Cloud computing versus in-house clusters: a comparative study

Omran Malik Omer Awad
The graduate college, Alneelain University
Khartoum, Sudan
omranawad@hotmail.com

Abdelmonim Mohamed Ali Artoli
Faculty of Science, Department of Physics,
Alneelain University, Khartoum, Sudan
and
College for Computer and Information Sciences
King Saud University
artoli@hotmail.com;aartoli@ksu.edu.sa

Awad Hag Ali Ahmed
Faculty of Computer and Information Science,
Alneelain University
Khartoum, Sudan

***Abstract*— Multi-core cloud clusters are best suited environment for academic institutions in the third world countries to gain supercomputing power and enable researchers to inquest new trends in scientific computing with affordable cost and less administrative load. This work aims to analyze the parallelism efficiency of a parallel computational fluid mechanics solver (the lattice Boltzmann method (LBM)) on multi-core cloud clusters. This paper demonstrates reliability and cost effectiveness of using tailor-made hired cloud clusters as compared to in-house high performance computing architecture. On these clusters we have found that the lattice Boltzmann implementation on a Cartesian grid is fully adaptive, highly flexible and cost effective to use for solving complex large fluid mechanical systems, such as flooding in real-time at a very low cost on leased cluster than in-house ones.**

Keywords: Computational fluid dynamics (CFD; clusters; Hyperthreading (HT), Lattice Boltzmann Method (LBM).

I. INTRODUCTION

Computational Science is considered a rapidly developing discipline influencing the development in different research areas such as engineering, biology, chemistry, and medicine [1]. Nowadays scientists and engineers spend more time in front of their personal computers, or a parallel superfast computers and less time in the physical laboratory [2]. The virtual scientific experiments software becomes a reality. The new approach of "simulate and analyze" replaced the old "trial and error" approach in several key science areas. Although a lot of commercial and free software is available to be used for scientific simulations. Working in new application areas and complex problems solution requires a good knowledge of fundamentals and mastering of new simulation tools, techniques, and algorithms.

In the old classical scientific approach, the physical system is first simplified and set in a form that suggests what type of phenomena and processes may be important, and correspondingly what experiments are to be conducted. In the absence of any known-type governing equations, dimensional inter-dependence between physical parameters can guide laboratory experiments in identifying key parametric studies [3]. The database produced in the laboratory is then used to construct a simplified engineering model which after validation will be used in other research, product development, design, and possibly lead to new technological applications. This approach has been used almost positively in every scientific discipline, i.e., engineering, physics, chemistry, biology, etc.

The simulation approach follows a parallel path but with some significant differences. First, the phase of the physical model analysis is more detailed: the physical system is formalized in a set of partial differential equations (PDEs), which represent continuum approximations to microscopic model. Such approximations are not possible for all systems, and sometimes the microscopic model should be used directly. Second, the laboratory experiment is replaced by simulation, i.e., a numerical experiment based on a discrete model. Such a model may represent a discrete approximation of the continuum partial differential equations, or it may simply represent a statistical representation of the microscopic model. In either case (continuum or discrete model), these algorithms have to be converted to software using an appropriate computer language, debugged and run on a workstation or parallel supercomputer. The output is usually a large number of files of a few Megabytes to hundreds of Terabytes, being especially large for simulations of time-dependent phenomena [4].

The term cloud computing reveals the state of the art trends toward information technology in general and in particular to the high performance parallel applications. The reason behind the rapidly increasing demand for the cloud computing services is cost effectiveness. There are so many different definitions to the term Cloud Computing [5] [6] [7] [8] [9]. To give a clear idea about cloud computing we introduce the most common definition according to the work done by Ebejer et al. [9]. This common definition focuses on computing resources accessible from anywhere throughout public network protocols. Also they specified five major characteristics of cloud computing, these characteristics are: on-demand self-service, broad network access, resource pooling, rapid elasticity and measured service. Their definitions stated that there are three basic service models

978-1-4799-3351-8/14/$31.00 ©2014 IEEE

which are (software as a Service, Platform as a Service, and Infrastructure

978-1-4799-3351-8/14/$31.00 ©2014 IEEE

as a Service). They have classified the deployment methods into

four main categories: private cloud, community cloud, public cloud and hybrid cloud. In this paper we have utilized the Amazon IaaS model. Amazon cloud services is a public deployment provider.

