OPENFABRICS ALLIANCE

2020 OFA Virtual Workshop

ONEAPI, ONECCL AND OFI: PATH TO HETEROGENEOUS ARCHITECTURE PROGRAMMING WITH SCALABLE COLLECTIVE COMMUNICATIONS

Sayantan Sur, Principal Engineer

Intel® Corp.

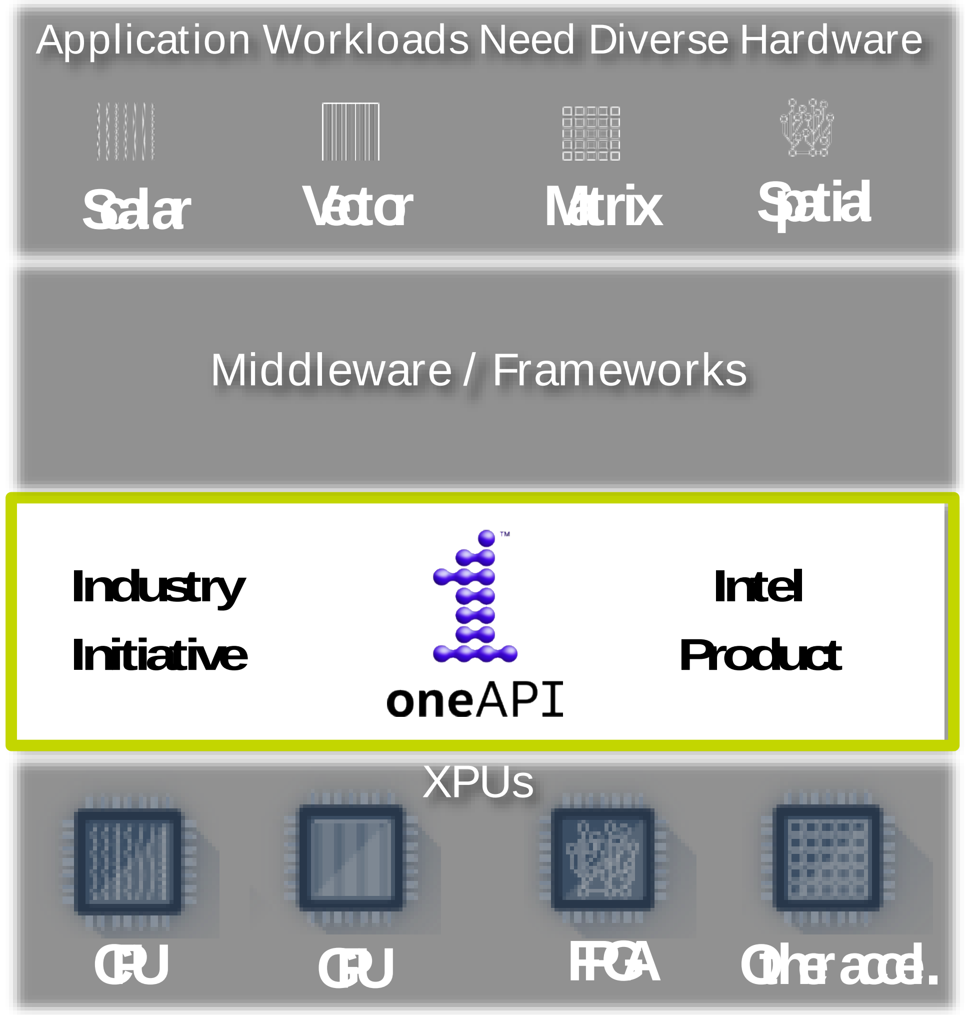
PROGRAMMING CHALLENGES FOR MULTIPLE ARCHITECTURES

Challenges:

- **Growth in specialized workloads**
- **No common language or APIs**
- **Inconsistent tool support across platforms**

Introducing oneAPI:

- **Unified and simplified language and libraries for expressing parallelism**
- **Based on industry standards and open specifications**
- **Interoperable with existing HPC programming models**



