Advanced Modeling & Simulation Seminar Series
NASA Ames Research Center
May 20, 2021

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

C++ Parallel Algorithms for GPU Programming
A Case Study with the Lattice Boltzmann Method

Jonas Latt *– University of Geneva*

Christophe Coreixas – University of Geneva

Outline

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

- C++ Parallel algorithms and GPU programming
  - Since C++17, many algorithms can be parallelized.
  - Different backends, different forms of parallelism. We focus on massively multi-threaded platforms.
  - NVIDIA HPC SDK (`nvc++`) offers the `stdpar` backend for GPUs.
- Applying this formalism to scientific problems
  - Many STL algorithms are available but not for all types of problems.
  - Stencil operations ? Non-local memory accesses ?
  - Idea: combine `for_each` style algorithms with memory-access operations (pointer arithmetics).
- Computational fluid dynamics: a lattice Boltzmann implementation
  - The concept is applied to a lattice Boltzmann code.
  - Complex simulation problems solved with state-of-the-art performance.

Part I

C++ PARALLEL ALGORITHMS AND GPUS

C++ Parallel Algorithms

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

```
vector<double> v = { 1, 2, 3, 4 }, w(4);
transform( execution::par, begin(v), end(v), begin(w),
           [](double const& element) { return 2. * element; } );
// w is {2, 4, 6, 8}
```

Execution policy

Read from v, write into w

Lambda function: defines the operation to be applied to elements of v.

```
#include <execution>
```

```
nvc++ -stdpar -o program program.cpp
```

List of parallel algorithms

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

adjacent_difference
adjacent_find
all_of
any_of
copy
copy_n
count
count_if
equal
exclusive_scan
fill
fill_n
find

find_end
find_first_of
find_if
for_each
for_each_n
inclusive_scan
move
none_of
partition
reduce
remove
remove_if

reverse
reverse_copy
rotate
rotate_copy
search
search_n
sort
stable_sort
swap_ranges
transform
transform_exclusive_scan
transform_inclusive_scan
transform_reduce

Execution Policies

- seq — Traditional non-parallel execution model
- unseq — Execution may be vectorized within a single thread
- par — Allows parallel, multi-threaded execution
- par_unseq — Allows parallel, multi-threaded execution and vectorization within each thread

Available implementations

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Common formalism

C++ Parallel STL

Hardware-specific implementation

Nvidia stdpar

GPU

Intel Threading-Building Blocks

CPU

Heterogeneous (device/host) vs Homogeneous

CUDA Unified Memory

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Execution of Parallel STL model on heterogenous platforms is possible thanks to CUDA unified memory.

```
vector<double> v = { 0., 1., 2., 3., 4., 5. };
for_each(begin(v), end(v), [](double& x) { x = sin(x / N * M_PI); });

for_each( execution::par_unseq, begin(v), end(v),
          [](double& x) { x = sqrt(x);  } );
```

Executed on host

Executed on device

In between, data is automatically transferred from host to device. Be aware of the performance cost!

WARNINGS

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

- Transfer from and to device is automatic, the performance cost is easily overlooked.
- `nvc++` compiler needs to see the definition of all functions called on the device.
- With CUDA Unified memory, only heap data is managed automatically, stack data is not.

No pointers to data on the stack !

```
void multiply_with(vector<double>& v, double a) {
    for_each( execution::par_unseq, v.begin(), v.end(),
              [&](double& x) { x = a * x; } );
}
```

Problem: `a` is captured by reference.

```
void multiply_with(vector<double>& v, double a) {
    for_each( execution::par_unseq, v.begin(), v.end(),
              [=](double& x) { x = a * x; } )
}
```

OK: `a` is captured by value.

Thread safety

- Race conditions
  – The C++ Standard allows implementations to assume that user-provided functions are free of data races.
  – In the heat equation (the example in Part 2), absence of data race is enforced by working with two arrays, one that's read-only and one that's write-only .
  – In the flow solver (the example in Part 3), a smart in-place algorithm is used: read and write operations are carried out on identical memory addresses.
- Synchronization
  – Synchronization of threads takes place between subsequent STL algorithm calls.
  – Similar to OpenMP's fork-join model.

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Part II

EXAMPLE: SOLVING A SCIENTIFIC PROBLEM WITH STDPAR

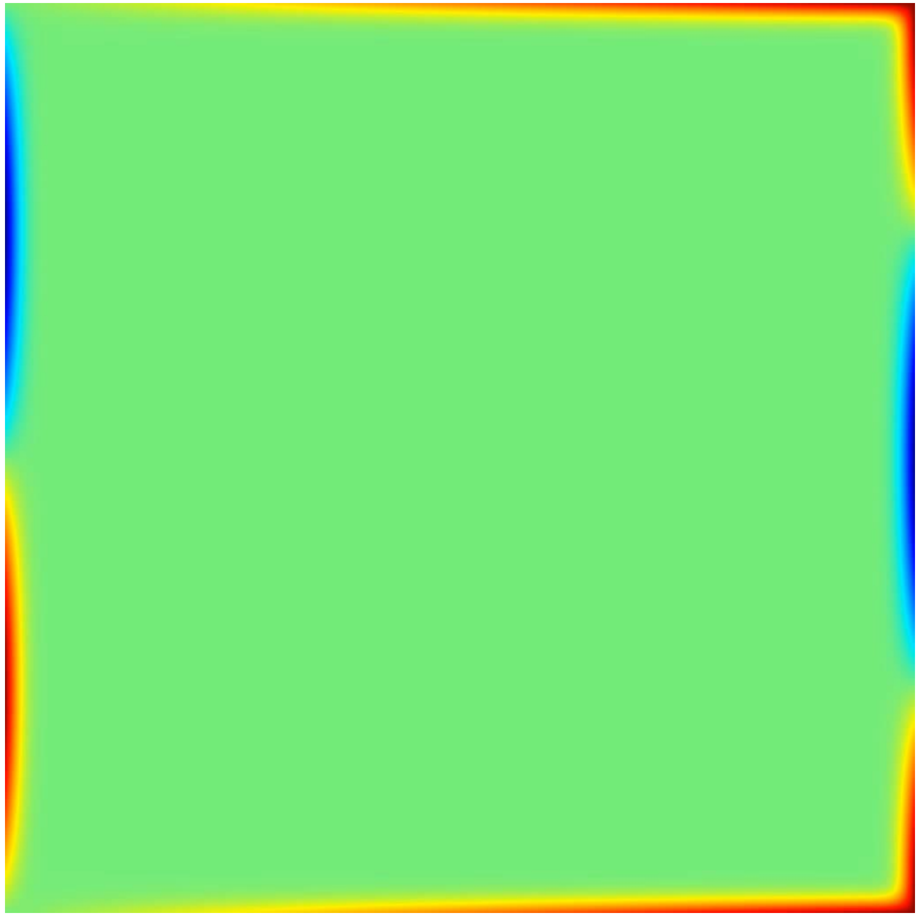
2D Heat Equation

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Time-dependent parabolic PDE

We apply a fixed temperature profile on boundaries

Without heat sources: convergence to stationary state



2D Heat Equation

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

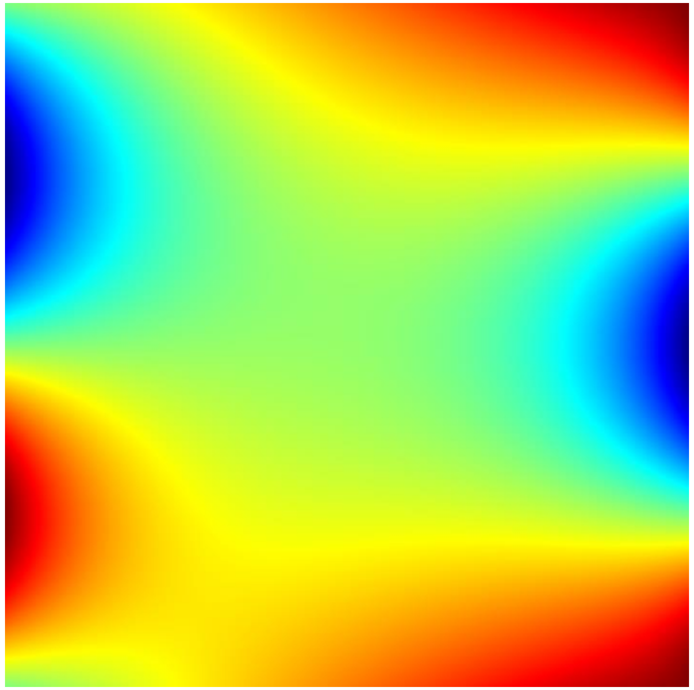
$$\frac{\partial u}{\partial t} = D\left(\frac{\partial^2}{\partial x^2}u(x,y) + \frac{\partial^2}{\partial y^2}u(x,y)\right)$$

