# An Evaluation of Using CCIX for Cache-Coherent Host-FPGA Interfacing

Sajjad Tamimi, Florian Stock, Arthur Bernhardt, Ilia Petrov, Andreas Koch

The 30th IEEE International Symposium On Field-Programmable Custom Computing Machines
FCCM 2022

May 16th, 2022

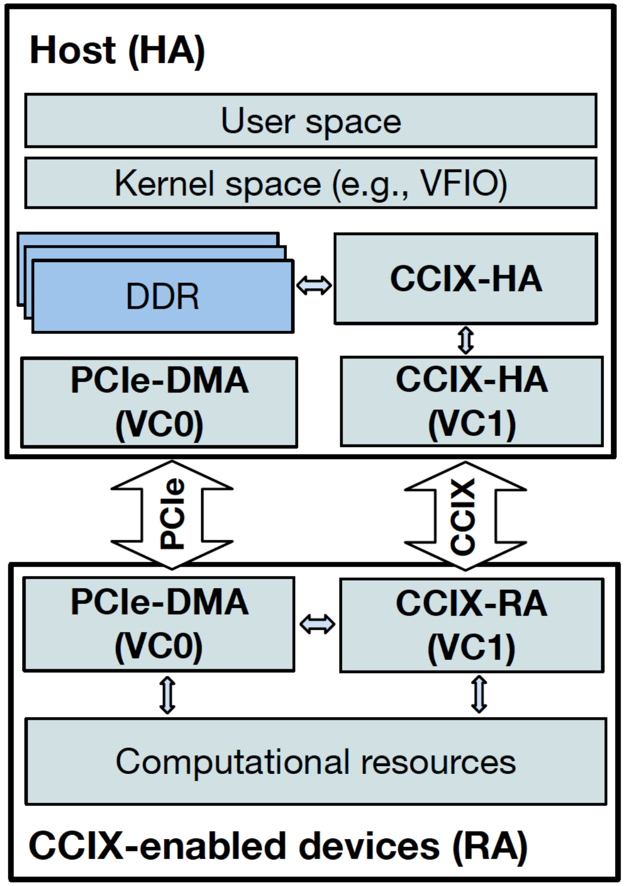
## Motivation and Goal

- PCI Express (PCIe)
  - Optimized for high-throughput bulk transfers
  - Not efficient for small transfer sizes (cache-line size)
  - No cache-coherency between host and accelerator
- Cache-coherent SVM interconnects
  - Cache-Coherent Interconnect for Accelerators (CCIX)
  - Compute Express Link (CXL)
- Examining the use of CCIX cache-coherent host-FPGA interfacing
  - Low-level measurements
  - Application-level use-case
- Database Management System (DBMS)
- Near-Data Processing (NDP)

## CCIX background

- Cache Coherent Interconnect for Accelerators (CCIX)
  - Advanced I/O interconnect (chip-to-chip interconnect)
  - Share data in a cache-coherent manner
  - Set of specifications
    - Extension to standard PCIe specification

- CCIX agent types:
  - Home Agent (HA)
  - Request Agent (RA)
- CCIX-enable devices:
  - Xilinx Alveo U280
  - Versal ACAP (VCK5000)
  - ARM Neoverse N1-SDP