DEEP LEARNING WITH COLLECTIVE COMMUNICATIONS

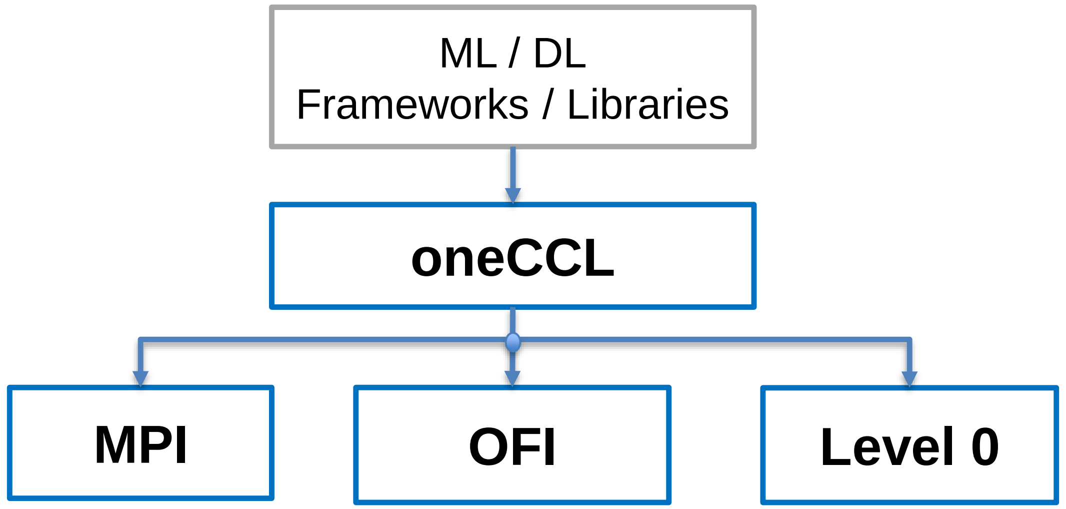
- **Deep learning is a branch of AI where a neural network (model) is trained using either labeled on unlabeled data**
- **Training a model is very compute intensive**
- **Distributing the computation either using model replication or distribution is required to train a model in reasonable amount of time**
  - Introduces communication in the training process
  - Communication is generally collective (several processors participate simultaneously)
  - Typical communication routines used are: Allreduce, Reduce-Scatter, Allgather, Reduce, Broadcast, Alltoall
- **oneAPI Collective Communications Library (oneCCL)**
  - Optimized implementations of Collective Communications
  - Exposes APIs that are friendly for Deep Learning Frameworks
- **Open specification:** https://spec.oneapi.com/versions/latest/elements/oneCCL/source/index.html
- **Open source implementation:** https://github.com/oneapi-src/oneCCL

ONECCL ARCHITECTURE AND FEATURES

ONEAPI COLLECTIVE COMMUNICATIONS LIBRARY (ONECCL)

- Built on top of lower-level communication middleware. MPI and libfabrics transparently support many interconnects, such as Intel® Omni-Path Architecture, InfiniBand*, and Ethernet.
- Enables efficient implementations of collectives that are heavily used for neural network training, including all-gather, all-reduce, and reduce-scatter.

MPI and OpenFabrics Interface (OFI) provide portable abstractions across wide variety of network fabrics



Level Zero provides low-level direct-to-metal interfaces that are tailored to the devices in a oneAPI platform

ONECCL PROGRAMMING MODEL

Key abstractions

- **Stream**
  - Encapsulates execution context for communication operation
- **Communicator**
  - Defines participants of communication operation
  - Rank = device (CPU or GPU)
  - Creation can be controlled with attributes
- **Collective**
  - Communication operation between communicator’s participant
  - Behavior can be controlled with attributes

