OFFLOADING SCATTER-GATHER VIA CUSTOM ACCELERATORS ON A COPA FPGA NETWORK PLATFORM

Shweta Jain, Andriy Kot, Pallabi Chatterjee, Venkata Krishnan

Intel Corporation

April 26th, 2022

OUTLINE

- Motivation for Scatter/Gather
- POC on COPA FPGA
- Inline Gather Operation
- Lookaside Scatter Operation
- Gather and Scatter Results
- OFI Support
- Future Work

MOTIVATION

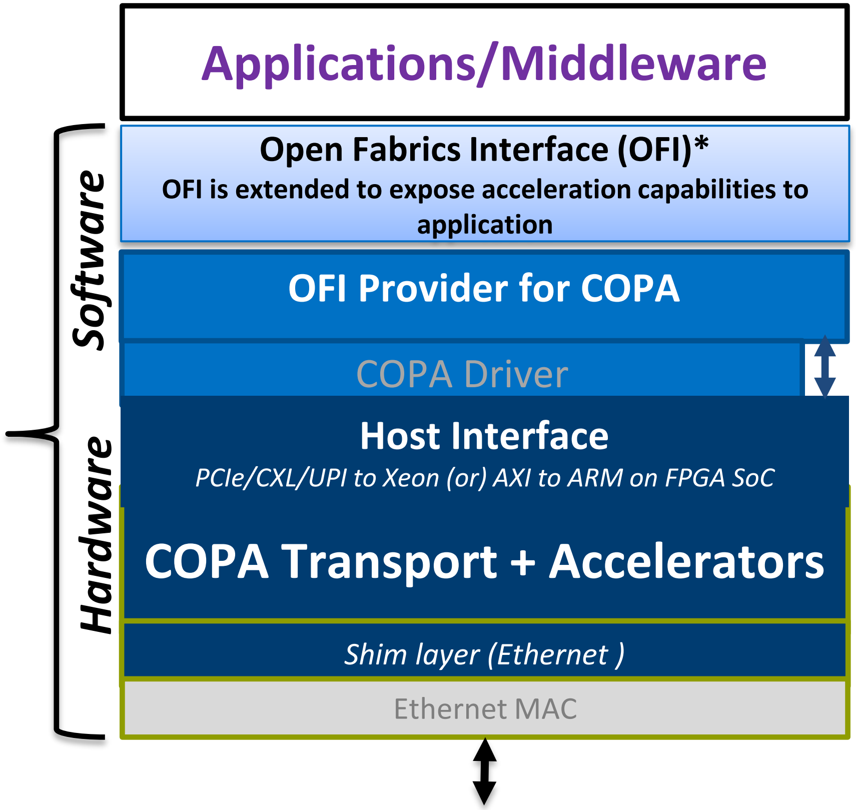
- Improve the performance of HPC middleware used for distributed programming by leveraging the COPA inline/lookaside accelerator capabilities

- Provide an enhanced OFI interface that exposes the acceleration and networking capabilities to upper-layer middleware/applications

COPA FRAMEWORK

Provides a flexible, integrated networking and accelerator framework with programming simplicity

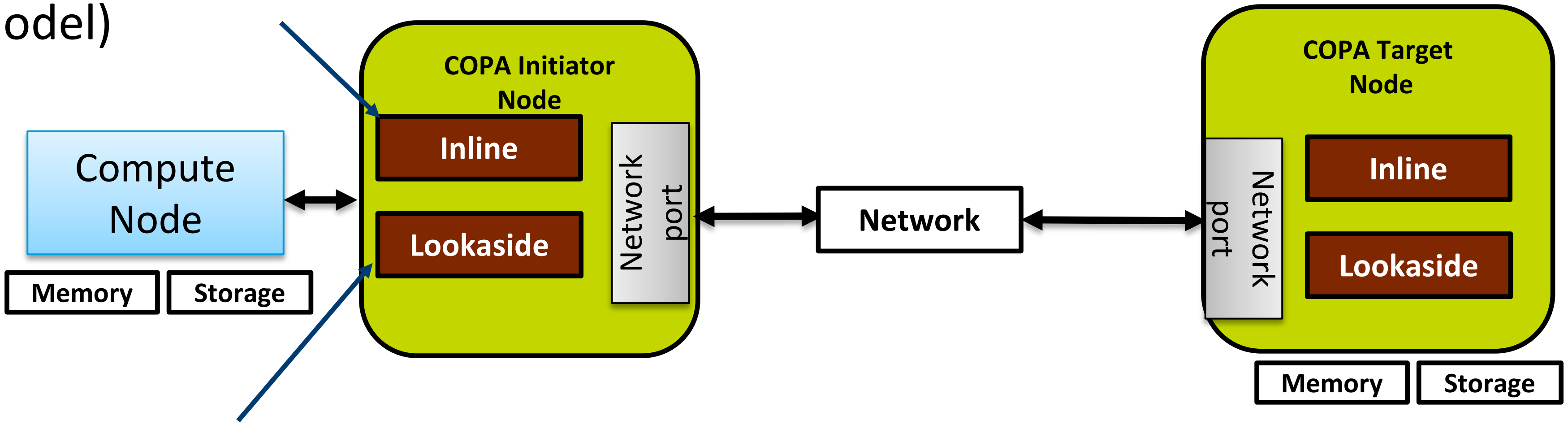
- Architected from the ground-up as a scalable system technology using FPGAs
- HW IP integrates flexible NIC and accelerator capabilities
- SW is based entirely on open standards
- Ease of integration with commercially available networking switches
- Uses RDMA based communication protocols



