Towards smart(er) High-Performance Networking Driving Future Simulations

with contributions by Microsoft, the whole SPCL deep learning team, and collaborators

MODSIM’23, August 2023, Seattle, WA, USA
The future of simulation and modeling hardware and software technologies?

Programming and Frameworks

Accelerators and Compute

Data Center Networking
Cloud and HPC Networks Converge

Cloud AI as a gravity well – HPC will follow
The Convergence of Hyperscale Data Center and High-Performance Computing Networks

Design and Deployment
- One-off vs. incremental
  - Proprietary networks vs. Ethernet
  - AI supercomputers in the cloud

Operations philosophy
- Run-to-completion jobs vs. high-reliability services
  - Checkpoint/restart vs. replicated instances
  - Large-scale training in the cloud

Service diversity
- Parallel jobs vs. opaque VM servers + microservices
  - Single context vs. QoS
  - Most will be AI-driven – serve LLMs

Protocol stacks and layers
- Proprietary vs. task-adapted flow control
  - Simple protocols vs. multi-traffic protocols
  - Lossless vs. lossy

Utilization and applications
- High peak low noise vs. low peak high noise
  - High bandwidth low latency vs. normal bandwidth high latency
  - AI demands highest bandwidths and reasonable latency

We discuss the differences and commonalities between network technologies used in supercomputers and data centers and outline a path to convergence at multiple layers. We predict that emerging smart networking solutions will accelerate that convergence.
Some Cloud-HPC networks are well on their way to convergence

Noise in the Clouds: Influence of Network Performance Variability on Application Scalability

Daniele De Sensi
daniele.desensi@inf.ethz.ch
ETH Zürich
Switzerland

Tiziano De Matteis
tiziano.dematteis@inf.ethz.ch
ETH Zürich
Switzerland

Konstantin Taranov
konstantin.taranov@inf.ethz.ch
ETH Zürich
Switzerland

Salvatore Di Girolamo
salvatore.digirolamo@inf.ethz.ch
ETH Zürich
Switzerland

Tobias Rahn
tobias.rahn@inf.ethz.ch
ETH Zürich
Switzerland

Torsten Hoefler
torsten.hoefler@inf.ethz.ch
ETH Zürich
Switzerland

ABSTRACT

Cloud computing represents an appealing opportunity for cost-effective deployment of HPC workloads on the best-fitting hardware. However, although cloud and on-premise HPC systems offer similar computational resources, their network architecture and performance may differ significantly. For example, these systems use fundamentally different network transport and routing protocols, which may introduce network noise that can eventually limit the application scaling. This work analyzes network performance, scalability, and cost of running HPC workloads on cloud systems. First, we consider latency, bandwidth, and collective communication patterns in detailed small-scale measurements, and then we simulate network performance at a larger scale. We validate our approach on four popular cloud providers and three on-premise HPC systems, showing that network (and also OS) noise can significantly impact performance and cost both at small and large scale. The full paper of this abstract can be found at [https://doi.org/10.1145/3570609](https://doi.org/10.1145/3570609).

ACM Reference Format:

1 INTRODUCTION

Network factors can contribute to increase network latency, decrease network bandwidth, and increase network noise [1] (i.e., performance variability induced by the use of the network). This limits the scalability and tampers cost-effectiveness. Although HPC applications can scale up to 42 million cores [4] on on-premise HPC systems, it is still unclear how far HPC applications could scale on the cloud.

In this work, we focused on network performance and noise, assessing the impact on performance, scalability, and cost of tightly-coupled HPC communication patterns at scale. In this extended abstract we only summarize the main findings. Interested readers can find the full paper at [https://doi.org/10.1145/3570609](https://doi.org/10.1145/3570609).

2 NETWORK PERFORMANCE

We measured network latency and bandwidth by running a 1-byte and a 16MB ping-pong respectively. We performed our analysis on the four main cloud providers (AWS, Azure, GCP, and Oracle), and three on-premise HPC systems (Alps, Daint, DEEP-EST).

Observation 1: On AWS and GCP, the peak bandwidth on a single connection is 50Gb/s and 30Gb/s respectively. A bandwidth of 80Gbps can only be reached by forking messages to be concurrently sent/received by/from multiple processes on different connections.

Observation 2: Azure and Oracle achieve network latency and bandwidth comparable to that of on-premise HPC systems. On the other hand, GCP and AWS achieve 25% lower bandwidth.

