A high-performance lattice Boltzmann implementation to model flow in porous media

Chongxun Pan, Jan F. Prins, Cass T. Miller

Abstract

We examine the problem of simulating single and multiphase flow in porous medium systems at the pore scale using the lattice Boltzmann (LB) method. The LB method is a powerful approach, but one which is also computationally demanding; the resolution needed to resolve fundamental phenomena at the pore scale leads to very large lattice sizes, and hence substantial computational and memory requirements that necessitate the use of massively parallel computing approaches. Common LB implementations for simulating flow in porous media store the full lattice, making parallelization straightforward but wasteful. We investigate a two-stage implementation consisting of a sparse domain decomposition stage and a simulation stage that avoids the need to store and operate on lattice points located within a solid phase. A set of five domain decomposition approaches are investigated for single and multiphase flow through both homogeneous and heterogeneous porous medium systems on different parallel computing platforms. An orthogonal recursive bisection method yields the best performance of the methods investigated, showing near linear scaling and substantially less storage and computational time than the traditional approach.

1 Introduction

Over the last two decades, the Lattice Boltzmann (LB) method has emerged as a powerful technique for simulating computational fluid dynamics. The LB method approximates fluid properties at the micro-kinetic scale and can handle complex boundary conditions in a relatively straightforward way. LB models are particularly successful for modeling complex systems, such as flow through porous geometries [1, 2, 3, 4], which is difficult to model using conventional macroscopic approaches. However, the LB method is computa-

Preprint submitted to Elsevier Science *8 January 2004*

tionally very expensive in terms of both the floating point operations and storage needed to simulate real systems. As a result, parallel computing is a prerequisite for most LB simulations, and computational limitations will continue to be a significant constraint for the foreseeable future for a wide range of porous medium systems. Therefore, achieving improved parallel performance for LB models is an important area of scholarly pursuit.

Furthermore, the small time scale of the LB method necessitates many timesteps to observe steady-state macroscopic flow behavior. Consequently a high-performance implementation must be able to utilize a large number of processors efficiently to minimize the time per step; it is not sufficient to gain high performance only for large simulations with low update rates. Rapid updates for modest lattice sizes using a large number of processors is a challenging implementation problem.

To our knowledge, all previously published implementations of LB methods to simulate flow in porous media are based on a full lattice representation (FLR) in which both fluid and solid lattice sites are stored and computed in a regular computational grid. This approach leads to straightforward parallel processing implementations but is wasteful in floating point operations and storage. This is so because lattice points that fall within a solid phase are not needed to resolve the flow field—one need only know their existence and geometric distribution. Using a FLR approach, parallelization can be implemented using one of three standard approaches corresponding to the dimensionality of the decomposition: a one-dimensional (1D) slice decomposition, a two-dimensional (2D) shaft decomposition, or a three-dimensional (3D) cubic decomposition. 1D slice and 2D shaft decompositions can be advantageous at small processor counts and in the presence of periodic boundary conditions. These decompositions result in equivalent-sized portions of the lattice distributed among each processor, but they do not account for the pore distributions of the specific porous medium. Therefore, these regular decompositions can cause a serious load imbalance for cases in which a large number of processors are required or the porous medium is highly heterogeneous. Identification and analysis of the performance of decomposition approaches for modeling single and multiphase flow in complex porous medium systems is an open issue. In a non-porous-medium LB application, Kandhai et al. [5] applied the orthogonal recursive bisection (ORB) method to generate approximately balanced subdomains for single-phase flow simulations in a chamber with a varying cross section.

The objectives of this work are (1) to develop an approach for separating domain decomposition approaches suitable for parallelization from LB simulation approaches; (2) to investigate a range of domain decomposition approaches for parallel processing performance; (3) to compare the performance of the new approaches to the performance of standard approaches; and (4) to investigate the sensitivity of the approaches' performance to the type of flow (single-phase or multiphase), the nature of the porous medium, and the characteristics of the parallel machine.

2 Formulation of lattice Boltzmann methods

The LB method approximates a solution to the Boltzmann equation on a regular lattice using a set of distribution functions. It is necessary to consider some of the details of this formulation in order to understand the parallel implementation aspects of this work, which are the focus of this effort. Following our previous work [6], we use the so-called d3q15 lattice structure, where "d3" indicates three dimensions, and "q15" indicates that each lattice point has 15 velocity vectors, $\mathbf{e}_i$ $(i = 1, 2, \cdots, 15)$, where $|\mathbf{e}_i| = 1$ for $i = 1, \ldots 6$, $|\mathbf{e}_i| = \sqrt{3}$ for $i = 7, \ldots, 14$, and $|\mathbf{e}_{15}| = 0$.

2.1 Single-phase flow

The primary variables are particle distribution functions $f_i(\mathbf{x}, t)$, which represent the probability of finding a fluid particle with velocity $\mathbf{e}_i$ at location $\mathbf{x}$ and time t. Macroscopic parameters are determined by the integration of the distribution functions. For example, the fluid density is given by $\rho = \sum_i f_i$ and the velocity by $\mathbf{v} = \sum_i f_i \mathbf{e}_i / \rho$. The evolution of $f_i(\mathbf{x}, t)$ in time is defined by the LB equation:

$$f_i(\mathbf{x} + \mathbf{e}_i, t + 1) - f_i(\mathbf{x}, t) = \frac{1}{\tau} \left[f_i^{(eq)} - f_i(\mathbf{x}, t)(\mathbf{x}, t) \right], \qquad (1)$$

where τ is the relaxation time. Equation (1) indicates that the fluid particle evolution involves two steps, namely traveling and collision steps. At the traveling step, each particle propagates to a neighboring lattice site in the direction of $\mathbf{e}_i$, as shown on the left-hand side of the equation. The right-hand side represents the collision term, which is simplified by the so-called BGK (Bhatnagar-Gross-Krook), or the single-time, relaxation approximation to the equilibrium distribution function $f_i^{(eq)}$ [7]. The functional form of $f_i^{(eq)} = f(\rho, \mathbf{v}^{(eq)})$ can be chosen (see, e.g. [8]), such that the fluid obeys the isothermal Navier-Stokes equation in the limit of incompressible flow, where the kinematic viscosity is given by $\nu = (2\tau - 1)/6$ [9].

We implement the no-slip boundary condition at a solid surface by a bounce-back scheme [10]. The bounce-back scheme assumes the wall with the no-slip boundary condition is located midway between the solid node and the fluid node. At the inlet and outlet of the medium, we apply constant pressures by implementing the pressure boundary conditions as described in [9, 11]; the flow field driven by the pressure difference can then be simulated.

Note that in the single-phase LB scheme, because of inherent spatial locality during the collision step, information exchange between processors is only required between neighboring particles at the traveling step. This has clear implications for parallel processing implementations.

2.2 Multiphase flow

More fluid components can be added to the LB model by the addition of fluid particles that possess different fluid properties. Similar to the single-phase LB equation, the LB equation for the kth fluid is given by [8]:

$$f_i^k(\mathbf{x} + \mathbf{e}_i, t + 1) - f_i^k(\mathbf{x}, t) = \frac{1}{\tau^k} \left[f_i^{k(eq)}(\mathbf{x}, t) - f_i^k(\mathbf{x}, t) \right], \quad i = 1, 2, \cdots, 15 \tag{2}$$

where τ^k is the relaxation time of the kth fluid. Fluid density of the kth component ρ^k, fluid velocity $\mathbf{v}^k$, and common velocity $\mathbf{v}$ are obtained by

$$\rho^k(\mathbf{x},t) = \sum_i f_i^k(\mathbf{x},t); \quad \mathbf{v}^k(\mathbf{x},t) = \frac{\sum_i f_i^k(\mathbf{x},t)\mathbf{e}_i}{\rho^k(\mathbf{x},t)}; \quad \mathbf{v}(\mathbf{x},t) = \frac{\sum_k(\rho^k\mathbf{v}^k/\tau^k)}{\sum_k(\rho^k/\tau^k)} \tag{3}$$

To simulate multiphase flow in porous media, long-range interactions $\mathbf{F}^k = \mathbf{F}^k_{f-f} + \mathbf{F}^k_{f-s}$ must be included, where $\mathbf{F}^k_{f-f}$ is the fluid-fluid interaction force and $\mathbf{F}^k_{f-s}$ is the fluid-solid interaction forces. The momentum change due to the interaction forces is included in the equilibrium function $f_i^{k(eq)} = f(\rho^k, \mathbf{v}^{k(eq)})$, where $\rho^k\mathbf{v}^{k(eq)} = \rho^k\mathbf{v} + \tau^k\mathbf{F^k}$.

