TACCL: Guiding Collective Algorithm Synthesis using Communication Sketches

Aashaka Shah, Vijay Chidambaram, Meghan Cowan, Saeed Maleki, Madan Musuvathi, Todd Mytkowicz, Jacob Nelson, Olli Saarikivi, Rachee Singh
University of Texas at Austin, Vmware Research, Microsoft Research, Cornell University
NSDI 23

Synthesizing Optimal Collective Algorithms

Zixian Cai, Zhengyang Liu, Saeed Maleki, Madanlal Musuvathi, Todd Mytkowicz, Jacob Nelson, Olli Saarikivi
Australian National University, University of Utah, Microsoft Research
PPoPP 21

梁恒中
2024.6.19

Backgroud

- AI models are getting larger and larger
- Communication is becoming the bottleneck
- Novel hardware and topologies require novel communication kernels to maximize performance
- Currently communication kernels are hand-written and manually optimized

Need a tool to generate communication kernels for a given topology
- TACCL
- SCCL

(α,β) Cost Model

Sending a message of L bytes along a link costs $\alpha + L * \beta$ time, where

- α is the latency of communication
- β is the inverse of bandwidth

For a collective communication, the cost is $a * \alpha + b * L * \beta$, therefore

- a is the latency cost
- b is the bandwidth cost

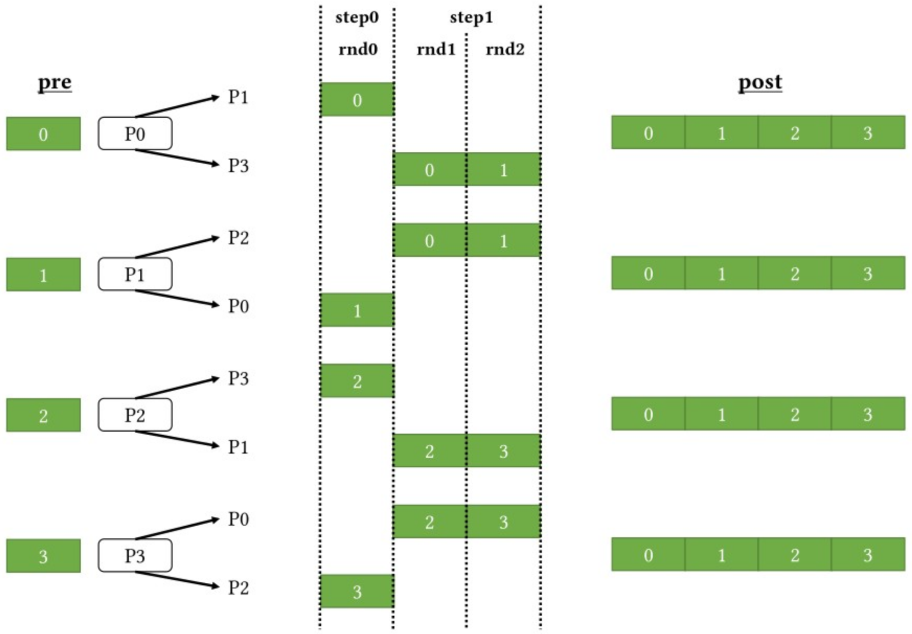
K-synchronous

Synchronous algorithm:

- proceeds in a sequence of synchronous communication steps, where nodes waits for other nodes finishing their rounds before starting next step
- To maximize bandwidth, input data is split into equal-sized chunks. Only one chunk could be sent per unit bandwidth in a round

K-synchronous algorithm

- with S steps and R rounds where R $\leq S + k$



Cost for L-byte input data divided into C chunks:

- $\alpha + r_s * (L/C) * \beta$ for step s
- $S * \alpha + R * (L/C) * \beta$ totally

Description of Collective Communication

An instance of SynColl is a tuple(G, S, R, P, B, pre, post) where:

- Parameters:
  - $G \in \mathbb{Z}_{\geq 0}$ is the global number of chunks
  - $S \in \mathbb{Z}_{\geq 0}$ is the total number of steps
  - $R \in \mathbb{Z}_{\geq 0}$ is the total number of rounds
- Topology:
  - $P \in \mathbb{Z}_{\geq 0}$ is the number of nodes
  - $B \subseteq \mathcal{P}([P] \times [P]) \times \mathbb{N}$ is the bandwidth relation
- Specification:
  - pre $\subseteq [G] \times [P]$ is the pre-condition, where chunks reside before the algorithm
  - post $\subseteq [G] \times [P]$ is the post-condition, where chunks reside after the algorithm

Solution of Collective Communication

A solution for SynColl(G, S, R, P, B, pre, post) is (Q, T) where:

- Q is a sequence $r_0, r_1, \ldots r_{S-1}$ such that $\Sigma r_i = R$
- T is sends in form of (c, n, n', s), which means chunk c is sent to node n' from node n at step s.