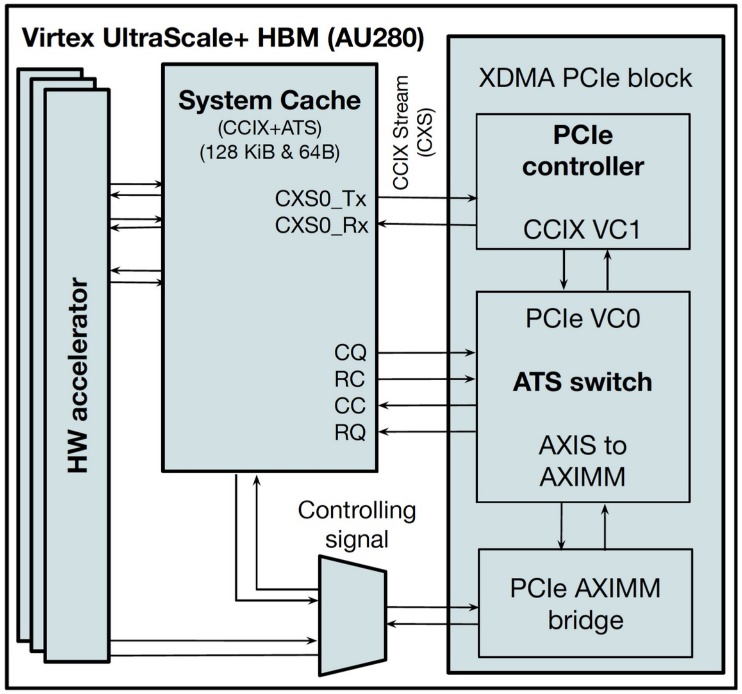
## CCIX Architecture and FPGA SoC

- System-on-Chip for request agent on
  - Virtex UltraScale+ HBM (AU280)
  - Versal ACAP (VCK5000)
- Address Translation Service (ATS)
  - Using virtual address
  - Address Translation Cache (ATC)
- CCIX timing model

$$\text{Latency} = \boxed{\begin{aligned} & P_{\text{ATC-hit}} \times \text{Latency}_{\text{ATC}} \\ & + (1 - P_{\text{ATC-hit}}) \times \text{Latency}_{\text{ATS}} \end{aligned}} \\ \boxed{\begin{aligned} & + P_{\text{cache-hit}} \times \text{Latency}_{\text{r/w-local}} \\ & + (1 - P_{\text{cache-hit}}) \times \text{Latency}_{\text{r/w-remote}} \end{aligned}}$$

$$\downarrow$$

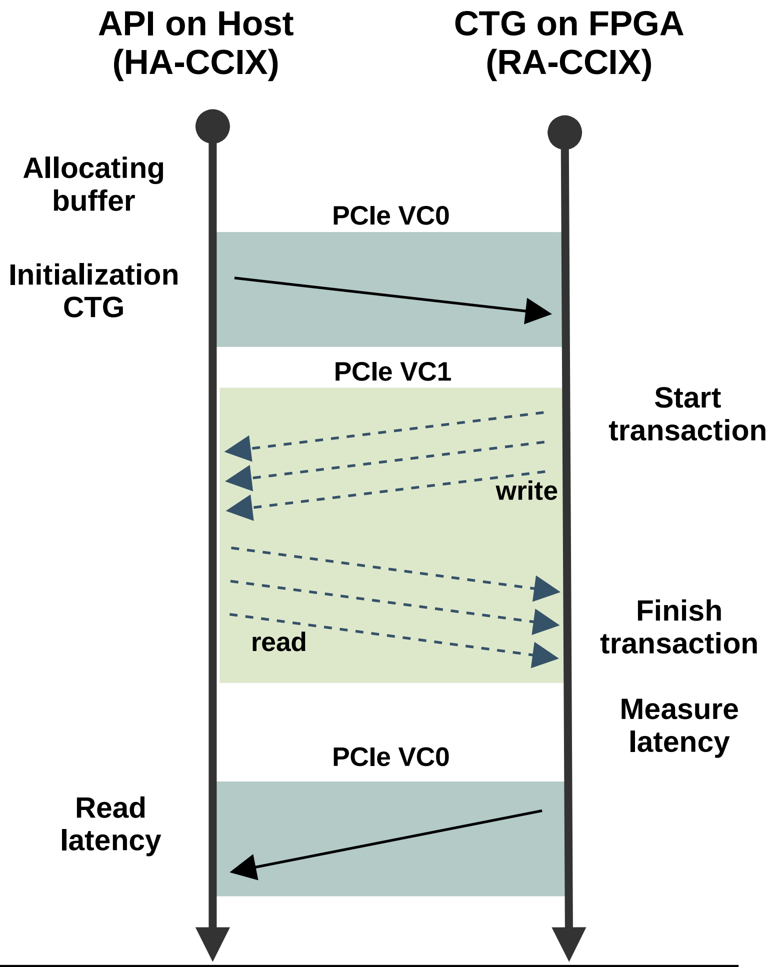
$$\text{Latency}_{\text{r/w-remote}} = \text{Latency}_{\text{CCIX}} + \text{Latency}_{\text{HA}}$$



