff distance. With this assumption, then the number of inter-atomic forces acting on one atom is at most

$$(1+\epsilon)\rho\frac{4}{3}\pi c^3 \;=\; (1+\epsilon)\frac{4}{3}\pi(c/d)^3 n$$

and at least

$$(1-\epsilon)\rho\frac{4}{3}\pi c^3 \;=\; (1-\epsilon)\frac{4}{3}\pi(c/d)^3 n. \tag{1}$$

1.4 Typical Parameters

We shall analyze the complexity of the generic problem as function of the following parameters:

- n, the number of atoms

- c/d the relative cutoff distance; and
- p, the number of processors

Although the analysis is valid for any combination of values, it is worthwhile to keep in mind typical values for these parameters.

A key problem in molecular dynamics is protein folding: the process whereby a protein molecule in a water solution folds and acquire its unique three-dimensional shape. A typical protein folding simulation may involve one protein with a few hundreds residues, hence a few thousand atoms, within a cell that contains tens of thousands of water molecules, for a total of $\approx 32,000$ atoms, in a cell of dimension $D \approx 70$Å. A similar sized system can be used to study DNA decoding in simple situations. A larger cell with $d = 120$Å containing up to 200,000 atoms may be needed to simulate more complex reactions at membranes [4].

A typical cutoff distance would be $c \approx 10$Å. so that $c/d \approx \frac{1}{7}$. A different cutoff may be used for different parts of the algorithm. Note that, in the case of Lennard-Jones forces, the cutoff distance is derived from the decay rate of the relevant potentials, and is determined by the required level of accuracy. This level is presently fairly low, since the force fields used to represent intra-molecular interactions are fairly inaccurate, at present. In the case of Coulomb interactions, the cutoff is additionally set so as to balance the amount of work done in real-space and the amount of work done in k-space: a larger cutoff increases the amount of work done in real space but decreases the amount of work done in k-space. Thus, depending on the accuracy needed, and on the relative efficiency of direct computations and of k-space computations, we may have $7\text{Å} \leq c \leq 14\text{Å}$, so that $\frac{1}{10} \leq c/d \leq \frac{1}{5}$.

Molecular dynamic simulations are routinely done on parallel systems with a tens to hundreds of processors. The Blue Gene machine is expected to perform such simulations on up to 32,000 processors [2].

2 Programming Model and General Results

2.1 Programming Model

We are interested in parallel algorithms that solve the N-body computation with cutoff on a set $\mathcal{P} = \{1, \ldots, p\}$ of p processors, with minimal computation and communication. We analyze the problem using the postal model [3] and proceed in Section 6 on page 31 to consider specific communication topologies.

We assume that processors communicate via point-to-point messages. This can be implemented by send and receive operations, or put/get operations, etc. Formally, a *message* is of the form

$$m = (sender, receiver,\ v_1, \ldots, v_k)$$

where *sender* and *receiver* are processor indices, and $v_1, \ldots, v_k$ are the data communicated from sender to receiver. An *execution* consists of a tuple $\langle \mathcal{M}, \prec \rangle$ where $\mathcal{M}$ is a set of messages, and $\prec$ is a partial order on the messages; messages involving the same processor (as sender or receiver) are totally ordered.

We assume that the communication of k data items takes time $\ell + \tau k$; ℓ is a measure of communication latency, and $1/\tau$ is a measure of communication bandwidth. This simple model is adequate for many systems where the main communication bottleneck is the software overhead for communication calls. Formally, a *schedule* for an execution $\langle \mathcal{M}, \prec \rangle$ assigns a *starting time* $begin(m)$ and a *completion time* $end(m)$ to each message, with the following properties:

1. If $m_1 \prec m_2$ then $end(m_1) < begin(m_2)$.

2. If $m = (p, q, v_1, \ldots v_k)$ then $end(m) - begin(m) \geq \ell + \tau k$

The *communication complexity* of an execution is defined to be the minimum, over all valid schedules, of the maximum completion time of a message in this schedule.

2.2 Generic Parallel Algorithm for the N-body Problem with Cutoff

A parallel implementation of the N-body generic problem on p processors will allocate force computations and atom updates to the processors. Communications may be required to broadcast updated atom locations to the processors that compute forces involving these atoms and to collect and sum forces computed at distinct nodes.

Consider first the case where the coordinates of each atom are updated by one processor only, and each pairwise force is computed by one processor only. We use the following notation.

$p(a)$ is the processor that updates the coordinates of atom a; $p()$ is a mapping from $\mathcal{A}$ to $\mathcal{P}$.

$p(a, b)$ is the processor that computes the force between atom a and atom b, where $(a, b) \in \mathcal{E}$; $p(,)$ is a mapping from $\mathcal{E}$ to $\mathcal{P}$ such that $p(a, b) = p(b, a)$.

We further denote by $\mathcal{A}(i)$ the set of atoms updated by processor i and by $\mathcal{E}(i)$ the set of forces (atom pairs) computed at processor i. $B(a) = \cup_b p(a, b)$ is the *broadcast set* of atom a.

An algorithm for the solution of the N-body problem with cutoff consists of the following four phases:

Broadcast: For each atom a, broadcast the location $\mathbf{x}(a)$ of atom a from processor $p(a)$ to each processor $i \in B(a)$.

Compute: For each pair $(a, b) \in \mathcal{E}$, compute the force $\mathbf{f}_{ab}$ at processor $p(a, b)$.

Reduce: For each atom a, sum-reduce all computed forces $\mathbf{f}_{ba}$ and collect the sum $\mathbf{f}_a$ at processor $p(a)$.

Update: for each atom a, apply the update function $U()$ on the coordinates of atom a at processor $p(a)$.

A more general formulation is required to capture computations where the same force may be redundantly computed at several nodes, or the same atom coordinates may be redundantly updated at several nodes. While such redundancy will increase computation, it might lead to reduced communication. Such a general algorithm is described by the following mappings.

$P(a)$ is the set of processors that update the coordinates of atom a. If the coordinates atom a are updated at a unique processor, then we denote this processor by $p(a)$.

$P(a,b)$ is the set of processors that compute the force between atom a and atom b; $P(a,b) = P(b,a)$.

$B(i,a)$ is the set of processors to which processor i broadcasts the location of atom a. If a is held at a unique processor, then we omit the first argument i. $B(i,a) = \emptyset$ if $i \notin P(a)$ and $P(a,b) \subseteq \cup_i B(i,a)$.

$R(i,a)$ is the set of processors that contribute to the sum-reduction that yields the value of $\mathbf{f}_a$ at node $i \in P(a)$. If a is held at a unique processor, then we omit the first argument i. $R(i,a) = \emptyset$ if $i \notin P(a)$; if $i \in P(a)$ and $(a,b) \in \mathcal{E}$ then $P(a,b) \cap R(i,a) \neq \emptyset$.

The previous algorithm generalizes as follows:

Broadcast: For each atom a and each processor $i \in P(a)$, broadcast the location $\mathbf{x}(a)$ of atom a from processor i to each processor in the set $B(i,a)$.

Compute: For each pair $(a,b) \in \mathcal{E}$ compute the force $\mathbf{f}_{ab}$ at each processor $i \in P(a,b)$.

Reduce: For each atom a and each processor $i \in P(a)$, sum-reduce all forces $\mathbf{f}_{ba}$ computed at nodes in $R(i,a)$ and collect the sum at processor i.

Update: For each atom a and each processor $i \in P(a)$, update the coordinates of a at i.

One can see that, if the computation of the functions $\mathbf{f}()$ and $U()$ are assumed to be "black-box" operations, and locations and forces are handled as atomic values, then this is the most general form of a parallel implementation of the generic problem, with one qualification: We assume here that the set of processors that holds the coordinates of atom a at the end of the iteration is the same set that held the coordinates of a at the start of the iteration. One can further generalize by dropping this assumption. We suspect, but have no proof, that this last generalization can not lead to further improvements.

An optimal distribution of atom updates and force computations will depend on the set $\mathcal{E}$ of interacting atom pairs. As this set changes in time, the distribution has to be updated. However, the locations of the atoms changes relatively slowly in molecular simulations. Thus, we shall assume that the distributions $P(a)$ and $P(a,b)$ and the communication patterns $B(i,a)$ and $R(i,a)$ may depend on atom locations, while ignoring the overhead for computing these distributions and remapping atoms.

2.3 A Duality Result

When the coordinates of each atom are updated at a unique node, and each force is computed at a unique node, then the broadcast tree for the coordinates of an atom a has the same root and the same leaves as the reduce tree that sums forces acting on atom a. In such a case the reduce phase can use the same communication pattern as the broadcast operation, in a reverse order.

More formally, assume that the coordinates of each atom are available at a unique processor (forces may be computed at multiple locations). Let $(\mathcal{M}, \prec)$ be a communication execution that implements the broadcast phase of the algorithm. Each message is of the form

$$m_i = (s_i, r_i,\ \mathbf{x}(a_1^i), \ldots,\ \mathbf{x}(a_{k_i}^i))$$

Note that a message may contribute simultaneously to multiple concurrent broadcasts.

We define a communication execution $(\bar{\mathcal{M}}, \succ)$ for the reduce phase by reversing the precedence relation and replacing each send by a receive and vice versa, as follows:

$\bar{\mathcal{M}} = \{\bar{m}\ :\ m \in \mathcal{M}\}$.

$\bar{m}_1 \prec \bar{m}_2$ iff $m_2 \prec m_1$.

If $m = (s, r,\ \mathbf{x}(a_1), \ldots,\ \mathbf{x}(a_k))$ then $\bar{m} = (r, s,\ \mathbf{f}(a_1), \ldots,\ \mathbf{f}(a_k))$. $\mathbf{f}(a)$ is the sum of all the forces acting on atom a that were previously computed or received at processor r.

Lemma 1
Assume that the execution $(\mathcal{M}, \prec)$ implements the broadcast phase of the algorithm. Then the execution $(\bar{\mathcal{M}}, \succ)$, defined above, is a valid implementation of the reduce phase of the algorithm.

Proof: For each broadcast message $m = (s, r,\ \mathbf{x}(a_1), \ldots\ \mathbf{x}(a_k))$ and each atom $a \in \{a_1, \ldots, a_k\}$, let $B(m, a)$ be the set of nodes to which node r broadcasts the position $\mathbf{x}(a)$ it received in message m. Let $\bar{m} = (r, s, \mathbf{f}(a_1), \ldots, \mathbf{f}(a_k))$ be the corresponding reduce message. We claim that, for each $a \in \{a_1, \ldots, a_k\}$, $\mathbf{f}(a)$ contains the sum of the forces acting on a computed locally or at nodes in $B(m, a)$.