A run is defined as a sequence of $V_0, V_1, \ldots V_S$, where

- $V_0 = \text{pre}$
- $V_{s+1} = V_s \cup \{(c, n') \mid (c, n) \in V_s \wedge (c, n, n', s) \in T\}$

V_s describes which chunk is available on which node at step s

A solution is valid if

- $V_S \subseteq \text{post}$ (or $\text{post} \subseteq V_S$?)
- $\forall s \in [S]$, $(L, b) \in B$, $|\{(c, n, n', s) \in T \mid (n, n') \in L\}| \leq b \cdot r_s$

SMT Encoding

SMT: Satisfiability Modulo Theories

Given constraints, SMT solver Z3 will try to search for a candidate solution

Constraints:

- $\forall(c, n) \in$ pre $\text{time}_{c,n} = 0$
- $\forall(c, n) \in$ post $\text{time}_{c,n} \leq S$
- $\forall(c, n) \notin$ pre $\text{time}_{c,n} \leq S \Rightarrow \Sigma_{(n',n) \in E}\ \text{snd}_{n',c,n} = 1$
- $\forall(c, n) \in E\ \text{snd}_{n,c,n'} \Rightarrow \text{time}_{c,n} < \text{time}_{c,n'}$
- $\Sigma_{(c,(n,n')) \in [G] \times L} (\text{snd}_{n,c,n'} \wedge \text{time}_{c,n'} = s) \leq b \cdot r_{\text{s}}$
- $\Sigma_{1 \leq s \leq S} (r_{\text{s}}) = R$

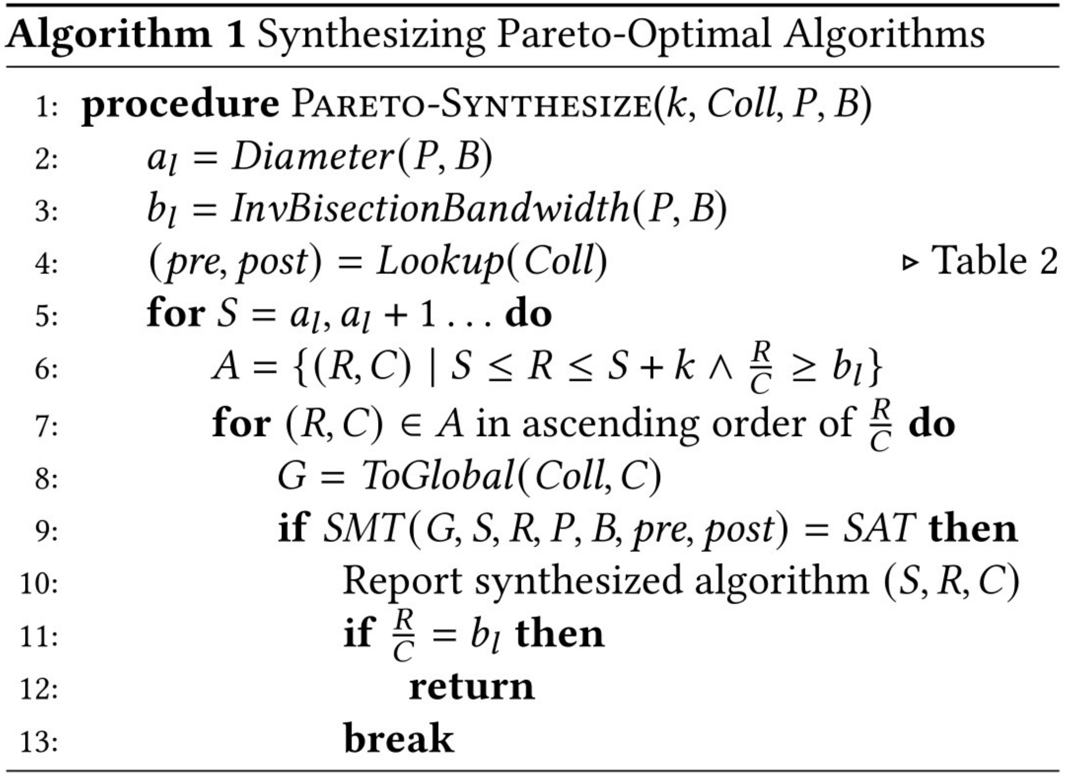
Synthesizing Algorithm

a_l: the lower bound of latency

b_l: the lower bound of bandwidth

The procedure checks if a (S, R, C) algorithm exists.

