Recent Advances in UCX for AMD GPUs

Edgar Gabriel
Arun Chandran

AMD
together we advance_

Outline

- New ROCm features in UCX 1.15
- v2 protocols status update
- Non-temporal buffer transfers on “Zen 3”/”Zen 4” architectures
- Summary

AMD
together we advance_

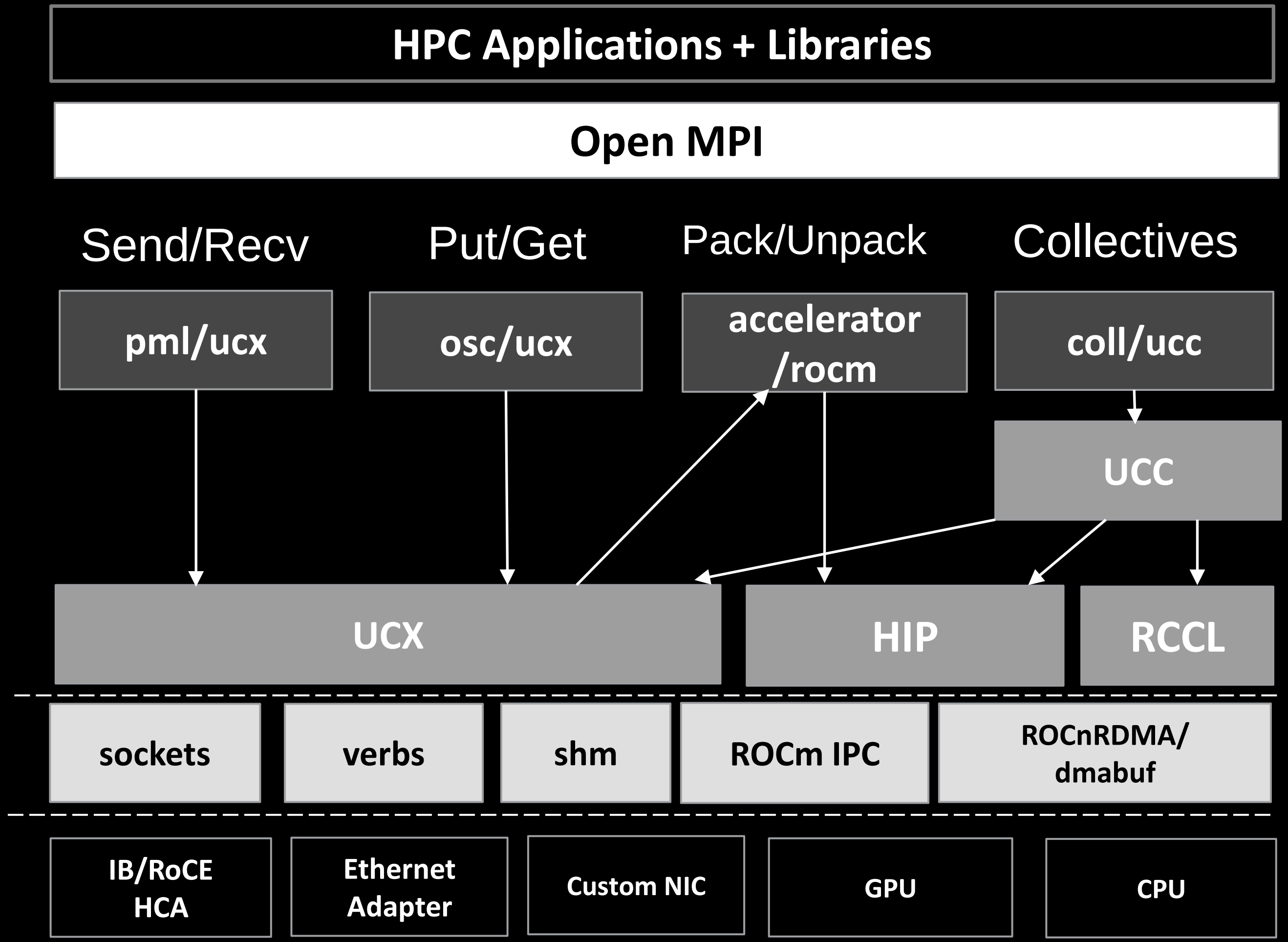
ROCm support in UCX

- **ROCm/COPY**: data transfer between host and device memory within a single process
- **ROCm/IPC**: data transfer between device memories of different processes on the same node
- GPUDirect RDMA for communication between processes on different nodes
- Memory type detection of ROCm memory
- Memory hooks for ROCm memory allocations

AMD together we advance_

ROCm Aware Open MPI Software Stack with UCX and UCC

- Recommended software stack for InfiniBand and RoCE networks
- Most stable and best tested configuration



