Load balancing of parallel cell-based blood flow simulations

Alowayyed, S.; Závodszky, G.; Azizi, V.; Hoekstra, A.G.

DOI
10.1016/j.jocs.2017.11.008
Publication date
2018
Document Version
Final published version
Published in
Journal of Computational Science

Journal of Computational Science 24 (2018) 1–7

Contents lists available at ScienceDirect

Journal of Computational Science

journal homepage: www.elsevier.com/locate/jocs

Load balancing of parallel cell-based blood flow simulations

S. Alowayyed[a,b,*], G. Závodszky[a,c], V. Azizi[a], A.G. Hoekstra[a,d]

[a] Computational Science Lab, Institute for Informatics, Faculty of Science, University of Amsterdam, The Netherlands
[b] King Abdulaziz City for Science and Technology (KACST), Riyadh, Saudi Arabia
[c] Department of Hydrodynamic Systems, Budapest University of Technology, Hungary
[d] ITMO University, Saint-Petersburg, Russian Federation

ARTICLE INFO

Article history:
Received 8 March 2017
Received in revised form 10 July 2017
Accepted 11 November 2017
Available online 13 November 2017

JEL classification:
47.11.–j, 47.57.E–

Keywords:
Load balance
Domain decomposition
Blood flow
Parallel suspension flow simulation

ABSTRACT

The non-homogeneous distribution of computational costs is often challenging to handle in highly parallel applications. Using a methodology based on fractional overheads, we studied the fractional load imbalance overhead in a high-performance biofluid simulation aiming to accurately resolve blood flow on a cellular level. In general, the concentration of particles in such a suspension flow is not homogeneous. Usually, there is a depletion of cells close to walls, and a higher concentration towards the centre of the flow domain. We perform parallel simulations of such suspension flows. The emerging non-homogeneous cell distributions might lead to strong load imbalance, resulting in deterioration of the parallel performance. We formulate a model for the fractional load imbalance overhead, validate it by measuring this overhead in parallel lattice Boltzmann based cell-based blood flow simulations, and compare the arising load imbalance with other sources of overhead, in particular the communication overhead. We find a good agreement between the measurements and our load imbalance model. We also find that in our test cases, the communication overhead was higher than the load imbalance overhead. However, for larger systems, we expect load imbalance overhead to be dominant. Thus, efficient load balancing strategies should be developed.

1. Introduction

Load imbalance is the situation where the workload between processes is unequal. In case of cellular suspension simulations it might arise due to the inhomogeneity of the cell distribution within the simulation domain. This is typically a major source of loss of parallel efficiency for bio-fluid flows simulations such as blood flow simulations. Blood is a dense suspension of red blood cells (RBCs). Typically, the volume fraction of RBCs in human whole blood is 40–45%. Blood also contains other cells (platelets, white blood cells) with volume fractions of less than 1% [1]. The large number of RBCs present in the blood (5 million/mm^3) [2] leads to complex rheological behaviour such as shear thinning [3]. Another interesting phenomenon is the cross-stream migration of the RBCs, which is caused by the RBCs' complex motion and deformability, and the presence of solid tube walls [4]. This migration results in a higher RBC concentration in the middle of a channel flow and a depletion of RBCs near the channel walls, leading to a cell-free layer (CFL) that acts as a lubricant speeding up the core of the channel flow [4,5].

For a more detailed understanding of the physiological processes related to cardiovascular disease (such as atherosclerosis, embolisms, thrombosis, and thrombolysis), we should be able to simulate hundreds of millions of individual RBCs and other blood constituents explicitly [6–8]. This requires extensive computing resources, notably on massive parallel computers [9].

Our cell-based blood flow model, presented in [8,10], uses the lattice Boltzmann method (LBM) [11], which is a mesoscale method for solving fluid flows. For this purpose, we used Palabos, a fully parallelised LBM-based fluid solver [12] available in the public domain. The immersed boundary method (IBM) [13] is a fluid-structure coupling method used to provide a bidirectional interaction between the fluid and the vertices of the RBCs. The RBCs themselves were modelled using a boundary element method. Hemocell, which is a cell-flow simulation code based on Palabos, was developed earlier and benchmarked [8]. The Hemocell code implements the vertices of RBCs using a spectrin-link model to represent an RBC [14], and also implements a systematic coarse-graining procedure to select a varying number of vertices [15]. Furthermore, the Hemocell code

* Corresponding author at: Computational Science Lab, Institute for Informatics, Faculty of Science, University of Amsterdam, The Netherlands.
E-mail addresses: S.A.Alowayyed@uva.nl (S. Alowayyed), g.zavodszky@uva.nl (G. Závodszky), V.W.AziziTarksalooyeh@uva.nl (V. Azizi), A.G.Hoekstra@uva.nl (A.G. Hoekstra).

https://doi.org/10.1016/j.jocs.2017.11.008

defines models for RBC–plasma and RBC–RBC interactions [8] using Lagrangian and Eulerian descriptions.