II. 2D9Q LBM SOLVER

Our work in this paper based on the lattice Boltzmann method, a newly adapted mesoscopic simulation solver. It is a discretization of the Boltzmann equation that describes the evolution of particles in kinetic theory [10]. Due to its simple implementation, straightforward parallelization and easy grid generation, the capability of the lattice Boltzmann method has been demonstrated in various complex applications including Newtonian blood flow simulations, non-Newtonian and suspension flows and complex geometry. As time-dependent flow simulations are known to be computationally expensive, a need for an efficient flow solver is crucial. Traditional Navier-Stokes solvers frequently use artificial compressibility and pressure projection methods to accelerate convergence. The governing equations for lattice Boltzmann method can be obtained from [11] [12].

Our code adopted the algorithm shown in Table 1.The source code is based on MPI-2 libraries with C language. It is executed under Linux OS.

Table 1 Lattice Boltzmann simulation algorithm

	Lattice Boltzmann Algorithm
1	Allocate memory
2	Initialize parameters
3	Partition the domain
4	***Do***
5	Compute mass and momentum
6	MPI_Reduce computed mass & momentum
7	Set upper and lower velocity boundary conditions
8	Set inflow and outflow pressure boundary conditions
9	Collide lattice sites
10	Stream sties
11	Set Bounce back on links
12	Send boundary info to west neighbor
13	Send boundary info to east neighbor
14	Update periodic boundary
15	***End do***
16	Compute output

We choose to divide the problem space geometry into n subdomains among x-axis. To ensure load balancing in simple way, we set $n = p$, where p is the number of processors involved in the solution. The subdomains then distributed randomly to the processors in a way that each processor will have one subdomain to process it. This technique has enhanced communications and lead to better load balancing, the disadvantage of this technique appear on small number of processors which cause increase time in local calculations. This drawback however is negligible when the code is executed in modern multi-core processors. The domain partitioning is illustrated in Fig. 1

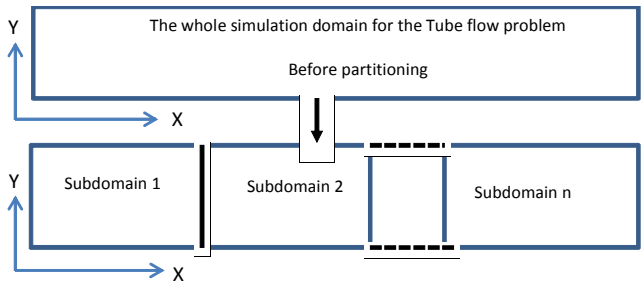

Fig. 1. Tube flow partitioning among x-axis

III. METHODOLOGY

In the following sections we will discuss the used performance measurements. Our benchmark consists of two-dimensional tube flow tested on five different cloud platforms and one real cluster. Before we go into our test procedures, we will give a brief introduction to measurement metrics being used throughout the next sections.

To evaluate the LBGK implementation on parallel environment we introduced a modified model to suite the selected scientific cloud computing environment. We consider that the parallel LBGK run time is composed as averages of four basic elements (computation time, communication time, idle time, and steal time). Here the steal time (***st***), valid only non-hardware virtualized environments (m1.xlarge, m2.4xlarge, and c1.xlarge). It shows the time when the real CPU was not available to the virtual machine because the hypervisor granted the CPU to another virtual machine running in the same computer or the hypervisor just needed the CPU power for itself.

We got the steal time value, when we issue the command *top* in Linux prompt and monitor the output. The steal time is not considered when we run the tests on hardware virtualized machines (cc1.4xlarge and c2.8xlarge) and it is negligible when it comes to m2.4xlarge and c1.xlarge).

A. Speedup and Efficiency Performance

One of the most widely used performance metrics for the investigation of parallel computations [13] are the relative speedup S_P, and the parallel efficiency E_P. The ideal efficiency would be 100%, which corresponds to a speedup equals to ***P*** (where ***P*** is the number of processors), but this efficiency can never be achieved due to the additional time needed: for the communication between the processors, and the idle time of some processors due to a different problem size per processor (load balancing effects).

For the sake of this paper, we will adopt the following simple formula for calculating speedup and parallel efficiency in our benchmarks

$$S_p = \frac{T_1}{T_P}, \quad E_P = \frac{S_P}{P} = \frac{T_1}{P \times T_p} \qquad eq.(1)$$

Which P denotes for number of cores, S_P for parallel speedup, and T_1 is the execution time for single processing core, and T_P is the execution time for $\boldsymbol{P}$ processing cores.

