Partitioning 3D space for parallel many-particle simulations

M. A. Stijnman [1], R. H. Bisseling *

Mathematical Institute, Utrecht University,
P.O. Box 80010, 3508 TA Utrecht, The Netherlands

G. T. Barkema

Institute for Theoretical Physics, Utrecht University,
Leuvenlaan 4, 3584 CE Utrecht, The Netherlands

Abstract

In a common approach for parallel processing applied to simulations of many-particle systems with short-ranged interactions and uniform density, the cubic simulation box is partitioned into domains of equal shape and size, each of which is assigned to one processor. We compare the commonly used simple-cubic (SC) domain shape to domain shapes chosen as the Voronoi cells of BCC, FCC, and HCP sphere packings. The latter three are found to result in superior partitionings with respect to communication overhead. Scaling of the domain shape is used to extend the range of applicability of these partitionings to a large set of processor numbers. The higher efficiency with BCC and FCC partitionings is demonstrated in simulations of the sillium model for amorphous silicon.

Key words: parallel computing, particle simulations, space partitioning
PACS: 02.70.Ns, 82.20.Wt, 61.43.Dq

* Corresponding author.
Email addresses: M.A.Stijnman@tn.utwente.nl (M. A. Stijnman), Rob.Bisseling@math.uu.nl (R. H. Bisseling), G.T.Barkema@phys.uu.nl (G. T. Barkema).
URLs: http://www.math.uu.nl/people/bisseling (R. H. Bisseling), http://www.phys.uu.nl/~barkema (G. T. Barkema).
[1] Present address: Department of Applied Physics, University of Twente, P.O. Box 217, 7500 AE Enschede, The Netherlands.

1 Introduction

Realistic simulations of molecular dynamics and other dynamic many-particle systems demand increasingly larger models. Calculations on these large models can be distributed over several processors of a parallel computer to improve performance. An excellent review of the state-of-the-art of parallel atomistic simulations has recently been published by Heffelfinger [1]. According to this work, and to the best of our knowledge, spatial decomposition of a cubic simulation box with periodic boundary conditions is done almost exclusively by partitioning into cubic domains of equal size, each of which is assigned to a processor. Exceptions to this rule are earlier work by Esselink and Hilbers [2] and Chynoweth et al. [3], who use a 2D decomposition motivated by the square mesh topology of their parallel machine. In case of density fluctuations, load imbalance between the processors might occur; here, we limit ourselves to homogeneous systems with a uniform density, such as bulk materials or liquids. Other methods are necessary for heterogeneous systems such as proteins in vacuum or stellar systems. Koradi et al. [4] have developed a partitioning method for heterogeneous systems, with processor domains defined as the Voronoi cell centered at the center of mass of a particle cluster.

In homogeneous many-particle systems, the major source of inefficiency inherent to the domain decomposition approach lies in the fact that particles interact over some distance, so that in particular particles near the surface of these domains interact with particles in neighboring domains. These particles near the surface thus cause communication with neighboring processors, redundant calculations, or both. For brevity, we call this entire extra work the communication overhead. In the case that the interaction range is much smaller than the lateral size of the domains, the communication overhead will roughly scale with the surface area of the domain. Hence, the optimal domain shape for many-particle systems with a uniform density and a short-range potential is a space-filling shape with minimal surface-to-volume (S/V) ratio.

Our work exclusively concerns a cubic simulation box with periodic boundary conditions, which is the most commonly used simulation cell. Other types of simulation cells have been proposed, such as the truncated octahedron [5,6] and the rhombic dodecahedron [7], see also a discussion by Allen and Tildesley [8].

This paper is organized as follows. First, we explore different shapes for the processor domains that are derived from simple-cubic (SC), body-centered-cubic (BCC) and face-centered-cubic (FCC) lattices. We determine their properties with respect to their use in parallel processing and discuss several implementation details. In the explanation of the basic ideas, we assume that a suitable number of processors is available. After that, we extend the range

of applicability by showing how to use these domain shapes for various numbers of processors. We discuss related partitionings such as two-dimensional hexagonal and hexagonal-close-packed (HCP) partitioning. We then apply the SC, BCC, FCC domain shapes in a representative many-particle simulation: the *sillium* model of amorphous silicon, as proposed by Wooten, Winer, and Weaire [9,10]. Finally, we present our conclusions.

2 Possible shapes for the processor domain

In this section, several domain shapes are discussed regarding their properties relevant for parallel processing. All lengths and distances are measured in fractions of the system to be simulated, which thus by definition has unit length edges. The domains assigned to each processor are equal in shape and size, and consequently have a volume of $1/p$ where p is the number of processors. The interaction range, i.e. the distance over which particles exert forces, equals the cut-off radius r_c, where $r_c \ll 1$. Relevant for our purpose is the volume of the *halo*, i.e. the region outside the domain but within a distance r_c. Particles in this halo interact with those inside the domain, causing communication overhead.

2.1 SC partitioning

The most straightforward three-dimensional division of a cube into identical domains is a division into $p = k^3$ smaller cubes, with k a positive integer. The resulting cubic domains have an edge length of $1/k$. The volume of the halo with radius r_c around each domain equals

$$V_{\mathrm{halo,SC}} = \frac{6r_c}{k^2} + \frac{3\pi r_c^2}{k} + \frac{4}{3}\pi r_c^3. \tag{1}$$

The first term is equal to r_c times the surface area of the domain. The second and third terms correspond to the extra volume of the halo located near the edges and corners of the domain, respectively. In simulations with short-range interactions as discussed here, run on parallel computers of reasonable size, we usually have that $r_c k \ll 1$, so that the first term dominates the others.

In the limit of very short-range interaction our problem reduces to the well-known Kelvin problem, which is to find a partitioning of space with minimal surface area. Kelvin [11] conjectured that the optimal solution is the Voronoi cell of the BCC lattice, slightly curved to satisfy Plateau's rules [12], but Weaire and Phelan [13] produced an even better partitioning (with 0.3% less

surface area), based on two different cells, and related to the β-tungsten structure. We limit ourselves to proposing partitionings that can be shown to be better than SC and that can be implemented in a relatively simple way.

