Petascale Application of a Coupled CPU-GPU Algorithm for Simulation and Analysis of Multiphase Flow Solutions in Porous Medium Systems

James E. McClure*, Hao Wang †, Jan F. Prins‡ Cass T. Miller§ and Wu-chun Feng †,
*Advanced Research Computing
Virginia Tech, Blacksburg, Virginia
Email: mcclurej@vt.edu
‡Dept. of Computer Science
University of North Carolina at Chapel Hill,
Chapel Hill, North Carolina
Email:prins@cs.unc.edu
§Dept. of Environmental Science & Engineering
University of North Carolina at Chapel Hill,
Chapel Hill, North Carolina
Email:casey_miller@unc.edu
†Dept. of Computer Science
Virginia Tech, Blacksburg, Virginia
Email: {hwang121, wfeng}@vt.edu

***Abstract*—Large-scale simulation can provide a wide range of information needed to develop and validate theoretical models for multiphase flow in porous medium systems. In this paper, we consider a coupled solution in which a multiphase flow simulator is coupled to an analysis approach used to extract the interfacial geometries as the flow evolves. This has been implemented using MPI to target heterogeneous nodes equipped with GPUs. The GPUs evolve the multiphase flow solution using the lattice Boltzmann method while the CPUs compute upscaled measures of the morphology and topology of the phase distributions and their rate of evolution. Our approach is demonstrated to scale to 4,096 GPUs and 65,536 CPU cores to achieve a maximum performance of 244,754 million-lattice-node updates per second (MLUPS) in double precision execution on Titan. In turn, this approach increases the size of systems that can be considered by an order of magnitude compared with previous work and enables detailed in situ tracking of averaged flow quantities at temporal resolutions that were previously impossible. Furthermore, it virtually eliminates the need for post-processing and intensive I/O and mitigates the potential loss of data associated with node failures.**

***Keywords*-CUDA, Heterogeneous Computing, Parallel Computing, GPGPU, Lattice Boltzmann Method**

I. Introduction

Multiphase flow processes in porous media are operative in many applications that include carbon sequestration, remediation of environmental contaminants and enhanced oil recovery. The ability to describe these processes in an averaged sense is essential due to the size and complexity of these systems, and computational studies have great promise to advance fundamental understanding of the transport behavior. Multi-scale descriptions have gained traction as a concrete way to resolve deficiencies in the existing mathematical formulations that are typically applied to describe multiphase transport phenomena in porous media [1, 2]. These development advance a theoretical basis for upscaled models and provide both general and specific forms of closures relations that can be evaluated, advanced, and validated using high resolution simulations that have until now been beyond reach computationally. Theoretically approaches have already established a rigorous connection between microscopic flow processes and the larger-scale macroscopic behavior, and accurately tracking changes in interfacial areas has been identified as critically important to the closure of thermodynamically-consistent models [3]. Directly simulating the microscopic details of flow processes that occur within the complex micrometer-sized interstices within the solid geometry provides a way to advance these models.

The lattice Boltzmann method (LBM) is well-established as a useful approach for numerical investigation of multiphase flow in porous media [4, 5, 6, 7]. The computational properties of the LBM make it well-suited for parallel implementation, and the performance and scalability for GPU-based implementations have been demonstrated for multiple computational fluid dynamics applications [8, 9, 10]. For a multiphase implementation the LBM targeting multiple GPU with MPI, Wang et al. reported performance of 2850 MLUPS on 64 GPUs, demonstrating that excellent weak scaling can be obtained provided that sufficiently large sub-domains

are retained by each MPI process [11]. In order to extend this performance to flow in porous media, GPU implementations must not only solve for the multiphase flow, but also accommodate complex solid boundary conditions within GPU kernels. An additional challenge is presented by the need to track the dynamics of multiphase flow in large, complex systems *in situ* so that the need to save the simulation state to disk at regular intervals can be avoided.

In this paper we construct a scalable approach to the simulation and analysis of multiphase flows that takes advantage of heterogeneous parallelism by coupling an GPU-based solution for the multiphase flow problem to an analysis algorithm used to extract morphological information on the CPU. This approach enables the practical application of the LBM to study multiphase flow in porous media on the world's largest GPU-equipped supercomputer. The implementation achieves near-ideal weak scaling and achieves a performance of $244,754$ MLUPS for double-precision execution on $4,096$ GPU. The specific contributions of this work are as follows:

1) development of a GPU-based simulator for multiphase flow in porous media using CUDA and MPI;
2) significant reduction in the need for I/O, data storage, data transfer and post-processing analysis due to performing all analysis on the CPU during the course of a simulation;
3) evaluation of the scaling and load-balance for the developed algorithm and demonstrate the performance; and
4) show that the method can be used to extract previously inaccessible scientific results from large-scale simulations.

II. Background

A. Scientific Merit

Porous media include a wide range of natural and synthetic materials in which flow processes can occur in the interior spaces, also known as the pore-space. Direct simulation of the microscopic details of multiphase flow processes requires knowledge of the pore-space geometry in which this flow occurs. In order to obtain sufficiently complex geometries, dense sphere packs are often used. The spheres represent the solid phase s, which is immobile and non-deformable for our simulations. An example sphere packs is shown in Fig. 1, in which non-overlapping spheres with log-normally distributed radii have been inserted into a cubic domain to serve as surrogate porous media [12]. The region of the domain occupied by solid phase is denoted by Ω_s. In many cases it is important to predict the behavior of two fluid phases, which are referred to as the wetting (w) and non-wetting (n) fluids.

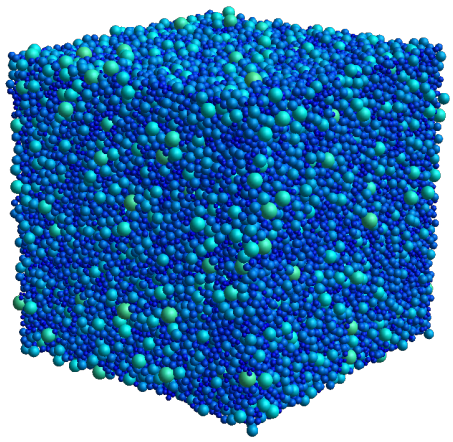

Figure 1. An example of a porous medium system constructed from a random close packing of 85,519 spheres with log-normally distributed radii

This classification is determined based on the contact angle formed where the two fluids meet the solid phase.