SCCL generates SPMD multi-process C++ code combined with CUDA kernels



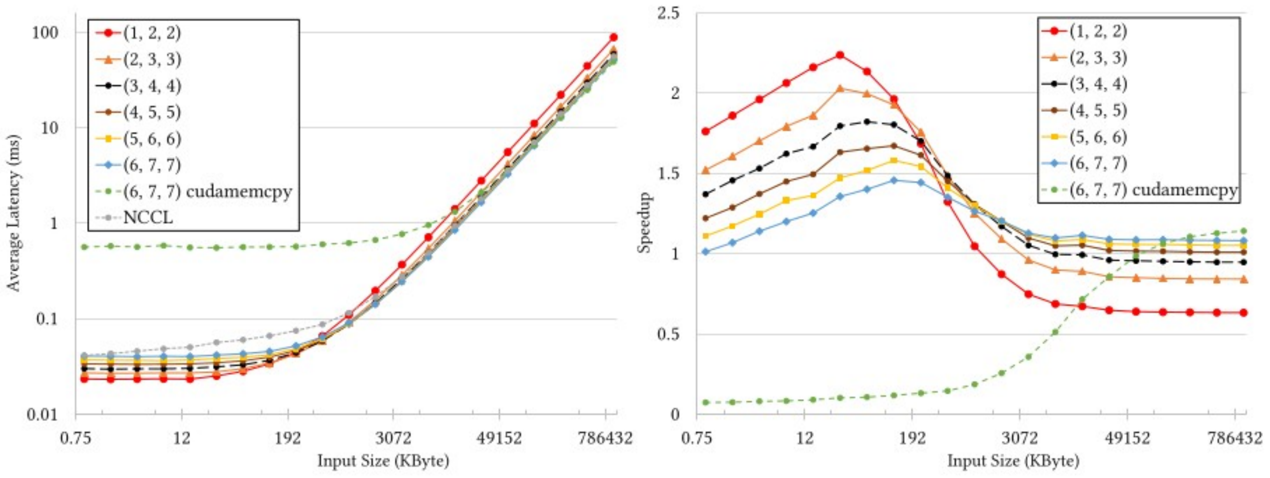
Algorithm 1 Synthesizing Pareto-Optimal Algorithms

```
procedure PARETO-SYNTHESIZE($k$, $Coll$, $P$, $B$)
    $a_l = Diameter(P, B)$
    $b_l = InvBisectionBandwidth(P, B)$
    $(pre, post) = Lookup(Coll)$                          ▷ Table 2
    for $S = a_l, a_l + 1 \ldots$ do
        $A = \{(R, C) \mid S \leq R \leq S + k \wedge \frac{R}{C} \geq b_l\}$
        for $(R, C) \in A$ in ascending order of $\frac{R}{C}$ do
            $G = ToGlobal(Coll, C)$
            if $SMT(G, S, R, P, B, pre, post) = SAT$ then
                Report synthesized algorithm $(S, R, C)$
                if $\frac{R}{C} = b_l$ then
                    return
                break
```

Evaluation

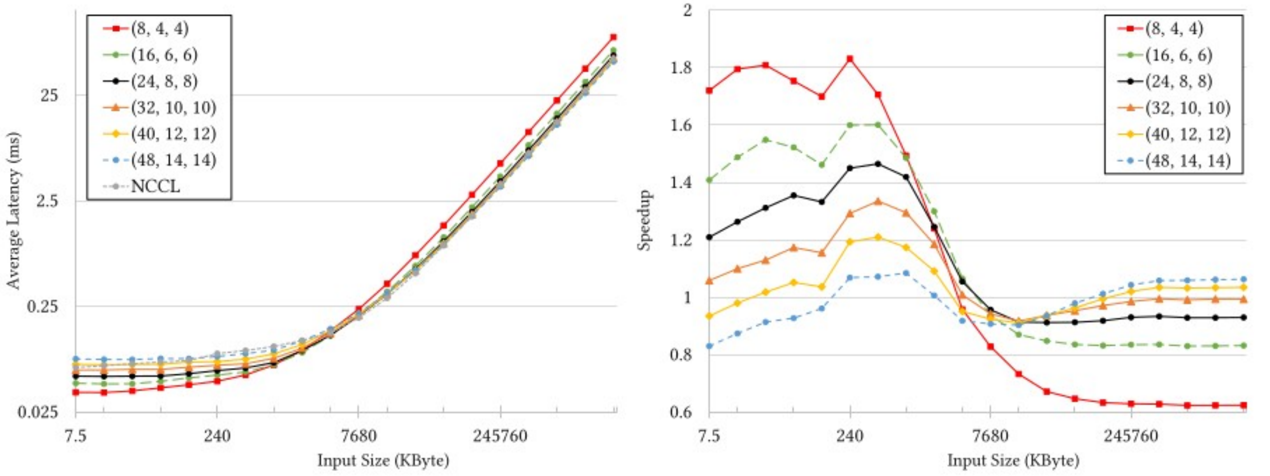
Performed on Nvidia DGX-1: 8 Tesla V100, each with 6 25GB/s NVLink ports

The longest synthesis time takes over 2 minutes, and the average time is about 10s.



(a) Average latency **(b)** Speedup

Figure 4. Allgather performance comparison with NCCL



(a) Average latency **(b)** Speedup

Figure 5. Allreduce performance comparison with NCCL

Motivation of TACCL

Drawbacks of SCCL

- Time consuming and poorly scalable
- Unware of heterogeneous connection

TACCL approach

- High-level inputs from algorithm designer: Communication sketch
- Seperating routing and scheduling

Communication Sketch

High-level input from algorithm designers to guide algorithm generation, mainly contains routing information

- The logical topology the algorithm is operating on
- Annotations on switches to abstract them away
- Algorithm symmetry on topology and collective
- Input size of data

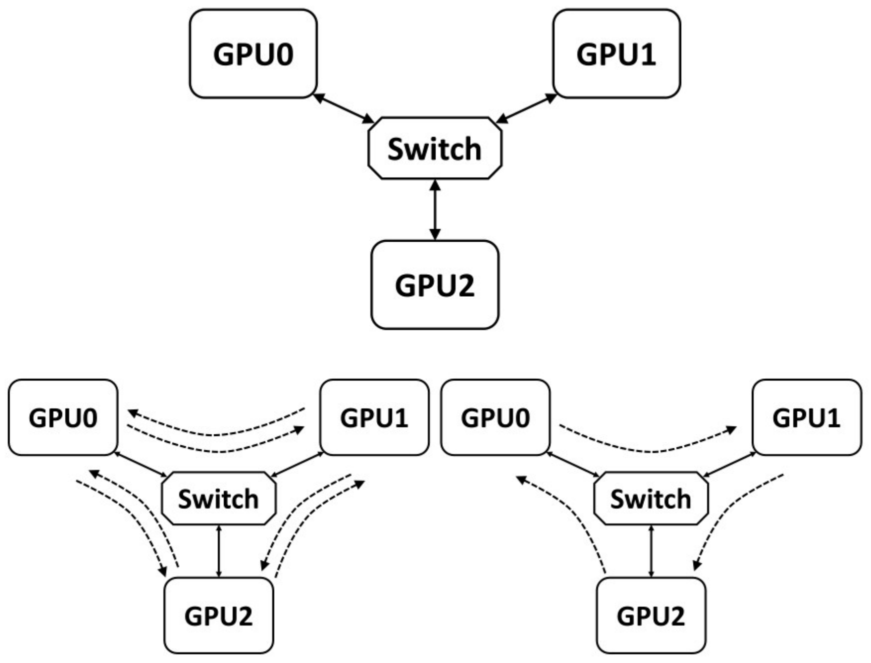
Logical Topology

Subset of physical topology by omiting NICs, switches and some links

Instead, switch-hyperedge is used to represent connection between GPUs connected by a switch

Different switch-hyperedge configurations lead to different performance: bandwidth drops as the number of connections increases.

(a) Physical topology with a switch.



