Project Title	SCAlable LAttice Boltzmann Leaps to Exascale
Project Acronym	SCALABLE
Grant Agreement No.	956000
Start Date of Project	01.01.2021
Duration of Project	36 Months
Project Website	www.scalable-hpc.eu

D3.5 – Adaptation of pre-processing

Work Package	**WP 3.5, Adaption of pre-processing**
Lead Author (Org)	**Raphael Kuate (CSGROUP)**
Contributing Author(s) (Org)	
Reviewed by	**Romain Cuidard (CSGROUP)**
Approved by	**Management Board**
Due Date	**01.07.2022**
Date	**29.08.2022**
Version	**V1.0**

Dissemination Level

X	PU: Public
	PP: Restricted to other programme participants (including the Commission)
	RE: Restricted to a group specified by the consortium (including the Commission)
	CO: Confidential, only for members of the consortium (including the Commission)

Versioning and contribution history

Version	Date	Author	Notes
0.1	25.07.2022	**Raphael Kuate (CSGROUP)**	Reviewed by Romain Cuidard
1.0	29.08.2022	Corentin Lefevre (Neovia)	Minor editions – Version approved by the MB

Table of Contents

List of Figures

Executive Summary

The main objective of SCALABLE for CS GROUP is the improvement of LaBS deployment in bigger clusters of thousands of cores, achieved by a transfer of performance technology from WaLBerla. In the earlier work package 3.1, we have investigated the choice of an appropriate data structure organization for LaBS. We have shown that neither structured nor mixed structured/unstructured data organization were significantly more efficient compared to the current LaBS unstructured data design on industrial applications involving complex simulations. Therefore, this work package 3.5 focusses on adapting the pre-process step of LaBS for a solver step targeting GPU processors.

1 Introduction

1.1 Context

Lattice Boltzmann methods (LBM) are nowadays trustworthy alternatives to conventional CFD methods, since it has been already shown in several engineering applications that they are faster than Navier-Stokes approaches in comparable scenarios. LBMs can handle complex geometries and a wide range of multiphysics applications that are of high industrial relevance. The main distinguishing feature of the LBM is its algorithmic locality stemming from an explicit time step. Thus, the LBM is especially well-suited to exploit advanced supercomputer architectures through vectorization, accelerators, and massive parallelization.
WaLBerla is one of the most advanced LBM research codes in the public domain. Its superb performance and unlimited scalability have been proven, reaching more than a trillion lattice cells already on Peta-scale systems. WaLBerla performance excels in academic use cases because of its carefully designed implicit blocks data structures. However, waLBerla is not compliant with industrial applications due to lack of a complex geometry engine and user-friendliness for non-HPC experts.
The CFD software LaBS is an industrial LBM code with capabilities at a proven high level of maturity, but with high scalability performance in improvement. Therefore, in the context of EuroHPC, SCALABLE will transfer the performance technology from waLBerla to LaBS. This collaboration will deliver improved scalability for LaBS as preparation for the upcoming European Exascale systems.

Commenté [A1]: Do our walberla colleagues agree with this assessment?

Commenté [A2R1]: I hope so. This part of the context was already stated in an earlier report.

1.2 Objective

The main objective of SCALABLE for CS GROUP is the improvement of LaBS deployment in bigger clusters of thousands of cores. The current usage context of LaBS, depending on the use case, is around a thousand of CPU cores. However, LBM simulations can also be suitable on GPUs clusters, since their level of regular parallelism is very high, as well as their memory bandwidth. Therefore, we have investigated on LaBS evolutions for the handling of GPUs clusters.
The remainder of this document is organized as follows. In the next section, we present the memory footprint improvements implemented and tested on the D3Q19HRR scheme for single and double precision computations. The last section before conclusion present results of the pre-processing step launched on CPU cores targeting solver step on GPU cores.

