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

D3.2 – Development of appropriate scheduling

Work Package	**WP 3.2, Development of appropriate scheduling**
Lead Author (Org)	**Raphael Kuate (CSGROUP)**
Contributing Author(s) (Org)	
Reviewed by	**Romain Cuidard (CSGROUP)** **Julien Billière (CSGROUP)**
Approved by	**Management Board (MB)**
Due Date	**01.01.2023**
Date	**02.02.2023**
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	18.11.2022	Raphael Kuate (CSGROUP)	TOC and V0.1
0.9	19.12.2022	Romain Cuidard (CSGROUP)	Review
1.0	02/02/2023	Corentin Lefevre (Neovia)	Version approved by the MB

Table of Contents

List of Figures

Executive Summary

The main objective of SCALABLE for CSGROUP is the improvement of LaBS deployment in bigger clusters of thousands of cores, achieved by a transfer of performance technology from WaLBerla. Therefore, in the earlier work package 3.1, we investigated the choice of an appropriate data structure organization for LABS. In this work package 3.2 and following the earlier work, we focus on the choice of an appropriate scheduling for LABS. We have implemented a "*delaying"* of some computations and obtained significant gains on overall simulation elapsed time. We have also examined the possibility of scheduling LABS simulation steps in terms of independently organized families and show that it would not improve significantly the simulation cost. In the other hand, the observations on the improvement of LABS amount of ***pdf*** (particle distribution functions) data exchanged examined in the earlier work package have been generalized and thus implemented at the Scheduling level.

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. LBM methods 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 proved, 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 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 to be prepared for the upcoming European Exascale systems.

1.2 Objective

The main objective of SCALABLE for CSGROUP is the improvement of LaBS scalability, thus its deployment in bigger clusters of thousands of cores. Among massive parallelization techniques, GPU clusters have nowadays become more efficient and reliable for industrial applications than a decade ago. LaBS future developments will thus target GPU clusters architectures as well as is does for CPU clusters. However, in order to maintain as much as possible a single code for both CPU and GPU, we will investigate solutions provided by the

C++17 standard parallelism combined with the NVC++ compiler, a compiler able to provide GPU and CPU executables from the same C++ source code.

The current usage context of LaBS, depending on the use case, is below a thousand cores. In the current document, we will present LaBS Scheduling strategy in terms of simulation steps. Compared to WaLBerla, we have experienced the scheduling of simulation steps using independent data families as WaLBerla block data structure, without significant speedup. However, we have introduced a "*delaying*" of some computations, in order to reduce the processes synchronizations. We have also extended the reduction of amount of data exchanged in LaBS during LBM data exchange step at scheduler level.
The remainder of this document is organized as follows. In the next section, we present the "delaying" of simulations steps, followed by tests of a scheduling strategy in terms of LaBS independent *families,* as far as WaLBerla operates on blocks of topologically independent data. The last subsection before the concluding section is devoted to the generalization of exchanged data reduction in LaBS at scheduling level.