Both Palabos and Hemocell utilise a domain decomposition algorithm for parallelisation [8]. Currently, domains are chosen to be equal in volume, i.e. they contain a similar number of Eulerian fluid points in each subdomain. This might create a load imbalance because of the CFL and the elevated RBC concentration at the core of the flow (i.e. the inhomogeneity of the RBC distribution, which is common in suspensions in general). Based on our previous benchmarks, we find that most of the computing time is spent on the RBC dynamics, namely RBC–RBC and RBC–fluid interactions [8]. Therefore, the computing time in a domain is mainly determined by the number of RBCs in that domain. This load imbalance might become a major source of loss of parallel efficiency when a simulation runs on a large number of processors. In addition, execution also suffers from communication overhead, and it is not immediately clear to what extent the load imbalance overhead is more severe than the communication overhead. We did not address this in earlier performance measurements [8] because in that study the domain was unbounded (no walls present). Clausen et al. [9] reported a decrease in efficiency because of load imbalances. However, to the best of our knowledge, a structured and detailed study, both theoretical and experimental, on the effect of RBC distribution in channel flows on overall performance has not been carried out.

Here, we discuss extensive load balance studies on suspension flow in channels, using the concept of the fractional load imbalance overhead proposed by Axner et al. [16] and the fractional communication overheads proposed by Fox [17]. We extend these concepts to suspension simulations, propose a theory to estimate the fractional load imbalance overheads, and validate this theory with measurements.

2. Parallel performance expressed in terms of fractional overheads

Our starting point was the performance prediction model of Axner et al. [16] and the notion of fractional communication overheads introduced by Fox [17]. First, assume that the execution time of a parallelised simulation on one processor, T_1, is equal to the execution time of the sequential simulation:

$$T_1 = T_{seq}. \tag{1}$$

We write the parallel execution time on P processors T_P as:

$$T_P = \left(\frac{T_{seq}}{P}\right) + \sum_j T_j^{Overhead}(P), \tag{2}$$

where $T_j^{Overhead}$ is the time spent on the different overhead types denoted by j. Overheads can include communication, or load imbalances. Axner et al. showed that it is possible to identify many separate sources of overheads. They also showed that overheads can, in principle, be measured and predicted with some degree of accuracy [16].

The speed-up and efficiency can be written as:

$$S_P = \frac{T_1}{T_P} = \frac{P}{(1+\sum_j f_j)}, \tag{3}$$

$$\varepsilon_P = \frac{S_P}{P} = \frac{1}{(1+\sum_j f_j)}, \tag{4}$$

where

$$f_j = \frac{T_j^{Overhead}}{(T_{seq}/P)} \tag{5}$$

are the fractional overheads. Note that, by combining Eqs. (2) and (5), $T_P = (T_{seq}/P)(1+\sum_j f_j)$. This means that every overhead j adds a fraction f_j execution time to the ideal execution time T_{seq}/P.

Note that if $f_j \ll 1$ for all j overhead types,

$$\varepsilon_P \approx 1 - \sum_j f_j. \tag{6}$$

Expressing different types of overheads j in terms of fractional overhead defined in Eq. (5) allows easy comparison of these overheads, as they appear in the equation for the parallel efficiency, and enable us to determine the severity of one overhead compared to another. Note that the overheads depend on the number of processors, not only explicitly, but also through the dependence of $T_j^{Overhead}$ on P. Thus, comparing the overheads as a function of P will give an indication of which overhead should be reduced first to improve the overall scalability of the simulation.

In this study, we mainly focus on finding an expression for the fractional load imbalance overhead f_{LI} for our cell-based blood flow simulations, and then compare it with the fractional communication overhead f_{comm}. However, this framework can be used for any type of parallel computing. More extended expressions and measurements are provided by Axner et al. [16].

2.1. Fractional load imbalance overhead

Assume that the communication time is zero and that the time taken by processor i to perform its required computations is t_i, where $i = 0, \ldots, P-1$. We now assume a load imbalance, so in general $t_i \neq t_j$. This will lead to a reduction in efficiency because the execution time of parallel code is determined by the slowest processor, i.e. the maximum t_i among processors:

$$T_P = t_i^{\max} = \left(\frac{T_{seq}}{P}\right) + \left[t_i^{\max} - \left(\frac{T_{seq}}{P}\right)\right], \tag{7}$$

where we have written the expression in a form matching Eq. (2). If there is no load imbalance, i.e. all t_i are equal, then $T_{seq} = Pt_i$ and the second term in Eq. (7) equals 0, as required. From Eq. (5), the fractional load imbalance overhead can be expressed as:

$$f_{LI} = \left[\frac{(t_i^{\max} - (T_{seq}/P))}{(T_{seq}/P)}\right] = \left(\frac{t_i^{\max}}{\langle t_i \rangle}\right) - 1, \tag{8}$$

where $\langle t_i \rangle$ is the average computation time per processor.

An interesting feature of Eq. (8) is that it also takes heterogeneous architectures into account, and expresses the load imbalances caused by the presence of faster and slower processors in a parallel machine. Load imbalance can arise due to the different workload per processor, but also by equal computational loads executed on processors with different computational speeds (as can happen in large HPC systems). Eq. (8) naturally captures both effects, as it is fully defined in terms of execution times per processors. It also captures the combined effect of unequal amounts of work per processor and unequal processor speeds.

