NCCLbpf: Verified Policies for GPU Collectives

Verified, Composable Policy Execution for GPU Collective Communication

Yusheng Zheng · University of California, Santa Cruz

Introduction: Policy hooks without native plugin risk

NCCL plugins choose algorithms, protocols, and channel counts inside the training process.

Native plugin code can dereference invalid pointers or corrupt state.

Policy updates normally require restarting affected jobs.

NCCLbpf brings verified userspace eBPF policies to existing NCCL plugin interfaces.

Background and Motivation: Ranks must agree

All ranks must choose compatible collective behavior.

Identical bytecode can still produce different choices when it reads rank-local inputs.

Policy decisions that require agreement must depend on replica-shared inputs.

The paper applies load-time taint analysis to reject dependencies on rank-local state.

Design: Separate policy from collective mechanism

Layer	Responsibility
NCCL	Existing collective algorithms and transport
Native plugin host	Plugin ABI, context preparation, output validation
bpftime runtime	Verification, JIT compilation, typed maps
eBPF policy	Algorithm, protocol, and channel decisions

NCCLbpf chooses among NCCL’s built-in algorithms.

Design: Threat model and architecture

The policy is untrusted; the **native host, verifier, and JIT** form the trusted boundary.

- NCCL calls the native plugin ABI.
- The host prepares context and validates outputs.
- Verified eBPF policies choose algorithms, protocols, and channels.
- Rank-sensitive choices require replica-shared inputs.

The policy cannot invent a new collective algorithm.

Design: Verification boundaries

Load-time checks

- Memory and stack safety
- Bounded execution and allowed helpers
- Rank-local input dependencies

Runtime host checks

- Cost-table dimensions and channel limits
- Preservation of unavailable algorithm/protocol combinations

A verified policy can still make a poor performance decision.

Implementation: Three plugin paths

The prototype uses bpftime in **tuner**, **profiler**, **and net** plugin paths.

- Tuner policies choose supported NCCL combinations.
- Profiler policies write typed telemetry maps.
- The net prototype wraps the **Socket backend**.

Policies can be verified, JIT-compiled, and swapped without changing NCCL source.

Evaluation: CPU dispatch overhead

One million calls per policy on an AMD EPYC 9575F.

Policy	P50	Increase over native
Native	20 ns	—
No-op / size-aware	100 ns	+80 ns
Lookup only	130 ns	+110 ns
Lookup and update	140 ns	+120 ns
SLO enforcer	150 ns	+130 ns

GPU end-to-end overhead is a separate measurement.

Evaluation: End-to-end GPU overhead

On **8 B300 GPUs with NVLink**, the measured small-message overhead is about **1.3 µs**, around 4% of a 32 µs baseline.

At **4 MiB and above**, measured overhead is **below 0.1%**.

CPU dispatch nanoseconds and GPU collective latency answer different questions.

Evaluation: Safety and hot reload

1. Verify the replacement.
2. JIT-compile it.
3. Atomically swap the active function pointer.
4. Retain the old version until in-flight calls drain.

Measurement	Reported result
Total reload, including verification and JIT	~9.4 ms
Hot-path pointer swap	1.07 **µ**s
Calls lost during the invocation test	0 / 400,000

Evaluation: Composable profiler and tuner

The profiler records telemetry in a typed eBPF map.

The tuner reads shared state to adapt its policy.

The composability experiment:

- Starts at **2 channels**
- Ramps to **12** over 100,000 calls
- Returns to **2** under an injected 10× latency spike
- Ramps back after recovery

Evaluation: Message-size sweep

8 NVIDIA B300 GPUs, NVLink 5, NCCL 2.29.7, CUDA 13.0

Message size	Default NVLS	Ring	Ring change
8 MiB	196.3 GB/s	249.7 GB/s	+27.2%
32 MiB	349.3 GB/s	402.4 GB/s	+15.2%
128 MiB	596.9 GB/s	628.9 GB/s	+5.4%
256 MiB	656.5 GB/s	632.5 GB/s	−3.7%

A global `NCCL_ALGO=Ring` setting loses the large-message advantage.

Evaluation: Size-aware AllReduce policy

The policy delivers **5.5–26.5%** higher AllReduce throughput in the evaluated **4–128 MiB** range.

It matches the default outside that targeted range.

Results use steady-state measurements after 2–3 warmup communicator creations.

Counterexample: a verified one-channel policy degrades throughput by **87–95%**.

Related Work: Where NCCLbpf sits

- **MSCCL / TACCL:** synthesize or tune collective schedules ahead of time.
- **AutoCCL:** adapts policy using native code without this verifier boundary.
- **gpu_ext:** extends at a lower driver layer.

NCCLbpf: a verified policy layer at existing NCCL plugin interfaces.

Discussion: Scale and transport limits

The safety suite accepts **7 safe programs** and rejects **7 unsafe programs**.

Current evidence comes from a **single node with 8 GPUs**.

Still to establish:

- Multi-node behavior with InfiniBand and larger rank counts
- Deeper integration with RDMA-aware policies
- Production-scale policy distribution and validation

Conclusion: Verified policy hooks for NCCL

Keep the collective implementation; make its policies verifiable and replaceable.

NCCLbpf combines load-time checks, typed state sharing, and atomic local replacement.

Paper source · Implementation

Questions?