2 Scheduling of simulation steps

2.1 Scheduling strategy

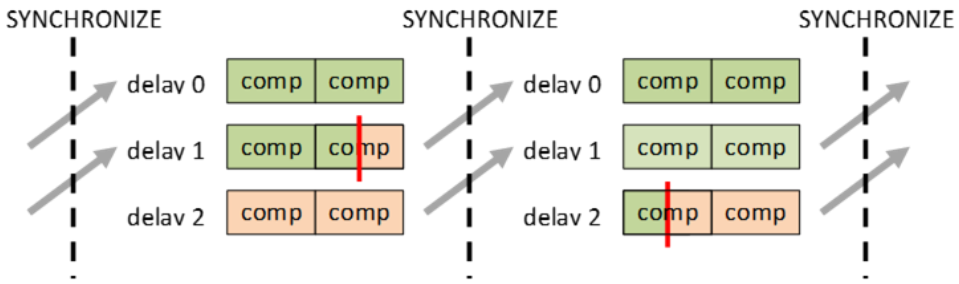
The data structure in WaLBerla[1] employs structured blocks of cells of fixed size in each dimension of the space [1] [2] [3] [4]. The whole simulation strategy in WaLBerla is based on these blocks i.e., all variables being computed are localized in blocks. LBM computations steps are performed independently on each block, regardless of their locality (distant or local process), completed with LBM data exchange step between neighboring blocks.
In LaBS, the unstructured cell design of data is combined with an organization into *families* of cells of which the same LBM computations 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. Since the neighborhood of a cell is needed for LBM computations, the neighbor of any cell is acceded in LaBS using *links.* Within each process, the LBM computation functions are then divided into two main types: functions using *links mono* i.e*., links* within the same *tribe,* and functions using *link multi* i.e*., links* across distinct *tribes* of the same process.
WaLBerla LBM functions and LaBS *link mono* LBM functions are equivalent in terms of data storage access since all computations in LaBS using *link mono* functions do not require any access to data of other *tribes* of the same process.

2.1.1 Delay on simulation steps

One of the main bottom necks in massive parallel simulations with data exchange at any time step, is the synchronization of calculations among involved processes. Thus, in LaBS the wait time due to unavailable data on some processes when these are needed by some other processes can increase sensitively the whole elapsed time of simulations on complex cases. We then analyzed in detail the set of computations performed within each time step, their order of magnitude in terms of duration and their usefulness by other processes. The

[1] https://www.walberla.net

scheduling of computations has been reordered according to *delays* and sub-synchronizations introduced within previous synchronizations at each time step.



The picture above shows how computations sorted w.r.t. delays are launched and synchronized.

Test case	Nb core	Fluid node/cores	compute	Send/recv	wait	total
S2A	64	0,7M	2%	27%	55%	21%
S2A long	64	0,7M	2%	29%	48%	19%
J84	160	0,9M	4%	47%	41%	20%
J84	288	0,5M	-4%	19%	71%	44%
J84 long	480	0,5M	2%	19%	40%	19%

LaBS simulations gains using delay on tests cases S2A and J84. Long simulations use two times more timesteps

The table above shows gains obtained using *delays* on computations. The computations elapsed time (compute) does not vary significantly. However, gains in data transfer (send/recv), wait time and overall simulation time (total) decrease sensitively such that on industrial tests cases, one has about 20% less elapsed time overall.

2.1.2 Tests of independent families

The data storage is AOS (array of structure) oriented by default in WaLBerla. In the other hand, the SOA (structure Of Array) is used by default in LaBS. Thus, we have implemented an AOS version of LaBS for more investigations.

We have estimated the simulation cost in LaBS as if all computations were done using *link mono* functions i.e., as if any *family* were structured such that its nodes have no neighborhood dependencies with other *families* of the same process. For this purpose, we compute the difference per node between *link mono* version and *link multi* version of the any function. The later computation is performed w.r.t. nodes of the same level, considering exactly all nodes really used by the function, but not all nodes of the simulation. These estimations neglect the LBM data exchange step between *families* of the same process. We use tests cases Lagoon and S2A.

2.1.2.1 Simulation parameters

- Lagoon case
  - number of internal nodes: 61 465 328
  - number of equivalent finest nodes: 43 523 886
  - number of levels of refinement: 10
  - *Tribes* of type *mono*: 86%
- S2A case
  - number of internal nodes: 62 469 974
  - number of equivalent finest nodes: 32 308 786
  - number of levels of refinement: 10
  - *Tribes* of type *mono*: 93%