COPA ACCELERATOR MODELS

Inline accelerators perform compute on data during transmit/receive operation (streaming model)

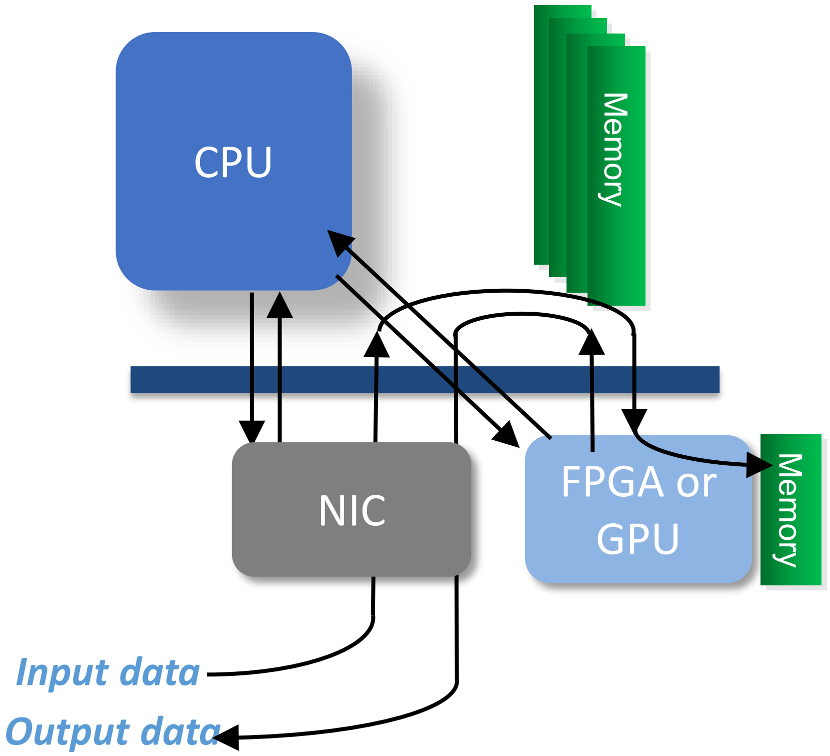
Naturally extends to offloading collectives, reduction, atomics, distributed hash lookup etc.



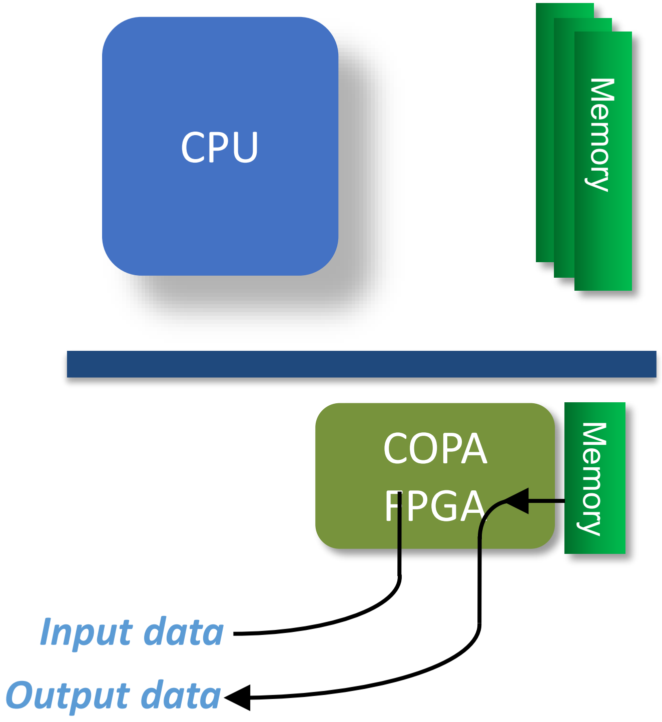
Lookaside accelerators – Traditional acceleration model. However, output data can be directly transmitted to target over network without requiring data movement back/forth to host

COPA REMOTE TRIGGERED ACCELERATION

No agent for orchestrating between accelerator & network (headless or FPGA becomes the head)



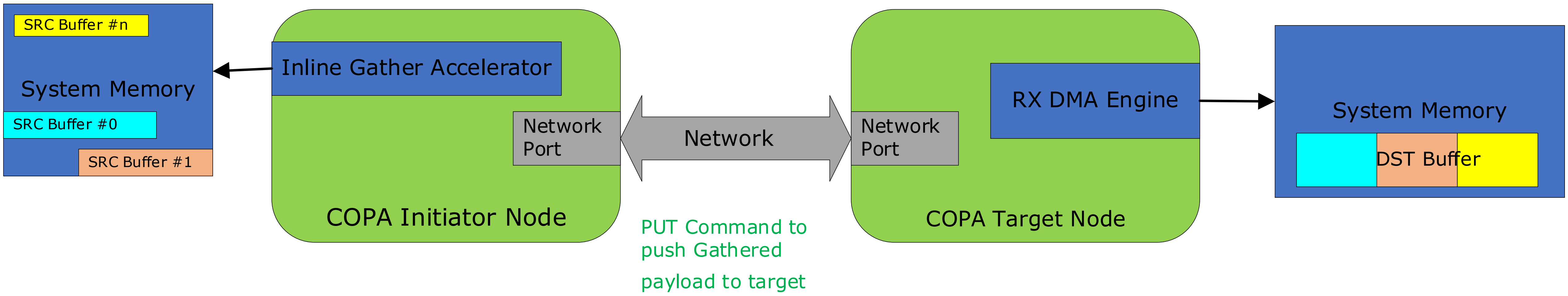
FPGA (or GPU) accelerator + NIC flows



COPA Remote Flow

INLINE GATHER

- **Gather Operation**
  - Combines data from multiple SRC buffers into a single payload and push it to a remote COPA node
  - Implemented as an INLINE COPA accelerator block
  - Performance limited by the available Network Bandwidth