SPARSE TENSOR ALLREDUCE IN ONECCL

- **Language models typically feature huge embedding tables within their topology**
- **Simple gradient computation followed by Allreduce not performant**
- **Gradients for such layers are computed for a smaller sub-tensor**
- **A sparse allreduce enables computation on sub-tensors**
  - "Language Modeling at Scale", M. Patwary, et. al. *Silicon Valley AI Lab, Baidu Research*

```
ccl_status_t CCL_API ccl_sparse_allreduce(
    const void* send_ind_buf, size_t send_ind_count,
    const void* send_val_buf, size_t send_val_count,
    void** recv_ind_buf, size_t* recv_ind_count,
    void** recv_val_buf, size_t* recv_val_count,
    ccl_datatype_t index_dtype,
    ccl_datatype_t dtype,
    ccl_reduction_t reduction,
    const ccl_coll_attr_t* attr,
    ccl_comm_t comm,
    ccl_stream_t stream,
    ccl_request_t* req);
```

Define Indices that are Valid

Define Send Buffer

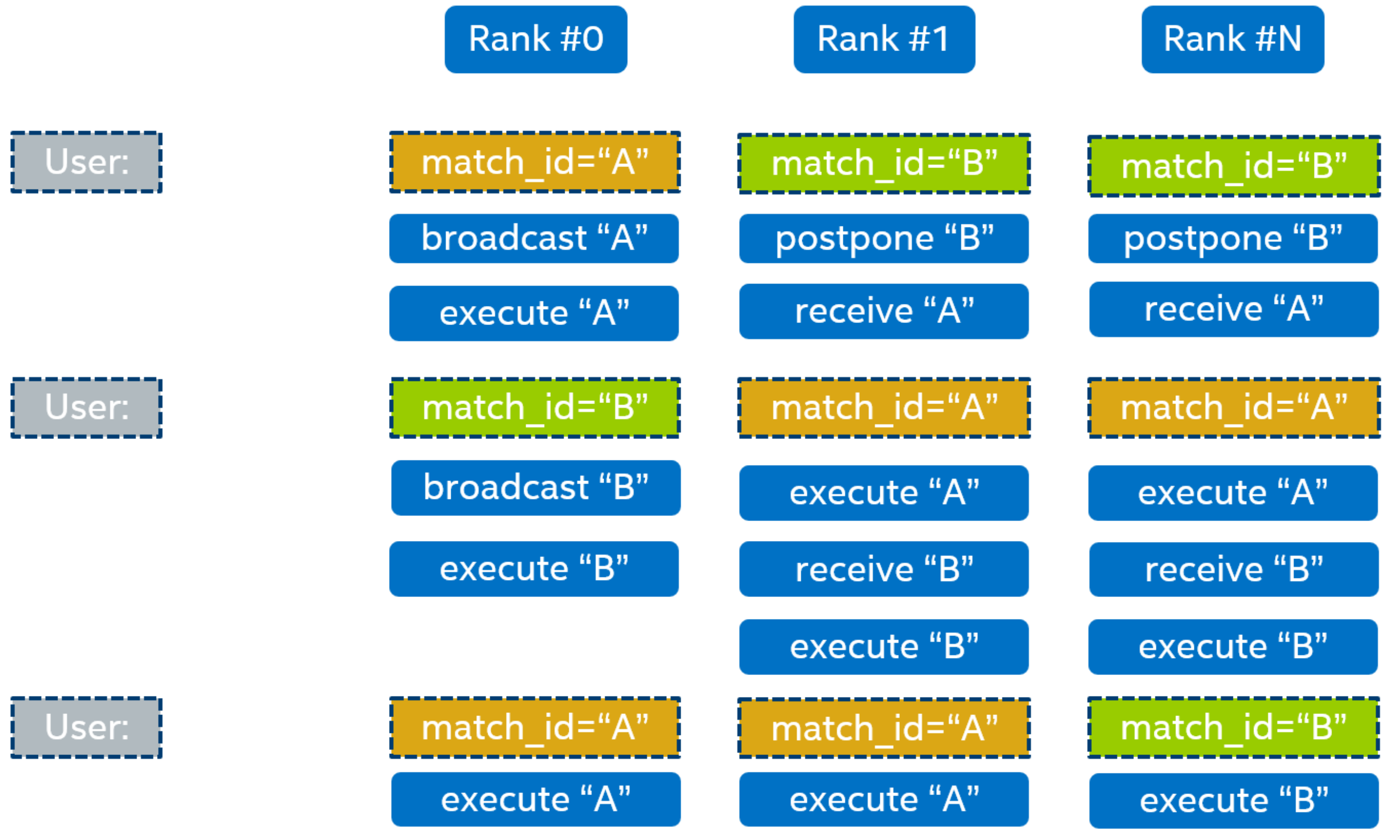
Output receive indices

Output receive buffer

Example of Application specific Collective APIs

UNORDERED COLLECTIVE SUPPORT

- Some frameworks deploy local scheduling approach for the graph of operations, which may result in different ordering of collective operations across different processes.
- In contrast, oneCCL provides a mechanism to arrange execution of collective operations in accordance with the user-defined identifier
- Increase productivity by directly mapping framework requirements to collective library