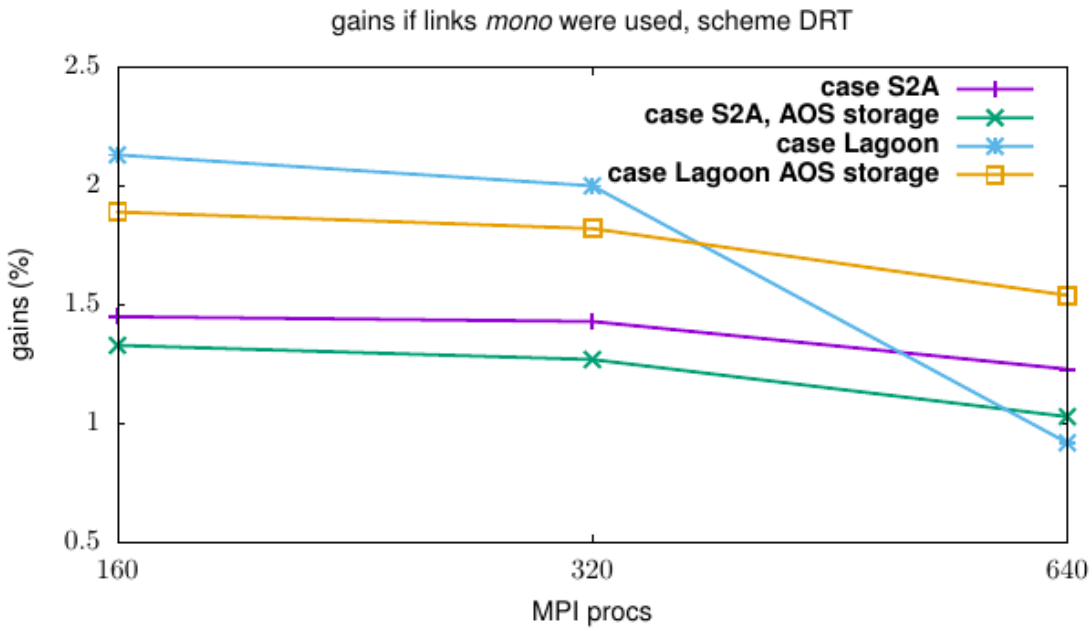


Figure 1 – LaBS simulation time for tests cases Lagoon and S2A within SOA and AOS storage. Estimated gains for independent families i.e., if only links mono were used.

2.1.2.2 Comparisons

We have observed that the estimated gains in terms of simulation time lacks scalability, since it decreases when the number of processes growths. This behavior results of the decreasing size of *link multi* per process when the number of processes growths. Even if these estimated gains neglect the LBM data exchange step between *families* of the same process, one can observe that the gains are weak. The main explanation lives in the weight of *tribes of type mono* per process in LaBS domain decomposition: 93% for S2A test case and 86% for Lagoon test case. The reason of which one has betters gains with Lagoon test case, since its *tribes* of type *multi* is more significant. We have also observed minor differences in LaBS between AOS and SOA data storage.