# Measurement Setup

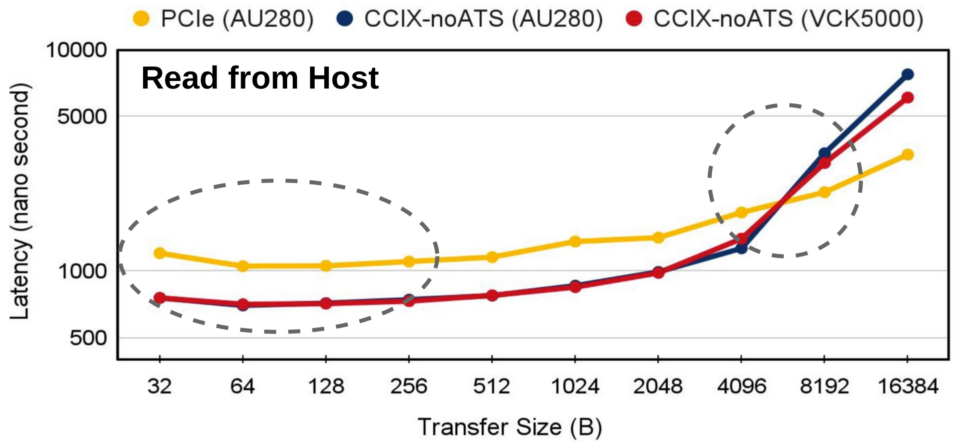
- Components
  - Software Application Programming Interface (API)
  - On-chip CCIX components
  - CCIX Traffic Generator (CTG)
  - Virtual addresses

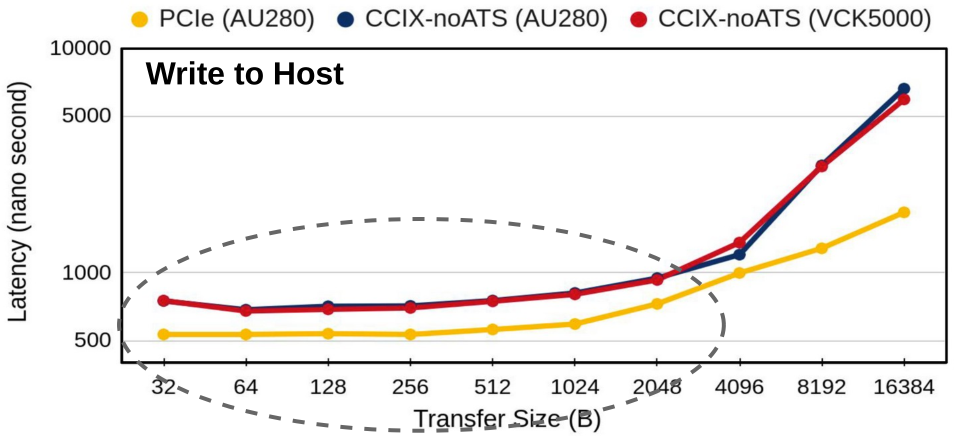
- CCIX-enabled host
  - ARM N1-SDP platform
- CCIX-enabled devices
  - Xilinx Alveo U280 (AU280)
  - Versal ACAP (VCK5000)



# Exp. 1: CCIX vs PCIe - Latency and Throughput

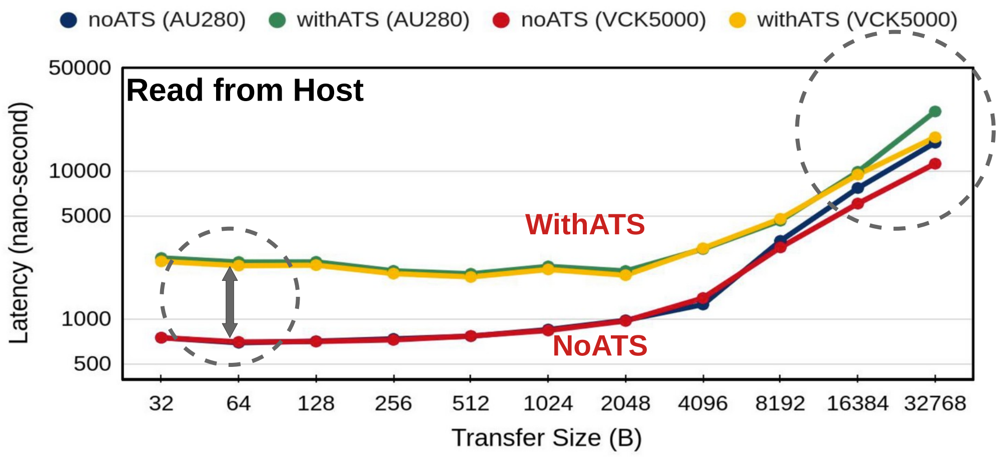
- Assumption
  - No address translation
  - PCIe traffic
    - TaPaSCo DMA-engine
    - Physical address
- Read from host
  - Better latency for less than 4KiB
  - Optimized packet protocol
- Write to host
  - Longer latency
- Read throughput (CCIX/PCIe)
  - 1KiB (3.3X)
  - 32KiB (0.87X)