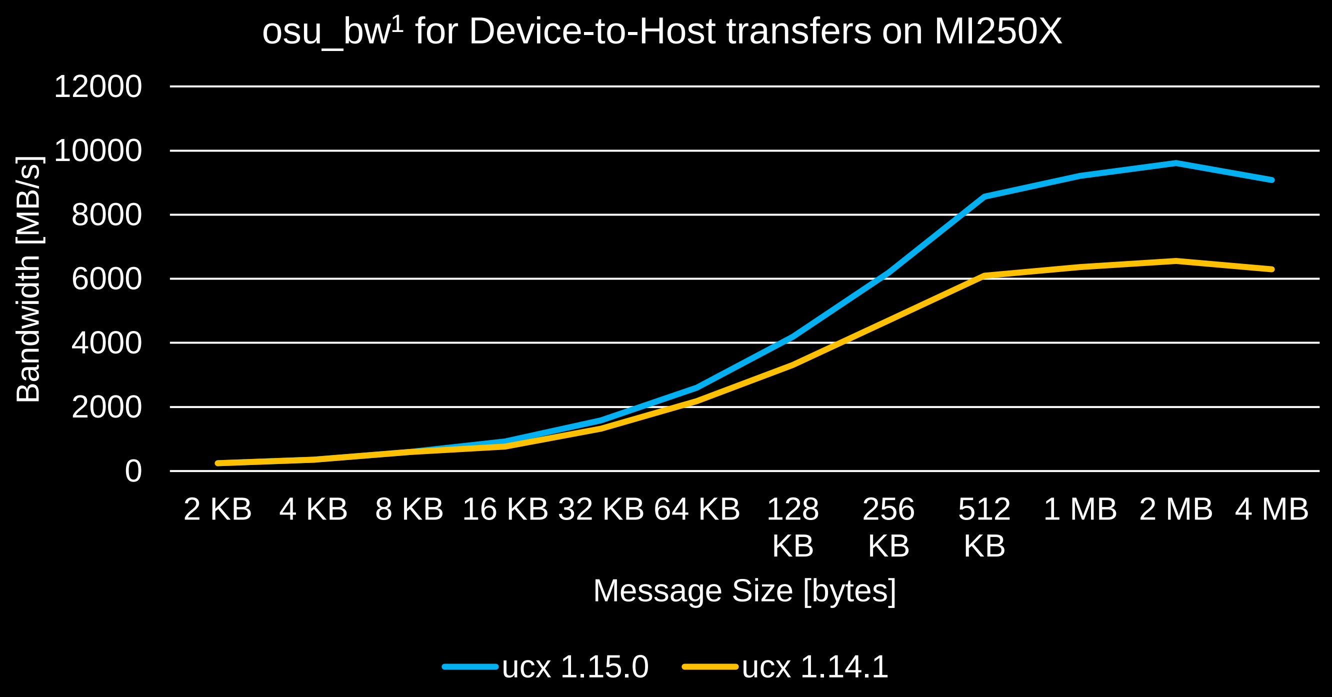
AMD together we advance_

New ROCm Features in UCX 1.15

Asynchronous ROCm/copy zero-copy operations

- Allows to overlap multiple stages in some protocols
- Potential to improve device-to-host and host-to-device transfers

- Unified progress function for ROCm/copy and ROCm/ipc components



[1] OSU Benchmark Suite https://mvapich.cse.ohio-state.edu/benchmarks/ (BSD License).

New ROCm Features in UCX 1.15

dma-buf support for ROCm devices

- dma-buf: Linux kernel subsystem providing a framework for sharing buffers across multiple devices
  - For example, RDMA capable network adaptor accessing a GPU device buffer
- Long-term replacement for the ROCnRDMA kernel component
- ROCm release 5.6 introduced functionality to export a device buffer for dma-buf sharing

AMD
together we advance_

New ROCm Features in UCX 1.15

dma-buf support for ROCm devices

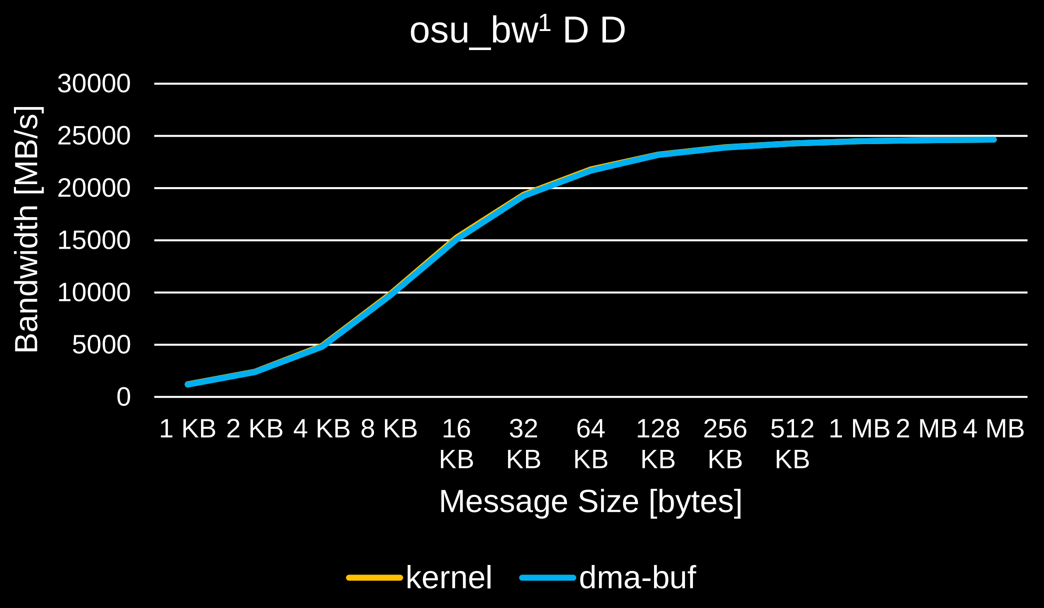
- Three components required to use dma-buf-based sharing for RDMA capable NICs
  - ROCm version with support for exporting dma-buf handle (`hsa_amd_portable_export_dmabuf()`)
  - Version of the libibverbs that supports dma-buf-based memory registration (`ibv_reg_dmabuf_mr()`)
  - Linux kernel with certain features enabled (CONFIG_DMABUF_MOVE_NOTIFY, CONFIG_PCI_P2PDMA)