For two-phase flow in porous media, capillary forces tend to dominate other effects and the evolution of the interfaces is a main factor that determines the equilibrium and non-equilibrium behavior. Extracting information about the dynamic behavior of interfaces is not currently possible in an experimental setting, and computational approaches currently provide the most practical way to generate detailed insight into these processes. For the systems considered in this work, three interfaces can exist in: the interface between the wetting and non-wetting fluids Ω_{wn}, the interface between the wetting fluid and the solid Ω_{ws} and the interface between the non-wetting fluid and the solid Ω_{ns}. Existing approaches used to have been evaluate interfacial area in porous medium systems that are approximately 400^3 in size [13]. In this work we develop an approach that can be applied to systems that are order of magnitude larger, with analysis applied thousands of times over the course of a single simulation.

In order to develop a coherent macroscopic multiphase flow model for porous media, it is necessary to comprehend the behavior of a wide range of averaged quantities that are obtained by integrating quantities over the phase volumes, interfaces between phases, and the common curve where all three phases coexist [1]. In this work we consider a relatively simple computational experiment that is of direct importance to the advancement of multi-scale models for multiphase flow in porous media. A non-equilibrium configuration of multiple phases is initialized within the pore-space of

a sphere packing, then the interfacial areas a_{wn}, a_{ws} and a_{ns} are evaluated numerically as the system relaxes toward equilibrium. The interfacial areas are computed directly at intervals of T time steps throughout the course of simulation. As a measure of the approach to equilibrium, we evaluate the change in surface energy:

$$\Delta E_s = \gamma_{wn}\Delta a_{wn} + (\gamma_{ns} - \gamma_{ws})\Delta a_{ns}. \quad (1)$$

The surface energy for the wn interface, γ_{wn}, can be computed based on the parameters for the simulation. The difference between the ns and ws interfacial energies, given by γ_{ns} and γ_{ns} respectively, is related to the equilibrium contact angle by Young's equation [14]. The contact angle can be determined from on the simulation parameters, permitting the computation of ΔE_s based on the time history of interfacial areas [15]. It is important to note that while the evaluation of the interfacial areas is a simple calculation, due to the fact that a numerical approximation for the interfaces is constructed explicitly it is also possible to track the transient behavior of virtually any scalar, vector, or tensor quantity averaged over the phase volumes, interfaces or the common curve.

B. GPU-Accelerated Supercomputer Architecture

Due to their superior performance and energy efficiency, GPUs are widely used to speedup scientific computational applications. In the Top500 list [16] published in November 2013, 41 systems, including two of the top ten supercomputers, were installed with GPUs. Since we carry out our experimental evaluations on Titan supercomputer, ranking on No. 2 in the Top500 list to date, we introduce Titan as the GPU-accelerated supercomputer architecture.

Titan uses NVIDIA Kepler K20x GPU on the computational nodes. Each Kepler K20x GPU has 6 GB GDDR5 SDRAM device memory and 2688 CUDA Cores in total. The peak performance of a GPU is 3.95 TFlops and 1.31 TFlops in single precision and double precision, respectively. A GPU contains 14 streaming multiprocessors (SMs), each of which consists of 192 CUDA Cores as SIMD (single instruction, multiple data stream) units. The device memory, also know as the global memory, is shared by all the SMs in a GPU with 250 GB/sec peak bandwidth.

We use Compute Unified Device Architecture (CUDA) programming model provided by NVIDIA to compose our program. The CUDA program consists of CPU codes and GPU codes. The functions that will be running on GPUs are called GPU kernels. A kernel function will be running in parallel by a large number of threads on GPU. The threads are grouped into the block of threads and the grid of blocks. When a kernel is launched (or offloaded) by the CPU, the parameters, such as the number of threads per block and the number of blocks per grid, should be specified. Other major parts of Titan include the AMD Opteron 6274 hexa-core processor and 32GB DDR3 main memory on each node, and Cray Gemini interconnect to connect nodes with a 3D torus architecture. We use CUDA for the parallel computation on GPU and MPI for the parallel communication on multiple nodes.

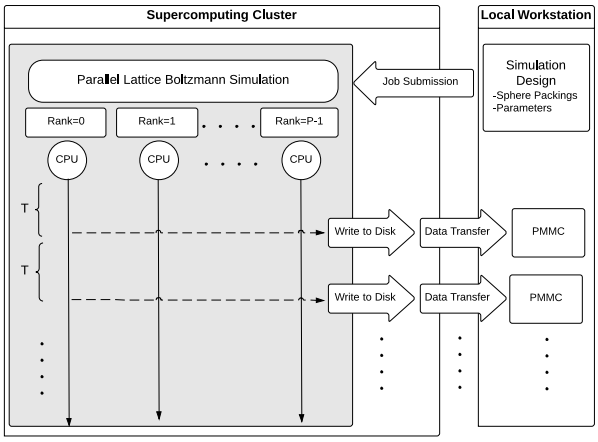

Figure 2. An example workflow depicting the simulation and analysis of multiphase flow. Simulation data is written to disk and transferred from the supercomputer to a local workstation where analysis is performed. The time interval for analysis T is thereby constrained by data movement and analysis bottlenecks.

III. Computational Approach

A. Overall Simulator Design

This work considers a multiphase implementation of the LBM that is written in CUDA and MPI and coupled to an image processing code (written in C++) that extracts the interfaces averages during the course of simulation. The multiphase flow simulator is a GPU-based implementation of the color model lattice Boltzmann scheme developed by McClure et al., a scheme that was constructed to use both GPU and multi-core CPU in tandem [15]. In this work, all LBM computations are performed on the GPU and CPU are dedicated to carrying out image-processing computations needed to extract information from the simulation. This analysis is performed using a porous medium marching cubes (PMMC) algorithm developed to extract the interfaces and calculate a variety of other averages [17]. The details for each aspect of this approach are presented in the following sections.

