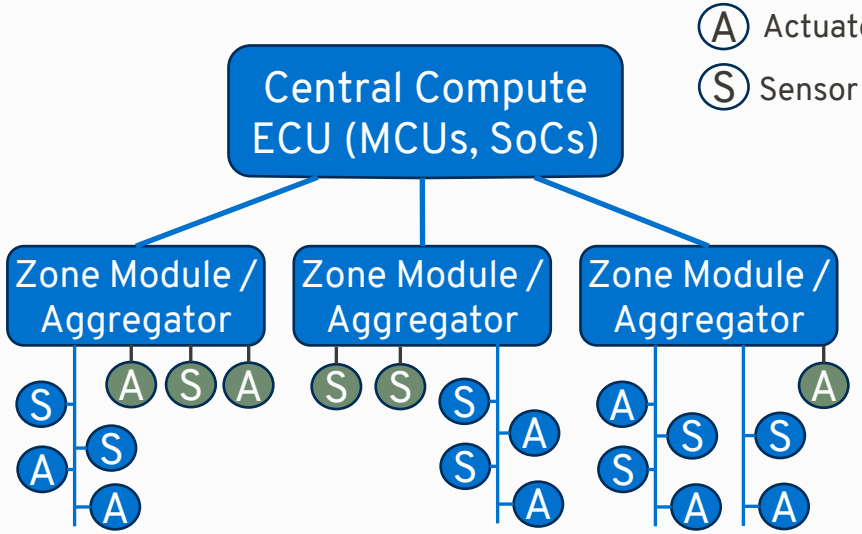
Automotive Remote Direct Memory Access (ARDMA) in a Software Defined Vehicle Architecture

Markus Jochim (presenter), Steve DiBella, Sudhakaran Maydiga, Nithya Somanath, Khaja Shazzad, Yuchen Zhou

General Motors, 10/16/2024

SDV Challenges



- **SDV's big promise:**

  Centralization simplifies updates, in-market enhancements. → Faster time to market.

- **SDV Latency challenges**
  - Centralization leads to longer paths w increased E2E latency
    Sensor S → network → Zone Module → network → Compute → network → Zone Module → network → Actuator A
  - E2E Latency contributors: Network, Functional Software, Com Stack Software
  - Task loops in Central Compute and Zone Modules drift in and out of phase (unless synced)

Low Latency Real Time Control Use Case Examples

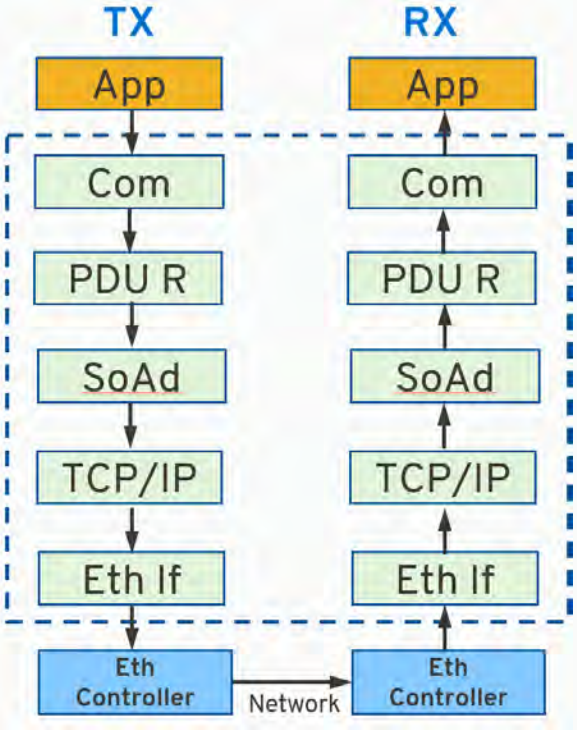
- Steer by Wire
- Anti-lock Braking
- Magnetic Ride (active suspension)

Sub-Millisecond End-to-End latency requirements *

* We are not claiming that the use cases listed necessarily have sub-millisecond latency requirements in all possible implementations. Latency requirements are implementation dependent.

Communication Stack significantly contribute to E2E Latency

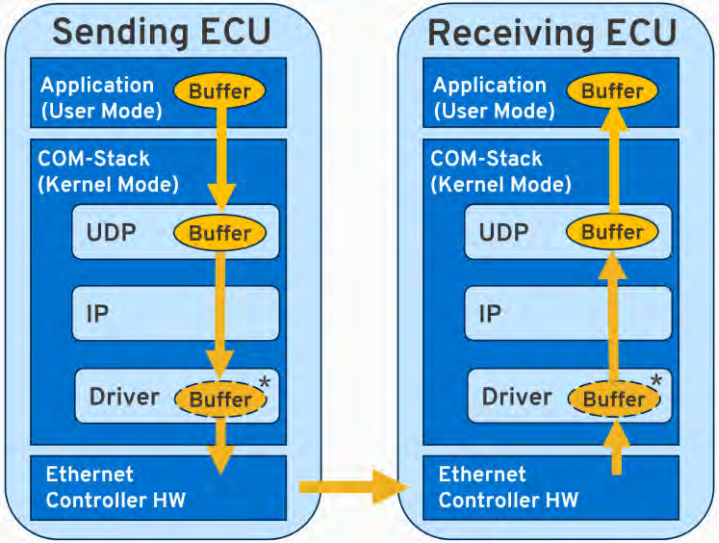
- **Example:**
  - Traditional Com Stack (not RDMA).
  - Single NXP S32K Core, AUTOSAR based, COM running with 1ms task period.
  - One 128-byte frame per loop. No functional software.