2.2. Fractional load imbalance for RBC suspensions

As discussed in [8], the execution time of the cell-based blood flow simulation is mainly determined by the computations required for simulating the mechanical behaviour of the RBCs. Take N as the total number of RBCs in the simulation and τ the required computing time per RBC. This leads to

$$T_{seq} = N\tau n_{iter}, \tag{9}$$

where n_{iter} is the total number of iterations (time steps). Note that we assumed that the additional time for the Lattice Boltzmann flow

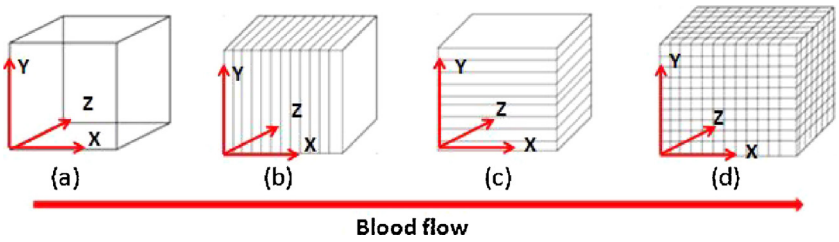

Fig. 1. (a) Sequential domain, with blood flow in the positive x-direction, (b) 1-D decomposition perpendicular to the blood flow (vertical), (c) 1-D decomposition parallel to the blood flow (horizontal), and (d) a 3-D decomposition.

computations can be neglected. We also assume that τ is constant. We can now calculate the time per iteration per processor i:

$$t_i = N_i \tau, \tag{10}$$

where N_i is the number of RBCs in the domain of processor i and $N = \sum_{i=0}^{P-1} N_i$. In Hemocell, this is not entirely correct, as a single RBC can be in several domains at the same time, thereby introducing small errors as some RBCs could be counted twice if they cross processor boundaries. We formulate the theory in this manner to keep it insightful; however, the expressions can be easily formulated in terms of the number of RBC discretisation points per processor. In general, $N_i \neq N/P$, which leads to a load imbalance. This could, for instance, occur because the subdomains in the domain decomposition are all equal in size but the concentration of RBCs differs in the computational domain (as in blood flow in an artery). Substituting Eq. (10) in Eq. (8) results in:

$$f_{LI} = \left(\frac{N_i^{\max} \tau}{(N\tau/P)} \right) - 1 = \left(\frac{N_i^{\max}}{\langle N \rangle} \right) - 1. \tag{11}$$

If the number of RBCs is the same in each processor, then $f_{LI} = 0$, as expected. To calculate f_{LI} theoretically, we need information on the concentration profile in the flow domain. This may not always be known, but for now we assume that it is available, either through experiments, theory, or earlier simulations. Thus, using the haematocrit profile, the domain size, the volume of RBCs, and knowing the domain decomposition used for the parallel simulation, we can calculate the number of RBCs in a processor, i, as follows:

$$N_i = H_i \left(\frac{V_i}{V_{RBC}} \right), \tag{12}$$

where N_i, H_i, and V_i are the number of RBCs, the local average haematocrit, and the volume of the domain of processor i, respectively. RBCs conserve their volume, so V_{RBC} is constant among the processors. From this, we can find the maximum and mean number of RBCs, and from that the theoretical value of f_{LI}. This value is compared with the actual measurements obtained from the simulations.

2.3. Simulation set-up

We considered flow in two types of domains. First, we considered flow between two parallel plates, which has a relatively simple 1-D "averaged" haematocrit profile. Next, we considered flow in a rectangular channel, which gives rise to a more complicated 2-D haematocrit profile. In all cases, we obtained simulation results for three domain sizes, resulting in different systems with numbers of RBCs ('small' – 4290 –; 'intermediate' – 7150 – and 'large' – 10868 –, respectively, all at a constant average Hematocrit of 38%) for both channel flow and the parallel plates cases. Moreover, we considered three different domain decompositions, as depicted in Fig. 1. We considered two types of one-dimensional decompositions (Fig. 1b and c), which allowed a detailed study of the load imbalance, and a three-dimensional decomposition (Fig. 1d) similar to those used in production runs. The intermediate two-dimensional decomposition case is not reported since the results are comparable to the 1-D and 3-D cases (data not shown). The number of cores ranged from 1 to 16 in all cases. Moreover, for the largest number of RBCs, we ran the 3-D decomposition using 1–1024 cores. All parameters are shown in Table 1. The domain size and the volume of red blood cells are expressed in Lattice Units (LU), where in these simulations 1 LU is equal to 1 μm. The relative size of RBCs is 8 LU. The RBCs can cross processor boundaries. As described in detail in Ref. [8], to do so three different fields (for fluid, surface particles and cell particles) with different envelope sizes are used and need to be transferred over domains.

As Fig. 1 shows, the blood flows in the positive x-direction. The channel flow case is a rectangular domain bounded by four no-slip walls parallel to the x–y and x–z planes, whereas the parallel plates case is bounded by two no-slip walls parallel to the x–z plane only. For both cases, a periodic boundary condition is applied on the inlet and outlet of the domain, and for the parallel plates also in the z-direction. The Reynolds number is tuned to be equal to the experiment in [9].

We measured the haematocrit profile in our simulations, based on the concentration of the Lagrangian points that construct the RBCs in a unit volume. For a measurement of the computational load this is more accurate than considering the RBCs' centroids. For the parallel plates case, we computed the average haematocrit profile in the y-direction; for the channel flow case, we computed the average profile in the y–z plane. Next, we used these measured haematocrit profiles and the number of processors to compute the fractional load imbalance overhead, according to Eqs. (11) and (12).