In order to enable large-scale simulations that can meet the scientific objectives outlined for this application, it is essential to define a scalable workflow. Very

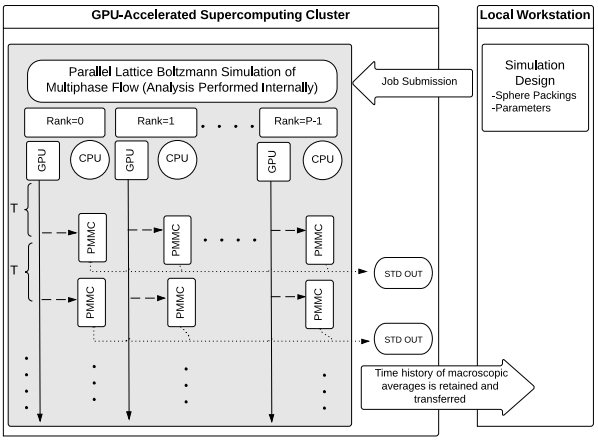

Figure 3. The overall design of the simulation tool considered in this work relies on GPU to execute a simulation for the multiphase flow while the CPU perform analysis at regular intervals throughout the simulation. As a consequence, the interval for analysis T can be chosen as a small value, yielding detailed information for the system dynamics.

large systems must be considered in order to produce generalizable results from pore-scale studies. However, it is cumbersome to retain and analyze the resulting data. This bottleneck is evident based on the workflow shown in Fig. 2. In this example, a CPU-based implementation of the LBM is used to provide the solution for the phase indicator field $\phi(\boldsymbol{x_i}, t)$, the pressure field $p(\boldsymbol{x_i}, t)$ and the velocity field $\mathbf{u}(\boldsymbol{x_i}, t)$. The simulation is executed by P MPI processes, each of which owns a subdomain of size $N \times N \times N$. In order to extract averaged information from the simulation, the solution was saved to disk every T time steps and transferred to a local workstation where the PMMC approach is applied to determine the interfacial areas.

In order to track the dynamics of the system, the simulation state is analyzed at intervals of T time steps. For the present study tracking the interfacial areas, it is sufficient to save only the phase indicator field to disk, although this is not true in general. If the analysis is to be performed as shown in Fig. 2, $8PN^3$ bytes will be written to disk and transferred off of the cluster for post-processing. The interval T is constrained by the extreme costs of data transfer, in particular. Furthermore, the total system size PN^3 is constrained by the amount of memory available on the desktop machine and the time required to execute the PMMC analysis does not scale with the number of processors P. As a consequence, the interval T becomes unacceptably large as the P increases. Note that in the present study we are interested in computing only four quantities: a_{wn}, a_{ws}, a_{ns} and ΔE_s.

In a GPU-based simulation, it is attractive to use the idle CPU cores to perform analysis so I/O and data transfer costs can be reduced. The strategy outlined in Fig. 3 provides a way to avoid data transfers and I/O while taking advantage of the available CPU. At intervals of T time steps, the requisite data is transferred from the GPU to the CPU using `cudaMemcpy`. Each MPI process then applies the PMMC to extract the local interfacial areas on the CPU, which are reduced to obtain a_{wn}, a_{ws} and a_{ns} using `MPI_Reduce`. Only these values are written to output, effectively eliminating the cost of data transfer since the size of the output data is no longer proportional to PN^3. The time interval T can be chosen such that the costs of carrying out the averaging have minimal impact on the performance of the LBM simulator itself. A typical multiphase LBM calculation for multiphase flow in porous media requires $10^5 - 10^7$ time steps. In this work we demonstrate that excellent performance can be sustained while performing averaging every $T = 10^3$ time steps such that dynamic information multiphase flow processes can be collected hundreds to thousands of times per simulation.

B. Multiphase Lattice Boltzmann Scheme

A variety of schemes have been constructed to model multiphase flows using the lattice Bolztmann method. The 'color model', originally proposed by Gunstensen, has since been extended to describe flows in porous media [7]. For the LBM, a set of discrete velocities provide the mechanism to track the transport behavior. The momentum transport solution relies on the D3Q19 discrete velocity set:

$$\begin{cases} \{0,0,0\}^T & \text{for } q = 0 \\ \{\pm 1,0,0\}^T, & \text{for } q = 1,2 \\ \{0,\pm 1,0\}^T, & \text{for } q = 3,4 \\ \{0,0,\pm 1\}^T & \text{for } q = 5,6 \\ \{\pm 1,\pm 1,0\}^T, & \text{for } q = 7,8,9,10 \\ \{\pm 1,0,\pm 1\}^T, & \text{for } q = 11,12,13,14 \\ \{0,\pm 1,\pm 1\}^T & \text{for } q = 15,16,17,18. \end{cases} \quad (2)$$

A mass transport solution can be performed using the simpler D3Q7 velocity set, which corresponds to $q = 0, 1, 2, \ldots, 6$ in Eq. 2. Two sets of distributions are instantiated locally from the density values ρ^w and ρ^n and the flow velocity $\mathbf{u}$. The mass transport distributions are constructed according to:

$$g_q^w = w_q \left[\rho^w \left(1 + \boldsymbol{\xi}_q \cdot \mathbf{u}\right) + \beta \frac{\rho^w \rho^n}{\rho^w + \rho^n} \left(\mathbf{n} \cdot \boldsymbol{\xi}_q\right) \right] \quad (3)$$

$$g_q^n = w_q \left[\rho^n \left(1 + \boldsymbol{\xi}_q \cdot \mathbf{u}\right) - \beta \frac{\rho^w \rho^n}{\rho^w + \rho^n} \left(\mathbf{n} \cdot \boldsymbol{\xi}_q\right) \right], \quad (4)$$

where $q = 0, 1, \ldots, 6$. The vector $\mathbf{n}$ is the unit normal to the interface, determined from phase indicator field:

$$\phi = \frac{\rho^w - \rho^n}{\rho^w + \rho^n}. \tag{5}$$

The gradient of the phase indicator field is computed according to

$$\mathbf{C}(\boldsymbol{x}_i, t) = \nabla\phi = \sum_{q=1}^{Q-1} \phi(\boldsymbol{x}_i + \boldsymbol{\xi}_q, t)\boldsymbol{\xi}_q, \tag{6}$$

and the unit normal vector is :

$$\mathbf{n} = \frac{\nabla\phi}{|\nabla\phi|}. \tag{7}$$

For the D3Q7 model, the weights are $w_0 = 1/3$ and $w_q = 1/9$ for $q = 1, 2, \ldots 6$. Eqs. 3 and 4 enforce immiscibility of the two phases while satisfying local mass and momentum conservation. Once the distributions have been computed locally, they are "pushed" to neighboring lattice sites to update the densities used in the next timestep:

$$\rho^k(\boldsymbol{x}_i, t + \delta t) = \sum_{q=0}^{6} g_q^k(\boldsymbol{x}_i - \boldsymbol{\xi}_q\delta t, t). \tag{8}$$

where $k = w, n$. This approach is distinct from other LBMs because the distributions are not stored in memory. This has implications for parallel implementation because the ρ^k must be updated atomically for multi-threaded computation of Eq. 8.

