Available online at www.sciencedirect.com

ScienceDirect

Procedia Technology 11 (2013) 1110 – 1116

Procedia Technology

The 4th International Conference on Electrical Engineering and Informatics (ICEEI 2013)

Implementation Strategy for D2Q9 Model on Desktop Grid Environment

Mahathir Rahmany[a,] *, Elankovan Sundararajan[a], Abdullah Mohd Zin[a], Shahrir Abdullah[b]

[a]*Faculty of Information Science & Technology, National University of Malaysia, Bangi, Selangor 43600, Malaysia*
[b]*Faculty of Engineering & Built Environment, National University of Malaysia, Bangi, Selangor 43600, Malaysia*

Abstract

Lattice Boltzmann Method (LBM) is a widely used method to simulate the computation of fluid dynamics. Its cost effectiveness and computing speed are some of the important reasons why it has become very popular in solving Computational Fluid Dynamics (CFD) applications. However, this method is not practical to be applied for large CFD problems in a single machine because it requires high memory usage and processing power. Desktop Grid Computing (DGC) is one of the possible choices among other methods such as Massively Parallel Processors (MPP), Symmetrical Multi Processors (SMP), multicores and GP-GPU in overcoming the above problem. The main idea in DGC is scavenging unused resources for compute intensive tasks. In this work, we develop a new strategy for executing the two dimensional and nine velocity (D2Q9) LBM model by using HTCondor as DGC middleware. In this strategy, we reduced the communication overhead by sending only the data required by other machines and leave the rest of the data in the compute node. This strategy also allows the HTCondor server to remember the machines with a particular portion of the data so that appropriate data can be sent for further computation until convergence criterion is achieved.

Keywords: Lattice Boltzman Method; D2Q9; desktop grid computing; Cavity Flow;

* Corresponding author. Tel.:+60-169-787-004.
E-mail address: mahathir.rahmany@gmail.com

doi:10.1016/j.protcy.2013.12.301

1. Introduction

In general, Computational Fluid Dynamics (CFD) is a computing technique to predict fluid flow and its effects on structures in the flow and interactions with boundary surfaces in details using mathematical and computational models [1]. Some advantages of this model is that it is economical and faster; provides a complete set of results, even in circumstances where experimental measurements would be difficult; allows investigating situations for which experiments would be impossible [2].

Lattice Boltzmann Method (LBM) has become a common numerical technique in Computational Fluid Dynamics (CFD) for simulations [3]. Flow through porous media, suspensions in fluids, Multi-component and Interfacial Dynamics, Direct Numerical Simulations (DNS) of Turbulence, Large Eddy Simulations (LES) and Non-Newtonian Fluids are some of the simulation model that utilizes this method [4]. It is also to note that the first proposal to simulate fluid flow problem using the Lattice Boltzmann equation was made 21 years ago [5].

Lattice Boltzmann Method (LBM) is the numerical discretization of the Boltzmann equation with collisional model [6] that is derived from discrete kinetic theory [7]. Many researches have been working on this method for almost 2 decades, and now Lattice Boltzmann Method has become a prominent choice to study fluid dynamics [4].

To perform millions of calculation in Computational Fluid Dynamics (CFD) with Lattice Boltzmann Method, computers need to work harder even though only approximate solutions can be achieved in many cases [8]. Single CPU machines with limited memory cannot perform the calculations fast enough. One way to overcome this problem is by exploiting parallel computing. Essentially it is the use multiple CPUs to perform parallel computation to decrease processing time [9]. Some other well-known approaches are by harnessing GP-GPU (General-purpose computing on graphics processing units) and Multicore.

Another way of overcoming the problem is by using DGC and the basic concept of this approach is to utilize idle computers. Therefore, it can be said that the resources for DGC is unlimited. DGC evolved from Distributed Computing. Distributed computing is the term used to utilize CPU cycles and storage space of many networked system to work together on problems that are computationally intensive [10]. The aim of DGC is to have a pool of very large computing networks for tackling large compute intensive problems very quickly [11] by harnessing the idle CPU cycles [12] from the people who has computer and connect it into internet all over the world. The idea of DG is that people lend idle cycle of their computer willingly.