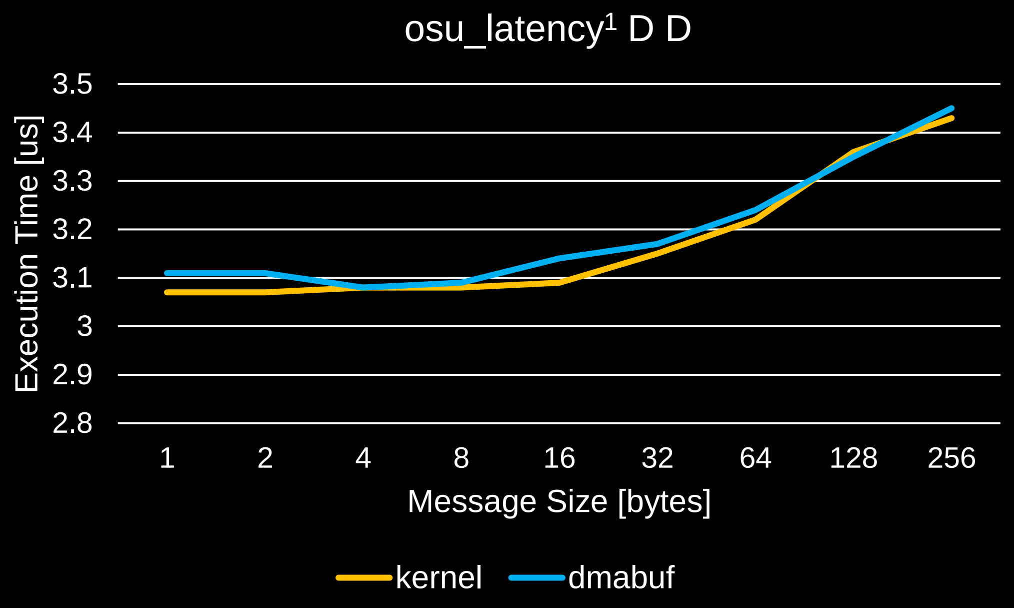
AMD
together we advance_

New ROCm Features in UCX 1.15

dma-buf vs. ROCnRDMA kernel component

- 200Gb InfiniBand, mofed 5.9-0.5.6.0
- MI210 GPUs
- UCX 1.15.0, Open MPI 5.0, osu benchmarks 7.2.0