3. Results

To demonstrate the introduced method, we measured the haematocrit profiles for both the channel flow and the parallel plates cases. Next, the theoretical values of the fractional load imbalance overhead were calculated from the maximum and average numbers of Lagrangian points in the subdomain using (Eq. (10)). Then, we directly measured the pure computing time per processor t_i with the Allinea MAP profiler [18], and from that we computed the fractional load imbalance overhead directly using (Eq. (8)). Finally, we compared the experimental values with the theory. We also measured communication overheads with the MAP tool, and computed related fractional overheads.

3.1. Haematocrit distribution analysis

The haematocrit profiles for 'small', 'intermediate' and 'large' systems for the parallel plates case are shown in Fig. 2. We first equilibrated the simulations by running until the local haematocrit converged to a stable profile with fluctuations below the 1% level (averaged over 1000 timesteps). For all system sizes, it is clear that the haematocrit is high near the centre of the channel, and there are fewer RBCs close to the wall. These profiles are in qualitative

Table 1
Parameters used in the simulations, where N denotes the number of RBCs, LU means Lattice Units (1 LU = (1 μm), and Re is the Reynolds number of the flow.

Parameter	'Small'	'Intermediate'	'Large'
Geometry		Channel flow and parallel plates	
Domain size	$126 \times 64 \times 128$ LU^3	$210 \times 64 \times 128$ LU^3	$320 \times 64 \times 128$ LU^3
Number of processors	1–16	1–16	1–1024
Number of red blood cells	4290	7150	10868
Volume of a red blood cell		91.5077 LU^3	
Flow type	Flow in the positive x-direction with walls parallel to the	(1) x–y and x–z planes for channel flow (2) x–z plane only for the parallel plates	
Haematocrit		38%	
Re		5.8	

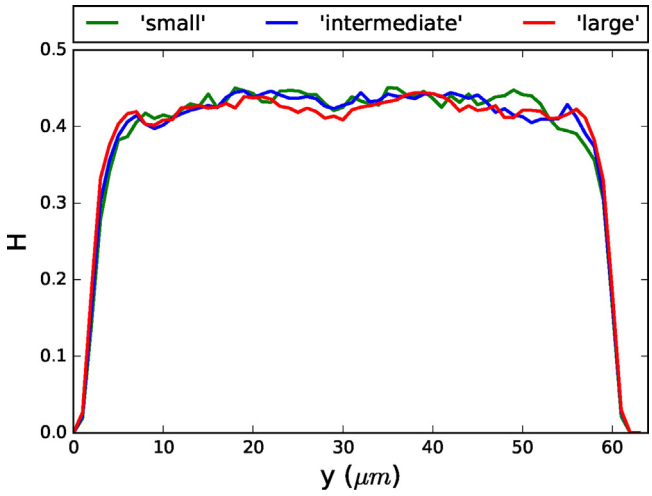

Fig. 2. Haematocrit distribution for the parallel plates case with different numbers of RBCs (N), average haematocrit H = 38%, distance between plates = 64 μm and Re = 5.8. Green is for 'small', blue for 'intermediate' and the red for 'large' systems.

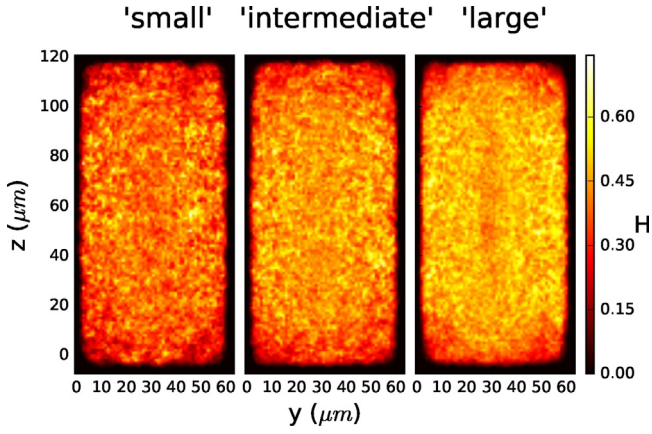

Fig. 3. Haematocrit distribution for the channel flow case with different numbers of RBCs (N), average haematocrit H = 38%, channel dimensions 128 μm × 64 μm and Re = 5.8. 'small' (left), 'intermediate' (middle), and 'large' (right) systems.

agreement with earlier simulations [5,6], although some details, such as the geometry used and the shear rate, are different.

The haematocrit for the channel flow is shown in Fig. 3. In all RBC cases, the rectangular haematocrit profiles clearly show depletion of RBCs near the domain walls, with pronounced depletion zones in the corners of the domain.

3.2. Measuring fractional load imbalance overheads

The results of the experiments are divided into two categories. The first category includes the results of fractional load imbalance overheads over the three different domain decompositions and all three system sizes for 1–16 cores. The second category includes the fractional overheads of the largest system size ('large' system) using 1–1024 cores. The Lisa computing facility [19] was utilised to obtain the results of the first category, while Cartesius [20] was used for the second. Moreover, we ran the same experiments on SANAM [21] giving, as expected, similar results for the load imbalance overhead (data not shown).