The claim is true in the base case where m has no successors. In this case $B(m, a) = \emptyset$, $\bar{m}$ is the first reduce message sent by processor r, and $\mathbf{f}(a)$ contains the sum of the forces acting on a that were computed at node r.

Assume that the claim is true for all broadcast communications that follow $m = (s, r,\ \mathbf{x}(a_1), \ldots,\ \mathbf{x}(a_k))$ hence all reduce communications that precede $\bar{m}$. Let $m_1, \ldots, m_t$ be the succeeding messages sent by r that forward the position of atom a. Then $B(m, a) = \cup_j B(m_j, a)$. By induction, the message $\bar{m}_j$ contains the sum of all the forces acting on atom a that were computed at nodes in $B(m_j, a)$. Hence, by construction, m will send the sum of all forces acting on a that were computed locally or at nodes in $B(m_j, a)$, for $j = 1, \ldots, t$. Hence the claim holds for m.

Now, let $m_1, \ldots, m_k$ be the messages sent by node s that contain the position of atom a. This position will only reach processors in $\cup B(m_j, a)$, so that all forces acting on atom a must be computed at these nodes, or locally. It follows that node s will receive or compute locally all forces acting on a in the reduce execution.

□

It is easy to see that $(\bar{M}, \succ)$ has the same communication complexity as $(M, \prec)$. Thus, if the coordinates of each atom are updated at a unique processor, then an optimal reduce phase will have a communication complexity that is lesser or equal to the communication complexity of the broadcast phase. Conversely, if each force is computed at a unique node, then one can use a similar argument to show that the broadcast phase can be implemented with no more communication than the reduce phase. If follows that, if there are no redundant computations, then one can assume that the broadcast phase and the reduce phase are "reverse" of each other, and have the same communication complexity.

In practice, reductions are often more expensive than broadcasts: while communication hardware may provide direct support to multicast, it may not support floating point reduction. This may justify the use of a different pattern for reductions as for broadcasts, which will also require redundancy in computations. Our analysis does not capture this distinction. Also, the results discussed in this section hold for parallel computation models where communication is symmetric: communication from i to j is as efficient as communication from j to i. A more detailed analysis is required if the symmetry assumption does not hold.

3 Algorithms

Plimpton presents in [12] three parallel algorithms for molecular dynamics:

Atom Decomposition: Each processor is allocated n/p atoms. The processor computes the forces acting on these atoms and updates their coordinates.

Space Decomposition Each processor is allocated a subvolume of the simulation cell. The processor handles the atoms in this subvolume.

Force Decomposition: The computed forces are evenly partitioned among processors.

We analyze in detail below the last two of these algorithms and present next a new algorithm that combines the best features of these two algorithms.

3.1 Space decomposition

We assume that p is perfect cube, to simplify notation. The simulation cell $\mathbf{D}$ is decomposed into $p = p^{1/3} \times p^{1/3} \times p^{1/3}$ cubic subcells $C(i,j,k)$, each of dimension $\tilde{d} = d/p^{1/3}$:

$$C(i,j,k) = [i\tilde{d}, (i+1)\tilde{d}) \times [j\tilde{d}, (j+1)\tilde{d}) \times [kd, (k+1)\tilde{d}),$$

where $0 \leq i,j,k < p^{1/3}$. The atoms in each subcell are allocated to one processor. Each processor computes all forces on atoms that have been allocated to it. We assume that $\tilde{d}^3 > \delta$, so that each subcell contains at most $(1+\epsilon)n/p$ atoms. We denote each processor by a triplet (i,j,k), $0 \leq i,j,k < p^{1/3}$. We have

$$p(a) = (\lfloor x_1(a)/\tilde{d} \rfloor, \lfloor x_2(a)/\tilde{d} \rfloor, \lfloor x_3(a)/\tilde{d} \rfloor)$$

$$P(a,b) = \{p(a), p(b)\};$$

Note that, in this formulation, each force is computed twice and no advantage is taken of symmetry.

This algorithm distributes computation evenly: each processor computes at most

$$(1+\epsilon)^2 \frac{4}{3}\pi(c/d)^3 n^2/p$$

forces. The reduction phase of the algorithm requires no communication. We analyze below the broadcast phase of this algorithm.

We assume that the processors (= subcells) are logically organized in a cyclical 3-dimensional grid. (This is a virtual grid – we still assume that each processor can directly communicate with any other.) The broadcast phase requires that each processor (i,j,k) broadcasts the positions of its atoms to all processors on the grid in a (cyclic) ball of radius of $r = \lceil c/\tilde{d} \rceil = \lceil (c/d)p^{1/3} \rceil$, centered at (i,j,k). We describe below a broadcast scheme where each processor broadcasts its data to all processors in a (cyclic) cube of dimension $2r+1$, centered at (i,j,k). This cube contains the ball. This is illustrated in Figure 1.

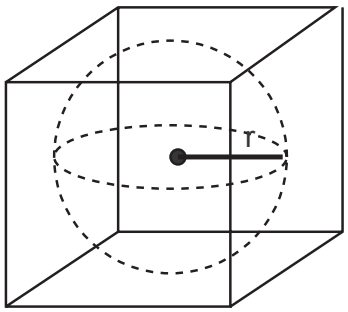

Figure 1: broadcast ball and broadcast cube

We denote by $A := Get((x,y,z),B))$ the circular shift operation that assigns to variable A at each processor (i,j,k) the value of variable B at processor $(i+x, j+y, k+z)$, where all additions are modulo $p^{1/3}$. These shifts can be implemented by get, put or send-receive operations. All communications for the algorithms described in this paper will be implemented only using this circular shift operation. Furthermore, we shall always use this shift operation in a SIMD mode, where all processors perform simultaneously the same shift. In some cases, we shall assume a "masked SIMD" mode, where all processors perform simultaneously the same shift, but some may send less data, or no data at all.

The broadcasts can be executed by a sequence of $6r$ circular shifts in the three dimensions, as described below.

Algorithm 1

let $A_0 =$ set of positions for atoms in local cell
assert $|A_0| \leq (1+\epsilon)n/p$

for $i := 1$ **to** r **do**
    $A_i := Get((1,0,0), A_{i-1})$
od
for $i := 1$ **to** r **do**
    $A_{-i} := Get((-1,0,0), A_{1-i})$
od
assert $A_{-r}, \ldots, A_r$ at processor (i,j,k)
        holds all atom positions from processors $(i+u, j, k), -r \leq u \leq r$
let $B_0 = \langle A_{-r}, \ldots, A_r \rangle$
assert $|B_0| \leq (1+\epsilon)(2r+1)n/p$
for $i := 1$ **to** r **do**
    $B_i := Get((0,1,0), B_{i-1})$
od
for $i := 1$ **to** r **do**
    $B_{-i} := Get((0,-1,0), B_{1-i})$
od
assert $B_{-r}, \ldots, B_r$ at processor (i,j,k)
        holds all atom positions from processors $(i+u, j+v, k), -r \leq u, v \leq r$
let $C_0 = \langle B_{-r}, \ldots, B_r \rangle$
assert $|C_0| \leq (1+\epsilon)(2r+1)^2 n/p$
for $i := 1$ **to** r **do**
    $C_i := Get((0,0,1), C_{i-1})$
od
for $i := 1$ **to** r **do**
    $C_{-i} := Get((0,0,-1), C_{1-i})$
od
assert $C_{-r}, \ldots, C_r$ at processor (i,j,k)
        holds all atom positions from processors
        $(i+u, j+v, k+w), -r \leq u, v, w \leq r$

The communication complexity for this broadcast algorithm is bounded by

$$6\ell r + \tau(1+\epsilon)(2r + 2r(2r+1) + 2r(2r+1)^2)n/p =$$
$$6\ell r + \tau(1+\epsilon)((2r+1)^3 - 1)n/p$$
$$\leq 6\ell(c/d)p^{1/3} + 8\tau(1+\epsilon)(c/d)^3 n + \text{ lower order terms.}$$

Note that this algorithm uses only nearest neighbor connections in the cyclical 3-dimensional grid.

3.2 An Improved Space Decomposition Algorithm

In the last algorithm, each processor (i,j,k) receives exactly once the position of each atom in a box of size $\tilde{d}(2r+1) \times \tilde{d}(2r+1) \times \tilde{d}(2r+1)$ surrounding $C(i,j,k)$. This includes many atom positions that processor (i,j,k) does not need. Figure 2 on the facing page illustrates this (in 2-dimensional space, for simplicity). The volume that contains atom

positions needed at the processor that owns the center cell is shown on the left. The volume that contains the positions actually transfered to that processor is shown on the right.

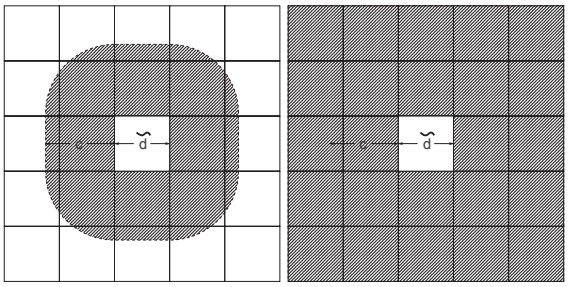

Figure 2: communication required (left) and performed (right)

Let $\mathrm{D}(a,(i,j,k))$ be the distance between the location of atom a and cell $C(i,j,k)$

$$\mathrm{D}(a,(i,j,k)) = \inf_{\mathbf{x}\in C(i,j,k)} \tilde{\mathrm{D}}(\mathbf{x}(a),\mathbf{x}).$$

An explicit definition can be given for that function. Define

$$\mathrm{D}(x,i) = \begin{cases} \min(|x - i\tilde{d}|, |x + d - (i+1)\tilde{d}|) & \text{if } x \le i\tilde{d}, \\ 0 & \text{if } i\tilde{d} < x \le (i+1)\tilde{d}, \\ \min(|(i+1)\tilde{d} - x|, |d - x + i\tilde{d}|) & \text{if } (i+1)\tilde{d} < x. \end{cases}$$

Then

$$\mathrm{D}(a,(i,j,k)) = \left(\mathrm{D}(x_1(a),i)^2 + \mathrm{D}(x_2(a),j)^2 + \mathrm{D}(x_3(a),k)^2\right)^{\frac{1}{2}}.$$

We make the following two observations:

1. Processor (i,j,k) does not need the position of atom a if $\mathrm{D}(a,(i,j,k)) \ge c$.
2. If processor (i',j',k') gets the position of atom a from a neighbor processor (i,j,k) during the broadcast, then $\mathrm{D}(a,(i',j',k')) > \mathrm{D}(a,(i,j,k))$: the position of an atom is routed to cells that are increasingly remote from the atom.

