Palabos Online Seminar Series
5th of May 2021

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

- Introducing the Palabos Online Seminar Series
- Presentation of the Palabos project "From CPU to GPU in 80 days"

Jonas Latt - Université de Genève (UniGe)

The Palabos Online Seminar Series

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Topic: Presentation of work achieved with Palabos

Presenters: Presentations are open to the community

Dates: The first Wednesday of every month, at 10 am CET or at 5 pm CET.

Further information:
https://palabos.unige.ch/community/palabos-online-seminar-series/

From CPU to GPU in 80 days

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Goal: Port substantial parts of Palabos to GPU

Dates: Project starts today, ends on 23rd of July

Palabos fork: gitlab.com/unigehpfs/palabos

Community involvement: Throughout the project, try it out, provide feedback

Project website: palabos.unige.ch/community/cpu-gpu-80-days/

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Overview: the general ideas

Idea: transfer existing application to GPU

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

```
// Allocate memory for the populations
MultiBlockLattice3D<T, DESCRIPTOR> lattice (
    nx, ny, nz,
    new BGKdynamics<T,DESCRIPTOR>(omega) );

// Specify type of boundary condition
OnLatticeBoundaryCondition3D<T,DESCRIPTOR>*
    boundaryCondition =
      createLocalBoundaryCondition3D<T,DESCRIPTOR>();

// Create initial and boundary condition
cavitySetup(lattice, parameters,
            *boundaryCondition);
```

```
// Loop over main time iteration.
for (plint iT=0; iT<20; ++iT) {
    lattice.collideAndStream();
}
```

The initialization does not need to be changed: it will be executed on CPU

For the time iterations: transfer data to the AcceleratedLattice (on GPU)

```
AcceleratedLattice<T, DESCRIPTOR> aLattice(lattice);
// Loop over main time iteration.
for (plint iT=0; iT<20; ++iT) {
    aLattice.collideAndStream();
}
```

Structure of the project

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Work package 1: Setup of test cases

Work package 2: AcceleratedLattice on CPU

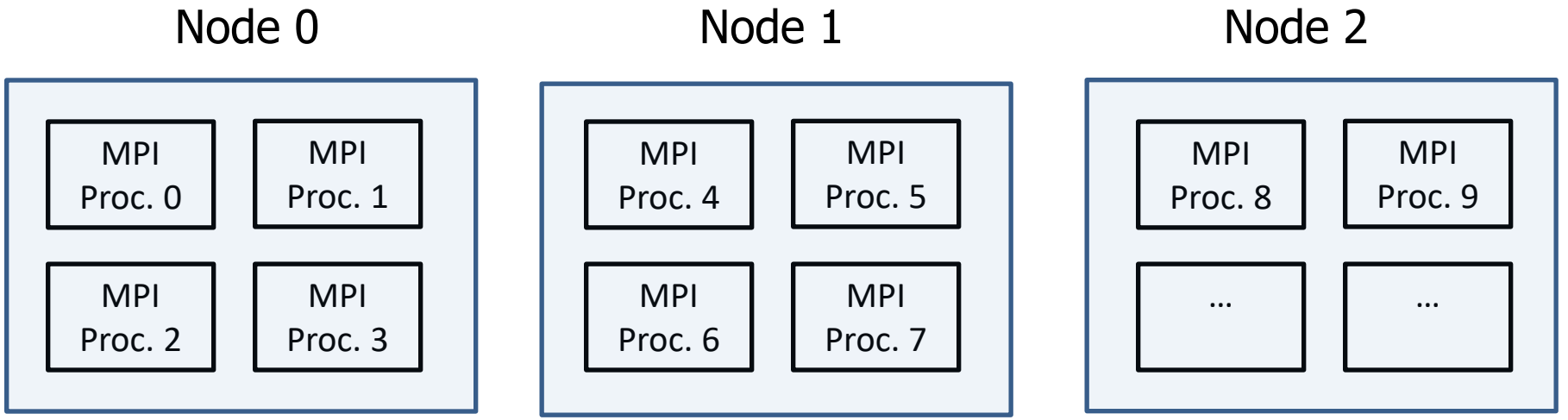
Work package 3: AcceleratedLattice on GPU

