

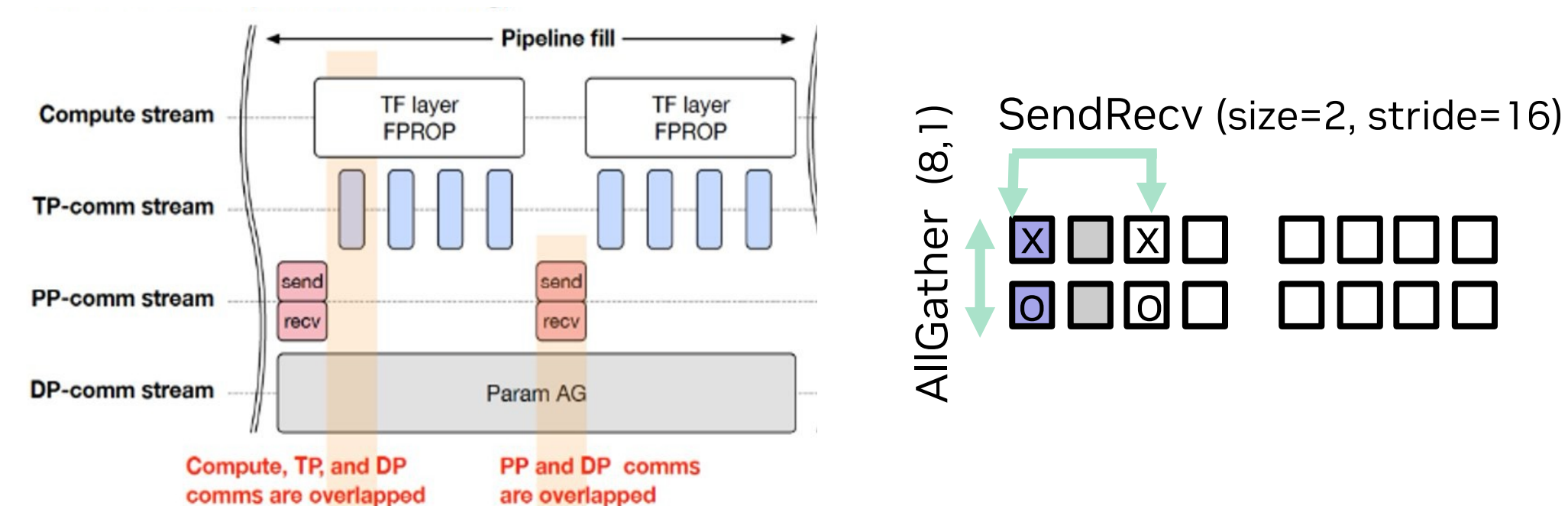
The problem

Microbenchmarking operations yields optimistic results

In real workloads, operations:

- Collide both network-wise and GPU-wise
- Run on complex topologies/ranks partitioning

For example, models frequently run many communication collectives, on a different partition of ranks:

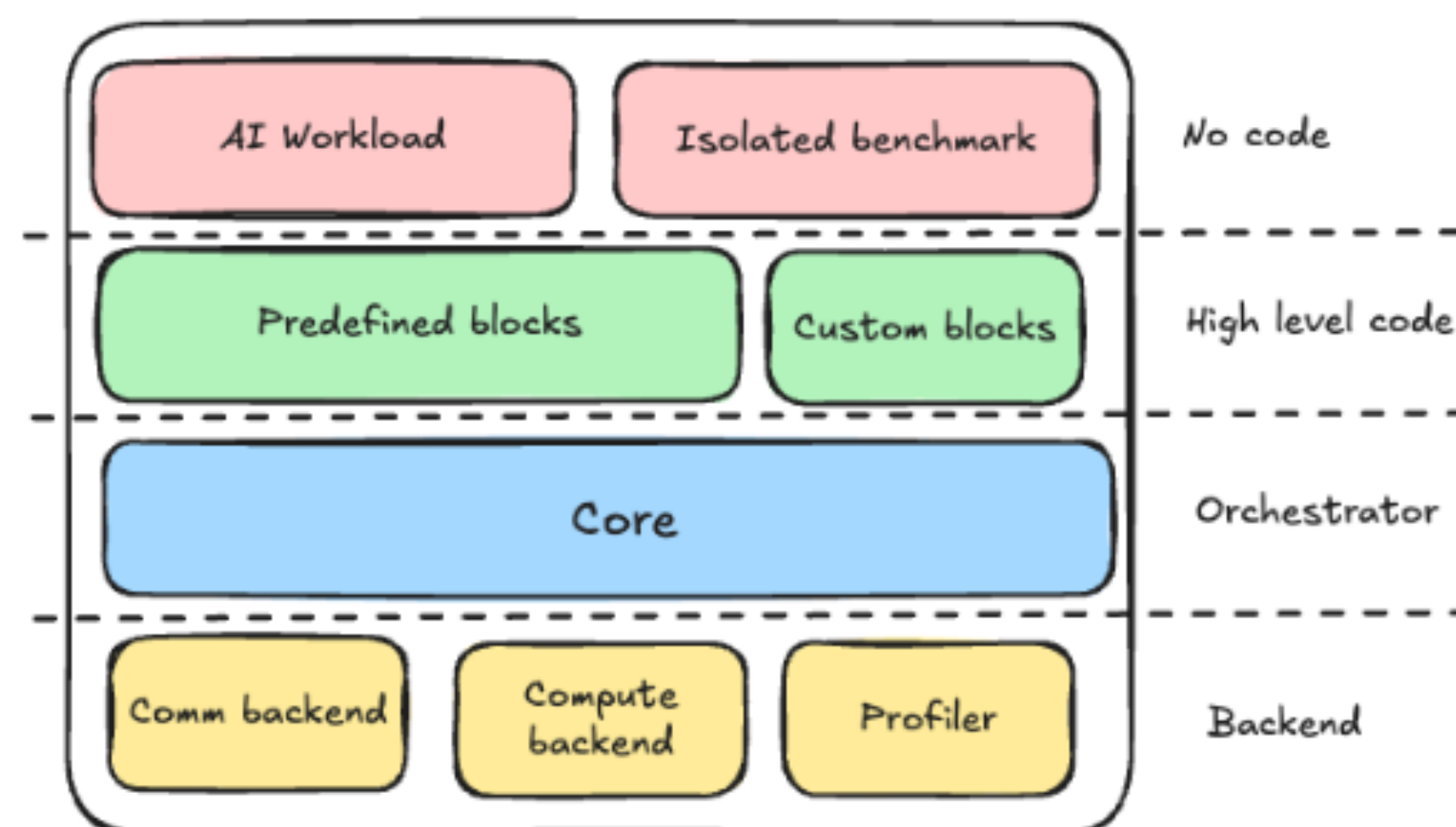


The solution

Maestro: a PyTorch based framework that allow a user to easily define any multi-dimensional workloads and analyze the behavior of each primitive.

By primitive, we mean:

- Communication (collectives, deepEP, etc..)
- Compute (cuDNN, cuBLAS, cutlass, etc..)
- High-level modules (torch.NN modules, Megatron blocks, FlashInfer kernels, etc..)



Goals

	Communication Engr.	AI researcher	Marketing
Goal	Maximize collective performance on realistic end-to-end workloads	Identify best overlap opportunities	Quantify and communicate added value on realistic patterns
Mean	Analyze primitives behavior under complex overlap schedules	Study experimental patterns performance	Compare “classic” patterns across environments
Stack used	Pre-configured blocks, edit the config file	Custom experimental blocks	No development: uses config files provided by AI experts

Input & Output

```
axes:
  ep:
    team_size: 4
    team_stride: 1
  dp:
    team_size: 4
    team_stride: 4

patterns:
  - name: "moe+allgather"
    ops:
      - block: "megatron_moe_layer"
        axis: ep
        name: "moe_layer_1"
        batch_size: 100
        is1: 4096
        hidden_size: 2048
        topk: 2
        dispatch_type: "allgather"
        num_experts: 8
      - block: "torch_all_gather"
        axis: dp
        name: "dp_allgather"
        vector_size: 250M
```