B. *Formulation of simulation Parameters*

In this section we will describe the test benchmark problem. The tube flow problem is the candidate; we consider the tube geometry to be of 100 lattice units width and 1000 lattice units' length as shown in the following figure.

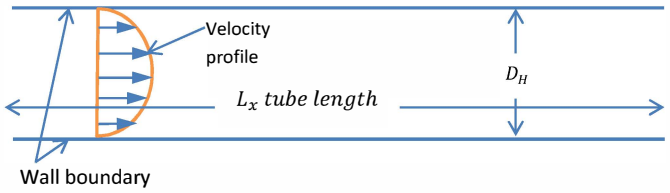

Fig. 2 Tube flow simulation geometry

The flow in the tube is considered to be at Re=100. Reynolds number (Re) is used to perform dimensional analysis of fluid dynamics problems, and as such can be used to determine dynamic similarity between different experimental cases [14] [15] [16].

To formulate the tube flow parameters we consider the Reynolds number equations as follows [17] :

$$R_e = \frac{vL}{\gamma} = \frac{vD_H}{\gamma} \qquad eq.(2)$$

Where $\boldsymbol{v}$ is the velocity, $\boldsymbol{L}$ is the characteristic length of the tube, γ is the kinematic viscosity, D_H is the characteristic tube diameter length

$$\gamma = \frac{\mu}{\rho} \qquad eq.(3)$$

Where μ is the dynamic viscosity of the fluid, ρ is the density of the fluid.

The velocity can be obtained from the following equation:

$$v = \left[\frac{(2\tau - 1)}{6}\right] e^2 \delta t \qquad eq.(4)$$

The lattice distance scales selected to be of $\Delta x = 0.1$ *and* $\Delta y = 0.1$, the initial velocity in x-direction is selected to be 0.1 lattice unit, by using the Eq. (4) we get:

$$\tau = \frac{\left[\frac{6v}{e^2} + 1\right]}{2} \qquad eq.(5)$$

By substituting v=0.001 and e=0.1 Eq. 5.3 we get $\tau = 0.8$ for relaxation time. Table 2 shows the complete parameters for the simulation initial setup.

Table 2 Tube flow simulation parameters

Parameter (in Lattice Units)	***Value***
Initial fluid velocity in x-direction (v_x)	0.1
Initial fluid velocity in y-direction (v_y)	0.0
Relaxation Time (τ)	0.8
Fluid density[1] (ρ)	1.0
Number of lattice cells in x-direction (L_x)	1000
Number of lattice cells in y-direction (L_y)	100
Pertubation[2] (*Statistical fluctuations*)	0.001

C. *Test bed platforms*

Our experiments have been conducted on five baseline platform architectures that are offered by Amazon scientific cloud computing services, beside the in-house cluster. These platforms are: four virtual cores machine (m1.xlarge), eight virtual cores machine (m2.4xlarge) and (c1.xlarge), eight (2 x Intel Xeon X5570, quad-core with hyperthread) machine (cc1.4xlarge), sixteen (2 x Intel Xeon E5-2670, eight-core with hyperthread) machine (cc2.8xlarge). Table 5 summaries the architectures used in our study.

Table 3 EC2 machine instances used for experiments on Amazon Cloud

EC2 Instances	***m1***	***m2***	***c1***	***cc1***	***cc2***
Cluster size (**n** nodes)	16	8	8	8	4
Virtual CPUs per machine	4	8	8	8	16
Computing Units per core[3]	2	3.25	2.5	2 x Intel Xeon X5570, with hyperthread	2 x Intel Xeon E5-2670, with hyperthread
Memory size in GB	15	68.4	7	22.5	60.5
Network Interface bandwidth	High	High	High	10 Gbps	10 Gbps
Support for H.W virtualization	No	No	No	No	Yes
Grouped in one place	No	No	Yes	Yes	Yes
System Architecture	64-bit	64-bit	64-bit	64-bit	64-bit
Operating System	Ubuntu 12			Centos 5.6	

Kauwolf cluster which has been built in King Abdulaziz University has the configuration that is shown in table.