A middleware is used on DGC to ease the deployment of compute intensive problems. Two of the popular middleware for DGC are BOINC and HTCondor. BOINC is a non-commercial and open source system that used the computing power and storage capacity when the idle cycle of thousands or millions of PCs that is connected internet to solve simple or complicated calculation problem [13]. HTCondor or known as just Condor before this is also an open source system and is the first middleware to introduce the concept of utilizing idle resource [21]. Some of the reason why it become so well-known in DGC are (a) user friendly, (b) available for windows and Linux, (c) support MPI library and Java programs, (d) easy to manage, and (e) widely used around the world. This is among the reasons HTCondor was used in this research.

In this article, we report the progress of our research on executing the LBM computation of two dimensions and nine directions (D2Q9) model by using HTCondor as DGC middleware. In Section 2, we review some related previous researches that utilizes LBM on distributed environment. Section 3 describes the overview of proposing methodology in order to do so. By regarding the methodology in section 3, we posit the most likely strategy to be implemented thus the execution process is running well in section 4. Finally the conclusion is in section 5.

2. Related Work

In this section, we elaborate some of the previous researches related to executing LBM under distributed computing. Many works on LBM in Distributed Environment have been carried out in the past. LBM has been implemented on cluster-computers, supercomputers, multi-core, multicore-platform PlayStations. Other works have concentrated on improving the code efficiency such as optimizing parallel code to improve cache performance. We have summarized some of the relevant past works in Table 1.

Table 1. Previous research on executing LBM under Distributed Environment

Reference	Objective	Result
Jens Wilke et al [14]	• Optimize cache performance for Parallel Lattice Boltzmann Method Codes in 2D. • To indicate the importance of the single CPU performance before using parallel computing method in the framework of a Computational Fluid Dynamics (CFD) Code based on the Lattice Boltzmann Method (LBM).	• By exploiting the benefits of hierarchical memory architectures of current CPUs, they ware have obtained factors of 2-3 in performance on various machines. • Unfortunately, the parallelization of cache-optimized codes is commonly tedious and error-prone due to their implementation complexity
Thomas Pohl et al [15]	• Optimize and profile the Cache Performance of Parallel Lattice Boltzmann Codes in 2D and 3D. • To demonstrate the importance of considering the single-CPU performance before using parallel computing methods in the framework of a Computational Fluid Dynamics (CFD) code based on the Lattice Boltzmann Method (LBM) in 2D and in 3D	• By applying optimized code transformations and thus exploiting the benefits of hierarchical memory architectures of current CPUs, the have obtained speedup factors of up to more than 3 on various machines.
Thomas Pohl et al [16].	• Evaluate performance of parallel large-scale Lattice Boltzmann Applications on three supercomputing architectures.	• Able to prove that these optimizations lead to excellent scale-up and speed-up behavior up to 512 processors on the Hitachi SR 8000. • Only dedicated supercomputers featuring network and memory connections with extremely high bandwidth and low latency are suitable for these kinds of codes with large data sets. In turn, it is difficult or practically impossible to achieve the same performance and speed-up behavior even with several hundreds of processors even on architectures like the SEI Altix.
Liu Peng et al [17].	• Parallelizing Lattice Boltzmann Flow Simulation on Emerging Multi-core platforms. The algorithm used critical section-free, dual representation in order to expose maximal concurrency and data locality. • To compute performance of emerging multi-core platforms-PlayStation3 (cell broadband engine) and Compute Unified Device Architecture (CUDA)- are tested using the parallelism Lattice Boltzmann Method (pLBM), which is implemented with multi-thread and message-passing programming.	• Show that parallelism Lattice Boltzmann Method (pLBM) achieves good performance improvement, 11.02 for cell over a traditional Xeon cluster and 8.76 for Compute Unified Device Architecture (CUDA) graphics processing unit (GPU) over a Sempron Central Processing Unit (CPU).
Bing He et al [18].	• Parallelizing simulation of compressible Fluid Dynamics using Lattice Boltzmann Method (LBM).	• Lattice Boltzmann Method (LBM) takes significantly less compute time for unsteady flows when it can take the same time step as traditional finite difference method and not violate any stability criteria. • For solving the Navier-Stokes equations, the Lattice Boltzmann Method (LBM) model requires twelve distribution

		functions at each grid point, which would mean greater memory requirement.
Samuel Williams et al [19].	• Optimize of a Lattice Boltzmann Computation on State-of-the-Art Multicore platforms.	• Showed that their auto-tuned LBMHD (Studying homogeneous isotropic turbulence in dissipative magnetohydrodynamics) application achieves up to a 15x improvement compared with the original code at a given concurrency.
Jun Ni, et al [20].	• Parallelization of a Lattice Boltzmann Method (LBM) for Lid-driven Cavity Flow. • To analyze the performance of Parallel Lattice Boltzmann Method (LBM) for a lid-driven cavity flow.	• The message passing communication has little influence on the performance of the parallel Lattice Boltzmann Method because the communication times take up only a relatively small portion of the total computation time.