2 Memory footprint

2.1 *Analysis*

In complex industrial simulations, one may expect that the critical memory consumption happens during the solver step, since its elapsed time is about the whole computation time, pre-processing and post-processing steps being of lesser importance. In some cases, we have observed that the pre-processing step of LaBS may have a bigger memory footprint than the solver step. In LaBS, the unstructured cell design of data is combined with an organization into *families* of cells of which the same LBM computation functions are applied. Many *families* with the same properties can be grouped into *tribes*. The storage of the variables computed is done at the *tribe* level. The way that data is chunked into *tribes* results in a compromising objective of solver speed and memory fragmentation and was designed for CPU clusters. However, with the target objective of deploying LaBS on GPU clusters, we have investigated these points together with the domain decomposition sub-step of the pre-processing, and it happens that during the migration of notes in the load-balancing sub-step, the memory footprint sometimes reaches its critical value.

Commenté [A3]: I don't get that

Commenté [A4R3]: elapsed time in solver step is about the whole elapsed time of the simulation

2.2 *Implementation*

Without describing technical aspects of all the pre-processing steps of LaBS and its memory fragmentation, we have implemented and tested an optimization of the memory footprint of LaBS on a D3Q19HRR scheme. These optimizations can be divided in two parts.

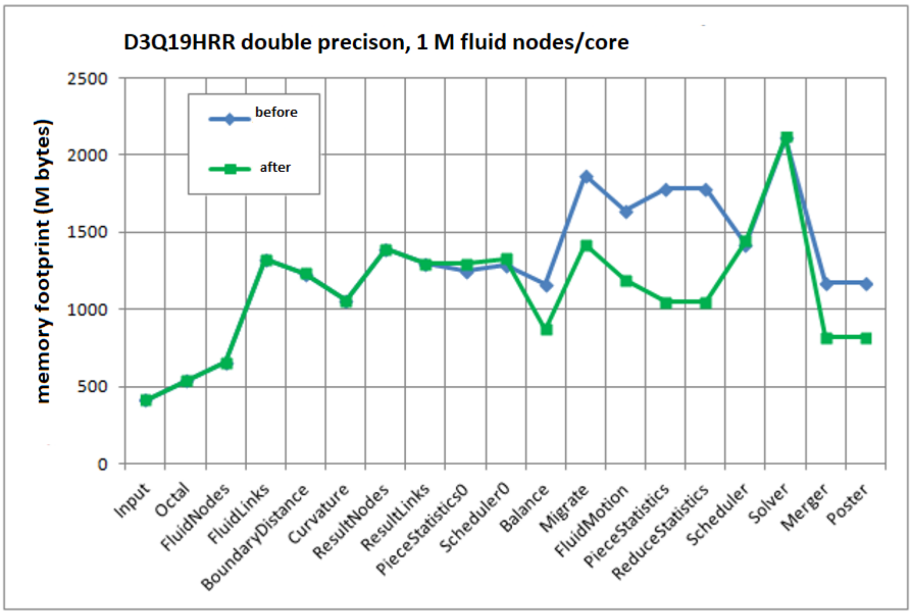
2.2.1 *Optimization of the migrate step*

The main problem in this pre-processing sub step is the using of a double data structure (AoS and SoA) *links* for accessing data from different parts of the code. As a first optimization, new links are serialized and recorded in temporary files, the old *links* are destroyed, and the new ones are reloaded from the temporary files.

2.2.2 *Memory defragmentation*

At the end of pre-processing sub-steps *Scheduler0, Scheduler, Migrate*, the data structures needed for the sub-steps *Surfaces, Cut, Piece, Schedule* are serialized and written to temporary files, the memory is cleared, removing chunks created by data unused for the next sub-steps. Data structure of the next sub-steps are then reloaded from temporary files.

The following pictures show some improvements on memory footprint particularly on the pre-processing step which includes all steps listed in the picture except *solver*, *merger* and *poster*.