INLINE GATHER CONT’D

VA+x
1000

A
B
C

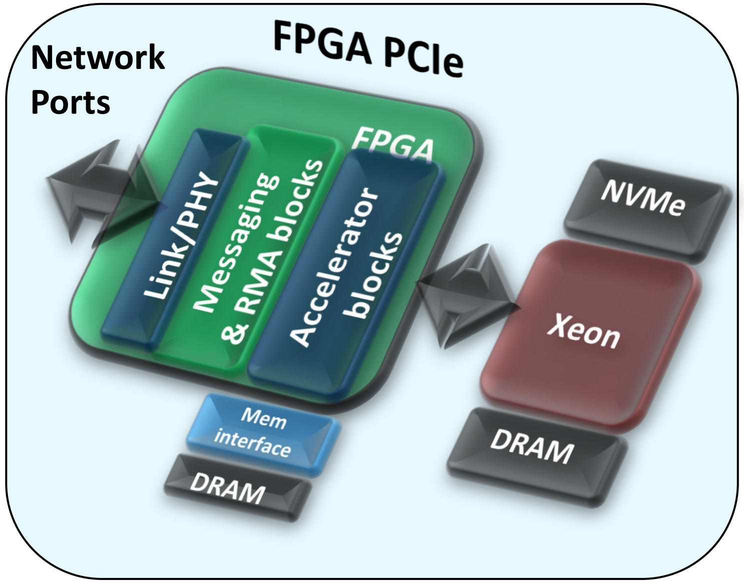
Addr A, len A
Addr B, len B
Addr C, len C
Addr D, len D

16 bytes per entry - 64 bits addr, 16 bit length

MAX_BATCH entries

MAX_BATCH = 256 entries

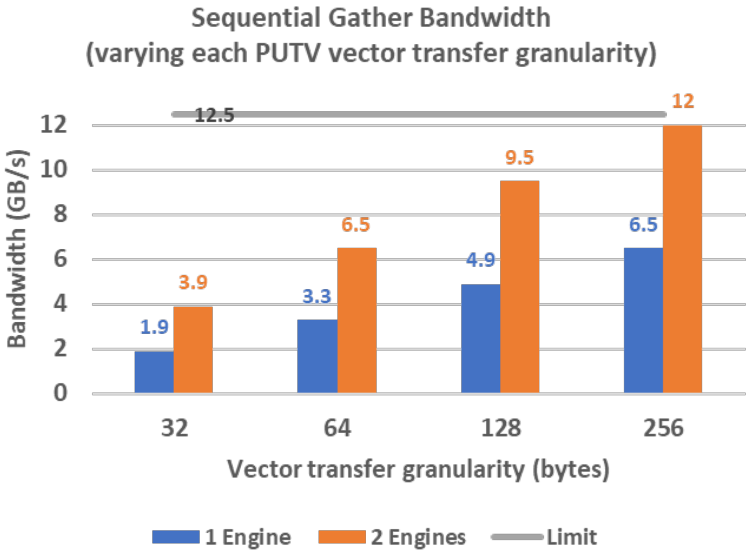
Batch buffer – source buffers

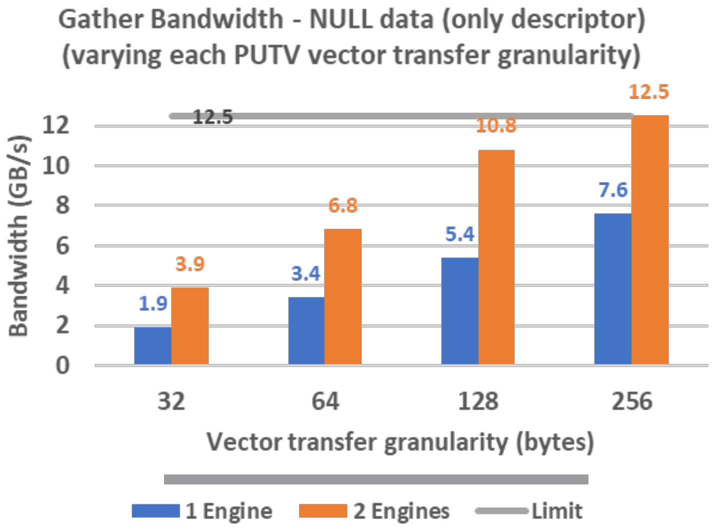


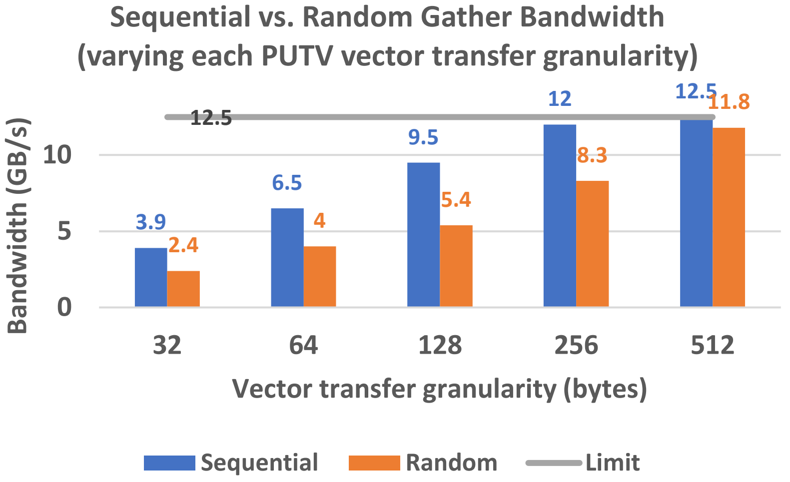
- **POC is based on COPA FPGA framework**
- Supported COPA Commands: **PUTV, MSGV**
- Outgoing PUT/MSG data is read (gathered) from multiple local source buffers
- Limitations:
  - Combined length of data “gathered” can’t exceed 9KB
  - Operates on physical addresses
  - Maximum achievable ~12.5GB/s (256b data path @ ~400MHz)