From the literature above, it shows there is no other research or study involving LBM under DGE. Therefore, it became our interest to explore the possibility of implementing D2Q9 model on DGE.

3. Proposed Technique

In this work, we develop strategy for implementation of LBM model, D2Q9 on DGE for Lid cavity flow [22] problem. The top boundary moves from left to the right (slip boundary) with constant velocity. The other three boundaries (i.e. left, bottom, right) does not move (no-slip boundary). For detail, there is two-dimensional (2D) media where the plate covers its top, so we want to simulate the particle dynamism when the plate moved from left to right. The Fig. 1 (a) below shows the illustration of two-dimensional model (2D) with two axis, *x* and *y*. Whereas, Fig. 1 (b) shows every particle move to nine directions with different velocities.

a b

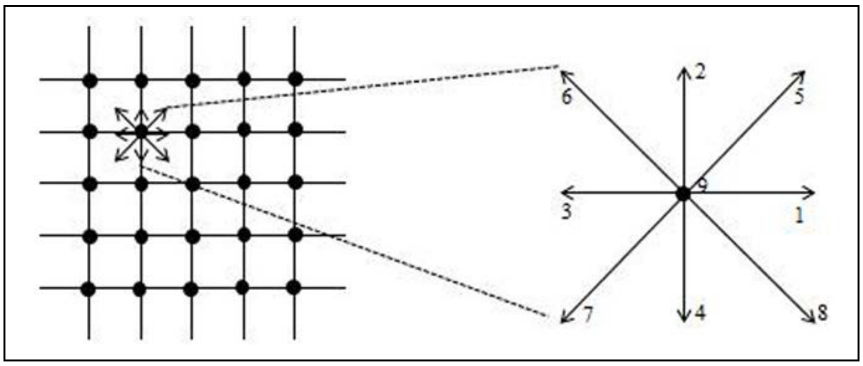

Fig. 1. Illustration of Two-Dimensions with nine velocity lattice model (D2Q9).

There are three steps to simulate the computation of D2Q9-cavity flow model, (a) Computation of initial value, (b) computation of Evolution, and (c) Computation of boundary conditions. In computation of initial value step, we compute the initial value for every particle that is static. This also includes the computation of the left, right, bottom and top boundary. When there is a movement on top boundary (slip boundary) from left to the right, the particle are limited to move in nine different directions with different velocity. Then simulation of that movement will be computed in the evolution part. Finally, boundary condition step will compute the collision when the particle hit the slip boundary i.e. the top, left, right and bottom boundary**.**

