Hila: Portable Lattice Field Theory Code Generator Enabling Advanced GPU Communication Through MPI and NCCL/RCCL

Aaron Haarti† & Kari Rummukainen
Computational Field Theory group, University of Helsinki

Background & Objectives

The Exascale Landscape

- Modern HPC is dominated by Arm and heterogeneous GPU architectures.
- Porting legacy Lattice Field Theory (LFT) code is a bottleneck for research.
- **Goal:** Achieve high performance without sacrificing code portability.

The `hila` Pre-processor

- Transforms high-level mathematical representations into architecture-specific code.
- **Supports:** x86, Arm, OpenMP, AVX, CUDA, HIP, and MPI.
- Designed for easy integration of new hardware architectures.

Research Objectives

- Integrate **NCCL/RCCL** for multi-GPU point-to-point communication.
- Implement and evaluate **computation-communication overlap**.
- Benchmark against **GPU-Aware MPI** implementations.
- Evaluate performance for general stencil point-to-point communication.

The hila Pre-processor

We design an LFT code generator which maps mathematical expressions to optimized C++ code. For example, the Wilson gauge action:

$$S_g[U(x)] = \frac{\beta}{N} \sum_{x,\mu<\nu} \Re \operatorname{Tr}[\mathbb{1} - U_\mu(x) U_\nu(x+\hat{\mu}) U_\mu(x+\hat{\nu})^\dagger U_\nu(x)^\dagger].$$

This expression is a vectorizable operation at and a reduction over all lattice points. Hila captures this structure via a DSL:

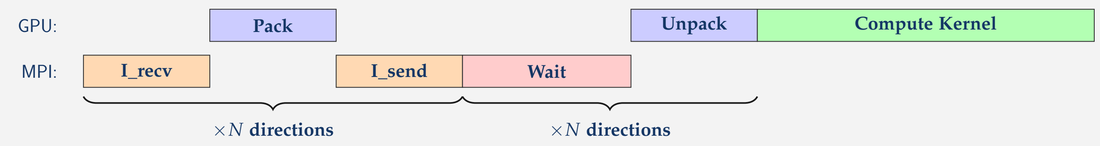
```
GaugeField<SU<N, double>> U;
... // initialize and update gauge field
Reduction<double> plaq = 0;
foralldir(mu) foralldir(nu) if (mu < nu) {
    onsites(ALL) {
      SU<N,double> U_loop = U[mu][X]*U[nu][X + mu]*U[mu][X + nu].dagger()*U[nu][X].dagger();
      plaq += (1.0 - real(trace(U_loop))) / N;
    }
  }
}
```

- **Mapping:** The `onsites(ALL)` is a **1-to-1 abstraction** to $\sum_x$.
- **Optimization:** Translates a vectorizable operation over X into **hardware-specific C++** such as AVX, HIP, CUDA.
- **Communication:** Complex data exchange patterns are hidden.
- **Flexibility:** Balances **prototyping** without compromising on performance.

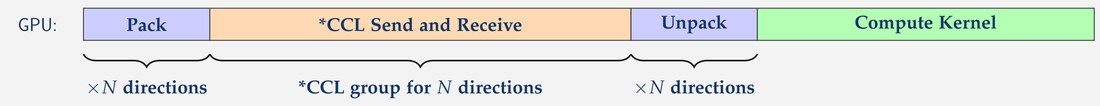
Hila is a mature framework actively used in production [1–4].

GPU communication schemes

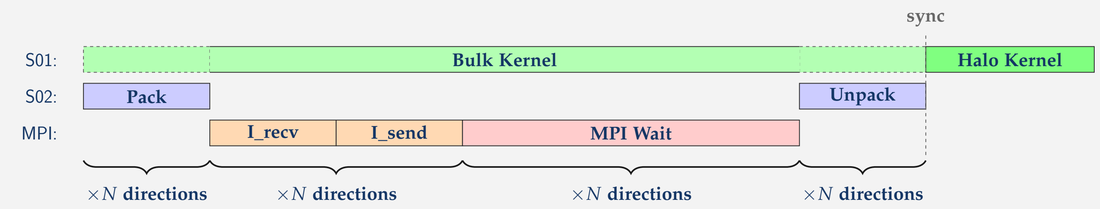
0. Standard Method (MPI)
Asynchronous GPU-aware MPI where memory packing, transfer, and unpacking must complete before computation begins:



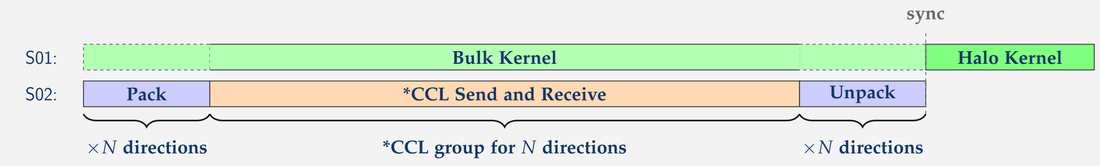
1. NCCL/RCCL
Direct interchange of MPI calls for GPU-native calls (NCCL/RCCL).



2. Overlap Scheme (Kernel Bifurcation)
Communication is hidden by bifurcating the compute kernel into independent **bulk** and **halo** kernels [5].



3. GPU Native Overlap Scheme
Includes both *CCL and kernel bifurcation. Utilizing native device-initiated networking via GPU streams for easy overlap [6].



UNIVERSITY OF HELSINKI †aaron.haarti@helsinki.fi

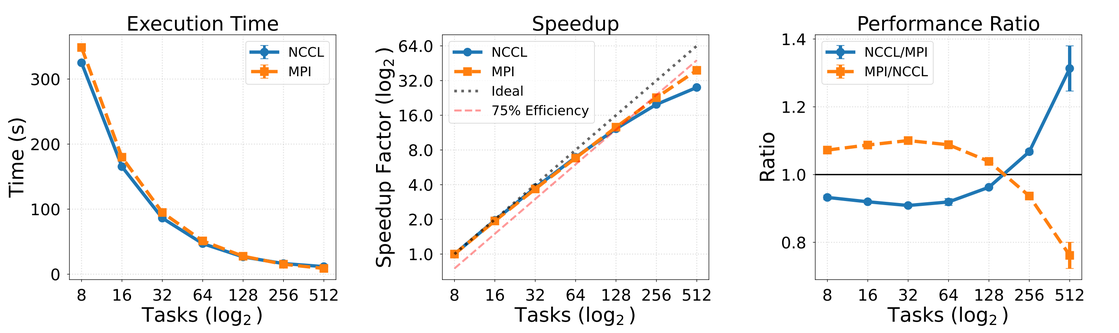
Results

***Benchmark**: Standard Model $SU(3)$ gauge field simulation with halo regions requiring point-to-point communication. $SU(N)$ gauge field simulations are heavily memory bound. Hila assigns one task per GPU.*

NCCL vs GPU-Aware MPI

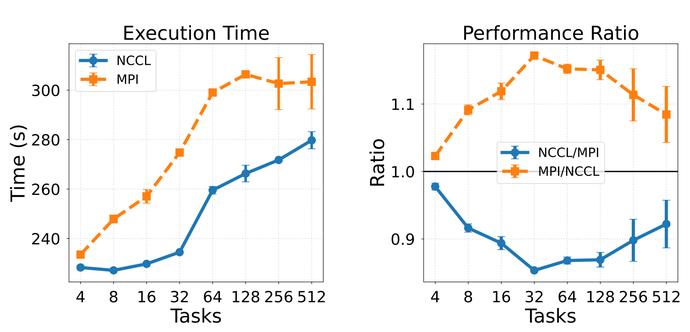
Experiments / tests have been performed on CSCS Alps system (GH200 96GB, Slingshot, MPICH, CUDA 12, aws-ofi-nccl, LL128 protocol) and CSC LUMI (MI250x, ROCm 6.0, Slingshot, MPICH, aws-ofi-rccl, LL128 protocol):

NCCL Strong scaling (Inter node): $V = \{256, 256, 256, 46\} \sim 90$GB/GPU at 8 tasks



- Starting at 8 tasks so that communication is done over network.
- Intra-node variance $\sim 1\%$ (statistically inconclusive) NCCL/MPI performance very similar.
- 8–64 tasks 13–103 MB messages: NCCL achieves 1.07×–1.10× speedup over MPI.

NCCL Weak scaling: lattice points per GPU constant ($\sim 65GB$ per GPU)



Message Size Distribution

Tasks (N)	36MB (%)	72MB (%)	144MB (%)	Total Msgs
4	95.1	4.9	0.0	12,313
8	95.1	4.9	0.0	18,470
16	63.4	35.0	1.6	18,417
32	31.7	65.0	3.3	18,451
64	23.3	24.6	52.1	50,100
128	23.3	24.6	52.1	50,113
256	23.4	24.6	52.1	50,096
512	23.4	24.6	52.1	50,122

- In weak scaling NCCL displays up to 1.17× speedup.
- **16 → 32 tasks** where NCCL scales better indicates that NCCL has available bandwidth.
- Algorithmic caveat triggers surge in message volume and size at 64 Tasks.

NCCL outperforms GPU-aware MPI for large messages ($>\sim 13$ MB), explained by NCCL subdividing point-to-point transfers into parallel channels, maximizing aggregate throughput [7]. RCCL performs $1.2\times$ – $3\times$ slower as tasks increase, we expect implementation maturity to be at fault. Further tests will be done following ROCm 6.3 LUMI upgrade (January 2026).

Overlap Scheme

Experiments / tests have been performed on NVIDIA "Thea" MGX Evaluation System, operated by the HPC Advisory Council. (GH200, CUDA 13, OpenMPI) and CSC LUMI (MI250x, ROCm 6.0, Slingshot, MPICH, aws-ofi-rccl, LL128 protocol):

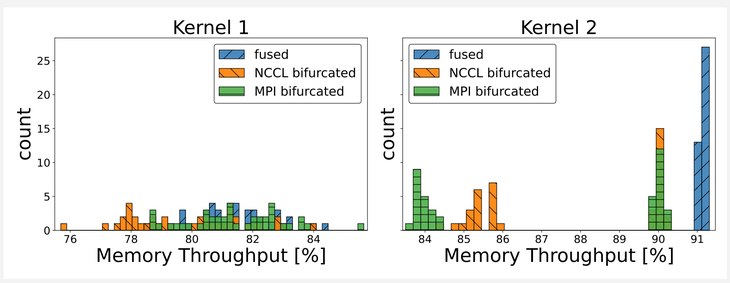


Figure 1: fused vs. bifurcated kernel memory throughput histogram (Thea)

	2 Tasks		4 Tasks	
	Time (ms)	Ratio	Time (ms)	Ratio
Default	31.265(40)	1.000	35.349(41)	1.000
MPI	30.870(74)	1.013(3)	34.342(14)	1.029(1)
NCCL	30.712(36)	1.018(2)	34.215(113)	1.033(4)

Table 1: standard method vs overlap (Thea)

- **Kernel 2: Saturates memory bandwidth.**
  - Bifurcation introduces overhead.
  - Resources are yielded to overlapping NCCL/MPI calls (branching in Fig. 1).
- **Kernel 1: Under-saturated memory bandwidth**
  - No significant overhead produced from bifurcation.
  - Tolerates overlapping communication calls without throughput degradation.
- Speedup is achieved when overhead < T_{comm}.
- Speedup increases with task count (Table 1).
- With ROCm 6.0 overhead causes critical performance penalty.

Conclusions

NCCL vs. GPU-Aware MPI

- NCCL outperforms MPI for messages $>\sim$13 MB, achieving 1.07×–1.17× speedup due to higher throughput.
- Intra-node NCCL and MPI performance are statistically equivalent.
- RCCL shows no performance gain; we expect this to be implementation immaturity.

Overlap Scheme & Platform Analysis

- Overlap is beneficial when communication time > bifurcation overhead.
- Bandwidth-saturated kernels suffer from bifurcation and must yield resources to NCCL/MPI calls.
- Under-saturated kernels tolerate overlap.
- Bifurcation on ROCm 6.0 shows performance penalty.

Further research will follow the ROCm 6.3 LUMI upgrade (Jan 2026) and on the upcoming CSC Roihu GH200 system (April 2026).

References

1. CFT-HY Group. *HILA: High-Performance Lattice Field Theory Library* `https://github.com/CFT-HY/HILA`. 2025.
2. Rindlisbacher, T. *et al. Phys. Rev. D* **112,** 114507 (2025).
3. Annala, J. *et al.* (June 2025).
4. Correia, J. *et al. Phys. Rev. D* **111,** 063532 (2025).
5. Drach, V. *et al.* (Mar. 2025).
6. Hamidouche, K. *et al. GPU-Initiated Networking for NCCL* 2025.
7. Hu, Z. *et al. Demystifying NCCL: An In-depth Analysis of GPU Communication Protocols and Algorithms* 2025.