3.2.1. Different domain decompositions

The first set of results shows the fractional load imbalance for the different types of domain decompositions described earlier. In this set, we will use a small number of cores (1–16) to show the load imbalance effect on different domain decompositions. Fig. 4 shows the theoretical and measured fractional load imbalance overheads for the parallel plates cases with different numbers of RBCs. We measure the fractional load imbalance overheads for 1000 time steps and report the average and the standard deviation. In the horizontal domain decomposition, the fractional load imbalance overheads display the same trend for all system sizes. This is attributed to the similarity in haematocrit profiles, as illustrated by Fig. 2 . Moreover, the noticeable increase in fractional overhead from two to four cores is caused by the fact that cores that hold the central domains have many more RBCs than cores that hold the domains close to the walls.

For the vertical domain decomposition (Fig. 4, middle), the fractional load imbalance overhead is an order of magnitude lower than for the horizontal case. In fact, in theory one would expect a zero fractional load imbalance overhead as the heamatocrit profile does not depend on the x-coordinate. However, note that in some cases the size of the flow domain lattice in the x-direction is not exactly divisible by the number of processors; this leads to small load imbalances caused by different domain volumes per processor. To confirm this, the theoretical fractional load imbalance overhead for the 'large' system case is zero, because the length of the domain in the x-direction for this case is 320 LU, which is perfectly distributed among the different numbers of cores. Measurements reveal that we do get a very small load imbalance, which we attribute to small fluctuations in the haematocrit. On the other hand, for 'small' and 'intermediate' systems, the size of the domain is not exactly divisible by the number of processors, which leads to unequal distribution of subdomains among the cores. This is immediately noticeable in Fig. 4 (middle) as a small fractional load imbalance overhead, both in theory and measurements. Thus, unlike the horizontal domain decomposition, the haematocrit distribution did not play a significant role in the vertical domain decomposition.

Finally, the 3-D domain decomposition (Fig. 4, right) shows the smallest fractional load imbalance overheads. This can be attributed to the method we use to decompose the domain, in which we first decompose the x-axis, then the y-axis, and finally the z-axis. Therefore, the number of cores on the y-axis was kept as small as possible. This ensures a smaller effect from the inhomogeneous haematocrit distribution.

Similarly, for the channel flow case, Fig. 5 shows the fractional load imbalance overhead for different decompositions. The effect

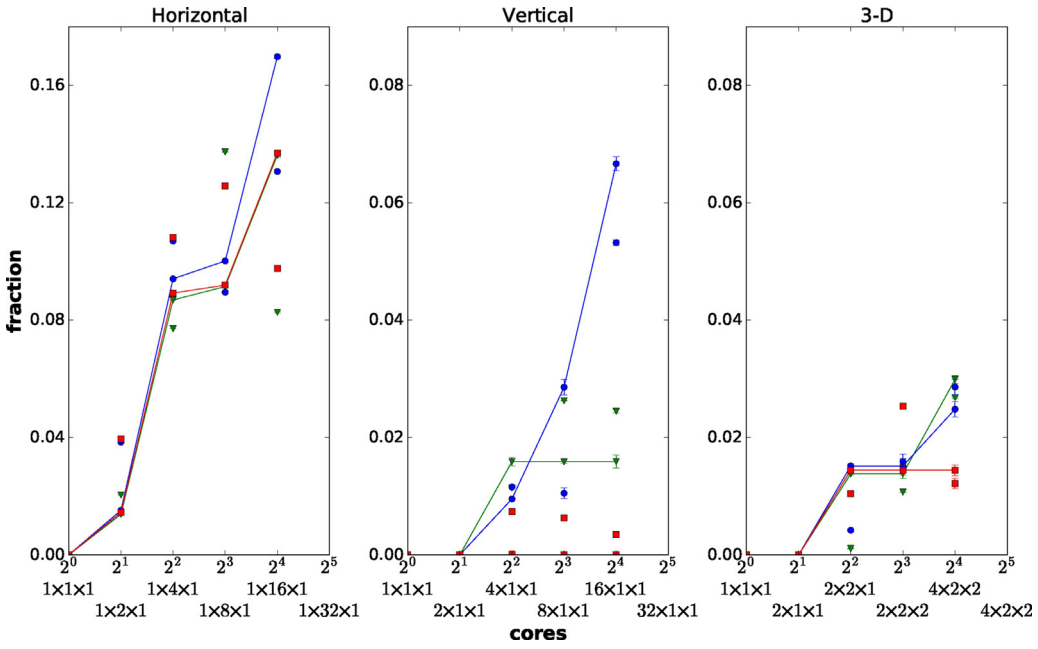

Fig. 4. Fractional load imbalance overheads for horizontal (left), vertical (middle) and 3-D (right) domain decompositions for the parallel plates case. The solid lines represent the theoretical values; the markers-only represent the measured values. The green (triangles marks), blue (squares marks), and red (rectangles marks) lines indicate 'small', 'intermediate' and 'large' systems, respectively. Note the difference in scale on the vertical axis between cases.