(b) Max. connections strategy. (c) Min. connections strategy.

Figure 3: Effects of switch-hyperedge policies.

Physical Topology

Use (α,β) cost model

Use a profiler to empirically derive α and β for different link by p2p communication

The profiler is also used to infer physical topologies for GPU, CPU and NIC.

TACCL Synthesizer

Given a communication sketch and a collective, the synthesizer routes and schedules chunks.

TACCL encodes the problem as a mixed integer linear program(MILP) and tries to mininize the maximum time chunks need to reach their destinations.

- start_time: when a chunk is available on a GPU
- is_sent: whether a chunk is sent over a link
- send_time: when a chunk is sent over a link
- Bandwidth and correctness constraints

TACCL Synthesizer

1.Routing

Determine the path from source to destination for each chunk, allowing overlapping to gain a lower bound

Constraints:

- for each link, the number of chunks that traverse that link multiplied by the transfer time of a chunk over that link.
- for the path of each chunk, the summation of transfer times of the chunk along every link in the path.

TACCL Synthesizer

2.Heuristic Ordering

Determines the chunk ordering for each link, solved by a greedy algorithm

- chunks which need to traverse the longest path from src to their final GPU have higher priority.
- chunks which have traversed the shortest path from their initial GPU to src have higher priority.

Here src is the sending side of the link.

TACCL Synthesizer

3. Contiguity and Exact Scheduling

Determines which chunks to be sent contiguously and gives exact scheduling based on the (α,β) cost model

- By sending chunks contiguously, latency cost is reduces
- By not doing so, subsequent chunks can be scheduled earlier

Evaluation

Platform:

- 2 Nvidia DGX-2 nodes
- Up to 4 Azure NDv2 nodes with 100Gbps IB NIC

Baseline: NCCL

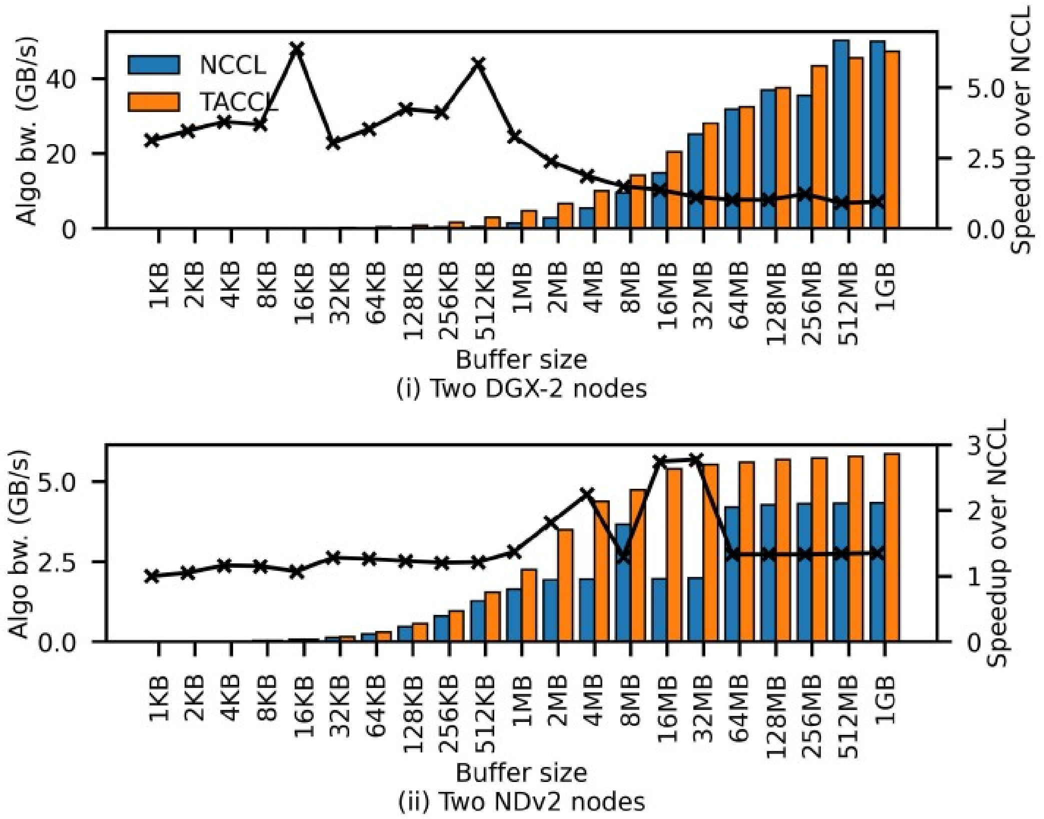


Figure 8: ALLREDUCE comparisons of NCCL to TACCL's best algorithm at each buffer size.

