Project:	HPCN Parallelization of 2D blood simulation code SuMO
Customer:	Institute for Informatics, University of Amsterdam
Contractor:	Electric Ant Lab B.V.
Contract:	Software Development Agreement from 26-03-2015
Date:	28-09-2015
Author:	E. Lorenz, Electric Ant Lab B.V.

PaSuMO – Parallel Suspensions of Membraneous Objects

Abbreviations in the text:
LBM - Lattice-Boltzmann Method, Eulerian Navier-Stokes solver used to describe blood plasma dynamics
IBM - Immersed-Boundary Method coupling Lagrangian discrete-element models to Eulerian fluid representations
LSP - Lagrangian surface points connected through mechanical models representing walls and cell membranes

1. Parallelization

The existing 2D simulation code, SuMO, has been analyzed regarding algorithm and data structures. It is clear that datastructures needed to change substantially for the parallelization of the algorithms.

1.1 Domain decomposition

The typical application case of the simulation code will be blood flow through complex irregular and sparse artery geometries. Different domain decomposition approaches (quad-tree cubic subdivision, rectangular subdivision, graph-partitioning) have been analyzed in respect of their memory usage and load-balancing in such cases. The tight coupling between the Eulerian LBM and the Lagrangian particles and their technical implications has been taken into account here.
Though it made a complete re-implementation of the LBM datastructures in form of sparse linearized arrays with neighborhood lists and the inverse location mapping (index to coordinate) necessary, graph-partitioning using METIS has been chosen. It takes only actual domain elements (LBM lattice nodes) into account and minimizes the size of subdomain interfaces with the fine granularity of actual LBM lattice nodes. Despite the increased complexity, memory footprint is minimized by not allocating memory for space outside the flow domain. METIS has been shown to produce excellent partitionings with minimal interface and good load-balance for related application cases (Axner et al, J. Comput. Phys. 227, 4895, 2008).

1.2 Parallelization paradigm

Since the target machine(s) are clusters such as Lisa (SURFsara) and the number of processes used in one simulation will be in the order of hundreds, pure MPI parallelization has been chosen.

1.3 Parallel datastructures and interface communication

For the parallelization of the datastructures it is necessary to distinguish between LBM fluid subsystem, the LSPs of the membrane model, the tight two-way coupling of LSPs and LBM, and cell objects, each having their own interaction range, and therefore their own communication range. Owing to the domain-decomposition approach, in all three cases, the parallel data structures differ substantially from the serial implementation. Also, the choice of pure MPI as parallelization paradigm makes explicit synchronization and migration of objects at the subdomain interfaces necessary.

Fluid subdomains (LBM)

Structure: LBM subdomains are represented as linearized arrays of the LBM domain (sparse domain decomposition, see above). As an array of structures every node contains the full fluid information at this node contiguously in memory. The LBM subdomains don’t change size, connectivity, or interfaces to other subdomains in time. Memory is allocated only once in the beginning for the computational domain plus the outer interface nodes needed for synchronization with neighbor subdomains. Communication neighbors are identified only once when nodes and interface nodes are read in (output from partitioning). LSPs interact with the fluid through IBM. The IBM interpolation kernel defines the LBM interface width. Best results have been obtained with a kernel width of just one lattice unit in each direction so an interface width of 1 lattice unit is sufficient.

Communication: At every timestep, just before LBM streaming, the inner interfaces are communicated to the appropriate domain neighbor. Since the whole inner and outer interfaces are contiguously aligned in the LBM array, start and end indices are sufficient to specify the part of the LBM array that needs to be sent and the start index on the receiving side. There is no need for buffers and serialization and de-serialization to create and read from them. However, there is room for improvements in this scheme as not all of the communicated LBM variables are actually needed to carry out the streaming into the actual subdomain from the outer interface.

Lagrangian surface points (LSP)

Structure: LSP subdomains contain arrays of LSP variables (points with position, forces, angles, connectivity, ...), again as an array of structures. The number of LSPs per subdomain varies in time, however a maximum number can be estimated from the maximum hematocrit, and memory can be allocated statically accordingly. Dynamic allocation (resizing the array on-the-fly) would cause too much overhead (at least when implemented in Fortran) and would make compact (contiguous) array filling necessary (on-the-fly reordering). In all cases, an indirect addressing method is necessary to translate unique LSP id's to array indices and back. A list of the LSP ID's is kept for this purpose. Received LSPs are registered there and LSPs that left the subdomain are de-registered.
LSPs are also registered as “resident” at each LBM lattice node for which another array of the same size as the LBM lattice but with a list of LSPs as the array element is used. Identification of the “mother node” of an LSP is a simple integer casting of the coordinates and a subsequent lookup of the lattice node index in the 2D array that maps coordinates to 1D-lattice indices. This allows for a quick identification of whether an LSP is inside the “core” domain, or the “inner” or “outer” interface.

Communication: Interaction between LSPs never goes beyond a distance of 1 lattice unit. The list of LSPs that need to be sent to the appropriate neighbor domain is therefore simply constructed by going through the “residents” lists of the “inner” interface nodes. Since the size of this list varies in time, the list size needs to be communicated first. Received LSPs are then added to the subdomain LSP list and registered as “active” or “ghost” depending on whether it's inside the domain or in the “outer” interface whose elements are only used as information source and are not updated inside the subdomain.

Cell subdomains (cells)

Structure: An array of structures contain information about cells (their volume, pressure, etc). While these could be independent entities with handling and communication similar to LSPs, it has been decided to reconstruct cells from LSPs using the connectivity stored in the LSP structure only when it is needed. Use is made of the fact that each LSP stores the ID's of the left and right neighbor in the cell membrane. Also, since cell dynamics (change of volume, area, etc) is much slower than that of LSPs or the fluid, cell reconstruction can be done at a lower frequency, e.g. only every 10 timesteps.

Communication: Cells can be much larger than the LSP interaction range and so some LSPs of a cell on the interface might be missing. During cell reconstruction missing LSPs are requested from the according neighbor domain. Once received, they are added to the list of LSPs as ghosts taking care of possible double entries. This approach reduces communication overhead compared to the introduction of a full cell synchronization envelope. Such an envelope would have at least the size of a maximally stretched cell (which is not straightforward to predict) and therefore would also result in a communication of cells and LSPs that actually don't play a role on the other side of the interface.

Immersed boundary coupling (IBM)

Fluid and cells are coupled through the immersed-boundary method, a tight coupling between fluid and LSP velocities and LSP forces and acceleration of the fluid at the location of an LSP. Mapping between Lagrangian and Eulerian information is through interpolations using a short-range kernel function. LSPs at the interfaces have sufficient lattice information to compute their velocity due to an up-to-date "outer" interface. After forces on LSPs are computed in the membrane model they are spread to the same outer interface nodes necessitating another communication step of forces. For this reason, force fields on the lattice have their own array separate from the LBM array.
Communication: Force arrays have the same size and alignment as the LBM array and nterface force fields are communicated the same way. The difference is that received forces are not overwriting existing information but added to the existing forces due to active LSPs on the receiving side.

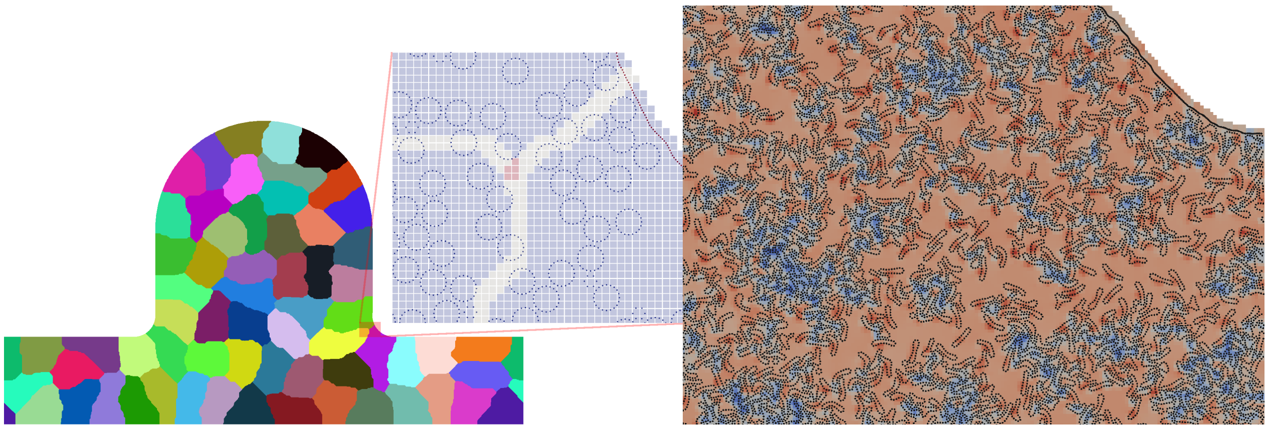

Fig 1: Left: Domain-decomposition of aneurysm geometry into 64 subdomains using METIS. The inset shows the indicated part of the domain with LBM nodes inside domain (light blue), LBM nodes being part of the interface (light gray for two and light red for three domain neighbors), together with boundary LSPs as distributed and smoothed along the walls (red dots) during pre-processing. Also shown are initial positions of cell LSPs. Right: LSPs (black) and pressure field during the equilibration process.

