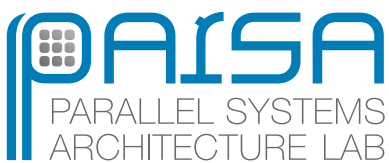
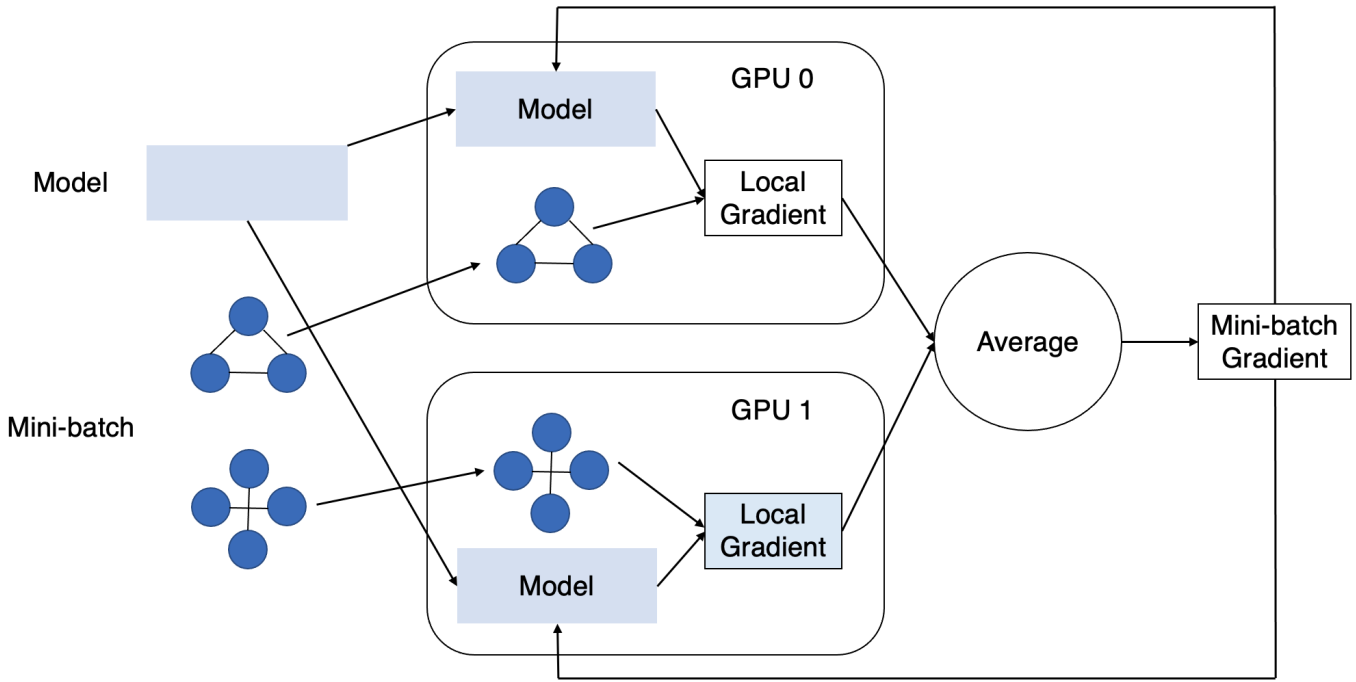
Blink: Fast And Generic Collectives for Distributed ML



Background: Training with Multiple GPUs



Inter-GPU communication becomes a more severe problem as model grows!

Elevator Pitch

PARALLEL SYSTEMS ARCHITECTURE LAB

- Cloud GPUs are allocated unaware of their topology
  - NCCL has bad topology policy: ring or nothing
  - NVLinks are commonly wasted
- Blink
  - Transparent
  - Dynamically generate efficient collective primitives
  - Better link utilization

Problem

PARALLEL SYSTEMS ARCHITECTURE LAB

- By default, NCCL builds the topology among GPUs using a ring
  - Impossible on GPU clusters with irregular topology
  - Single ring could cause link under-utilization
  - Ring's capacity is also restricted by the slowest link
- NCCL also constructs binary tree, but only for small dataset
- Otherwise, NCCL fallback to PCI-Express
  - Waste of NVLink!