GATHER BENCHMARK RESULTS

- **Sequence of Gather operations – Gather payload on Initiator node memory and Push to Target node memory**
- **Performance – sequential and random gather**
  - Granularity of transfer varied from 32B to 256B (Memory BW is < 16GB/s & FPGA data path at ~12.5GB/s). PUTV descriptor fetch followed by read of batch buffer stalls pipeline (seen when using one engine) – Having 2 engines addresses the problem (weak ordering model)
  - Random gather shows impact of DRAM pages open/close without reuse
  - Design implemented on Stratix 10 FPGA card

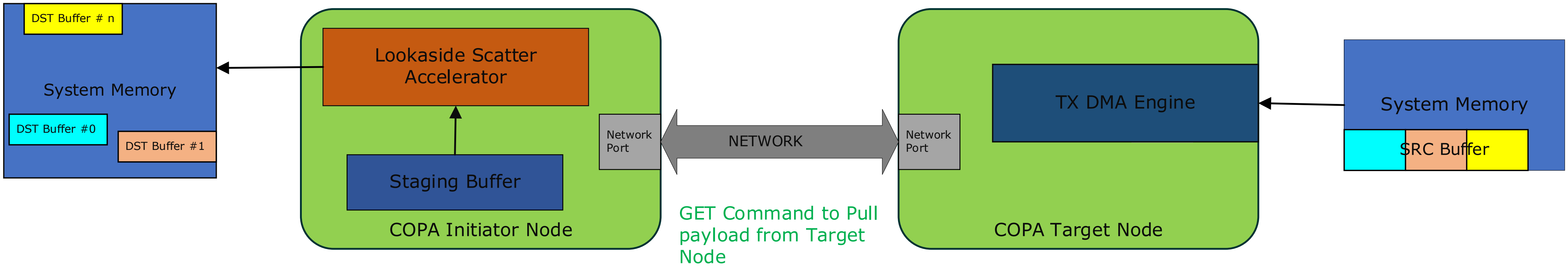






LOOKASIDE SCATTER

- **Scatter Operation**
  - Distributes a single payload from remote COPA node into multiple DST buffers
  - Implemented as a LOOKASIDE COPA accelerator block
  - Requires an intermediate staging buffer to store the incoming payload
  - Performance limited by the available Network Bandwidth and the System Memory Bandwidth