To solve for the momentum transport, a set of distributions f_q are constructed to track the fluid properties associated with each discrete velocity $\boldsymbol{\xi}_q$ appearing in Eq. 2. The distributions evolve according to the lattice Boltzmann equation (LBE):

$$f_q(\boldsymbol{x}_i + \boldsymbol{\xi}_q\delta t, t + \delta t) - f_q(\boldsymbol{x}_i, t) = \mathcal{J}_q(\boldsymbol{x}_i, t), \tag{9}$$

for $q = 0, 1, \ldots 18$. The distributions can be used to compute the fluid pressure:

$$p = \frac{1}{3}\sum_{q=0}^{18} f_q, \tag{10}$$

and the fluid momentum:

$$\mathbf{j} = \rho\mathbf{u} = \sum_{q=0}^{18} f_q\boldsymbol{\xi}_q. \tag{11}$$

The term $\mathcal{J}_q$ accounts for changes to f_q that result from inter-molecular collisions and interaction forces. The form of $\mathcal{J}_q$ for our multiphase implementation matches what is presented by McClure et al., where additional details are available [15]. Most typically, solution of Eq. 9 is performed in two steps, known as the collision step:

$$f_q^*(\boldsymbol{x}_i, t) = \mathcal{J}_q(\boldsymbol{x}_i, t), \tag{12}$$

and the streaming step:

$$f_q(\boldsymbol{x}_i + \boldsymbol{\xi}_q\delta t, t + \delta t) = f_q^*(\boldsymbol{x}_i, t). \tag{13}$$

The collision step is the more computationally intensive aspect of the LBM, whereas the streaming step relies entirely on memory bandwidth. For flow in porous media it is important to rely on multi-relaxation time (MRT) approaches that are both more accurate and more computationally intensive compared with the single-relaxation time approaches that are frequently encountered in performance studies. In the MRT approach the form of the collision operator is constructed from linear combinations of the distributions [18, 19]:

$$\hat{f}_m = \sum_{q=0}^{Q-1} M_{m,q} f_q \tag{14}$$

and the collision term is defined as:

$$\mathcal{J}_q = \sum_{m=0}^{Q-1} M_{q,m}^* \lambda_m \left(\hat{f}_m^{eq} - \hat{f}_m\right). \tag{15}$$

The transformation matrix $M_{m,q}$ and its inverse $M_{q,m}^*$ are defined for the D3Q19 velocity structure by d'Humiëres and Ginzburg [20]. Each moment relaxes toward its equilibrium value $\hat{f}_m^{eq}$ at a unique rate specified by λ_m. In the formulation proposed by Arhenholz et al. the non-zero equilibrium moments incorporate additional terms to accommodate an anisotropic contribution to the stress tensor due to the surface tension [7]:

$$\hat{f}_1^{eq} = \left(j_x^2 + j_y^2 + j_z^2\right) + \sigma|\mathbf{C}| \tag{16}$$

$$\hat{f}_9^{eq} = \left(2j_x^2 - j_y^2 - j_z^2\right) + \sigma\frac{|\mathbf{C}|}{2}\left(2n_x^2 - n_y^2 - n_z^2\right) \tag{17}$$

$$\hat{f}_{11}^{eq} = \left(j_y^2 - j_z^2\right) + \sigma\frac{|\mathbf{C}|}{2}\left(n_y^2 - n_z^2\right) \tag{18}$$

$$\hat{f}_{13}^{eq} = j_x j_y + \sigma\frac{|\mathbf{C}|}{2} n_x n_y \tag{19}$$

$$\hat{f}_{14}^{eq} = j_y j_z + \sigma\frac{|\mathbf{C}|}{2} n_y n_z \tag{20}$$

$$\hat{f}_{15}^{eq} = j_x j_x + \sigma\frac{|\mathbf{C}|}{2} n_x n_z \tag{21}$$

where the parameter σ is linearly related to the interfacial tension and the relaxation parameters λ_m relate to the fluid kinematic viscosity.

For flow in porous media the distributions can be ignored at all lattice sites that satisfy $\boldsymbol{x}_i \in \Omega_s$, where Ω_s is the part of domain occupied by the solid. A "bounce-back" rule is applied to set a no-slip boundary condition for the fluid phase at the solid wall: the most commonly used rule for setting solid boundary conditions:

$$f_q(\boldsymbol{x}_i, t + \delta t) = f_{q+1}(\boldsymbol{x}_i, t) \text{ if } q \text{ is odd}, \tag{22}$$

$$f_q(\boldsymbol{x}_i, t+\delta t) \quad = \quad f_{q-1}(\boldsymbol{x}_i, t) \text{ if } q \text{ is even.} \qquad (23)$$

An analogous expression is applied for the mass transport distributions g_q^w and g_q^n.

C. GPU Implementation of the LBM

Implementation of the LBM on a GPU represents a considerably different task from CPU implementation. Each GPU contains a large number of relatively simplistic cores that are capable of carrying out tasks in parallel. The CPU and GPU have their own dedicated memory spaces, and initialized variables are copied to the GPU where the main computations are performed. At each interval T data is copied back to the CPU so that analysis can be performed. CUDA kernels are based on a SIMD model in which many threads execute identical instructions simultaneously. Domain decomposition on a GPU is structured so as to take advantage of the multi-threaded framework. The number of threads and the number of threadblocks can be varied to maximize performance. Kernels provide instructions to the GPU that account for all computations required to complete each time step of the LBM. A set of registers are created to store the nineteen distributions required to perform computations at each lattice site. These values reside in fast register memory once they have been accessed from the main arrays. Computations are expressed in terms of these registers in order to maximize performance.