Measured Com Stack TX + RX latency contribution: ≈ 950µs

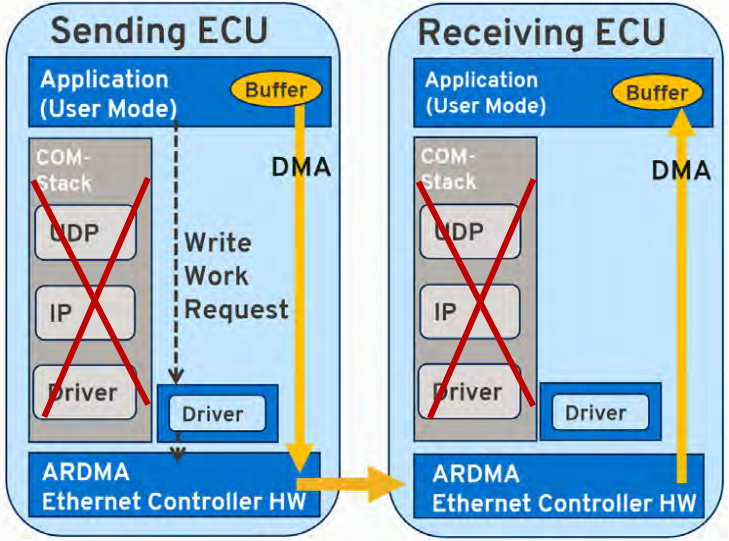
ARDMA reduces Latency and MCU Utilization

No ARDMA



- Generally, more Copy & Data Manipulations
- Context Switching
- Higher MCU Utilization

ARDMA



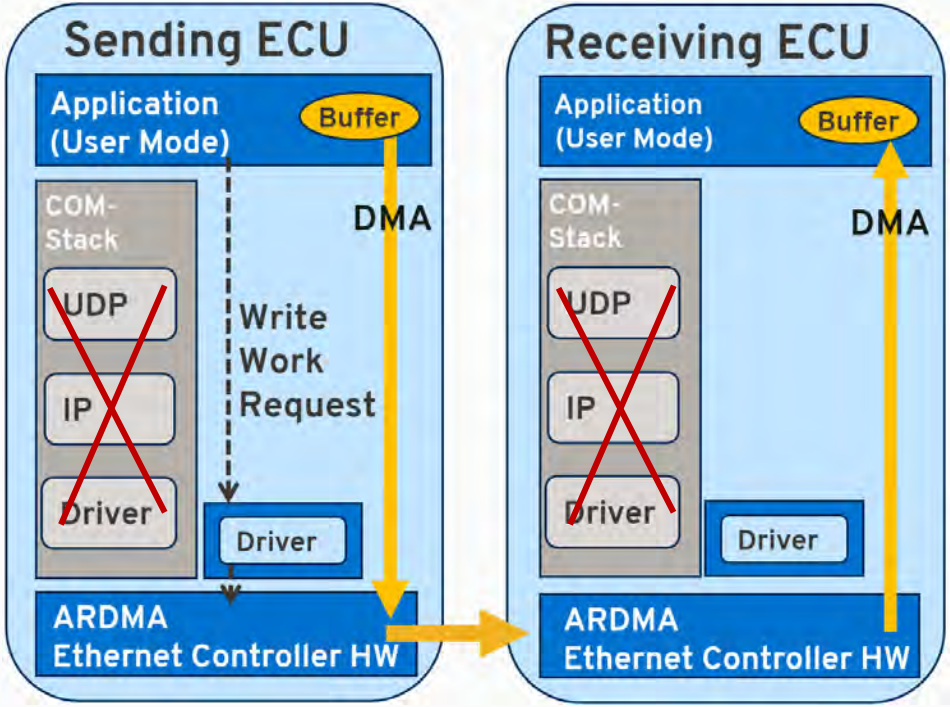
- No traditional COM Stack
- True Zero Copy
- Fewer Context Switches
- Lower MCU Utilization

RESEARCH & DEVELOPMENT

* Depending on the implementation, buffer in Driver may or may not be needed.