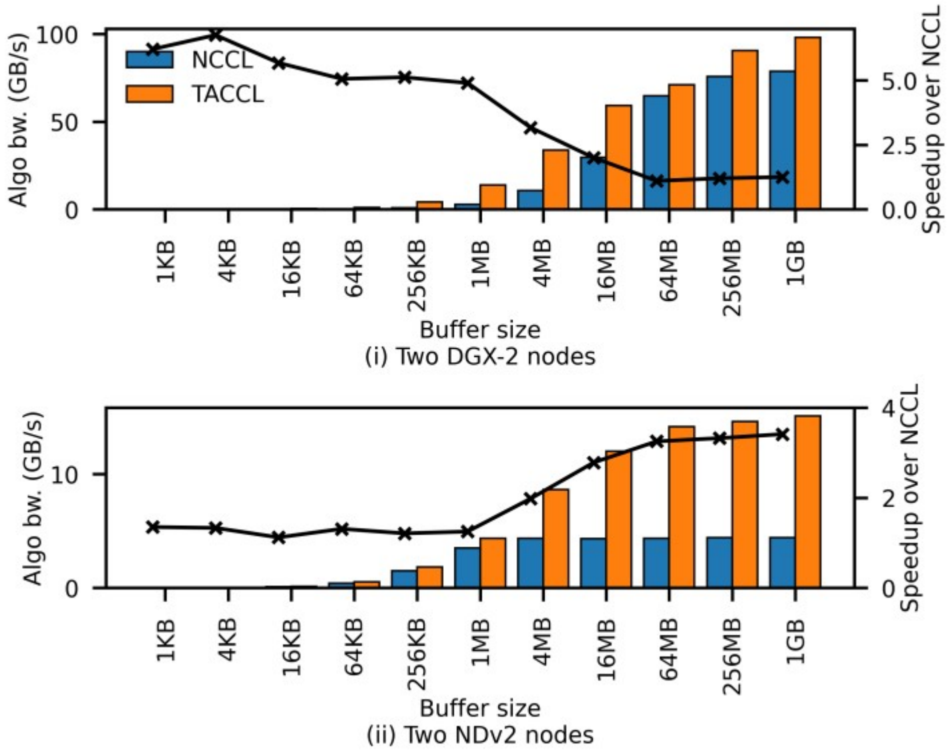


Figure 6: ALLGATHER comparisons of NCCL to TACCL's best algorithm at each buffer size.

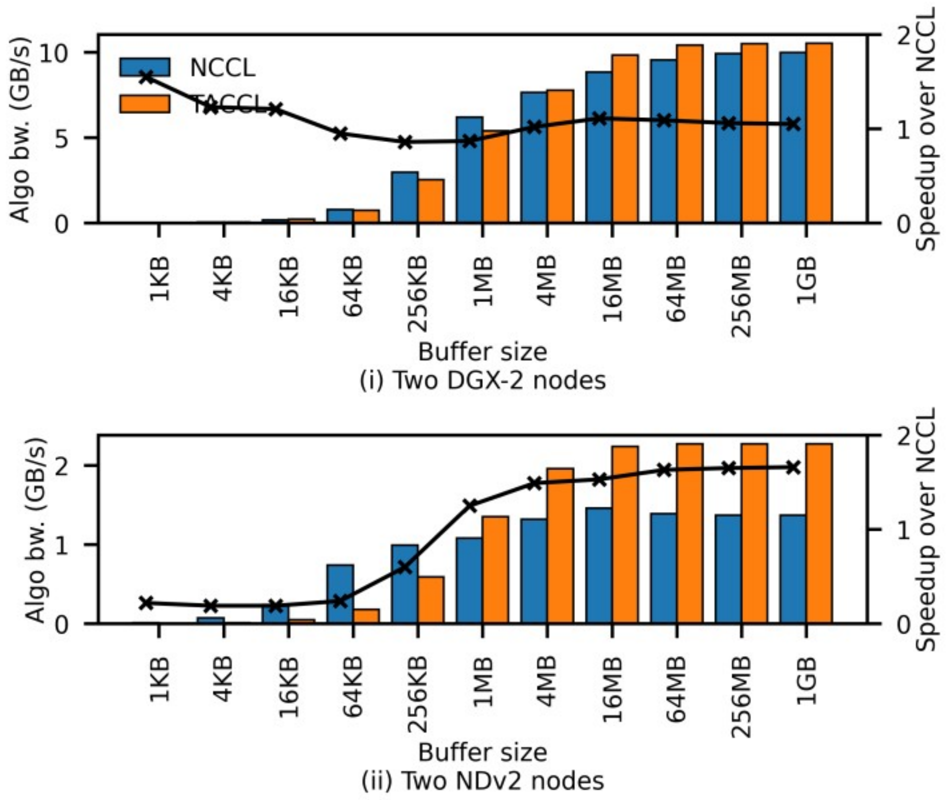
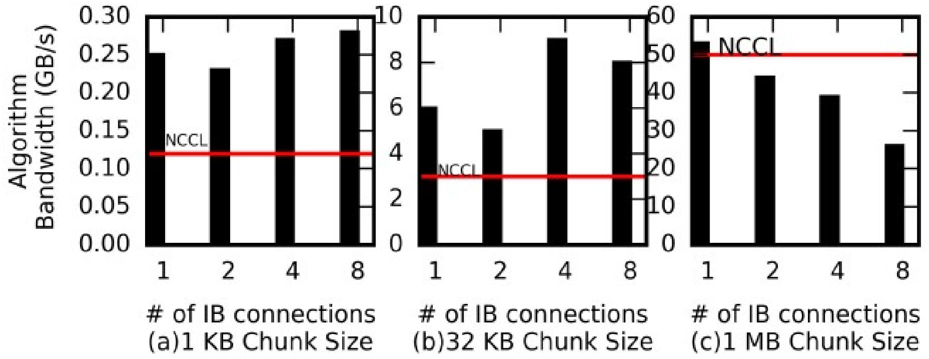
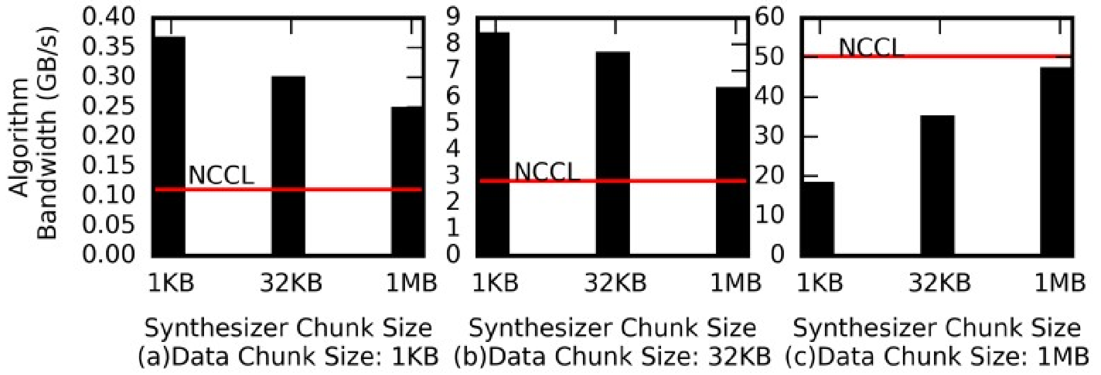


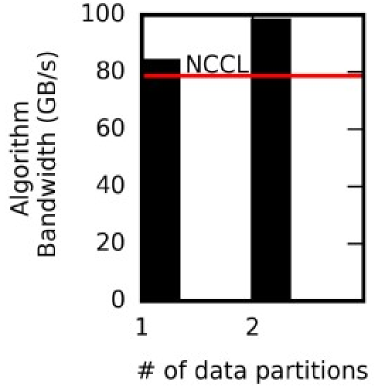
Figure 7: ALLTOALL comparisons of NCCL to TACCL's best algorithm at each buffer size.