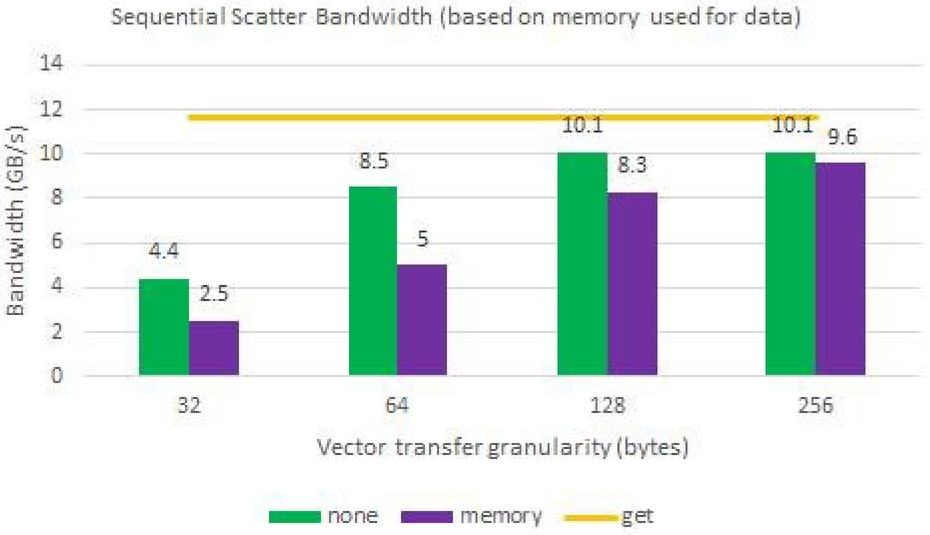


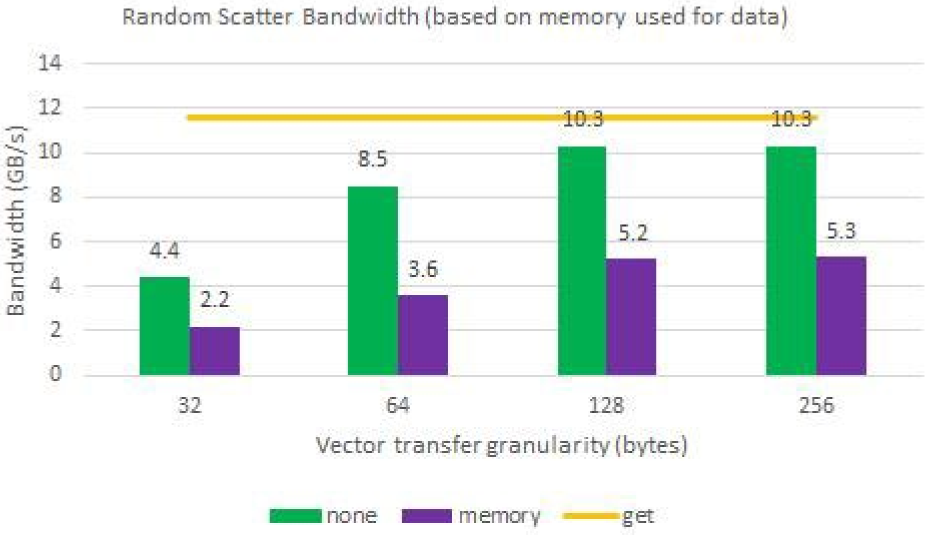
LOOKASIDE SCATTER CONT'D

- POC is based on COPA FPGA framework
- Batch buffer format identical to PUTV
- The address/length of each destination endpoint is specified by the batch buffer
- Supported COPA command : **GETV**
  - Incoming GET data is stored within Lookaside accelerator memory
  - The completion of GET request triggers the Scatter operation within Lookaside accelerator based on the batch buffer entries
- Limitations:
  - Operates on physical addresses
  - Individual segments should be a multiple of 32

SCATTER BENCHMARK RESULTS

- **Sequence of Scatter operations – Pull payload from Target node memory and do a Scatter operation on the Initiator Node memory**
- **Performance**
  - Granularity of transfer varied from 32B to 256B
  - Peak performance dependent on speed of GET operation
  - Design implemented on Stratix 10 FPGA



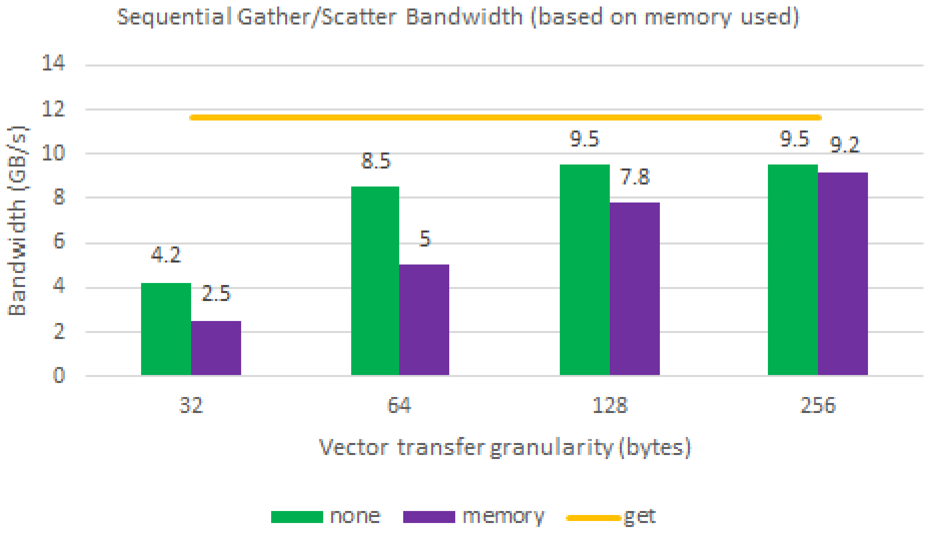


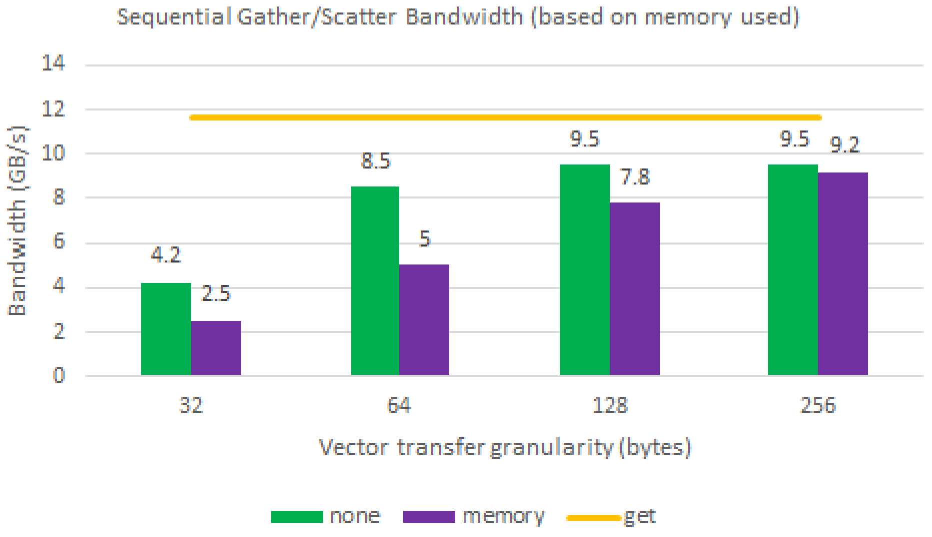
- **POC uses physical addresses**
- **Max. achievable ~11.6 GB/s**

COMBINED GATHER/SCATTER FLOW

Sequence of Operations

- Perform Gather Operation on Initiator Node
- Push the Gathered payload to the Target node
- The Initiator node posts a remote trigger Scatter request to Target Node
- Target node performs Scatter Operation





OFI Software Extension

- Low-level scatter and gather operation will be incorporated into respective calls of COPA OFI provider: fi_readv, fi_readmsg, fi_writev, fi_writemsg and fi_inject_write
- Scatter is an optional capability based on the availability of lookaside accelerator it will adaptively turned on or off
- Additional inline acceleration added via extensions is compatible and enabled together with scatter/gather support