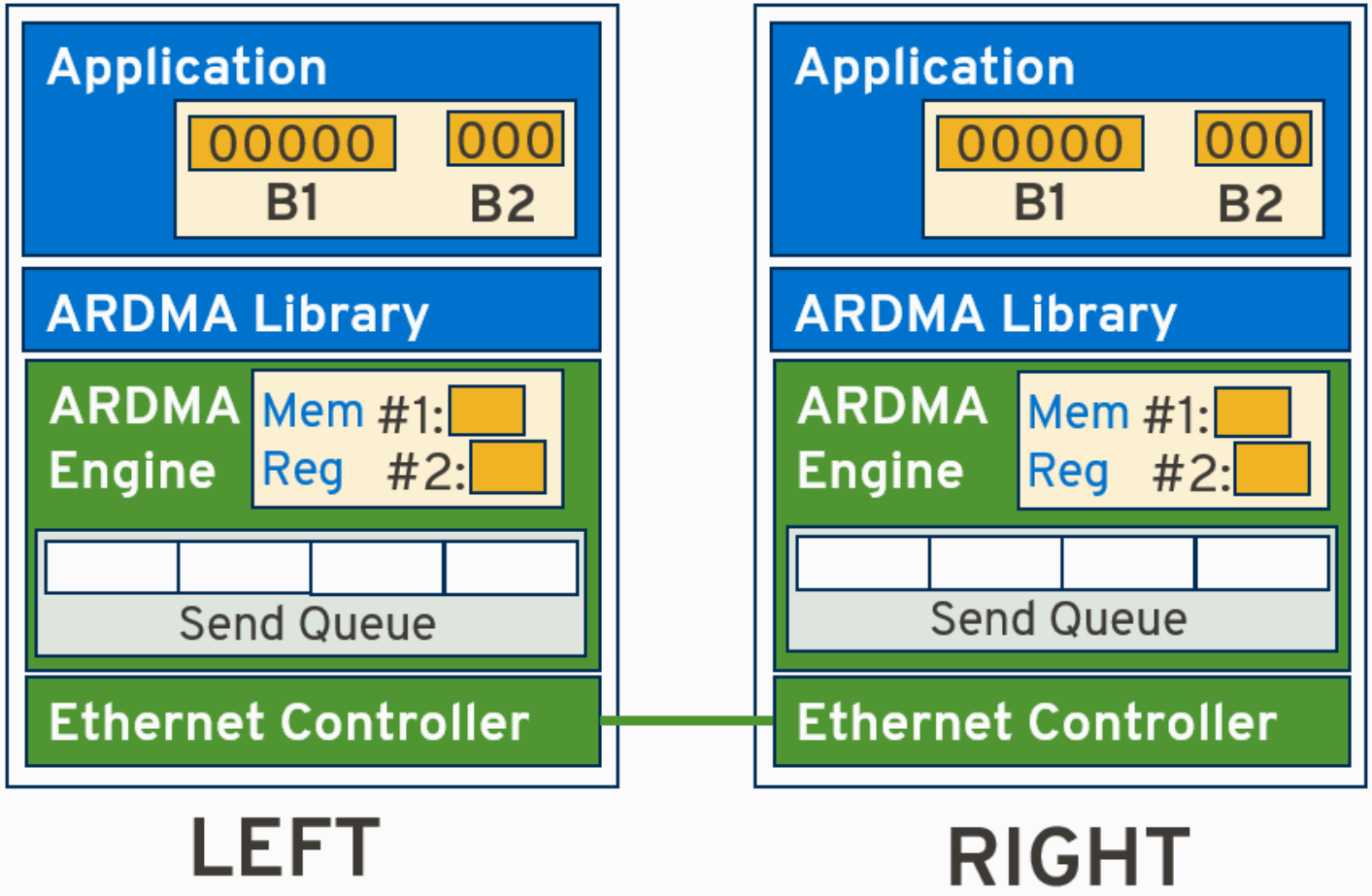
ARDMA Basic Principles

- Key Operations:
  - **Remote Write**
  - **Remote Read**
- Transport Protocol in HW



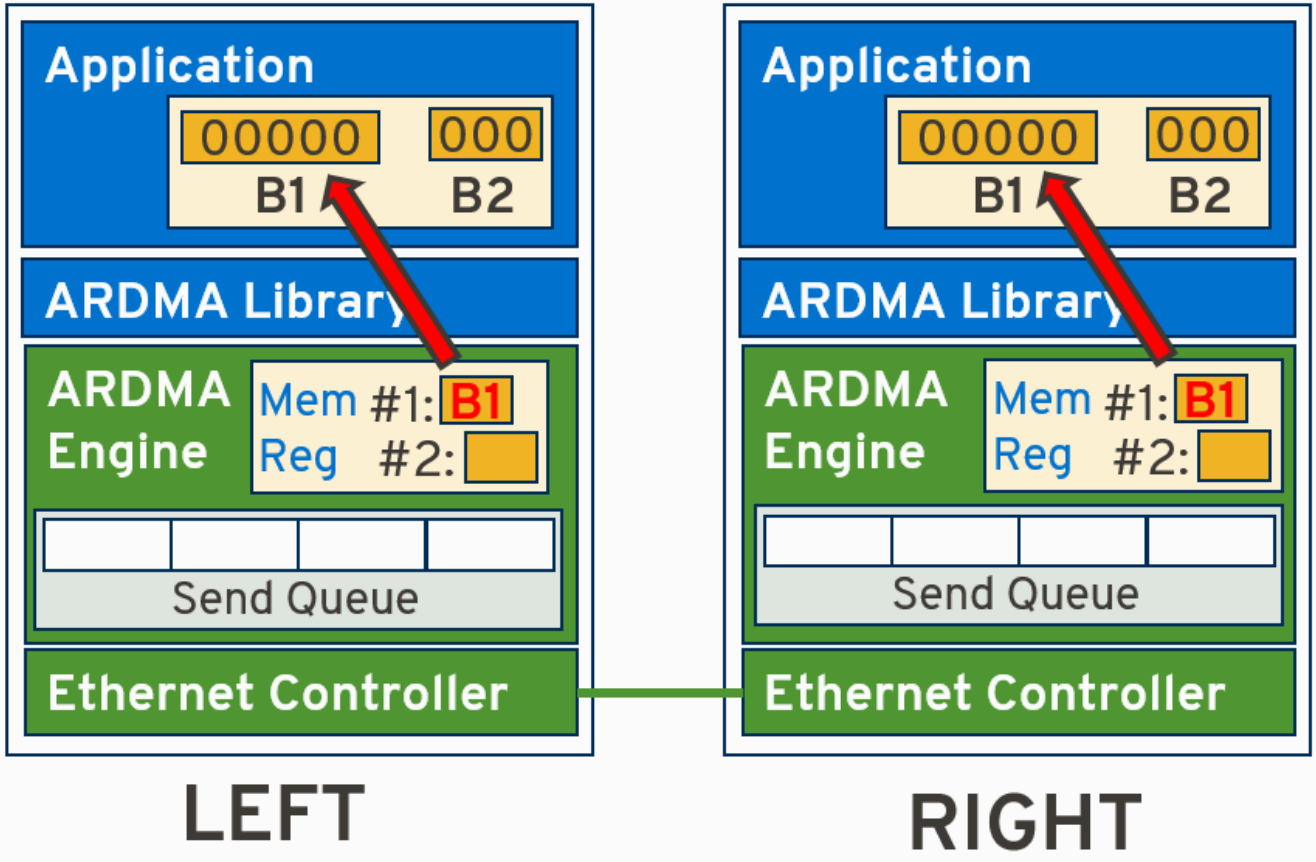
Example: ARDMA Remote Write

Simplified Example: ARDMA Remote Write (1/6)