Define axes – “how to partition the ranks”: teams are created based on a rule

Tied to a GPU stream

Define blocks – Blocks are reusable units that enqueue a GPU kernel

They are the “primitives” that we want to benchmark

“Famous” blocks are pre-configured in Maestro

Custom blocks are easy to implement

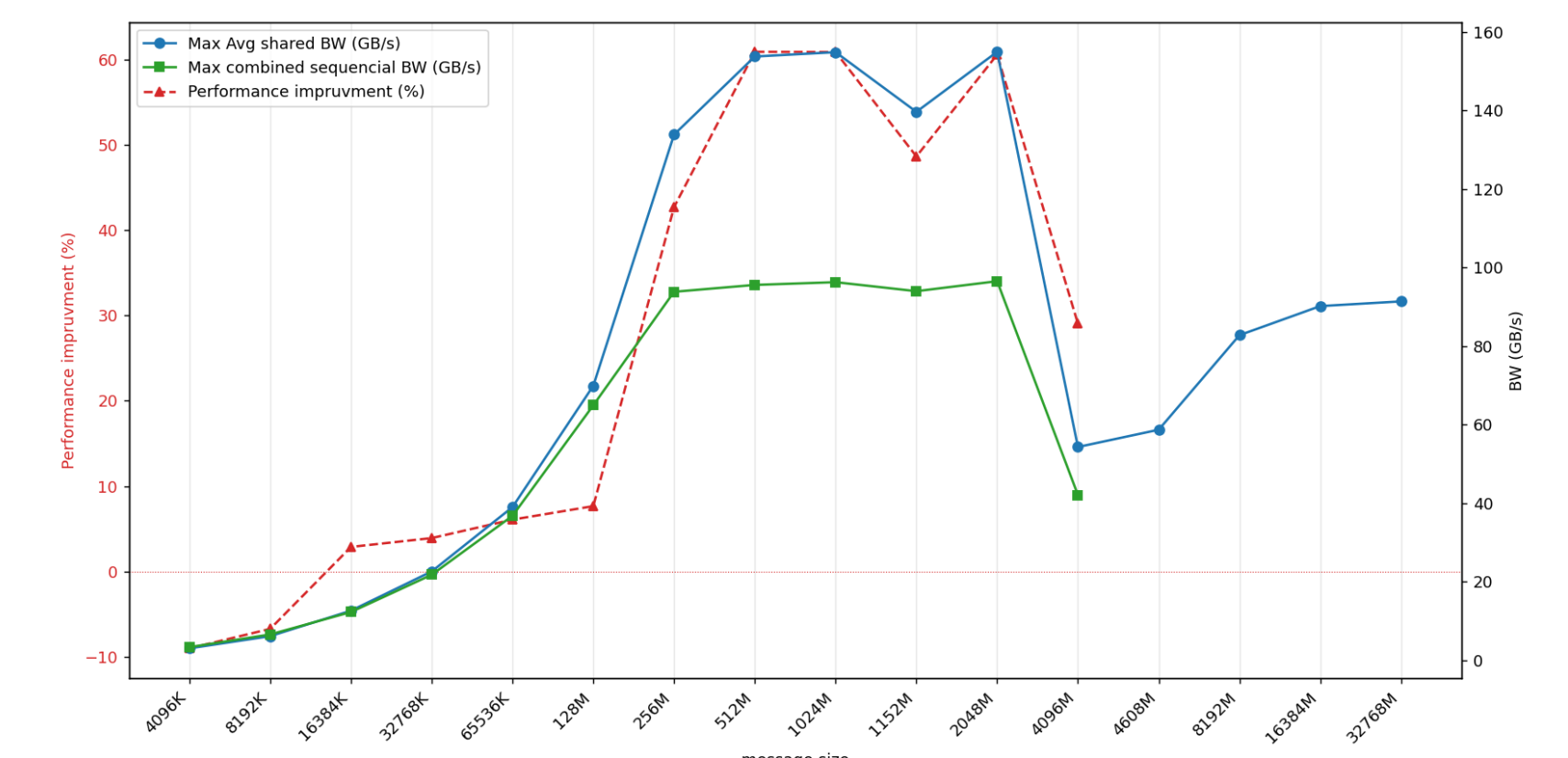
Pattern: moe+allgather	Block	Size (B)	Avg latency (ms)	Min lat. (ms)	Max lat. (ms)	Avg BW (GB/s)
	moe_layer_1	N/A	40.1	38.5	41.4	N/A
	dp_allgather	4G	87.5	74.8	127.1	49.9