Work package 4: Framework for dynamics objects and data processors

Work package 5: Improvement, acceleration

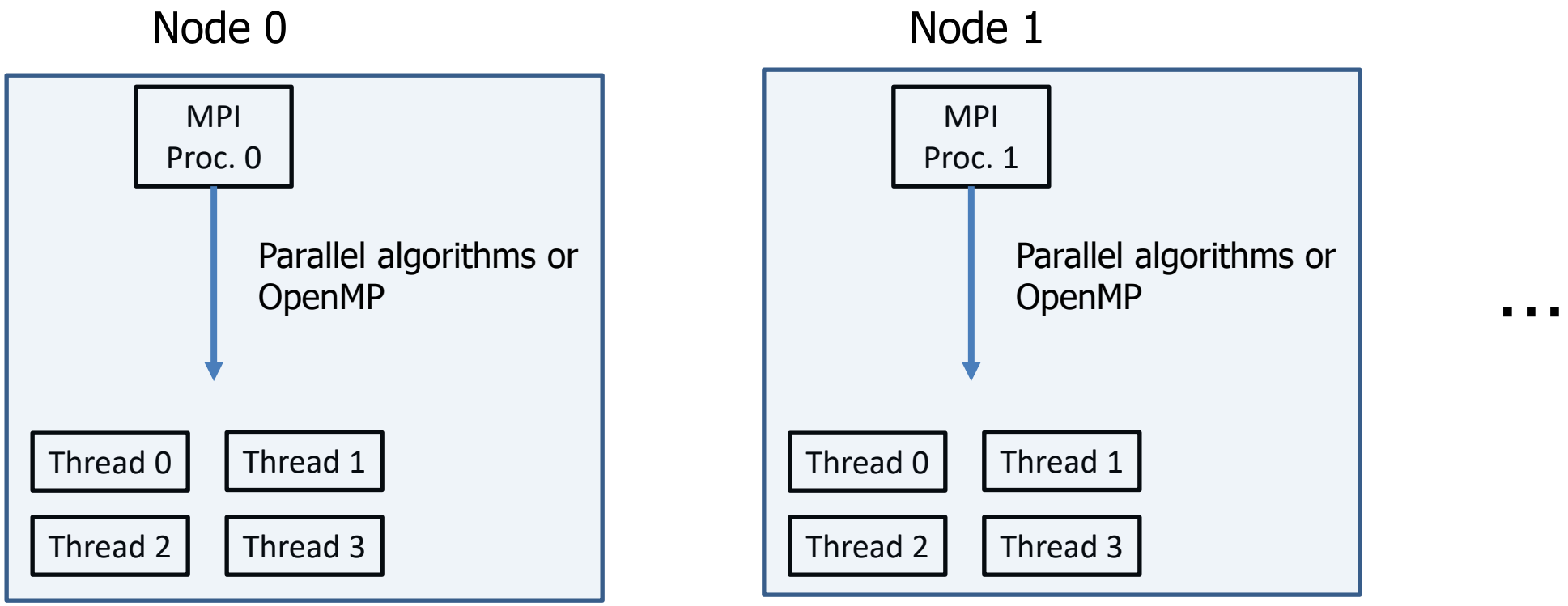
Parallelism in Palabos, currently

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE



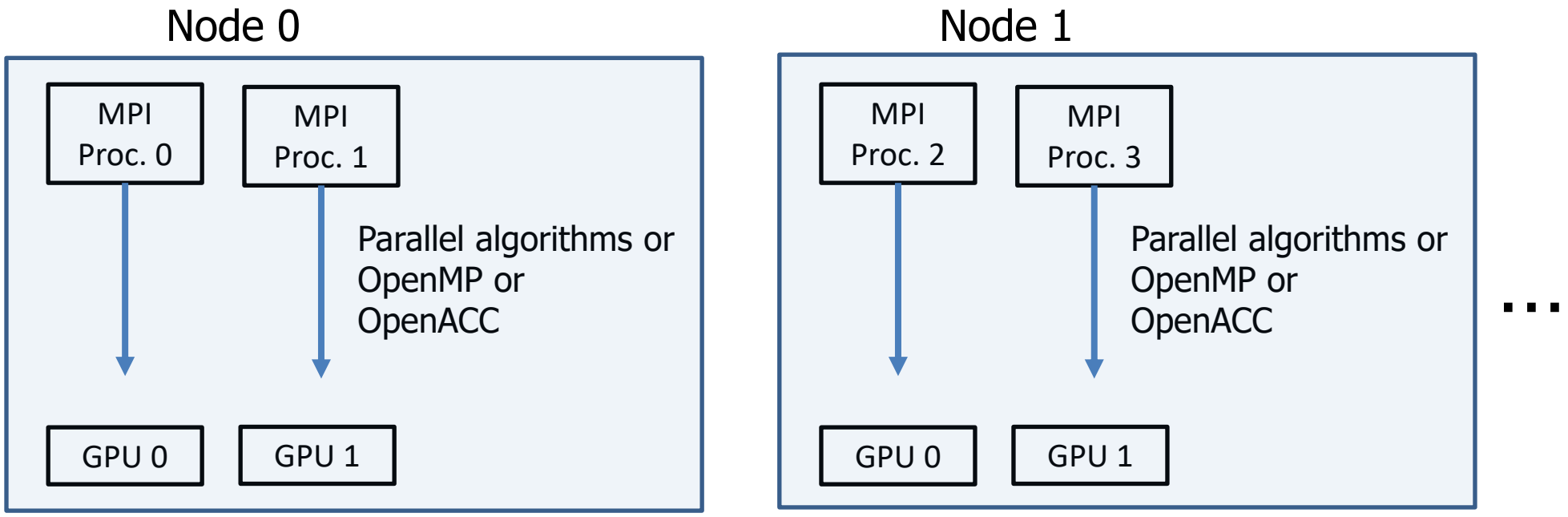
Our project: Hybrid parallelism

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE



Our project: Hybrid parallelism

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE



The test cases

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

1. Taylor-Green vortex [Uniform collision model, no boundary condition].
2. Resolved flow in a porous media [Mesh-aligned inflow and outflow, bounce-back nodes].
3. Multi-component flow segregation with pseudo-potential approach [Multi-phase coupling, no boundary condition].
4. Flow around a sphere (no mesh refinement) [Off-lattice boundary condition around the obstacle, subgrid-scale model].
5. Flow inside a tube (channel with circular cross-section) [Off-lattice boundary condition around the obstacle, subgrid-scale model].

Implementation attempt: C++17 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`