Blue: Software Green: Hardware

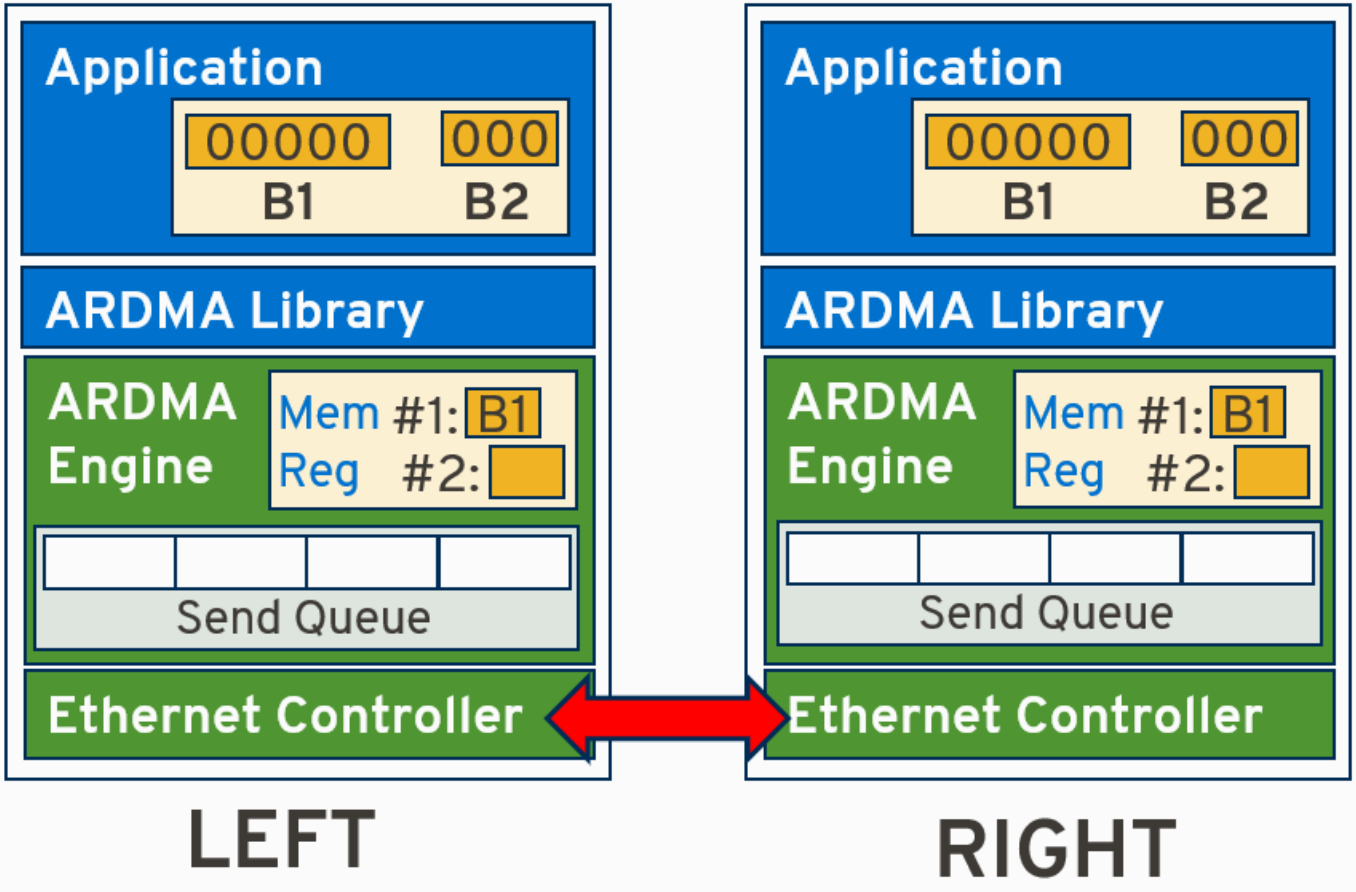
Simplified Example: ARDMA Remote Write (2/6)



Left App calls:
mem_reg(B1, #1)

Right App calls:
mem_reg(B1, #1)

Simplified Example: ARDMA Remote Write (3/6)



Left App calls:
connect_init()

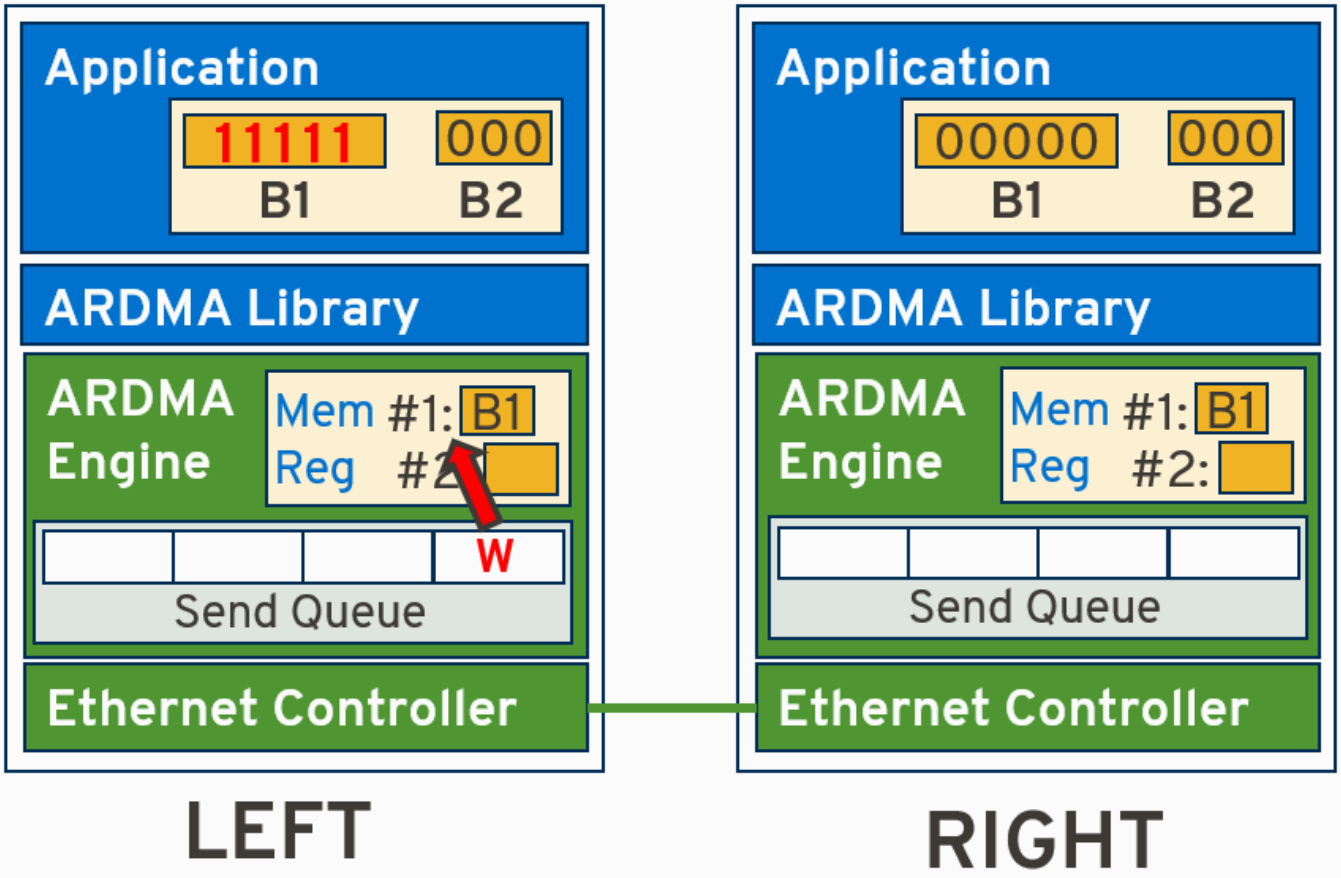
Right App calls:
connect_listen()