Thus, if processor (i,j,k) does not need the position of atom a, then this position is neither needed at processors that receive the position of atom a from processor (i,j,k). Therefore, one can filter out unnecessary position transfers by applying the following rule:

> The position of atom a is shifted from processor (i,j,k) to processor (i',j',k') only if $d(a,(i',j',k')) < c$

If this rule is applied, then each processor (i,j,k) will receive only the positions of atoms a such that $\mathrm{D}(a,(i,j,k)) < c$. As indicated in Figure 2, these atoms are contained in a shell of thickness c around $C(i,j,k)$. In 2-dimensional space, the shell is the disjoint union of 4 quadrants of a circle of radius c and four rectangles of dimension $c \times \tilde{d}$. Similarly, in 3-dimensional space, this shell is the disjoint union of

- 8 octants of a ball of radius c, one for each of the corners of the cube $C(i,j,k)$;
- 12 quadrants of a cylinder of radius c and height $\tilde{d}$, one for each of the vertices of the cube; and
- 6 boxes of dimension $\tilde{d} \times \tilde{d} \times c$, one for each face of the cube.

The total volume of this shell is

$$\frac{4}{3}\pi c^3 + 3\pi c^2 \tilde{d} + 6c\tilde{d}^2 = \frac{4}{3}\pi c^3 + 3\pi c^2 d p^{-1/3} + 6cd^2 p^{-2/3}.$$

Therefore, the number of atom positions received by each processor is at most

$$(1+\epsilon)n\left(\frac{4}{3}\pi(c/d)^3 + 3\pi(c/d)^2 p^{-1/3} + 6(c/d)p^{-2/3}\right)$$

The communication complexity is at most

$$6\ell\left\lceil (c/d)p^{\frac{1}{3}}\right\rceil + \tau(1+\epsilon)n\left(\frac{4}{3}\pi(c/d)^3 + 3\pi(c/d)^2 p^{-1/3} + 6(c/d)p^{-2/3}\right)$$

There are two cases of interest.

Low parallelism where the cutoff distance is smaller than the subcell size: $c < \tilde{d} = d/p^{1/3}$, or $p < (d/c)^3$. In our typical problem, where $d = 70$Å and $c = 7$Å, this means $p < 1000$. This encompasses the large majority of current runs.

In the low parallelism case, then each cell will communicate only with its 26 neighbors. The algorithm performs two shifts in each dimension, for a total of six shifts. The total communication time is bounded by

$$6\ell + \tau(1+\epsilon)n(c/d)p^{-\frac{2}{3}}\left(\frac{4}{3}\pi(c/d)^2 p^{\frac{2}{3}} + 3\pi(c/d)p^{\frac{1}{3}} + 6\right) < 6\ell + \tau(1+\epsilon)\alpha n(c/d)p^{-\frac{2}{3}}, \quad (2)$$

where $\alpha = \frac{4}{3}\pi + 3\pi + 6 < 20$.

If $p \ll (d/c)^3$ then we have a bound of

$$6\ell + 6\tau(1+\epsilon)n(c/d)p^{-\frac{2}{3}} + \text{ lower order terms.}$$

High parallelism where the cutoff distance is large, relative to the cell size: $c > d/p^{\frac{1}{3}}$, or $p > (d/c)^3$. This is relevant for massively parallel systems such as Blue Gene or to simulations that use a large cutoff distance. We can then bound communication time by

$$6\ell\lceil (c/d)p^{\frac{1}{3}}\rceil + \tau(1+\epsilon)n(c/d)^3\left(\frac{4}{3}\pi + 3\pi(d/c)p^{-\frac{1}{3}} + 6(d/c)^2 p^{-\frac{2}{3}}\right) < 6\ell\lceil (c/d)p^{\frac{1}{3}}\rceil + \tau(1+\epsilon)\alpha n(c/d)^3. \quad (3)$$

If $p \gg (d/c)^3$ then we have a bound of

$$6\ell\lceil (c/d)p^{\frac{1}{3}}\rceil + \tau(1+\epsilon)\frac{4}{3}\pi n(c/d)^3 + \text{ lower order terms.}$$

A naive implementation of filtering would require an expensive computation whenever an atom position is communicated, thus negating, in practice, the advantage accruing from saved communication. This overhead can be avoided or mitigated, in a variety of ways: e.g., filter thresholds can be precomputed and data can be packaged so that it is simple to drop from a message the positions that need not propagate further.

3.3 Load Balancing

When the number of processors is large, relative to the number of atoms, then the space decomposition method may lead to load imbalance. This happens in the "high parallelism" case, where $p \gg (d/c)^3$. In this case, however, the communication radius $r = p^{1/3}(c/d) \gg 1$. It is possible to improve the load balance without much increasing the communication by rebalancing atoms across neighbor cells. Assume that $p^{1/3} = q \times s$ We can divide the simulation cell $\mathbf{D}$ into $q \times q \times q$ "macrocells", each divided into $s \times s \times s$ "microcells". Atoms within a macrocell are evenly distributed across the s^3 microcells so as to improve load balance. Then, rather than broadcasting in a ball of radius $r = p^{1/3}(c/d)$ in the processor grid, it is now required to broadcast in a ball of radius $r+s$, in order to have each microcell collect the locations of atoms within cutoff distance of the atoms in the microcell. One can trade off load balance against communication by choosing a larger value for s, to improve load balance, or a smaller one, to reduce communication.

Atoms can be distributed to microcells within a macrocell randomly, for good load balance. Alternatively, one can use a nested bisection algorithm [8] to divide atoms evenly among microcells, while approximately preserving relative distances.

3.4 Force decomposition

We present here a version of force decomposition similar to that used in [2].

Assume that p is perfect square. We consider the processors to be organized into a $\sqrt{p} \times \sqrt{p}$ logical 2-dimensional grid. Processors are labeled (i,j), $0 \le i,j < \sqrt{p}$. Atoms are allocated evenly to processors. Each processor holds n/p atoms.

Let $A(i,*) = \cup_j A(i,j)$ and let $A(*,j) = \cup_i A(i,j)$. We allocate to processor (i,j) all forces due to interactions of atoms in $A(i,*)$ with atoms in $A(*,j)$. We thus have

If $p(a) = (i,j)$ then $B(a) = \{(i,0),\ldots,(i,\sqrt{p}-1),(0,j),\ldots(\sqrt{p}-1,j)\}$. I.e., the positions of the atoms held at a processor (i,j) are broadcast to all processors in row i and all processors in column j. After this broadcast phase each processor (i,j) holds the positions of the atoms in $A(i,*) \cup A(*,j)$

If $p(a) = (i,j)$, $p(b) = (i',j')$ and $ab \in \mathcal{E}$ then $P(a,b) = \{(i,j'),(i',j)\}$. These are the two processors that holds both the positions of a and of b, after the broadcast. This is illustrated in Figure 3 on the following page.

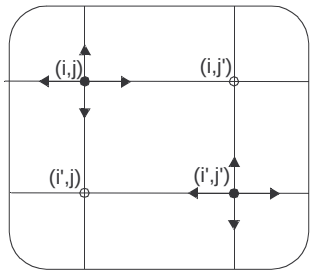

Figure 3: Communication pattern for force decomposition

Reductions can proceed either on columns, or on rows. In the former case we have, if $p(a) = (i, j)$, then $R(a) = \{(0, j), \ldots, (\sqrt{p} - 1, j)\}$.

We describe below in Section 3.6 on page 19 a randomized allocation of atoms to processors that results, with overwhelming probability, into a balanced partition of forces to processors, so that each processor computes $\Theta((c/d)^3 n^2/p)$ inter-atomic forces.

We assume that the processors are organized in a virtual 2-dimensional mesh with wraparound connections, and use $A := Get((x, y), B)$ to denote a cyclic shift in that mesh.

The row and column broadcasts can be executed by $\sqrt{p}$ successive cyclic shifts, each involving n/p atoms, as described below.

Algorithm 2

let $A_0 =$ set of atom positions that are locally available
for $m := 1$ **to** $\sqrt{p} - 1$ **do**
    $A_m := Get((1, 0), A_{m-1})$
od
assert $A_0, \ldots, A_{\sqrt{p}-1}$ at processor (i, j)
        contains the positions of atoms in $A(*, j)$
let $B_0 = A_0$
for $m := 1$ **to** $\sqrt{p} - 1$ **do**
    $B_m := Get((0, 1), B_{m-1})$
od
assert $B_0, \ldots, B_{\sqrt{p}-1}$ at processor (i, j)
        contains the positions of atoms in $A(i, *)$

The communication complexity for the broadcast phase is bounded by

$$2\ell\sqrt{p} + 2\tau\sqrt{p}(n/p) \;=\; 2\ell\sqrt{p} + 2\tau n/\sqrt{p}$$

The reduce phase requires only communication on column – the "reverse" of half of the broadcast algorithm. The communication complexity for the reduce phase is $\ell\sqrt{p} + \tau n/\sqrt{p}$

and the total communication complexity is

$$3\ell\sqrt{p} + 3\tau n/\sqrt{p} \tag{4}$$

The force decomposition algorithm computes most forces twice. This can be avoided by selecting only one of the two possible locations for the computation of each force. However, if this is done, then reductions will have to occur on both rows and columns.

Our analysis has ignored the overhead of identifying, at each processor, the $\Theta((c/d)^3 n^2/p)$ pairs of atoms within the cutoff distance out of $(n/\sqrt{p}) \times (n/\sqrt{p}) = n^2/p$ pairs available at the processor after the broadcast phase. Testing each pair would require too much computation. However, it is possible to use faster algorithms. Given n points and a cutoff distance, one can find all m pairs within the cutoff distance in time $O(n \log n + m)$, rather than $O(n^2)$ [6]; given the constant density assumption, a simple $O(n + m)$ average complexity algorithm can be designed. Furthermore, as we assume that atom locations change slowly, it is possible to precompute the set of interacting pairs – using a slightly increased cutoff distance, so that the same precomputed structure can be used for many iterations. Therefore, we shall ignore the overhead of identifying interacting atom pairs.

3.5 Mapping to 3-dimensional Mesh

We reformulate in this section the force decomposition algorithm assuming a virtual 3-dimensional grid of processors, rather than a 2-dimensional grid. This will be useful in order to understand the hybrid algorithm described in Section 4 on page 21. The reformulation is derived from an embedding of the 2D mesh into the 3D mesh. Let $r = p^{1/6}$. We assume, to simplify notation, that r is integer. Let (i, j) be a processor in the 2D mesh, where $0 \leq i, j < r^3$. Let $i_1 = i \div r^2$ and let $i_2 = i \bmod r^2$, so that $i = i_1 r^2 + i_2$, where $0 \leq i_1 < r$ and $0 \leq i_2 < r^2$. Similarly, let $j_1 = j \div r^2$ and let $j_2 = j \bmod r^2$. We then map node (i, j) in the 2D grid to node $(i_2, j_2, i_1 r + j_1)$. It is easy to see that this mapping is a bijection.

If processor (i, j) in the 2D mesh is mapped to processor (i', j', k') in the 3D mesh then the column $(*, j)$ is mapped onto the processor set $(u, j', rv + (k' \bmod r))$, $0 \leq u < r^2$, $0 \leq v < r$. Thus, to implement the row broadcast, we need a broadcast in the first dimension of the 3D grid, and a segmented broadcast in the third dimension, over subsets of the form $(i, j, k + ur), 0 \leq u < r$. The first broadcast can be implemented by a cyclic shift in the first grid dimension. The segmented broadcast can be implemented by a cyclic shift in the third dimension, with hops of length r. This is illustrated in Figure 4 on the next page, line A, for $r = 4$. Similarly, the row $(i, *)$ is mapped onto the set $(i', u, (k' \div r^2)r + v)$, $0 \leq u < r^2$, $0 \leq v < r$. Thus, to implement the row broadcast, we shall need a broadcast in the second dimension of the 3D grid, and a segmented broadcast over subsets of the form $(i, j, kr + u), 0 \leq u < r$. This segmented broadcast can be implemented by segmented cyclic shifts, as shown in Figure 4 on the following page, line B. However, if we do so, then the communication pattern is not SIMD, anymore: different nodes shift data with different displacements. We can, instead, use regular shifts in both directions, as shown in Figure 4 on the next page, lines C1 and C2: after 4 shifts in both directions, the broadcast is complete. This, however will result in twice as many messages and twice as much data being sent. If we filter communication so that no data is sent across segment boundaries (marked with dotted arrows in Figure 4 on the following page, lines C1 and C2), then we

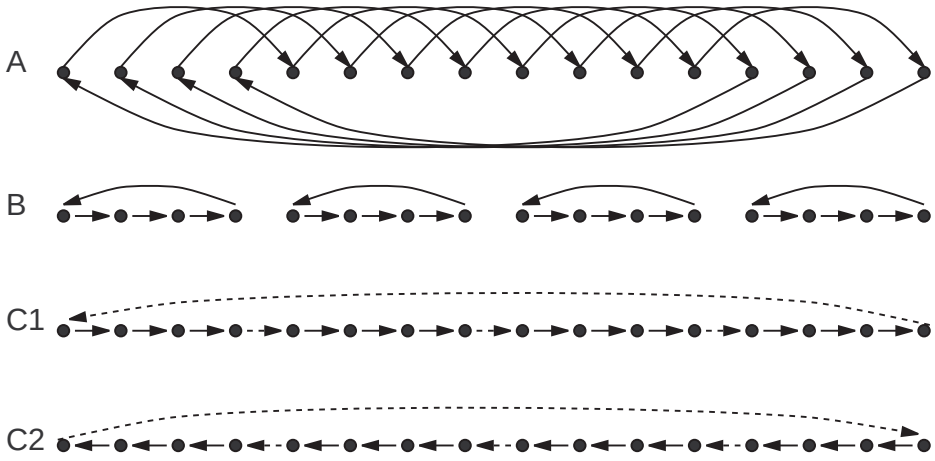

Figure 4: Communication pattern for segmented broadcasts

are back to a situation where each node sends and receives $\approx r$ messages: the i-th node in a group of r ($0 \leq i < r$) will receive i messages from its left and $r - i - 1$ from its right, for a total of $r - 1$. It will send $r - i$ messages to its left, if $i > 0$ and $i + 1$ messages to its right, if $i < r - 1$, for a total of $r + 1$, if $0 < i < r - 1$, or 1, if $i = 0$ or $i = r - 1$.

The resulting broadcast algorithm is shown below – without the last improvement that results from filtering.

Algorithm 3

2D column broadcast
let $A_0 =$ set of positions that are locally available
for $i := 1$ **to** $r^2 - 1$ **do**
    $A_i := Get((1, 0, 0), A_{i-1})$
od
let $B_0 = \langle A_0, \ldots, A_{r^2-1} \rangle$
for $i := 1$ **to** $r - 1$ **do**
    $B_i := Get(0, 0, r), B_{i-1})$
od
2D row broadcast
let $C_0 = A_0$
for $i := 1$ **to** $r^2 - 1$ **do**
    $C_i := Get((0, 1, 0), C_{i-1})$
od
let $D_0 = \langle C_0, \ldots, C_{r^2-1} \rangle$
for $i := 1$ **to** $r - 1$ **do**

$D_i := Get((0,0,1), D_{i-1})$
od
for $i := 1$ **to** $r-1$ **do**
$D_{-i} := Get((0,0,-1), D_{1-i})$
od

The communication complexity for the broadcast phase is

$$\ell(2(r^2-1)+3(r-1)) + \tau(n/p)(2(r^2-1)+3(r-1)r^2)$$
$$= 2\ell p^{1/3} + 3\tau n/\sqrt{p} + \text{ lower order terms}$$

If we assume that the communication model has the flexibility to implement the communication pattern of Figure 4 on the preceding page, line B, then we have a communication complexity of

$$2\ell((r^2-1)+(r-1)) + 2\tau\frac{n}{p}((r^2-1)+(r-1)r^2)$$
$$= 2\ell p^{1/3} + 2\tau n/\sqrt{p} + \text{ lower order terms}$$

The same result holds (up to lower order terms) if we assume that the communication model has the flexibility needed to take advantage of message filtering at block boundaries. (Note that the abstract model introduced in Section 2.1 on page 6 does not require SIMD communication.)

3.6 Probabilistic Analysis

We show in this subsection that randomization can ensure, with overwhelming probability that the force decomposition algorithm is load balanced. The proof uses Chernoff bounds for tails of binomial distributions. This assumes that the average number of forces per processor is sufficiently large. Specifically, we assume that

$$\frac{n}{\log_2 n} \geq (d/c)^3 \frac{k\sqrt{p}}{\frac{8}{3}\pi(1-\epsilon)} \tag{5}$$

for some constant $k > 3/2$. This assumption is easily satisfied for typical problem parameters.

We assume that atoms are partitioned deterministically across rows: each row i is allocated as set A_i of $n/\sqrt{p}$ atoms. The atoms are next randomly distributed to columns. For a fixed atom a and a fixed column j, the number $X(a,j)$ of atoms b that interact with a and have been allocated to column j is the sum of $N = N(a)$ independent Poisson trials, where $\frac{4}{3}\pi(1+\epsilon)(c/d)^3 n \geq N \geq \frac{4}{3}\pi(1-\epsilon)(c/d)^3 n$, and each trial has probability of success $1/\sqrt{p}$. The expected value is $\mu = N/\sqrt{p}$.

Let $\eta = 1/n^k$. Then, using (5) on the current page, we have

$$\frac{\log_2(1/\eta)}{\mu} - 1 = \frac{k\log_2 n}{N/\sqrt{p}} - 1 \leq \frac{k\sqrt{p}\log_2 n}{\frac{4}{3}\pi(1-\epsilon)(c/d)^3 n} - 1 \leq 1.$$

We now use the Chernoff bound (4.11) on page 72 in [11], to obtain that

$$\Pr(X(a,j) > 2\mu) \leq \eta.$$

If $X(a,j) < 2\mu$ for each atom a and each column j, then the number of forces computed at processor (i,j) is bounded by

$$\begin{aligned}\sum_{a \in A_i} X(a,j) &\leq \sum_{a \in A_i} 2N(a)/\sqrt{p} \\ &\leq \frac{n}{\sqrt{p}} 2\frac{4}{3}\pi(1+\epsilon)(c/d)^3 n/sqrtp \\ &= \frac{8}{3}\pi(1+\epsilon)(c/d)^3 n^2/p.\end{aligned}$$

The probability of this occurring is at least

$$1 - n\sqrt{p}\eta \geq 1 - n^{3/2}\eta = 1 - n^{3/2-k}.$$

3.7 Force Decomposition vs. Space Decomposition

If we compare (2) and (3) on page 14 to (4) on page 17, we see that the space decomposition method generates less communication in the case where c is small relative to $d/p^{1/3}$. In particular, the space decomposition method is always superior in the low parallelism case ($p^{1/3} < d/c$). This is to be expected. The communication pattern of the force decomposition method does not depend at all on the cutoff distance c, while the space decomposition method takes advantage of locality and has less communication when c is small.

On the other hand, the space decomposition method generates less traffic when c is large. In particular, if $c \approx d/2$ (no cutoff), then, in the space decomposition method, each position is broadcast to each processor; in the force decomposition method, it is broadcast to $\Theta(\sqrt{p})$ processors. If we ignore the "latency" terms and focus on the "bandwidth" term, that is proportional to the communication volume, then the space decomposition method is superior when

$$(1+\epsilon)\frac{4}{3}\pi n(c/d)^3 < 3np^{-}\frac{1}{2},$$

or

$$p < \left(\frac{9}{4\pi(1+\epsilon)}\right)^2 (d/c)^6.$$

Using the typical problem parameters listed in Section 1.4 on page 5 ($d = 70$Å, $c = 10$Å), we see that the space decomposition method is likely to be superior up to computations using many thousands of processors. However, in order to provide a reliable estimate on the crosspoint between the two algorithms, one needs detailed numbers obtained from actual runs or from accurate simulations.

We proceed in the next section to define a hybrid algorithm that takes advantage of locality, like the space decomposition method, but avoids the expense of a broadcast to all processors, like the force decomposition method. The new algorithm is superior to both algorithms described in this section.