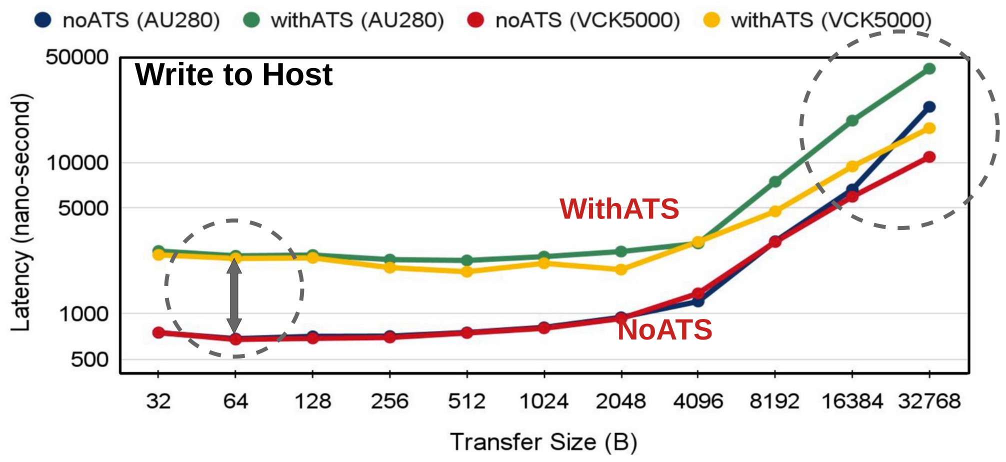




# Exp. 2: Cost of Address Translation

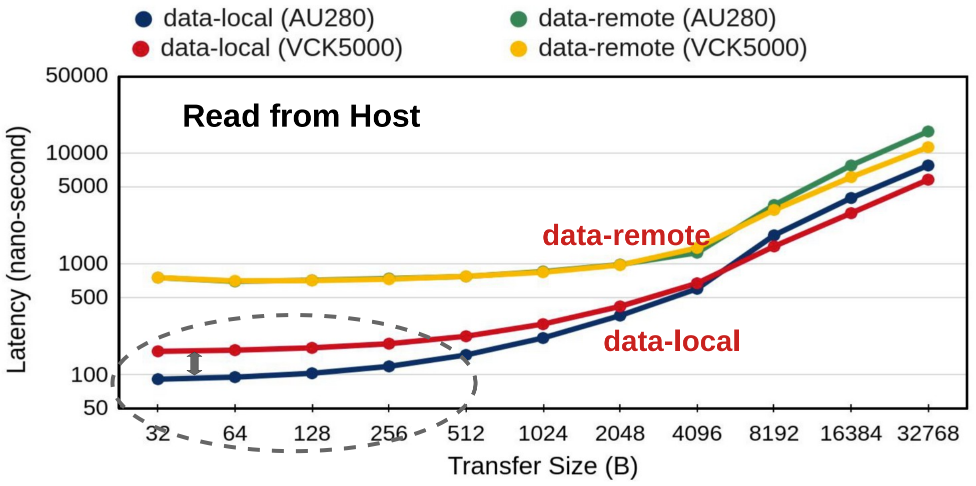
- Assumption
  - **withATS:** SC miss + ATC miss
  - **noATS:** SC miss + ATC hit
- Small transfers:
  - ATS overhead is **3X** of transfers
- Big transfers (**>32KiB**)
  - Transfer latency dominates the ATS overhead
  - Size ↑ → #ATS req. ↑
- Eliminating ATS latency
  - Linux huge page