We follow the Shan-Chen model [12, 13, 14], in which nearest neighbor interactions are used to define the interparticle forces: i.e., the fluid-fluid interaction force $\mathbf{F}^k_{f-f}$ on fluid k at site $\mathbf{x}$ is the sum of the forces between the fluid k particle at $\mathbf{x}$ and the fluid k' particles at neighboring sites $\mathbf{x}'$

$$\mathbf{F}^k_{f-f}(\mathbf{x}) = -\psi^k(\mathbf{x})\sum_{k'}\sum_{\mathbf{x}'} G_{kk'}(\mathbf{x},\mathbf{x}')\psi^{k'}(\mathbf{x}')(\mathbf{x}'-\mathbf{x}), \tag{4}$$

where $\psi^k(\rho^k)$ is a function of the local fluid density. In Eq. (4), $G_{kk'} = G_{k'k}$ is a symmetric matrix defined by Green's function. By choosing $G_{k'k}$ properly, fluids can separate so that immiscible multiphase flow behavior results [14]. $\mathbf{F}^k_{f-s}$ between the fluid k at site $\mathbf{x}$ and the solid at site $\mathbf{x}'$ was suggested by Martys and Chen [3]:

$$\mathbf{F}^k_{f-s}(\mathbf{x}) = -\rho^k(\mathbf{x})\sum_{\mathbf{x}'} G_{ks}(\mathbf{x},\mathbf{x}')(\mathbf{x}'-\mathbf{x}). \tag{5}$$

To simulate multiphase flow, we add non-wetting phase (NWP) and wetting phase (WP) reservoirs consisting of two lattice layers at both horizontal ends of the medium geometry and enforce fixed-pressure boundary conditions at the ends of these reservoir layers. Additional details regarding multiphase simulation methods are available in our earlier work [15].

Two aspects of LB simulations of multiphase flow in porous media warrant consideration from a parallel processing perspective. First, more communication work is required for the multiphase case than for the single-phase case. This is because information exchange between processors is required not only at the traveling step for each fluid component, but also during the calculations of the interaction forces $\mathbf{F}^k_{f-f}$, for which fluid densities in the neighboring lattice sites are required. Second, the fluid boundary layers added to implement boundary conditions changed the geometry of the simulation domain compared to the original porous medium geometry. As a result, multiphase LB models present a separate, distinct, and more challenging application for parallel processing decomposition and simulation than single-phase LB models in porous medium systems.

3 Approach

3.1 High-level design

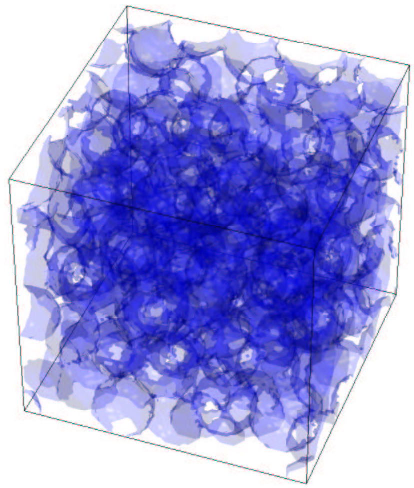
The standard LB implementation stores both fluid and solid nodes in a full lattice, which is usually decomposed into a number of equal-sized, regular-shaped subdomains for parallel processing. By adding ghost layers of lattice points to each of the subdomains, the traveling step can be accomplished by sending values to the ghost layers of the neighboring processors. Neighbor relationships between processors are readily known due to the lattice's simple structure.

However, for many porous medium problems of practical interest, there are two serious drawbacks involved in this approach. First, memory is wasted by storing information for solid phase nodes. For many of these problems, we can assume that the solid phase is immobile. The solid nodes do not participate in calculations of LB equations, so all the variables (e.g. distribution functions, density and velocity) at solid nodes are simply zero. For a typical unconsolidated natural porous medium whose porosity is 0.3–0.4, up to 70% of the memory is wasted storing zero in real numbers, and the porosity of a fractured porous medium can be less than 5%. This results in a severe waste of memory that limits the range and scale of the LB applications that can be addressed. In addition, part of computational time can be wasted either on executing logical operations to distinguish fluid and solid nodes, or on non-productive computation on solid nodes if they are not singled out.

The second drawback of the standard LB approach is the potential load imbalance problem caused by typical domain decomposition approaches. One of the main advantages of the LB method is its suitability for a large class of different geometries and porous medium types, no matter whether the solid phase is homogeneously or heterogeneously distributed over the grid of the geometry. Obviously, for a heterogeneously distributed solid phase, regular domain decomposition approaches can result in a severe load imbalance. Hence the efficiency of the parallel program decreases significantly due to the idle synchronization time.

For a homogeneous porous medium, some researchers suggest that the load imbalance among different processors is negligible [5, 16, 17]. However, due to the fact that homogeneity exists only at a sufficiently large scale for certain types of media, we contend that this assertion is not the case if a large number of processors are used. To demonstrate this point, we show a simple calculation on a medium comprised of a homogeneous sphere pack, labeled as GB1b, the geometry of which is shown in Fig. 1. The porosity of the GB1b is 0.36 and the relative standard deviation of the diameter of the grain size is 4.7%; more properties of GB1b are given in [15]. We discretize the domain with 64^3 lattice points and decompose the lattice into equal-sized cubic subdomains. N_i, the number of fluid nodes distributed to processor i is computed, denoting the work assigned to the processor. If 8 processors are employed, the largest workload $\mathrm{Max}(N_i)$ is 4.8% more than the average workload $\langle N_i \rangle$, which indicates an acceptable balance. When we use 64 processors, $\mathrm{Max}(N_i)$ is 39.7% larger than $\langle N_i \rangle$; when using 512 processors, $\mathrm{Max}(N_i)$ is 138.9% larger than $\langle N_i \rangle$, which means on the average each processor is idle 60% of the time waiting for the slowest processor. We conclude from this example that even for simulations of a homogeneous porous geometry, substantial load imbalance can occur if a large number of processors is used.

To avoid these drawbacks, we have developed an improved LB implementation, which on one hand maintains most of the advantages of a full lattice, while on the other hand reduces the run time and memory requirement, and improves the parallel performance.


(a)

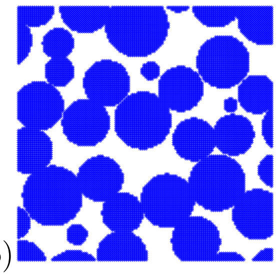

(b)

Fig. 1. Geometry of the homogeneous sphere-packed medium GB1b: (a) 3D isosurface view; and (b) 2D cross section view. Blue and white areas stand for solid and fluid spaces, respectively.

To minimize memory requirements, instead of representing the full lattice, we employ a sparse lattice representation (SLR), which uses an indirect addressing (IA) data structure to store only the fluid node information. Our new LB simulator, the diagram of which is shown in Fig. 2, consists of three parts: (1) a pre-processing routine, (2) a parallel LB simulator, and (3) a post-processing routine. The focus of the study is on the first two parts. The pre-processing routine converts the regular 3D porous medium input to the SLR and decomposes the domain with respect to the number of processors being applied in the parallel LB simulator. The post-processing routine writes the SLR back to a regular 3D lattice output for purposes of flow visualization. It is important to notice that by using the SLR, the natural coordinates of each node in the lattice are hidden, so we can develop generic LB models that are independent of decomposition strategies.

3.2 Sparse representation

The pre-processing routine reads in the full representation of the porous medium, and generates a 1D array V. Each element of V is of derived type node_info, which provides a collection of related pieces of data for each fluid site. Fig. 3 shows a simple example in which a 2D 4×4 porous medium domain is distributed into four processors. In Fig. 3, the grey and white blocks stand for the solid and fluid nodes, respectively. Hence, V in this

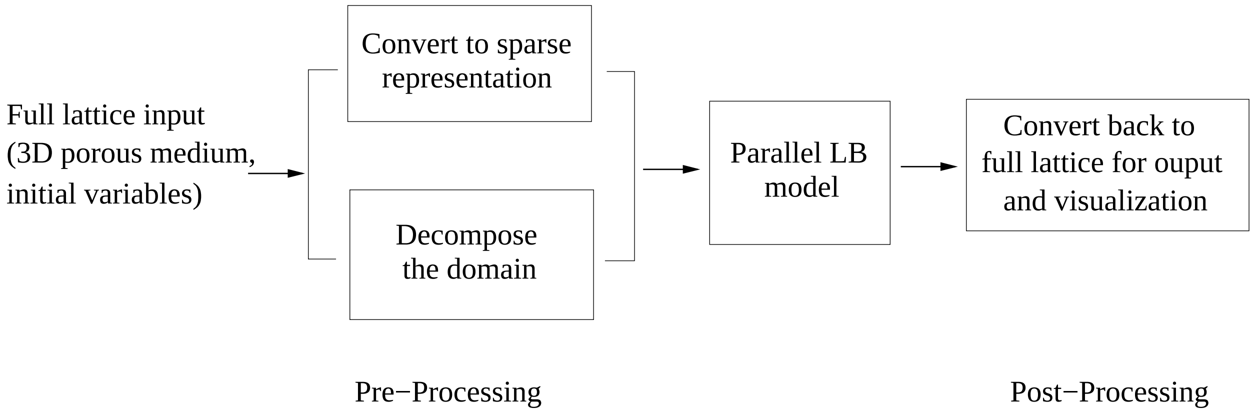


Fig. 2. High-level design of the new LB simulation system.

example consists of 11 elements of node_info, each of which provides the following data for a fluid node:

(1) integer indices corresponding to the x, y and z coordinates of the node in the regularly spaced lattice;
(2) the processor index to which the node is distributed; and
(3) node and processor indices for the 14 neighboring lattice points.