4 New Hybrid Method

Consider a partition of atoms to p processors such that each processor holds $\Theta(n/p)$ atoms, and each atom is updated by a unique processor. We say that two processors p and q *interact* if each holds atoms $a \in A(p)$ and $b \in A(q)$ such that $ab \in \mathcal{E}$. Interacting processors communicate, either directly or indirectly, during an iteration. With a space decomposition, each processor interacts with $O((c/d)^3 p)$ other processors. This is optimal, as each atom interacts with $\Omega((c/d)^3 n)$ other atoms, hence each processor interacts with

$$\Omega\left(\frac{(c/d)^3\, n}{n/p}\right) \;=\; \Omega\left((c/d)^3 p\right)$$

other processors. However, the broadcast scheme used in the algorithm, where each processor broadcasts its positions to all other processors it interacts with, is not optimal.

The broadcast scheme used by the force decomposition method has the property that for any two processors p_1 and p_2, there is a processor p_3 so that both p_1 and p_2 send all their data to p_3. Then p_3 can compute all interactions between atoms from p_1 and atoms from p_2. In general, one needs a communication scheme so that if processors p_1 and p_2 interact, then there is a processors p_3 so that both p_1 and p_2 send all their data to p_3. This is weaker than the condition satisfied by the space decomposition method. We are going to show in this section that such a communication scheme can be defined, for a space decomposition, so that each processor will broadcast its atom positions to only $O\left(\sqrt{(c/d)^3 p}\right)$ other processors, i.e., to the square root of the number of processors it interacts with. This is superior to the simple space decomposition method and equal to the force decomposition method when $c = \Omega(d)$.

We use the same allocation of atoms to processors as described for the space decomposition method. The simulation cell is divided into $p^{1/3} \times p^{1/3} \times p^{1/3}$ subcells, each containing at most $(1+\epsilon)n/p$ atoms. Each subcell is allocated to one processor. We assume that the processors are organized in a logical 3-dimensional cyclical grid. Atoms at processor (i,j,k) interact only with atoms at processors within (cyclical) Euclidian distance $r = \lceil (c/d)p^{1/3} \rceil$ on the grid. Let $\tilde{r} = \lceil \sqrt{r+1} \rceil$. If $m \le r$ then $m = m_1 \tilde{r} - m_2$, where $0 < m_i \le \tilde{r},\ i = 1,2$.

The algorithm will have each processor (i,j,k) broadcast the positions of the atoms it holds to a set $B(i,j,k)$ of processors, where

$$\begin{aligned} B(i,j,k) = \{&(i+u, j, k+w_2),\ -r \le u \le \tilde{r},\ 0 < w_2 \le \tilde{r}\} \\ &\cup \{(i, j+v, k+rw_1),\ -r \le v \le r,\ 0 < w_1 \le \tilde{r}\}. \end{aligned}$$

This is similar to the broadcast pattern used in Algorithm 3 on page 18, the 3D version of the force decomposition algorithm, except that the communication is localized to a neighborhood of the broadcasting node.

Claim 1
If processor (i,j,k) interacts with processor (i',j',k'), then there exist a processor $(\hat{i},\hat{j},\hat{k})$ such that $(\hat{i},\hat{j},\hat{k}) \in B(i,j,k) \cap B(i',j',k')$

Proof: Assume w.l.o.g. that $k = k' + w$, where $0 \leq w \leq r$ (all equalities are $\bmod p^{1/3}$). Then $k - k' = w_1\tilde{r} - w_2$, where $0 < w_1, w_2 \leq \tilde{r}$. Let $\hat{k} = k + w_2 = k' + w_1\tilde{r}$. Let $\hat{i} = i'$, and let $\hat{j} = j$. Then $(\hat{i}, \hat{j}, \hat{k}) = (i + u, j, k - w_2)$, where $0 < w_2 \leq \tilde{r}$ and $|u| = |i - i'| \leq r$; and $(\hat{i}, \hat{j}, \hat{k}) = (i', j + v, k' + w_1\tilde{r})$, where $|v| = |j - j'| \leq r$ and $0 < w_1 \leq \tilde{r}$.

□

The computation of forces between atoms in $A(i,j,k)$ and atoms in $A(i',j',k')$ can be evenly allocated to the nodes in $B(i,j,k) \cap B(i',j',k')$ so that each node will compute at most $(1+\epsilon)^2(c/d)n^2/p$ forces. We describe below the broadcast phase of this algorithm. Since the coordinates of each atom are maintained by a unique processor, the results in Section 2.3 on page 9 imply that reduce phase can be implemented with the same cost as the broadcast phase.

Algorithm 4

let A_0 be set of positions for atoms in local cell
assert $|A_0| \leq (1+\epsilon)n/p$
for $i := 1$ **to** r **do**
    $A_i := Get((1,0,0), A_{i-1})$
od
for $i := 1$ **to** r **do**
    $A_{-i} := Get((-1,0,0), A_{1-i})$
od
assert $A_{-r}, \ldots, A_r$ at processor (i,j,k) holds all atom positions
        from processors $(i+u, j, k), -r \leq u \leq r$
let $B_0 = \langle A_{-r}, \ldots, A_r \rangle$
assert $|B_0| \leq (1+\epsilon)(2r+1)n/p$
for $i := 1$ **to** $\tilde{r}$ **do**
    $B_i := Get((0,0,1), B_{i-1})$
od
assert $B_0, \ldots, B_{\tilde{r}}$ at processor (i,j,k) holds all atom positions
        from processors $(i+u, j, k+v), -r \leq u \leq r, 0 < v \leq \tilde{r}$
for $i := 1$ **to** r **do**
    $A_i := Get((0,1,0), A_{i-1})$
od
for $i := 1$ **to** r **do**
    $A_{-i} := Get((0,-1,0), A_{1-i})$
od
assert $A_{-r}, \ldots, A_r$ at processor (i,j,k) holds all atom positions
        from processors $(i, j+u, k), -r \leq u \leq r$
let $C_0 = \langle A_{-r}, \ldots, A_r \rangle$
assert $|C_0| \leq (1+\epsilon)(2r+1)n/p$
for $i := 1$ **to** $\tilde{r}$ **do**
    $C_i := Get((0,0,r), C_{i-1})$
od
assert $C_0, \ldots, C_{\tilde{r}}$ at processor (i,j,k) holds all atom positions from

processors $(i, j+u, k+v\tilde{r}), -r \leq u \leq r, 0 < v \leq \tilde{r}$

The communication complexity for this broadcast pattern is

$$\ell(4r+2\tilde{r}) + \tau(1+\epsilon)(4r+2\tilde{r}(2r+1))n/p = \\ 4\ell(c/d)p^{1/3} \ + \ 4\tau(1+\epsilon)(c/d)^{3/2}n/\sqrt{p} \ + \ \text{lower order terms.}$$

The reduction phase can be implemented be reversing the broadcast communication pattern. The total communication complexity is

$$8\ell(c/d)p^{1/3} \ + \ 8\tau(1+\epsilon)(c/d)^{3/2}n/\sqrt{p} \ + \ \text{lower order terms.} \tag{6}$$

In the low parallelism case ($p < (d/c)^3$), then the hybrid algorithm has the same complexity, within a small constant factor, as the unimproved space decomposition algorithm. In this regime, the hybrid algorithm can be improved similarly to the space decomposition algorithm.

At the other extreme, when $c \approx d/2$ (no cutoff), the communication complexity of the hybrid algorithm is

$$4\ell p^{1/3} + \sqrt{8}\tau(1+\epsilon)n/\sqrt{p} + \text{ lower order terms}$$

This is the same, within a small constant factor, as the 3D version of the force decomposition algorithm (Algorithm 3 on page 18).

In the high parallelism regime ($p \gg (d/c)^3$) we can borrow the technique from Section 3.3 on page 15, in order to improve load balancing.

5 Lower Bounds

We prove in this section lower bounds on communication that show that the algorithm described in the previous section is optimal, among algorithms that perform minimal work. To simplify the (fairly tedious) derivations of this section we do not attempt to derive the best possible constants, or use the weakest preconditions.

Consider an instance of the N-body problem with cutoffs to be solved with p processors. We assume the following inequalities

$$n \geq p \tag{7}$$

$$d/c > \mu, \text{ for some constant } \mu > 3 \tag{8}$$

We use the notation introduced in Section 1.3 on page 3, with the following additions.

$n_i = |P(i)|$ is the number of atoms updated by processor i.

$e_i = |\mathcal{E}(i)|$ is the number of forces computed by processor i.

m_i is the communication volume at processor i, i.e., the number of atom positions sent or received plus the number of forces sent or received by processor i.

The following inequalities are satisfied

$$\sum_i n_i \geq n \tag{9}$$

$$\sum_i e_i \geq (1-\epsilon)(c/d)^3 n^2 \tag{10}$$

$$e_i \leq (n_i + m_i)^2 \tag{11}$$

The first inequality is obvious. The second follows from (1) on page 5. The third follows from the fact that the number of atom positions available at processor i is at most $n_i + m_i$, so that the number of forces that can be computed at this processor is bounded by

$$\binom{n_i + m_i}{2} < (n_i + m_i)^2.$$

We say that a set $L \subset A$ *c-isolates* set $M \subset A$ if, whenever $ab \in \mathcal{E}$, $a \in M$, $b \notin M$, then either $a \in L$ or $b \in L$. We define the *c-surface* of set M, $\mathrm{S_c}(M)$, to the least size of a set that c-isolates M.

Theorem 1
$m_i \geq \mathrm{S_c}(P(i))$.

Proof: Let L consist of all atoms a such that either processor i receives a force that acts on a or processor i receives the position of a. Clearly, $m_i \geq |L|$. Let $a4and4b$ be two atoms such that $ab \in \mathcal{E}$, $a \in P(i)$, $b \notin P(i)$ If processor i computes the force between a and b then it must receive the position of b. Otherwise, processor i must receive from another processor a sum of forces acting on atom a that contains the force $\mathbf{f}_{ab}$. Thus either $a \in L$ or $b \in L$. It follows that L c-isolates $P(i)$, so that $|L| \geq \mathrm{S_c}(P(i))$.

□

We can thus prove lower bounds on communication by proving lower bounds on the surface of the set of atoms maintained by each processor.