2 Pre-Processing

2.1 Domain decomposition

A simulation setup workflow has been developed that takes a b/w bitmap (in PPM format) filename and the number of subdomains as input arguments for a single call of `pasumo_MkCase`. It creates a partitioning with METIS, identifies inner domain nodes and nodes shared between each pair of subdomains at their interface, as well as domain neighbors and neighbors of each lattice node list.
Enumeration of fluid nodes is such that "inner" nodes (those that don't need to take part in communication) are contiguous in memory. Same applies to all interface nodes, at inner and outer interfaces separately, so that there communication can be achieved through send/receive of a contiguous chunk of memory without the need of serialization/deserialization later in the simulation.

2.2. Flow boundaries

Running `pasumo_MkCase` also adds a layer of boundary nodes to the flow domain and distributes Lagrangian points along the former boundary for IBM-based flow boundary conditions. These point walls are then smoothed on the scale of a few lattice spacings to remove stair-case effects induced by the regular lattice of the LBM flow domain. An example can be seen in the upper right corner of the inset in Fig 1.

2.3 Initial conditions of cells

As a last step, `pasumo_MkCase` distributes cells in the flow domain. Starting from a random coordinate inside the flow domain (using fluid node coordinates as set of possible coordinates), in an MD simulation relaxing the positions so that all cell-cell distances are larger than the diameter of a circle with the same area as a cell. When this state is reached, equidistant LSPs are distributed over the circumference of the still circular cell. Compared to a membrane in mechanical equilibrium, LSP-LSP distances are compressed and strong membrane stretch forces are expected. The simulation starts with ramping to allow for a stable relaxation into the RBC shape.
None of the pre-processing methods have been parallelized so far. In cases of large domains and large number of cells actual runtimes for setting up the simulation might be substantial, taking a few hours in the largest cases tested so far (approx. 13*10^6 fluid nodes and 5*10^5 cells each with 27 LSPs).

2.2 Geoms, Cases, and Runs

Domain-decomposition and initial conditions are not unique to every simulation. Other flow conditions such as a pressure gradient could be applied to the same system. The configuration of a simulation is therefore separated in three steps. Geometries are stored as ppm bitmaps, cases are initial conditions using a geometry, and runs are actual simulations with a configuration pointing to a case.

3 Parallel performance – Strong scaling

The model aneurysm geometry in Fig 1 with an hematocrit of 45% has been used to profile the performance of the subparts of the simulation over a wide range of numbers of processors (nproc). The system contains approximately 825*10^3 fluid nodes and 31.5*10^3 cells consisting of more than 604*10^3 LSPs in total. Larger problems could have been chosen but would have been problematic regarding memory and compute time for the case nproc=2.

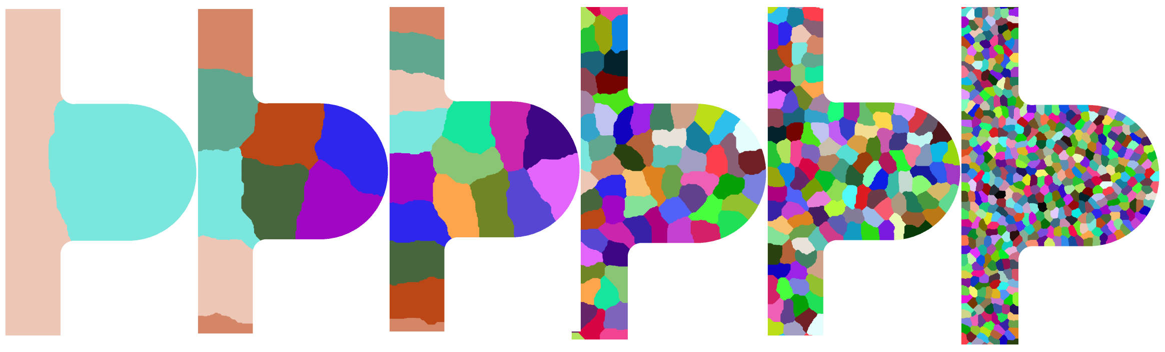
Fig 2: Example decompositions of the model aneurysm with a channel width of 250mum and a lattice resolution of 1mum for nproc=2,8,16,64,128,512.

METIS produces very satisfactory domain decompositions. Subdomain sizes do not exceed deviations larger than 2% from the mean subdomain size. Number of subdomain neighbors and interface length also have a narrow distribution (boundary subdomains being a natural exception).