Performance metrics are reported per block

Example Use Cases

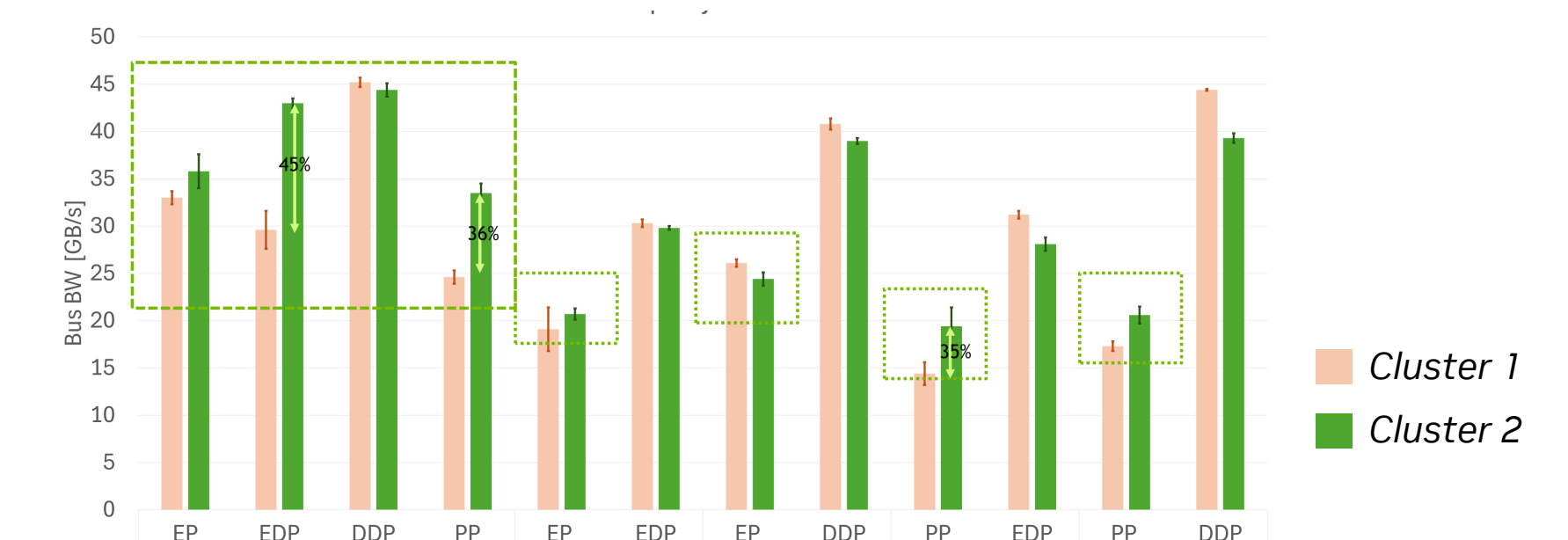
1. Benchmarking performance of overlapping optimizations

All-Gather + Reduce-Scatter is a very frequent overlapping pattern in AI workloads, SHARP is one optimization that allow both of the collectives to run on full bandwidth, maestro was used to prove the performance of this algo



2. Comparing two fabrics

Fabric hardware and configuration can change a lot when it comes to handling complex load, maestro allows to easily compare how the same pattern perform on two different fabrics, here is an example graph of a multi-dimensional workload comparison on two different clusters:



3. Measuring impact of network prioritization (PG's traffic_class)

We ran twice the same pattern - Overlapping AllToAll + Reduce Scatter But we changed which one gets the priority, here is the performance of each collective, each cell is one run of the two collectives overlapping:

