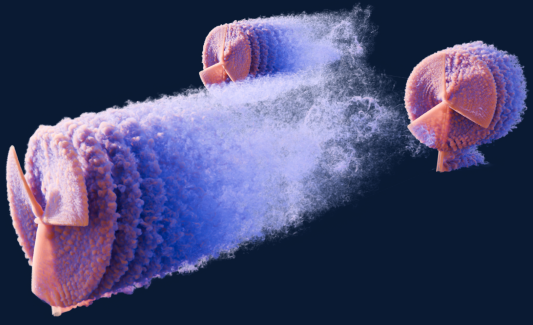
A Hardware Abstraction for Exascale Lattice Boltzmann Simulations: Rotor-Resolved Wind Farms on Aurora and LUMI



Adrian Kummerländer[a], Fedor Bukreev[a], Leonardo Dorneles[a,b], Marcio Dorn[b], Shota Ito[a], and Mathias J. Krause[a]

[a] Karlsruhe Institute of Technology, Germany [b] Universidade Federal do Rio Grande do Sul, Brazil

SC26 | November 17, 2026 | McCormick Place, Chicago, Illinois

KIT | NHR National High-Performance Computing Center

UFRGS Universidade Federal do Rio Grande do Sul

openlb.net

DFG

Falcon

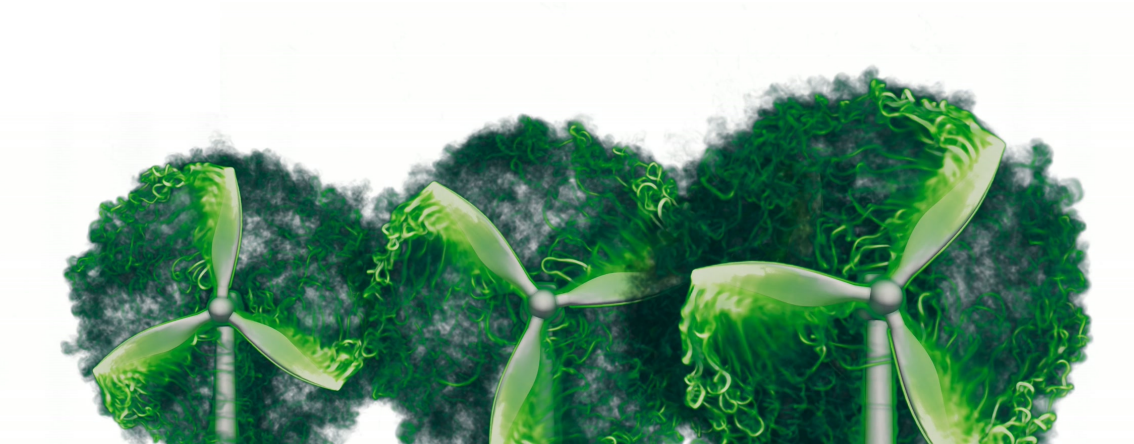
Funded by the European Union

A rotor-resolved wind farm, wall-modelled LES, on **12,288 Intel GPUs**

$5.94 \cdot 10^{12}$	$51.2 \cdot 10^{12}$	86%	3 vendors
cells	cell updates / s	weak scaling, $1 \rightarrow 2048$ nodes	one source

▶ **The number is not the contribution**

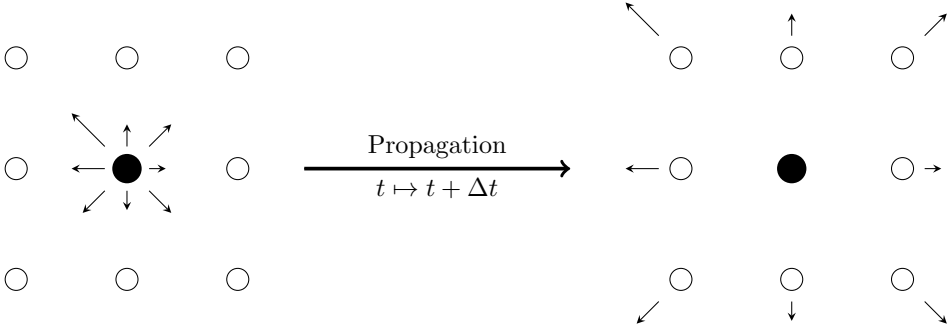
LES, wall model, turbulent inflow, torque integration: written **once**, against a cell concept. The same source compiles to **CUDA, HIP and SYCL**. This talk is about that abstraction.



Collide and Stream



Moving blades, wall model, turbulent inflow and torque, all **separate operators** over one uniform lattice, at **trillions of cells**.