We now describe item (3) in more detail. Having lost the natural ordering, neighboring nodes are no longer immediately known, so the neighboring information has to be added locally. For each node, there are three types of neighbors: (a) a solid node, so the bounce-back scheme applies; (b) a fluid node located inside the same processor; and (c) a fluid node located at a different processor, so message passing between processors is required. For example, the fluid node marked with velocity vectors at the bottom of Fig. 3 has neighbors along directions 6, 7 and 8 that are solid nodes, assuming periodic boundary conditions are applied; neighbors along directions 1, 2 and 5 belong to the second type, and those along directions 3 and 4 belong to the third type.

In order to avoid expensive sorting and searching operations to locate the neighboring fluid nodes in the topologically unstructured grid, the location of the fluid nodes has to be added. For example, as shown in Fig. 3, we need to know that the indices of neighbors along directions 1, 2 and 5 of the marked node in processor (PE) 01 are 3, 2 and 4, respectively, and the neighbors along direction 3 and 4 are correspondingly the second node in PE 03, and the second node in PE 00. That is to say, for each of 14 neighboring nodes, a maximum of 7 bytes of integer storage is required, including a 1-byte indicator to distinguish the above-mentioned three types of neighbors, a 4-byte integer showing the index of the node if it is a fluid node of the second type, and an additional 2-byte integer showing the processor index if it is a fluid node of the third type.

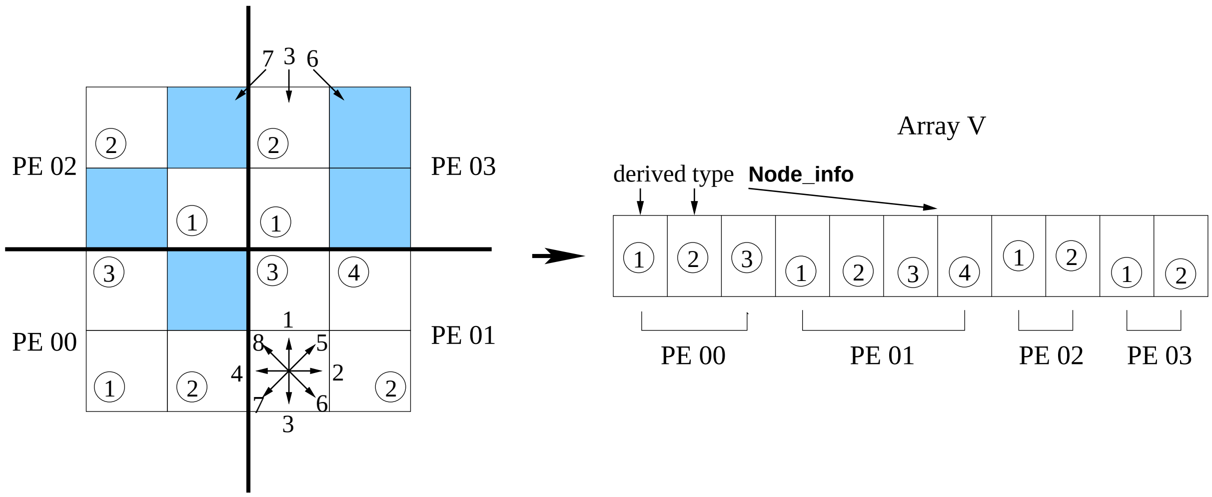


Fig. 3. Conversion from the full lattice structure of the porous medium to the sparse representation. Grey and white areas stand for solid and fluid spaces, respectively.

3.3 Memory requirements

Array V is constructed only once and written into an output file by the pre-processing routine, then in the LB simulation model each processor reads in the corresponding part of the output file to obtain its subdomain information. It is clear that by using the SLR, each processor needs more memory per fluid node to store the medium information, but no memory is allocated for solid nodes. We now compare the total memory requirements for the standard FLR and the SLR. For the FLR approach, the required memory M_{FLR} for a single-phase flow LB model is estimated, assuming a single processor is used, as

$$M_{FLR} = N[1 + 8 + (15 \times 8 \times 2)] = 249N, \tag{6}$$

where N is the total number of nodes in the lattice. The first term on the right-hand side (RHS) represents a 1-byte fluid/solid indicator per lattice site. The next term on the

RHS represents the memory used for the density (8-byte real), and the last represents the distribution functions at 15 directions (8-byte real) for the current and next time steps.

The required memory for the SLR is

$$M_{SLR} = N\phi\{[(3 \times 2) + (14 \times 7)] + [8 + (15 \times 8 \times 2)]\} = 352N\phi, \qquad (7)$$

where ϕ is the porosity of the porous medium. The first term on the RHS stands for the memory used to store medium information per fluid lattice site, including x, y, z coordinates indices (2-byte integer each) and 14 neighboring data (7-byte each, as explained in previous section). The second term on the RHS is for storage of real number density and distribution functions.

From Eq. (6) and Eq. (7), one can see that the SLR achieves memory advantage at $\phi \leq 0.71$, which means for a typical unconsolidated porous medium, the porosity of which ranges from 0.3 to 0.4, approximately half of the memory is reduced for a single-phase LB application. For multiphase LB models, the amount of the saved memory is even more significant, since much more real number storage is required for multiple fluid components, and interaction forces among them. The storage of these real number variables can exceed that of the medium structure information, so consequently M_{SLR}/M_{FLR} approaches the porosity ϕ.

3.4 Domain decomposition strategies

Kandai et al.[5] pointed out that in LB simulations, minimizing the computational load imbalance is more important than minimizing the communicational imbalance. In LB problems, the workload distribution is proportional to the number of fluid nodes. This distribution distribution is static because the solid phase is immobile in our applications. Therefore, to circumvent the workload imbalance induced by the solid phase, we have examined the following non-uniform domain decomposition methods.

(1) Rectilinear partitioning. The grid is split into rectilinear-shaped subdomains such that the workload is balanced, as shown in Fig. 4(a), where the labels identify pro-

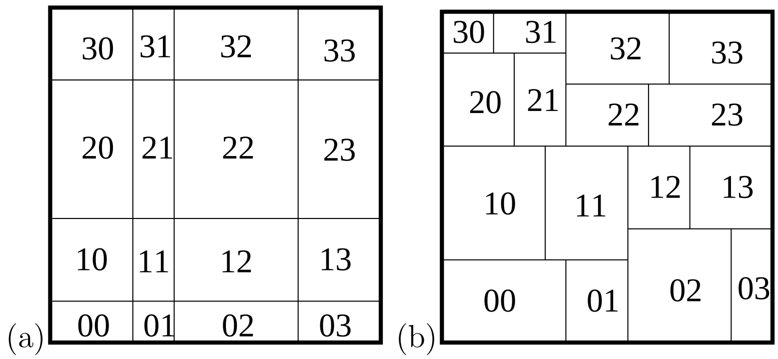


Fig. 4. Two-dimensional examples of the non-uniform domain decompositions on 16 processors: (a) rectilinear partitioning; and (b) orthogonal recursive bisection (ORB) decomposition.

cessors in a 4×4 grid, to which the respective fraction of the domain is mapped. The method has the advantage of being simple to code, and the subdomain boundaries maintain a similar shape as they do in the standard domain decomposition approaches. This partitioning can generate subdomains, the number of which is not necessarily an integer divisor of the lattice size. In other words, the choice of the number of processors employed is less restricted than in the standard decompositions.

(2) Orthogonal recursive bisection (ORB) [18]. ORB techniques partition the domain by subdividing it into equal parts of work by successively subdividing along orthogonal coordinate directions. ORB reduces the higher dimensional partitioning into a set of recursive one-dimensional partitioning problems, with the cutting direction varied at each level of the recursion. ORB partitioning is restricted to $p = 2^k$ processors. Fig. 4(b) illustrates the ORB strategy on a 2D grid, where for $p = 2^4$ processors, recursively subdividing the domain four times yields 16 partitions. There are several other ORB techniques available based on different ways of subdividing the domain: for example, the hierarchical ORB (ORB-H) splits the problem into $p > 2$ parts and recurses on each part, and the ORB-median of medians (ORB-MM) releases the constraints of straight-edged cuts so that highly irregular partitions are generated. For simplicity, we only use the ORB in this study.

In the LB parallel model each processor independently reads in its subdomain, i.e., each processor i stores an array V_i, consisting of N_i (number of fluid nodes in the subdomain) elements, as shown in Fig. 5. Quantities that characterize the LB state, for example, the density Den and distribution functions Dist in single-phase models, are defined for each node in the fluid phase. The size of the array Dist is $15N_i$ because our LB model has 15 fixed velocity vectors, $\mathbf{e}_i$. In this section, we will restrict our description to the single-phase flow case for a clear demonstration. The parallel implementation for the multiphase LB model follows essentially the same communication pattern described here, but involves additional information that must be exchanged.

As mentioned previously, there are three situations that occur during the traveling step, in which each fluid particle hops to a neighboring lattice site in the direction of its velocity $\mathbf{e}_i$. Interprocessor communications are only incurred for the third type, where the neighboring fluid nodes lie in different processors. Due to the symmetry of the lattice structure, the number of messages sent out from processor i is identical to the number received by the same processor N_{ci}. Note that the variation of N_{ci} among processors is a parameter indicating how well the communicational work load is balanced.

Even though the array V_i provides the complete neighboring information needed for message passing between subdomains during the traveling step, these values are scattered throughout the array Dist. To reduce the communication overhead, they must be packed into contiguous messages for transmission, and unpacked on the receiving side. We allocate a small amount of memory for a send buffer and receive buffer to each subdomain. Values required to send to other processors are copied into Send_Dist, a real number array including N_{ci} distribution functions extracted from Dist. Some care must be taken here. In Send_Dist, values sent to the same neighboring processor should be arranged contiguously. Thus, Send_Dist consists of a number of messages, each of which is sent to the corresponding part of the destination processors. The number of messages, equivalent to the number of the neighboring processors, is named N_{pi}. Each processor also receives N_{ci} messages from other processors, held in the receive buffer Recv_Dist, which is partitioned

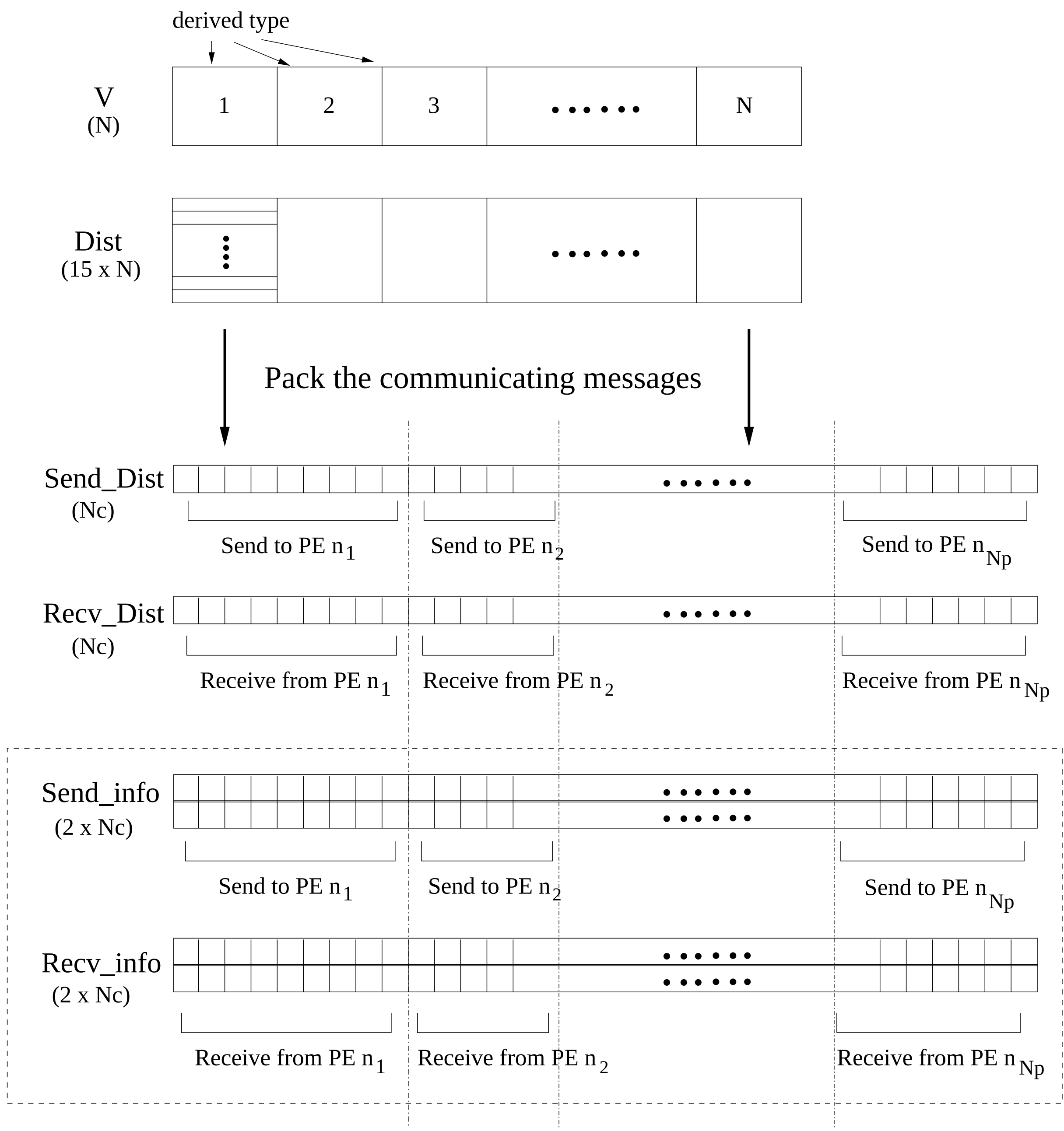


Fig. 5. Message passing implementation for each processor using the the indirect addressing data structure.

to the same sizes as its counterpart Send_Dist.

Having lost the natural coordinates of lattice nodes by using the IA structure, each processor needs to know where to back up the messages from Recv_Dist to the corresponding locations of Dist. Therefore, we add an array Send_info, a two by N_{ci} integer array, to give the auxiliary information for each sending message. This array holds the direction j of the particle velocity $\mathbf{e}_j (1 \leq j \leq 14)$. The second is the index of the node at the neighboring processor, to which the message is sent. As shown in Fig. 5, Send_info also consists of N_{ci} messages, and each message is sent to corresponding part of Recv_info at the destination processor. Moreover, Recv_info and Recv_Dist must be in the corresponding order so that after message transfer, Recv_Dist can correctly update Dist, given the location information provided in Recv_info.

Constructing Send_info and generating Recv_info by point-to-point communications need only be done one time because of the fixed communication pattern once the porous medium is known. The cost is usually negligible when amortized over multiple LB time step iterations, and only a small amount of additional memory is required. By constructing and receiving messages in buffers Send_Dist and Recv_Dist, the communication overhead is reduced at each time step. It also allows the overlapping of the data exchange process between subdomains and internal computation, so one can apply the asynchronous non-blocking message passing routines to achieve good parallel efficiency.

In summary, the algorithm for the parallel LB model is as follows:

(1) Each processor i reads the subdomain decomposed by the pre-processing routine, array V_i with N_i elements generated;
(2) Each processor constructs the integer array Send_info from array V_i corresponding to messages needed to be passed to other processors,;
(3) Each processor exchanges corresponding information about communication sizes and locations with its N_{pi} neighboring processors;
(4) The distribution functions are initialized by the collision step, and the appropriate pressure boundary condition is applied;
(5) Then the LB time iterations proceed until the equilibrium state is reached. At each iteration, the following steps are performed:
    (a) Traveling step starts. Each processor updates Send_Dist from array Dist;
    (b) Asynchronous non-blocking send from Send_Dist and non-blocking receive to Recv_Dist are initialized;

(c) Then computation is performed on the interior nodes, including the bounce-back and traveling between internal nodes;

(d) A wait barrier is issued for the completion of the non-blocking send-receive operation;

(e) Guided by Recv_info, each processor updates Dist from Recv_Dist, which indicates the completion of the travel step;

(f) The collision step is applied to all nodes and the pressure boundary condition is re-applied.