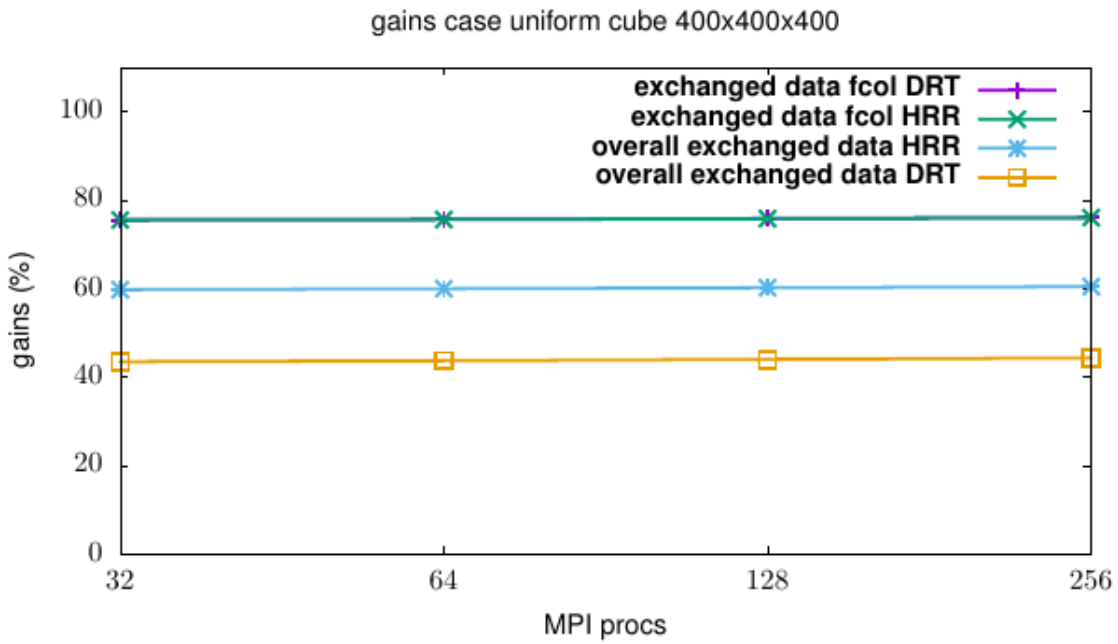
From these weak estimated gains, we conclude that having topological independent *families* would not improve LaBS scalability on industrial simulations, unless the weight of *tribes* of

type *multi* become more important than those of type *mono* in domain decomposition, the latter case has not yet been observed in industrial applications.

2.1.3 Scheduling of data exchange

In the earlier report Wp 3.1, we have presented tests of LaBS data exchange amount reduction on uniform tests cases for the main LaBS ***pdf*** data component “fcoll” for the DRT scheme. We have extended this data exchange amount reduction to all LaBS schemes: DRT, DRTTherm (Thermic DRT) and HRR. This generalization is also applied to all suitable pdf data components: “fcoll” and “fcolR” for all schemes, and to its thermic equivalent "hcoll" and “hcolR” for scheme DRTTherm. The suitable ***pdf*** data are those which are not involved in chimera interpolation. However, nodes subject to transitions, rotating domain interface, sponge or porous zones are also excluded from data exchange amount reduction. In laBS, the whole data exchange mechanism is registered at scheduling level, on *tribes* of nodes with neighbors belonging to distant processes. So, the scheduling of tribes of nodes involved in data exchange among processes have been remodeled accordingly.

The tests configuration uses the same simulation parameters for cases Lagoon and S2A, as in above tests.