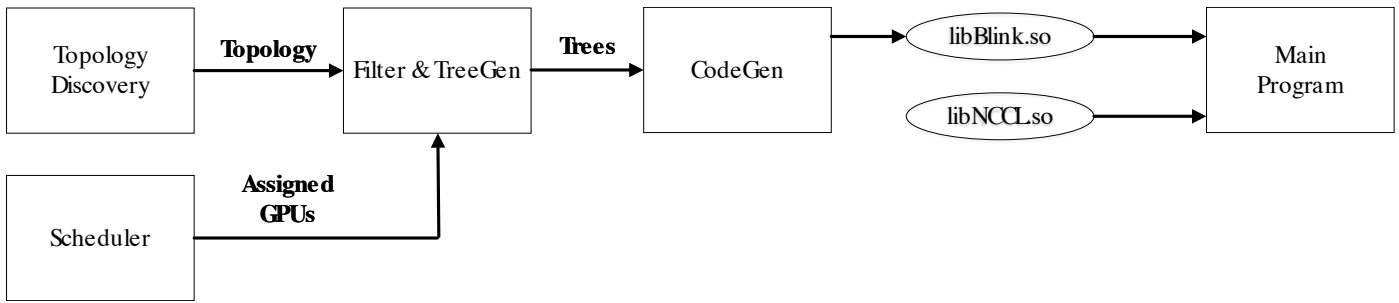
Insights

PARALLEL SYSTEMS ARCHITECTURE LAB

- Trees are better template than ring
  - Adaptability to irregular topology
  - Natural for broadcasting and gathering
- GPUs and NVLinks are not the bottleneck when using spanning trees
  - GPUs support data transferring while computation
  - NVLinks can be multiplexed without severely impacting peak throughput
- Topology can be probed during runtime
  - Collective can be implemented with trees accordingly

Solutions

Blink, a runtime to generate collective primitives



Solutions: Tree Generation

PARALLEL SYSTEMS ARCHITECTURE LAB

- Formalized as an optimization problem
- Approximating packing
- Minimizing tree count to increase the data transferred by single tree
  - Enabling large data chunks

Solutions: Code Generation

PARALLEL SYSTEMS ARCHITECTURE LAB

- Detect chunk size by profiling and feedback
  - Prefer large chunk to amortize the control cost
  - Too large chunk can cause long synchronization time
  - Once benefit decreases, reduce the chunk size
- Manual and fair scheduler for shared link based on CUDA Stream

Takeaway

PARALLEL SYSTEMS ARCHITECTURE LAB

- Communication optimizations can be delayed to runtime
- Spanning trees are better topology template than rings
  - Spanning irregular topology
  - Utilize all possible links

Test-of-time Award?

PARALLEL SYSTEMS ARCHITECTURE LAB

- Maybe no
- It may get integrated into NCCL
  - People then will use NCCL blindly and forget this paper :(
- The paper does not examine the effect when large amount of GPUs are involved
  - A future trend
  - Even finding suboptimal spanning trees can be hard

Reason for Rejection

PARALLEL SYSTEMS ARCHITECTURE LAB

- A clear and nice paper, thus it should be accepted.
- A flaw: Why NCCL only uses double binary trees when the transferred data is small?
  - A guess: double binary tree can cause congestion without carefully planned

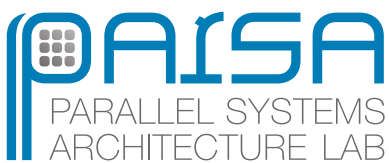
Thank You!

PARSA
PARALLEL SYSTEMS ARCHITECTURE LAB

For more information, please visit us at

parsa.epfl.ch

MSCCLang: Microsoft Collective Communication Language



Elevator Pitch

PARALLEL SYSTEMS ARCHITECTURE LAB

- Manual tuning of GPU collective algorithm is necessary
  - Squeezing performance using system and workload's properties
  - Research and system designer
- Writing new high-performance collective primitive is hard
- MSCCLang
  - Programmer just needs to specify the data dependency and schedule hints
  - Scheduling and code generation are done by program