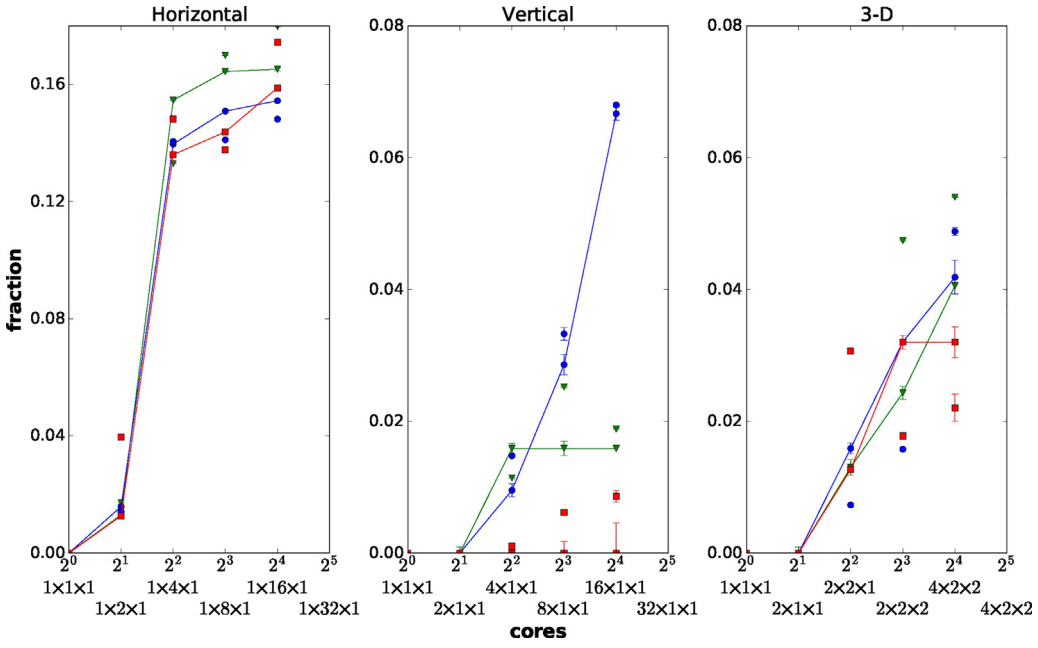

Fig. 5. Fractional load imbalance overheads for horizontal (left), vertical (middle), and 3-D (right) domain decompositions for the rectangular channel case. The solid lines represent the theoretical values; the markers-only represent the measured values. The green (triangles marks), blue (squares marks), and red (rectangles marks) lines indicate 'small', 'intermediate', and 'large' systems, respectively. Note the difference in scale between cases.

of the extra walls is clearly visible. In the horizontal domain decomposition (Fig. 5, left), we note a steep increase in the fractional load imbalance overhead from two to four cores, almost double the increase of the parallel plates case. On the other hand, the fractional load imbalance overhead for the vertical domain decomposition is approximately the same as the parallel plates case, which was expected.

The extra walls are most noticeable in the 3-D domain decomposition. In this case, the fractional load imbalance overhead increased by a factor two.

Finally, we observe that the measured values are in agreement with the theoretical values, in the sense that we find the general trend, and that our simple model always correctly identifies the order of magnitude of the fractional load imbalance overhead. In some cases there is a mismatch between the model and the measurement, e.g. for the large system with 3-D decomposition (see right panel in Fig. 5). This can be attributed to fluctuations in RBC count, and to other sources of load imbalance that do not scale linearly with the RBC count per processor (such as the size of the overlap regions to resolve RBC that cross processor boundaries [8]).