PRIORITIZATION OF COLLECTIVES

Individual collective operations can set the priority with which they are executed

- **This allows to postpone execution of non-urgent operations to complete urgent operations earlier**
- **Optimizes use cases such as overlapping, mixed model/data parallelism etc.**
- **The priority is a non-negative number; priority increases with value**
- **coll_attr.priority lets the caller set priority, or environment variable CCL_PRIORITY**

Values

- None - default mode when all collective operations have the same priority.
- Direct - you explicitly specify priority using coll_attr.priority.
- LIFO (Last In, First Out) - priority is implicitly increased on each collective call. In this case, you do not have to specify priority.

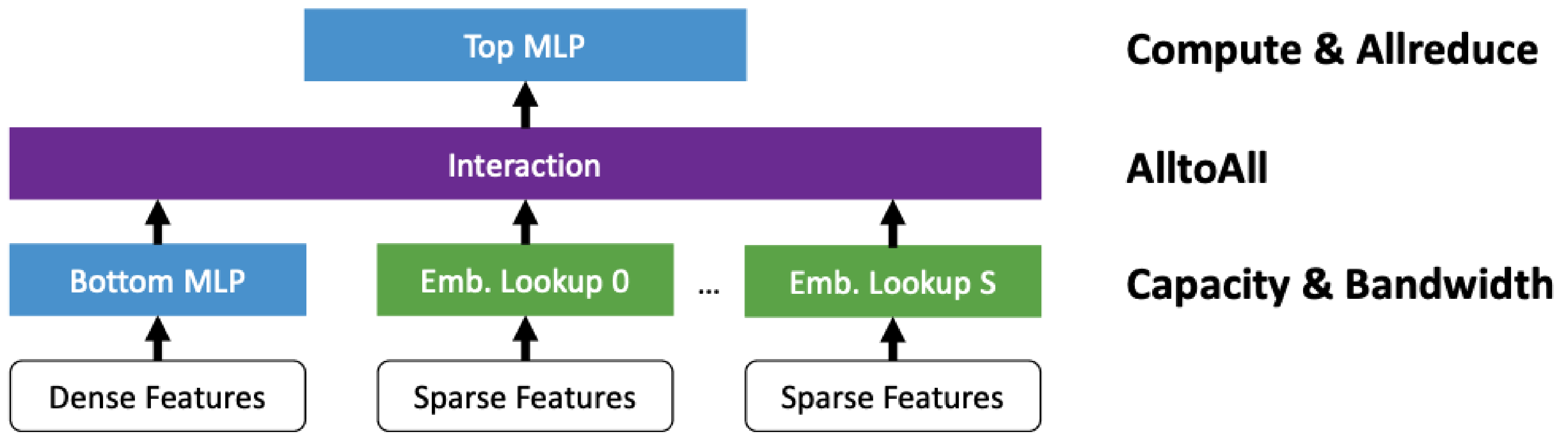
CACHING COLLECTIVE INFORMATION

- **Collective initialization can be costly**
  - Allocation of internal structures and buffers
  - Registration of memory
  - Rendezvous Handshake with peers
- **oneCCL enables amortization of these overheads by caching collective internal representations and reusing them on the subsequent calls**
- Set **coll_attr.to_cache = 1** and **coll_attr.match_id = <match_id>**, where**<match_id>** is a unique string
  - <match_id> should be the same for a specific collective operation across all ranks
  - If the same tensor is a part of different collective operations, match_id should have different values for each of these operations