$$\frac{u(t+\Delta t) - u(t)}{\Delta t}$$

Time:
Forward Euler

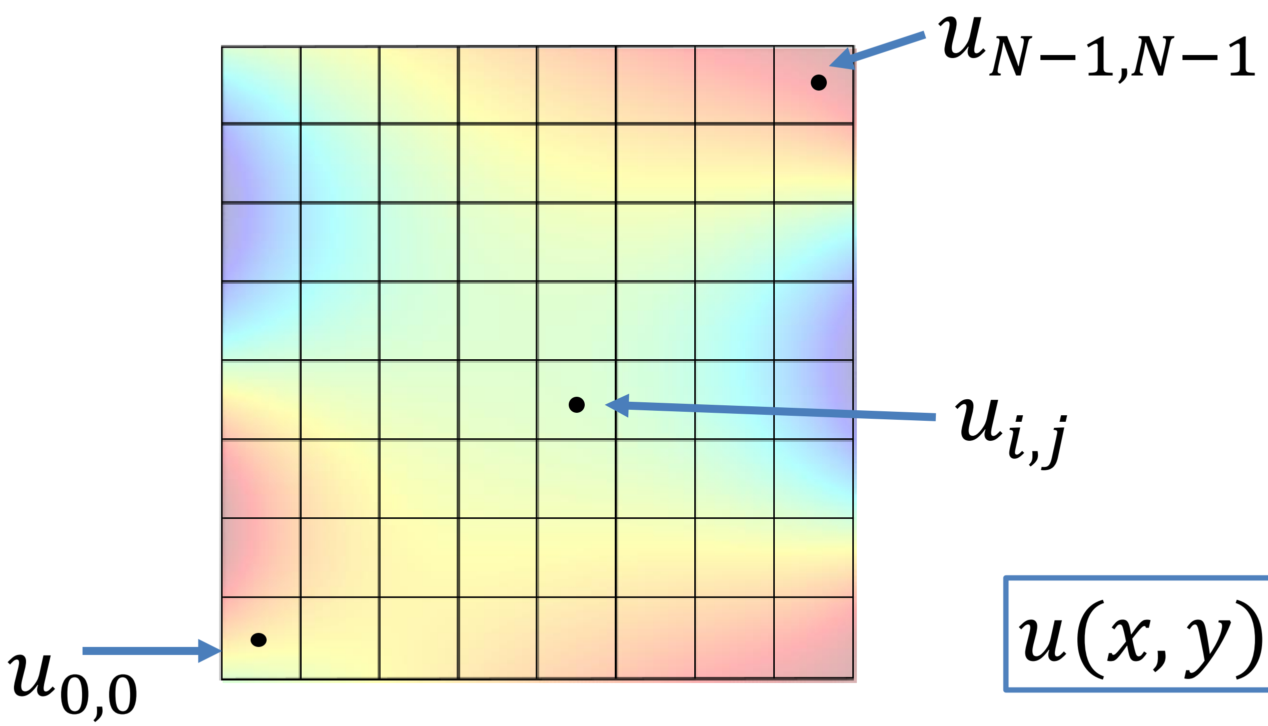
$$\frac{u(x+\Delta x,y) + u(x-\Delta x,y) - 2 * u(x,y)}{\Delta x^2}$$

Central differences (2nd order)



2D Heat Equation

$$u(t+\Delta t) = \left(1 - 4D\frac{\Delta t}{\Delta x^2}\right)u(t) + D\frac{\Delta t}{\Delta x^2}\big(u(x+\Delta x, y) + u(x-\Delta x, y) + u(x, y+\Delta x) + u(x, y-\Delta x)\big)$$



$$\boxed{u(x,y) \equiv u_{i,j}}$$

An implementation with algorithms

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

```
double iterate(vector<double>& u, vector<double>& utmp, int N) {
    double l2 = transform_reduce (
        execution::par,
        begin(u), end(u), begin(utmp),
        0., plus<double>(),
        [Dconst](double const& u_from, double& u_to) {
            u_to = u_from * (1. - 4. * Dconst)
                    + Dconst * ( u[i-1] ... );
                return sqr(u_to - u_from);
            }
        } );
    return l2;
}
```

`transform_reduce` allows a reduction.

Reduction is a sum.

Lambda expression receives an element from `u` and one from `utmp` and returns a term for the sum.

Wait… how to access neighbors ?
We have access to the current element only.

Pointer arithmetics provides index

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

```
double* u_ptr = &u[0];
…
[u_ptr, Dconst](double const& u_from, double& u_to) {
    size_t i = &u_from – u_ptr;
    u_to = u_from * (1. - 4. * Dconst)
               + Dconst * ( u_ptr[i-1] + u_ptr[i+1] +
                            u_ptr[i-N] + u_ptr[i+N] );
    return sqr(u_to - u_from);
} );
```

Capture a pointer to `u` heap data.

The address of `u_from` reveals the linear index.

The complete lambda expression

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

```
[u_ptr, Dconst](double const& u_from, double& u_to) {
    size_t i = &u_from - u_ptr;
    size_t iX = i / N;
    size_t iY = i % N;
    bool on_boundary_x = iX == 0 || iX == N-1;
    bool on_boundary_y = iY == 0 || iY == N-1;
    if (on_boundary_x || on_boundary_y) {
        return 0.;
    }
    else {
        u_to = u_from * (1. - 4. * Dconst)
                   + Dconst * ( u_ptr[i-1] + u_ptr[IND(i+1)] +
                                u_ptr[i-N] + u_ptr[IND(i+N] );
        return sqr(u_to - u_from);
    }
} );
```

Exclude boundaries from computations.

Try it out:
www.gitlab.com/UnigeHPFS/paralg

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Part III

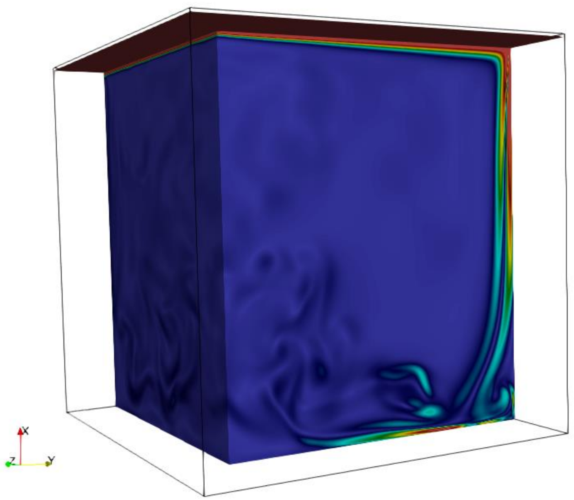
A COMPLETE EXAMPLE: COMPUTATIONAL FLUID DYNAMICS

The STLBM code

https://www.gitlab.com/UnigeHPFS/stlbm

- A collection of parallel-STL based code templates for fluid dynamics.
- Straightforward: a fully self-contained code in 600 lines.
- Approach: Lattice Boltzmann Method (LBM)

Flow in a lid-driven cavity

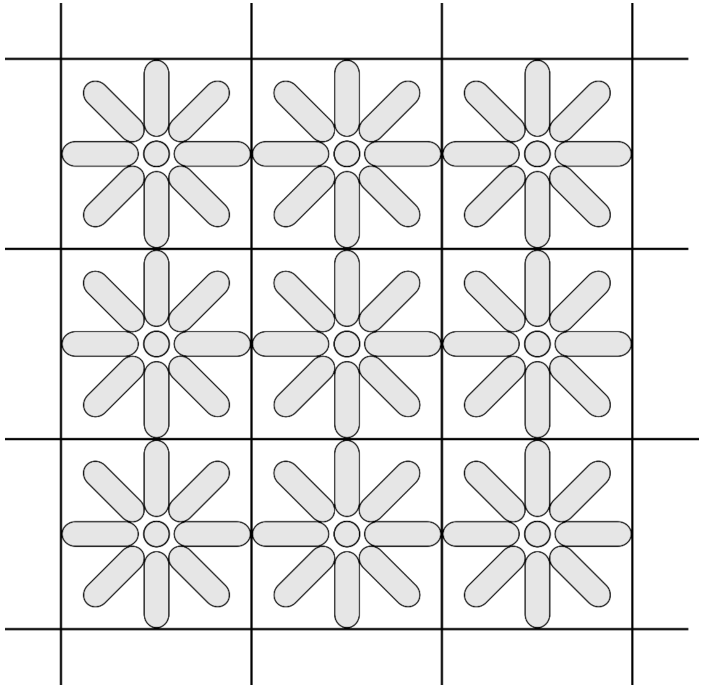


Porous media flow

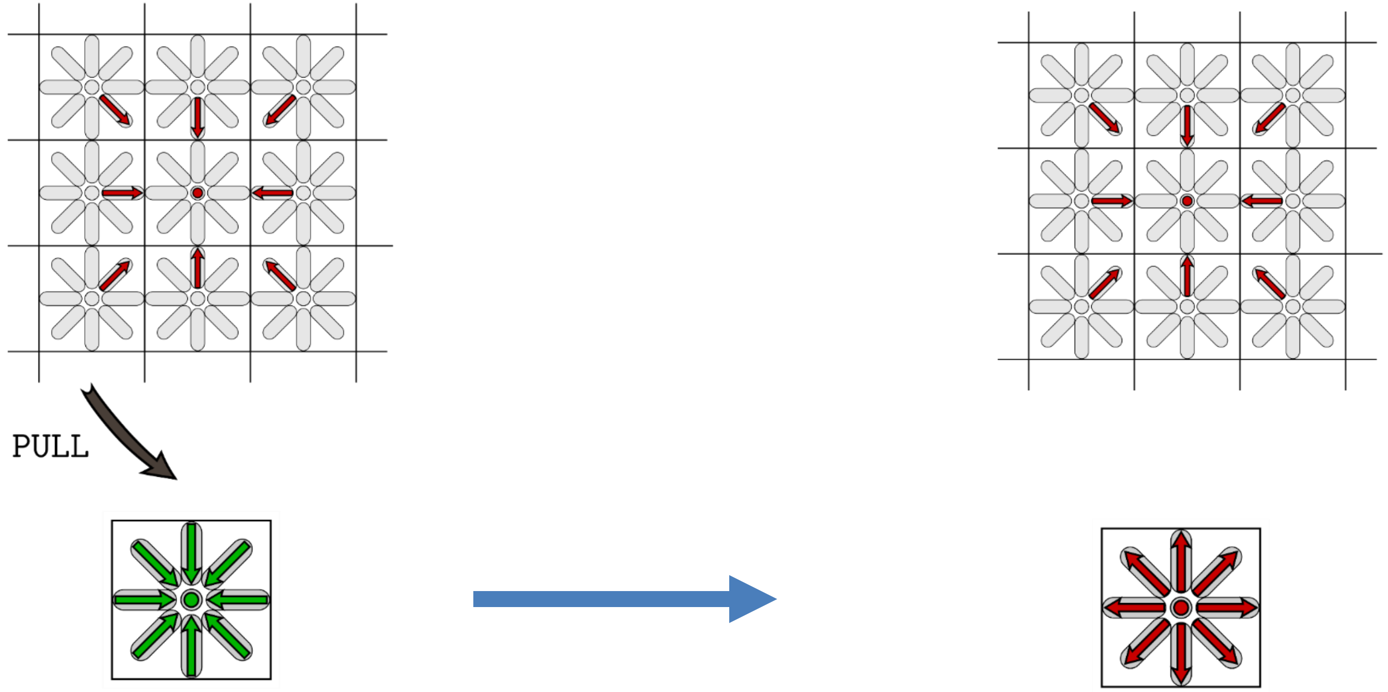
J. Latt, C. Coreixas, J. Beny,
Cross-platform programming model for many-core lattice Boltzmann simulations
arXiv preprint arXiv:2010.11751, 2020

Lattice Boltzmann scheme

- 9 variables per cell (2D), 19 or more variables per cell (3D).
- Similar to heat equation algorithm: nearest-neighbor cell access.
- Computations performed on every cell are more involved.
- Thread safety: we use in-place algorithm and select the memory access pattern carefully.