What about Cloud-AI networks? The 101 of AI communication patterns ...

Communication is (largely) a logical 3D Torus

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
(Network and memory) bandwidth is the new oil in AI supercomputing

- Memory bandwidth can be satisfied using HBM3 and friends
  - Technologies are quickly becoming available

- Network bandwidth is more complex and requires full-system and packaging tricks

- HPC:
  - Slingshot ('21): 200G per GPU
  - InfiniBand CX-7 ('22): 400G per NIC

- AI:
  - Google TPUv2 ('21): 1T
  - AWS Trainium ('21): 1.6T
  - DGX-2 (A100, ‘21): 4.8T (islands of NVLink)
  - Tesla Dojo ('22): 128T → Broadcom TH5 / NVIDIA Spectrum 4: 51.2T

- Performance models indicate even higher demands
  - Massive transformer EDAGs have really bad cuts

Conventional HPC topologies are unaffordable for AI bandwidths!

- A fat tree with 16k accelerators and 1.6T would cost $680M!

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
Co-designing an AI supercomputer with unprecedented and cheap bandwidth

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
A bandwidth-cost-flexibility tradeoffs

Global Topology (e.g., Fat Tree)

- (large) reduce bandwidth
- global bandwidth
- placement flexibility
- injection bandwidth

HammingMesh (many configurations)

- $ - $ - $

Local Topology (e.g., 2D Torus)

- $ - $

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
HammingMesh cost vs. bandwidth – simulated using SST (0.6M core hours)

<table>
<thead>
<tr>
<th>Topology</th>
<th>Large Cluster (≈16,000 accelerators)</th>
</tr>
</thead>
<tbody>
<tr>
<td>nonbl. FT</td>
<td></td>
</tr>
<tr>
<td>50% tap. FT</td>
<td></td>
</tr>
<tr>
<td>75% tap. FT</td>
<td></td>
</tr>
<tr>
<td>Dragonfly</td>
<td></td>
</tr>
<tr>
<td>2D HyperX(^2)</td>
<td></td>
</tr>
<tr>
<td>Hx2Mesh</td>
<td></td>
</tr>
<tr>
<td>Hx4Mesh</td>
<td></td>
</tr>
<tr>
<td>2D torus</td>
<td></td>
</tr>
</tbody>
</table>

Single switch per row/column
Practical usage – topology mapping and fault tolerance

- Mapping logical job topologies
  - 1D, 2D - obvious
  - 3rd dimension map onto switches

- Fault-tolerance
  - Nodes may fail
  - We fail the whole board
    *Remaining nodes run single-node jobs*
  - High flexibility!

- Simple greedy allocation scheme
  - Some added tricks (details in paper)

```plaintext
1-3:3x3; 4-5:2x3, 6-7:1x3, 8-9:1x2, 10-19:1x1

[ 1  1  1  2  4  2  2  4 ]
[ 1  1  2  4  2  2  4  ]
[ 1  1  2  -  2  2  - ]
[ 3  3  3  5  4  5  6  4 ]
[ 3  3  3  5  7  5  6  8 ]
[ 3  3  3  5  7  5  6  - ]
[ -  9 10 11  7 12 13  8 ]
[ -  9 14 15 16 17 18 19 ]
```

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
Experimental workloads

- Efficiency of the greedy allocation scheme
  - And all tricks

Alibaba’s ML-as-a-service (MLaaS) cluster with 6,742 GPUs workload trace
Experimental workloads

- Efficiency of the greedy allocation scheme
  - Now with random failures!

Alibaba’s ML-as-a-service (MLaaS) cluster with 6,742 GPUs workload trace

256 total
64 total

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
Alltoall results

AllToAll - Small Topologies (~ 1,000 nodes)

Throughput (Gb/s)

Message Size

nonblocking fat tree

2D HyperX

Dragonfly (1004 Gb/s at 4MiB)

fat tree 50% tapered

fat tree 75% tapered

Hx4Mesh

Hx2Mesh

2D Torus

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
Allreduce results

- Allreduce algorithms: (1) ring – optimal bandwidth, high latency, (2) torus – half bandwidth, lower latency

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
Full deep neural network communication

- First large-scale mini-app suite for communication in Deep Learning jobs
  - Many relevant and scalable networks: ResNets, BERT, CosmoFlow, DLRM, GPT-2, GPT-3, MoE, ...
  - Portable MPI C code – easy to adapt
  - Reproducible (also for other works)