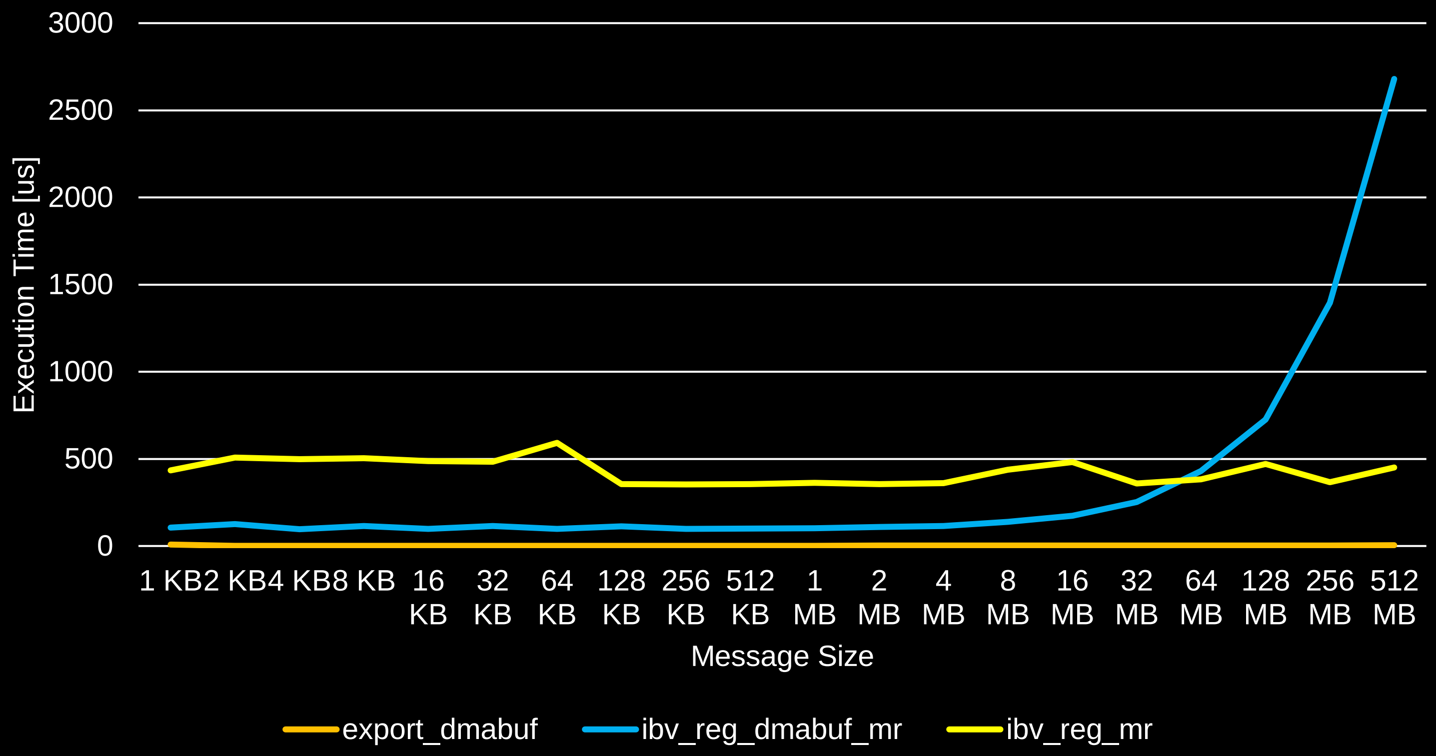


[1] OSU Benchmark Suite https://mvapich.cse.ohio-state.edu/benchmarks/ (BSD License).

AMD
together we advance_

New ROCm Features in UCX 1.15

- Dma-buf vs. kernel component: registration costs
  - Ubuntu 22.04 with custom compiled Linux kernel 5.15
  - ROCm 5.7.1
  - mofed 5.9-0.5.6.0



AMD
together we advance_

New ROCm Features in UCX 1.15

- Setting device_id for AMD GPUs
  - PCIe BDF for each GPU stored and can be used (e.g., for distance calculations)
  - Easier identification in debugging output
- Added logic for GPU assignment of ROCm memory domains
  - UCX uses internally the ROCm runtime layer functionality
  - Runtime layer does not have the notion of ‘current device’ that has been set (e.g. hipSetDevice())
  - Revamped logic allows to determine current device for some scenarios (interception of device allocation using memory hooks, HIP_VISIBLE_DEVICES)

v2 Protocols Status Update

- Intra-node correctness for ROCm device buffers ✓
- Inter-node correctness for ROCm device buffers ✓
- Intra-node performance for ROCm device buffers
  - Point-to-point ✓
  - Collectives (✓) some overhead (20-50%) observed for short messages compared to ucx 1.15 for some collective operations
- Inter-node performance for ROCm device buffers
  - Point-to-point (✓)
    - RoCE ✓
    - IB still under investigation
  - Collectives
    - RoCE ✓
    - IB still under investigation