The memory system of the GPU functions most efficiently when the values read by each threadblock are aligned within contiguous blocks of memory. When data is properly aligned, the GPU is able to coalesce memory accesses within each threadblock into a smaller number of memory transactions. In order to use memory bandwidth most efficiently, each half-warp of sixteen threads should access contiguous memory blocks of size 32, 64, or 128 bytes. As a consequence, the layout of the distributions $f_0, f_1, \ldots f_{18}$ must accommodate these memory accesses. A full description of how the GPU solution of Eqs. 9 – 21 is performed is available from McClure et al [15]. In this work we also implement the solution of Eqs. 3 – 8 equations on GPU. This calculation maps to the GPU in a straightforward way, although it is different from the solution of Eq. 9 in that atomic operations are required to determine Eq. 8. Double precision `atomicAdd` was therefore implemented as suggested in CUDA programming guide [21].

D. Distributed Memory Implementation of the LBM using GPU

In order to obtain a distributed memory solution for the multiphase flow problem, cubic sub-domains $\Omega^{(p)}$ are associated with each MPI process $p = 0, 1, \ldots P-1$. The sub-domains are of equal size with $N \times N \times N$ lattice sites. The position of the solid phase is initialized

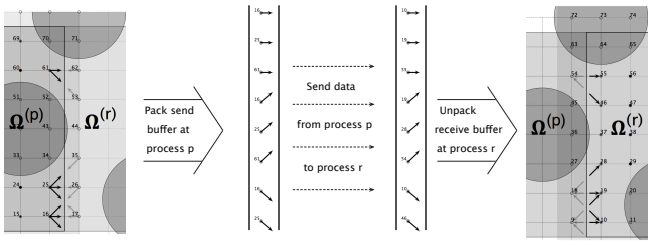

Figure 4. Graphical depiction of the MPI communication pattern along a boundary for the LBM. Communicated distributions are packed into contiguous send buffers on the GPU, ignoring lattice sites at any $\boldsymbol{x}_i \in \Omega_s$. The data is then sent to the neighboring process and unpacked to the appropriate location on the GPU.

using information provided from a sphere packing. The list of spheres is read by the MPI process with $p = 0$, then broadcast to all other processes. The sub-region of the sphere packing assigned to process p is determined from a cartesian process grid $p = kP_xP_y + jP_x + i$, where (i, j, k) provides the location of the subdomain associated with process p. The full domain therefore corresponds to a cartesian grid with $P = P_x \times P_y \times P_z$ sub-domains, where P_x, P_y and P_z are the number of sub-domains in each direction. A solution for the multiphase flow problem is obtained by concurrently solving the LBM within each sub-domain and using MPI to exchange information at the sub-domain boundaries. Each MPI process requires enough memory to store the sub-domain and a halo of width one that is used to facilitate communication.

Based on the straightforward domain decomposition strategy employed in our approach, each subdomain communicates with immediately adjacent sub-domains only. Communication is required to execute the streaming step for the D3Q19 momentum transport solution given in Eq. 13, for the D3Q7 mass transport solution given in Eq. 8, and to fill in the values of ϕ within the halo so that the color gradient can be computed in Eq. 6. To complete the streaming step at lattice site $\boldsymbol{x}_i$, processor r must provide to processor p all distributions $f_q^\alpha(\boldsymbol{x}_i)$ that satisfy $\boldsymbol{x}_i - \boldsymbol{\xi}_q \in \Omega^{(r)}, \boldsymbol{x}_i - \boldsymbol{\xi}_q \notin \Omega_s$. The symmetry of the discrete velocity set ensures that each value received by processor p from processor r will mirror a value sent by p to r. Each process p must therefore communicate with 18 other processes to solve its part of the problem. These include those process that share one of the six faces: $(i+1, j, k), (i-1, j, k), (i, j+1, k), (i, j-1, k), (i, j, k+1), (i, j, k-1)$ and the processes which share one of the twelve edges: $(i+1, j+1, k), (i-1, j-1, k), (i+1, j-1, k), (i-1, j+1, k), (i+1, j, k+1), (i+1, j, k+1), (i-1, j, k-1), (i+1, j, k-1), (i-1, j, k+1), (i, j+1, k+1), (i, j-1, k-1), (i, j+1, k-1), (i, j-1, k+1)$. Full periodic boundary conditions are imposed such that

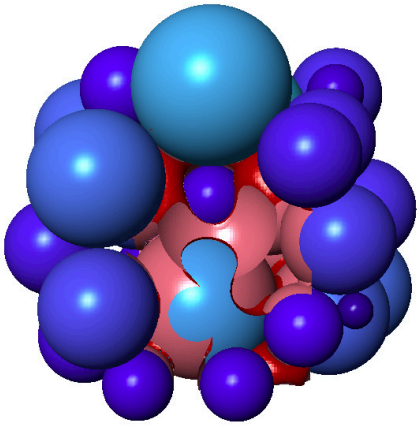

Figure 5. Closeup view of a multiphase interfacial configuration in a sub-region of a sphere packing. The wn interface Ω_{wn} (red) and the ns interface Ω_{ns} (pink) are also shown, constructed using the PMMC algorithm. The solid spheres are also depicted.

all neighboring processes are determined.

Wang et. al described in detail the steps required to implement GPU-based communications for a similar D3Q7/D3Q19 multiphase flow formulation [11]. A significant difference in our implementation is that the presence of an immobile solid phase significantly reduces the number of communications that must be performed for flows in porous media. The region of space occupied by the solid phase Ω_s typically represents 60-70% of the total volume. Since communications never need to be performed for any lattice sites $\boldsymbol{x}_i \in \Omega_s$, it is possible to significantly reduce the size of the messages passed. This communication procedure is summarized by Fig. 4. The requisite communications are identified by first constructing a list of the lattice sites $\boldsymbol{x}_i$ on each boundary for which $\boldsymbol{x}_i \notin \Omega_s$. This list is then communicated to the adjacent receiving process, and appropriately sized send and receive buffers are allocated by each process within GPU memory. Send and receive buffers are allocated for each pair of communicating processes. Pack and unpack kernels are then constructed so that communicated values are placed contiguously within the send and receive buffers.