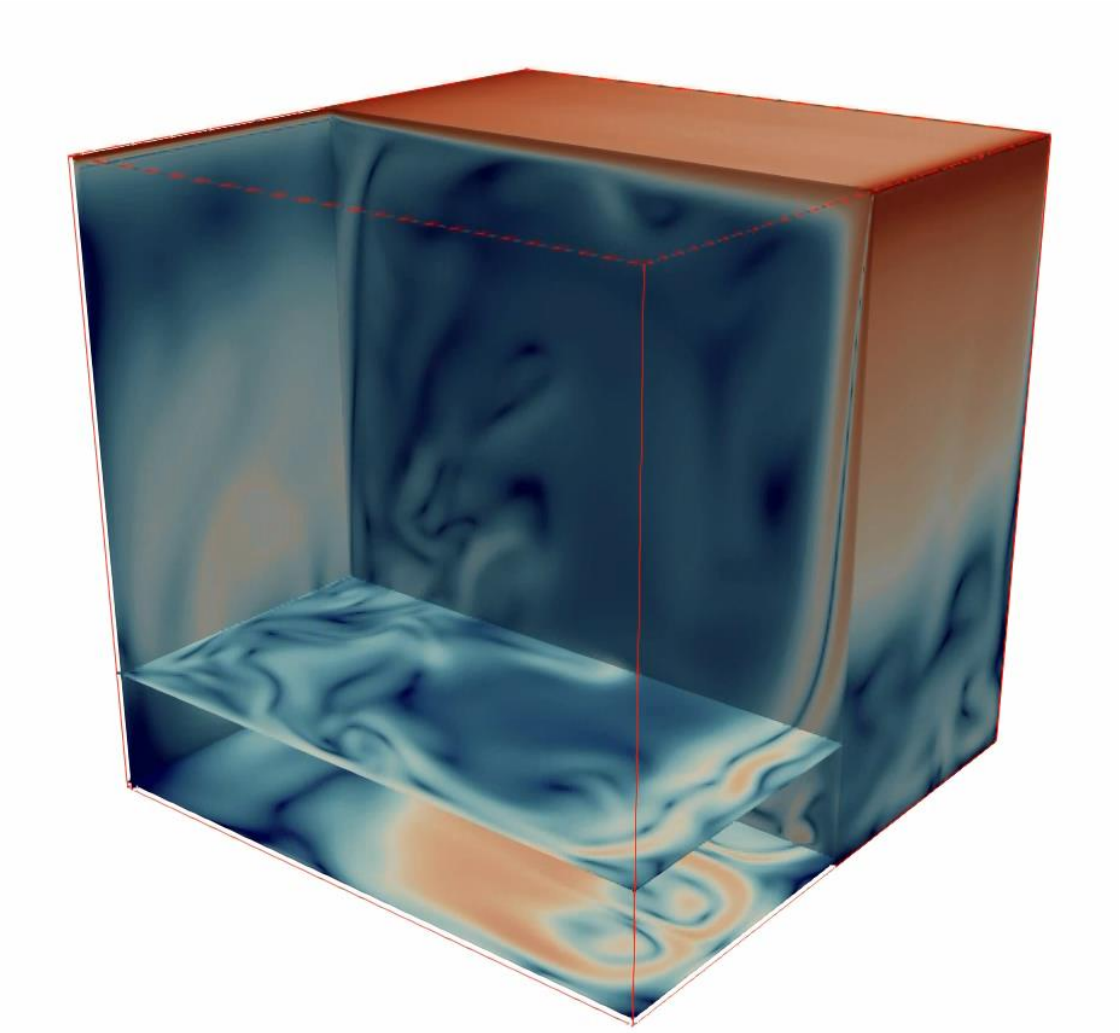
Lambda function: defines the operation to be applied to elements of `v`.

```
#include <execution>
```

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

Application: The STLBM project

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE



Reynolds: 10’000

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

400 x 400 x 400 domain (homogeneous mesh)