[1] Density of water in room temperature $\rho = 1000\ Kg/m^3$

[2] Used to drive the flow in the tube in case of body force absence by making pressure gradient in the tube

[3] One compute unit (CU) provides the equivalent CPU capacity of a 1.0-1.2 GHz 2007 Opteron or 2007 Xeon processor according to Amazon EC2 documentation

Table 4 Kauwolf cluster nodes specifications

Item	*Description*
CPU	Intel ® Core ™ 2 Duo Quad CPU Q9400 2.66 GHz
FSB	1333 MHz front side bus speed
CPU cores	4 cores each with 64-bit instruction set
Memory	3.56 GB SDRAM, 6M of L2 cache

IV. RESULTS

In this section we will present the obtained results for the two types of HPCs, namely the Kauwolf cluster (built by the researcher at King Abdul-Aziz University) and the Amazon cloud initiated clusters. These results are then compared in terms of speedups performance and parallel efficiency.

A. *SPEEDUP PERFORMANCE*

The graph in Fig. 3 shows the speed up performance comparison between Kauwolf and m1.xlarge cluster which is the least performing platform among Amazon clusters. We notice significant variations in the speedup performance of our CFD solver on Kauwolf cluster as compared to the m1.xlarge cluster. These variations can be attributed to one or more of several factors including network congestion, virtualization impact, and memory bandwidth.

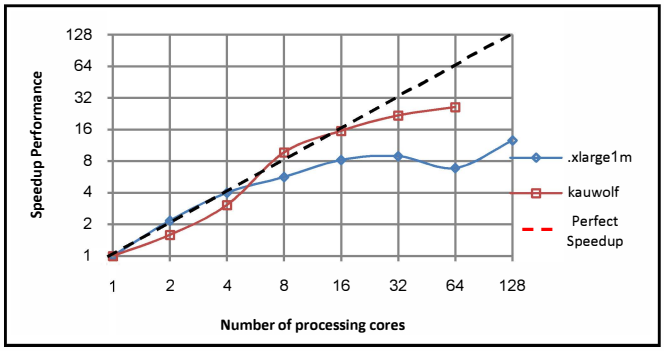

Fig. 3 Speedup performance for Kauwolf and m1.xlarge Amazon clusters

In Fig. 4 a speedup comparison between Kauwolf cluster and cc2.8xlarge cluster the largest platform configuration in Amazon clusters at the research time is presented. It has been noticed that cc2.4xlarge has achieved better performance in term of stability in speedup even for higher numbers of processing cores. In other words the performance is linear among various number of processing cores. This makes it a candidate for high performance computing applications.

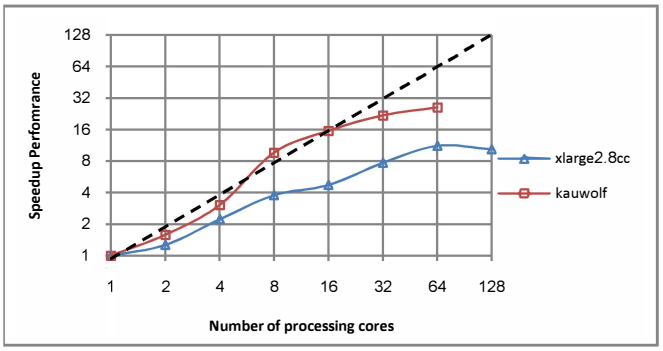

Fig. 4 Speedup performance for Kauwolf and cc2.8xlarge Amazon clusters

The speedup comparison for all test beds shown in Fig. 5. It is obvious that cc1.4xlarge and cc2.8xlarge achieved the best performance in term of speedup among all other Amazon clusters. This speedup is very acceptable for running parallel applications such as CFD ones.

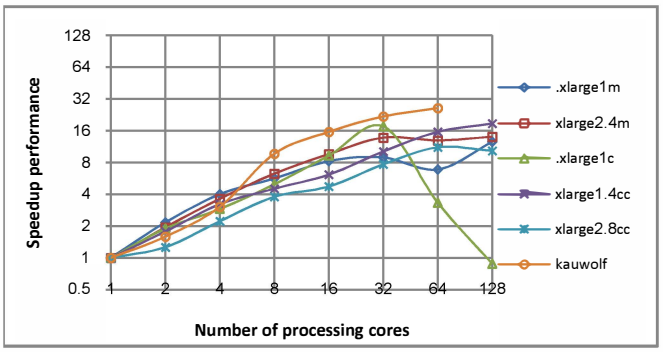

Fig. 5 Speedup performance for Kauwolf and all Amazon clusters

B. *PARALLEL EFFICIENCY*

To compare the Kauwolf cluster with the smallest test bed in Amazon clusters presentation is shown in Fig. 6. In this graph m1.xlarge outperform Kauwolf cluster in term of parallel efficiency, although it is the least performing Amazon cluster. This shows that cloud cluster are efficient for running parallel applications and highly scalable.