Engine Status:
Config -> RTSR(*1)

RESEARCH & DEVELOPMENT

(*1): RTSR = Ready to Send and Receive

Simplified Example: ARDMA Remote Write (4/6)



Left App:

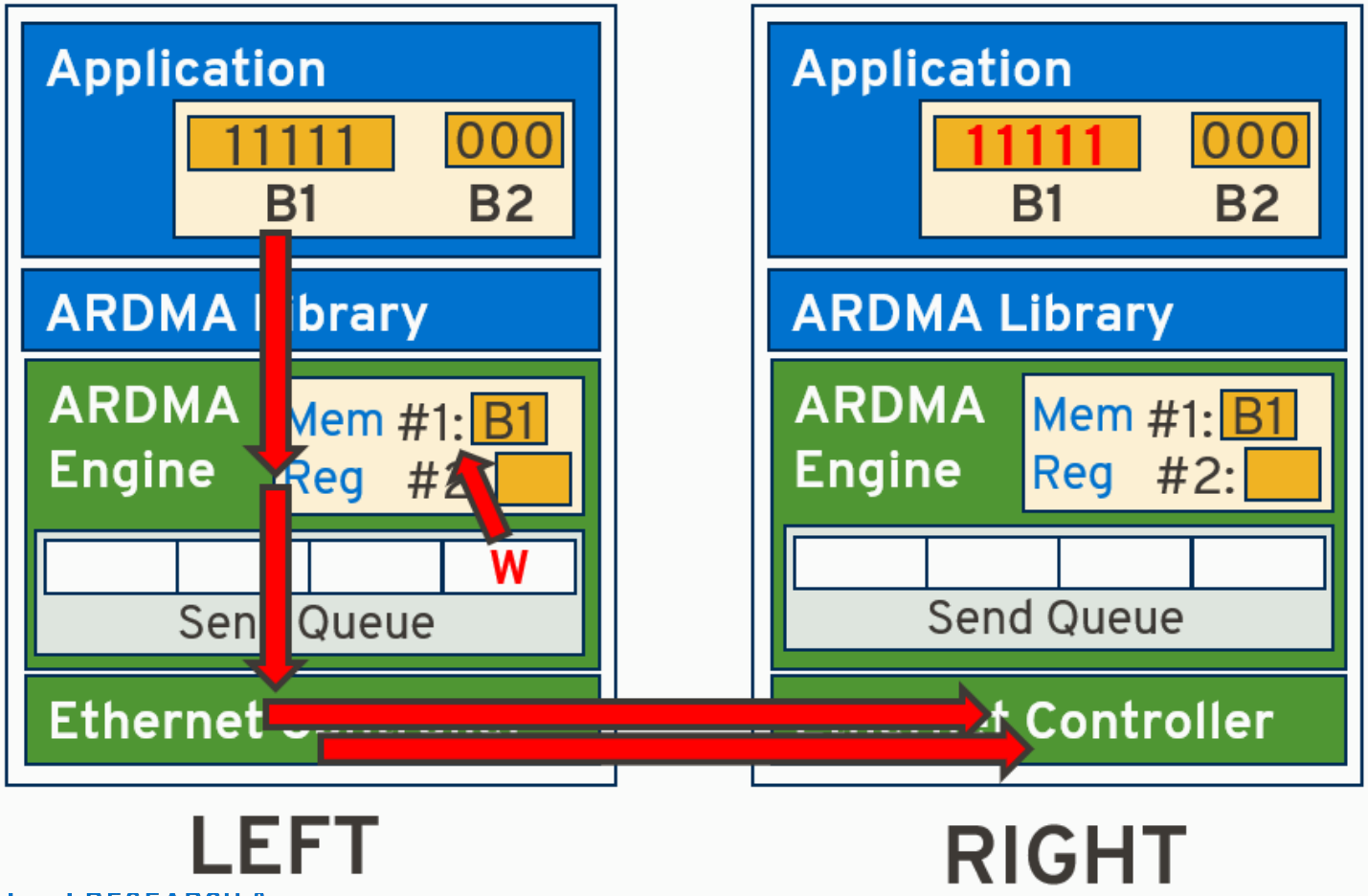
1) Writes local data:
   B1: 11111

2) Creates a write work request W (*1)

3) Calls post_send(W) (*2)

(*1): Work request W has a reference to the memory registration #1
(*2): post_send() copies W into Send Queue.