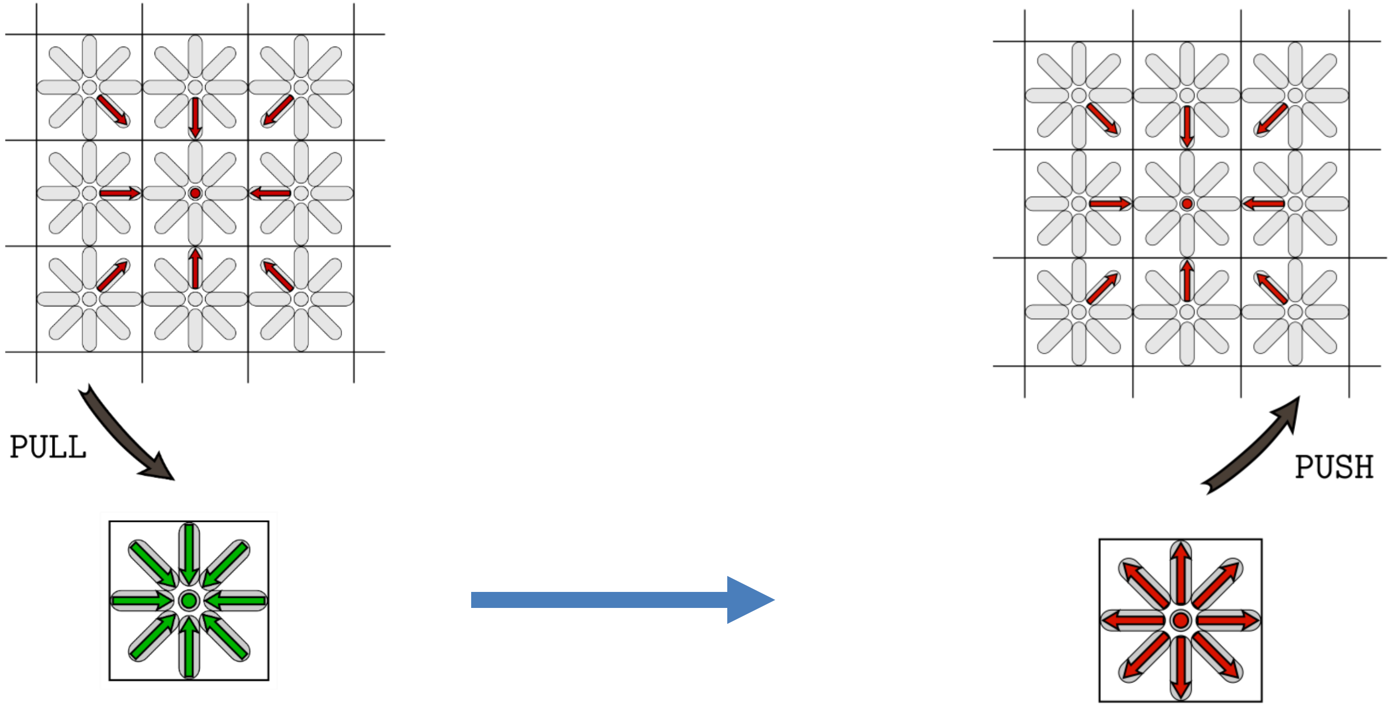
LBM: treatment of a single cell



Treatment of one cell: PULL - COMPUTE - PUSH

LBM: treatment of a single cell

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE



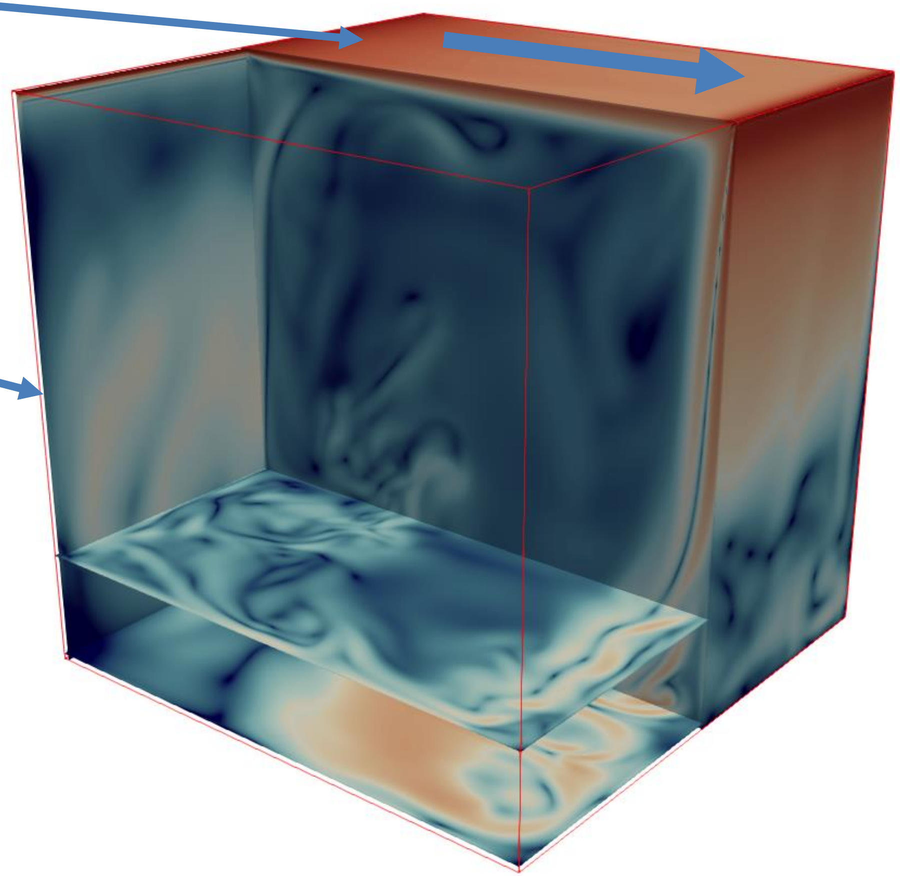
Thread safety: same data accessed at PULL and PUSH.

AA-Pattern: Bailey et al. *http://ieeexplore.ieee.org/document/5362489/*

Test case: 3D Lid-driven cavity

Moving top lid: constant velocity from left to right.

Cubic box with no-slip walls



Test case: 3D Lid-driven cavity

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

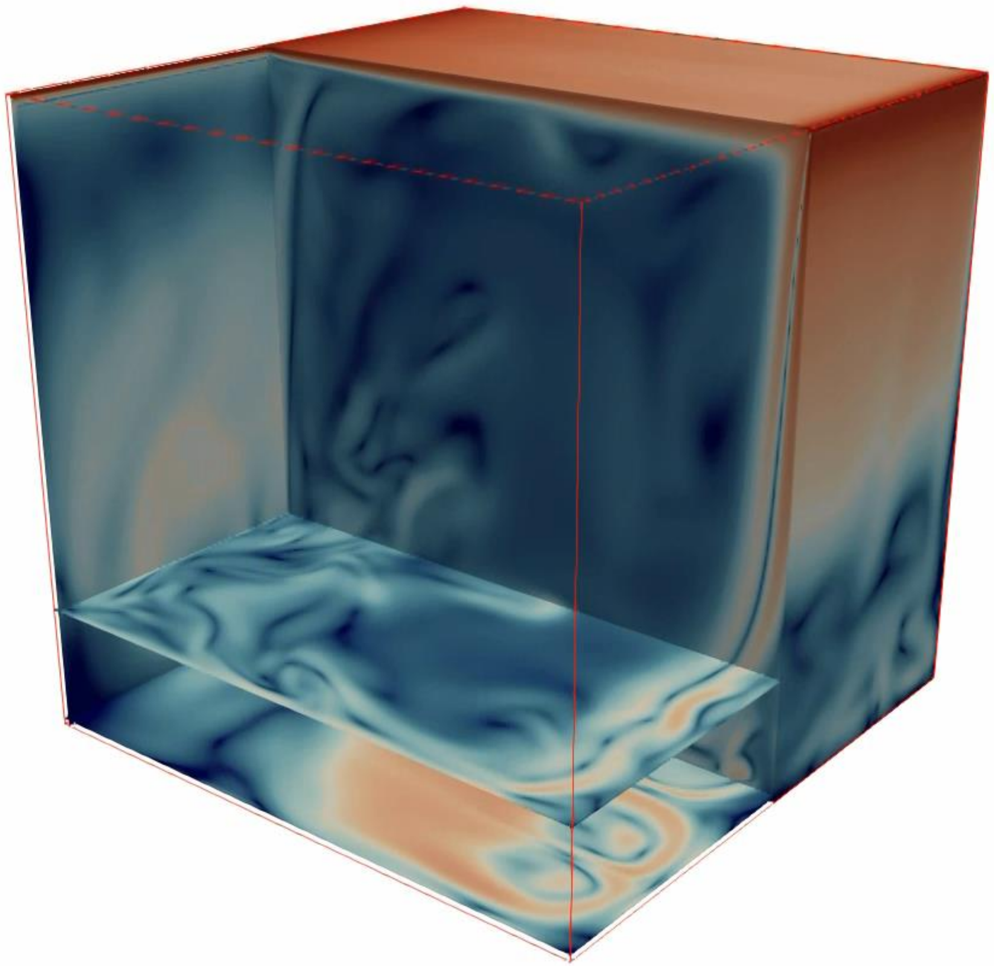
Reynolds: 10’000

LB model: D3Q19, Recursive-regularized with omega_bulk = 1, no subgrid-scale model.