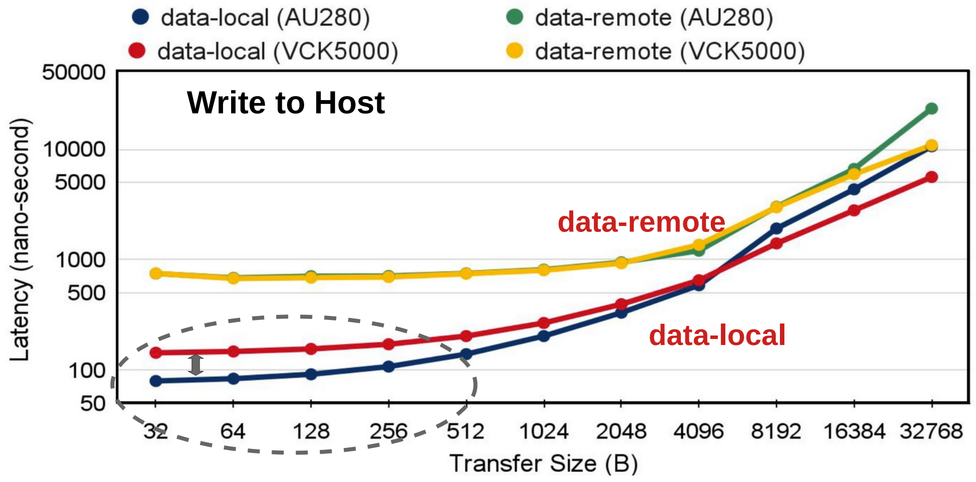




# Exp. 3: Data-Locality

- Assumption
  - **data-local:** SC hit
  - **data-remote:** SC miss
  - No address translation
- **data-remote**
  - PCIe latency
  - Host latency (HA)
- **data-local** (AU280)
  - Write: ~80ns
  - Read: ~100ns
- **data-local** (VCK5000)
  - Write: ~150ns
  - Read: ~170ns



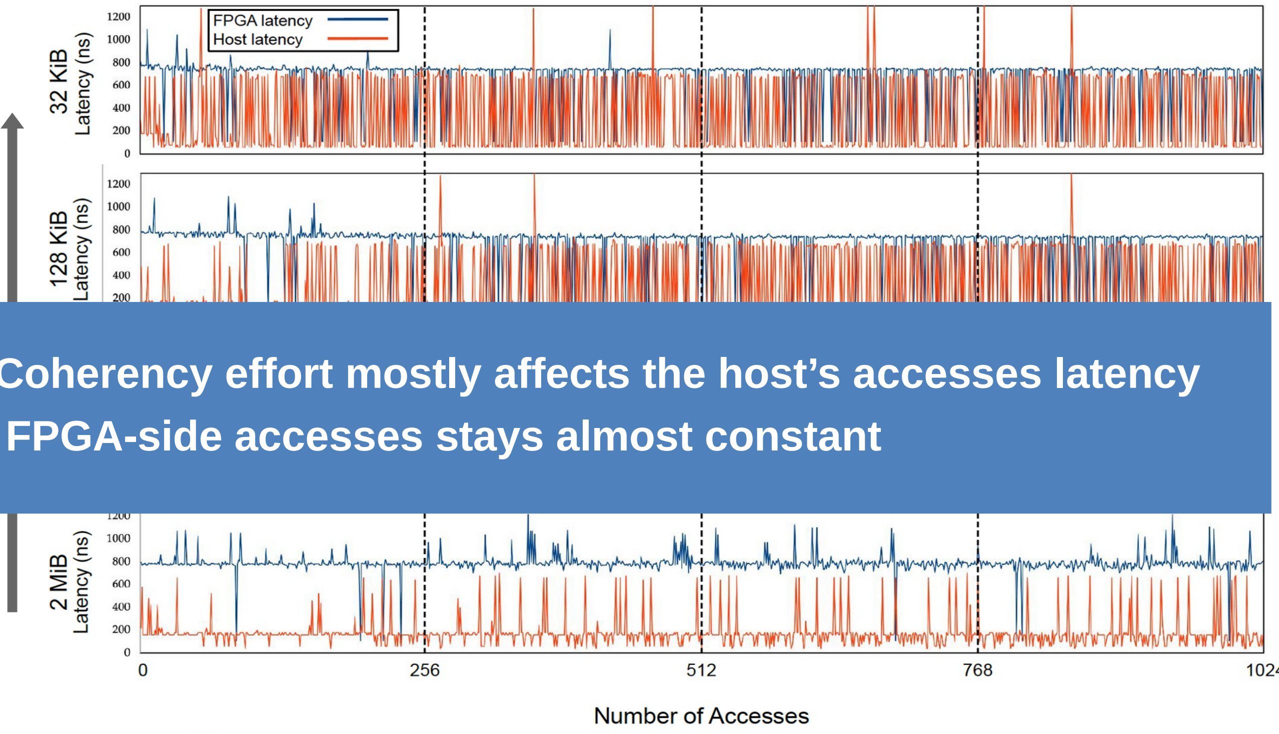


# Exp. 4: Coherency Efforts

- **Simultaneously** accessed and modified by the **host** and the **accelerator**
- Increasing **degree of contention** (with shrinking address range)