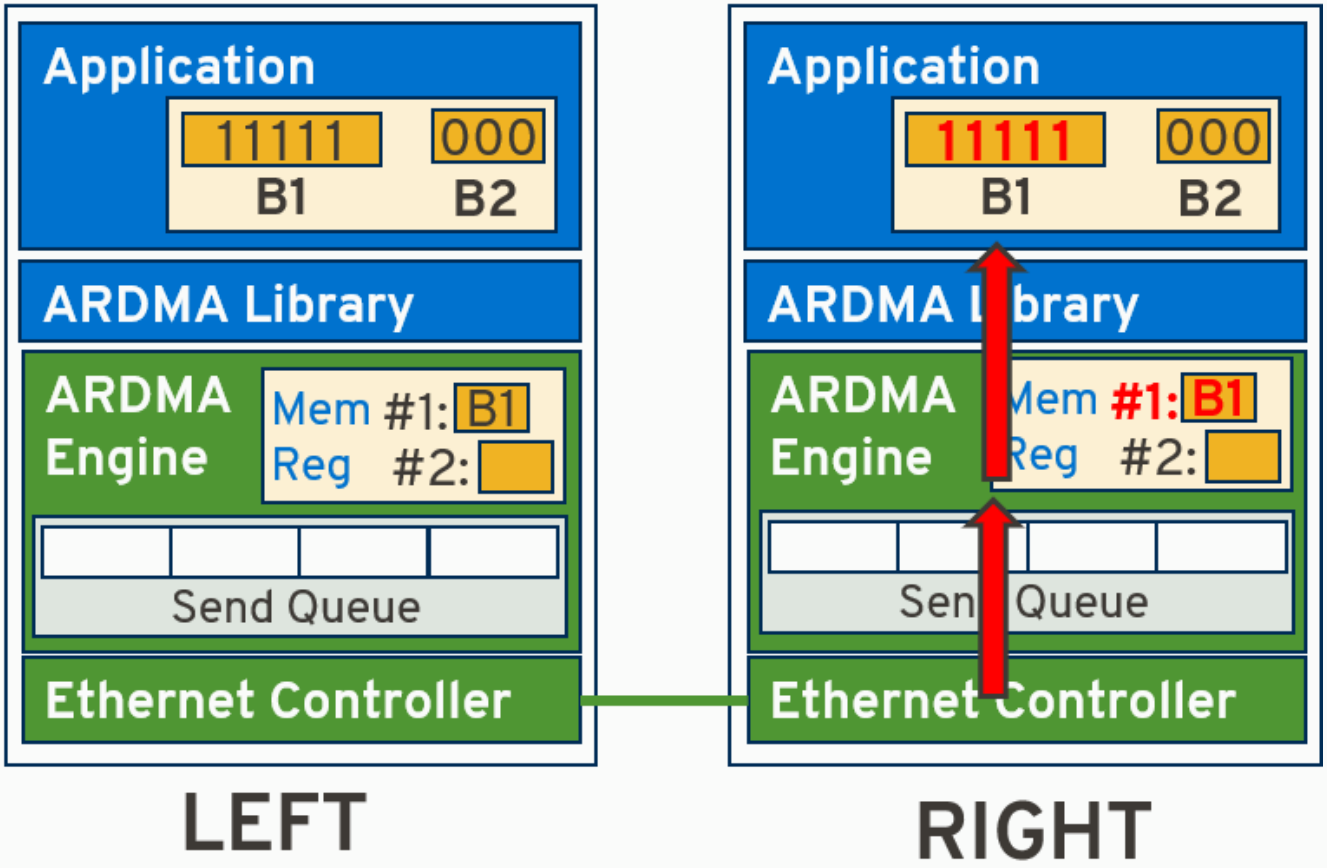
Simplified Example: ARDMA Remote Write (5/6)



Left Engine:

1) Processes W
2) DMA reads data from B1
3) Transport data + ARDMA header
4) Remove W from Queue

Simplified Example: ARDMA Remote Write (6/6)



Right Engine:

5) Transport receives B1 data + header
6) Extracts Mem Reg number (#1) from ARDMA header
7) DMA writes data to B1

Simplified example does not show ACK messages and Completion Queues that ARDMA supports.
Simplified example also doesn't show Cyber related operations, Atomic Operations, etc.

Work Request ACK messages

- Optional:
  ACK for successful read/write work requests
- **ACK is part of the hardware transport protocol**
  - Very efficient!
  - Software Com Stack ACKs are **not** efficient
    Eth Controller receives message -> Software Stack receives message -> Software Stack sends ACK message -> Eth Controller sends ACK

From Data Center RDMA to Automotive RDMA (ARDMA) (1/2)

- **Formal ARDMA spec written from scratch.**
- **Spec Version 0.5**
  - Purpose: Demo latency & MCU load reductions
  - Two Implementations: i) FPGA ii) Software (*1)
  - Don't copy a word! Don't copy a line of code!
- **Lightweight spec for embedded platforms & control**
  V0.5: 16 functions, ≈60 pages, Spec. 100% done (*2)

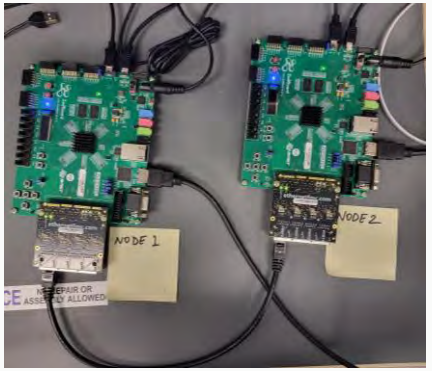
(*2) Data Center RDMA Spec "InfiniBand Architecture Specification Vol. 1" > 2,000 pages

(*2) SoftARDMA implements ARDMA Engine in software.