Collide: perfectly local

Stream: nearest neighbour only

- **No global solve: every operator reads one cell or its nearest neighbours**

Three Vendors, Many Applications



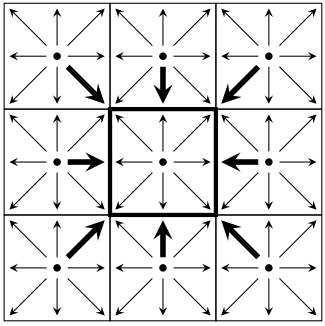
No shared accelerator, no shared compiler, no shared programming model. And not one application either: OpenLB ships **125+ examples** across 22 physics categories.

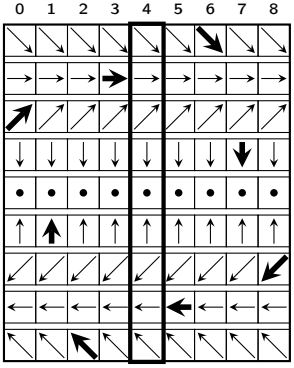
	Aurora (ALCF)	**LUMI-G** (EuroHPC)	**HoreKa Green** (NHR@KIT)
Nodes	10,624	2,978	128 benchmarked
GPUs per node	6× Intel Max 1550 (12 tiles)	4× AMD MI250X (8 GCDs)	4× NVIDIA A100
Programming	**SYCL / OneAPI**	**HIP / ROCm**	**CUDA**
Device memory	768 GB HBM2e	512 GB HBM2e	160 GB HBM2
Triad bandwidth	≈ 993 GB/s per tile	≈ 1.25 TB/s per GCD	≈ 1.40 TB/s per GPU
Interconnect	HPE Slingshot 11 (Dragonfly)		InfiniBand HDR200

Add CPU/OpenMP+AVX-512 and the naive answer is **four implementations** of every one of them, starting with a 4,878-expression collision kernel.

- **Nobody wants to maintain that four times, physics needs to be hardware-agnostic**

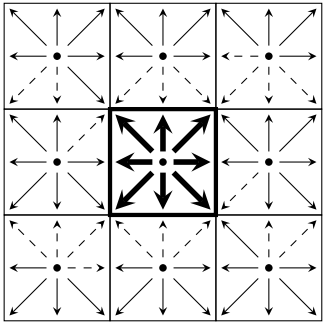
Implicit Propagation: Periodic Shift

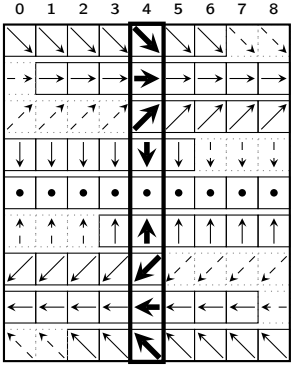




- **Propagation becomes a pointer rotation of cyclic population arrays**
  - No streaming step, no second population copy: every model sees a **static** lattice, never a swap
  - **CUDA, CPU**: two virtual address ranges onto one physical buffer, a hardware ring buffer
  - Each backend needed its own route to that view: see the appendix

Implicit Propagation: Periodic Shift





- **Propagation becomes a pointer rotation of cyclic population arrays**
  - No streaming step, no second population copy: every model sees a **static** lattice, never a swap
  - **CUDA, CPU**: two virtual address ranges onto one physical buffer, a hardware ring buffer
  - Each backend needed its own route to that view: see the appendix

Operators on Data



Model code is written against the **concept of a cell**: never a memory layout, a kernel launch, or a vendor.

Data described by **Fields**:

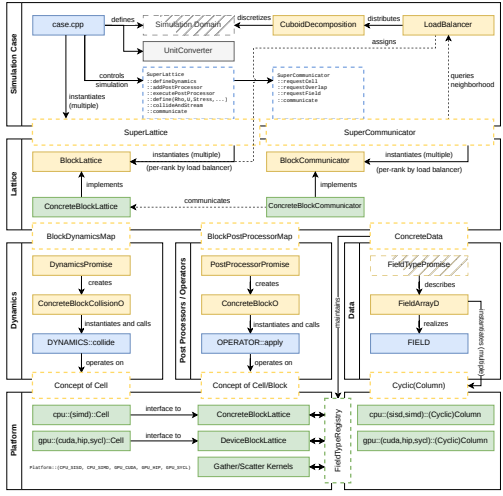
```
struct FORCE : FIELD_BASE<0,1,0> { };
struct OMEGA : FIELD_BASE<1,0,0> { };
```