4 Results and discussion

4.1 Computational environments

The new simulators for both single-phase and multiphase flow are written in Fortran 90 using the Message Passing Interface (MPI standard) library. The codes have been tested on a variety of parallel computer platforms, including an IBM RS/6000 SP, IBM p690, SGI Origin 2400, and a Linux Beowulf cluster. The characteristics of these machines are shown in Table 1. Because we need to simulate sufficiently large systems for the performance measurements with a large number of processors, the parallel machine we primarily use is the IBM-SP system. Unless indicated otherwise, results in next sections are obtained on the IBM-SP.

4.2 Sequential performance results

More important than reducing memory consumption, the use of the SLR leads to less computational and communication work than the standard FLR approach. The reason is because all the work related to solid nodes is eliminated, i.e. for a typical porous medium, the simulation work is then limited to 30–40% of the entire lattice sites. We have performed sequential performance tests to compare the run time achieved using each of the two approaches. In Fig. 6, we plot for single-phase and two-phase LB simulations, the average

Table 1
Characteristics of parallel machines used in the work

Machine	IBM RS 6000 SP	IBM p690	SGI Origin 2400	AMD Beowulf Cluster
Configuration	180 4-way SMP nodes	single 32-way SMP node	single 48-way SMP node	64 2-way SMP nodes
Processor	Power-3 II 375 MHz	Power-4 1.3 GHz	MIPS R12000 400 MHz	Athlon MP 1900 1.6 GHz
Memory per node/ total memory	2 GB / 360 GB	128 GB	24 GB	2 GB / 128 GB
L1 Cache I/D	32 KB / 32 KB	32 KB / 32 KB	32 KB / 32 KB	64 KB / 64 KB
L2 (/ L3) Cache	8 MB	1.5 MB / 128 MB	8 MB	256 KB
Peak FLOPS (64-bit)	1.5 GF	5.2 GF	0.8 GF	3.2 GF

CPU time for 100 time steps and varying problem sizes. Note that the largest lattices used for the multiphase simulations were slightly smaller than the single-phase case due to memory limitations on a single processor of the IBM-SP. Two media are used in the test: the GB1b porous medium and a full-fluid medium without a solid phase. In the porous medium test case, we clearly see that the SLR results in significantly faster execution times for both single-phase and two-phase simulations. For single-phase simulations on problem sizes 32^3 and 64^3, the ratio of CPU time between the SLR and FLR is close to 0.36, which is the analytical limit of ϕ.

In the full-fluid test case, we find that, despite its more complicated addressing system which is anticipated to slow down the execution, the SLR is still slightly faster than the FLR, especially for larger problem sizes and two-phase simulations. We hypothesize that during the traveling steps in the FLR, a certain amount of time is spent evaluating the conditional expressions to determine if the neighboring sites are fluid or solid. For multiphase models, in addition to the traveling steps, more time is required during the calculations of nearest neighbor fluid-solid interaction forces. However, these conditional statements are unnecessary in the SLR approach because the neighboring information is analyzed by the pre-processor routine and known locally. To further verify this, we eliminated the bounce-back operations for the FLR approach during the traveling step in the single-phase full-fluid medium simulations. As illustrated with the unfilled squares at

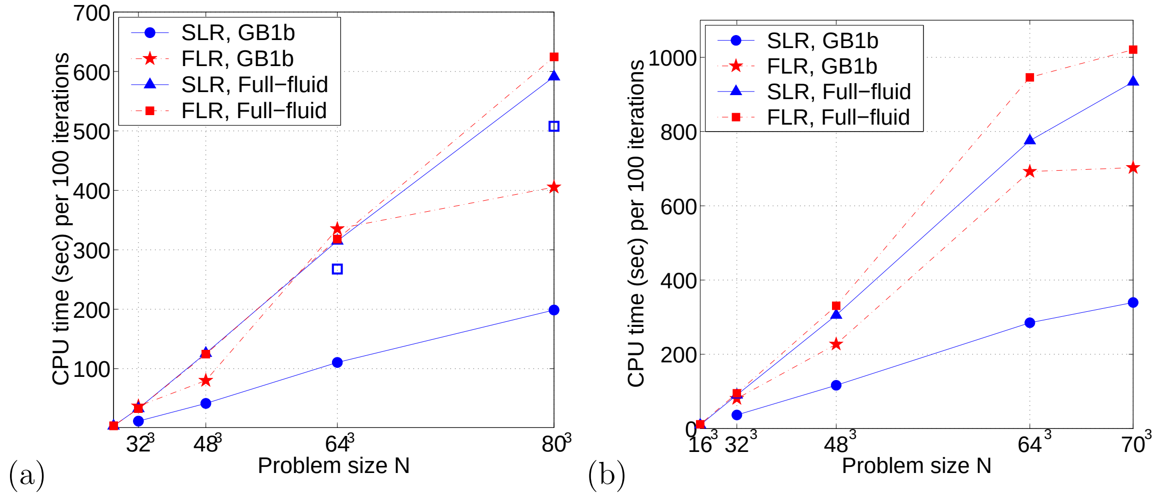


Fig. 6. Comparison of sequential performance using the sparse lattice representation and the full lattice representation on: (a) single-phase flow simulations; and (b) two-phase flow simulations. Media used in the test are, respectively, the GB1b sphere-packed medium with lattice sizes varying from 32^3 to 80^3 ($\phi = 0.34$–0.36), and a full-fluid case with lattice sizes varying from 16^3 to 80^3 ($\phi = 1.0$).

problem size 64^3 and 80^3 in Fig. 6(a), we find that the FLR approach then runs around 20% faster than the SLR approach, which is consistent with our hypothesis.

4.3 Memory hierarchy

In the above implementations, fluid nodes in array V are inserted from the 3D full lattice in a Cartesian order, i.e., starting from x direction and then y and z. To improve locality of reference in multidimensional lattices, we consider an ordering originally presented by Morton in 1966 [19], who defined the indexing of a 2D array as in Fig. 7. Much recent work has shown that for many matrix problems, Morton-order (or Z-order) offers higher locality over the classic Cartesian order and enables algorithms to reduce cache misses (e.g. [20, 21]).

To explore how Morton-order influences LB applications, we performed a comparison experiment. We entered the fluid nodes into array V following the Morton-order enumeration of their indices. We compare the running time on a single processor with the time obtained when V is filled in row-major order. Running time results are reported on three machines: SGI Origin, Beowulf Cluster and IBM-SP. Single-phase LB simulations using a

0	1	2	3	4	5	6	7
8			......				15
16			......				23
24			......				31
32			......				39
40			......				47
48			......				55
56	57	58	59	60	61	62	63

0	1	4	5	16	17	20	21
2	3	6	7	18	19	22	23
8	9	12	13	24	25	28	29
10	11	14	15	26	27	30	31
32	33	36	37	48	49	52	53
34	35	38	39	50	51	54	55
40	41	44	45	56	57	60	61
42	43	46	47	58	59	62	63

Fig. 7. Row-major indexing (left) and Morton-order indexing (right) of a two-dimensional 8×8 matrix.

single processor are carried out on both the GB1b porous medium and the full-fluid with different lattice sizes. The run time ratio between the algorithms using Morton-order and row-major order are shown in Table 2.

Results in Table 2 show that Morton-order processing of the fluid sites provides some performance advantage in all cases. However, for several reasons, the advantage is not dramatic. First, the sparsity of the fluid sites decreases the amount of data reuse available under any ordering of references. Second, the large caches in the IBM and SGI processors decrease the penalty for reference patterns with lower reuse, and the pre-fetch engine on the IBM system decreases the cost of Cartesian ordering in the full-fluid case. However, in the case of a commodity processor like the AMD processor used in the Beowulf cluster, the value of Morton-order references becomes more pronounced, and we expect this behavior to become more prevalent in future commodity processors. Finally, the computational intensity of the LB simulation (i.e., the number of arithmetic operations per memory reference) is sufficiently large to hide the latency of cache misses in the lowest cache levels.

For the remainder of the results reported, the Morton-order evaluation is not used. However, the Morton-order evaluation is independent of the framework of the new implementation and can be applied when needed. Additional work is required to adapt the Morton order to domains whose sizes are not a power of two.

Table 2
Run time ratio between algorithms using Morton-order and row-major order on three machines for sequential single-phase LB simulations performed on the GB1b porous medium and a full fluid system for a range of lattice sizes.