E. Porous Media Marching Cubes Algorithm (PMMC)

In the LBM, the interfacial regions tend to be diffuse due to the way interactions are defined between the fluid components. In order to evaluate the various interfacial areas, the interfaces must first be extracted based on the values of the phase indicator field. The marching cubes algorithm provides a straightforward way to construct interfaces as iso-contours of a three-dimensional function defined on a lattice. In this algorithm, lists of triangles are constructed to represent the corresponding surfaces using linear interpolation [22]. The surfaces obtained from the marching cubes algorithm are widely used in visualization; in our case the triangles are used as the a framework for numerical methods.

We used a previously developed variant of the marching cubes algorithm that is designed specifically for the case of two-phase flow in porous media. The objective is to explicitly construct a numerical representation for all three interfaces in a two-fluid phase system: Ω_{wn}, Ω_{ws} and Ω_{ns}. A typical multiphase interfacial configuration, including the Ω_{wn} and Ω_{ns} interfaces, is shown in Fig. 5 with the solid phase spheres. To construct these surfaces, the PMMC algorithm requires: (1) the distance to the solid surface evaluated at each lattice site and (2) the values of the phase indicator field. The method is described in detail by McClure et al. [17]. The basic steps are as outlined follows:

1) construct the solid surface using the marching cubes algorithm;
2) using linear interpolation, evaluate the phase indicator field at the vertices of each triangle on the solid surface;
3) again using linear interpolation, identify points on the contact curve where all three phases coexist;
4) trim the solid surface at the contact line to form the list of triangles that approximate the interfaces Ω_{ws} and Ω_{ns};
5) Using vertices obtained from marching cubes algorithm and vertices on the common curve, construct a list of triangles that approximate the Ω_{wn} interface within each cube.

In order to visualize the surface, all of the triangles must be retained after they have been computed. However, visualization is not an objective of the present study; we simply wish to evaluate the component interfacial areas. As a result, triangle lists can be constructed locally within each cube and then discarded once the local contribution to a_{wn}, a_{ws} and a_{ns} has been determined. Since the full list of triangles is not retained, the memory required by this approach is minimal. More importantly, it avoids the possibility that dynamic memory allocations will be necessary. The total number of triangles can change significantly as flow processes evolve, both globally and within an individual process. However, since the maximum number of possible triangles within a cube can be determined easily, this approach minimizes the memory footprint for the CPU while guaranteeing safe execution.

As the PMMC algorithm is embarrassingly parallel, the steps required to integrate this analysis into a parallel LB simulator are straightforward. The distance to

the solid surface is evaluated at the beginning of the simulation and stored in CPU memory. Since the solid is immobile, these values do not change and do not need to be transferred to the GPU at any point during the simulation. Information about the phase position is provided by the phase indicator field ϕ, which evolves on the GPU based on the multiphase flow dynamics. As a result, this data needs to be transferred from the GPU to the CPU each time that the PMMC algorithm is applied.

For more generic interfacial averages such as surface averaged curvatures, velocities and orientations, local construction of the triangle lists remains the most judicious approach. In the general case, a larger amount of data must be transferred from the GPU to the CPU, including the pressure and velocity fields. Both the number of averaged quantities and the complexity of the averages computed impact the computational cost associated with the CPU-based analysis. Since the amount of computation depends on the amount of interfacial area present, it is difficult to establish a concrete estimate for the associated computational costs. However, our objective is to enable a simulation for which T is sufficiently small to resolve the changes in the macroscopic behavior.

IV. Results and Discussion

A. Parallel Performance of the LBM

Parallel performance of the multiphase LBM simulator was evaluated using Titan. Two sphere packings were generated to serve as the domain for these calculations: a packing of $11,402$ spheres with lognormal mean $\mu = -2.23$ and variance $\sigma = 0.1$, and a packing of $85,539$ spheres with lognormal mean $\mu = -2.24$ and variance $\sigma = 0.1$. The volume fraction occupied by Ω_s was 0.631 for each packing. Strong scaling results were obtained using these random close packings discretized to obtain lattice sizes of 960^3 and 1920^3, respectively. Details for these simulations including the parallel efficiency are provided in Table I, and a scaling plot is shown in Fig. 6. The ideal scaling line is based on a simulation performed in a smaller packing consisting of $1,896$ spheres discretized to 540^3 and simulated using a $3\times3\times3$ grid of processes. A performance of 76 MLUPS were obtained for each node in this simulation. Based on the initial points in our strong scaling plots, the weak scaling for this code is very nearly ideal.

B. Evaluation of Non-Equilibrium Interfacial Areas

In this section, we demonstrate the practical results that are obtained by combining the GPU-based implementation of the multiphase LBM coupled to with the CPU-based multi-scale analysis tools described in the previous sections. We consider the simulation of a multiphase flow process in a packing of 11,402 spheres,

Table I
Strong Scaling for the Multiphase LBM

# Spheres	Process Grid	N^3	MLUPS	Efficiency
11,402	$5\times5\times5$	192^3	9,407	0.98
11,402	$8\times8\times8$	160^3	15,130	0.91
11,402	$8\times8\times8$	128^3	31,275	0.80
11,402	$10\times10\times10$	120^3	50,907	0.67
85,539	$10\times10\times10$	192^3	74,338	0.97
85,539	$15\times15\times15$	128^3	199,692	0.77
85,539	$16\times16\times16$	120^3	244,754	0.78

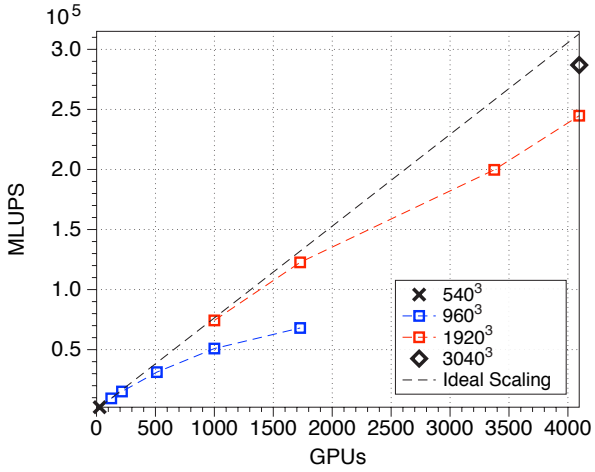

Figure 6. Scaling results obtained for several domain sizes on Titan, including strong scaling for both a 960^3 lattice and a 1920^3 lattice. Ideal scaling is shown based on a 540^3 simulation performed on 27 GPU. Results are reported in million-lattice-node updates per second (MLUPS).

discretized to obtain a lattice of 1200^3 and distributed across $P = 1000$ MPI processes on Titan. The PMMC approach was applied to extract multi-scale averages (in this case the interfacial areas) at an interval of $T = 1000$ time steps. The simulation was initialized to correspond with a non-equilibrium multiphase configuration corresponding to a the non-wetting phase saturation equal to 0.4. The system was then allowed to relax toward equilibrium for a wall time equal to two hours. The resulting plots for the interfacial areas Fig. 7 (a) - (c). The change in surface energy is shown for each interval T in Fig. 7 (d), which provides a measure of the proximity to the ultimate equilibrium state.

The non-equilibrium interfacial area results obtained using this approach are quite well-resolved. The relaxation process is plotted for a_{wn}, a_{ws} and a_{ns} in Fig. 7 (a) (b) and (c). Each value was computed 163 times over the course of $163,000$ time steps. The total size of the output from this simulation was 20kB. In order to produce the same results using the workflow shown in Fig. 2, the required size of the output files can be computed in a

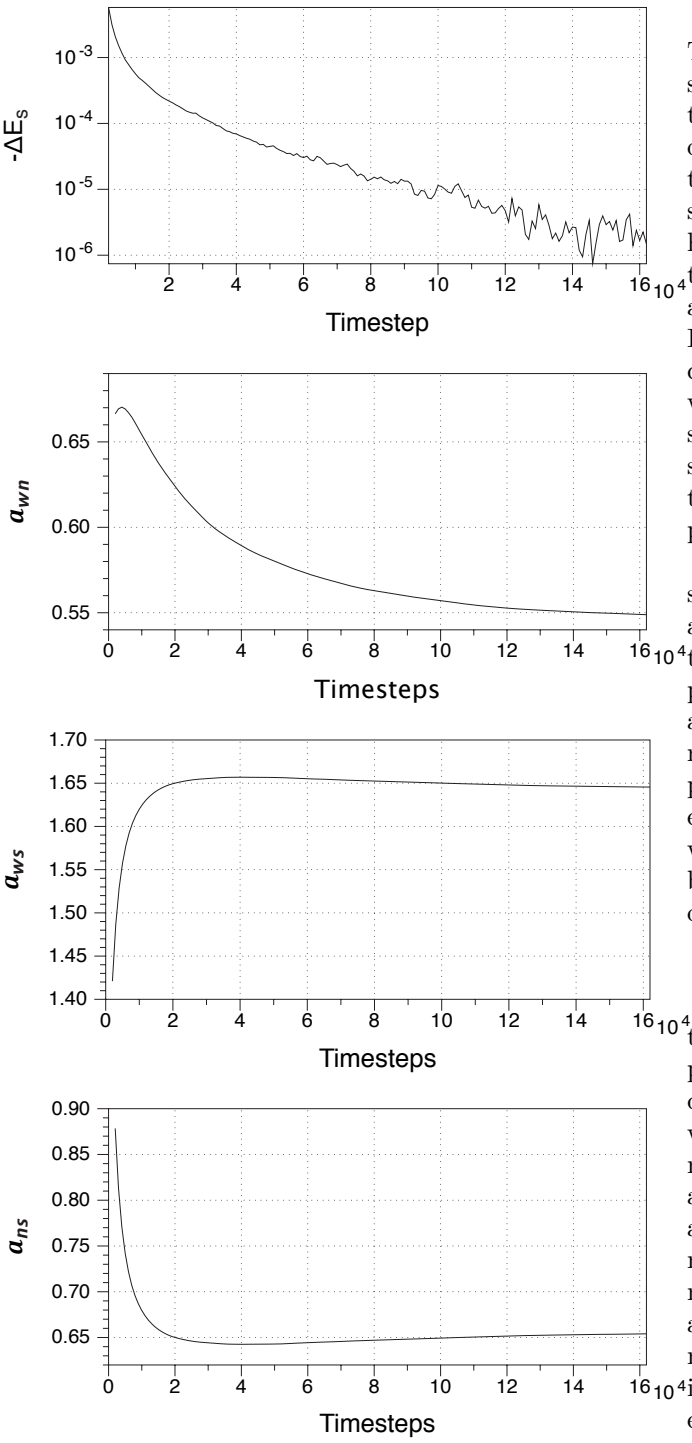

Figure 7. Interfacial area and surface energy measurements conducted during a multiphase flow simulation within a packing $11,402$ spheres on a lattice size of 1200^3. Area measurements were calculated every 10^3 time steps and the duration of the simulation was $163,000$ time steps.

straightforward manner:

$$163 \times PN^3 \times 8\text{B} = 2.25\text{TB}. \tag{24}$$

The costs of transferring this data alone represent a significant bottleneck that constrains the spatial resolution T that multi-scale information can be accessed. Not only does this represent the first opportunity to study these dynamics in detail, but the result is obtained for a system that is an order of magnitude larger than what has been considered in previous studies. Also note that this calculation is for a two-hour simulation in which a single multiphase flow configuration was considered. In order to perform a complete study it is desirable to consider hundreds of configurations corresponding to a wide range of fluid saturations. Without an integrated strategy to extract multi-scale information from the simulation as it proceeds, many of the opportunities to advance the state of scientific understanding for this problem would be inaccessible.

The capability to extract multi-scale information from simulations of this nature has immediate implications and will enable a diverse set of computational studies that were previously impossible. Averaged interfacial properties and their time derivatives can be computed accurately and efficiently. As a consequence, a wide range of constitutive relationships that have been proposed based on theoretical work can now be studied and evaluated within a computational setting. Ongoing work will expand the range of multi-scale information that can be explored within this approach and pursue the closure of new models for multiphase flow in porous media.

V. Conclusions

A scalable approach was constructed to simulate multiphase flow in porous medium systems. The approach provides a viable means to extract interfacial areas during the execution of multiphase flow simulations, which can be scaled across thousands GPU-accelerated nodes. The CPU cores are used to extract interfacial areas in parallel during the course of the simulation, an approach which enables high resolution studies of the non-equilibrium dynamics of multiphase flow in porous media. The practical reduction in I/O, data transfer and storage requirements is approximately nine orders of magnitude. The peak performance for the LBM reported in this paper was $244,754$ MLUPS on $4,096$ GPU-equipped nodes on the Titan supercomputer.

Acknowledgment

This work was supported in part by National Science Foundation grant 0941235, Department of Energy grant DE-SC0002163x, the GPU-accelerated HokieSpeed supercomputer via NSF CNS-0960081 from the NSF MRI-R2 program and used resources of the Oak Ridge Lead-

ership Computing Facility at oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725

References

[1] W. G. Gray and C. T. Miller, "Thermodynamically constrained averaging theory approach for modeling flow and transport phenomena in porous medium systems: 8. Interface and common curve dynamics," *Advances in Water Resources*, vol. 33, no. 12, pp. 1427–1443, DEC 2010.

[2] ——, "TCAT analysis of capillary pressure in non-equilibrium, two-fluid-phase, porous medium systems," *Advances in Water Resources*, vol. 34, no. 6, pp. 770–778, 2011.

[3] V. Joekar-Niasar and S. M. Hassanizadeh, "Specific interfacial area: The missing state variable in two-phase flow equations?" *Water Resources Research*, vol. 47, 2011.

[4] C. Pan, J. F. Prins, and C. T. Miller, "A high-performance lattice Boltzmann implementation to model flow in porous media," *Computer Physics Communication*, vol. 158, no. 2, pp. 89–105, 2004.

[5] J. Wang, X. Zhang, A. Bengough, and J. Crawford, "Domain-decomposition method for parallel lattice Boltzmann simulation of incompressible flow in porous media," *Physical Review E*, vol. 72, no. 1, Part 2, JUL 2005.

[6] M. G. Schaap, M. L. Porter, B. S. B. Christensen, and D. Wildenschild, "Comparison of pressure-saturation characteristics derived from computed tomography and lattice Boltzmann simulations," *Water Resources Research*, vol. 43, 2007.

[7] B. Ahrenholz, J. Toelke, P. Lehmann, A. Peters, A. Kaestner, M. Krafczyk, and W. Durner, "Prediction of capillary hysteresis in a porous material using lattice-Boltzmann methods and comparison to experimental data and a morphological pore network model," *Advances in Water Resources*, vol. 31, no. 9, pp. 1151–1173, SEP 2008.

[8] M. Bernaschi, M. Bisson, T. Endo, S. Matsuoka, M. Fatica, and S. Melchionna, "Petaflop biofluidics simulations on a two million-core system," in *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*, ser. SC '11. New York, NY, USA: ACM, 2011, pp. 4:1–4:12. [Online]. Available: http://doi.acm.org/10.1145/2063384.2063389

[9] J. Myre, S. D. C. Walsh, D. Lilja, and M. O. Saar, "Performance analysis of single-phase, multiphase, and multicomponent lattice-Boltzmann fluid flow simulations on GPU clusters," *Concurrency and Computation-Practice & Experience*, vol. 23, no. 4, pp. 332–350, MAR 2011.

[10] C. Obrecht, F. Kuznik, B. Tourancheau, and J.-J. Roux, "Scalable lattice Boltzmann solvers for CUDA GPU clusters," *Parallel Computing*, vol. 39, no. 6-7, pp. 259–270, JUN-JUL 2013.

[11] H. Wang, S. Potluri, D. Bureddy, C. Rosales, and D. K. Panda, "Gpu-aware mpi on rdma-enabled clusters: Design, implementation and evaluation," *IEEE Transactions on Parallel and Distributed Systems*, vol. 99, no. PrePrints, p. 1, 2013.

[12] A. L. Dye, J. E. McClure, C. T. Miller, and W. G. Gray, "Description of non-Darcy flows in porous medium systems," *Physical Review E*, vol. 87, no. 3, MAR 18 2013.

[13] M. L. Porter, M. G. Schaap, and D. Wildenschild, "Comparison of interfacial area estimates for multiphase flow through porous media using computed microtomography and lattice-Boltzmann simulations," *Water Resources Research*, vol. to be submitted, 2009.

[14] A. W. Adamson, *Physical Chemistry of Surfaces*. New York: John Wiley & Sons, 1990.

[15] J. E. McClure, J. F. Prins, and C. T. Miller, "A novel heterogeneous algorithm to simulate multiphase flow in porous media on multicore cpu-gpu systems," *Computer Physics Communications*, vol. Under Review, 2013.

[16] TOP500 Supercomputing Sites, "TOP500 Supercomputing List," http://www.top500.org/.

[17] J. E. McClure, D. Adalsteinsson, C. Pan, W. G. Gray, and C. T. Miller, "Approximation of interfacial properties in multiphase porous medium systems," *Advances in Water Resources*, vol. 30, no. 3, pp. 354–365, 2007.

[18] C. Pan, L.-S. Luo, and C. T. Miller, "An evaluation of lattice Boltzmann schemes for porous medium flow simulation," *Computers & Fluids*, vol. 35, no. 8–9, pp. 898–909, 2006.

[19] P. Lallemand and L. S. Luo, "Theory of the lattice Boltzmann method: Dispersion, dissipation, isotropy, Galilean invariance, and stability," *Physical Review E*, vol. 61, no. 6, pp. 6546–6562, 2000.

[20] D. d'Humiéres, I. Ginzburg, M. Krafczyk, P. Lallemand, and L. S. Luo, "Multiple-relaxation-time lattice Boltzmann models in three dimensions," *Philosophical Transactions of the Royal Society of London Series A-Mathematical Physical and Engineering Sciences*, vol. 360, pp. 437–451, 2002.

[21] *NVidia CUDA Programming Guide, Version 5.0*, NVIDIA, 2012.

[22] W. E. Lorenson and H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics*, vol. 21, no. 4, pp. 163–

169, 1987.