In the case of SC, the surface area equals

$$S_{\mathrm{SC}} = \frac{6}{k^2} = \frac{6}{p^{\frac{2}{3}}}, \tag{2}$$

and because the volume of the cube assigned to a processor equals $V = 1/p$, the resulting S/V ratio is

$$(S/V)_{\mathrm{SC}} = 6 \cdot p^{\frac{1}{3}}. \tag{3}$$

2.2 BCC partitioning

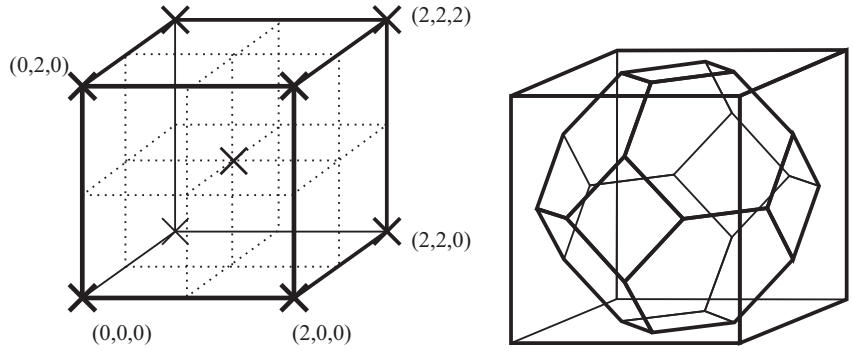
A sphere with volume $V = 1/p$ and radius R has a ratio

$$(S/V)_{\mathrm{sphere}} = \frac{4\pi R^2}{4\pi R^3/3} = (36\pi p)^{\frac{1}{3}} \approx 4.836 \cdot p^{\frac{1}{3}}, \tag{4}$$

which is the minimum ratio that can be achieved; it is about 19.6% better than for SC. Although we cannot fill space with spheres, we can try to use sphere packings to obtain a domain shape with a low S/V ratio. In one well-known sphere packing, the spheres are centered at the sites of a body-centered-cubic (BCC) lattice, with spheres at the corners and the centers of cubic cells. The BCC unit cell is displayed in Figure 1(a). Each unit cell adds two sphere centers to the lattice, as only one corner point is part of the unit cell; the other seven corner points are considered part of neighboring cells. By repeating this unit cell the BCC lattice is generated. The lattice is then rescaled, such that the length of the edges of a unit cell becomes $1/k$.

The domain of a processor is formed by the Voronoi cell of a lattice point, i.e., the space closest to that point, measured by Euclidean distance. The model cube can now be divided into $p = 2k^3$ Voronoi cells, as generated by the BCC lattice. It turns out that each Voronoi cell is a truncated octahedron, as shown in Figure 1(b). All cells are translated copies of each other. This is also the shape that Kelvin proposed as a solution to the Kelvin problem.

The truncated octahedron generated by the BCC lattice fits into a cube with edge length $1/k$. Each of its six square faces has a surface area of $1/(8k^2)$ and each of its eight hexagonal faces has a surface area of $3\sqrt{3}/(16k^2)$. This results



(a) (b)

Fig. 1. (a) The basic BCC lattice cell. Sphere centers are marked by '×'. (b) The BCC Voronoi cell, a truncated octahedron.

in a total surface area of

$$S_{\mathrm{BCC}} = \frac{6\sqrt{3}+3}{4k^2}. \tag{5}$$

Substitution of $k = (p/2)^{\frac{1}{3}}$ and $V = 1/p$ gives

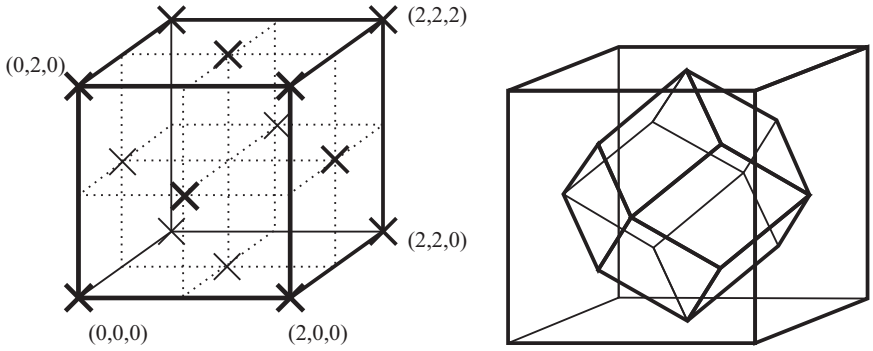
$$(S/V)_{\mathrm{BCC}} = \frac{(6\sqrt{3}+3)p^{\frac{1}{3}}}{2^{\frac{4}{3}}} \approx 5.315 \cdot p^{\frac{1}{3}}. \tag{6}$$

This is about 11.4% better than for SC.

2.3 FCC partitioning

In one of the optimal dense sphere packings, the spheres are placed at sites of a face-centered-cubic (FCC) lattice, with spheres at the corners of cubic cells and at the centers of the faces. The FCC unit cell is shown in Figure 2(a). The corresponding Voronoi cell is a rhombic dodecahedron, as shown in Figure 2(b). Each cubic unit cell adds four points to the lattice, so with this partitioning $p = 4k^3$ processors can be used. After the lattice is rescaled such that each unit cell has an edge of length $1/k$, the Voronoi cell can be considered as being made up of a cube with edge length $1/(2k)$ and six pyramids with height $1/(4k)$, each covering one face of the small cube. The surface area of the rhombic dodecahedron equals 24 times the surface area of one of the triangular faces of the pyramid, which is $1/(8\sqrt{2}k^2)$. The surface area of the domain in the FCC partitioning therefore equals

$$S_{\mathrm{FCC}} = \frac{3}{\sqrt{2}k^2}. \tag{7}$$



(a) (b)

Fig. 2. (a) The basic FCC lattice cell. Sphere centers are marked by '×'. (b) The FCC Voronoi cell, a rhombic dodecahedron, translated to the center of the cube for reference.

Substitution of $k = (p/4)^{\frac{1}{3}}$ and $V = 1/p$ gives

$$(S/V)_{\mathrm{FCC}} = 3 \cdot 2^{\frac{5}{6}} p^{\frac{1}{3}} \approx 5.345 \cdot p^{\frac{1}{3}}, \tag{8}$$

which is slightly more than for BCC, but still about 10.9 % better than for SC.

3 Practical implementation issues

For a partitioning method to be of practical use, it must be easy and inexpensive to determine the processor that owns a given particle and the processors that have the particle in their halo. The latter are the processors that need to obtain particle data through communication. For modularity, it is best to express the determination of these processor numbers as generic functions, which are called by the application program. No partitioning-related computations should be done directly by the application program; instead, the generic functions should be called. This way, it becomes easy to replace the partitioning method. Furthermore, we need a generic function that determines the partitioning method to be used for a given value of p. This function is called once, before the start of the simulation. How to choose the partitioning method for an arbitrary number of processors is the subject of Section 4. Our generic functions will be made available for general use. [2]

[2] See `http://www.math.uu.nl/people/bisseling/software.html`

3.1 Assigning particles to processors

In implementations, one needs an efficient procedure to determine the processor that owns a particle with coordinates (x, y, z). This processor corresponds to the nearest lattice site. Note that we can break ties arbitrarily between processors, in case particles are located exactly (or within the accuracy determined by the machine precision) on the boundary of two domains since this event is unlikely to occur often. With SC partitioning for $p = k^3$ processors, the processor number s can then be found by:

$$s = \lfloor kx \rfloor + k\lfloor ky \rfloor + k^2 \lfloor kz \rfloor, \tag{9}$$

where $\lfloor x \rfloor$ is the largest integer smaller than or equal to x. (Similarly, we define $\lceil x \rceil$ as the smallest integer larger than or equal to x.)