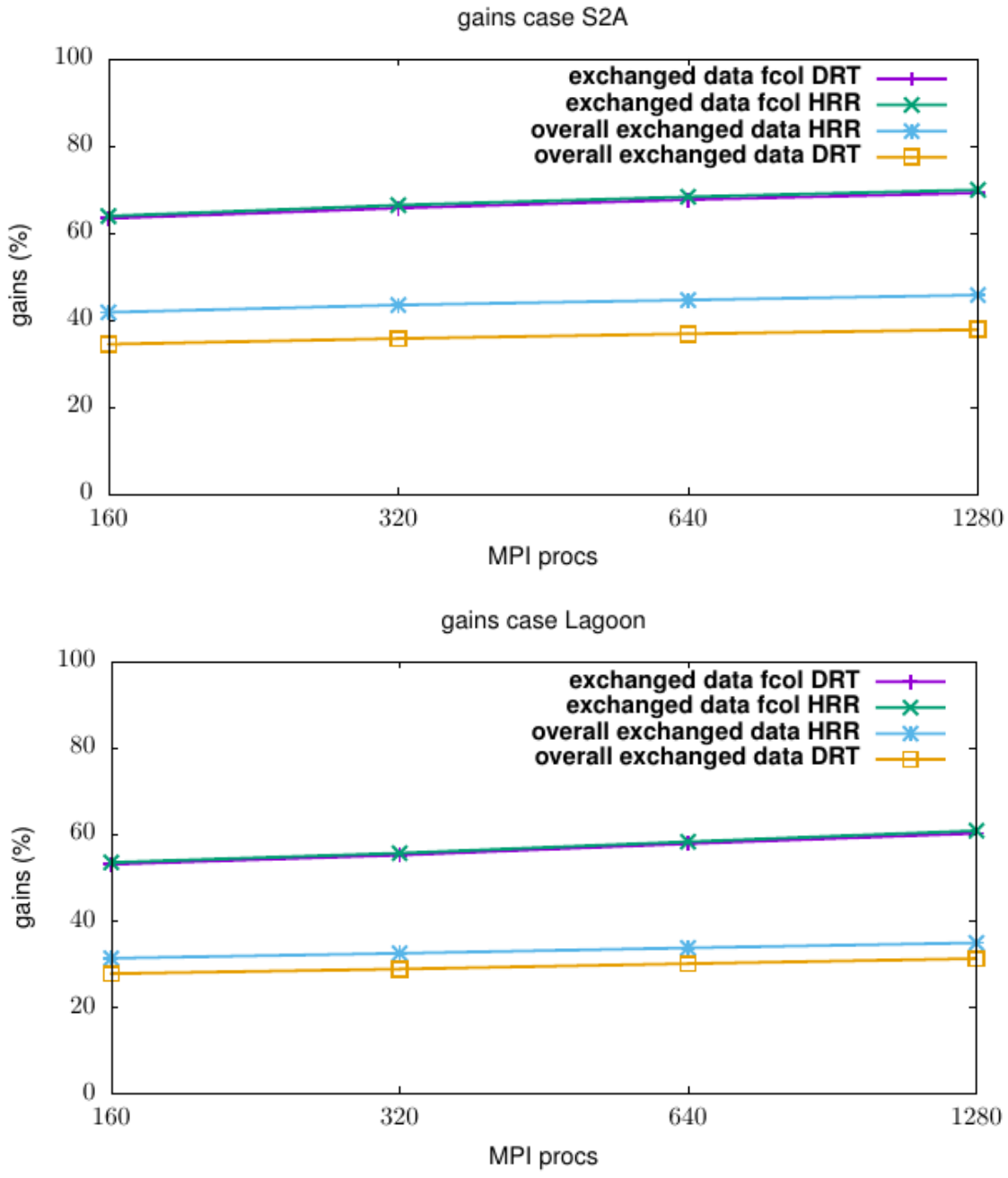


Figure 2 – Exchanged data amount gains on different geometries. Top picture: uniform mesh, middle picture: S2A case, bottom picture: Lagoon case. The schemes DRT and HRR are tested. Top curves: gains over pdf components involved in data reduction, remaining curves: gains over all exchanged data.

2.1.3.1 Comparisons

The top curves of the pictures above show that gains over the data components of which the reduction is applied is about more than 50%, up to 80% on uniform tests cases. Since LaBS solves more complex physics, many other data than ***pdf*** components are exchanged. The overall gains on the total amount of data exchanged is thus lesser but remains important, for about more than 25% and up to 50% or more, depending on the geometry. The results of these tests also confirm that the HRR scheme is less memory consuming than DRT, since it uses less non ***pdf*** data than DRT scheme.

3 Conclusion and perspectives

The amount of exchanged data is an aspect of LaBS elapsed time spent in wait when synchronizing processes. The other aspect of the wait time is the subject of the next task:

- **Wp 3.3: Development of appropriate load-balancing**

We have implemented *delays* on computations and obtained sensitive elapsed time gains on overall simulations. We have also generalized in LaBS the reduction of amount of data during LBM data exchange step and point out the neglectable advantage of having independent blocks of data or AOS data storage as implemented in WaLBerla. This reduction of amount of data exchanged combined with the next task on the development of appropriate load-balancing could improve much more the LaBS elapsed time spent in wait during process synchronization, which is one of the dead neck of LaBS scalability in thousands of processes configuration.

4 Bibliography

[1] 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.

[2] 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.

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

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