

Collective Communication Synthesis: From SCCL to TACCL

CSE 561 Network Communication System, Autumn 26

Sheng Wei, Yusong Yan



Outlines

- SCCL Basics: Cost model and its limitations (Original recipe)
- TACCL: (The upgrade)
 - Overall Architecture
 - Some new seasoning of the original recipe
 - Communication Sketches
 - Profiled Topology
 - Target Collective
 - Synthesizer
 - Routing
 - Heuristic Ordering
 - Contiguity and Exact Scheduling
 - Backend
 - Eval
- Discussion

Two Papers and Code Repo

1. SCCL: Synthesizing optimal collective algorithms – PPOPP'21 (Best Paper Award)

<https://dl.acm.org/doi/10.1145/3437801.3441620>

2. TACCL: Guiding Collective Algorithm Synthesis using Communication Sketches -- NSDI'23

<https://arxiv.org/abs/2111.04867>

3. MSCCL github Repo

<https://github.com/microsoft/msccl>

Synthesizing Optimal Collective Algorithms Basics

Classic α - β Cost Model:

$$T(L) = a * \alpha + b * L * \beta$$

L: Message size

T(L): Estimated time to send message with size L between two nodes

a: Latency cost

b: Bandwidth cost

α : Fixed cost of startup latency

β : Transfer time per byte

Synthesizing Optimal Collective Algorithms Basics - cont'

Question:

Two chunks message, each of size L , need to travel from $A \rightarrow B \rightarrow C$;

Both links starts from t_0 ;

Each link has $\alpha=5s$, $L_{\square}=10s$;

The two links can operate simultaneously;

A node can only forward the message after it completely receive it.

- Option 1: Send them separately
- Option 2: Send them in batch



Synthesizing Optimal Collective Algorithms Basics - cont'

Option 1:

Links\Time(s)	0-15	15-30	30-45
A->B	C1	C2	
B->C		C1	C2

$$T_{\text{separated}} = 3 * (\alpha + L_{\square}) = 3 * 15 = 45s$$

Synthesizing Optimal Collective Algorithms Basics - cont'

Option 2:

Links\Time(s)	0-25	25-50
A->B	C1 + C2	
B->C		C1 + C2

$$T_{\text{batch}} = 2 * (\alpha + 2 * L_{\square}) = 2 * (5 + 20) = 50\text{s}$$

Synthesizing Optimal Collective Algorithms Basics - cont'

Winner: Option1

Saving startup latency by combining chunks, or forward smaller chunks earlier to overlap communication is a trade-off.

Trade-off !

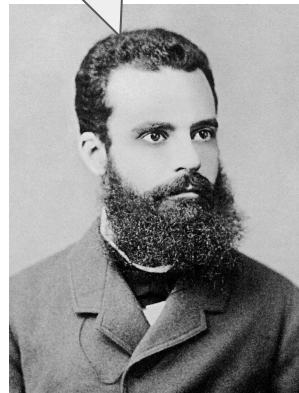
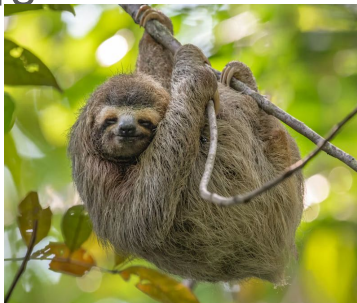


Image of Vilfredo Pareto from Wikipedia

SCCL Workflow

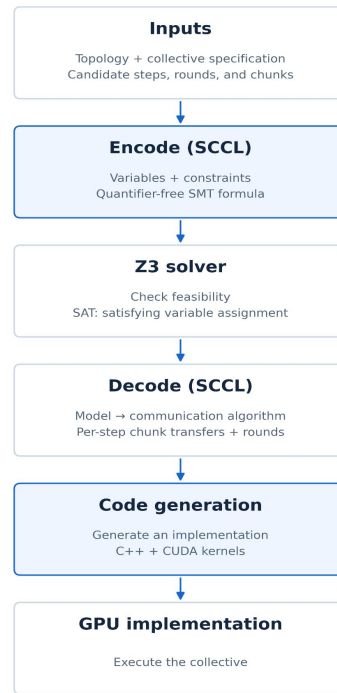
Slow!! Hard to scale, synthesis takes more than 24
hrs



from: Getty Images



from: cat meme



One feasible candidate shown.
Outer parameter search may continue after SAT or UNSAT.

GPT6.1 Gen

TACCL Architecture

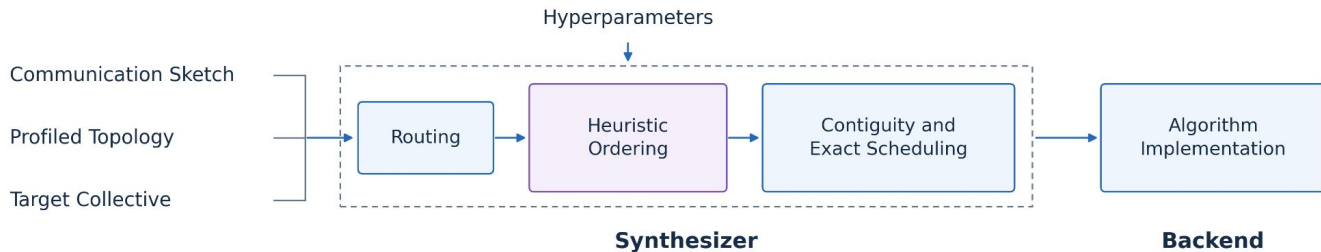
TACCL leverages communication sketches to reduce the search space for synthesizer;

Relax the synthesis problem, just solve routing of all data chunks, heuristic order chunks send

Over the same links as per BW constraints.



From: Know Your Meme



GPT6.1 Gen, replicate of TACCL Figure 1.

TACCL Inputs

- Communication Sketches

- Logical topology
- Switch-hyperedge policies
- Algorithm symmetric
- Input size

- Physical Topology Profiler

- To disambiguate the topology of multi-GPU systems
- alpha-beta cost model: TP measures and infers the alpha and beta costs of different type of links in a GPU systems

Link	Azure NDv2		Nvidia DGX-2	
	α (us)	β (us/MB)	α (us)	β (us/MB)
NVLink	0.7	46	0.7	8
InfiniBand	1.7	106	1.7	106

Table 1: Experimentally obtained α and β costs for Azure NDv2 and Nvidia DGX-2 nodes.

TACCL Synthesizer

Problem Formulation:

MILP problems: NP-hard, still need to reduce the search space.

Three Steps:

Routing (MILP): Select chunk routes using relaxed bandwidth constraints

Ordering Heuristic (Greedy): Fix transfer order on links and shared switch resources

Contiguity and Exact Scheduling(MILP): Optimize transfer timing and batch under the fixed routes and orders

Hyperparameters:

Buffer Size: Sets the balance between startup latency and bw costs

Chunk Partitioning: Enables finer routing and pipelining at greater synthesis cost

Switch-Hyperedge Policy: Favors fewer or more logical connection

TACCL Backend

1. **TACCL runtime:** Supports all NCCL comm backends.
2. **Execution format:** Converts the synthesized algorithm into TACCL-EF, an XML representation of chunk operations and dependencies.
3. **GPU mapping:** Maps communication instructions to GPU threadblocks and channels.
4. **Parallel instances:** Replicates the algorithm over data subdivisions to improve bandwidth utilization.

More details please check the MSCCL Repo

TACCL Eval

- **Collective performance:** Up to $6.7\times$ faster Allgather and 66% faster Alltoall than the evaluated NCCL baseline.
- **Application benefit:** $1.94\times$ speedup for Transformer-XL and $2.36\times$ for BERT.
- **Synthesis efficiency:** Most evaluated configurations take seconds to minutes; an 80-GPU Allgather takes under 8 minutes while SCCL takes longer than one day.



from: cat meme

Conclusion

SCCL: Uses SMT solving to synthesize Pareto-optimal collective algorithms within its k-synchronous model.

TACCL: Uses communication sketches, heuristic ordering, and staged MILP optimization to make synthesis practical for larger, heterogeneous GPU clusters.

Key tradeoff: Restricting and decomposing the search improves scalability, but does not establish global optimality.

Shared lesson: Topology and message size matter—specialized communication algorithms can outperform generic choices.

Open Questions and Discussion

AI & Agents: Could an LLM agent propose better MILP encodings or decompositions that reduce synthesis time? How would we verify that its changes preserve correctness and do not exclude better algorithms?

Sketches Extensions: What additional hints would help TACCL—shared PCIe/NIC bottlenecks, competing traffic, or application communication patterns? Would these improve performance enough to justify more complex sketches?

Benchmarks: Would TACCL's benefits hold across irregular topologies, mixed GPU generations, larger clusters, and different message sizes? Which experiments would best test its generality?

Robustness: How sensitive are synthesized algorithms to inaccurate α - β estimates or changing network contention? When is re-profiling and re-synthesis worth the cost?