To assign processor numbers to domains with the BCC shape, it is helpful to note that a BCC lattice consists of two simple-cubic lattices, shifted with respect to each other over a vector $(\frac{1}{2k}, \frac{1}{2k}, \frac{1}{2k})$. One simply determines the nearest site in each of the two sublattices, compares the two, and takes the nearest. Here, one can take Manhattan distances, defined by $\|(x, y, z)\|_1 = |x| + |y| + |z|$, because these are cheaper to compute than Euclidean distances. Since the sum of the Manhattan distances of an arbitrary point to the two nearest sites equals $3/(2k)$, the nearest one is located at a Manhattan distance of less than $3/(4k)$. The procedure to calculate the processor number for the BCC partitioning is outlined in the pseudo-code below, where $[x] = \lfloor x + 1/2 \rfloor$ denotes the integer nearest to x and ' mod ' denotes the modulo operator (needed to wrap around the periodic boundaries):

$D = |kx - [kx]| + |ky - [ky]| + |kz - [kz]|$
if $D < \frac{3}{4}$ **then**
  $s = ([kx] \bmod k) + k([ky] \bmod k) + k^2([kz] \bmod k)$
else
  $s = k^3 + \lfloor kx \rfloor + k\lfloor ky \rfloor + k^2 \lfloor kz \rfloor$
end if

For the FCC lattice, it is helpful to note that it can be obtained from a cubic superlattice by removing all grid points for which the total of the coordinates is odd. First, we determine the nearest grid point of this cubic superlattice. If the sum of the coordinates of that grid point is even, that point is the nearest FCC lattice site. If the sum is odd, the closest lattice point can be found by rounding one of the coordinates in the 'wrong' direction, i.e. in the opposite direction of the nearest integer coordinate; the coordinate that should be rounded 'wrongly' is the one with the largest rounding error (and hence the smallest error in the opposite direction). Using the notation $]x[$

for rounding 'wrongly' (in contrast to $[x]$ for usual rounding), defined by the relation $[x]+]x[= \lfloor x \rfloor + \lceil x \rceil$, this procedures thus becomes:

$(P_x, P_y, P_z) = ([2kx], [2ky], [2kz])$
if $P_x + P_y + P_z$ is odd **then**
  if $|2kx - P_x| > |2ky - P_y|$ **and** $|2kx - P_x| > |2kz - P_z|$ **then**
    $P_x =]2kx[$
  else if $|2ky - P_y| > |2kz - P_z|$ **then**
    $P_y =]2ky[$
  else
    $P_z =]2kz[$
  end if
end if
$(P_x, P_y, P_z) = (P_x \bmod 2k, P_y \bmod 2k, P_z \bmod 2k)$
$s = P_x + 2kP_y + 4k^2\lfloor P_z/2 \rfloor$

3.2 Assigning particles to processor halos

Our goal is to find out whether a particle $\vec{x} = (x, y, z)$ assigned to a particular processor s belongs to the halos of other processors. We present a method for producing a list that contains all the halo processors of a particle (and perhaps some other processors). We assume that r_c is sufficiently small to guarantee that the halo of a processor domain is contained in one shell of neighboring processor domains.

Particles assigned to a processor s can belong to halos of processors s_j with whom the processor shares a face. Each face j lies in a plane characterized by $\vec{x} \cdot \hat{n}_j - m_j = 0$, where $\hat{n}_j$ is a unit vector perpendicular to the plane. The distance of a particle $\vec{x}$ to this plane is easily obtained by $d_j = |\vec{x} \cdot \hat{n}_j - m_j|$. If $d_j \leq r_c$, we include processor s_j in the list. Otherwise, the particle is too far from the domain across the plane, so that it cannot be in the halo.

Particles assigned to a processor s can also belong to halos of processors s_j with whom the processor only shares an edge. Here, the relevant distance is the distance to the line characterized by $\vec{x} = \vec{a}_j + \lambda\hat{b}_j$ that contains the edge. Here, $\hat{b}_j$ is a unit vector. The distance d_j from a particle $\vec{x}$ to this line is given by $d_j^2 = ||\vec{x} - \vec{a}_j||^2 - ((\vec{x} - \vec{a}_j) \cdot \hat{b}_j)^2$, where the norm is the Euclidean norm. If $d_j \leq r_c$, we include processor s_j in the list.

Particles assigned to a processor s can also belong to halos of processors s_j with whom the processor only shares a vertex $\vec{v}_j$. If $d_j \leq r_c$, where $d_j = ||\vec{x} - \vec{v}_j||$, we include processor s_j in the list.

In general, a table of the applicable perpendicular vectors $\hat{n}_j$, offsets m_j, directional vectors $\hat{b}_j$, line points $\vec{a}_j$, and vertices $\vec{v}_j$ can be precomputed at the beginning of the simulation, after which the relevant distances can be computed easily during the simulation whenever required.

For SC partitioning, the perpendicular vectors as well as the directional vectors are unit vectors along coordinate axes; this allows for even more simplified tests. However, all three cases occur: processor s shares a face with the six processors $s \pm 1$, $s \pm k$, and $s \pm k^2$; it shares an edge with the twelve processors $s \pm 1 \pm k$, $s \pm 1 \pm k^2$, and $s \pm k \pm k^2$; and it shares a vertex with the eight processors $s \pm 1 \pm k \pm k^2$. (We assume for brevity that wrap around does not occur in the processor numbering.) For SC partitioning, the output list contains exactly all the halo processors.

For BCC partitioning, two processors never share only an edge or a vertex. Processor s shares a square face with the six processors $s \pm 1$, $s \pm k$, and $s \pm k^2$. It shares a hexagonal face with the eight processors $s + k^3 - \alpha$, where $\alpha = \alpha_1 + \alpha_2 k + \alpha_3 k^2$ with $\alpha_i \in \{0, 1\}$, in the case $s < k^3$, and with the eight processors $s - k^3 + \alpha$ otherwise. For BCC partitioning, the output list may contain additional processors, since here the distance to a face can be larger than the distance to the plane in which the face lies. For $r_c \ll 1/k$, the number of particles with additional processors in their list becomes negligible.

For FCC partitioning, processor s shares a rhombic face with twelve processors $s \pm 1$, $s \pm 2k$, $s \pm 1 \pm 2k$, $s \pm 1 - 4k^2$, and $s \pm 2k - 4k^2$, in the case of even $s + \lfloor s/(2k) \rfloor$ (equivalent to even $P_x + P_y$ in the FCC numbering); in the odd case, the term $-4k^2$ should be replaced by $+4k^2$. The processor also shares a vertex with the six processors $s \pm 2$, $s \pm 4k$, and $s \pm 4k^2$. For FCC partitioning, the output list may contain additional processors, namely some of the neighbors that share a face. This is because the distance to a face can be larger than the distance to the plane in which the face lies. Neighbours that only share a vertex cannot be among the additional processors. On the other hand, we must then be sure that our method includes such a neighbor in the list whenever the particle is close enough to that neighbor. Fortunately, this is guaranteed, because the angles between the faces intersecting in the shared vertex are such that the shared vertex is the nearest point of the neighboring processor to the particle.