Fig 3 shows timing of the simulation runs using a number of processors ranging from nproc=2 to 256 run on lisa, the cluster at SURFsara. 'dyn' denote times spent in pure computation routines, 'comm mpi' stands for time spent on actual MPI calls, and 'comm ovh' denote times needed for serialization/de-serialization and related buffer-construction routines.

In Fig 3a the time spent in routines of the subparts of the simulation are plotted over the number of processors. We see that the total runtime (tot) does scale with ~1/nproc up to nproc=256. With 512 cores total communication time (tot comm) does increase and brings an end to this scaling, as expected for strong scaling. At this point the ratio of effective interface width (LBM, LSP, IBM, cell subsystems combined) to computing (any dynamics inside the subdomain) becomes significant. The linear scaling continuous up to nproc=256 where subdomains have become relatively small already (~ 3200 compute nodes, ~280 interface nodes, ~ 123 cells). Communication overhead (tot comm ovh, mainly serialization/de-serialization) does scale sublinear beyond nproc=256. Total computing (tot dyn) and total handling (tot handl, helper functions) also continue to scale linear beyond this point. The total runtime (total) includes the time for a single output of data at timestep 100. As long as no parallel filesystem is used output does not scale and so a leveling off of the runtime can be observed at nproc=64. In a realistic simulation, however, data would be written much less frequently (approx at every 10^4 timesteps) so this is effect would show only at much larger nproc.

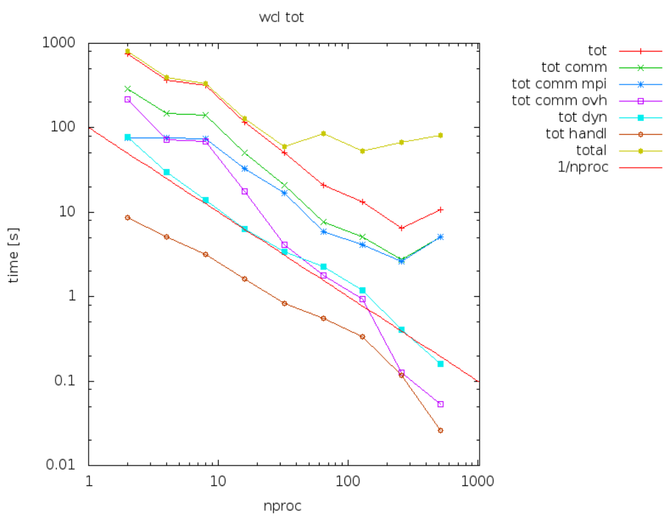

Fig 3a: Scaling of total runtime and time spent in computation and communication routines. The 'tot' data represents a sum of all sub-module measurements. The 'total' dataset includes a single output of data at the final timestep.

In Fig 3b the communication times are plotted. It can be seen that for all subsystems (LBM, LSP, IBM, cells) the sub-linear decrease and final increase in total communication time is caused by the time MPI needs to communicate the buffers. The ratio of interface size to domain size grows with nproc but also the ratio of latency to sending time becomes larger.
Overhead times (ovh) are very small and almost constant for all subsystems except LSP. This is due to the

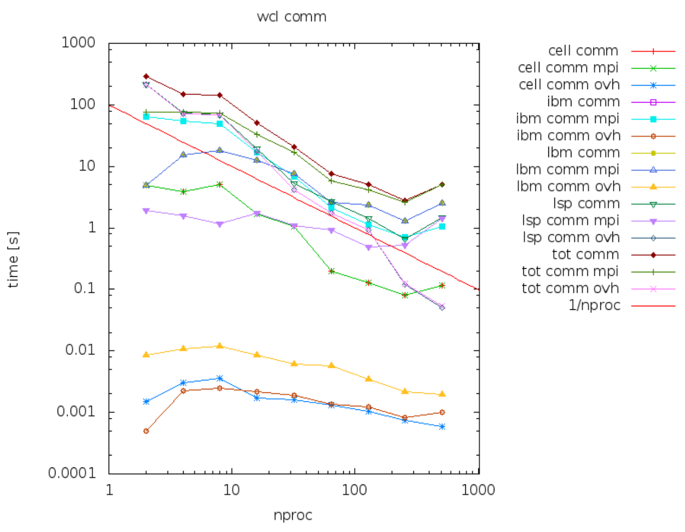

Fig 3b: Times spent in communication routines, pure MPI communication calls, and serialization/de-serialization routines ('ovh').