560k iterations

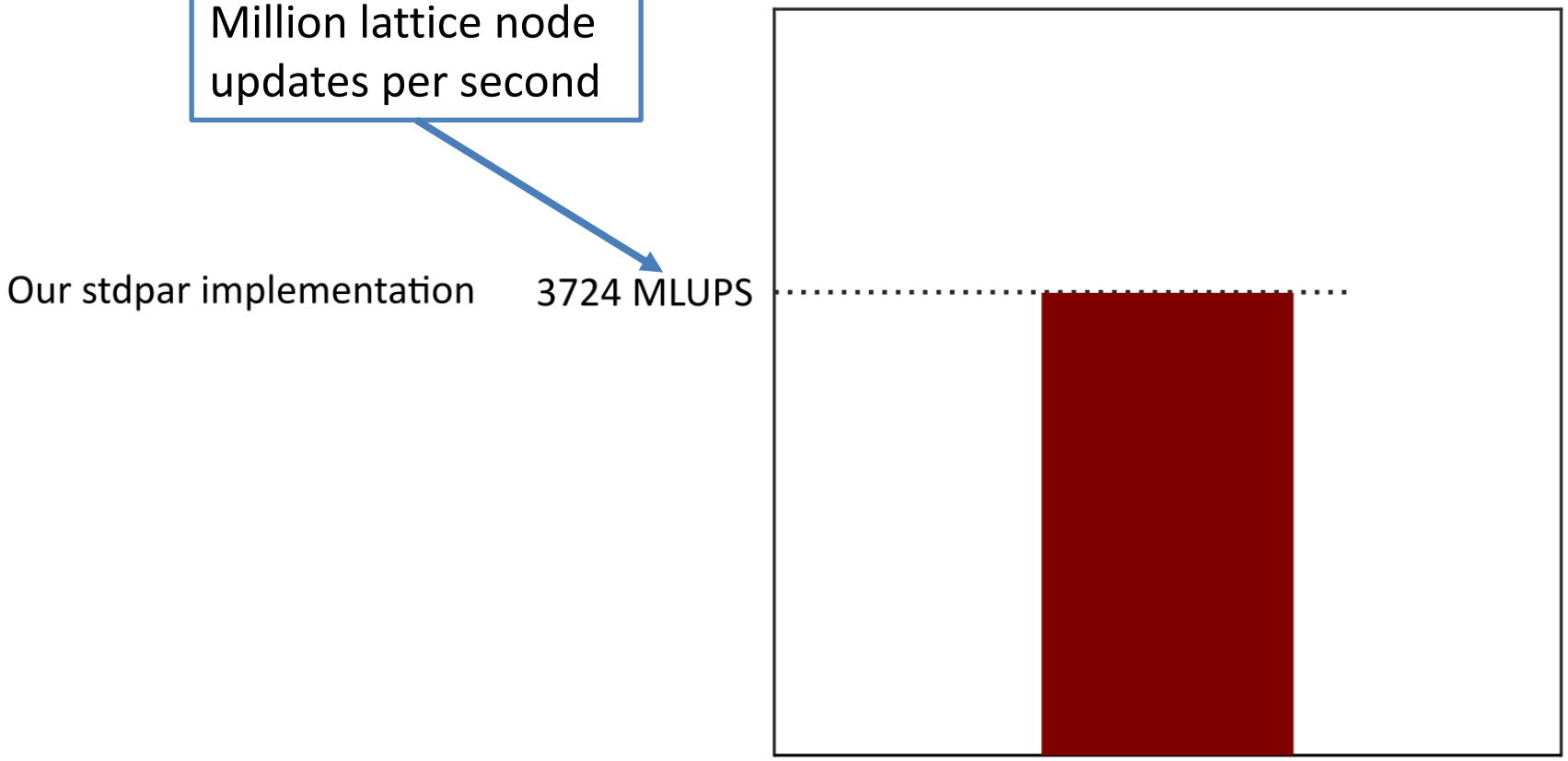
2:40 hours on a A100

https://gitlab.com/unigehpfs/stlbm

Performance

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

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

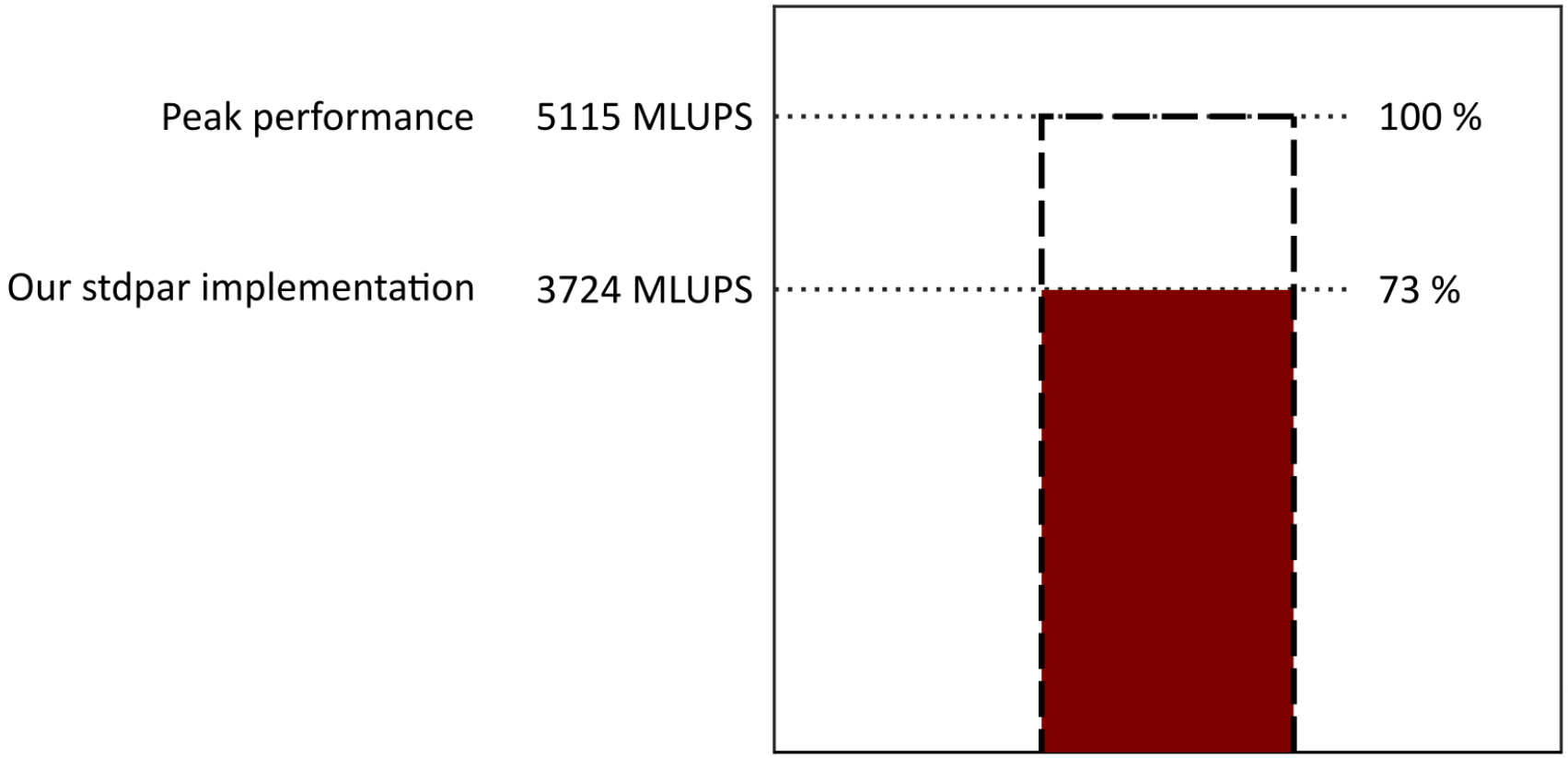