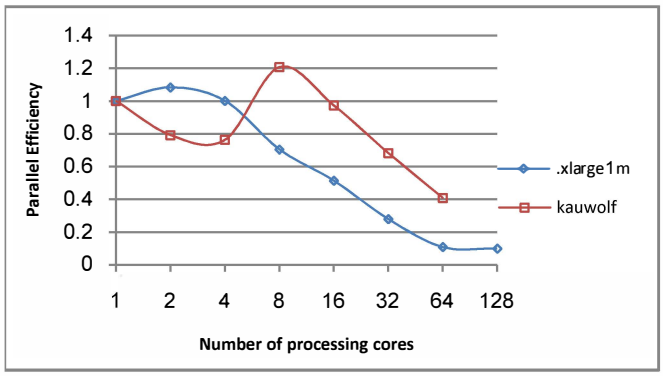

Fig. 6 Parallel Efficiency for Kauwolf and m1.xlarge Amazon clusters

Finally, in Fig. 7 we present a parallel efficiency comparison between Kauwolf cluster and cc2.8xlarge cluster. In this graph we notice that cc2.8xlarge parallel efficiency for small number of processing cores is fluctuating, but the performance is much linear when the number of processing cores goes higher.

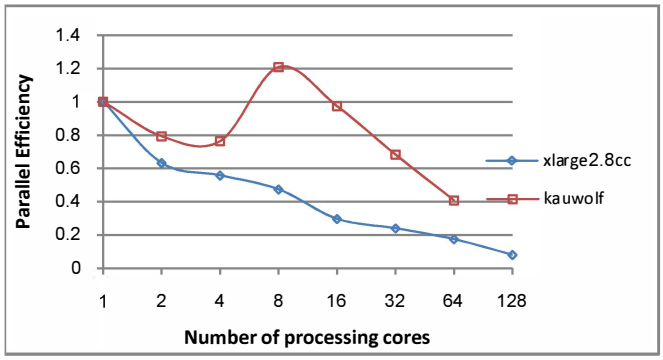

Fig. 7 Parallel Efficiency for Kauwolf and cc2.8xlarge Amazon clusters

In Fig. 8 all Amazon test beds performance results are presented with Kauwolf cluster results included also. we may conclude that the efficiency of the cloud cluster decays with increasing number of processing cores if the number of processing cores is large enough (more than 8 processors). The fluctuation in the efficiency for this in-house cluster may be attributed to the fact that the researcher was using the campus network which is sometimes not reliable due to network traffic.

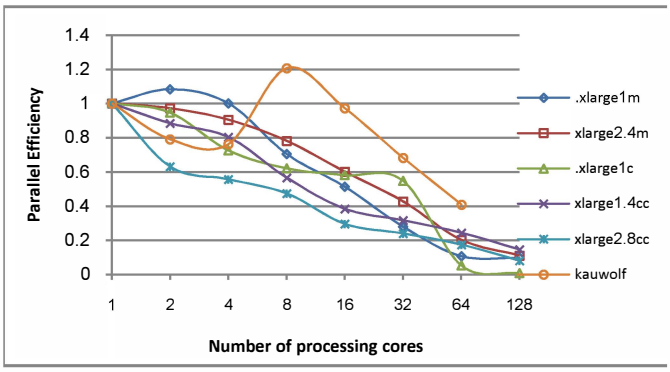

Fig. 8 Parallel Efficiency for Kauwolf and all Amazon clusters

V. CONCLUSION

The objectives of this work were to develop an in house parallel CFD solver based on 2D lattice Boltzmann method, capable of simulating the hydrodynamics of fluids in 2D, to analyze the performance of the developed solver among both commodity of the shelf Beowulf cluster built for academic researches and cloud computing clusters, to benchmark several scientific cloud computing platforms to assess the validity of the cloud computing services for running scientific computing applications reliably and efficiently.