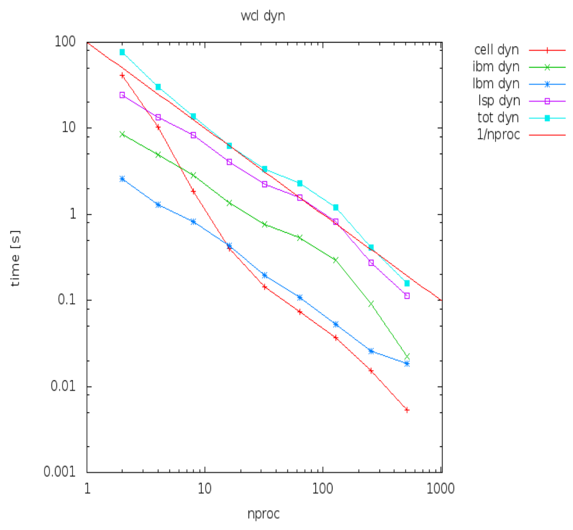

Fig 3c: Scaling of the times spent in computation routines for each of the subsystems.

serialization, de-serialization of buffers of variable length (extra length communication step) and the additional registration of the LSPs on the receiving side. All these processes scale linearly with the number of LSPs over the whole range of nproc. For the LSP subsystem actual MPI time stays almost constant ans is smaller than the LSP communication overhead for all nproc<256.

In Fig 3c the scaling of times spent in computing routines are shown for the different subsystems. All scale linearly up to the maximum number of processors used. The cell subsystem shows an even more drastic decrease of the computation time possibly due to a part of it scaling worse than linearly with the number of entities in the subdomain.

For a hematocrit of 45% the parallel code achieves a speed of approximately 0.18 MLUPS (mega lattice-site updates per second) per core. With the serial version, SuMO, the same problem can be computed in less than half the time at approximately 0.5 MLUPS. However, when communication overhead is not taken into account, PaSuMO's speed can be estimated at 0.6 MLUPS per core providing an honest computation speed comparison. The slightly larger speed means that the many optimizations applied to datastructures and dynamics in PaSuMO also improved the pure computation speed.

As can be seen for similar parallel simulations, runtimes are dominated by the communication overhead. As seen in all Fig 3, this communication overhead scales linearly and can therefore be efficiently compensated by using a larger number of cores. Using the speeds mentioned above, the amount of processors needed for PaSuMO to compute a given problem faster than using SuMO can be estimated as nproc>2.7. As pointed out earlier, the amount of data that is communicated can be further reduced by filtering the structures that are sent. In most cases, however, this would make additional pre and post-communication processing and extra communication steps necessary that will introduce more overhead and/or latency. Also, asynchronous/non-blocking communication has been implemented where possible but tricks to hide more communication behind computation are thinkable.

4 Summary

In this project the existing 2D blood simulation code SuMO has been successfully parallelized using MPI allowing to efficiently run simulations using hundreds of compute cores. Domain decomposition has been achieved using the graph-partitioning tool METIS resulting in very good load balancing (equal domain sizes) and minimization of the interfaces between subdomains at which information has to be exchanged. In this approach no memory is allocated for space outside the flow domain, however, a so-called sparse-geometry implementation involving a 1D-array representation of LBM system including indirect addressing had to be realized.
In the resulting code, PaSuMO, the datastructures of all four subsystems (LBM, LSP, IBM, cells) have been redesigned to be able to optimize the communication at the interfaces and handle variable number of objects in a subdomain. For lattice systems (the LBM fluid and IBM force fields) this resulted in a scheme in which serialization/de-serialization and buffers were not necessary. For the Lagrangian subsystems (LSPs and cells) the variable number of entities in a subdomain made the implementation of enhanced indirect addressing schemes necessary.
All models implemented present in SuMO have been successfully re-introduced into the new code, often in optimized form, including LSP-based boundary treatment, and validated in a number of parallel simulations.
A parallel scaling analysis has been carried out for strong scaling using a model aneurysm geometry containing 825*10^3 fluid nodes, 31.5*10^3 cells, and 604*10^3 LSPs representing a realistic use case. Linear scaling of the total runtime with the inverse number of processors has been shown up to nproc=256 demonstrating the efficiency of the implementation. The scaling of the computation, MPI communication, and communication overhead has been analyzed and discussed.
All domain-decomposition and pre-processing steps have been unified in a single tool. Configuration of a simulation is designed independently for cases and runs facilitating efficient reuse of initial conditions. All configurations are done via clear human-readable text files.

5 Code repository

The code has been pushed to a UvA git repository at
`git@hex.science.uva.nl:elorenz/pasumo.git` and access has been provided to users and developers of the Computational Science Lab at UvA. A backup is kept at an internal repository at Electric Ant Lab B.V..