FUTURE WORK

- Migration to Intel AgileX FGPA cards
- Dense and Sparse Matrix Algebra
- Shuffle Operations (High Performance Data Analytics)

BACKGROUND

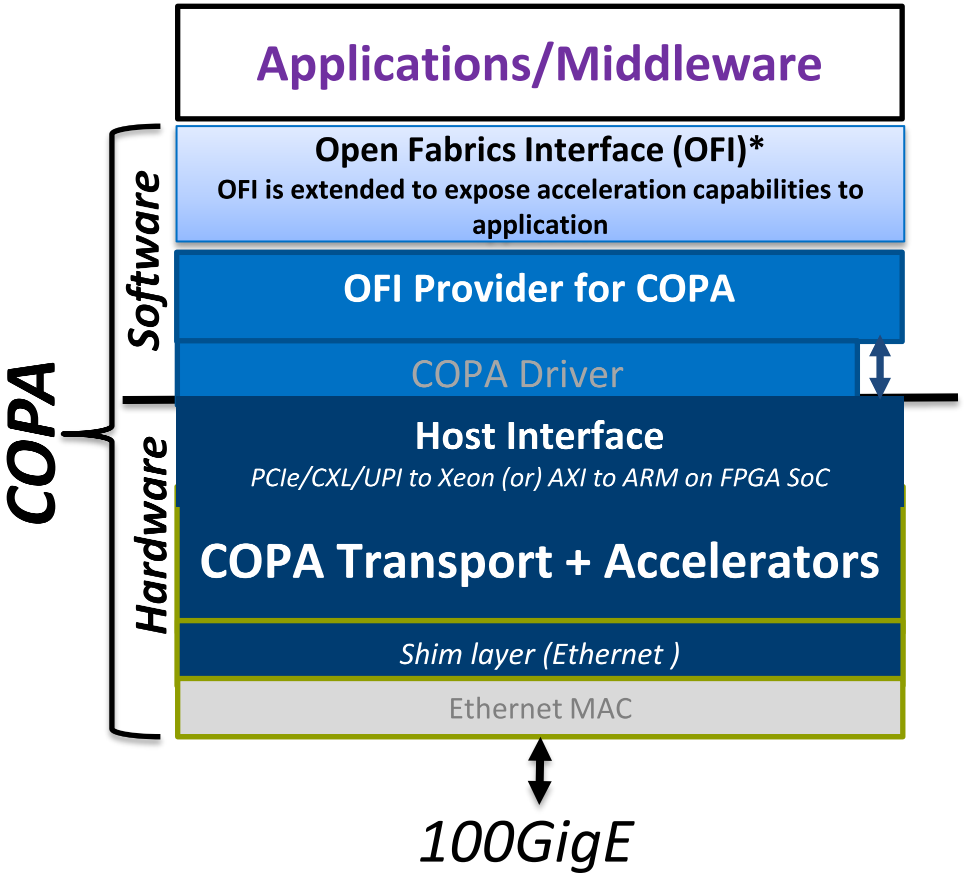
COPA † IS THE POC PLATFORM FOR OFI EXTENSIONS
(A SOFTWARE/HARDWARE FRAMEWORK FOR DISTRIBUTED FPGA COMPUTING)

Provides an integrated networking and accelerator framework with programming simplicity

- Supports RDMA (PUT/GET) based communication over commodity networks.
- Accelerators invoked as part of communication.
- Familiar environment developed around open standards (e.g. libfabric/OFI)

Customizable framework for specific deployments

- Provides a modular architecture - can add necessary IP (accelerator) blocks and new features for a customized solution