	SGI	Beowulf Cluster	IBM-SP
64^3 (GB1b)	87.3%	93.0%	98.8%
128^3 (GB1b)	89.2%	91.8%	–
256^3 (GB1b)	–	89.5%	–
64^3 (Full-fluid)	73.5%	78.8%	95.2%
128^3 (Full-fluid)	74.3%	77.3%	–

4.4 Parallel performance results

Computational speedup and efficiency are the two typical measurements for parallel performance evaluation. The speedup s is calculated as the ratio of CPU time taken for a parallel algorithm to run on a single processor T_1 and CPU time taken to run on p processors T_p. The efficiency ϵ is a measure of the scalability for a parallel program, defined as the ratio between the speedup s and the number of processors p, i.e.,

$$s = \frac{T_1}{T_p} \tag{8}$$

$$\epsilon = \frac{s}{p}. \tag{9}$$

In our evaluation of parallel performance, running times T_p are relative to a fixed number of LB time steps (step 5 described in Section 3.5). We do not take into account the I/O and the serial initialization time, which consumes a negligible fraction of a typical simulation.

4.4.1 Domain decomposition effect

Important factors controlling the efficiency of parallelization include (1) the balance of computational workload among processors, defined by $B_{\mathrm{cp}} = \mathrm{Max}(N_i)/\langle N_i \rangle$; (2) the balance of the communication workload among processors, defined by $B_{\mathrm{cm}} = \mathrm{Max}(N_{ci})/\langle N_{ci} \rangle$; (3) the ratio between communication and computation work, defined by $\langle N_{ci} \rangle / \langle N_i \rangle$; and (4) N_{pi}, the number of neighboring processors associated with processor i.

We have conducted a series of experiments to compare the different domain decomposition methods and analyze the above-mentioned parameters that influence parallel performance. In Table 3 (also plotted in Fig. 8(a)), we present the single-phase flow simulation results with respect to five domain decomposition methods, namely regular 1D (slice), 2D (shaft), 3D (cubic) decompositions, rectilinear decomposition, and ORB. The test case is carried out on the homogeneous GB1b medium with 64^3 lattice nodes, using 64 processors. Table 4 lists the results for the same medium but using 256 processors. From Tables 3 and 4, we find that computational workload balance is a dominant factor contributing to efficient parallelization and that ORB leads to the best computation-balanced subdomains, hence the optimal parallel efficiency.

The results in Tables 3 and 4 also imply that $\langle N_{ci} \rangle$ and $B_{\rm cm}$ only have a minor impact on the parallel efficiency at the test problem size. To verify this conclusion, we have conducted an experiment to analyze the average and maximum time, over all processors, spent on different steps (step 5(a) through 5(f) described in Section 3.5) in one LB time iteration. Results are listed in Table 5. Four cases from Tables 3 and 4 are considered, including the cubic and ORB decomposition using 64 and 256 processors, respectively. A synchronization barrier is added at the end of step 5(f) for the purpose of accurate time counting, even though the barrier is not needed and not present in the simulator.

In Table 5, the computation time corresponds to all serial work in the simulation step. The message passing time is the time required to initiate the send and receive operations, the communication wait time is the unoverlapped communication time, and the idle time is the total time from imbalance in the simulation step.

In the LB models we apply asynchronous communication in order to overlap the process of message transfer and the interface-independent computation. Provided that the communication/computation ratio $\langle N_{ci} \rangle / \langle N_i \rangle$ is sufficiently small so that the interior node computation time can offset the data exchange time, the overhead of communication can become negligible. However, we only observe partial overlapping in Table 5. The situation could be improved by forcing more interior node computation, such as adding some of the work in the collision step. However, we do not expect much improvement on parallel

Table 3
Comparison of different domain decomposition methods on single-phase simulations of GB1b with 64^3 lattice nodes using 64 processors.

Decompositions	$B_{\text{cp}}(\%)$	$\langle N_{ci} \rangle$	$B_{\text{cm}}(\%)$	N_{pi}	$\langle N_{pi} \rangle$	Efficiency ϵ
Slice ($1 \times 1 \times 64$)	131.6	11385	141.5	2	2	0.765
Shaft ($1 \times 8 \times 8$)	167.0	2559	155.6	7 - 8	8	0.719
Cubic ($4 \times 4 \times 4$)	152.0	1759	156.9	11 - 21	16	0.787
Rectilinear ($4 \times 4 \times 4$)	138.9	1761	148.5	9 - 20	16	0.867
ORB	116.1	1890	142.7	7 - 21	13	1.05

The numbers after the decompositions are respectively nx, ny, nz, which stand for the number of partitions in the x, y, z directions.

Table 4
Comparison of different domain decomposition methods on single-phase simulations of GB1b with 64^3 lattice nodes using 256 processors.

Decompositions	$B_{\text{cp}}(\%)$	$\langle N_{ci} \rangle$	$B_{\text{cm}}(\%)$	N_{pi}	$\langle N_{pi} \rangle$	Efficiency ϵ
Shaft ($1 \times 8 \times 8$)	198.2	2242	177.8	7 - 8	8	0.780
Cubic ($4 \times 4 \times 4$)	233.2	765	220.7	7 - 22	15	0.607
Rectilinear ($4 \times 4 \times 4$)	206.7	768	218.5	7 - 22	15	0.732
ORB	127.0	875	158.7	5 - 20	12	0.867

performance because, in LB models, the total communication time, including the message passing time spent on step 5(b) and communication wait time on step 5(d), contributes a relatively small part of the total execution time (around 5% using 64 processors and 10% using 256 processors). This finding is consistent with the study of Kandai et al.[5], who pointed out that minimizing the computational load imbalance is more important than minimizing the communication imbalance in LB simulations. Hence, the ORB leads to the best parallel efficiency by significantly reducing the idle time, which is largely dependent on the computational load balance.

The major disadvantage associated with the ORB method compared to the regular decomposition methods is the possible irregular communication pattern, which causes higher communication overhead. The irregular communication pattern includes the higher number of neighboring processors N_{pi} and the larger size of messages to be transferred N_{ci} for each processor i. However, we find that in our LB implementations the disadvantages

Table 5
Average and maximum time of all processors (in millisecond) spent on different parts of a single simulation time step.

Time	Computation	Message passing	Communication wait	Idle	Total
Cubic(64 PEs)	15.74/23.60	0.19/0.29	0.88/1.48	6.95/11.86	23.76
ORB (64 PEs)	15.57/17.67	0.17/0.28	0.74/1.12	1.73/2.83	18.21
Cubic(256 PEs)	4.40/9.43	0.20/0.38	0.58/1.07	4.63/7.32	9.81
ORB (256 PEs)	4.39/5.80	0.16/0.52	0.59/1.11	1.36/2.13	6.50

associated with the ORB are insignificant. We have observed, from Table 3 and Table 4, a similar range of N_{pi} among cubic, rectilinear and ORB methods. Interestingly, $\langle N_{pi} \rangle$ of ORB is even less than those of the cubic and rectilinear methods. In addition, compared to the cubic and rectilinear methods, ORB does not lead to substantially larger size of communication work N_{ci}, even using 256 processors. By using the sparse representation, we eliminate the message passing associated with the solid nodes, so with the regular decompositions each processor does not have the same communication work and the same number of the neighboring processors. Moreover, balancing the number of fluid nodes actually helps to balance the size of communication work now that message passing occurs only between fluid nodes.

To further verify our finding, we performed tests on different porous media. Fig. 8(a) and Fig. 8(b) plot B_{cp}, B_{cm} and efficiency ϵ for the homogeneous GB1b medium for both the single-phase and two-phase simulations, respectively. Fig. 8(c) and Fig. 8(d) show the results for a heterogeneous sphere-packed medium, labeled as RSP23. The porosity of RSP23 is 0.33 and the relative standard deviation of the grain size is 64.7%; more properties of the medium are seen in [6]. For two-phase simulations, two additional layers of void space are added at both horizontal sides of the domain, representing the non-wetting phase and wetting phase reservoirs. Thus the flow domain is actually $68 \times 64 \times 64$. It is evident that for both test cases with 64 processors, none of the regular decompositions is efficient for either the homogeneous or heterogeneous medium. Among the three regular decompositions, slice decomposition provides the best workload balance, but it generates subdomains with the highest communication/computation ratio, which results in high communication overhead. Compared with cubic decomposition, rectilinear de-