Theorem 2
There exist positive constants k, η and λ that depend on d and c so that if $\delta \leq c/2k$, then the following holds. Let M be a set of $m = |M| \leq \eta n$ atoms. Then

$$\mathrm{S_c}(M) \geq \lambda \min(m, m^{\frac{2}{3}} n^{\frac{1}{3}} c/d)$$

Before we prove this theorem, let us show how it can be used to to derive lower bounds on communication. We restrict ourselves to algorithms that are load balanced:

$$\forall i \; e_i \leq 2(c/d)^3 n^2/p$$

(The constant 2 is arbitrary – any small constant $> 1 + \epsilon$ will do.)

Theorem 3 (low parallelism)
There exists a constant p_0 such that, if $p_0 < p < (d/c)^3$, then

$$\max_i m_i > \lambda (c/d) n/p^{\frac{2}{3}}.$$

Proof: Since $\sum_i n_i \geq n$, then $n_i \geq n/p$, for some i. There are two cases to consider.

Case A: $\exists\, i$ such that $n/p \le n_i \le \eta n$. Then, by Theorem 2 on the preceding page,

$$m_i \ge \lambda \min(n/p,\, (n/p)^{\frac{2}{3}} n^{\frac{1}{3}} (c/d)) \;=\; \lambda (c/d) n/p^{\frac{2}{3}}.$$

Case B: $\exists\, i$ such that $n_i > \eta n$. As the algorithm is load balanced, then processor i computes at most $2n^2(c/d)^3/p$ forces. Since each atom interacts with at least $(1-\epsilon)n(c/d)^3$ other atoms, processor i does not compute all forces acting on at least

$$\eta n - \frac{2n^(c/d)^3/p}{(1-\epsilon)n(c/d)^3} = n(\eta - \frac{2}{(1-\epsilon)p}$$

atoms. Thus

$$m_i \ge n(\eta - \frac{2}{(1-\epsilon)p} \ge \lambda (c/d) n/p^{\frac{2}{3}},$$

for large enough p.

□

The lower bound in the theorem above matches the upper bound of (2) on page 14, within a constant factor. Thus, the space decomposition algorithm generates a minimal communication volume, in the low parallelism case. In this range, the space decomposition algorithm is identical to the hybrid algorithm who thus is optimal, as well.

Theorem 4 (high parallelism)
Let $p \ge (d/c)^3$. Then

$$\max_i m_i > \frac{1}{2}(1-\epsilon)^{\frac{1}{2}} (c/d)^{\frac{3}{2}} n/p^{\frac{1}{2}}.$$

Proof:

Case A: $\exists\, i$ such that

$$n_i \ge \frac{1}{2}(1-\epsilon)^{\frac{1}{2}} (c/d)^{\frac{3}{2}} n/p^{\frac{1}{2}}.$$

Then

$$\begin{aligned} m_i &\ge \lambda \min(n_i, n_i^{\frac{2}{3}} n^{\frac{1}{3}} (c/d)) \\ &\ge \lambda \min\left(((1-\epsilon)/4)^{\frac{1}{2}} (c/d)^{\frac{3}{2}} n/p^{\frac{1}{2}}, ((1-\epsilon)/4)^{\frac{1}{3}} (c/d)^2 n/p^{\frac{1}{3}} \right) \end{aligned}$$

But $(c/d)^{\frac{1}{2}} p^{\frac{1}{6}} \ge 1$, so that

$$\begin{aligned} m_i \ge & \lambda \min\left(((1-\epsilon)/4)^{\frac{1}{2}} (c/d)^{\frac{3}{2}} n/p^{\frac{1}{2}}, ((1-\epsilon)/4)^{\frac{1}{3}} (c/d)^{\frac{3}{2}} n/p^{\frac{1}{2}} \right) \\ = & ((1-\epsilon)/4)^{\frac{1}{2}} (c/d)^{\frac{3}{2}} n/p^{\frac{1}{2}}. \end{aligned}$$

Case B: $\forall i$,

$$n_i < \frac{1}{2}(1-\epsilon)^{\frac{1}{2}}(c/d)^{\frac{3}{2}}n/p^{\frac{1}{2}}.$$

Then (10) on page 24 and (11) on page 24 imply that

$$\max_i (n_i + m_i)^2 \geq \frac{1}{p}(1-\epsilon)(c/d)^3 n^2$$

so that

$$\max_i m_i \geq (1-\epsilon)^{\frac{1}{2}}(c/d)^{\frac{3}{2}}n/p^{\frac{1}{2}} - \frac{1}{2}(1-\epsilon)^{\frac{1}{2}}(c/d)^{\frac{3}{2}}n/p^{\frac{1}{2}} = \frac{1}{2}(1-\epsilon)^{\frac{1}{2}}(c/d)^{\frac{3}{2}}n/p^{\frac{1}{2}}.$$

□

The lower bound given in the last theorem matches the upper bound given in (6) on page 23, within a constant factor. Thus, the hybrid algorithm generates a minimal communication volume, in the high parallelism case.

We now turn to the proof of Theorem 2 on page 24. We first prove a continuous version of this theorem, for Euclidian spaces, next derive from it the discrete theorem.

We first introduce several definitions. We denote by $\mathrm{V}(U)$ the *volume* (i.e., Lebegue measure) of a set $U \subset \mathbb{R}^n$. This is well-defined when U is compact.

We denote by B_r^n the ball of radius r centered at the origin in $\mathbb{R}^n$; the superscript n will be supressed when obvious from the context. We have $\mathrm{V}(B_r^n) = v_n r^n$, where

$$v_n = \mathrm{V}(B_1^n) = \begin{cases} \frac{\pi}{m!} & \text{for } n = 2m, \\ \frac{2(2\pi)^m}{(2m+1)!} & \text{for } n = 2m+1. \end{cases}$$

Given two sets A and B in $\mathbb{R}^n$, we denote by $A + B$ the set

$$A + B = \{x + y \ : \ x \in A, y \in B\}.$$

Note that if A and B are compact then $A + B$ is compact.

We denote by $B(U, r)$ the set $B(U, r) = U + B_r$. It is easy to see that if U is compact then

$$B(U, r) = \{\mathbf{y} \in \mathbb{R}^n \ : \ \mathrm{D}(U, \mathbf{y}) \leq r\}.$$

We use the following inequality

Theorem 5 (Brunn-Minkowski Inequality)
Let A and B be two compact sets in $\mathbb{R}^n$. Then

$$\mathrm{V}^{\frac{1}{n}}(A + B) \geq \mathrm{V}^{\frac{1}{n}}(A) + \mathrm{V}^{\frac{1}{n}}(B).$$

Equality obtains only in the following three cases: (1) $\mathrm{V}(A + B) = 0$; (2) A or B consists of a single point; (3) A and B are similar.

The reader is refered to Chapter 2 of [5] for a proof to this inequality.

Corollary 1
Let U be a nonempty compact set in $\mathbb{R}^n$. Let r be the radius of a ball with the same volume as U, i.e. $\mathrm{V}(U) = \mathrm{V}(B_r)$. Then, for any $c > 0$, $\mathrm{V}(U, c) \geq \mathrm{V}(B_{r+c})$; equality obtains only if U is a ball.

This theorem is analogous to the well-known isoperimetric theorem that states that a ball has minimum surface for a given volume. Here, we consider the volume of a shell of a fixed thickness, and claim that the volume of the shell is minimized when it encases a ball. The classical isoperimetric result can be derived from the result in this corollary by considering the limit when the thickness of the shell goes to zero; see [5].

Proof: We have, by the Brunn-Minkowski inequality,

$$\begin{aligned}\mathrm{V}^{\frac{1}{n}}(U, c) = \mathrm{V}^{\frac{1}{n}}(U + B_c) &\geq \mathrm{V}^{\frac{1}{n}}(U) + \mathrm{V}^{\frac{1}{n}}(B_c) \\ &= \mathrm{V}^{\frac{1}{n}}(B_r) + \mathrm{V}^{\frac{1}{n}}(B_c) = v_n^{\frac{1}{n}} r + v_n^{\frac{1}{n}} c = v_n^{\frac{1}{n}}(r + c) = \mathrm{V}^{\frac{1}{n}}(B_{r+c}).\end{aligned}$$

Equality obtains if and only if U is similar to a ball, i.e., if and only if U is a ball.

□

Let U, V be subsets of $\mathbb{R}^n$. We say that V *c-isolates* U if

$$\forall\, \mathbf{x} \in U, \forall\, \mathbf{y} \notin U \ \mathrm{D}(\mathbf{x}, \mathbf{y}) < c \rightarrow \mathbf{x} \in V \vee \mathbf{y} \in V.$$

We define the *c-surface* $\mathrm{S_c}(U)$ of a compact set $U \subset \mathbb{R}^n$ to be the least volume of a set that c-isolates U.

Theorem 6
Let U be a compact set of volume $\mathrm{V}(U) = v_n r^n$, so that r is the radius of a ball of the same volume as U. Then

$$\mathrm{S_c}(U) \geq v_n(r^n - \max(0, (r - c)^n)).$$

I.e., the volume of a set V that c-isolates U is minimized when U is a ball and V is a shell extending from the surface of the ball to a depth of c.

Proof: Since V c-isolates $V \cup U$ we can assume w.l.o.g. that $V \subseteq U$. Let $U_1 = \{\mathbf{x} \in U : \forall \mathbf{y} \notin U \ \mathrm{D}(\mathbf{x}, \mathbf{y}) \geq c$ be the set of points in U that are at distance at least c from points outside U. If $U_1 = \emptyset$ then each point in U is at distance $< c$ from U^c, so that $V \supseteq U$, and $\mathrm{V}(V) \geq v_n r^n = \mathrm{V}(U)$.

Otherwise, if $U_1 \neq \emptyset$, let $U_2 = B(U_1, c)$. Let $\mathrm{V}(U_1) = v_n r_1^n$ and let $\mathrm{V}(U_2) = v_n r_2^n$. By the previous Corollary, $r_2 \geq r_1 + c$. It is easy to see that $V \supseteq U - U_1$ and $U \supset U_2$, so that $r \geq r_2 \geq r_1 + c$. Thus

$$\mathrm{V}(V) \geq \mathrm{V}(U) - \mathrm{V}(U_1) = v_n(r^n - r_1^n) \geq v_n(r^n - (r - c)^n).$$

□

We now return to the proof of theorem (2) on page 24

Proof: We need to prove that if L c-isolates M, $|L| = \ell$ and $|M| = m$, then

$$\ell \geq \lambda \min(m, m^{\frac{2}{3}} n^{\frac{1}{3}} (c/d)). \tag{12}$$

It is easy to see that if L isolates M then L isolates $L \cup M$. Therefore, it is sufficient to prove for the case where $L \subseteq M$.