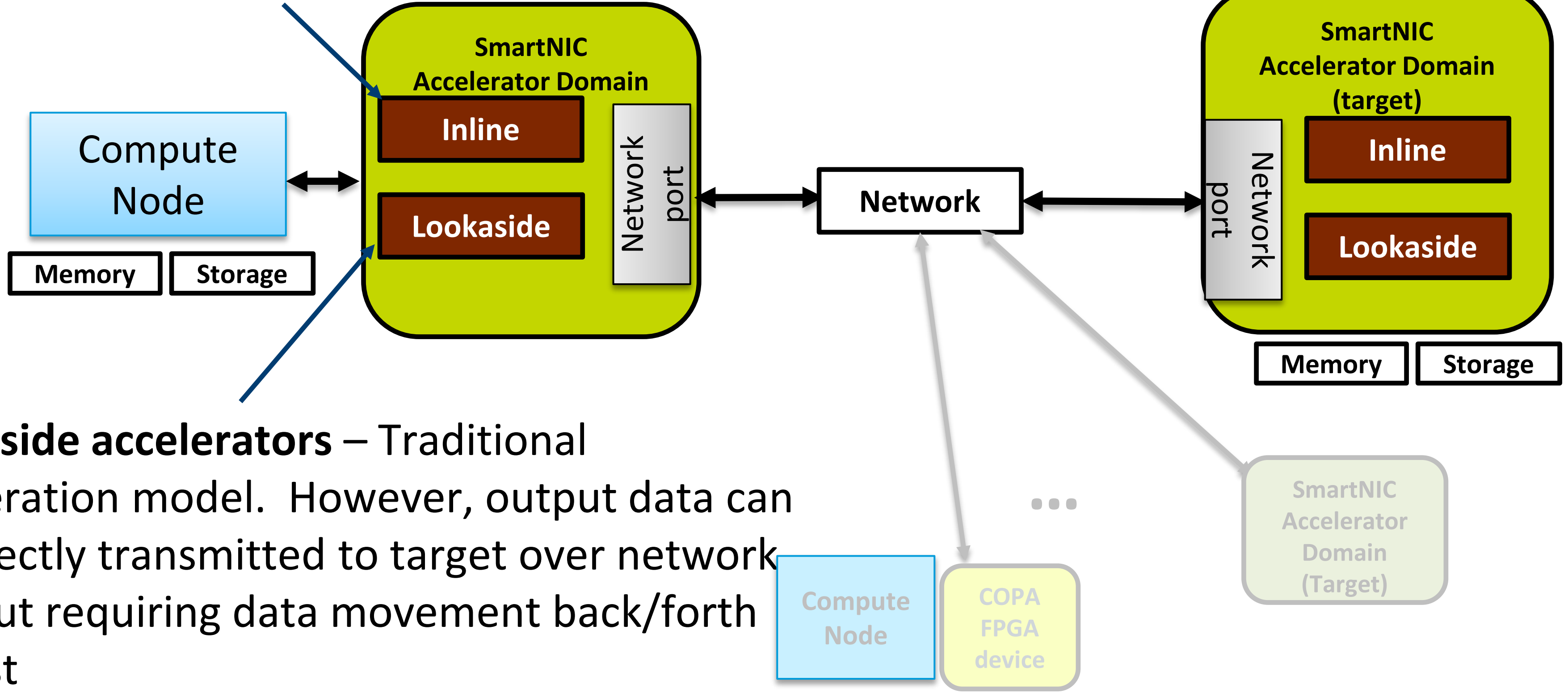


†**COPA** = **CO**nfigurable network **P**rotocol **A**ccelerator

ACCELERATOR MODELS (INTEGRATED WITH NETWORKING)

Inline accelerators perform compute on data during transmit/receive operation (streaming model)

Remote Mode Inline/Lookaside accelerators can be triggered by incoming packet. **No host/OS involvement**



Naturally extends to offloading collectives, reduction, atomics, distributed hash lookup etc.

Could hang off a switch port (headless) or be an integral part of the switch

Lookaside accelerators – Traditional acceleration model. However, output data can be directly transmitted to target over network without requiring data movement back/forth to host

APPROACH – USE OFI (WITH EXTENSIONS)

Extend a network API to include acceleration support to support a truly scalable model

- **Extending an accelerator API (e.g. OpenCL) to support networking is not scalable**

Applications can use OFI directly.

Middleware need to be extended to use underlying accelerator support

Intel® MPI Library	MPICH	Open MPI SHMEM	Sandia SHMEM	GASNet	Charm++	Clang UPC	Chapel	Others: rsockets, PMDK, Spark, ZeroMQ, TensorFlow, MxNET, NetIO, Intel MLSL, …

libfabric Enabled Middleware

libfabric

Advanced application oriented semantics

Tag Matching	Scalable memory registration	Triggered Operations	Remote Completion Semantics	Multi-Receive buffers	Shared Address Vectors	Unexpected Message Buffering

Streaming Endpoints	Reliable Datagram Endpoints	**Accelerators inline/lookaside local & remote**

OFI Providers

Sockets TCP, UDP	Verbs	Cisco usNIC	Cray GNI	Intel OPA, PSM	Shared Memory	Mellanox UCX	IBM Blue Gene	Network Direct	HPE Gen-Z	RxM, RxD,	**NEW PROVIDER**

COPA SOFTWARE STACK

- Applications / Middleware
- Open Fabrics Interface (OFI)
- OFI provider for COPA (libfabric)
- COPA hardware library (libchw)
- COPA device driver
- COPA FPGA hardware

OFI provider for COPA (libfabric)

- Open Fabrics Interface (OFI) implementation
- Main application programming interface to COPA functionality
- Extended for acceleration
- All access to COPA FPGA through COPA hardware library (libchw) for kernel bypass

COPA hardware library (libchw)

- Main system programming interface to COPA FPGA hardware (ioctl syscalls)
- Opens device and mmap's hardware command queues and event queues into user process address space
- Optionally hardware emulation mode - COPA functional model

COPA device driver

- Memory map CSRs and command queues at initialization time
- Initialize E2E connection table
- Provides memory pinning and virtual-to-physical address translation
- Not accessed on critical path - Userspace application software interacts with hardware using mmaped command queues

HOW DO WE ACCELERATE APPLICATIONS?

Option 1:

Accelerate / improve middleware interface standards

For example:

OpenSHMEM v1.6:

- Non-blocking collectives
- Per-PE fence

Option 2:

Application-aware accelerator optimizations

Extend middleware interfaces

For example:

- FI_ACCELERATION*
- SHMEMX interfaces

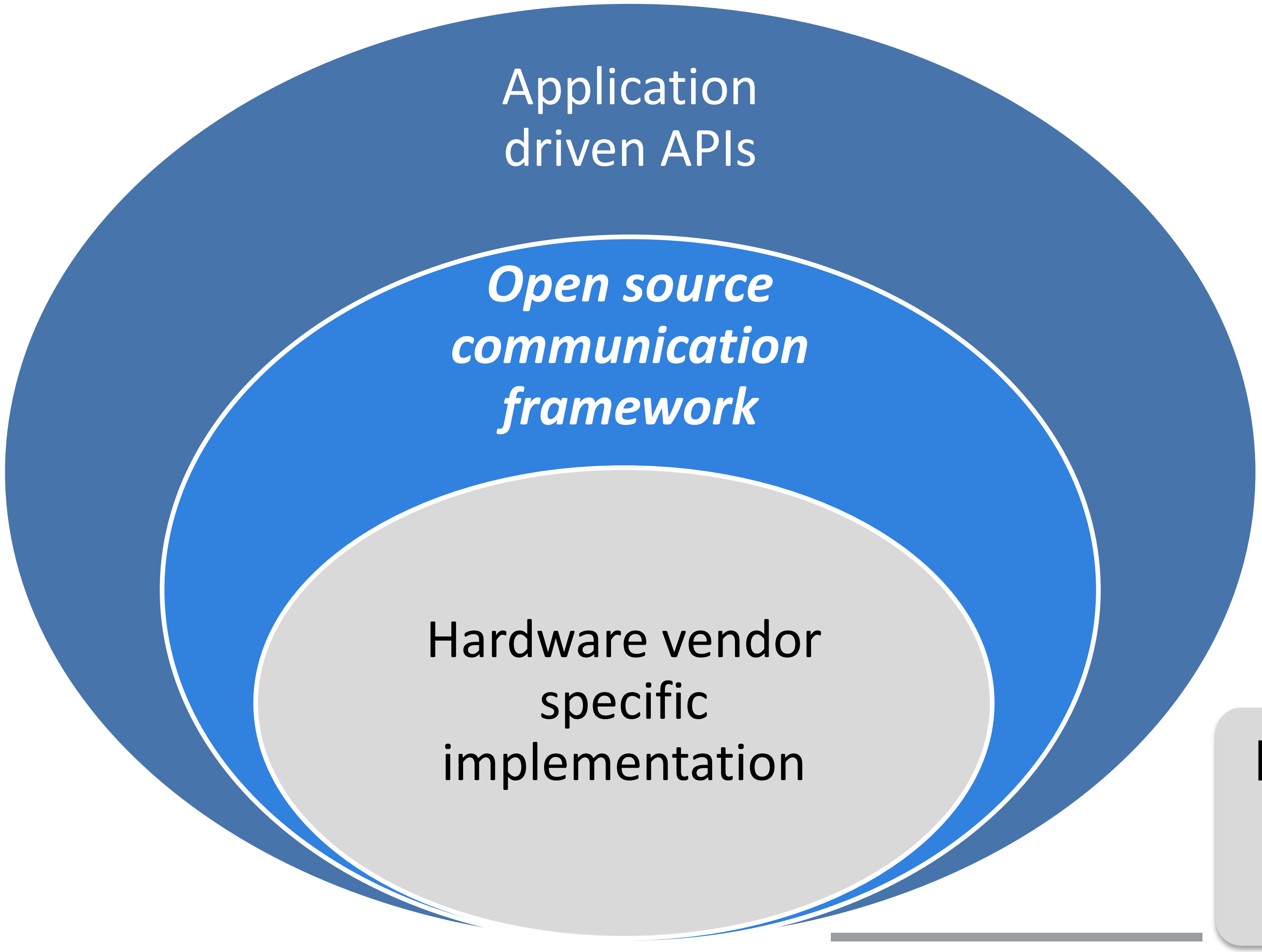
Offload custom app-specific patterns

For example:

- Custom collective ops
- Data transformation (e.g. compression, filtering)

* Enhancing OFI for Invoking Acceleration Capabilities on an Integrated Networking/Accelerator Platform (COPA) (OFAWS 2020)

CURRENT VISION OF SOLUTION



Based on internal hardware prototyping – FPGA-based

APIs targeting application use of specific accelerations

Extend existing communication framework to support acceleration functions

Define mechanism to pass input/output parameters and invoke acceleration

OFI COPA PROVIDER

- **Full featured OFI provider**
- **Only small changes needed to add acceleration to existing OFI-enabled middleware and applications**
- **Temporary until official OFI support**
- **Minimal OFI extensions to enable "inline" and "lookaside" COPA acceleration**
  - Extend semantics of data structures and operations
  - Define new FLAGS for acceleration
- **Implements a wide variety of interfaces to support many kinds of HPC middleware**
  - FI_MSG, FI_TAGGED, FI_RMA
  - FI_PROGRESS_MANUAL, FI_THREAD_COMPLETION, FI_AV_MAP
  - FI_EP_RDM

ENABLE ACCELERATION

- **New FI_ACCELERATION flag informs provider application wants inline accelerator to be invoked during a data movement operations**
- **FI_ACCELERATION flag can be set on the endpoint object to invoke acceleration on all endpoint data movement operations**
  - fi_control() with FI_SETOPTS
- **Alternatively, FI_ACCELERATION flag can be specified for individual data movement operations**
  - fi_write_msg()
  - fi_read_msg()

ACCELERATOR OUTPUTS

- Output data may be provided as a result of acceleration
- Available for endpoints bound to a completion queue initialized with data format
  - FI_CQ_FORMAT_DATA
  - FI_CQ_FORMAT_TAGGED
- FI_ACCELERATION flags, etc., are set in the flags field
  - FI_CQ_FORMAT_MSG

- Normally the completion entry data field is for remote metadata
- Extend the data field semantics for initiator acceleration output

```
struct fi_cq_data_entry {
  void      *op_context; /* operation context */
  uint64_t flags;        /* completion flags */
  size_t   len;          /* size of received data */
  void      *buf;        /* receive data buffer */
  uint64_t data;         /* completion data */
};
```

LOOKASIDE ACCELERATION

- **Local operation – no fabric communication involved**
- **Complex accelerators that do not fit in the packet pipeline (inline acceleration)**
- **Same mechanism as inline to invoke lookaside acceleration**
  - fi_read(), fi_write(), etc.
  - FI_ACCELERATION
- **Lookaside accelerator flags**
  - FI_LOOKASIDE_ACCELERATION_*
- **Current restrictions**
  - physically contiguous memory for all inputs and outputs

COPA SYSTEM ORGANIZATION