Reached through **Cells** and **Parameters**:

```
cell[iPop] = ...;
auto force = cell.getField<FORCE>();
parameters.get<OMEGA>();
```

An **Operator** is a constrained template:

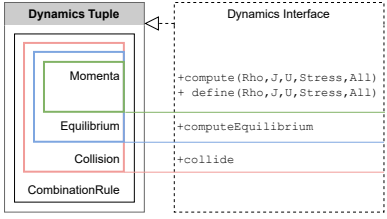
```
template <concepts::Cell       CELL,
          concepts::Parameters PARAMETERS>
CellStatistic<V> collide(CELL& cell,
                         PARAMETERS& params);
```

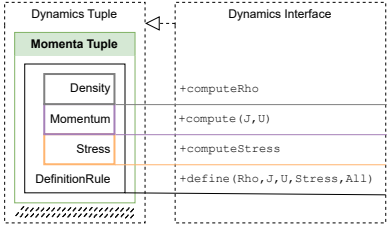


- The same `collide()` is instantiated for CUDA, HIP, SYCL and AVX-512

The Dynamics Tuple: a DSL for Cell Models

A cell model is a **type**: momenta × equilibrium × collision, composed at compile time.






```
using BGK = dynamics::Tuple<
  T, DESCRIPTOR,
  momenta::BulkTuple,
  equilibria::SecondOrder,
  collision::BGK
>;

using ForcedSmagorinskyBGK = dynamics::Tuple<
  T, DESCRIPTOR,
  momenta::BulkTuple,
  equilibria::SecondOrder,
  collision::SmagorinskyEffectiveOmega<collision::BGK>,
  forcing::Guo
>;
```

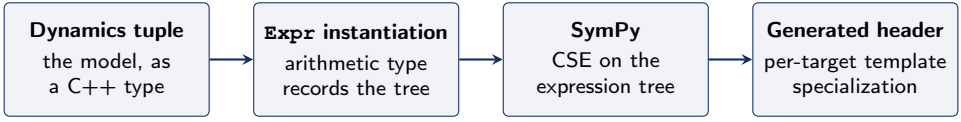
Per-cell, so disparate models coexist in one domain. Forcing is *injected* at compile time; the collision operator is not modified.

- **Because the model is a type, the compiler sees all of it**

Automated Common Subexpression Elimination



C++ cannot eliminate subexpressions across class and function boundaries, so OpenLB extracts the expression tree and does it externally.



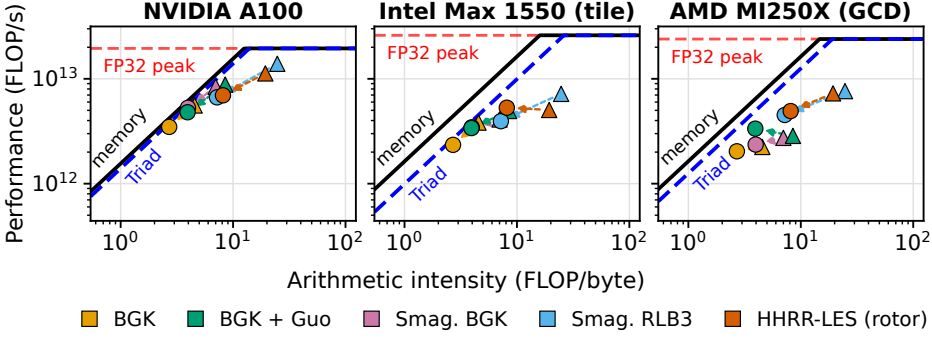
4,878 → 2,048
expressions in the production HHRRLBM-LES collision (−58%)

up to 2.5×
faster, on hardware the model source knows nothing about

- **Close-to-literature C++ in, algebraically optimized kernels out**

Roofline on Three Vendors

Generated collision kernels before (▲) and after (●) CSE



1.5–2.5×
CSE speedup of the rotor model on all three vendors

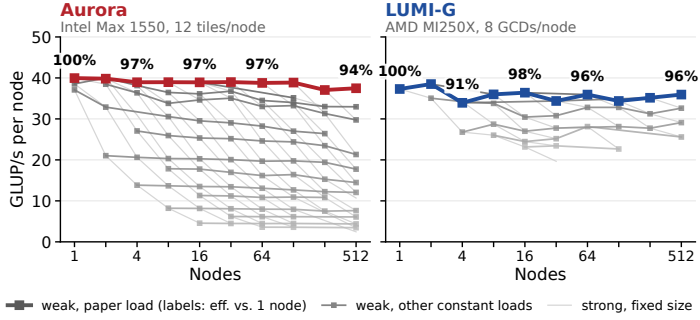
87%
of empirical Triad bandwidth (Intel Max 1550, BGK)

- **CSE moves the kernels toward the memory roof on all three vendors**

What the Abstraction Costs at Scale



Lid-driven cavity, single precision: the same generated kernels, two vendors.