200x200x200 domain
Double-precision
A100 GPU

https://gitlab.com/unigehpfs/stlbm

Performance vs. Peak performance

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

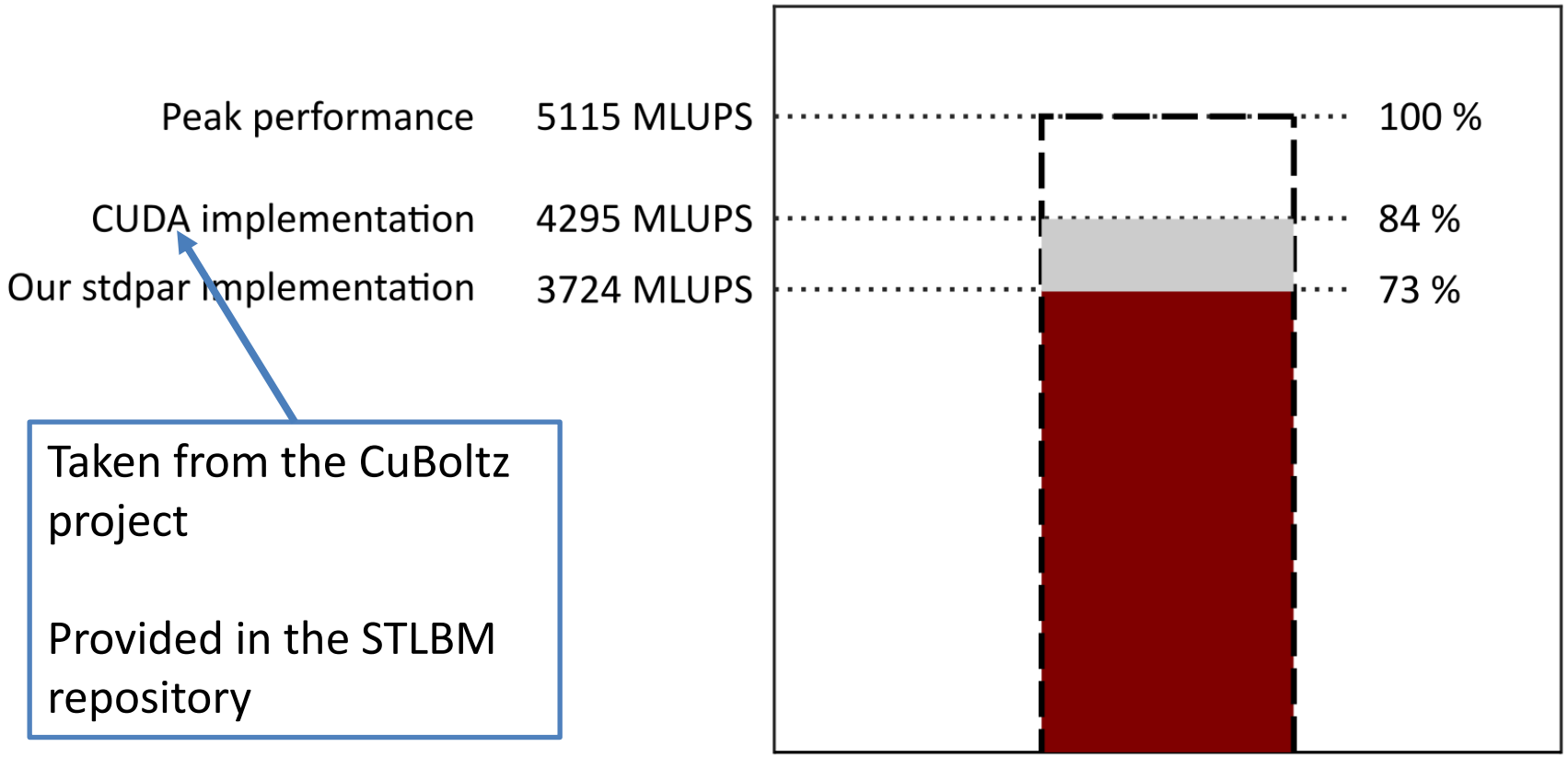


200x200x200 domain
Double-precision