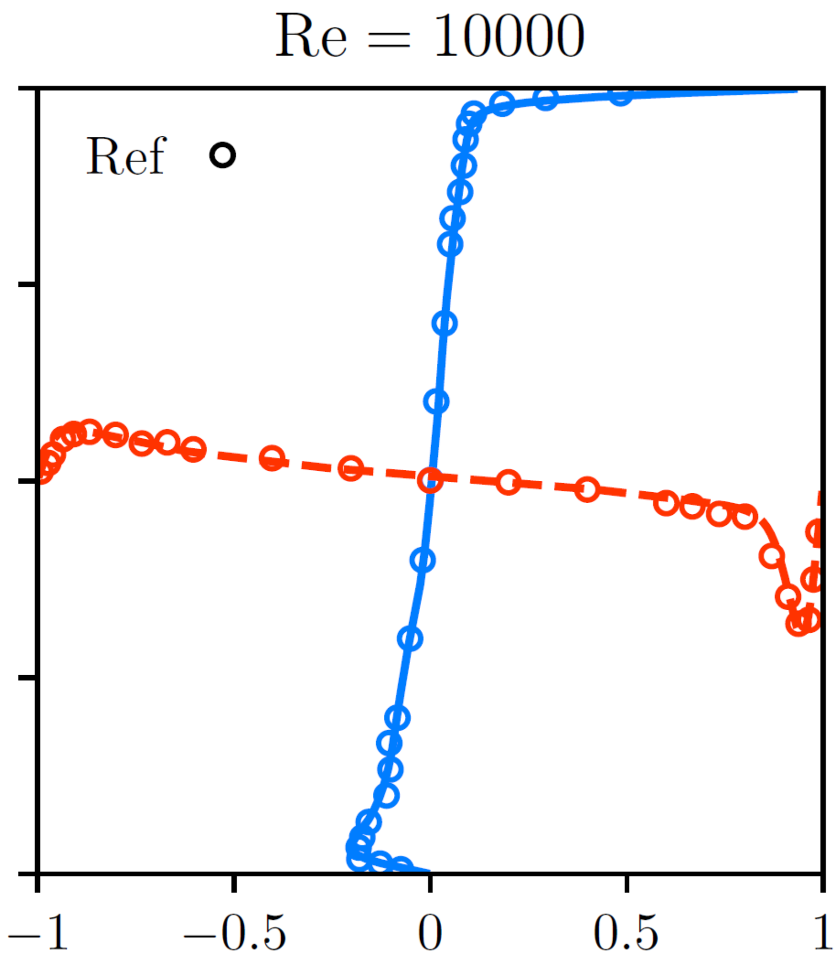
400 x 400 x 400 domain (homogeneous mesh)

560k iterations

2:40 hours on a A100



Validation: 3D cavity



400 x 400 x 400 domain
(Homogeneous mesh)

3 Million iterations

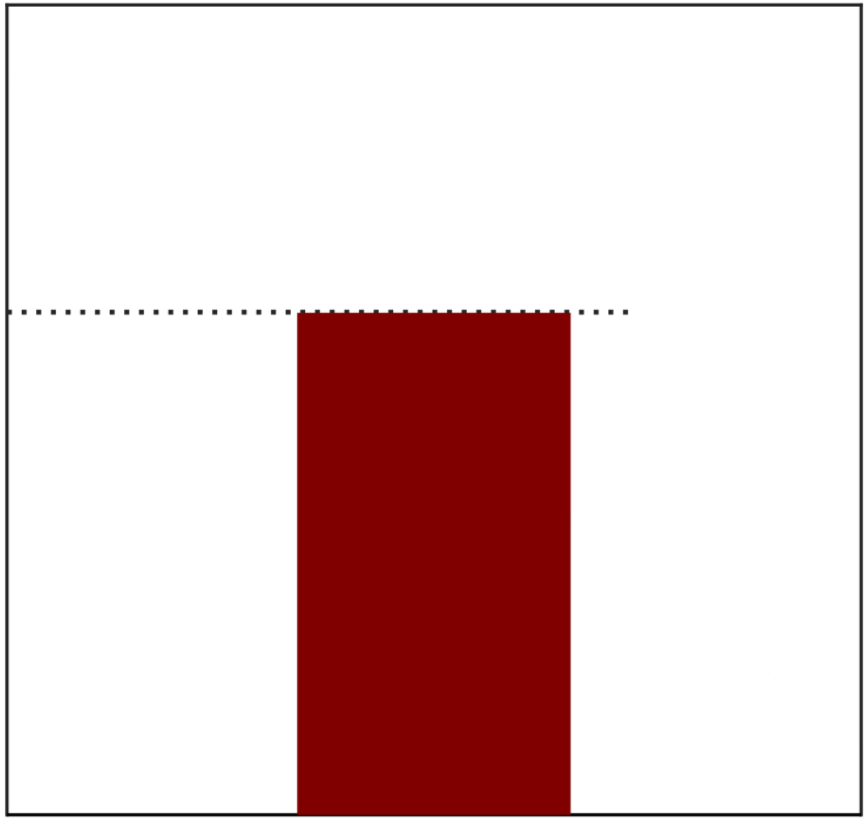
15 hours on a A100

Performance

UNIVERSITÉ
DE GENÈVE
FACULTY OF SCIENCE

"Mega-LUPS":
Million lattice node updates per second

Our stdpar implementation 3724 MLUPS

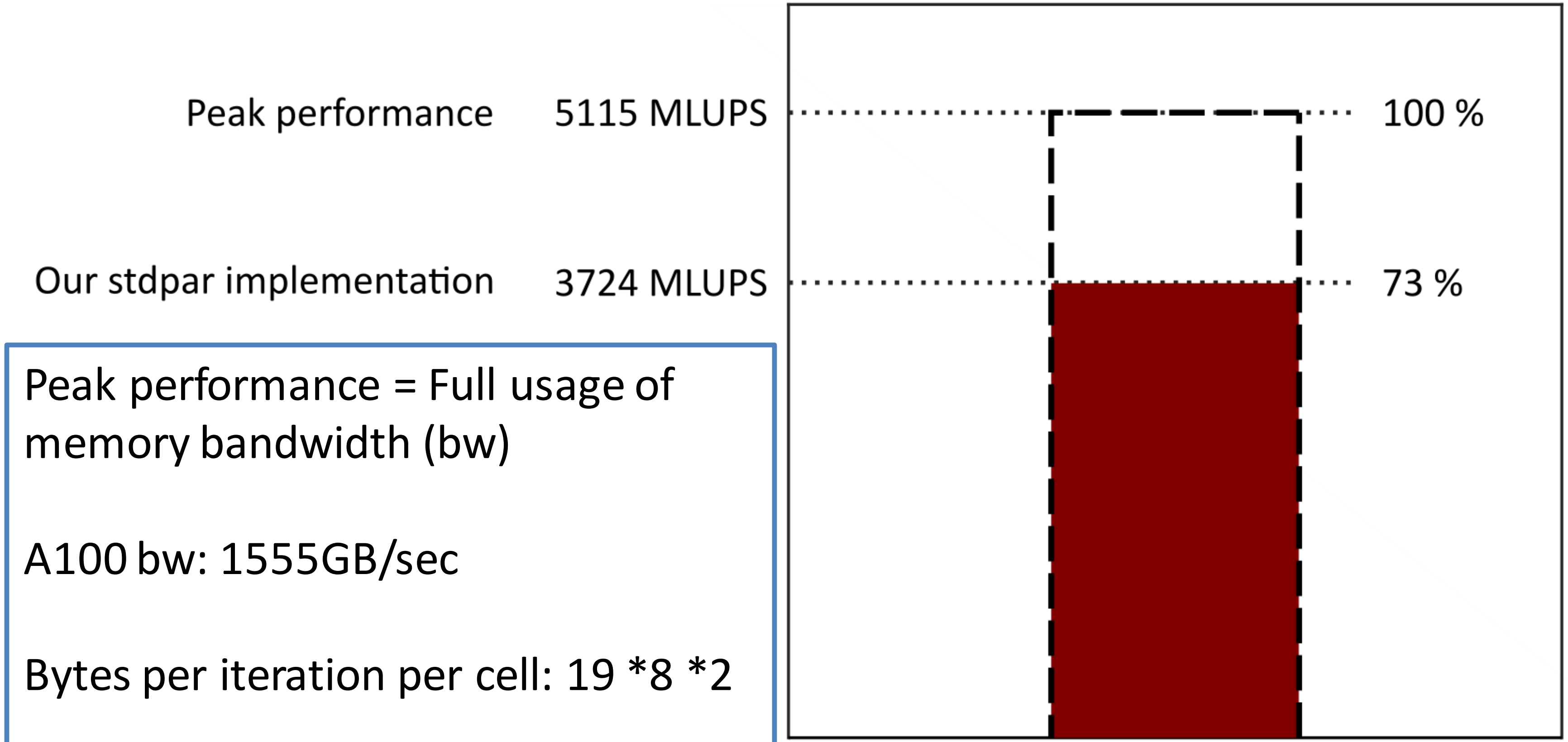


200x200x200 domain
Double-precision
A100 GPU

Performance vs. Peak performance

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

200x200x200 domain
Double-precision
A100 GPU



Peak performance = Full usage of memory bandwidth (bw)