AMD
together we advance_

Non-Temporal Buffer Transfer

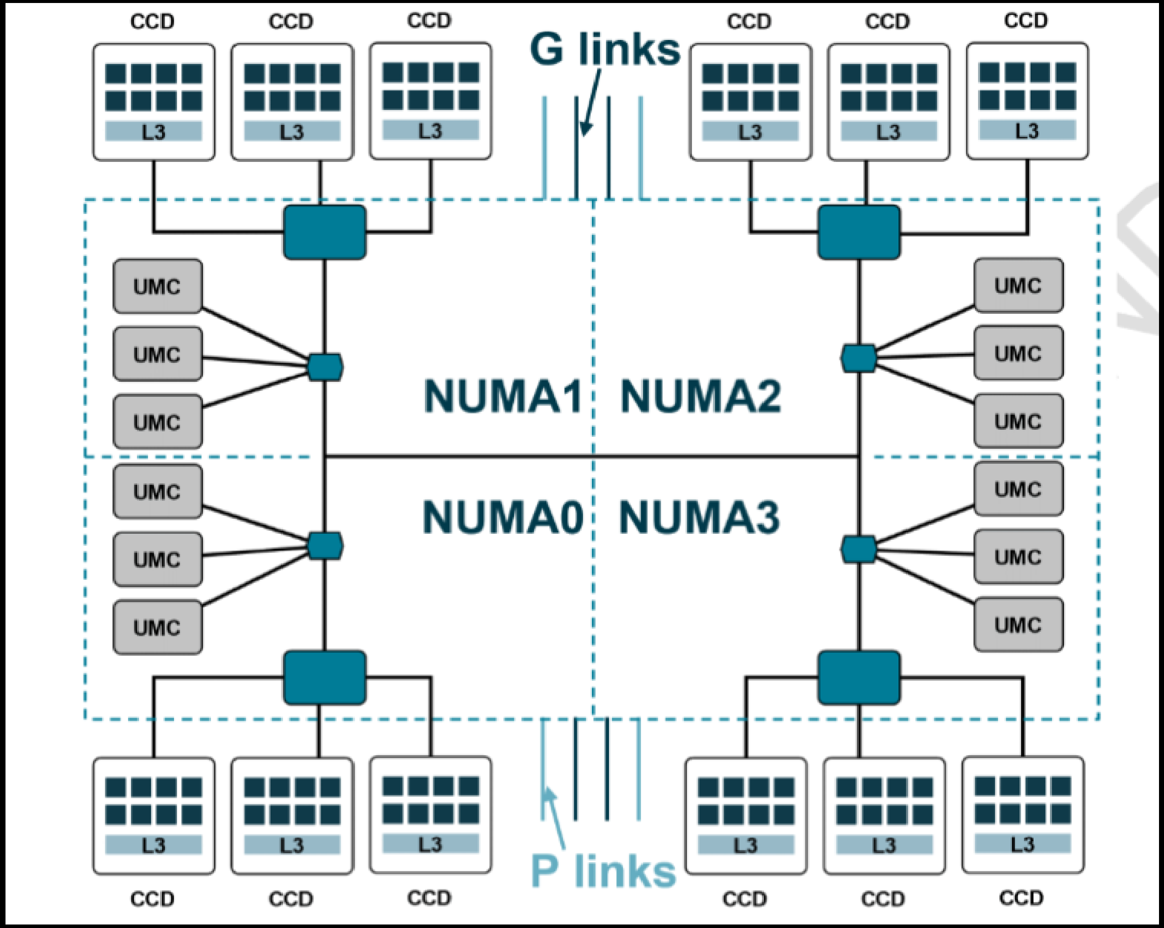
AMD
together we advance_

Hybrid MPI Applications on “Zen 3”/”Zen 4”

- Processors based on AMD’s “Zen 3”/”Zen 4” architecture typically organize CPU cores into clusters of eight or more that share a common L3 cache(CCD).
- Hybrid MPI applications often employ 'n-ranks X m-threads' configuration

  n <= Number of unified L3caches in system
  m >= Number of cores that share a common L3cache

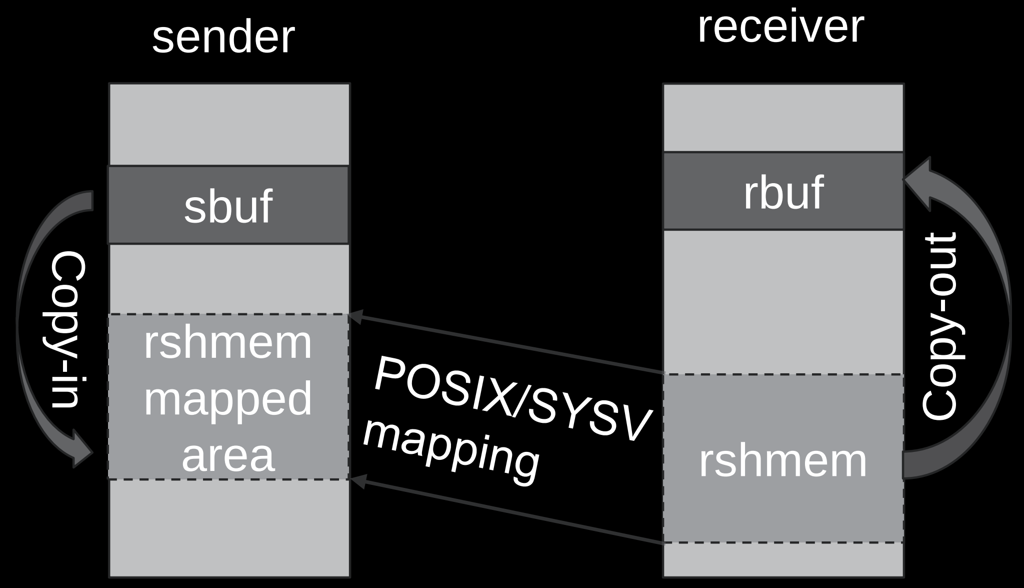
  For example: 16 ranks X 8 OpenMP threads combination on a single “Milan” node
- Non-temporal buffer transfers try to improve the data transfer speed/application performance for such hybrid workloads.
- Communication libraries use copy-in, copy-out (shared memory) or single copy mechanisms to move data between ranks.
- Only UCX PML is considered for optimization here: https://github.com/openucx/ucx/pull/9408



AMD
together we advance_

Handling UCX Copy-in, Copy-out (CICO)