ONECCL PERFORMANCE WITH DEEP LEARNING RECOMMENDER SYSTEMS

Optimizing Deep Learning Recommender Systems’ Training on CPU Cluster Architectures
D. Kalamkar, E. Georganas, S. Srinivasan, J. Chen, M. Shiryaev, and A. Heinecke
https://arxiv.org/abs/2005.04680

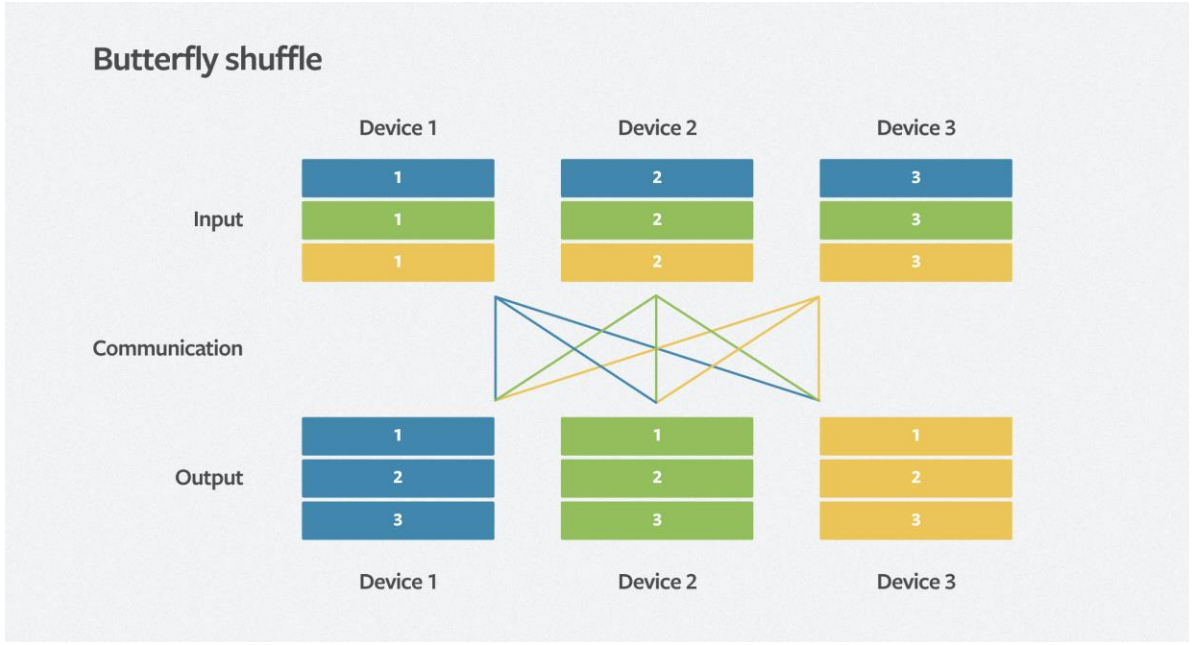
DEEP LEARNING RECOMMENDATION MODEL (DLRM)



- DLRM comprises of MLPs (multi-layer perceptron) and Embedding table look-ups and the corresponding interaction operations
- Stresses all important aspects of the underlying hardware platform at the same time: compute capabilities, network bandwidth, memory capacity and memory bandwidth
- DLRMs mark the beginning of a new era of deep learning workloads