A100 bw: 1555GB/sec

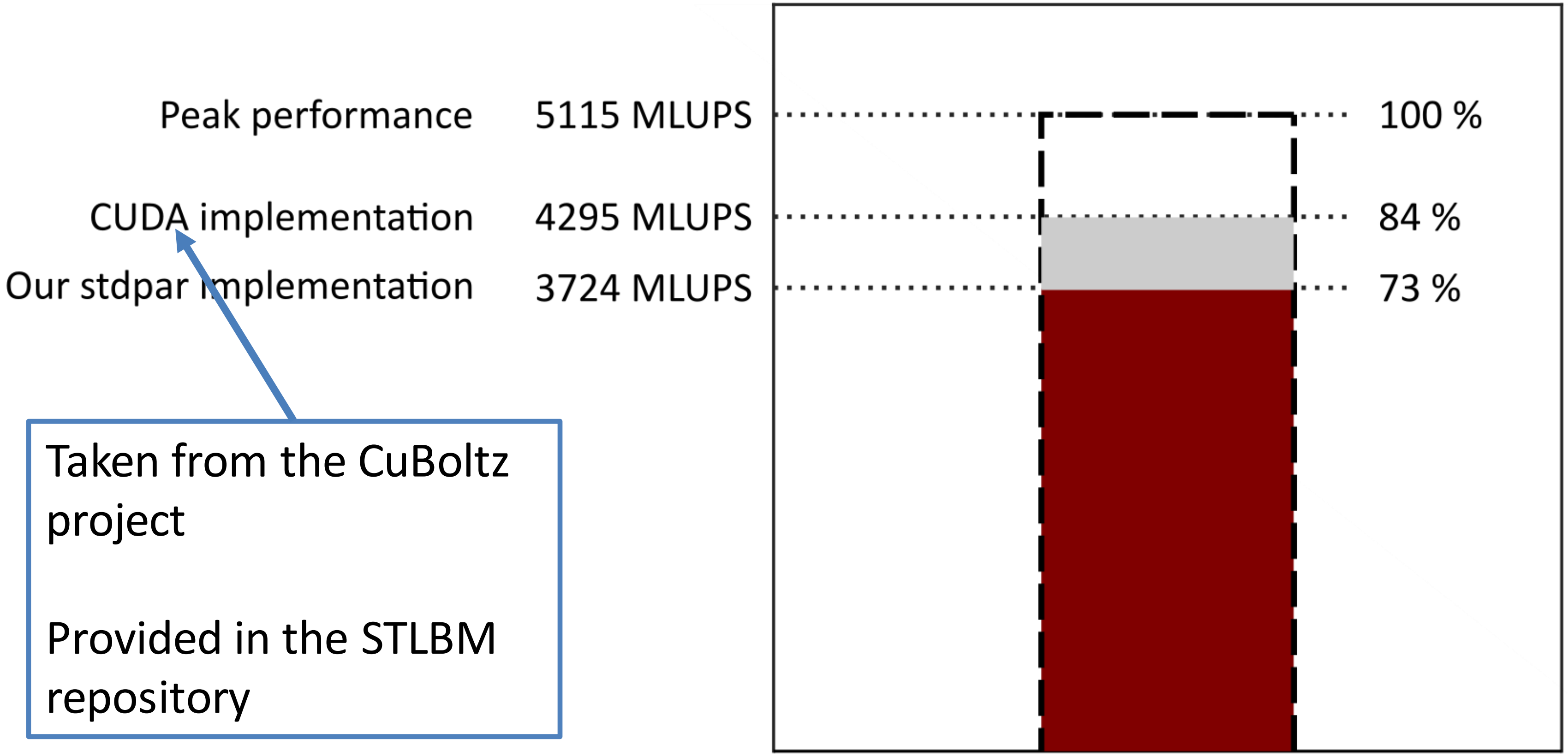
Bytes per iteration per cell: 19 *8 *2

Peak MLUPS = $1.555 * 10^6 / (19*8*2)$

Performance vs. Cuda code

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

200x200x200 domain
Double-precision
A100 GPU



Taken from the CuBoltz project

Provided in the STLBM repository

Performance on different platforms

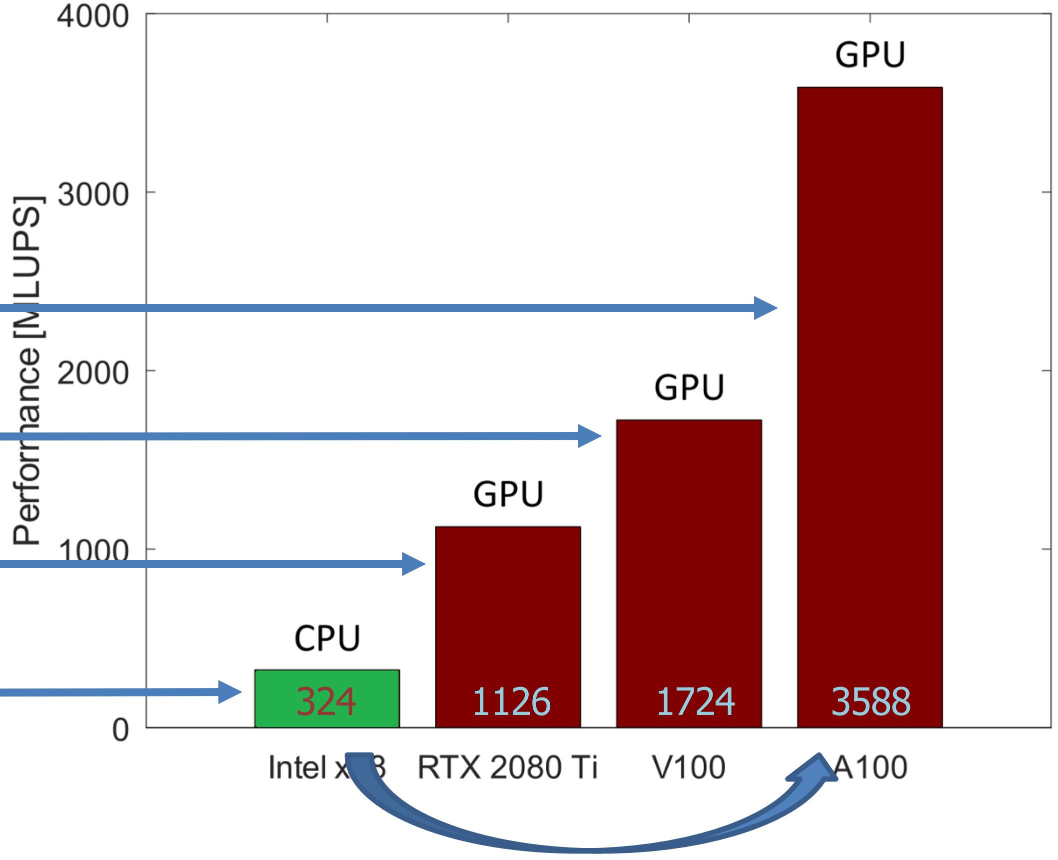
UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Nvidia A100 (Ampere)

NVidia V100 (Volta)

NVidia RTX 2080 Ti (Turing)

Intel Xeon Gold 6240R CPU, 48 cores



128 x 128 x 128
Double-precision

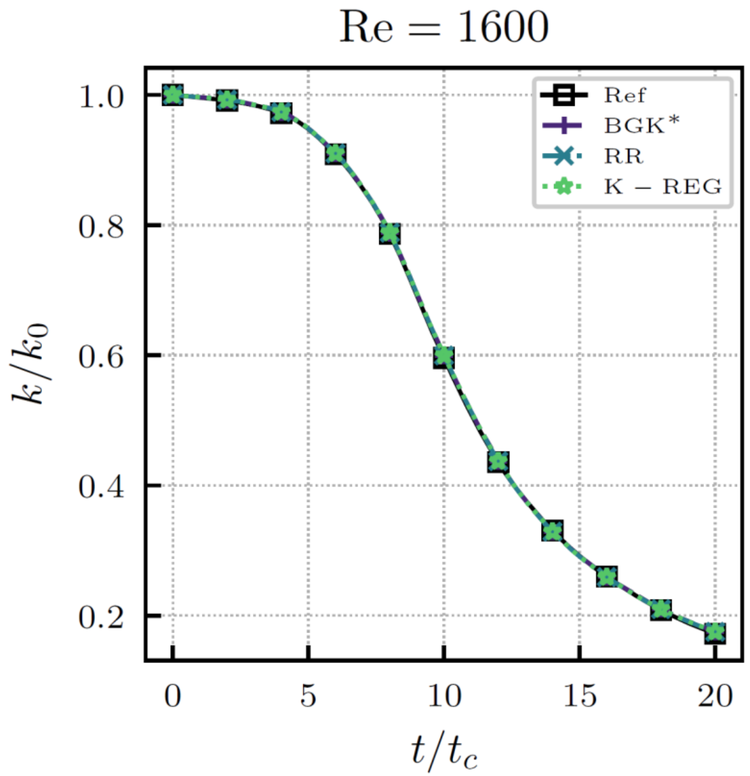
11-fold speedup

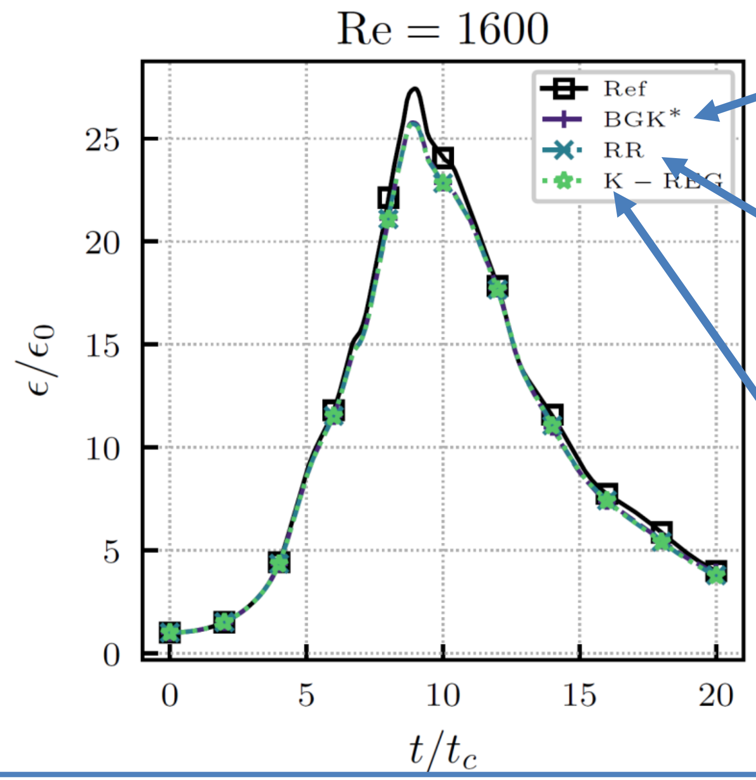
Collision models: low Reynolds

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Decaying 3D Taylor-Green vortex

D3Q27, Re: 1600, Mach: 0.2, Resolution: 512 x 512 x 512





"Traditional" BGK model, but with extended equilibrium.