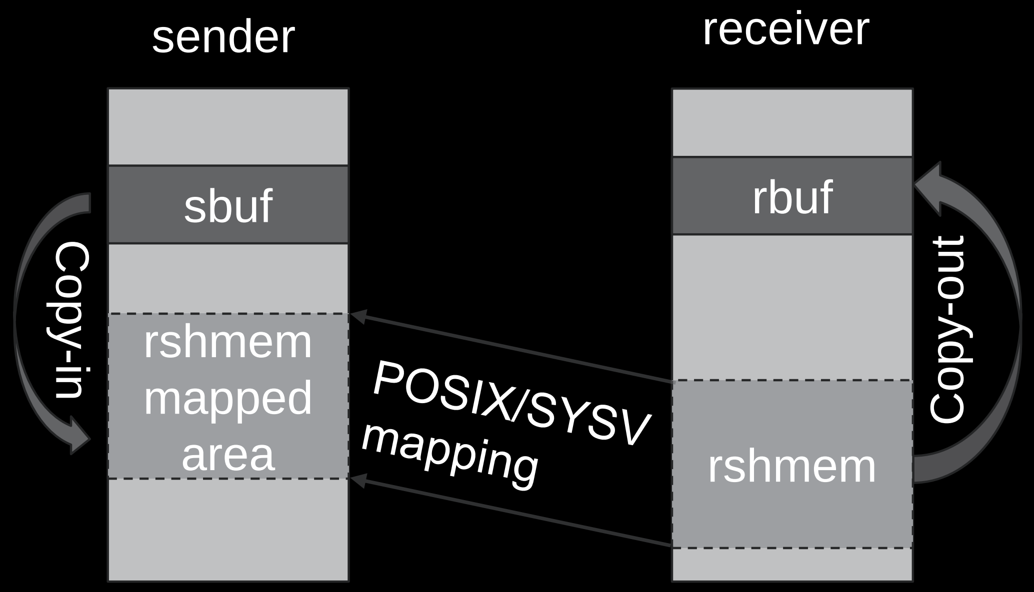
- Non-temporal buffer transfer targets the hybrid workloads where sender and receiver do not share a common L3 cache (i.e., pinned to two different CCD)
- Use two distinct mechanism in place of glibc's memcpy for both copy-in and copy-out
  - New Copy-in routine:
    - Copy sender's buffer to receiver's shared memory
    - Use non-temporal store instead of normal store instructions while storing the data to shared memory destination
  - New Copy-out routine:
    - Copy from shared memory to receiver's buffer
    - Use 'Loads with PREFETCHNTA' instead of normal load instruction while loading from shared memory



AMD
together we advance_

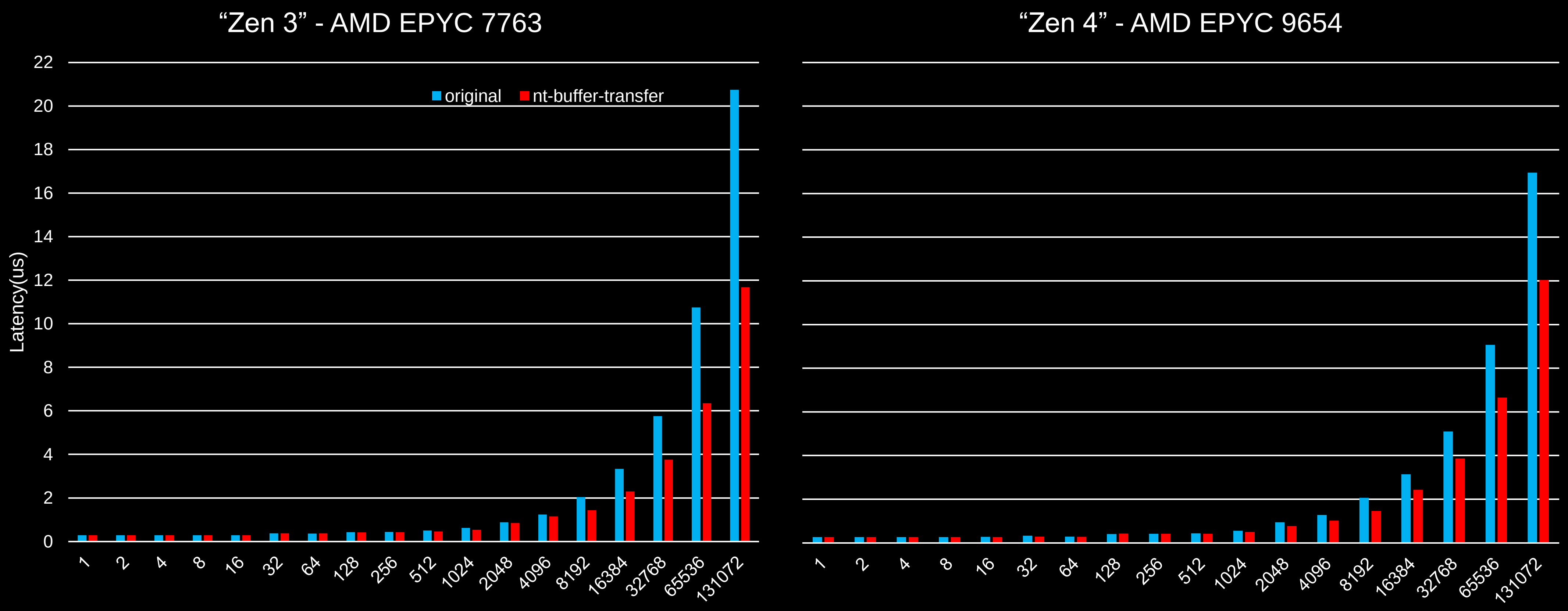
Handling UCX Copy-in, Copy-out (CICO)

- Advantages of using non-temporal load and store instruction
  - Reduces data transfer latency by circumventing cache-to-cache data transfers, which tend to be slower, when ranks/processes are situated in different CCD
  - Reduces the cache pollution, tends to keep only the application buffer in the caches



AMD
together we advance_

CICO osu_latency[1] Benchmark Results (map-by l3cache)