We say that a set $U \subset \mathbb{R}^3$ is *δ-rectifiable* if there is a disjoint set of cubes $\{\hat{C}_1, \ldots, \hat{C}_r\}$ such that $U \subseteq \cup \hat{C}_i$ and a disjoint set of cubes $\{\breve{C}_1, \ldots, \breve{C}_s\}$ such that $U \supseteq \cup \breve{C}_i$, where all cubes are of dimension $\geq \delta$, and

$$2 \sum_{i=1}^{3} \mathrm{V}(\breve{C}_i) \geq \mathrm{V}(U) \geq \frac{1}{2} \sum_{i=1}^{r} \mathrm{V}(\hat{C}_i).$$

(The constant 2 in the definition is arbitrary and can be replaced by any constant > 1.)

If U is a δ-rectifiable set then the number of atoms with positions in U is at least $\frac{1}{2}(1-\epsilon) n \mathrm{V}(U)/D^3$ and at most $2(1+\epsilon) n \mathrm{V}(U)/D^3$.

We first prove the corollary ignoring cyclicality: we assume that each atom in M is at least $c/2$ apart from the boundaries of the cell $\mathbf{D}$. We consider the noncyclical cell $\mathbf{D}'$ of dimension $d + c$, obtained by adding a boundary of thickness $c/2$ around the original cell. Then $L \subseteq M$ isolates M in the cyclical cell $\mathbf{D}$, where cyclical distance $\tilde{\mathrm{D}}(,)$ is used, iff it isolates M in the noncyclical cell $\mathbf{D}'$, where noncyclical distance $\mathrm{D}(,)$ is used.

Let

$$U = \bigcup_{a \notin M} B(\mathbf{x}(a), k\delta),$$

where $k > 1$ is a constant to be defined later. Let

$$V = B(U, C - k\delta) - U$$

One can verify that the sets U^c (the complement of U) and V are δ-rectifiable, for k large enough (independent of the atom numbers and locations) and for $c \geq 2k\delta$. This defines k.

Let b be an atom such that $\mathbf{x}(b) \in V$. By the definition of V, we have $\mathbf{x}(b) \notin U$. This implies, by the definition of U, that $b \in M$.

By the definition of V, there exists a point $\mathbf{y} \in U$ such that $\mathrm{D}(\mathbf{x}(b), \mathbf{y}) < c - k\delta$. But, if $\mathbf{y} \in U$ then there exists an atom $a \notin M$ such that $\mathrm{D}(\mathbf{x}(a), \mathbf{y}) < k\delta$. Thus $\mathrm{D}(\mathbf{x}(a), \mathbf{x}(b)) < c$. It follows that $b \in L$. Thus all atoms with positions in V are in L. It follows that

$$\ell \geq \frac{1}{2}(1-\epsilon) n \mathrm{V}(V)/d^3. \tag{13}$$

Let $M_1 = M - L$ and let $m_1 = m - \ell = |M_1|$. If $a \in M_1$, then $a \notin L$, so that $\mathbf{x}(a) \notin V$. Since L c-isolates M_1 then for any atom $b \notin M$, $\mathrm{D}(a, b) > c > k\delta$. This implies that $\mathbf{x}(a) \notin U$. Thus, U^c contains the positions of all atoms in M_1. It follows that

$$m - \ell \leq 2(1+\epsilon) n \mathrm{V}(U^c)/d^3 \tag{14}$$

By definition, V $(c-k\delta)$-isolates U, hence $(c-k\delta)$-isolates U^c. It follows, by Theorem 6 on page 27, that

$$\mathrm{V}(V) \geq \frac{4}{3}\pi(r^3 - \max(0,(r-c+k\delta)^3)), \tag{15}$$

where $\mathrm{V}(U^c) = \frac{4}{3}\pi r^3$. The result now follows by putting (13), (14) and (15) on the current page together. We have

$$\ell \geq \alpha\frac{n}{d^3}(r^3 - \max(0,(r-d+k\delta)^3))$$

and

$$m - \ell \leq \beta\frac{n}{d^3}r^3$$

for some positive constants α and β. We consider 3 cases:

Case A: $\ell \geq m/2$. Then (12) on the facing page follows, if $\lambda \leq \frac{1}{2}$.

Case B: $\ell < m/2$ and $r \leq 2(c-k\delta)$. Then $r-(c-k\delta) \leq r/2$, so that

$$\ell \geq \alpha\frac{n}{d^3}\frac{7}{8}r^3.$$

Also,

$$\frac{m}{2} < m - \ell \leq \beta\frac{n}{d^3}r^3,$$

so that

$$\ell \geq \frac{7\alpha}{16\beta}m$$

and (12) on the preceding page follows, if $\lambda \leq \frac{7\alpha}{16\beta}$.

Case C: $\ell < m/2$ and $r > 2(c-k\delta)$. Let $f(x) = 3 - 3x + x^2$. Then $f'(x) = -3 + 2x < 0$, if $x < 1.5$. Thus, if $x < x_0 \leq 1.5$ then $f(x) > f(x_0)$. Thus, since$(c-k\delta)/r < \frac{1}{2}$, then

$$\begin{aligned} r^3 - (r-(c-k\delta))^3 &= r^2(C-k\delta)\left(3 - 3\frac{c-k\delta}{r} + \left(\frac{c-k\delta}{r}\right)^2\right) \\ &> r^2(c-k\delta)((3-3(0.5)+(0.5)^2) = 1.75r^2(c-k\delta) > 0.875r^2c. \end{aligned}$$

Thus,

$$\ell > 0.875\alpha\frac{n}{d^3}r^2c$$

and

$$\frac{m}{2} < m - \ell \leq \beta\frac{n}{d^3}r^3$$

so that

$$\ell > 0.875\alpha \frac{n}{d^3} c \left(\frac{md^3}{2\beta n}\right)^{\frac{2}{3}} = \frac{0.875\alpha}{(2\beta)^{\frac{2}{3}}}(c/d)m^{\frac{2}{3}}n^{\frac{1}{3}}.$$

(12) on page 28 follows, if

$$\lambda \leq \frac{0.875\alpha}{(2\beta)^{\frac{2}{3}}}.$$

We turn now to the cyclic case, and drop the assumption that the atoms in M are $c/2$ apart from the cell boundary. Let $B \subseteq M$ be the set of atoms in M that are at distance $< c/2$ from the boundaries of the cell $\mathbf{D}$.

We assume that

$$b = |B| \leq 3m(c/d) \tag{16}$$

Let $\widehat{M} = M - B$. Then atoms in $\widehat{M}$ are at least $c/2$ apart from the cell boundaries. Furthermore, $B \cup L$ isolates $\widehat{M}$. It follows that

$$\ell + b \geq \lambda \min((m-b), (m-b)^{\frac{2}{3}}n^{\frac{1}{3}}(c/d)) \tag{17}$$

Let

$$\alpha = \frac{\mu + 3}{\mu - 3}$$

Then $\alpha > 0$.

We consider two cases

Case A:

$$\ell + b \geq \alpha(m-b)$$

Then, using (16) on the current page, we have

$$\ell > \alpha m - (1+\alpha)b \geq m(\alpha - 3(1+\alpha)c/d) \geq m(\alpha - 3(1+\alpha)\mu) = m.$$

Thus, (12) on page 28 holds, if $\lambda' \leq 1$.

Case B:

$$\ell + b < \alpha(m-b)$$

Then, if $\lambda \leq \alpha$, (16) on the current page and (17) on this page imply that

$$\ell + b \geq \lambda(m-b)^{\frac{2}{3}}n^{\frac{1}{3}}c/d \geq \lambda(1-3c/d)^{\frac{2}{3}}m^{\frac{2}{3}}n^{\frac{1}{3}}c/d \geq \lambda(1-3/\mu)^{\frac{2}{3}}m^{\frac{2}{3}}n^{\frac{1}{3}}c/d$$

If also $b < \frac{1}{2}\ell$ then

$$\ell \geq \frac{1}{2}\lambda(1-3/\mu)^{\frac{2}{3}}m^{\frac{2}{3}}n^{\frac{1}{3}}c/d$$

and (12) on page 28 holds, if $\lambda' \leq \frac{1}{2}\lambda(1-3/\mu)^{\frac{2}{3}}$.

Otherwise, if $b \geq \frac{1}{2}\ell$, then

$$9m(c/d) \geq 3b \geq b+\ell \geq \lambda(1-3/\mu)^{\frac{2}{3}}m^{\frac{2}{3}}n^{\frac{1}{3}}C/D$$

so that

$$m \geq (\lambda/9)^3(1-3/\mu)^2 n$$

Thus the corollary holds, with

$$\eta = (\lambda/9)^3(1-3/\mu)^2.$$

We assumed that $|B| \leq 3m(c/d)$. We will now reduce the general case to a case where this assumption holds. Since $|M| = m$, there must be a horizontal slice of the cell $\mathbf{D}$, of thickness c, that contains no more than $m(c/d)$ atoms from M. We can cyclically shift all atoms in cell $\mathbf{D}$ so that this slice is centered around the horizontal face of $\mathbf{D}$, with no change in the relative positions of the atoms. The same applies to each of the other two dimensions. After the shifts, we have $|B| \leq 3m(c/d)$.

□

6 Mesh Topology

We have so far assumed a machine where each node can communicate directly with each other node, and ignored possible network congestion. This is an acceptable approximation for small and medium sized parallel systems that use multistage networks or other low congestion networks. However, for very large machines such as Blue Gene [2] this type of interconnect may be too onerous. Indeed, Blue Gene plans to use a 3D mesh. It is important to understand how well the algorithms described in this paper map to such a topology.

We also capture in this section another feature of Blue Gene (and of many other relevant systems); namely, that each node is a multiprocessor, so that multiple communications may be supported simultaneously at each node. We assume a multiprocessing factor of m at each node.

We represent a parallel system with a specific interconnect by a directed graph $G = \langle V, E\rangle$, where $V = \{1, \ldots, p\}$ is the set of nodes E is the set of directed edges (links). (We assume here a direct interconnect, where each nodes holds a processor and a switch. The formalism can be easily extended to indirect interconnects, where switching nodes and processor nodes are distinct.)

We modify the formalism of Section 2.1 on page 6 as follows.

Let $CE = \langle \mathcal{C}, \prec\rangle$ be a communication execution. As we assume that each node is a multiprocessor, we do not assume anymore that $\prec$ totally orders communications at each node.