Problem

PARALLEL SYSTEMS ARCHITECTURE LAB

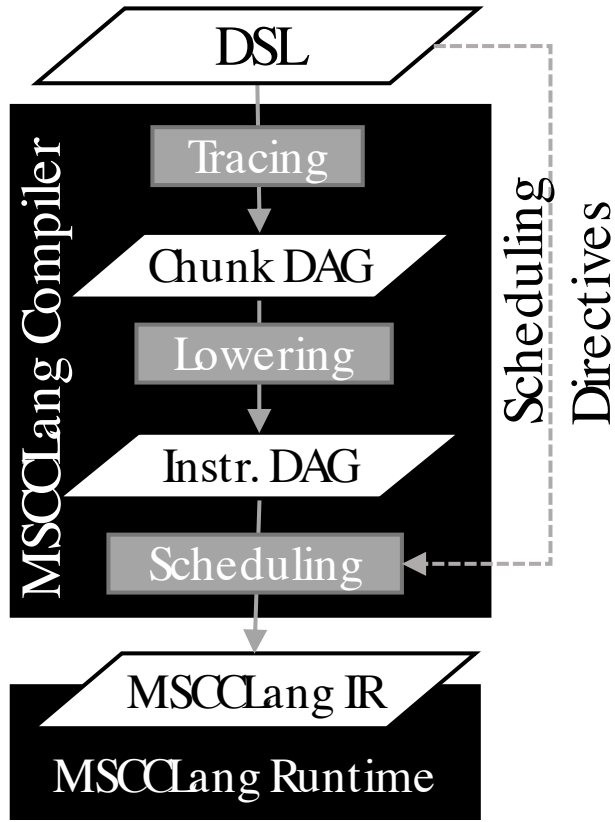
- Writing high-performance GPU collectives is challenging
  - Fine-grained parallelism extraction (e.g., link multiplexing)
  - Low-level implementation detail (e.g., primitive, deadlock)
  - Schedule tasks with dynamic information (e.g., pipelining)

Insights

PARALLEL SYSTEMS ARCHITECTURE LAB

- Programmers should focus on the collective development
  - Essentially, the data dependency among multiple GPUs
- Low-level optimizations and implementation details are well formalized
  - With data dependency the hint from the programmer, tools can do it

Solutions


DSL
MSCCLang Compiler
Tracing
Chunk DAG
Lowering
Instr. DAG
Scheduling
Scheduling Directives
MSCCLang IR
MSCCLang Runtime

Solutions: Domain-specific Language

PARALLEL SYSTEMS ARCHITECTURE LAB

- Each GPU exposes the shared buffer as the source operands
- Using copy and reduce to construct data dependency
- Applying schedule directives for tuning
  - Channel
  - Parallelize
  - Aggregation

Solutions: Compiler

PARALLEL SYSTEMS ARCHITECTURE LAB

- Trace input program to get data dependency
- Insert communication (send/receive) and synchronization primitive
- Fuse MSCCL primitives
  - Avoid resource allocation
  - Utilize complex primitive provided by NCCL
- Channel allocation
  - Multiplexing the same link to improve utilization
- Generate program in MSCCL-IR

Solution: Runtime

PARALLEL SYSTEMS ARCHITECTURE LAB

- Extension of NCCL and the interpreter of MSCCL-IR
  - Starts all threadblocks in parallel
  - Picking protocols based on the program's buffer size requirement
  - Overlapping the execution of different stages, i.e., pipelining

Takeaway

- Tools can actually do good work in terms of using low-level primitive
- Correct abstraction to the programmer is important

Test-of-time Award?

PARALLEL SYSTEMS ARCHITECTURE LAB

- Yes
- Infrastructure to accelerate more related researches
- Not only for ML but also for general HPC

Reason for Rejection

- The paper is written in low quality
  - Quickly fall into implementation details
  - Complex examples
  - Terminology inconsistency (e.g., buffer and chunk, channels)
- It is not very clear how practical the collective *AllToNext* is in ML
  - Paper: Pipeline processing with multiple GPUs?

Comparison between Blink and MSCCL

PARALLEL SYSTEMS ARCHITECTURE LAB

- Problem: the problem of NCCL
  - Blink: NCCL is unaware of topology
  - MSCCL: Hard to write high-performance collective with NCCL primitives
- Insights
  - Blink: Runtime topology probing + primitive generation with trees
  - MSCCL: Tools can do low-level optimization automatically
- Solutions:
  - Blink: Transparent runtime -> better adaptability, but low manual control
  - MSCCL: A compiler + runtime -> low development effort, but manual topology

Thank You!

PARSA
PARALLEL SYSTEMS ARCHITECTURE LAB

For more information, please visit us at

parsa.epfl.ch