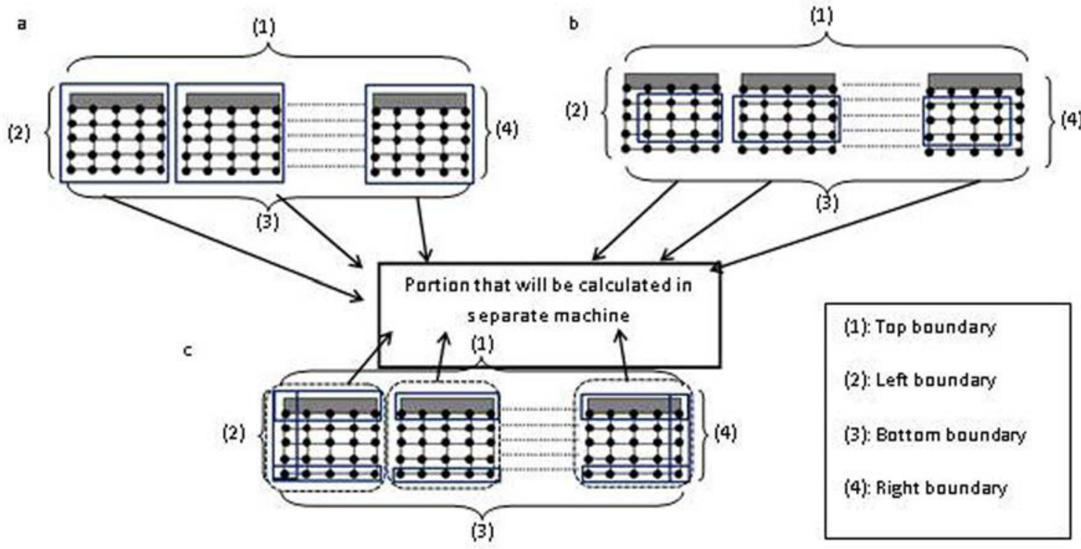

Fig. 2. Distribution of lattice in computing: (a) Initial value (b) Evolution (c) Boundary.

Fig. 2 depicts the distribution of lattice to different machines. In each step of the computation, we divide the lattice into a number of portions. Then each of the portion will be distributed to a computing node within the DGE to be computed. For the boundary step computation, Fig. 2 (c) shows that a machine computes the top, left and bottom boundary; a machine computes top, right and bottom boundary; and some machines just compute top and bottom boundary. To manage the distribution to the desktop computer we use HTCondor as the middleware. In the next section, we explain the details on how the distribution strategy works.

4. DGE and Its Middleware

The main purpose of Desktop Grid (DG) is harnessing the wasted resource of desktop computer where nowadays, especially the latest one, usually has resources above the needed. Even it has surplus resource, sometimes the users just use it for pretty ordinary application, such as word processor or browsing. So, DG tries to recycle those unused resource and act nearly as common grid computing. The powerful of this DG is depend mainly how much resource that can be scavenged. Nevertheless, DG cannot collect the resource with arbitrary from the desktop computer owner. In order to do so, DG has to attempt to spark the interest and convince of the desktop computer owner so as the result they want to volunteer their resources.

The user who want to devote their resource to DG is called volunteer. In term of volunteer, it means many likely can be happen. For example, the volunteer can decide whether when and how they want to volunteer their desktop resource; if the internet connection among the volunteer and the DG server stabile or not; also the possibility of the volunteer threaten the DG server. Obviously DG runs above thoroughly unstable environment.

Hence, DG needs a middleware which can tailor between DG and its volunteer. DG has some midllewares but not unconditionally uniform especially on task distribution. In this research we use HTCondor as middleware. To distribute the tasks, HTCondor uses the push concept where the tasks are pushed to the volunteer machine. In this case HTCondor uses match-making procedure based on the classified ads that is published by both HTCondor server and clients (volunteer). If the ads is match each other than the server will push the task to the client. In the server part, the content of ads are the requirement that must be fulfill by the client in order the application can be run. Otherwise, client advertise about condition of itself, for example the specification and when or how it is free. Beside those things, HTCondor also manage the checkpointing of the task if something happen to the client, such as network or power is down or the owner takes over the machine suddenly.

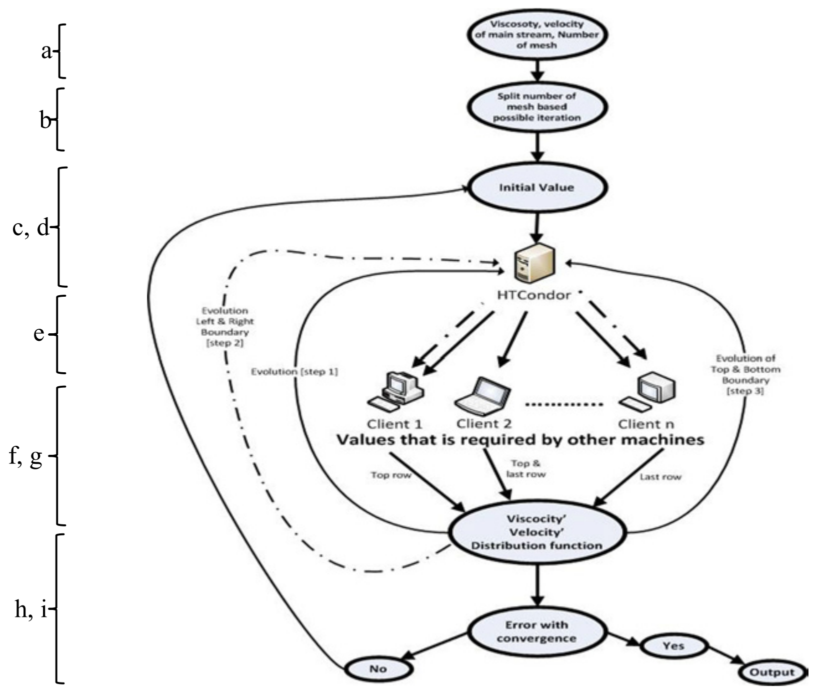
a
b
c, d
e
f, g
h, i
Viscosoty, velocity of main stream, Number of mesh
Split number of mesh based possible iteration
Initial Value
HTCondor
Evolution Left & Right Boundary [step 2]
Evolution [step 1]
Evolution of Top & Bottom Boundary [step 3]
Client 1
Client 2
Client n
Values that is required by other machines
Top row
Top & last row
Last row
Viscocity' Velocity' Distribution function
Error with convergence
No
Yes
Output

Fig. 3. Distribution process by HTCondor middleware.

5. The Proposed Implementation Strategy

This section explains the strategy that we proposed to execute the D2Q9 model under DGE. The scenario of the problem that we consider in this work is listed below:

- A media is filled with a type of liquid. This liquid is initially in static state and a plate covers the top of the media.
- When the plate moves from the left to the right, the particles in the fluid becomes dynamic.
- In this work, we simulate the behavior of the fluid when it moves from static to dynamic state.

To perform the computation under the DGE, we have to create possible portions of the whole computation. Then each portion will be sent to the client by HTCondor in order to be computed. We have listed the proposed steps to compute the behavior of the fluid is given below:

a) Set the value of viscosity and velocity of mainstream, and number of mesh.
b) Split number of mesh based on possible iteration.
c) Prepare the HTCondor submission file for initial value computation.
d) Submit the application to the DGE for the initial value computation.
e) HTCondor will distribute the application to the machines that join the DG.
f) Besides sending the generated result to the server, these machines also keep the needed result to compute the evolution process. The results sent to the server are results that are needed by other machines for the evolution process.
g) The evolution process that includes the boundary computation (step 1, 2 & 3).
h) The step (g) and (h) also have the same scenario as step (f) where the machine sends the needed result for the next computation besides keeping the needed result to themselves.
i) The iteration continues until convergence is reached. Finally, the results from machines are gathered.

6. Conclusion

In this paper, we elaborated the proposed idea of executing the D2Q9 model under DGE. We examined the algorithm for sequential D2Q9 cavity flow and developed a new strategy to be implemented on DGE. The proposed strategy still has high communication overhead that impacts the speedup that can be obtained. Further research is necessary to overcome communication overhead problem which still exist in our approach.

Acknowledgements

We as the authors would like acknowledge Ministry of Higher Education Malaysia for Fundamental Research Grant Scheme (FRGS, Project Code: UKM-TT-02-FRGS0205-2010) and National University of Malaysia as research facilities provider.

References

[1] Computational Fluid Dynamics Analysis Applied to Transportation Research, Argonne National Laboratory, http://www.anl.gov/TRACC/docs/fact_sheets/Comp_Fluid_Dynamics.pdf.

[2] Norberto Fueyo, COMPUTATIONAL FLUID DYNAMICS, ftp://ftp.cordis.europa.eu/pub/ist/docs/rn/fueyo.pdf.

[3] Bailey, Peter, Joe Myre, Stuart DC Walsh, David J. Lilja, and Martin O. Saar. "Accelerating lattice Boltzmann fluid flow simulations using graphics processors." In International Conference on Parallel Processing, ICPP'09, 2009.p. 550-557. IEEE.

[4] Luo, Li-Shi, Manfred Krafczyk, and Wei Shyy. "Lattice Boltzmann method for computational fluid dynamics." Encyclopedia of Aerospace Engineering, 2010.