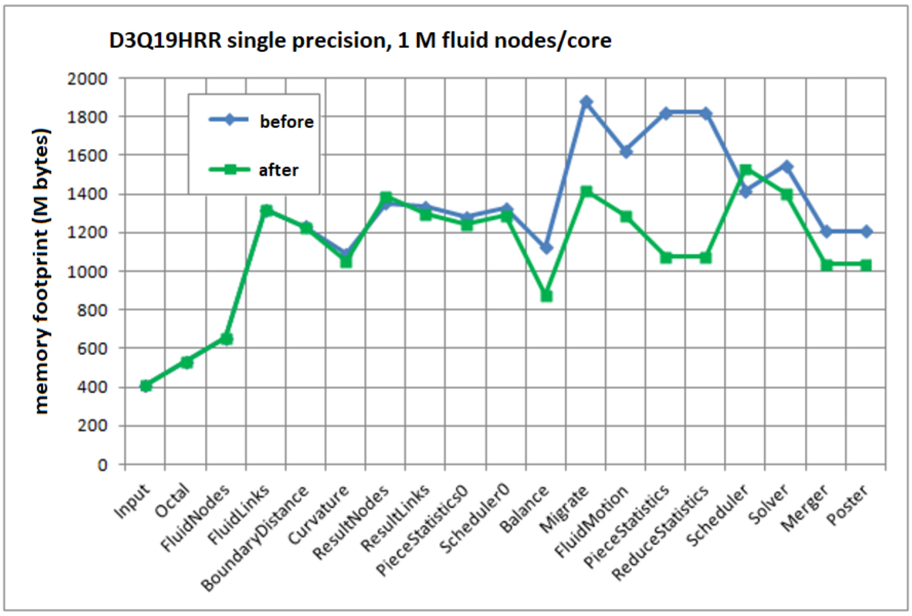


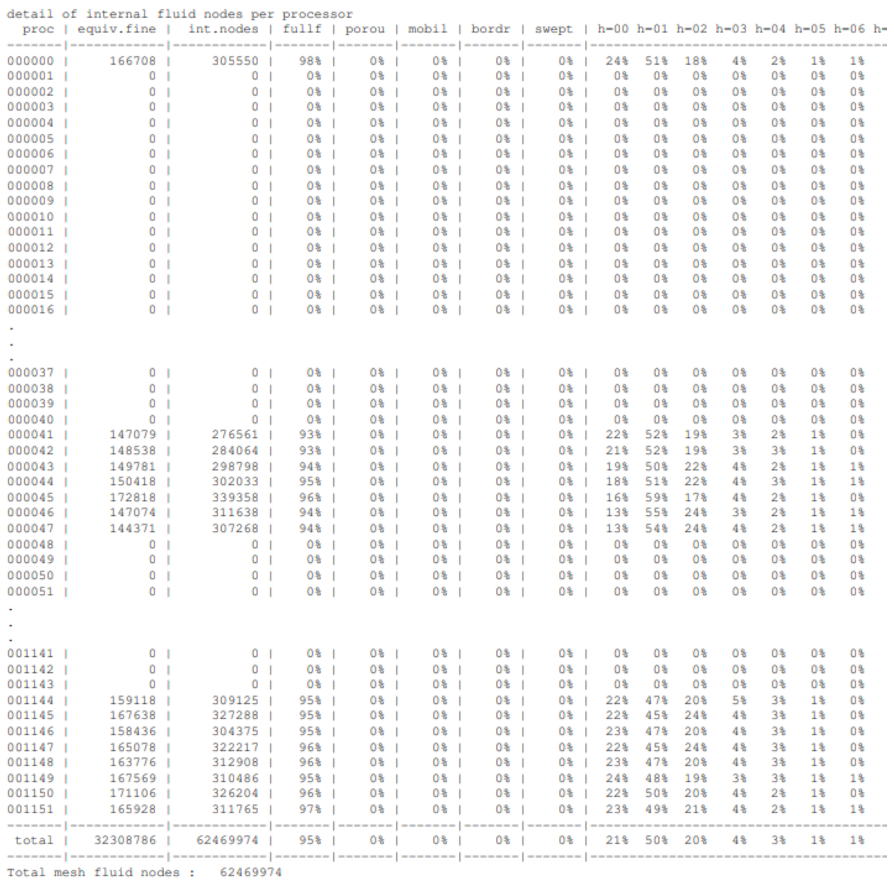
Figure 1- Memory footprint improvements. Scheme D3Q19HRR with double precision (top picture) and single precision (bottom picture) computations.