From Data Center RDMA to Automotive RDMA (ARDMA) (2/2)

- **Spec Version 1.0**
  - Address automotive requirements not covered by Data Center RDMA
  - Exploit optimizations possible for automotive control use cases
  - 50% done. Expecting about 100 pages total.

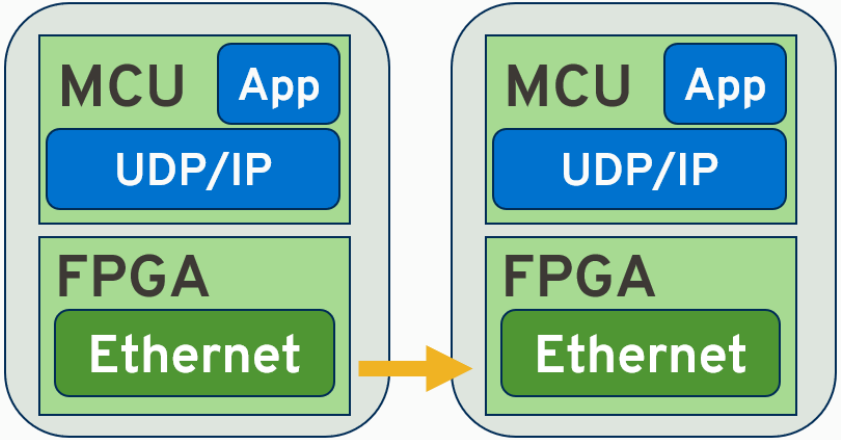
- **Some Examples**
  - Automotive cyber security
  - Atomic operations for control
  - Low Latency notifications
  - Periodic work requests



ZedBoard: Xilinx Zynq-7000
(ARM Cortex-A9 & FPGA)

Latency Measurement Setup (1/2)

UDP Send/Receive

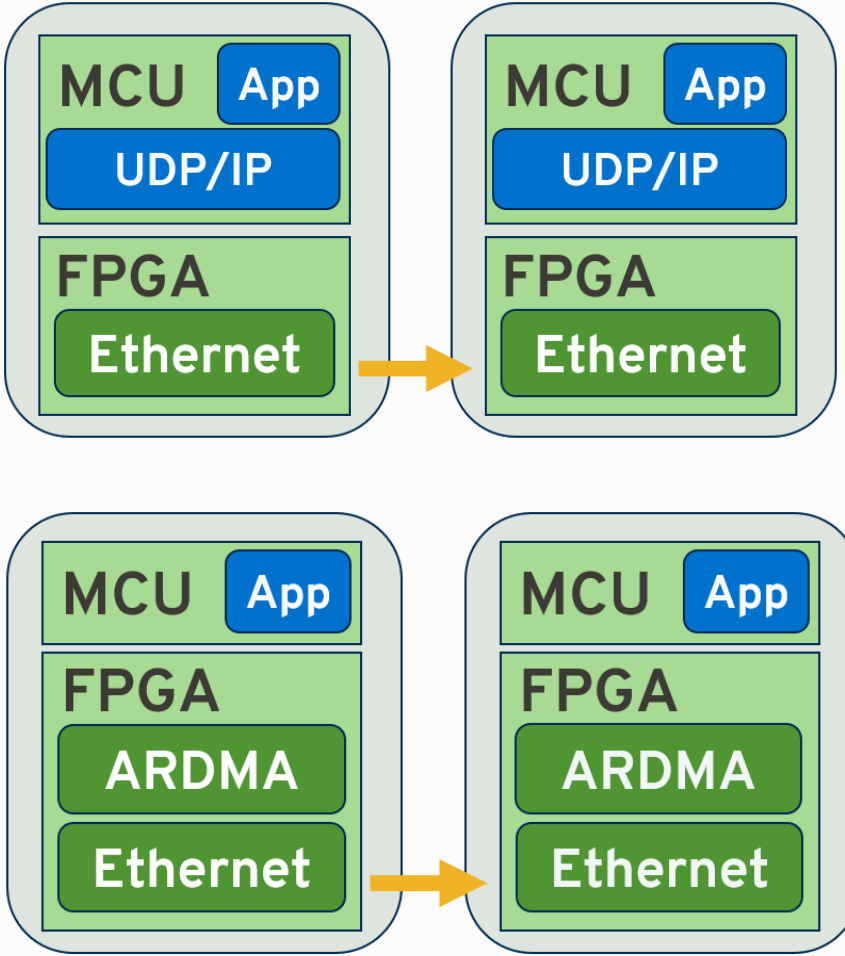


Latency Measurement Setup (2/2)