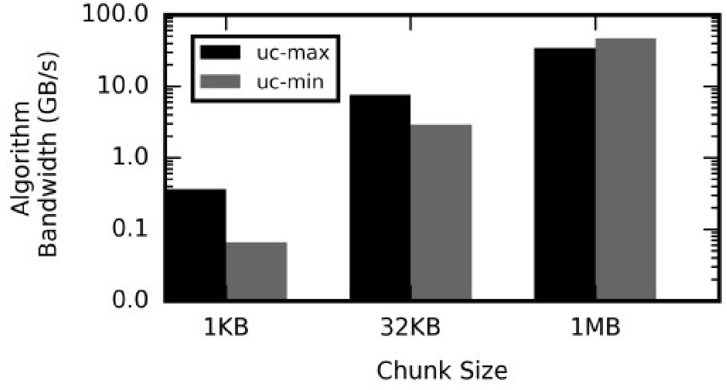
(a) Logical topology



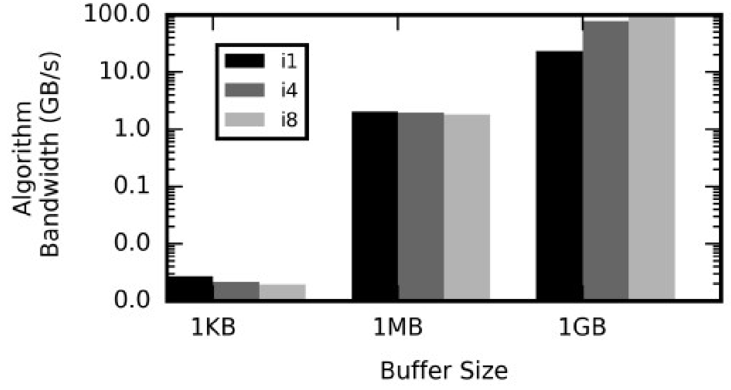
(b) Chunk size



(c) Data partition

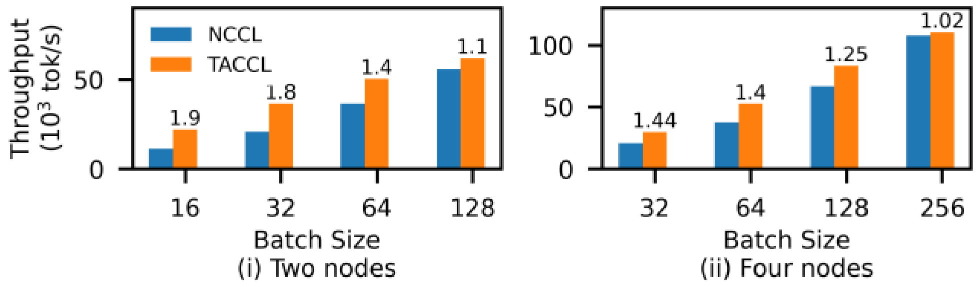


(d) Switch-hyperedge strategies

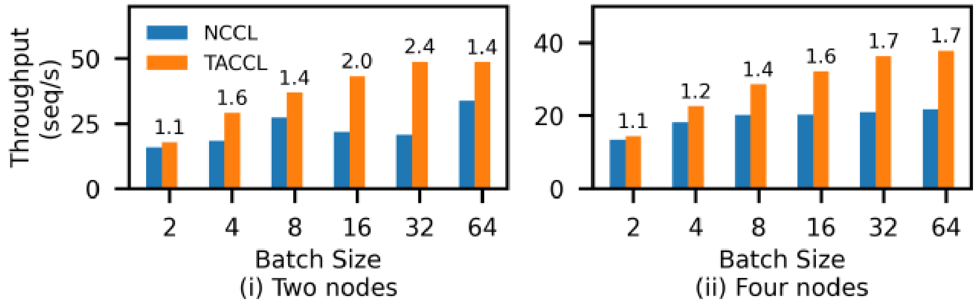


(e) Runtime instances

Figure 9: Algorithm bandwidth of ALLGATHER algorithms on DGX-2 by varying different inputs to TACCL



(a) Transformer-XL



(b) BERT

AllGather		AlltoAll		AllReduce	
Sketch	Time(s)	Sketch	Time(s)	Sketch	Time(s)
dgx2-sk-1	35.8	*dgx2-sk-2*	92.5	*dgx2-sk-1*	6.1
dgx2-sk-2	11.3	*ndv2-sk-1*	1809.8	*dgx2-sk-2*	127.8
ndv2-sk-1	2.6	*ndv2-sk-2*	8.4	*ndv2-sk-1*	0.3