3 Load balancing for solver step targeting GPU cores

The objective of launching LaBS on GPU clusters needs some adaptions of the pre-processing, since only arithmetical calculations – and consequently the solver step – are most suitable for GPU cores. Considering a cluster with P GPU cores and N CPU cores, one may want to use the entire N CPU cores for pre-processing, the results of which must target the P GPU cores in the solver step. We have adapted the load balancing such that the pe-processing step can be launched on N CPU core for a solver step targeting P GPU cores, N > P.

3.1 Algorithm summary

The number P of GPU cores needed is read as argument of the LaBS executable and recorded in the *detect* sub-step of the pre-processing, it may also be automatically detected in the future GPU version of LaBS. In the *detect* sub-step, the CPU IDs of processes within each cluster node are also recorded. The different sub-steps of the pre-processing are executed as usually on N CPU cores until the *balance* sub-step. At this *balance* sub-step, instead of load-balancing among the N CPU cores as usual, the data is rebalanced among P CPU cores, each of which mapping exactly the number of GPU cores of each cluster node, emptying the remaining CPU IDs (not mapped with any GPU core) of each cluster node recorded at detect step.



```
detail of internal fluid nodes per processor
  proc | equiv.fine |  int.nodes | fullf | porou | mobil | bordr | swept | h-00 h-01 h-02 h-03 h-04 h-05 h-06 h-
-------|------------|------------|-------|-------|-------|-------|-------|-------------------------------------
000000 |     166708 |     305550 |   98% |    0% |    0% |    0% |    0% |  24%  51%  18%   4%   2%   1%   1%
000001 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000002 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000003 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000004 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000005 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000006 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000007 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000008 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000009 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000010 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000011 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000012 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000013 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000014 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000015 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000016 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
.
.
.
000037 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000038 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000039 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000040 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000041 |     147079 |     276561 |   93% |    0% |    0% |    0% |    0% |  22%  52%  19%   3%   2%   1%   0%
000042 |     148538 |     284064 |   93% |    0% |    0% |    0% |    0% |  21%  52%  19%   3%   3%   1%   0%
000043 |     149781 |     298798 |   94% |    0% |    0% |    0% |    0% |  19%  50%  22%   4%   2%   1%   1%
000044 |     150418 |     302033 |   95% |    0% |    0% |    0% |    0% |  18%  51%  22%   4%   3%   1%   1%
000045 |     172818 |     339358 |   96% |    0% |    0% |    0% |    0% |  16%  59%  17%   4%   2%   1%   0%
000046 |     147074 |     311638 |   94% |    0% |    0% |    0% |    0% |  13%  55%  24%   3%   2%   1%   1%
000047 |     144371 |     307268 |   94% |    0% |    0% |    0% |    0% |  13%  54%  24%   4%   2%   1%   1%
000048 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000049 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000050 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
000051 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
.
.
.
001141 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
001142 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
001143 |          0 |          0 |    0% |    0% |    0% |    0% |    0% |   0%   0%   0%   0%   0%   0%   0%
001144 |     159118 |     309125 |   95% |    0% |    0% |    0% |    0% |  22%  47%  20%   5%   3%   1%   0%
001145 |     167638 |     327288 |   95% |    0% |    0% |    0% |    0% |  22%  45%  24%   4%   3%   1%   0%
001146 |     158436 |     304375 |   95% |    0% |    0% |    0% |    0% |  23%  47%  20%   4%   3%   1%   0%
001147 |     165078 |     322217 |   96% |    0% |    0% |    0% |    0% |  22%  45%  24%   4%   3%   1%   0%
001148 |     163776 |     312908 |   96% |    0% |    0% |    0% |    0% |  23%  47%  20%   4%   3%   1%   0%
001149 |     167569 |     310486 |   95% |    0% |    0% |    0% |    0% |  24%  48%  19%   3%   3%   1%   1%
001150 |     171106 |     326204 |   96% |    0% |    0% |    0% |    0% |  22%  50%  20%   4%   2%   1%   0%
001151 |     165928 |     311765 |   97% |    0% |    0% |    0% |    0% |  23%  49%  21%   4%   2%   1%   1%
-------|------------|------------|-------|-------|-------|-------|-------|-------------------------------------
 total |   32308786 |   62469974 |   95% |    0% |    0% |    0% |    0% |  21%  50%  20%   4%   3%   1%   1%
-------|------------|------------|-------|-------|-------|-------|-------|-------------------------------------
Total mesh fluid nodes :   62469974
```