4 Using an arbitrary number of processors

In the previous sections, we have explained the different space partitioning methods assuming that we have a suitable number of processors available. For example, for SC partitioning we assume that we have $p = k^3$ processors available for the whole cubic simulation box so that each domain assigned to

a processor is itself a cube. For the BCC and FCC partitionings we assume that we have $p = 2k^3, 4k^3$ processors available, respectively. This assumption, however, is not essential for using the partitioning method. For example, if we have $p = k_1k_2k_3$ processors available, with each $k_i \geq 1$ an integer, we can generalize the SC partitioning by splitting the cubic simulation cell into p rectangular blocks of size $\frac{1}{k_1} \times \frac{1}{k_2} \times \frac{1}{k_3}$. The resulting S/V ratio is

$$(S/V)_{\mathrm{SC}} = 2 \sum_{i:\, k_i > 1} k_i, \tag{10}$$

where the index i only runs over values with $k_i > 1$. Terms $k_i = 1$ do not contribute, for the following reason. If $k_1 = 1$ and $k_2, k_3 > 1$, then the first dimension is not cut, and hence the two faces of size $\frac{1}{k_2} \times \frac{1}{k_3}$ of the block do not represent interprocessor boundaries so that the area $2k_1/p$ of these faces does not contribute to S. Similarly, if $k_1 = k_2 = 1$ and $k_3 > 1$, we have a partitioning into *slices* and only two faces contribute to S. In that case $S = 2 = 2k_3/p$. If p is a cube, the S/V ratio is minimal for $k_1 = k_2 = k_3 = p^{\frac{1}{3}}$.

We can apply BCC partitioning if we have $p = 2k_1k_2k_3$ processors available. This can be done by first rescaling the cubic simulation box by a factor of k_i in each dimension i, giving a rectangular simulation box of size $k_1 \times k_2 \times k_3$, which contains $p/2$ cubes of size $1 \times 1 \times 1$. We can now partition each cube, as explained in Subsection 2.2, assigning points in space to the nearest corners and centers of cubes. This defines a partitioning of the rescaled simulation box for p processors. The partitioning of the original simulation box is then obtained by scaling back. The resulting processor domains are 'deformed' truncated octahedra, since they have been scaled by a factor $1/k_i$ in each dimension i. The domains are still translated copies of each other, because BCC is a lattice. The resulting S/V ratio is

$$(S/V)_{\mathrm{BCC}} = \frac{1}{2} \sum_{i:\, k_i > 1} k_i + 3 \left(\sum_i k_i^2 \right)^{\frac{1}{2}}. \tag{11}$$

The same procedure can be applied in the FCC case, where $p = 4k_1k_2k_3$, with a resulting ratio

$$(S/V)_{\mathrm{FCC}} = 2 \sum_{i<j} (k_i^2 + k_j^2)^{\frac{1}{2}}. \tag{12}$$

We now have extended the applicability of the partitioning methods: the SC method can be used for every number of processors p, the BCC method for every even p, and the FCC method for every multiple of four. For a particular partitioning method, the S/V ratio is minimal if $k_1 = k_2 = k_3 = (p/c)^{\frac{1}{3}}$, where $c = 1, 2, 4$ is the constant corresponding to the method (SC, BCC,

FCC, respectively). In that case, the S/V ratios $6, 5.315, 5.345$ times $p^{\frac{1}{3}}$ are attained. If equality cannot be achieved, the factors k_i should be chosen as close as possible to $(p/c)^{\frac{1}{3}}$. For SC, BCC, and FCC, we may assume without loss of generality that $k_1 \leq k_2 \leq k_3$.

As an example, we take the case $p = 8$. We can apply the SC method with the triples $(k_1, k_2, k_3) = (1, 1, 8), (1, 2, 4), (2, 2, 2)$, giving $S/V = 16, 12, 12$, respectively. We can apply the BCC method (with constant $c = 2$) with the triples $(k_1, k_2, k_3) = (1, 1, 4), (1, 2, 2)$, giving $S/V = 14.728, 11$, respectively. We can also apply the FCC method (with constant $c = 4$) with the triple $(k_1, k_2, k_3) = (1, 1, 2)$, giving $S/V = 11.773$. The surprising conclusion for the case $p = 8$, which at first sight seems to be tailored to the SC method, is that both the BCC and FCC methods are better than SC. The gain of BCC with $(k_1, k_2, k_3) = (1, 2, 2)$ compared to the best SC method is about 8.3%.

Table 1 presents a summary of the best possible partitionings for each of the three methods SC, BCC, FCC in the range $p = 1$–32. The table gives a triple for which the lowest S/V ratio is achieved, and the corresponding ratio. For convenience, we express the S/V ratio in terms of $p^{\frac{1}{3}}$. This allows us to compare ratios for different values of p, and it also makes it easy to see whether we are close to the best BCC value of $5.315 \cdot p^{\frac{1}{3}}$. The table illustrates the efficiency benefits of having several different partitioning methods in our toolbox. The SC method is superior when the number p does not have many prime factors. It is the only applicable method if p is odd. The BCC method, on the other hand, is better than SC when p is highly composite and has prime factors such that the simulation box can be cut into $p/2$ almost cubic blocks. This happens for $p = 8, 12, 16, 24, 32$. The BCC method is superior to FCC in the range studied, except for $p = 32$. This is mainly due to the fact that the BCC method requires only one factor of 2, instead of two such factors as FCC does, and to a lesser extent it is due to the 0.6% lower S/V ratio in the ideal case. The largest gain in the range $p = 1$–32 is found for $p = 16$, where the savings are 16.3%. The FCC method is best for $p = 32$, where it has an ideal split in all three dimensions.