93.8% to 512 nodes

$1.53 \cdot 10^{12}$ cells at 19.2 TLUP/s

96% to 512 nodes

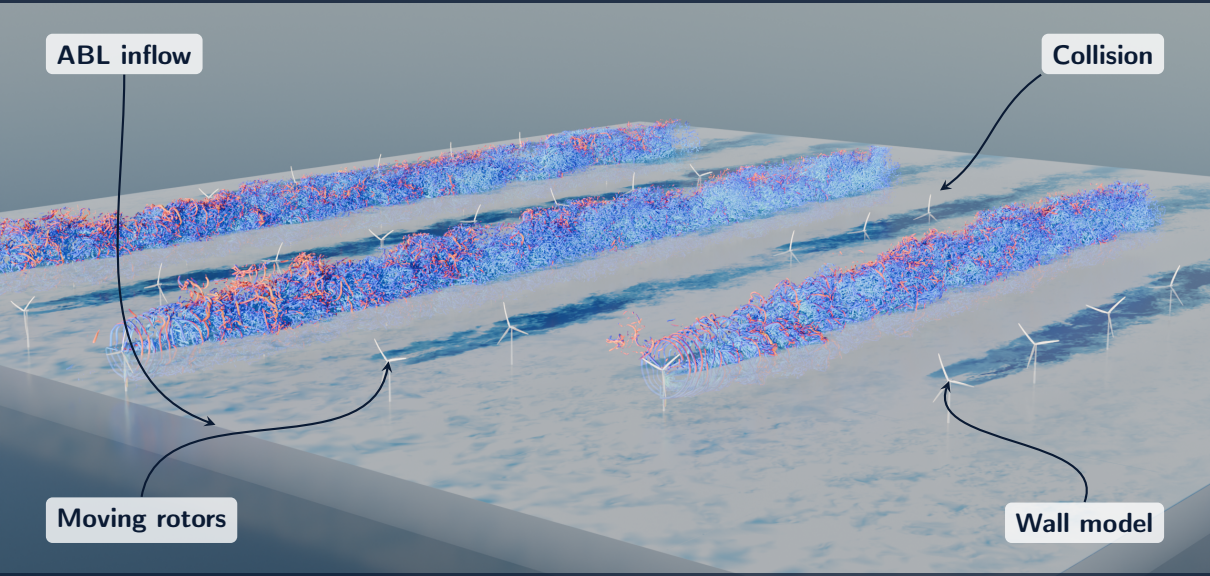
$1.73 \cdot 10^{12}$ cells at 18.4 TLUP/s

LUMI continues to 1024 nodes at **91%**: $3.46 \cdot 10^{12}$ cells, 34.6 TLUP/s.

- **The abstraction costs nothing at scale: halo exchange hides under the bulk**

The Production Case

Lillgrund: **48 rotor-resolved turbines**, 3.3–4.3 D spacing, one domain

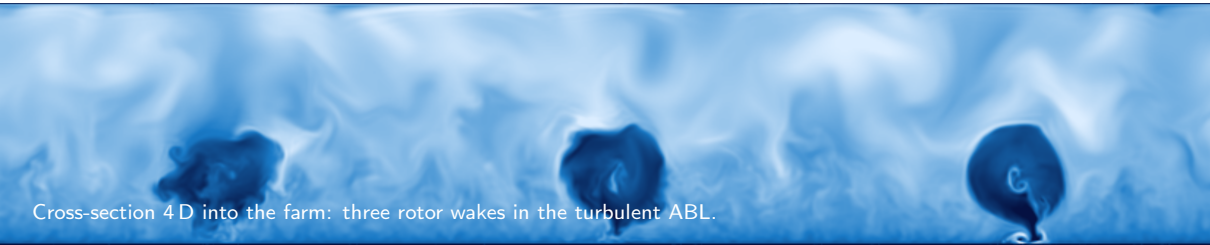


Not a plain lid-driven cavity: all four operators are active in every time step.