[5] Begum, Romana, and M Abdul Basit. Lattice Boltzmann Method and its Applications to Fluid Flow Problems. European Journal of Scientific Research 22,. http://www.eurojournals.com/ejsr_22_2_08.pdf. 2008. no. 2: 216-231

[6] Xu, Kun, and Xiaoyi He. "Lattice Boltzmann method and gas-kinetic BGK scheme in the low-Mach number viscous flow simulations." Journal of Computational Physics 190, no. 1, 2003. p. 100-117.

[7] Nourgaliev, R. Robert., Truc-Nam Dinh, T. G. Theofanous, and D. Joseph. "The lattice Boltzmann equation method: theoretical interpretation, numerics and implications." International Journal of Multiphase Flow 29, no. 1, 2003. p. 117-169.

[8] Wikipedia, Computational Fluid Dynamics, http://en.wikipedia.org/wiki/Computational_fluid_dynamics.

[9] Yao, Chao-Hsu. "Grid Computation–The Fastest Supercomputer in the World". See http://csa. com/discovery guides/grid/review.pdf .2006.

[10] Erlanger, Leon. "Distributed computing: An introduction." Extreme Tech 4, 2002.

[11] Sarmenta, Luis FG. "Sabotage-tolerance mechanisms for volunteer computing systems." Future Generation Computer Systems 18, no. 4, 2002. Pg 561-572.

[12] Linux Magazine, Building distributed applications with BOINC – IDLE CYCLES, www.linux-magazine.com, issue 17. 2006.

[13] Janßen, Christian, and Manfred Krafczyk. "Free surface flow simulations on GPGPUs using the LBM." Computers & Mathematics with Applications 61, no. 12, 2011. p. 3549-3563.

[14] Wilke, Jens, Thomas Pohl, Markus Kowarschik, and Ulrich Rüde. "Cache performance optimizations for parallel lattice Boltzmann codes." In Euro-Par 2003 Parallel Processing. Springer Berlin Heidelberg, 2003. p. 441-450.

[15] Pohl, Thomas, Markus Kowarschik, Jens Wilke, Klaus Iglberger, and Ulrich Rüde. "Optimization and profiling of the cache performance of parallel lattice Boltzmann codes in 2D and 3D." PARALLEL PROCESSING LETTERS. 13, 2003.

[16] Pohl, Thomas, Frank Deserno, Nils Thurey, Ulrich Rude, Peter Lammers, Gerhard Wellein, and Thomas Zeiser. "Performance evaluation of parallel large-scale lattice Boltzmann applications on three supercomputing architectures." In Proceedings of the 2004 ACM/IEEE conference on Supercomputing, IEEE Computer Society, 2004. p.21.

[17] Peng, Liu, Ken-Ichi Nomura, Takehiro Oyakawa, Rajiv K. Kalia, Aiichiro Nakano, and Priya Vashishta. "Parallel lattice Boltzmann flow simulation on emerging multi-core platforms." In Euro-Par 2008–Parallel Processing, Springer Berlin Heidelberg, 2008. p. 763-777.

[18] He, Bing, Wei-Bing Feng, Wu Zhang, and Yu-Ming Cheng. "Parallel Simulation of Compressible Fluid Dynamics Using Lattice Boltzmann Method." In The first International Symposium on Optimization and System Biology (ISM'07), 2007. Pp. 451-58.

[19] Williams, Samuel, Jonathan Carter, Leonid Oliker, John Shalf, and Katherine Yelick. "Optimization of a lattice boltzmann computation on state-of-the-art multicore platforms." Journal of Parallel and Distributed Computing 69, no. 9, 2009. Pp 762-777.

[20] Ni, Jun, Yongxiang Zhang, Ching-Long Lin, and Shaowen Wang. "Parallelization of a lattice Boltzmann method for lid-driven cavity flow." UI research booth, Supercomputing , 2003.

[21] Litzkow, Michael J., Miron Livny, and Matt W. Mutka. "Condor-a hunter of idle workstations." In Distributed Computing Systems, 1988., 8th International Conference on, pp. 104-111. IEEE, 1988.

[22] Mussa, M. A., S. Abdullah, C. S. N. Azwadi, N. Muhamad, and K. Sopian. "Numerical simulation of lid-driven cavity flow using the lattice Boltzmann method." In WSEAS International Conference. Proceedings. Mathematics and Computers in Science and Engineering, edited by S. Kartalopoulos, A. Buikis, N. Mastorakis, and L. Vladareanu, no. 13. WSEAS, 2008.