A100 GPU

https://gitlab.com/unigehpfs/stlbm

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

https://gitlab.com/unigehpfs/stlbm

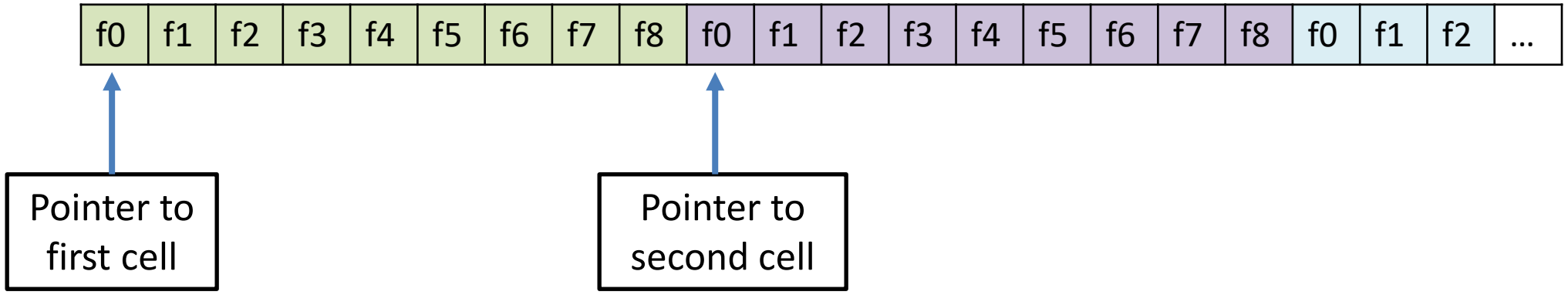
UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Some technical details