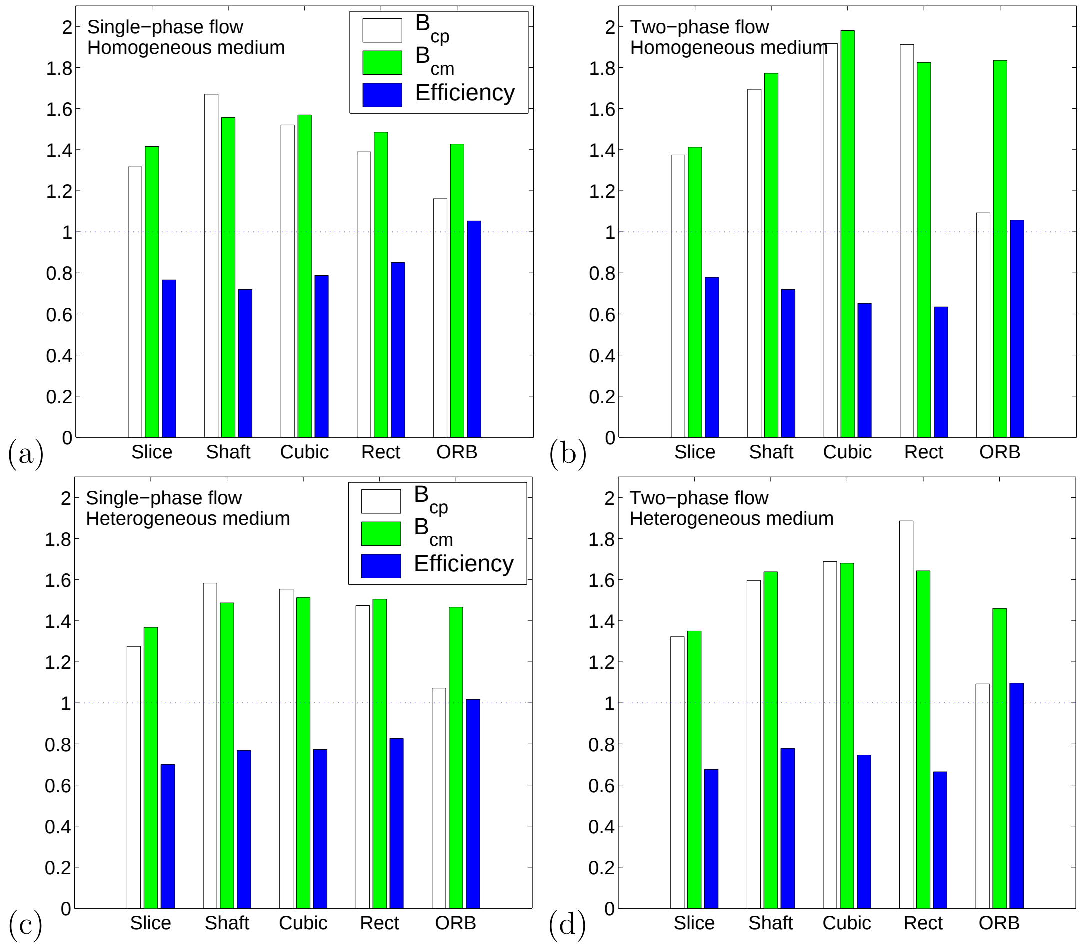


Fig. 8. Comparison among the different domain decomposition methods. In all test cases, the domain size is 64^3 and 64 processors are used: (a) single-phase flow simulation on a homogeneous medium GB1b (porosity $\phi = 0.36$); (b) two-phase flow simulation on GB1b; (c) single-phase flow simulation on a heterogeneous sphere packing medium RSP23 ($\phi = 0.33$); (d) two-phase flow simulation on RSP23. $B_{\rm cp} = {\rm Max}(N_i)/\langle N_i \rangle$ represents the computational load balance, and $B_{\rm cm} = {\rm Max}(N_{ci})/\langle N_{ci} \rangle$ represents the communicational load balance.

composition slightly improves workload balance for the homogeneous medium, but for the heterogeneously distributed media, it does not have much of an advantage in terms of computational balance and parallel efficiency. On the other hand, for single-phase and two-phase simulations on both porous media, the computational workloads obtained by the ORB are approximately balanced. For our test problem size and 64 processors, the ORB decomposition is on average 37% more efficient than regular decompositions for the GB1b medium, and 49% for the heterogeneous RSP23 medium.

Furthermore, in Fig. 9 we show, for a fixed lattice size (GB1b with 64^3), the speedups

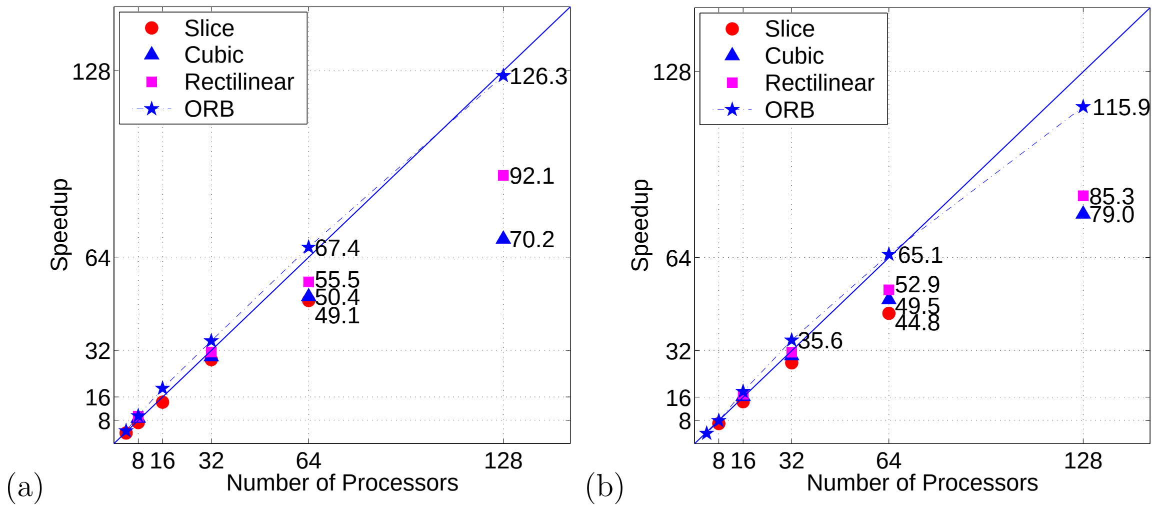


Fig. 9. Speedup s with respect to single processor vs. the number of processor p on GB1b medium with 64^3 lattice nodes using different domain decomposition methods: (a) single-phase flow simulation; and (b) two-phase flow simulation.

versus increasing numbers of processors (up to 128) using slice, cubic, rectilinear and ORB methods. We can observe that (1) ORB improves the parallel efficiency significantly in LB applications compared to standard approaches, especially when a large number of processors is applied; (2) rectilinear partitioning improves parallel performance to a limited extent; and (3) for homogeneous porous media, while using a moderate number of processors, regular decomposition methods can be acceptable approaches, but they cause serious workload imbalance problems when a large number of processors are used.

4.4.2 Lattice size effect

Next, Fig. 10 shows the parallel efficiencies using the new LB implementation and ORB decomposition on both single-phase and two-phase simulations, for various lattice sizes N and a different numbers of processors p. Efficiency ϵ of problem size 64^3 is with respect to the CPU time of a single processor, while ϵ of 128^3 is with respect to 8 processors, and ϵ of 256^3 to 32 processors due to the memory constraint on the IBM-SP. The results in Fig. 10(a) and Fig. 10(b) show that ϵ first increases due to the cache effect then decreases steadily for the 64^3 lattice, because of scaling problems as p increases in size. In other words, when p increases to a certain number (more than 128 processors in our case), the communication time becomes dominant. We expected this behavior. We also see the

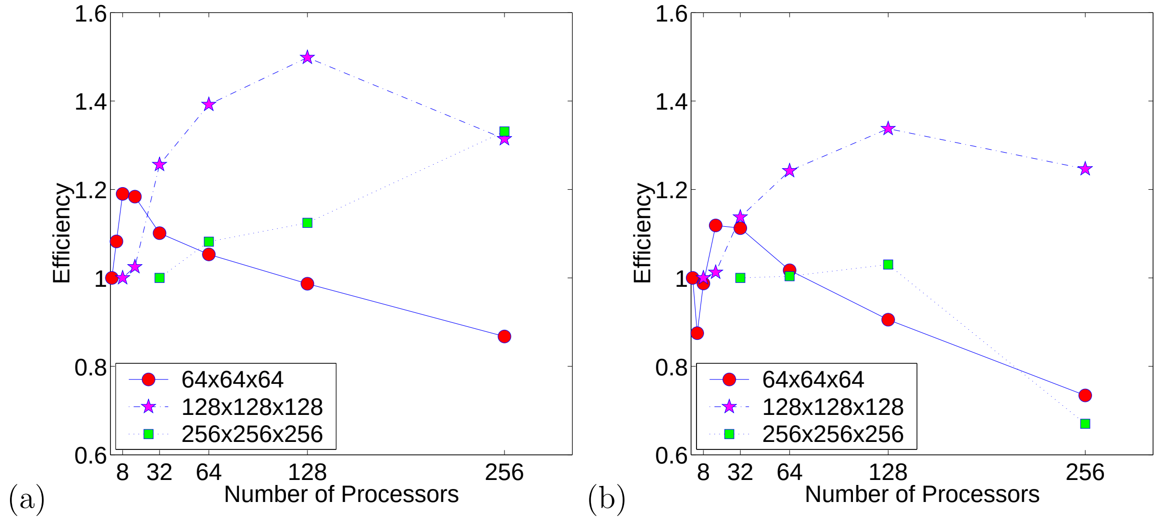


Fig. 10. Parallel efficiency ϵ vs. the number of processors p on GB1b medium with different problem size 64^3, 128^3 and 256^3 for (a) single-phase flow simulations; and (b) two-phase flow simulations. The ORB decomposition method is applied. Efficiency ϵ of problem size 64^3 is with respect to the CPU time of a single processor, ϵ of 128^3 is with respect to 8 processors, and ϵ of 256^3 to 32 processors.

superlinear speedups achieved on both 128^3 and 256^3 lattices, because more of the problem fits in the additional cache memory when more processors are used. The results prove that the parallel implementation method can be scaled up with respect to both the number of processors and the problem sizes.