What Runs Every Step

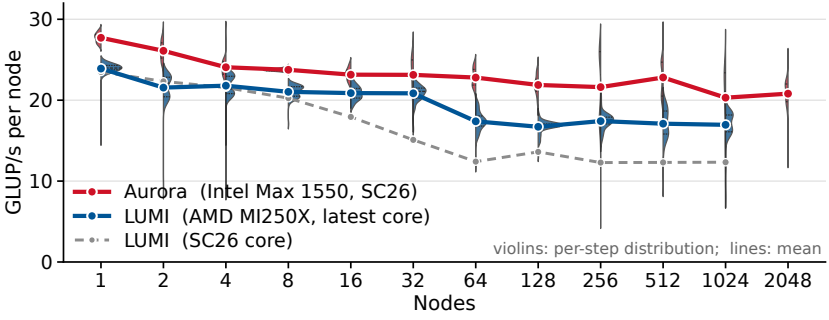
Four operators over one uniform lattice, each written against the cell concept.

Collision	HHRRLBM-LES, 3rd-order recursive regularized	a `dynamics::Tuple`
Wall model	Spalding law on the moving blade surface, Newton iteration	a cell-local operator
Moving rotors	Brinkman penalization: solids as lattice porosity, no remeshing	a `POROSITY` field
Turbulent inflow	volumetric vortex method, synthetic ABL eddies	a cell-point coupling



Cross-section 4 D into the farm: three rotor wakes in the turbulent ABL.

One Wind Farm, Two Supercomputers



Aurora 2048 nodes, 12,288 GPUs

$5.94 \cdot 10^{12}$ cells · 51.2 TLUP/s

86% weak scaling, $1 \rightarrow 2048$

101% from $64 \rightarrow 2048$

LUMI 1024 nodes, 8,192 GCDs

$1.48 \cdot 10^{12}$ cells · 17.4 TLUP/s

77% weak scaling, $1 \rightarrow 1024$

95% from $64 \rightarrow 1024$

- **One application, both machines: no architecture-specific model code**

Summary

3 vendors NVIDIA, AMD and Intel GPUs from one application source: models written against the *concept of a cell*

2.5× from automated CSE: the production collision generated down from 4,878 to 2,048 expressions

$5.94 \cdot 10^{12}$ cells at $51.2 \cdot 10^{12}$ updates/s: the largest GPU-accelerated LBM simulation reported to date

86% weak scaling to 12,288 GPUs on Aurora, 77% to 8,192 GCDs on LUMI, both from a single node at full WMLES physics



Adrian Kummerländer

kummerlaender@kit.edu

OpenLB Framework

www.openlb.net

Publications

SC26 A. Kummerländer, F. Bukreev, L. Dorneles, M. Dorn, S. Ito, M. J. Krause. *A Hardware Abstraction for Exascale Lattice Boltzmann Simulations: Rotor-Resolved Wind Farms on Aurora and LUMI*. 2026.

HPCSE '26 A. Kummerländer, L. Dorneles, F. Bukreev, N. A. Watson, J. Grafen, M. Dorn, M. J. Krause. *Lattice Boltzmann Methods for Locally Refined Wall-Modeled Large Eddy Simulations on Heterogeneous HPC*. To appear.

HPCSE '25 A. Kummerländer, F. Bukreev, Y. Shimojima, S. Ito, M. J. Krause. *Large-Scale Simulations of Turbulent Flows using Lattice Boltzmann Methods on Heterogeneous HPC*.

OMAE 2026 N. A. Watson, A. Kummerländer, F. Bukreev, M. J. Krause, I. Owen. *Assessing Helideck Turbulence of an Offshore Platform Using Delayed Detached Eddy Simulation and the Lattice Boltzmann Method*. 10.1115/OMAE2026-180944

IJNMHFF 2026 A. Kummerländer, S. Ito, M. Schecher, D. Dapelo, S. Simonis, M. J. Krause, F. Bukreev. *Efficient Wall-Modelled Large Eddy Simulation of Rotors Using Homogenized Lattice Boltzmann Methods*. 10.1108/HFF-09-2025-0724

CMAME 2026 A. Kummerländer, B. Tur, M. Haase, F. Bukreev, M. Döllinger, M. J. Krause, S. Kniesburges. *Efficient Fluid Structure Interaction Simulation of Vocal Fold Oscillations Using a Homogenized Lattice Boltzmann Method*. 10.1016/j.cma.2026.119009

JPDC 2025 A. Kummerländer, F. Bukreev, D. Teutscher, M. Dorn, M. J. Krause. *Optimization of Single Node Load Balancing for Lattice Boltzmann Methods on Heterogeneous HPC*. 10.1016/j.jpdc.2025.105169



10th Spring School on LBM
Erlangen, Germany
5–9 April 2027
www.openlb.net