The practical implementation for an arbitrary number of processors is slightly more complicated than in the special case $k_1 = k_2 = k_3$ treated in Section 3. The methods presented for assigning particles to domains and halos should be generalized to take the rescaling into account. We omit the details for the sake of brevity.

p	best triple (k_1, k_2, k_3)			S/V ratio (in $p^{\frac{1}{3}}$)		
	SC	BCC	FCC	SC	BCC	FCC
1	(1,1,1)			**0.000**		
2	(1,1,2)	(1,1,1)		**3.175**	4.124	
3	(1,1,3)			**4.160**		
4	(1,2,2)	(1,1,2)	(1,1,1)	**5.040**	5.259	5.345
5	(1,1,5)			**5.848**		
6	(1,2,3)	(1,1,3)		**5.503**	6.301	
7	(1,1,7)			**7.319**		
8	(2,2,2)	(1,2,2)	(1,1,2)	6.000	**5.500**	5.886
9	(1,3,3)			**5.769**		
10	(1,2,5)	(1,1,5)		**6.498**	8.396	
11	(1,1,11)			**9.892**		
12	(2,2,3)	(1,2,3)	(1,1,3)	6.115	**5.995**	6.760
13	(1,1,13)			**11.058**		
14	(1,2,7)	(1,1,7)		**7.468**	10.341	
15	(1,3,5)			**6.488**		
16	(2,2,4)	(2,2,2)	(1,2,2)	6.350	**5.315**	5.794
17	(1,1,17)			**13.223**		
18	(2,3,3)	(1,3,3)		**6.105**	6.134	
19	(1,1,19)			**14.241**		
20	(2,2,5)	(1,2,5)	(1,1,5)	**6.631**	7.343	8.556
21	(1,3,7)			**7.249**		
22	(1,2,11)	(1,1,11)		**9.279**	13.837	
23	(1,1,23)			**16.175**		
24	(2,3,4)	(2,2,3)	(1,2,3)	6.240	**5.502**	6.243
25	(1,5,5)			**6.840**		
26	(1,2,13)	(1,1,13)		**10.127**	15.436	
27	(3,3,3)			**6.000**		
28	(2,2,7)	(1,2,7)	(1,1,7)	**7.245**	8.742	10.246
29	(1,1,29)			**18.878**		
30	(2,3,5)	(1,3,5)		**6.437**	6.999	
31	(1,1,31)			**19.737**		
32	(2,4,4)	(2,2,4)	(2,1,2)	6.300	5.889	**5.345**

Table 1
Best triple and resulting S/V ratio for three 3D partitioning methods for p =1–32. The lowest ratio is given in boldface.

5 Related partitioning methods

We have presented a basic partitioning toolbox containing three partitioning methods that can handle most processor numbers in an efficient way. Still, we cannot claim optimality and in fact sometimes it would be better to use a different method. We will briefly discuss a few related methods in this section.

We can modify the FCC method by taking only the face centers of the FCC unit cell as centers of Voronoi cells, giving three domains per unit cell. The resulting domain shape for $p = 3k^3$ processors is an octahedron consisting of two pyramids with height $1/(2k)$ and base edges of length $1/k$. We call the resulting method the *octahedral* method. Now, the centers of the Voronoi cells do not form a lattice and the domains are not translated copies of each other. Nonuniform scaling yields processor domains that differ from each other because they have a different surface area; their volume is the same. The domain with the largest surface area determines the S/V ratio of the method. This makes the octahedral method much less attractive than other methods in the case of nonuniform scaling.

The S/V ratio of the octahedral method for $p = 3k_1k_2k_3$ processors with $k_1, k_2 \leq k_3$ is

$$(S/V)_{\text{Oct}} = 3\left((k_1^2 + k_3^2)^{\frac{1}{2}} + (k_2^2 + k_3^2)^{\frac{1}{2}}\right). \tag{13}$$

Note that this formula is not symmetric in k_1, k_2, k_3, so that it only holds if k_3 is the largest parameter. In the ideal case $k_1 = k_2 = k_3 = k$, the ratio equals

$$(S/V)_{\text{Oct}} = 6\sqrt{2}k = 2^{\frac{3}{2}} \cdot 3^{\frac{2}{3}} p^{\frac{1}{3}} \approx 5.883 \cdot p^{\frac{1}{3}}. \tag{14}$$

This is slightly better than the ideal SC ratio, but much worse than BCC and FCC. We found that for $p = 81$, the octahedral method is best, since BCC and FCC cannot be applied, and SC with the triple (3,3,9) has a higher ratio of $6.934 \cdot p^{\frac{1}{3}}$. In the same way, the octahedral method is best for all $p = 3k^3$ with k odd. In the range $p \leq 1024$, this method is the best for $p = 81, 375, 525, 735$. In practice, the octahedral method will be useful only on rare occasions.

For small numbers of processors, it may be useful to partition only two dimensions. For the SC method, this leads to the choice $k_3 = 1$. In two dimensions, hexagonal lattice packing is an optimal sphere packing and its Voronoi cells, regular hexagons, have a minimal S/V ratio. We can use this lattice to obtain an alternative method for partitioning a square simulation box. For $p = 2k^2$, we can partition the box into k^2 squares with edges of length $1/k$. Each square can then be divided into a central (nonregular) hexagon and four quarter hexagons as depicted in Figure 3, and these domains can be assigned

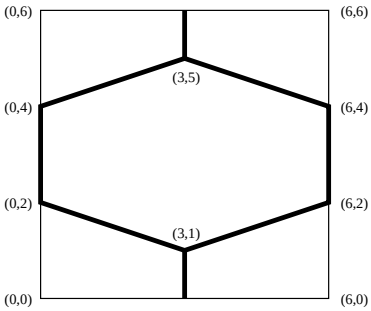


Fig. 3. The basic two-dimensional hexagonal unit cell with edge length 6. The boundaries between processor domains are shown in boldface.

to processors. For $p = 2k_1k_2$, we can rescale in the same way as before, and obtain a ratio

$$(S/V)_{\text{2D Hex}} = \frac{4}{3}\left((k_1^2 + 9k_2^2)^{\frac{1}{2}} + k_1 - \delta_{1,k_1}\right). \tag{15}$$

(The term k_1 drops for $k_1 = 1$.) For $k_1 \approx \sqrt{3}k_2$, we approach regular hexagons and obtain a ratio of

$$(S/V)_{\text{2D Hex}} \approx 4\sqrt{3}k_2 = 2^{\frac{3}{2}}3^{\frac{1}{4}}p^{\frac{1}{2}} \approx 3.722 \cdot p^{\frac{1}{2}}. \tag{16}$$

This is better than the ideal two-dimensional SC ratio of $4p^{\frac{1}{2}}$. Of course, we can only approach the exact ratio $k_1/k_2 = \sqrt{3}$, but never achieve it, since k_1 and k_2 must be integers. We can use the hexagonal method to partition a cubic simulation box by partitioning space based on the first two coordinates; equation (15) also gives the S/V ratio for the 3D case. This method is best for $p = 4, 6, 12, 20, 28$, giving S/V ratios (in $p^{\frac{1}{3}}$) of 4.708, 5.314, 5.654, 6.292, and 7.122 respectively, cf. Table 1. For $p > 28$, it can be shown that three-dimensional SC partitioning is better than two-dimensional hexagonal partitioning.

FCC is only one member of an infinite family of related optimal sphere packings, see [14]; another one is hexagonal-close-packed (HCP). We can construct an HCP packing by starting with a layer of spheres centered at the points $\lambda(1,0,0) + \mu(\frac{1}{2}, \frac{1}{2}\sqrt{3}, 0)$, with λ, μ integers, thus forming a two-dimensional hexagonal lattice in the $z = 0$ plane. Then we add a new hexagonal layer on top of the first one, identical but translated by $(\frac{1}{2}, \frac{1}{6}\sqrt{3}, \frac{1}{3}\sqrt{6})$, and a third layer in the same position as the first one, but translated by $(0, 0, \frac{2}{3}\sqrt{6})$. We continue adding layers in this way, also in the negative z-direction.

The Voronoi cells of HCP, twisted rhombic dodecahedrons, have the same volume and surface area as those of FCC. HCP packing has the disadvantage that its sphere centers do not form a lattice, which means that Voronoi cells are not necessarily translated copies of each other. Still, all the Voronoi cells

r	best triple $(\frac{k_1}{2^q}, \frac{k_2}{2^q}, \frac{k_3}{2^q})$				S/V ratio (in $p^{\frac{1}{3}}$)			
	SC	BCC	FCC	HCP	SC	BCC	FCC	HCP
0	$(1,1,1)$	$(\frac{1}{2},1,1)$	$(\frac{1}{2},\frac{1}{2},1)$	$(1,\frac{1}{2},\frac{1}{2})$	6.000	5.750	5.886	**5.376**
1	$(1,1,2)$	$(1,1,1)$	$(\frac{1}{2},1,1)$	$(1,\frac{1}{2},1)$	6.350	**5.315**	5.794	5.650
2	$(1,2,2)$	$(1,1,2)$	$(1,1,1)$	$(1,1,1)$	6.300	5.889	**5.345**	5.574

Table 2
Best triple and resulting S/V ratio for four 3D partitioning methods for $p = 2^{3q+r}$. The table is valid for FCC with $p \geq 4$, SC and HCP with $p \geq 8$, and BCC with $p \geq 16$. The lowest ratio is given in boldface.

in the even layers are translated copies of each other, and the same holds for the cells in the odd layers. The two cell types are related, for instance by a reflection: one cell can be obtained from the other by reflection in the $y = 0$ plane.

We can derive an HCP partitioning method for $p = 4k_1k_2k_3$ processors, by rescaling similar to the two-dimensional hexagonal case. The S/V ratio of the resulting processor domains is

$$(S/V)_{\text{HCP}} = (k_1^2 + 9k_2^2)^{\frac{1}{2}} + k_1 - \delta_{1,k_1} + (k_1^2 + k_2^2 + \frac{64}{9}k_3^2)^{\frac{1}{2}} + (k_2^2 + \frac{16}{9}k_3^2)^{\frac{1}{2}}. \quad (17)$$

Because we scale along the coordinate axes, all the domains have an identical S/V ratio. (This is in contrast to the previous nonlattice case, octahedral partitioning.) For $k_1 \approx \sqrt{3}k_2$ and $k_1 \approx \frac{2}{3}\sqrt{6}k_3$, we approach the ideal HCP ratio of about $5.345 \cdot p^{\frac{1}{3}}$, but we cannot achieve it exactly. Implementing HCP is more difficult than implementing FCC, because HCP is not derived from a cubic grid and because of the alternating layers with two different domains.

The number of processors of a parallel computer is often a power of two. We write $p = 2^{3q+r}$, with $q \geq 0$ and $r = 0, 1, 2$. We can take k_1, k_2, k_3 equal or as near as possible to equal, and then substitute these values into equations (10)–(12), for the partitioning methods SC, BCC, and FCC. For HCP, we can find the best triple using equation (17). Table 2 summarizes the results for four partitioning methods for the important case where p is a power of two. Note that the table is not valid for small values of p, since then terms are dropped from the sums in the S/V formulae. Because of symmetry, the given triple (k_1, k_2, k_3) can be permuted for SC, BCC, and FCC; this cannot be done for HCP. Note that the three methods BCC, FCC, and HCP nicely supplement each other, and that they all outperform SC.

6 Application: amorphous silicon

We have applied SC, BCC, and FCC partitioning to the construction of models of amorphous silicon, following the *sillium* model proposed by Wooten, Winer, and Weaire [9,10], with recent algorithmic improvements [15]. This has produced the best models of amorphous silicon that are available to date.

Within the sillium approach, an atomic configuration consists of the coordinates of all N atoms, together with a list of the $2N$ bonds between them. The energy of a configuration is obtained from the Keating potential [16]:

$$E = \frac{3}{16}\frac{\alpha}{d^2}\sum_{\langle ij\rangle}\left(\mathbf{r}_{ij}\cdot\mathbf{r}_{ij} - d^2\right)^2 + \frac{3}{8}\frac{\beta}{d^2}\sum_{\langle jik\rangle}\left(\mathbf{r}_{ij}\cdot\mathbf{r}_{ik} + \frac{1}{3}d^2\right)^2 . \tag{18}$$

Here, α and β are the bond-stretching and bond-bending force constants, respectively; $d = 2.35$ Å is the equilibrium Si-Si bond length in the diamond structure. Usual values are $\alpha = 2.965$ eV/Å^2 and $\beta = 0.285\ \alpha$.

The construction of a well-relaxed configuration starts from a configuration in which atoms with random coordinates are four-fold connected. This network is then relaxed through a sequence of many proposed bond transpositions, accepted with the Metropolis acceptance probability [17] given by

$$P = \min\left[1, \exp\left(\frac{E_b - E_f}{k_b T}\right)\right], \tag{19}$$

where k_b is the Boltzmann constant, T the temperature, and E_b and E_f are the total *quenched* energies of the system before and after the proposed bond transposition.

With the approach given above, and described in more detail in Refs. [9,10], Wooten, Winer, and Weaire obtained 216-atom models of *a*-Si with a bond angle deviation as low as 10.9 degrees. A decade later, using the same approach but more computing power, Djordjević, Thorpe, and Wooten [18] produced larger (4096-atom) models of even better quality, with a bond angle deviation of 11.02 degrees for configurations without four-membered rings and 10.51 degrees when these rings were allowed [18]. With some additional algorithmic improvements, Barkema and Mousseau generated 1000-atom models with a bond angle deviation of 9.2 degrees [15], and one 4096-atom model with a bond angle deviation of 9.89 degrees. Exploiting parallel processing, we have generated a 10,000-atom model with a bond-angle deviation as low as 9.88 degrees and a 20,000-atom model, used primarily for our benchmarking. The structural and electronic properties of these models are discussed in [19]; here

we focus on computational aspects, using the 20,000-atom model in all our experiments.

In our parallel program, the simulation box containing 20,000 atoms was divided by three partitioning methods: SC, BCC, and FCC. For each method, we used the best triple (k_1, k_2, k_3) given by Table 1. In addition, we used SC with $k_1 = k_2 = 1$ and $k_3 = p$ as a separate 'slices' method, to show the effects of one-dimensional partitioning. Due to the three-body term in the Keating potential, each processor needs two extra layers of atoms near the surface of its domain. Therefore, in our application, the halo of a processor consists of all the atoms outside the processor that are at most two bonds away from an atom in the processor. As a consequence, the communication of the halo atoms is determined by the connectivity information in the model. It turns out that the halo size can be well approximated by a cut-off distance of 1.3 times the average bond length of 2.35Å. For the 20,000-atom model, with a simulation box size of about 70.4Å, this gives a value of $r_c \approx 0.043$ (in units where the box size is one.) Note that this is a rough approximation, and that the true interaction cut-off is determined by the connectivity and not by the distance between two atoms.

The parallel program, which is based on the BSPlib communications library [20], was tested for different values of p on a Cray T3E parallel computer. As a simple performance metric, we take the efficiency E_p, defined as

$$E_p = \frac{T_1}{pT_p}, \tag{20}$$

where T_p is the execution time of one iteration of the global relaxation procedure on p processors and T_1 is the time for one processor without communication overhead.

The results of the efficiency measurements are shown in Figure 4. It is clear that in general the efficiency decreases as p increases, although with fluctuations. It can be seen for instance that the better S/V ratio of FCC as given in Table 1 for $p = 32$ indeed leads to higher efficiency. In fact, for all values of p studied, the measured relative performance of the three methods SC, BCC, and FCC agrees with the theoretical predictions of the table; our data show that this holds even in the case where efficiencies nearly coincide in the plot. Note in particular the high efficiency that can be obtained for powers of two, such as $p = 16$ and $p = 32$. It is clear that using slices should be avoided, because in that case efficiency decreases rapidly. For $p = 4$, slices and SC with the triple $(1, 2, 2)$ given in Table 1 are different partitionings, but their S/V ratio is equal. Here, slices perform better, because the halo volume for slices does not have lower order terms, such as those given in equation (1) for a cube.

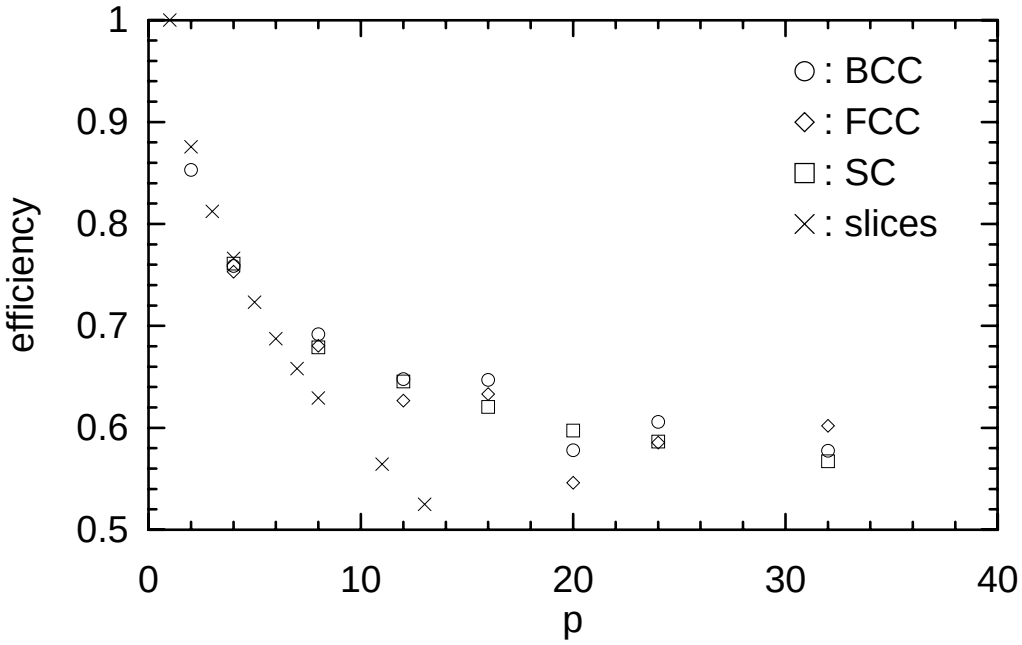


Fig. 4. Measured efficiency as a function of the number of processors p.

	SC	BCC	FCC
halo communication	0.502	0.484	0.407
halo total	2.841	2.714	2.423
iteration total ($= T_p$)	8.580	8.427	8.086

Table 3
Time (in ms) of parts of one relaxation iteration for three 3D partitioning methods for $p = 32$.

The differences in efficiency we measured are less pronounced than Table 1 might suggest. This is because the efficiency metric takes the whole computation into account, whereas the improvement of the FCC method over the other methods is only in the halo-related part. Table 3 gives additional timing information on parts of a relaxation iteration for $p = 32$. The pure communication part is about 19% faster for FCC than for SC. The total halo part, which includes communication of halo atoms but also all computations and data structure manipulations related to the halo, shows a considerable time savings of about 15% for FCC, which is close to the value predicted by the S/V ratio. Note that for brevity we called the halo-related operations 'communication overhead', but that in fact this involves much more than the communication itself.

To investigate the communication properties further, we counted the atoms in the inner regions and the halos. Both the maximum and the average over the p processors are listed in Table 4; the maximum number of interior atoms determines the computation time, whereas the maximum number of halo atoms

determines the communication time. Also displayed is the ratio of the average number of halo atoms to the average number of interior atoms. (The ratio of the averages is the best measure of the effects studied, since it is less noisy than the ratio of the maxima.) The last column of the table gives the experimental halo/interior ratio divided by the theoretical S/V ratio. This value should be more or less constant, and it is a test of our assumption that the S/V ratio is a good measure of relative halo size. The value for SC with $p = 2$ should be equal to the cut-off radius r_c, since the halo has volume $2r_c$ in that case. The value 0.0458 agrees reasonably well with the previously determined value $r_c \approx 0.043$. For larger numbers of processors the value increases slowly, mainly due to terms in r_c^2/k, cf. equation (1) for the SC case with $p = k^3$. Similar effects occur for BCC and FCC. The table shows for instance that for $p = 32$ FCC indeed has fewer particles in its halo than SC and BCC, explaining the results of Figure 4.

7 Conclusion and future work

We have proposed three new space partitionings, based on Voronoi cells of the BCC, FCC, and HCP sphere packings, for use in parallel particle simulations with uniform density, short-range interactions, and a cubic simulation box. The advantage of these new partitionings is that they yield domains with a lower S/V ratio than the commonly used SC partitioning and thus reduce the communication overhead of a parallel computation. The ratio, expressed in terms of $p^{\frac{1}{3}}$, is about 11% lower in the ideal BCC, FCC, and HCP case compared to the ideal SC case. The gain for a particular number of processors p, however, may even be larger.

The new partitioning methods are particularly advantageous for numbers of processors that are a power of two, a case that commonly occurs. For $p = 16, 128, 1024, \ldots$, BCC partitioning achieves the ratio of $5.315 \cdot p^{\frac{1}{3}}$, which is the lowest ratio known to date for a single allowed domain shape. Over a century ago, Kelvin conjectured that this is optimal; for a single allowed domain shape the conjecture remains an open problem. For $p = 32, 256, 2096, \ldots$, FCC partitioning is close to optimal, achieving a ratio of $5.345 \cdot p^{\frac{1}{3}}$. For $p = 8, 64, 512, \ldots$, HCP partitioning achieves a ratio of $5.376 \cdot p^{\frac{1}{3}}$. The savings thus obtained by BCC, FCC, and HCP, are 16.3, 15.2, 10.4%, respectively, compared to SC. This means that the communication overhead can be reduced considerably by using the partitioning method with the lowest S/V ratio for the number of processors available.

The main advantage of the new methods is a reduction in communication volume. The number of messages sent/received per processor is 26 for SC, 14 for BCC, and 18 for FCC. This number can be important for message-passing

p	partitioning	interior		halo		ratio	ratio/(S/V)
		max	avg	max	avg	halo/interior	
1	SC	20000	20000	0	0	0.000	
2	SC	10012	10000	1832	1830	0.183	0.0458
	BCC	10009	10000	2382	2377	0.238	0.0457
4	SC	5055	5000	1961	1934	0.387	0.0484
	BCC	5014	5000	2020	1995	0.399	0.0478
	FCC	5019	5000	2156	2139	0.428	0.0504
8	SC	2551	2500	1546	1517	0.607	0.0506
	BCC	2534	2500	1457	1393	0.557	0.0507
	FCC	2535	2500	1534	1517	0.607	0.0515
12	SC	1708	1667	1244	1206	0.723	0.0517
	BCC	1694	1667	1190	1159	0.695	0.0507
	FCC	1720	1667	1361	1332	0.799	0.0516
16	SC	1284	1250	1054	1033	0.826	0.0517
	BCC	1277	1250	904	874	0.699	0.0522
	FCC	1278	1250	991	961	0.769	0.0527
24	SC	857	833	817	790	0.948	0.0527
	BCC	854	833	737	704	0.845	0.0533
	FCC	861	833	834	805	0.966	0.0537
32	SC	649	625	689	661	1.058	0.0529
	BCC	647	625	665	626	1.002	0.0536
	FCC	653	625	614	581	0.930	0.0547

Table 4
The maximum and average number of atoms in the interior of the processor domains and in the halos. Also listed is the ratio between the average number of halo atoms and interior atoms and this ratio divided by the theoretical S/V ratio.

parallel computers that have a high start-up cost for sending messages. The number of messages can be reduced to six for SC by sending data on halo particles via another neighbor [21]. The same can be done for FCC, reducing the number to 12. Thus SC has an advantage in this respect, which however is only important for relatively small halo sizes.

We have shown that the new methods are indeed of practical use, because they are not much more difficult to implement than the SC method and because

they are applicable for a wide range of processor numbers: BCC is applicable for all even p, and FCC and HCP for all multiples of four. For SC, BCC, and FCC, we discussed in detail how to assign efficiently particles to processors and to halos of other processors. In applications, such assignments can be done through generic functions, which we will make available for the main partitioning methods. We have demonstrated our methods in the parallel construction of a model of amorphous silicon, using SC, BCC, and FCC partitioning, and we have obtained significant savings in communication time in this application.

There is still potential for improvement of the S/V ratio for particular numbers of processors. We have discussed the hexagonal partitioning method, which is a two-dimensional method that may be worthwhile for small p (it beats all other methods for $p = 4, 6, 12, 20, 28$), and the octahedral method, which may be useful if p is a multiple of three. For numbers of processors that are prime (a less common case), we have partitioning methods that are better than SC partitioning into p slices. This is outside the scope of the present study, and will be the subject of future work.

Acknowledgements

We thank the National Computing Facilities foundation and the High Performance Applied Computing center at Delft University of Technology for providing access to a Cray T3E. We thank the referees for their comments which greatly helped to improve the exposition of this paper.

References

[1] G. S. Heffelfinger, Comput. Phys. Commun. 128 (2000) 219.

[2] K. Esselink and P. A. J. Hilbers, J. Comput. Phys. 106 (1993) 108.

[3] S. Chynoweth, U. Klomp, and L. Scales, Comput. Phys. Commun. 62 (1991) 297.

[4] R. Koradi, M. Billeter, and P. Güntert, Comput. Phys. Commun. 124 (2000) 139.

[5] D. J. Adams, Chem. Phys. Lett. 62 (1979) 329.

[6] D. J. Adams, in: D. Ceperley (Ed.), The Problem of Long-Range Forces in the Computer Simulation of Condensed Media, NRCC Workshop Proceedings, Vol. 9 (1980) p. 13.

[7] S. S. Wang and J. A. Krumhansl, J. Chem. Phys. 56 (1972) 4287.

[8] M. P. Allen and D. J. Tildesley, Computer Simulation of Liquids (Oxford University Press, Oxford, 1987).

[9] F. Wooten, K. Winer, and D. Weaire, Phys. Rev. Lett. 54 (1985) 1392.

[10] F. Wooten and D. Weaire, Solid State Physics 40 (1987) 1.

[11] W. Thomson Lord Kelvin, Phil. Mag. Lett. 24 (1887) 503.

[12] J. Plateau, Statique Expérimentale et Théorique des Liquides Soumis aux Seules Forces Moléculaires (Gauthier-Villars, Ghent, Paris, 1873).

[13] D. Weaire and R. Phelan, Phil. Mag. Lett. 69 (1994) 107.

[14] J. H. Conway and N. J. A. Sloane, Sphere Packings, Lattices and Groups (Third edition, Springer-Verlag, New York, 1998).

[15] G. T. Barkema and N. Mousseau, Phys. Rev. B 62 (2000) 4985.

[16] P. N. Keating, Phys. Rev. 145 (1966) 637.

[17] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller, J. Chem. Phys. 21 (1953) 1087.

[18] B. R. Djordjević, M. F. Thorpe, and F. Wooten, Phys. Rev. B 52 (1995) 5685.

[19] R. L. C. Vink, G. T. Barkema, M. A. Stijnman, and R. H. Bisseling, Phys. Rev. B 64 (2001) 245214.

[20] J. M. D. Hill, B. McColl, D. C. Stefanescu, M. W. Goudreau, K. Lang, S. B. Rao, T. Suel, T. Tsantilas, and R. H. Bisseling, Parallel Comput. 24 (1998) 1947.

[21] G. C. Fox, M. A. Johnson, G. A. Lyzenga, S. W. Otto, J. K. Salmon, and D. W. Walker, Solving Problems on Concurrent Processors: Vol. I, General Techniques and Regular Problems (Prentice Hall, Englewood Cliffs, NJ, 1988).