Recursive-regularized, a robust modern collision model (the one used for the cavity).

Cumulant approach with regularization of high-order moments: another robust choice.

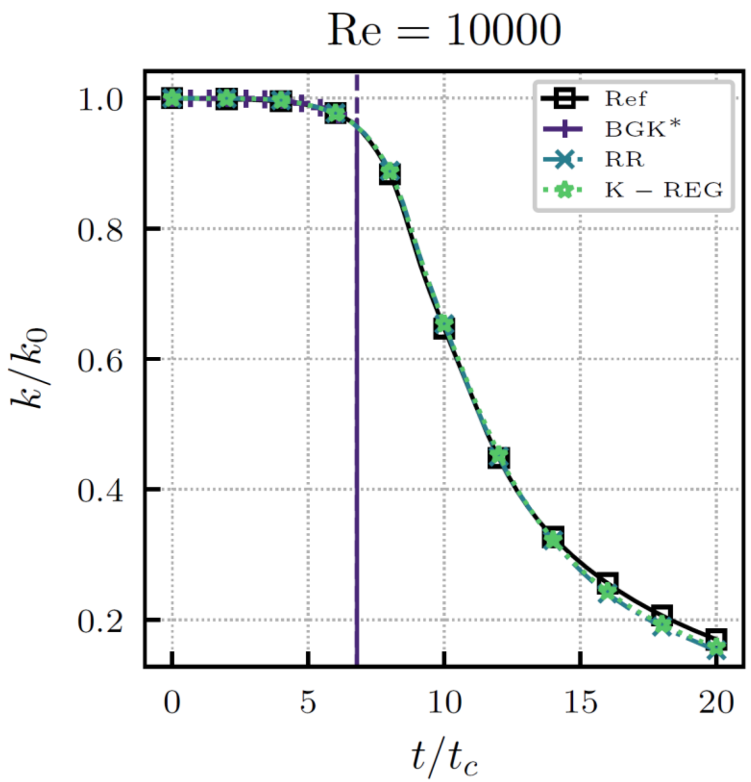
Laizet et al., *3D Taylor-Green vortex Direct Numerical Simulation statistics from Re=1250 to Re=20000,* Zenodo, 2019

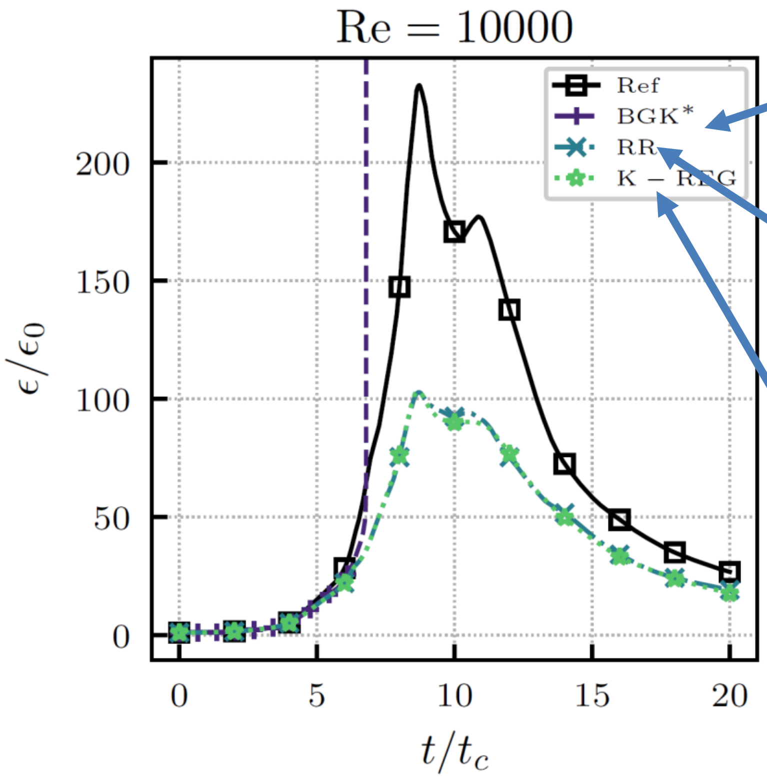
Coreixas, Chopard & Latt. *Comprehensive comparison of collision models…* Phys. Rev. E, 2019, 100, 033305.
Coreixas, Wissocq, Chopard & Latt. *Impact of collision models on the physical properties…* Phil. Trans. R. Soc. A, 2020, 378, 20190397.

Collision models: high Reynolds

Decaying 3D Taylor-Green vortex

D3Q27, Re: 10’000, Mach: 0.2, Resolution: 512 x 512 x 512





“Traditional” BGK model, but with extended equilibrium.

Recursive-regularized, a robust modern collision model (the one used for the cavity).

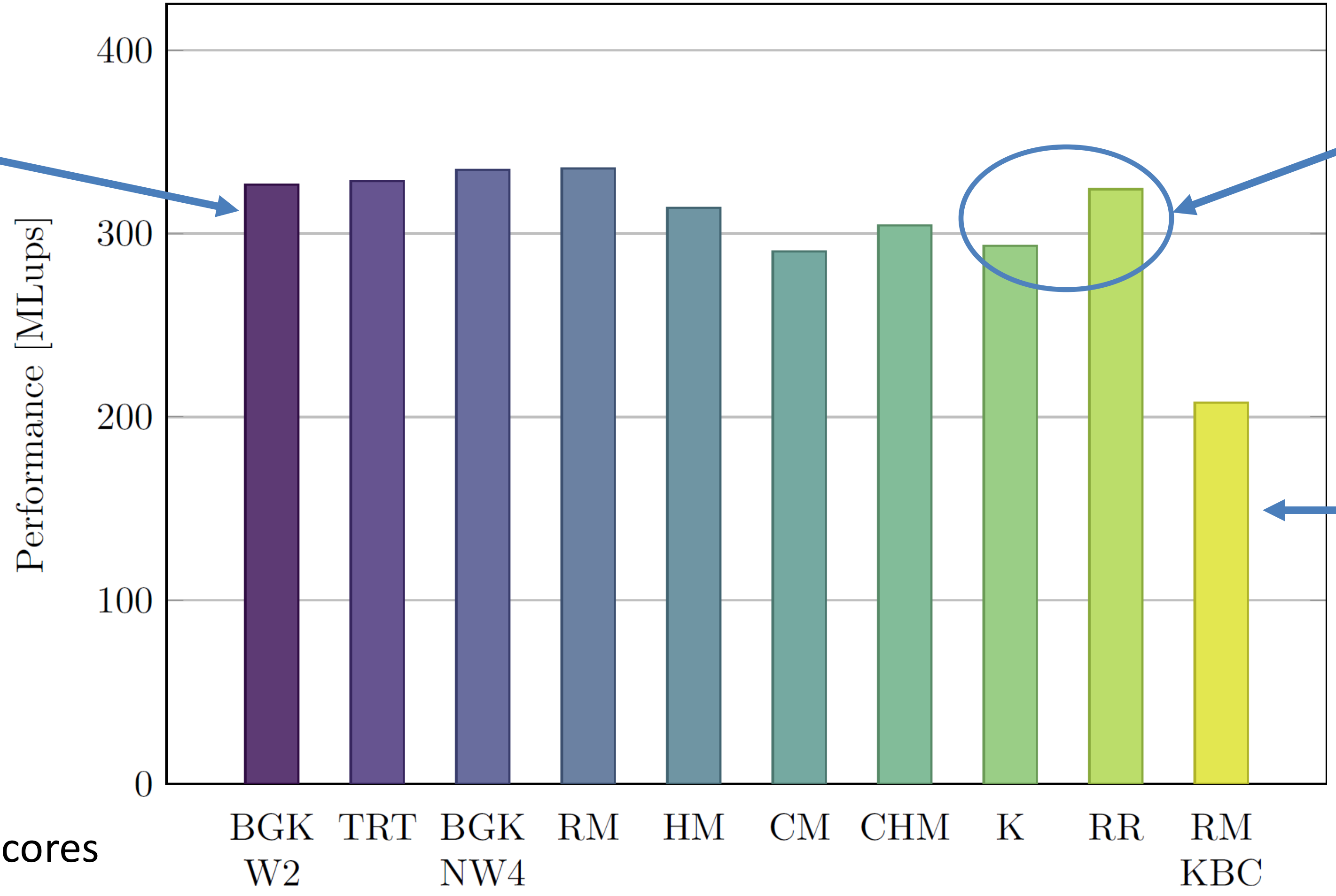
Cumulant approach with regularization of high-order moments: another robust choice.

Laizet et al., *3D Taylor-Green vortex Direct Numerical Simulation statistics from Re=1250 to Re=20000,* Zenodo, 2019

Coreixas, Chopard & Latt. *Comprehensive comparison of collision models…* Phys. Rev. E, 2019, 100, 033305.
Coreixas, Wissocq, Chopard & Latt. *Impact of collision models on the physical properties…* Phil. Trans. R. Soc. A, 2020, 378, 20190397.

Performance of collision models: CPU

The most standard, BGK model



The more robust alternatives from the previous slides.

KBC: another one, also used in the literature.