COMMUNICATION ASPECTS

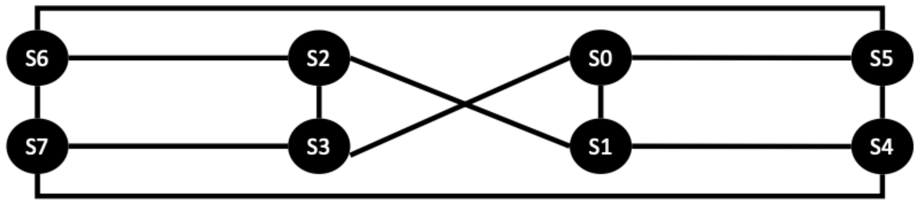
- **Allreduce**
  - Reducing the weight gradients in the backward pass of the MLPs we need ensure overlap with the GEMM compute
  - To reduce overhead of the communication in the SGD:
    - Overlapped the SGD solver with the back-propagation MLP kernels
    - Devote ‘S’ threads for communication of gradient weights, and remaining threads in compute
- **Alltoall**
  - For switching between data and model parallelism during the interaction operation
  - DL frameworks such as PyTorch used to lack primitives for supporting this communication pattern
  - We recently added experimental support for alltoall primitive to their distributed backend



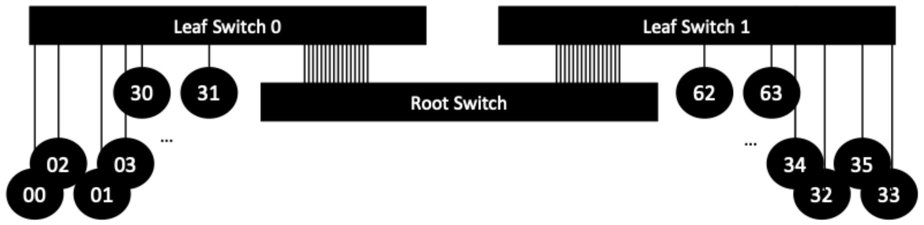
https://ai.facebook.com/blog/dlrm-an-advanced-open-source-deep-learning-recommendation-model/

EXPERIMENTAL SETUP

- **Platform**
  - Intel® Xeon Cascade Lake 8280 system featuring 8 sockets
  - The Platinum series processor offers 3 point-to-point Ultra Path Interconnect (UPI) links
  - 28 cores at an AVX512 turbo frequency of 2.4 GHz and 1.8GHz AVX512 base frequency
  - Memory:6 dual-rank 16 GB DDR4-2666 DIMMs per socket offering 105 GB/s memory bandwidth
- **Interconnect**
  - OPA-100 NICs
  - Topology: Pruned fat-tree with 16 nodes with 32 sockets connected to one switch and then both leaf switches connected with 16 links to the root switch