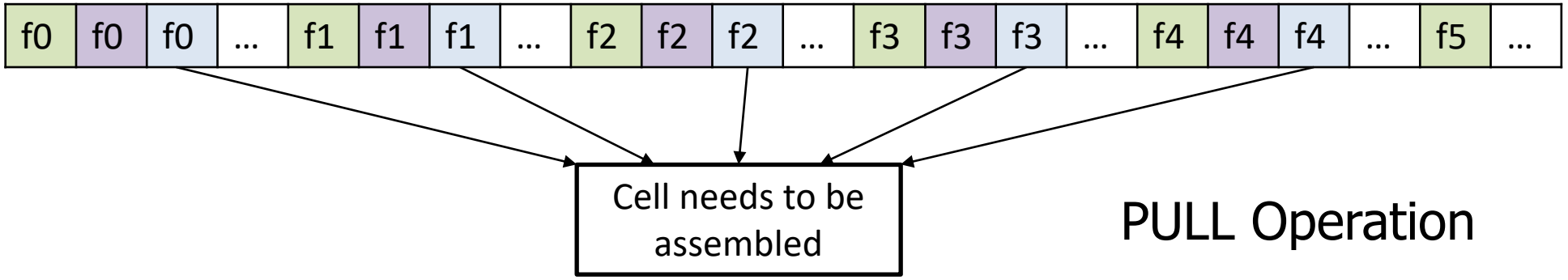
Data Layout

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Array-of-Structure



Structure-of-Array



PULL Operation

Data Layout

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

Array-of-Structure ("old Palabos")

```
Cell* cell = &lattice[i];
collide(*cell);
```

Structure-of-Array ("new Palabos")

```
Cell cell;
PULL(cell);
collide(cell);
PUSH(cell);
```

Also: we need the PULL and PUSH step because data is stored in different places at even and odd time steps (see description of "AA Pattern" in STLBM project).

Challenge: rewrite data processors

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

```
void BoussinesqThermalProcessor3D<T,FluidDescriptor,TemperatureDescriptor>::process (
        Box3D domain,
        BlockLattice3D<T,FluidDescriptor>& fluid,
        BlockLattice3D<T,TemperatureDescriptor>& temperature )
{

    for (plint iX=domain.x0; iX<=domain.x1; ++iX) {
        for (plint iY=domain.y0; iY<=domain.y1; ++iY) {
            for (plint iZ=domain.z0; iZ<=domain.z1; ++iZ) {
                T *u = temperature.get(iX+offset.x,iY+offset.y,iZ+offset.z).getExternal(velOffset);
                Array<T,FluidDescriptor<T>::d> vel;
                fluid.get(iX,iY,iZ).computeVelocity(vel);
                vel.to_cArray(u);

                T *force = fluid.get(iX,iY,iZ).getExternal(forceOffset);
                T localTemperature = temperature.get(iX+offset.x,iY+offset.y,iZ+offset.z).computeDensity();
                const T diffT = localTemperature - T0;
                for (pluint iD = 0; iD < D::d; ++iD)
                {
                    force[iD] = gravOverDeltaTemp[iD] * diffT;
                }
            }
        }
    }
}
```

Loop needs to be distributed to GPU threads

Cell access here assumes array-of-structure

Fortunately: data processors often refer to templated algorithms which we can reuse in the AcceleratedLattice.

Challenge: rewrite Dynamics classes

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

A typical Palabos line of code:

```
cell.getDynamics().collide(cell);
```

- Here, we access the dynamic type of the local collision model.
- This is dynamic polymorphism: every cell has its own collision model.
- Collision models can be nested, too.
- GPUs don't like polymorphism, or function calls through pointers, at all.

The only solution: write out the required collision terms in a non-polymorphic way.

Fortunately: dynamics classes often refer to templated algorithms which we can reuse.

Conclusions

UNIVERSITÉ DE GENÈVE
FACULTY OF SCIENCE

- Our project: port as much as possible of Palabos from CPU to GPU.
- Many unknowns: Parallel algorithms or OpenMP / OpenACC ? How much code needs to be rewritten ?
- A community project: stay tuned, try it out, interact.