The part of the LaBS log file above shows how nodes are distributed only among the target GPU P processes. The figure above shows a comparison of the elapsed time of a run with full N cores versus a run with N → P cores, i.e., N cores in pre-processing steps and P cores in solver step. One can observe that the elapsed time for the N → P configuration decreases as P decreases, since the size of external nodes (communication halo) growths with the number of sub-domains of the domain decomposition.

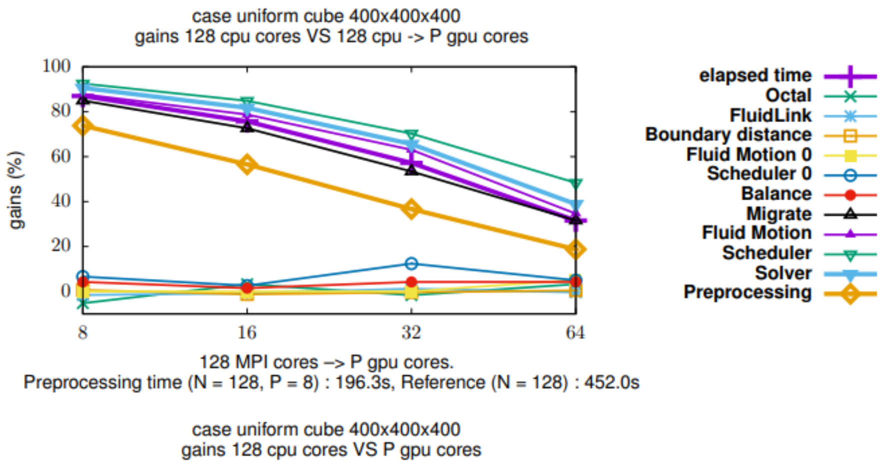


Figure 2- Load balancing on N CPU cores targeting P GPU cores. Only the steps launched after *Balance* have their elapsed time reduced in a configuration with N CPU cores → P GPU cores compared to a whole N CPU cores configuration, since data exchange is reduced, external nodes (communication halo) decreasing as well as P decreases, compared to N.

4 Conclusion and perspectives

We have presented initial improvements of the pre-processing steps of LaBS for the solver step targeting GPU cores, in order to deploy LaBS on GPU clusters. As perspectives, the improvement of the **solver step memory footprint** and the **implementation of a GPU version of the solver step** are the next work for the achievement of the current developments.

5 Bibliography

[1] F. Schornbaum, "Block-Structured Adaptive Mesh Refinement for Simulations on Extreme-Scale Supercomputers," 2018.

[2] C. Feichtinger, S. Donath, H. Köstler, J. Götz and U. Rüde, "WaLBerla: HPC software design for computational engineering simulations," *Journal of Computational Science,* vol. 2, pp. 105-112, 2011.

[3] C. Godenschwager, F. Schornbaum, M. Bauer, H. Köstler and U. Rüde, "A Framework for Hybrid Parallel Flow Simulations with a Trillion Cells in Complex Geometries," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, New York, NY, USA, 2013.

[4] F. Schornbaum and U. Rüde, "Massively Parallel Algorithms for the Lattice Boltzmann Method on NonUniform Grids," *SIAM J. Sci. Comput.,* vol. 38, 2016.