- Full network simulations (using SST with MPI driver)

Relative Cost Savings (Communication Overhead of DNN Workloads)

TH et. al.: HammingMesh: A Network Topology for Large-Scale Deep Learning, SC22 and arXiv (2209.01346)
Remote direct memory access (RDMA) over converged Ethernet (RoCE) was an attempt to adopt modern RDMA features into existing Ethernet installations. We revisit RoCE’s design points and conclude that several of its shortcomings must be addressed to fulfill the demands of hyperscale data centers.
Getting there – Some RDMA Issues at Hyperscale

1) PFC requires excessive buffering for lossless transport – requires full BDP = BW * RTT + MTU buffer!
   - Assuming 600ns traversal latency (FEC, arbitration, forwarding, wire delay), 9 kiB packets, 8 priorities

Getting there – Some RDMA Issues at Hyperscale

- 1) PFC requires excessive buffering for lossless transport – requires full BW*RTT+MTU buffer!
  - Per 800G port for longer distance links, BDP grows

Getting there – Some RDMA Issues at Hyperscale

- **2) Victim flows, congestion trees, PFC storms, and deadlocks**

- **3) Go-back-N retransmission**
  - Simple recovery of lost packets (seq. number missing)
  - Yet, no real support for multi-pathing
  - Also retransmits full BDP on single loss (not a significant bandwidth loss though, <0.001% in practice)

- **4) Congestion control and collocated traffic**
  - Interference with other traffic types, simple CC is not necessarily compatible!
  - Led to invention of DCQCN, TIMELY, HPCC, and likely many more – somewhat hacky?

5) Header sizes
- RoCEv2 is basically an InfiniBand BTH strapped onto a UDP/IP packet
- Overhead: 22B L2, 20B IP, 8B UDP, 12B BTH, 4B ICRC → min packet size 66B
- Limits message rate and processing efficiency

6) No smart stacks
- Should have support for Smart NICs, e.g., sPIN NICs
- INC and INT are somewhat tagged on

7) Security issues
- ReDMARk issues – whole different talk on RDMA security
  https://www.youtube.com/watch?v=VGQe-OpICq8
- Even NVMe-of is broken (see NeVerMore paper at CCS’22)
- Fixes available with sRDMA ideas (Usenix Security’21)

---

Getting there – Some RDMA Issues at Hyperscale

- 8) Link Level Reliability
  - FEC is becoming an issue – new concatenated, segmented, and direct FEC increase latency!
  - RS272 (LL-FEC) can help but only to a limited degree

Looking forward: CC/LB is becoming harder!
- Larger messages will be sent within a single BDP! → higher fraction of traffic
- CC/LB management will not get a good signal 😞
Conclusions

- Design and Deployment
  - Cheapest, flexible
  - Proprietary networks vs. Ethernet
  - As a superset of the cloud
  - Operations philosophy
  - Run everything on the cloud, high-reliability services
  - Cheaper protocols vs. replicated resources
  - Large-scale training in the cloud
  - Service diversity
  - Parallel jobs vs. compute VM servers vs. microservices
  - Single vs. Multi-tenant
  - Ensuring SLAs in the cloud
- Protocol stacks and layers
  - Properties vs. state-based flow control
  - Simple protocols vs. stateful service protocols
  - Load vs. latency
- Utilization and applications
  - High peak low variance vs. low peak high noise
  - High bandwidth low latency vs. varied bandwidth high latency
  - AI demands highest bands, costs and measurable latency

More of SPCL’s research:
- youtube.com/@spcl
  - 175+ Talks
- twitter.com/spcl_eth
  - 1.2K+ Followers
- github.com/spcl
  - 2K+ Stars

... or spcl.ethz.ch
Three systems dimensions in large-scale super-learning …

High-Performance I/O
• Quickly growing data volumes
  • Scientific computing!
  • Use the specifics of machine learning workloads
  • E.g., intelligent prefetching

High-Performance Compute
• Deep learning is HPC
  • Data movement!
  • Quantization, Sparsification
  • Drives modern accelerators!

High-Performance Communication
• Use larger clusters (10k+ GPUs)
• Model parallelism
  • Complex pipeline schemes
• Optimized networks

Data Movement Is All You Need: A Case Study on Distribution and Parallelism

More details and similar content: youtube.com/@spcl