UDP
Send/Receive

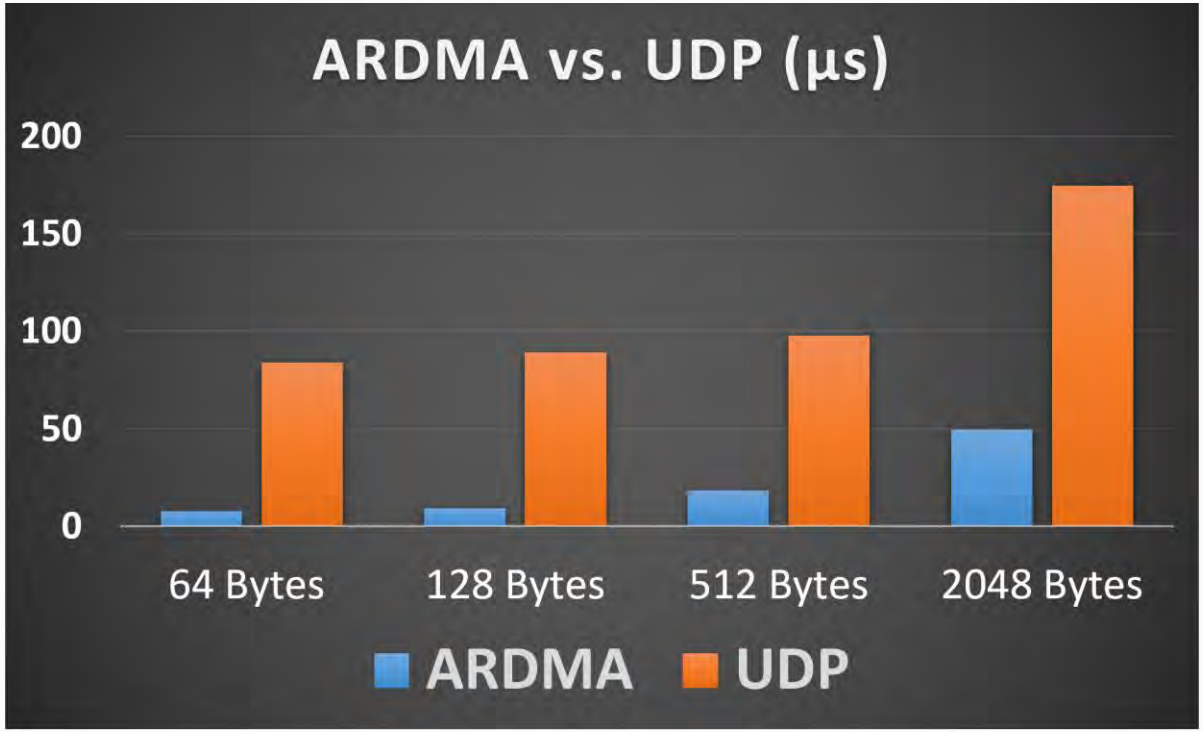
versus

ARDMA
Remote Write



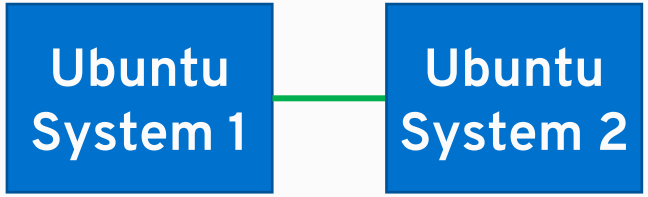
E2E Latency of ARDMA vs UDP

- Cycle accurate Verilog simulation of ARDMA FPGA implementation
- Measurements with different payloads on a ‘clean system’

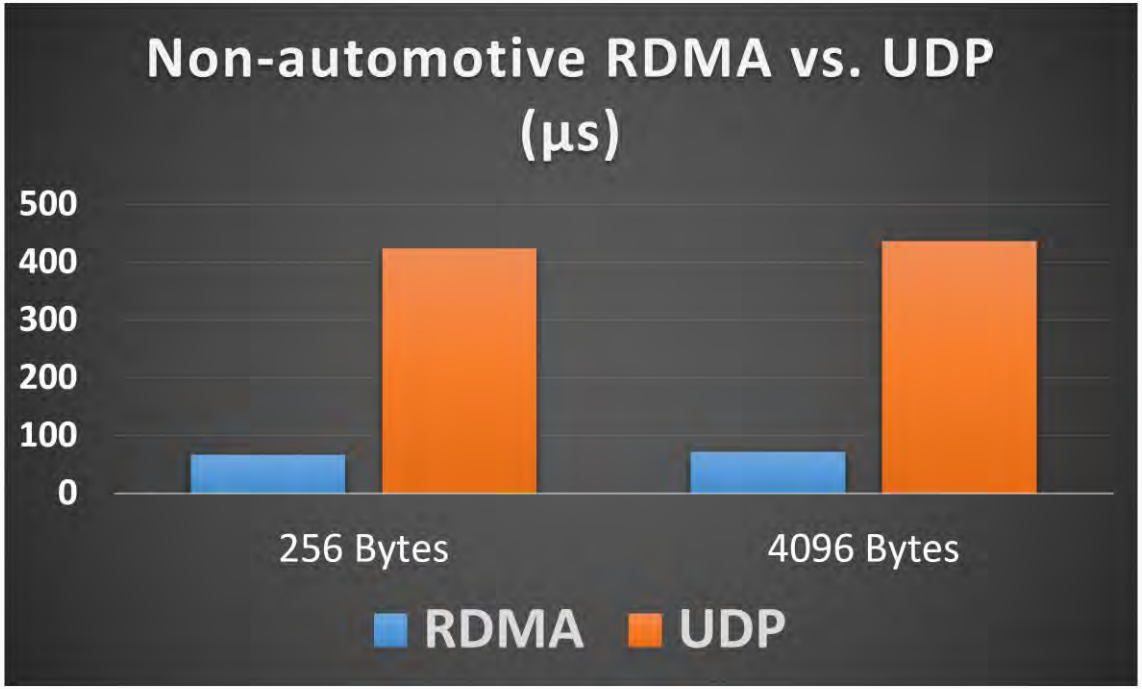


- Simulation doesn’t account for DRAM load created by other MCU tasks
- Future work: - Transition from Verilog simulation to FPGA hardware.
  - Compare performance in presence of functional software (MCU load)

E2E Latency of Data Center RDMA vs. UDP



PCs with RDMA capable NICs*



- Higher latency for both RDMA and UDP compared to previous slide.
- Relative RDMA vs UDP latency is similar to relative ARDMA vs. UDP latency from previous slide.

* Ethernet NIC : Mellanox Technologies NVIDIA MCX631102AN-ADAT ConnectX-6 Lx

Collaborate to make ARDMA useful to our industry…

ARDMA will benefit from broad adaptation and support

Interested in discussion and collaboration?
Just reach out!

markus.jochim@gm.com