4.4.3 Platform effect

Finally, we plot in Fig. 11 the total execution time for 300 time steps versus p on different parallel computer platforms at the North Carolina Supercomputing center, including the IBM-SP, SGI Origin 2400, IBM p690 and the Linux Beowulf cluster. For a clear comparison, the speedup values with respect to the single processor CPU time on the IBM-SP are indicated in the figure. We find that the performance results are not exactly in order with those predicted by Table 1. We observe faster execution on the SGI than on the IBM-SP even though the SGI has lower peak FLOPS than IBM-SP. For all the parallel computers, the execution time is nearly proportional to $1/p$, which implies good parallel performance of our LB implementation. Superlinear speedup also occurs with 16 processors on the SGI due to the cache effects.

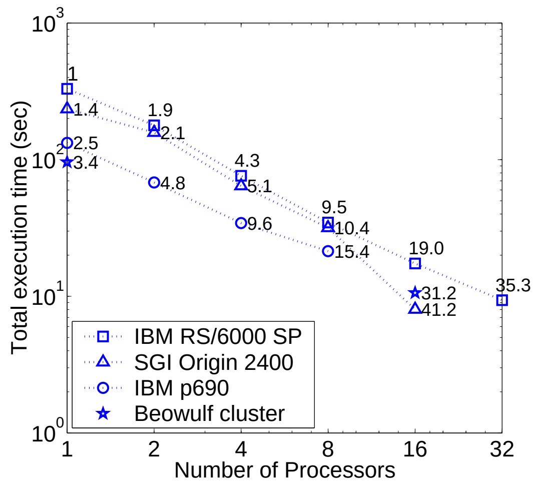


Fig. 11. Total execution time for 300 time steps vs. p on different parallel computers. The numbers shown in the figure are the corresponding speedup values relative to the CPU time of a single processor on the IBM-SP.

5 Conclusions

We have developed a new sparse representation approach for simulating single-phase and multiphase flow in porous medium systems using the lattice Boltzmann method. Standard approaches represent the full lattice of a porous medium using a regular grid, and the parallelization is based on regular 1D, 2D or 3D decompositions of the grid. We have shown that such approaches have serious disadvantages, including waste of computational time and memory, as well as a load imbalance problem due to the presence of the solid phase. The SLR approach is implemented only to store information related to the fluid phase nodes of the porous media, using an IA data structure. For sequential runs significantly faster execution is achieved compared to a code based upon the standard approach. The use of the IA data structure also allows us to develop generic parallel LB simulators that decouple from the domain decomposition step.

To achieve improved parallel efficiency, we have investigated several domain decomposition strategies, including standard decompositions, rectilinear decomposition, and an orthogonal recursive bisection (ORB) decomposition. We find that for the cases examined in this work the computational workload balance is a dominating factor contributing to efficient parallelization. Regular decompositions are acceptable approaches only for homogeneous media and a moderate number of processors. Otherwise, they lead to poor parallel efficiencies due to workload imbalance problems. Rectilinear decomposition improves parallel performance to a limited extent because of its geometric constraints of straight-line cutting.

We have shown that the ORB method leads to good workload balance among subdomains and that excellent parallel efficiencies are obtained on both homogeneous and heterogeneous media for single-phase and multiphase LB applications. We also have shown that the ORB domain decomposition method can be scaled up with respect to both the number of processors and the problem size to achieve good performance on different parallel computers. We believe that our new LB implementation combined with the ORB decomposition is a promising approach that can greatly enhance the applicability of LB methods for a wide range of flow and transport simulations at a practical scale.

Acknowledgments

This work was supported by National Science Foundation grants EAR-9901660 and DMS-0112069, National Institute of Environmental Health Science grant P42 ES05948, and a generous allocation of supercomputing time from the North Carolina Super Computing Center. We thank Mark Reed for useful discussions and assistance with MPI implementations of our codes.

References

[1] O. van Genabeek and D. H. Rothman. Macroscopic manifestations of microscopic flows through porous media: Phenomenology from simulation. *Annual Review of Earth and Planetary Sciences*, 24:63–87, 1996.

[2] S. Chen and G. D. Doolen. Lattice Boltzmann method for fluid flows. *Annual Review of Fluid Mechanics*, 30:329–364, 1998.

[3] N. Martys and H. Chen. Simulation of multicomponent fluids in complex three-

dimensional geometries by the lattice Boltzmann method. *Physical Review E*, 53(1b):743–750, 1996.

[4] D. Zhang, R. Zhang, S. Chen, and W. E. Soll. Pore scale study of flow in porous media: Scale dependency, REV, and statistical REV. *Geophysical Research Letters*, 27(8):1195–1198, 2000.

[5] D. Kandhai, A. Koponen, A.G. Hoekstra, Katajam M., J. Timonen, and P.M.A. Sloot. Lattice-Boltzmann hydrodynamics on parallel systems. *Computer Physics Communications*, 111:14–26, 1998.

[6] C. Pan, M. Hilpert, and C. T. Miller. Pore-scale modeling of saturated permeabilities in random sphere packings. *Physical Review E*, 64(6):article number 066702, 2001.

[7] P. Bhatnagar, E. Gross, and M. Krook. A model for collsion processes in gases. *Physical Review*, 94:511–525, 1954.

[8] S. L. Hou, X. W. Shan, Q. S. Zuo, G. D. Doolen, and W. E. Soll. Evaluation of two lattice Boltzmann models for multiphase flows. *Journal of Computational Physics*, 138(2):695–713, 1997.

[9] Q. Zou and X. He. On pressure and velocity boundary conditions for the lattice Boltzmann BGK model. *Physics of Fluids*, 9(6):1591–1598, 1997.

[10] R. S. Maier, D. M. Kroll, Y. E. Kutsovsky, H. T. Davis, and R. S. Bernard. Simulation of flow through bead packs using the lattice Boltzmann method. *Physics of Fluids*, 10(1):60–74, 1998.

[11] R. S. Maier, R. S. Bernard, and D. W. Grunau. Boundary conditions for the lattice Boltzmann method. *Physics of Fluids*, 8(7):1788–1801, 1996.

[12] X. Shan and H. Chen. Lattice Boltzmann model for simulating flows with multiple phases and components. *Physical Review E*, 47:1815–1819, 1993.

[13] X. Shan and H. Chen. Simulation of nonideal gases and liquid-gas phase transitions by the lattice Boltzmann equation. *Physical Review E*, 49(4):2941–2948, 1994.

[14] X. Shan and G. Doolen. Diffusion in a multi-conponent lattice Boltzmann equation model. *Physical Review E*, 54(4):3614–3620, 1996.

[15] C. Pan, M. Hilpert, and C. T. Miller. Lattice-Boltzmann simulation of multiphase flow in water-wet porous media. In *Water Resources Research Institute of the University of North Carolina 2002 Annual Conference*, 2002.

[16] J.C. Desplat, I. Pagonabarraga, and P. Bladon. Ludwig: A parallel lattice-Boltzmann code for complex fluids. *Computer Physics Communications*, 134:273–290, 2001.

[17] N. Satofuka and T. Nishioka. Parallelization of lattice Boltzmann method for incompressible flow computations. *Computational Mechanics*, 23:164–171, 1999.

[18] V. Kumar, Grama. A., Gupta. A., and Karypis. G. *Parallel Computing.* The Benjamin/Cummings Publishing Company, Inc., California, 1994.

[19] G. M. Morton. *A computer oriented geodetic data base and a new technique in file sequencing*. IBM Ltd, Ottawa, Ontario, 1966.

[20] M. M. Anguh and R. R. Martin. A two-dimensional inplace truncation walsh tranform method. *Journal of Visual Communication and Image Representation*, 7(2):116–125, 1996.

[21] D. S. Wise, G. A. Alexander, J. D. Frens, , and Y. H. Gu. Language support for morton-order matrics. *ACM Sigplan Notices*, 36(7):24–33, 2001.