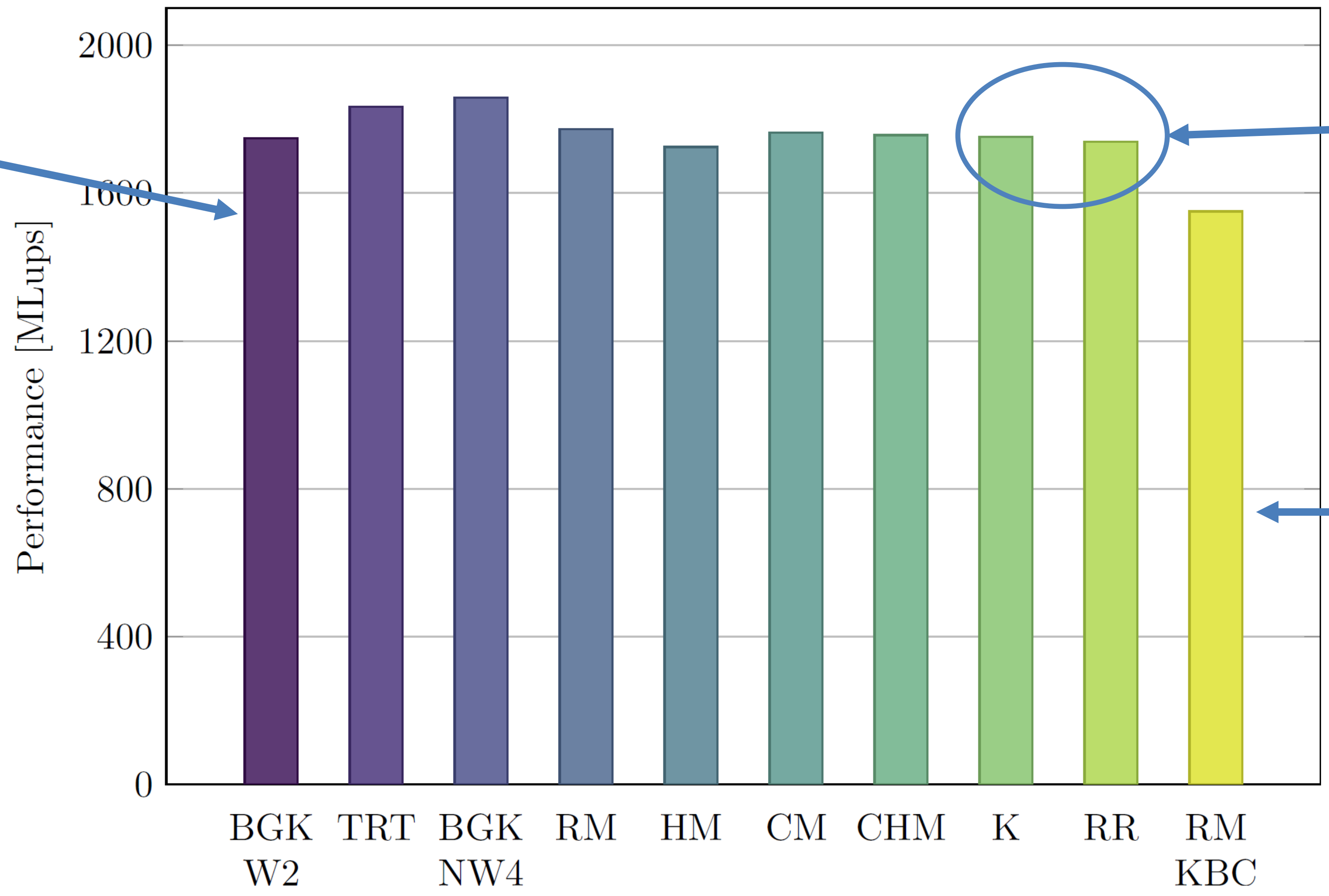
Resolution: 128 x 128 x 128

Lattice: D3Q19

CPU: Xeon Gold 48 cores

Performance of collision models: GPU

The most standard, BGK model



The more robust alternatives from the previous slides.

KBC: another one, also used in the literature.

Resolution:
128 x 128 x 128

Lattice: D3Q19

GPU: V100

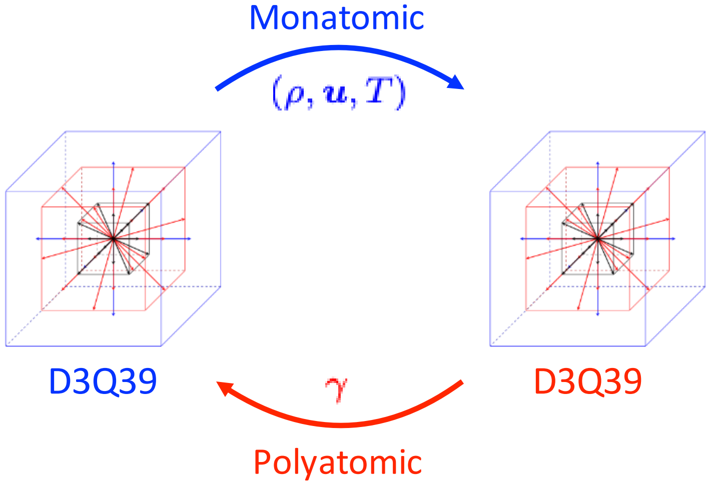
Conclusion

- Parallel STL: allows parallelization of numerical algorithms in a fully hardware agnostic way.
- For some algorithms, the performance on GPUs is almost the same as in an “optimal” code.
- A few hundred lines of code, without hardware-specific knowledge, achieve cluster-level performance.
- Performance varies little from one collision model to another: robust, modern collision models should be preferred (code can be copy-pasted from the STLBM project).

Outlook

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

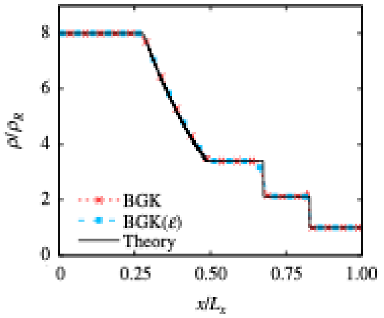
Compressible LBMs

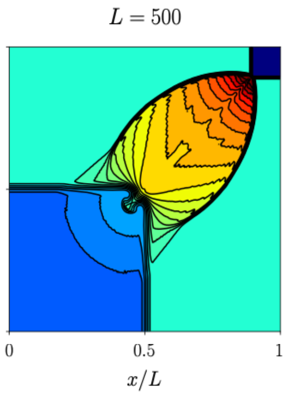


Performance (D3Q39Q39)

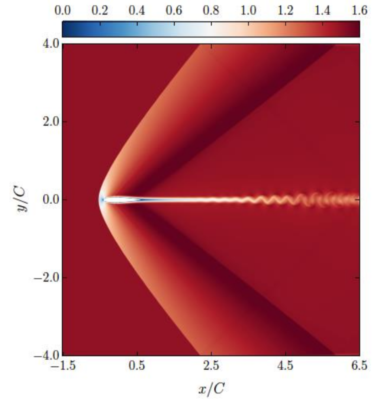
Hardware	i7-8700 (3.2 GHz)	Volta 100	Ampere 100
MLUPS	0.67	30	~ 60
μs/pt/it	1.49	0.033	~ 0.016

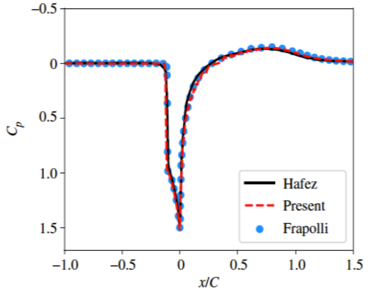
Riemann problems



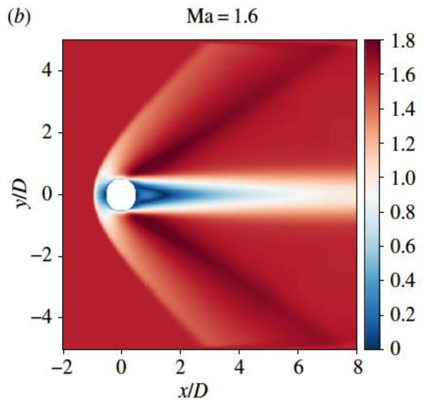


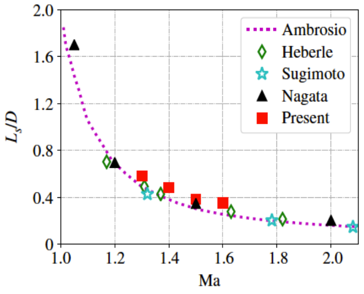
Flow past NACA0012





Flow past sphere





Latt et al., Efficient supersonic flow simulations using lattice Boltzmann methods based on numerical equilibria , *Phil. Trans. R. Soc. A*, **2020**.
Coreixas & Latt, Compressible lattice Boltzmann methods with adaptive velocity stencils: An interpolation-free formulation, *Phys. Fluids*, **2020**

Thank you for your attention

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

High Performance Fluid Simulation Group:
www.unige.ch/hpfs/

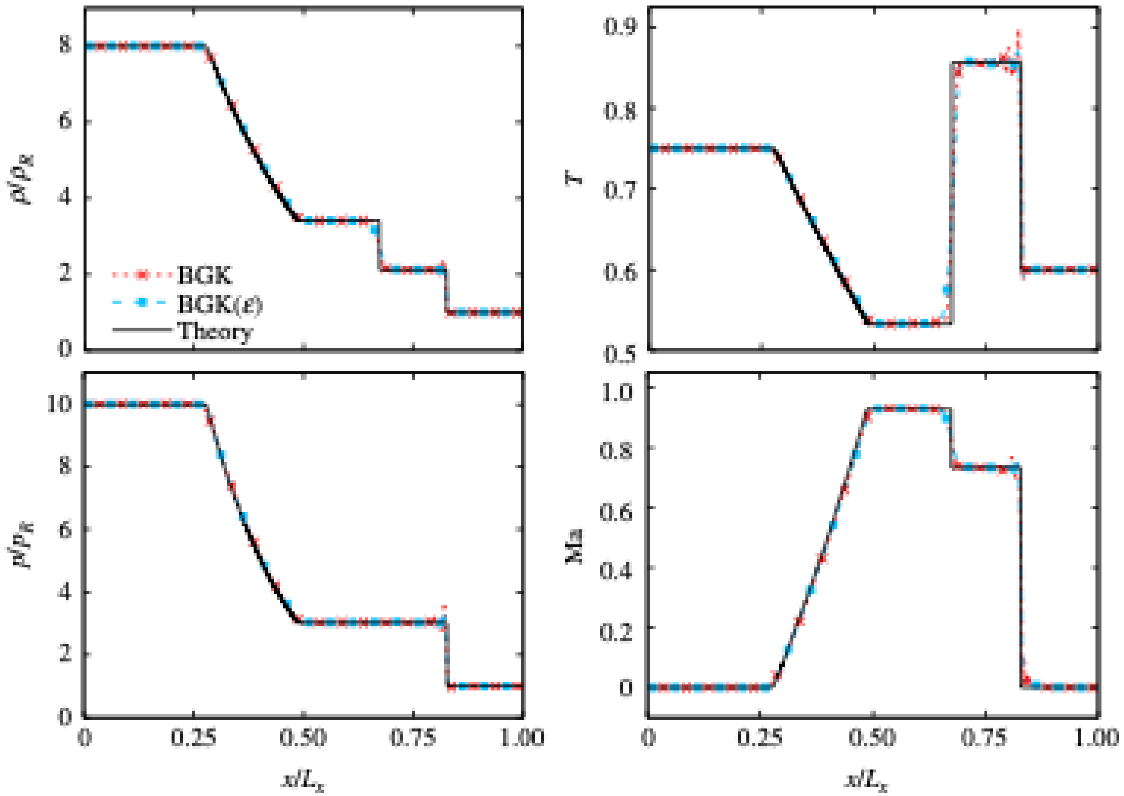
Example codes from this presentation:
www.gitlab.com/UnigeHPFS/paralg

STLBM source code:
www.gitlab.com/UnigeHPFS/stlbm

We thankfully acknowledge
- The Swiss PASC project for funding.
- The NVIDIA HPC Software and Compiler teams for access to a A100.