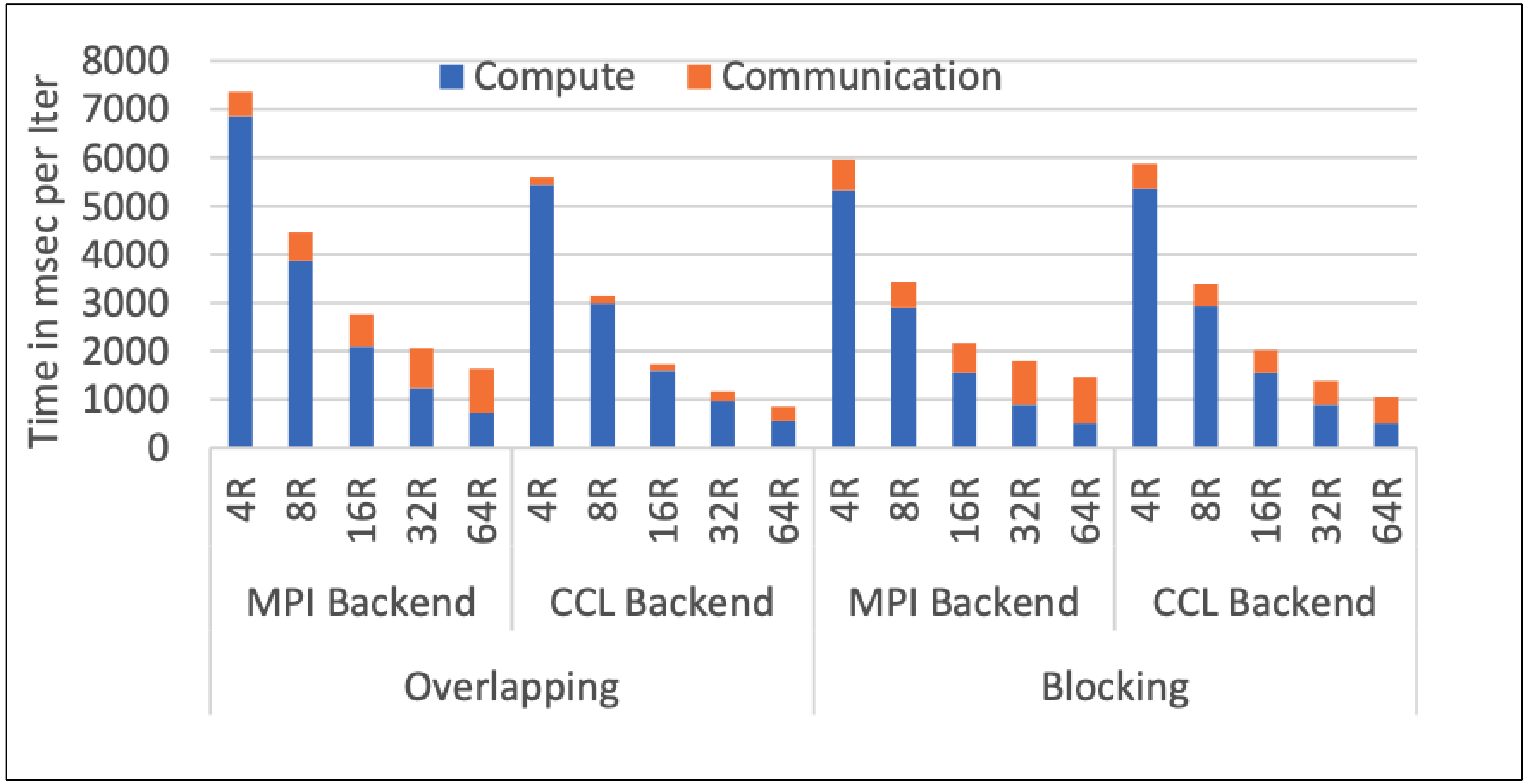


Node Topology



Fabric Topology

COMMUNICATION OVERLAP RESULTS



DLRM "Large" Strong Scaling (4-64 ranks) with and without overlapping

- Compute kernels slowed down due to communication overlap
- PyTorch MPI backend thread was interfering with compute threads
- CCL provides a mechanism to bind the communication threads to specific cores
- Communication threads isolated from compute threads on separate cores
- Reduces interference and enables much better overlap of compute and communication

Intel® Xeon Cascade Lake 8280 system featuring 8 sockets
The Platinum series processor offers 3 point-to-point Ultra Path Interconnect (UPI) links
28 cores at an AVX512 turbo frequency of 2.4 GHz and 1.8GHz AVX512 base frequency
Memory:6 dual-rank 16 GB DDR4-2666 DIMMs per socket offering 105 GB/s memory bandwidth

Intel® OPA-100 NICs
Topology: Pruned fat-tree with 16 nodes with 32 sockets connected to one switch and then both leaf switches connected with 16 links to the root switch

CONCLUSIONS

- **oneCCL provides an interface that matches DL training workload requirements**
- **Easy to integrate into several frameworks**
- **Provides a unified interface that can layer over many underlying interfaces**
- **Leverages OFI/Libfabric to map to underlying hardware**
- **Level0 interface provides portability over range of accelerators**
- **Recent research shows very good performance**
- **Open-source reference implementation available**
  - https://github.com/oneapi-src/oneCCL

LEGAL DISCLAIMER & OPTIMIZATION NOTICE

OPENFABRICS ALLIANCE

2020 OFA Virtual Workshop

THANK YOU