A *routing* for CE is a mapping that associates each message $m = (s, r, v_1, \ldots, v_k)$ with a directed path $route(m) = (e_1, \ldots, e_j),\ e_i \in E$, that connects the sending node s to the receiving node r in G.

We assume that a communication consists of three suboperation: send, route and receive. A schedule specifies start times and end times for each of these three suboperations. The sender processor services $m = (s, r, v_1, \ldots, v_k)$ in the interval $(begin(m, send), end(m, send))$; the edges on the route of m service m in the interval $(begin(m, route), end(m, route))$; and the receiver processor services c in the interval $(begin(m, recv), end(m, recv))$. A valid schedule fulfills the following conditions

1. $end(m, send) - begin(m, send) \geq \ell_s + \tau k$
2. $begin(m, route) - begin(m, send) \geq \ell_s$
3. $end(m, route) - begin(m, route) \geq \ell_w j + \tau k$
4. $begin(m, recv) - begin(m, route) \geq \ell_w j$
5. $end(m, recv) - begin(m, recv) \geq \ell_r + \tau k$
6. Each processor services at most m communications at any time.
7. Each edge services at most one communication at any time.

This model assumes that all wire delays ℓ_w are identical, and is conservative in that it assumes that all edges on a route are tied up while the message is routed. This is an adequate approximation when wiring delays are small relative to message length. On the other hand, the model may underestimate communication as it assumes that routing is computed offline, or that an online routing algorithm will achieve performance close that achieved by an offline algorithm. This is an adequate approximation for the simple routing patterns considered here.

As we consider only balanced communication patterns, where each node sends and receives the same number of messages, it is sufficient to use the parameter $\ell = \ell_s + \ell_r$.

Let $M = \{m_1, \ldots, m_n\}$ be a set of communications such that m_i communicates k_i data through a route of length j_i. We say that m_i *conflict* with m_j of their routes share a common edge. $M_1, \ldots, M_r$ is valid partition M if it is a partition of M such that no two communications in the same subset conflict. The following weak result is sufficient for our purposes.

Lemma 2
Let $M_1, \ldots M_r$ be a valid partition of M such that each processor sends and receives no more than m communication operations in M_i. Then the communications of M can be scheduled in time

$$\ell + \sum_{i=1}^{r} (\ell_w \max_{m_s \in M_i} j_s + d\tau \max_{m_s \in M_i} k_s.$$

Proof: The messages are routed in r successive rounds, where messages in M_i are routed at round i. The routing of messages in M_i can be completed in time $\ell_w \max_{m_s \in M_i} j_s + \tau \max_{m_s \in M_i} j_s$. The result follows.

□

We shall first consider a 3D grid topology with wrap around connections:

$$V = \{(i,j,k) \,:\, 1 \leq i,j,k \leq \sqrt{p}\}$$

and each node (i,j,k) is connected to nodes $(i \pm 1, j \pm 1, k \pm 1)$, where addition is modulo $\sqrt{p}$. Note that we assume that edges can support simultaneous communication in both directions.

Consider the broadcast in Algorithm 4 on page 22. It consists of communications on the x axis followed by communication on the z axis, and communications on the y axis, followed by communication on the z axis. The shifts in directions $(1,0,0)$ $(-1,0,0)$ $(0,1,0)$ and $(0,-1,0)$ are all independent and all use disjoint edges. Thus, the communications can be reordered to occur in R phases, where four independent shifts are performed at each phase. The communications in each phase can be completed in time $\ell + \ell_w + \tau(1+\epsilon)n/p$, if $m \geq 8$. The shifts on the x and y axes can be completed in communication time

$$\ell r + \ell_w r + \tau r(1+\epsilon)n/p.$$

The communications on the z axis can be performed in $\tilde{r}$ phases, where each phase consists of four independent shifts in directions $(0,0,\pm 1)$ and $(0,0,\pm r)$. The communications in each phase can be partitioned into at most $\tilde{r}+2$ conflict-free rounds: $\tilde{r}+1$ rounds for the shift at distance $\tilde{r}$ and one round for the shift at distance one. Thus, each phase can be completed in time $\ell + \ell_w((\tilde{r}+1)\tilde{r}+1) + \tau(\tilde{r}+2)(2r+1)(1+\epsilon)n/p$, and all shifts on the z axis can be completed in communication time

$$\ell\tilde{r} + \ell_w\tilde{r}(\tilde{r}^2+\tilde{r}+1) + \tau\tilde{r}(\tilde{r}+2)(2r+1)(1+\epsilon)n/p$$

assuming, again, that $m \geq 8$. Total communication time for the broadcast is

$$\begin{aligned}\ell(t+\tilde{r}) + \ell_w(r+\tilde{r}^3+\tilde{r}^2+\tilde{r})) + \tau(1+\epsilon)(r+\tilde{r}(\tilde{r}+2)(2r+1))n/p \\ = \ell(c/d)p^{\frac{1}{3}} + \ell_w(c/d)^{3/2}p^{\frac{2}{3}} + 2\tau(1+\epsilon)(c/d)^2 n/p^{\frac{1}{3}} + \text{ lower order terms}\end{aligned}$$

We next consider a 3D grid without wrap around connections, which is the Blue Gene topology. A well-known technique simulates a grid with wrap around connections on a grid without wrap around connections, by embedding a cycle on each line, as shown in Figure 5 on the following page

This embedding maps each edge of the grid with wraparound into a path of length ≤ 2; thus the length of each route is increased by a factor of two. One can easily see that individual shifts in the same direction that were edge disjoint in the torus are edge disjoint after the embedding. On the other hand, while positive and negative shifts in the same dimension used disjoint edges on the torus, they now use the same edges in the embedded torus. Thus, each routing round on the torus needs to be split into two subrounds in the embedded torus. The communication time becomes

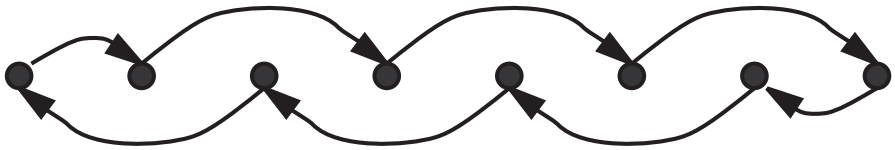

Figure 5: Efficient embedding of cycle in line

$$\ell(c/d)p^{\frac{1}{3}} + 4\ell_w(c/d)^{3/2}p^{\frac{2}{3}} + 4\tau(1+\epsilon)(c/d)^2 n/p^{\frac{1}{3}} + \text{ lower order terms}$$

The analysis of this section has not taken into account constraints on mesh routing due to the availability of a limited number of virtual channels, and the need for deadlock avoidance. However, the communication patterns used in this section are very simple, involving only shifts in one dimension. Thus, the results of the analysis should hold even when one takes into account such restrictions.

7 Conclusion

We have defined in this paper a new algorithm for a direct solution of the N-body problem with cutoff, and have shown that this algorithm is optimal, if work is kept balanced. We hope, in subsequent work, to report results form actual measurements of an implementation of the hybrid algorithm.

The lower bound proofs assumed that the location(s) where each force is computed and each atom coordinates are updated are fixed. We suspect that the bounds hold even if these locations are allowed to change, but have no formal proof.

The lower bound on communication is not valid if work is not balanced – in particular, one can perform all the computation on one processor, with no communication. It is likely that a trade-off can be shown between computation and communication. It would be nice to generalize the the lower bound results so as to optimize this trade-off, for any particular value of a and b.

The lower bounds where shown for a restricted computation model, that prohibits methods such as the multipole algorithm. The existence of such an algorithm proves that some restrictions on the computation model are needed, in order to prove the lower bounds. However, it is likely that one can generalize, e.g., allowing arbitrary linear combinations of coordinates or forces in the computation. The lower bounds on communication would then follow from dimension arguments.

8 Acknowledgments

We thank Jose Castanos and Don Coppersmith for carefully reading an early version of this paper, and Miklos Ajtai, Isaac Chavel and especially Michael Schub for offering help with the geometric inequalities of Section 5.

References

[1] M. P. Allen and D. J. Tildesley. *Computer Simulation of Liquids.* Oxford Science Publications, Oxford, UK, 1987.

[2] George S. Almasi, Călin Caşcaval, José G. Castaños, Monty Denneau, Wilm Donath, Maria Eleftheriou, Mark Giampapa, Howard Ho, Derek Lieber, José E. Moreira, Dennis Newns, Marc Snir, and Jr. Henry S. Warren. Demonstrating the scalability of a molecular dynamics application on a petaflop computer. In *International Conference on Supercomputing*, 2001.

[3] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. In *Proceedings of the 4th Annual Symposium on Parallel Algorithms and Architectures*, pages 13– 22, 1992.

[4] John Board and Klaus Schulten. The fast multipole algorithm. *IEEE Computational Science & Engineering*, 2:56–59, 2000.

[5] Yu. D. Burago and V. A. Zalgaller. *Geometric Inequalities.* Springer-Verlag, Berlin, 1988.

[6] Matthew T. Dickerson and David Eppstein. Algorithms for proximity problems in higher dimensions. *Computational Geometry Theory and Applications*, 5:277–291, 1996.

[7] P. Ewald. Die Berechnung optischer und elektrostatischer Gitterpotentiale. *Ann. Phys.*, 64:253–287, 1921.

[8] G. Fox, M Johnson, G Lyznega, S Otto, J salmon, and D Walker. *Solving Problems on Concurrent Processors.* Prentice-Hall, 1988.

[9] L. Greengard and V. Rokhlin. A fast algorithm for particle simulation. *J. Computational Physics*, 73(2):325–348, 1987.

[10] L. F. Greengard. *The Rapid Evaluation of Potential Fields in Particle Systems.* MIT Press, 1988.

[11] Rajiv Motwani and Prabhakar Raghavan. *Randomized algorithms.* Cambridge University Press, 1995.

[12] S. Plimpton. Fast parallel algorithms for short-range molecular dynamics. *J. Comp. Phys.*, 117:1–19, 1995.

[13] V. E. Taylor, R. Stevens, and K. Arnold. Parallel molecular dynamics: Implications for massively parallel machines. *Journal on Parallel and Distributed Computing*, 45(2):166–175, September 1997.

[14] L. Verlet. Computer 'experiments' on classical fluids. I. Thermodynamical properties of Lennard-Jones molecules. *Physical Reviews*, 159:98–103, 1967.