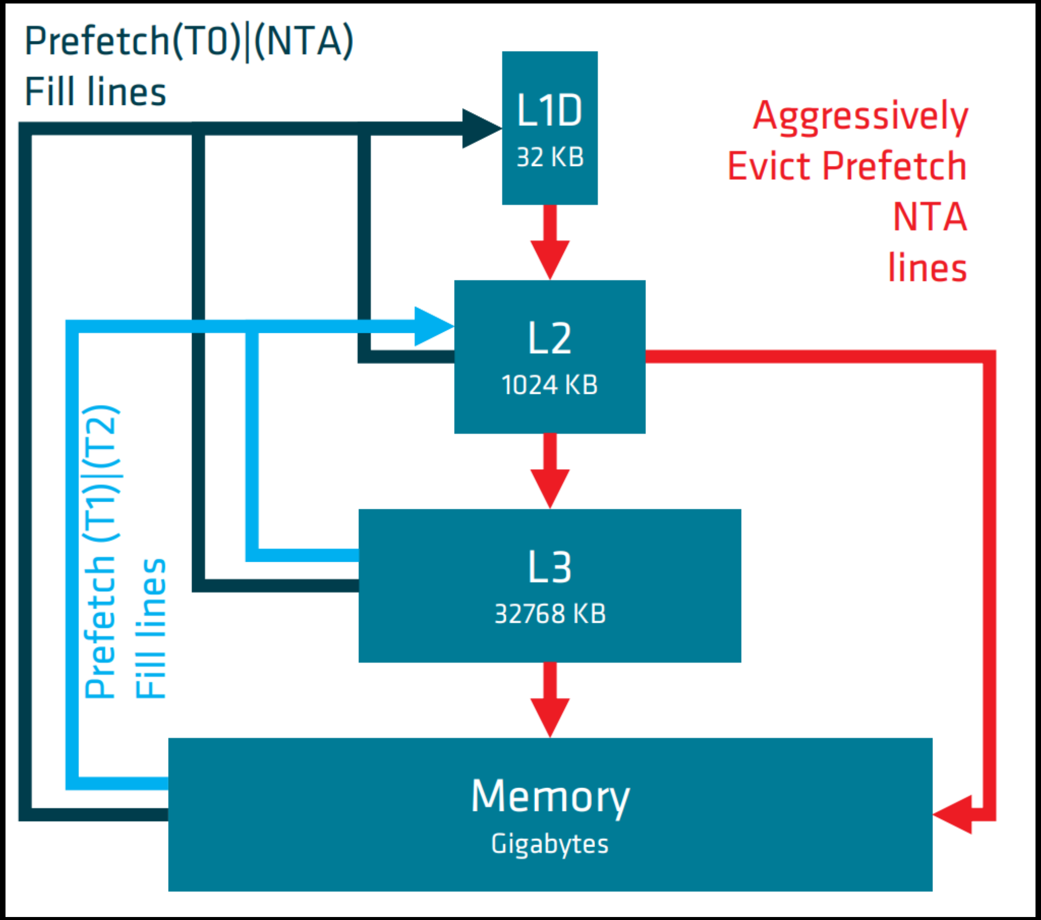
Gain: "Zen 3" up to 43 % @128 KB, "Zen 4" upto 28 % @128 KB

[1] OSU Benchmark Suite https://mvapich.cse.ohio-state.edu/benchmarks/ (BSD License).