To achieve these objectives, it was necessary to build an in-house cluster and to rent computing facilities from cloud provider as well. We have built Kauwolf as our in-house cluster at King Abdulaziz University in addition to five clusters in Amazon cloud computing services with different architectures being motivated for a financial and computing requirement.

To compare the performance of these clusters, a highly optimized computational CFD solver (The lattice Boltzmann solver) was implemented. The comparison results were given in figures 1 – 8 which compared speedup performance and parallel efficiency. From these figures we can conclude that:

- Cloud computing clusters achieve better cost per computing power than in-house clusters.
- General purpose platforms in Amazon could be used as high performance cluster for simulating CFD phenomena with reduced cost in comparison with high end dedicated platforms for scientific cloud computing in Amazon such as cc1.4xlarge and cc2.8xlarge platforms.

VI. FUTURE WORK

We are hoping to use the findings of this work to implement a general purpose highly optimized, self-tuning, less costing and robust CFD solver to be used for forecasting and engineering problems.

Although many research institutes launched projects for management tools to automate clusters deployment on Amazon cloud computing the researcher believe that still these tools needs

to be optimized. Also we need to develop general purpose tools that are dynamically adaptable to other cloud computing providers. We recommend that this issue would be tackled in a future research.

ACKNOWLEDGMENT

Work conducted by the researchers is supported (in part) by King Abdulaziz University – Faculty of Computing and Information Technology – Khulais Branch.

REFERENCES

[1] A. A. Mohamed, Lattice Boltzmann method fundamentals and Engineering - Applications with Computer codes, Springer, 2011.
[2] D. Becker, T. Sterling, D. Savarese, J. Dorband, U. Ranawake and C. Packer, "BEOWULF: A parallel workstation for scientific computation," 1995.
[3] G. E. Karniadakis and R. M. Kirby, Parallel scientifc computing in C++ and MPI, Cambridge University Press.
[4] P. Pacheco, An introduction to Parallel Programming, USA: University of San Francisco, 2011.
[5] J. Rhotas, Cloud computing explained: Implementation Handbook for Interprises, 2010.
[6] Reese, Cloud Application Structures: Building applications and infrastruture, 2009.
[7] T. G. Peter Mell, "The NIST Definition of Cloud computing," *NIST Special Publication,* 2011.
[8] V. Mauch, M. Kunze and M. Hillenbrand, "High Performance cloud computing," *Future Generation Computer Systems,* vol. 29, 2013.
[9] J. P. Ebejer, S. Fulle, G. M. Morris and P. W. Finn, "The Emerging Role of Cloud Computing in Molecular Modeling," *Journal of Molecular Graphics,* 2013.
[10] Abdelmonem. M. Artoli, A. G. Hoekstra and P. M. Sloot, "Mesoscopic simulations of systolic flow in the human abdominal aorta," *Journal of Biomechanics,* vol. 39, 2006.
[11] Abdelmonem. M. Artoli, D. Kandahi, H. C. Hoefsloot and A. G. Hoekstra, "Lattice Boltzmann, a robust and accurate solver for Interactive computational Hemodynamics," *Computational Science,* vol. 2657, pp. 1034-1043, 2003.
[12] R. Benzi, S. Succi and M. Vergassola, "The Lattice Boltzmann equations: Theory and applications," *Nuclear Physics,* vol. 222, no. 3, 1992.
[13] Drikakis and R. Zahner, "Investigation of the efficiency of a 3D parallel Implicit and Multiblock Navier-Stokes Solver," *Parallel Computational Fluid Dynamics: Algorithms and Results using Advanced Computers,* pp. 289-296, 1997.
[14] Abelmonem. M. Artoli, D. Kandhai, H. C. Hoefsloot and A. G. Hoekstra, "Lattice Boltzmann, a Robust and Accurate Solver for Interactive Computational Hemodynamics," *Computational Science,* vol. 2657, pp. 1034-1043, 2003.
[15] Abdelmonem. M. Artoli, D. Kandhai, H. J. Hoefsloots, A. G. Hoekstra and P. M. Sloot, "Lattice BGK simulations of flow in symmetric bifuraction," *Future Generation Computer Systems,* vol. 20, no. 6, pp. 909-916, 2004.
[16] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Oxford University Press, 2002.
[17] Abdelmonem. M. Artoli, "Mesoscopic Computational Haemodynamics," *PHD Thesis,* 2003.