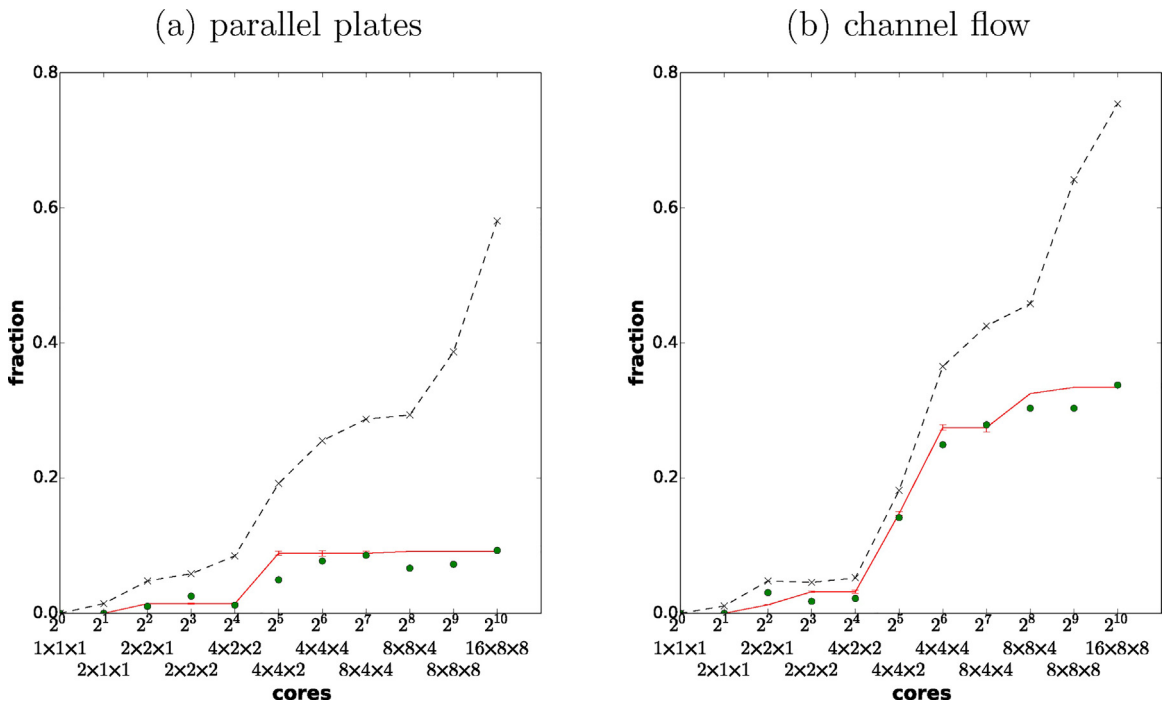

Fig. 6. Fractional communication, and load imbalance overheads for the parallel plates case (left) and channel flow (right) using 3-D domain decompositions. The solid lines represent the theoretical values of the fractional load imbalance overhead; the filled circles represent the measured values. The dashed line is the fractional communication overhead.

3.2.2. Large number of cores

For this set of experiments, we used the same configurations as the previous section; however, now only with 'large' system and a larger number of cores (1–1024 cores) were used, and we only considered 3-D decompositions. Moreover, we compared the fractional load imbalance overhead with other measured overheads. In our measurements, we considered the communication, and load imbalance overheads. Fig. 6 illustrates the fractional overheads for the parallel plates case. Here, the communication overhead is dominant. The load imbalance caused by the existence of external walls is observed when the number of cores in the y-direction is increased. For example, the increase in fractional load imbalance overhead from 16 cores ($4 \times 2 \times 2$) to 32 ($4 \times 4 \times 2$) is due to doubling the number of cores in the y-dimension from two to four.

Note that the total combined overheads (i.e. sum of the fractional communication and the fractional load imbalance overhead) can be as large as 1, or even higher. The case of 1 for instance means that with respect to the ideal situation with zero overheads the execution time is doubled.

The communication overhead in Fig. 6 was measured using the Allinea Map profiler. These measurements are substituted in Eq. (5), assuming $T_{seq} = T_1$. The source of communication overhead is mainly due to transmitting data between processors boundaries (related to the fluid flow field and RBCs crossing processor boundaries [8].

As before, the fractional load imbalance overhead for the channel flow, as shown in Fig. 6b, is higher than for the parallel plates case. The communication pattern is less dominant in this case, and, for a small number of cores, the fractional communication overheads are comparable to the fractional load imbalance overheads.

In general, the fractional communication overhead showed the same order of magnitude as the fractional load imbalance overhead and an increase was observed in both cases. We observed an increase in communication overhead in the channel flow case, due to the fact that communication depends lineary on the haematocrit. Higher haematocrit in the middle of the channel rises both fractional communication and load imbalance overheads.

Experiments have been carried out on both lisa and Cartesius. Comparing Figs. 4–5 with 6 shows, up to 16 processors, a similar fractional load imbalance overhead.

4. Discussion

We introduced fractional load imbalance overhead for parallel suspension simulations, specifically cell-based blood flow simulations. We formulated a methodology to estimate the fractional load imbalance overhead based on heamatocrit profiles, and compared it to direct run-time measurements, demonstrating good agreement. We used the haematocrit profile because of its importance in the blood flow. These calculations could, however, be generalised to any suspension simulation.

In the 3-D domain decomposition, the fractional load imbalance overhead was smaller than the fractional communication overhead. However, for larger system sizes (i.e. for order of magnitudes more RBCs), we expect a decrease in fractional communication overhead while the fractional load imbalance overhead remains the same. Given the size of the fractional load imbalance overheads on 1024 processors, $f_{LI} \approx 0.5$, the best achievable efficiency in this setup would be 2/3 (see Eq. (6)). Add to that communication overheads, and it is clear that for productions runs one needs to achieve better load balancing, reducing the fractional load imbalance overhead with at least one order of magnitude.

In this work we mainly concentrated on differences in local Hematocrit due to the presence of solid walls. However, there are also inhomogeneities possible in unbounded flows, caused by structure formation such as clustering [22]. So, if processor domains become smaller than typical cluster sizes, that could also lead to load imbalance. However, as these clusters are very dynamic objects, we would expect that this would lead to an increase in fluctuations in the fractional load imbalance overhead.

For future work will develop a load balance library (LBL), which will reduce the imbalance situation, and in turn will lead to a decrease in the fractional overheads and an increase in efficiency. The LBL could achieve load balance by adaptive non-equal domain decomposition, by tuning the clock speed of individual processors, or a combination of both. Taking the clock speed adjustments into account will add an energy reduction factor into the equation, in addition to the targeted efficiency, where in the end the reduction of wall clock speed is the only target. We plan to rely on METIS, with haematocrit profiles as input. After the initial phase of the simulation this profile might change. Therefore, rebalancing of the load should take place, but only when the rebalancing overhead is much lower than that due to the load imbalance itself and the fractional load imbalance is more than a given (problem specific) threshold value.

Acknowledgements

We acknowledge partial funding from the European Union Horizon 2020 research and innovation programme under grant agreement No 671564, the ComPat project. Author SA acknowledges funding by King Abdulaziz City for Science and Technology (KACST), Saudi Arabia. Author AGH acknowledges partial financial support by the Russian Scientific Foundation, grant # 14-11-00826. We acknowledge the Dutch Science Foundation for awarding us access to Cartesius, The Netherlands and the national center for computer Technology and applied math/Sanam research team, Saudi Arabia.