Results – Sod Shock Tube

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE



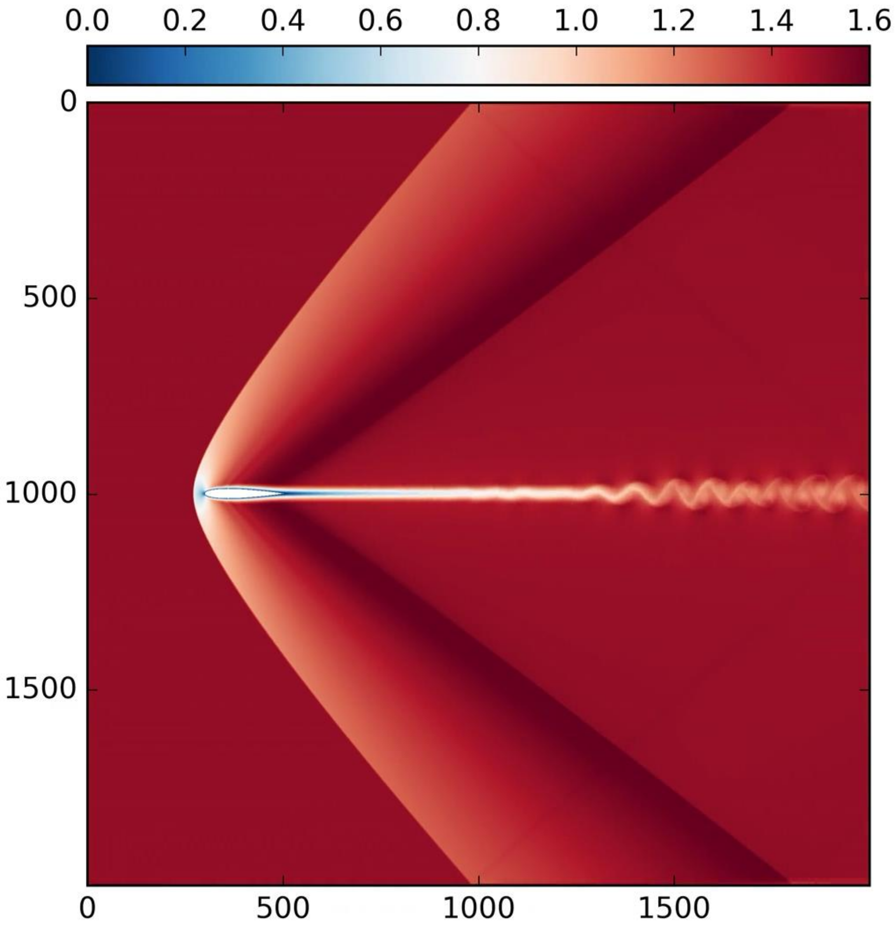
- Good robustness even using the BGK operator
- Kinetic sensor allows for **inviscid** simulations

$$\epsilon = \frac{1}{V} \sum_{i=0}^{V-1} \frac{|f_i - f_i^{eq}|}{f_i^{eq}}$$

Latt et al., Efficient supersonic flow simulations using lattice Boltzmann methods based on numerical equilibria
Phil. Trans. R. Soc. A, **2020**, 378, 20190559

Results – 2D NACA0012

Mach number field



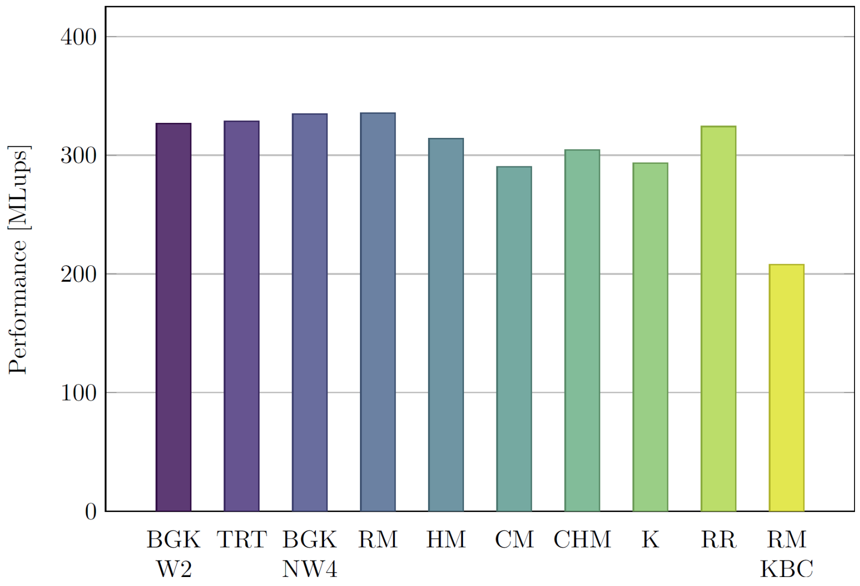
- **Surprisingly**, all BCs have a **good** behavior
- **Only the BB** leads to **spurious** oscillations

Latt et al., Efficient supersonic flow simulations using lattice Boltzmann methods based on numerical equilibria
Phil. Trans. R. Soc. A, **2020**, 378, 20190559

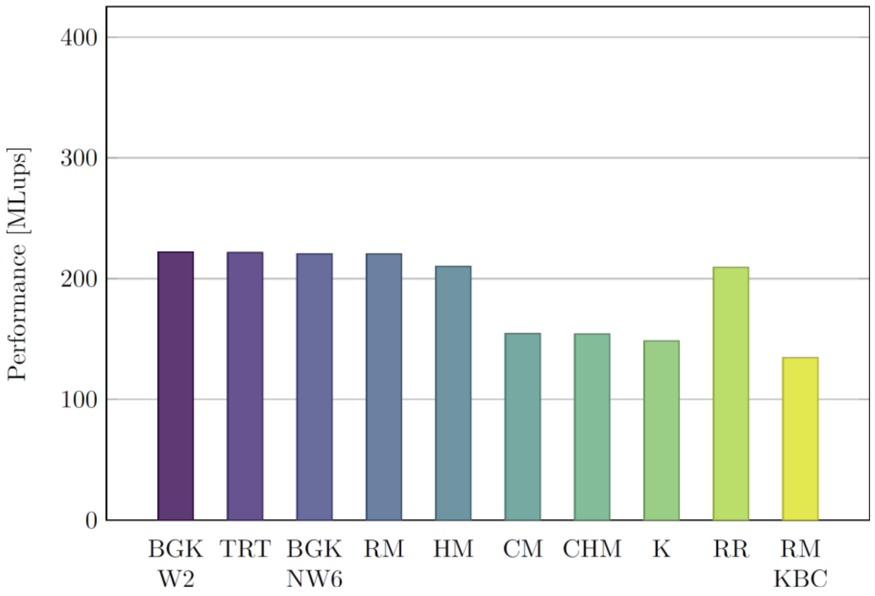
D3Q27 Lattice – CPU (Xeon Gold x48)

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

D3Q19



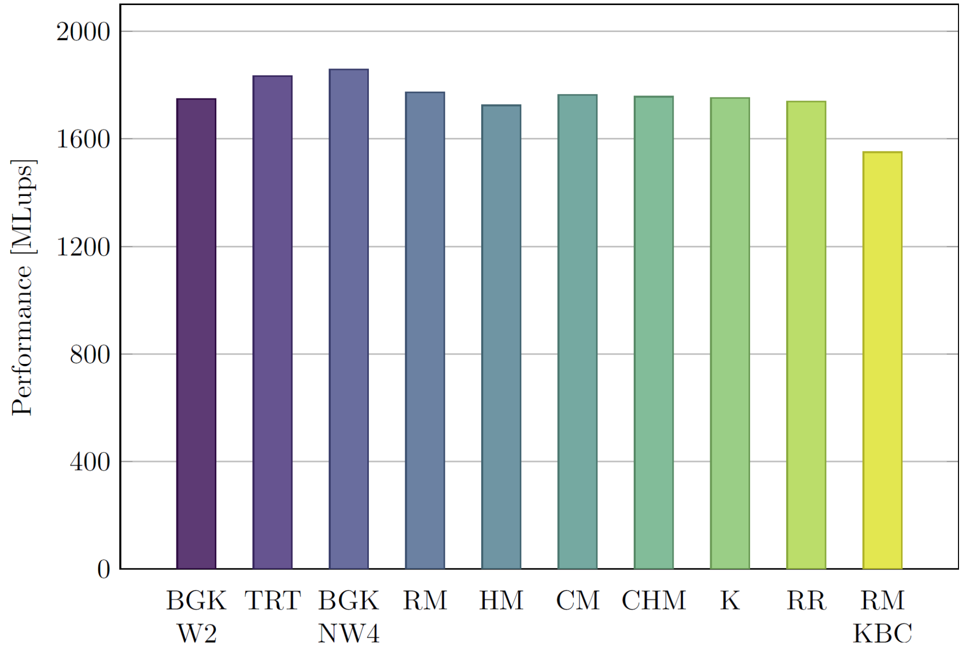
D3Q27



D3Q27 Lattice – GPU (V100)

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

D3Q19



D3Q27