AMD
together we advance_

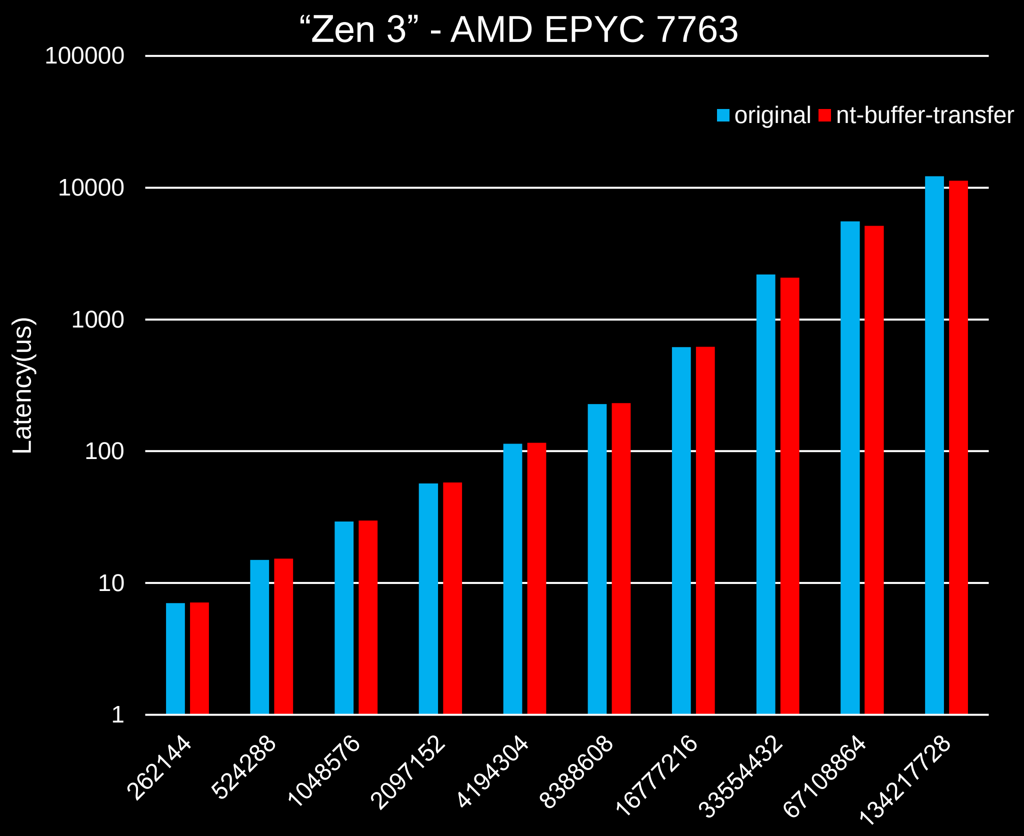
UCX Single Copy (SCOPY) Optimization

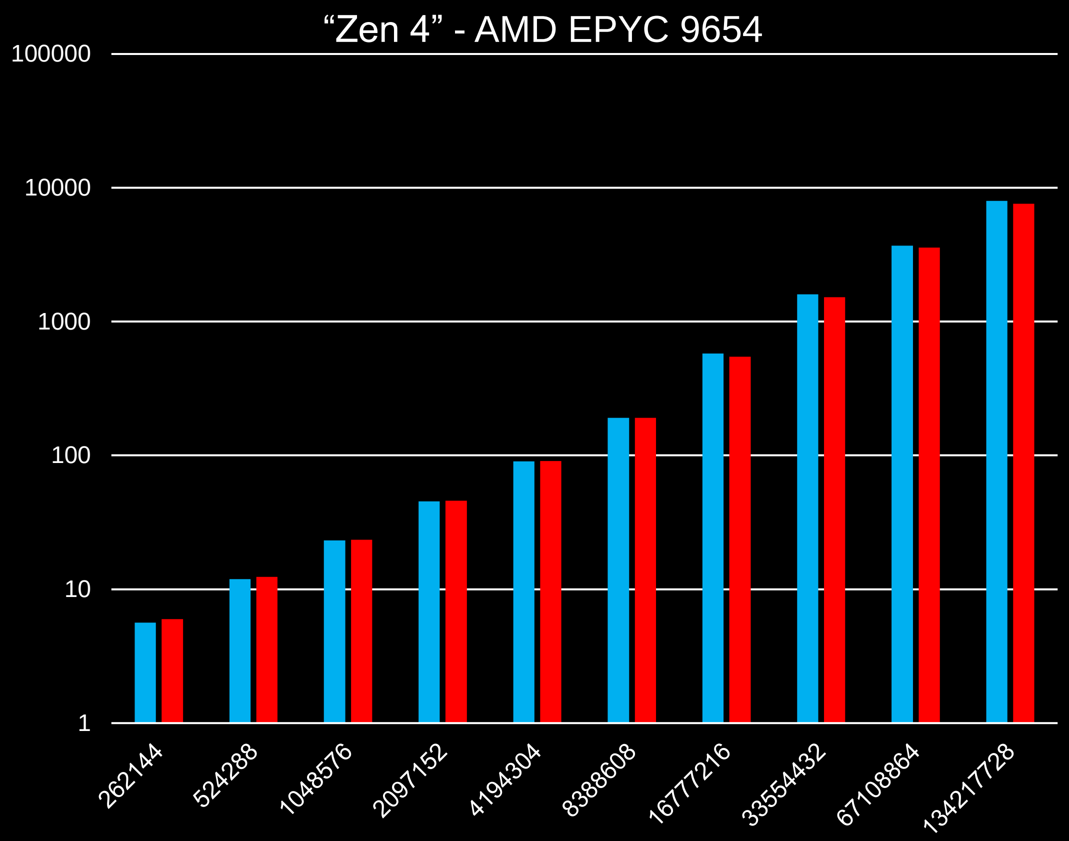
- Only xpmem is considered as it is the only single copy mechanism that uses memcpy in userspace
- The new method uses ‘loads with PREFETCH NTA’ instead of normal load instructions while copying from the sender
- It reduces cache pollution in the receiver
- The cache line bouncing effect in the sender is reduced for sizes less than L3 cache size



AMD
together we advance_

SCOPY osu_latency[1] Benchmark Results (map-by l3cache)





Gain: "Zen 3" 5-7 % for larger sizes, "Zen 4" 4-5 % for larger sizes

[1] OSU Benchmark Suite https://mvapich.cse.ohio-state.edu/benchmarks/ (BSD License).

AMD
together we advance_

DISCLAIMER

The information contained herein is for informational purposes only, and is subject to change without notice. While every precaution has been taken in the preparation of this document, it may contain technical inaccuracies, omissions and typographical errors, and AMD is under no obligation to update or otherwise correct this information. Advanced Micro Devices, Inc. makes no representations or warranties with respect to the accuracy or completeness of the contents of this document, and assumes no liability of any kind, including the implied warranties of noninfringement, merchantability or fitness for particular purposes, with respect to the operation or use of AMD hardware, software or other products described herein. No license, including implied or arising by estoppel, to any intellectual property rights is granted by this document. Terms and limitations applicable to the purchase or use of AMD’s products are as set forth in a signed agreement between the parties or in AMD's Standard Terms and Conditions of Sale. GD-18

©2023 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo and combinations thereof are trademarks of Advanced Micro Devices, Inc. Other product names used in this publication are for identification purposes only and may be trademarks of their respective companies.

AMD