References

[1] O.K. Baskurt, Hardeman, Rampling, H.J. Meiselman, Handbook of Hemorheology and Hemodynamics, 2007.
[2] X. Li, P.M. Vlahovska, G.E. Karniadakis, Continuum- and particle-based modeling of shapes and dynamics of red blood cells in health and disease, Soft Matter 9 (1) (2013) 28–37.
[3] H.A. Barnes, The Flow of Suspensions, A Handbook of Elementary Rheology, vol. 331(6019, 2000, pp. 119–140.
[4] D.A. Fedosov, B. Caswell, A.S. Popel, G.E.M. Karniadakis, Blood flow and cell-free layer in microvessels, Microcirculation 17 (8) (2010) 615–628.
[5] H. Lei, D.A. Fedosov, B. Caswell, G.E. Karniadakis, Blood flow in small tubes: quantifying the transition to the non-continuum regime, J. Fluid Mech. 722 (2013) 214–239.
[6] L. Mountrakis, E. Lorenz, A.G. Hoekstra, Where do the platelets go? A simulation study of fully resolved blood flow through aneurysmal vessels, Interface Focus 3 (February) (2013) 20120089.
[7] D.A. Fedosov, H. Noguchi, G. Gompper, Multiscale modeling of blood flow: from single cells to blood rheology, Biomech. Model. Mechanobiol. 13 (2) (2014) 239–258.
[8] L. Mountrakis, E. Lorenz, O. Malaspinas, S. Alowayyed, B. Chopard, A.G. Hoekstra, Parallel performance of an IB-LBM suspension simulation framework, J. Comput. Sci. 9 (2015) 45–50.
[9] J.R. Clausen, D.A. Reasor, C.K. Aidun, Parallel performance of a lattice-Boltzmann/finite element cellular blood flow solver on the IBM Blue Gene/P architecture, Comput. Phys. Commun. 181 (6) (2010) 1013–1020.
[10] L. Mountrakis, Transport of blood cells studied with fully resolved models, Phd thesis), Universiteit van Amsterdam, Amsterdam, 2015.
[11] S. Succi, The Lattice Boltzmann Equation for Fluid Dynamics and Beyond, Clarendon Press/Oxford University Press, Oxford/New York, 2001.
[12] www.palabos.org (accessed 21.04.16).
[13] C. Peskin, The immersed boundary method, Acta Numer. 11 (January) (2002) 479–517.
[14] D.A. Fedosov, B. Caswell, G.E. Karniadakis, Systematic coarse-graining of spectrin-level red blood cell models, Comput. Methods Appl. Mech. Eng. 199 (29–32) (2010) 1937–1948.
[15] I.V. Pivkin, G.E. Karniadakis, Accurate coarse-grained modeling of red blood cells, Phys. Rev. Lett. 101 (11) (2008) 118105.
[16] L. Axner, J. Bernsdorf, T. Zeiser, P. Lammers, J. Linxweiler, A.G. Hoekstra, Performance evaluation of a parallel sparse lattice Boltzmann solver, J. Comput. Phys. 227 (10) (2008) 4895–4911.
[17] G.C. Fox, I.G. Angus, in: Solving Problems on Concurrent Processors, Prentice Hall, Englewood Cliffs, NJ, 1988.
[18] www.allinea.com/products/map (accessed 19.04.16).
[19] userinfo.surfsara.nl/systems/lisa (accessed 19.04.16).
[20] userinfo.surfsara.nl/systems/cartesius (accessed 19.04.16).
[21] http://www.top500.org/site/50458 (accessed 19.04.16).
[22] L. Mountrakis, E. Lorenz, A.G. Hoekstra, Scaling of shear-induced diffusion and clustering in a blood-like suspension, EPL (Europhys. Lett.) 114 (1) (2016) 14002.

S.A. Alowayyed is a PhD candidate in Computational Science Lab (CSL) at the University of Amsterdam. His thesis is concerned with Multiscale computing patterns. He received his Master in high performance computing from the University of Edinburgh. Saad Also works as a researcher in King Abdulaziz for Science Technology (KACST).

G. Závodszky graduated as a physicist and later obtained his PhD in the topic of fluid dynamics. Currently, he is an assistant professor at the Budapest University of Technology and also works at Computational Science Lab of the University of Amsterdam. Topics of his interest include blood flows of vascular diseases, cellular transport mechanics, high performance computing and GPU programming.

V. Azizi is a scientific programmer that loves to work on high performance computing problems and get the best performance possible. He graduated from the University of Amsterdam in 2016 and worked on this project as a scientific programmer.

A.G. Hoekstra holds a PhD in Computational Science from the University of Amsterdam and currently is a professor in Computational Science at the University of Amsterdam and the national research university ITMO, St Petersburg, Russia. His research focuses on multi-scale multi-science modelling, large-scale simulations, and high performance computing, mainly in the biomedical domain and complex systems science. He has a long-standing expertise in Computational Biomedicine, Complex Systems simulations, and high performance parallel and distributed computing. He has published over 250 research papers. He currently leads the Computational Science Lab at the University of Amsterdam.