  → Increasing **efforts required to maintain coherency**

- 2X1024 random accesses
- No address translation



**Coherency effort mostly affects the host's accesses latency**

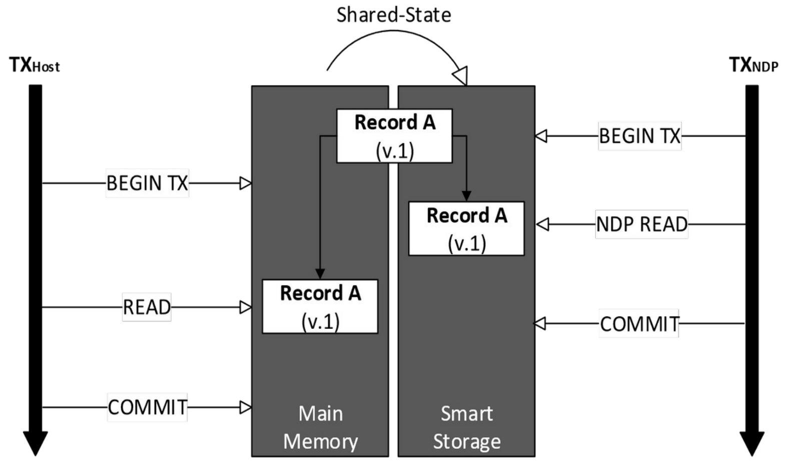
**FPGA-side accesses stays almost constant**

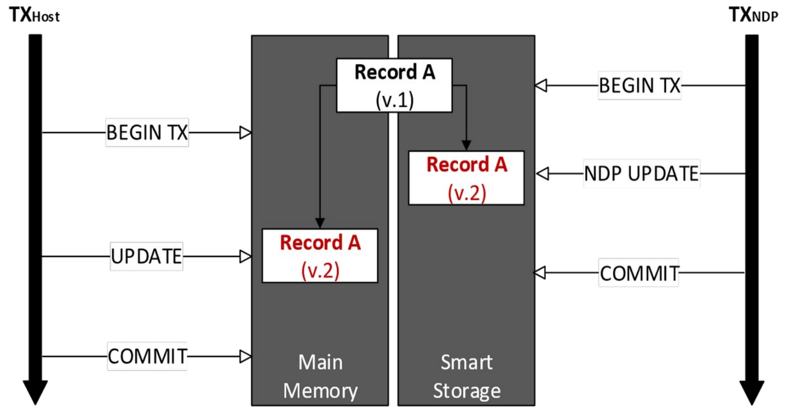
# Database Application: Motivation

- Near-Data Processing (NDP)
  - Reduce data transfer
  - Improve overall system performance

- **Read-Only NDP**
  - Intervention-free

- **Update NDP**
  - Concurrent modifications to the same record
  - Host modifies data
  - NDP modifies data in-situ
  - **write/write** conflict
  - Synchronization problems

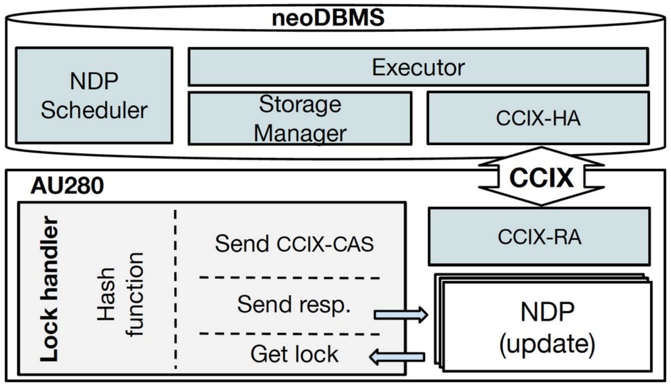


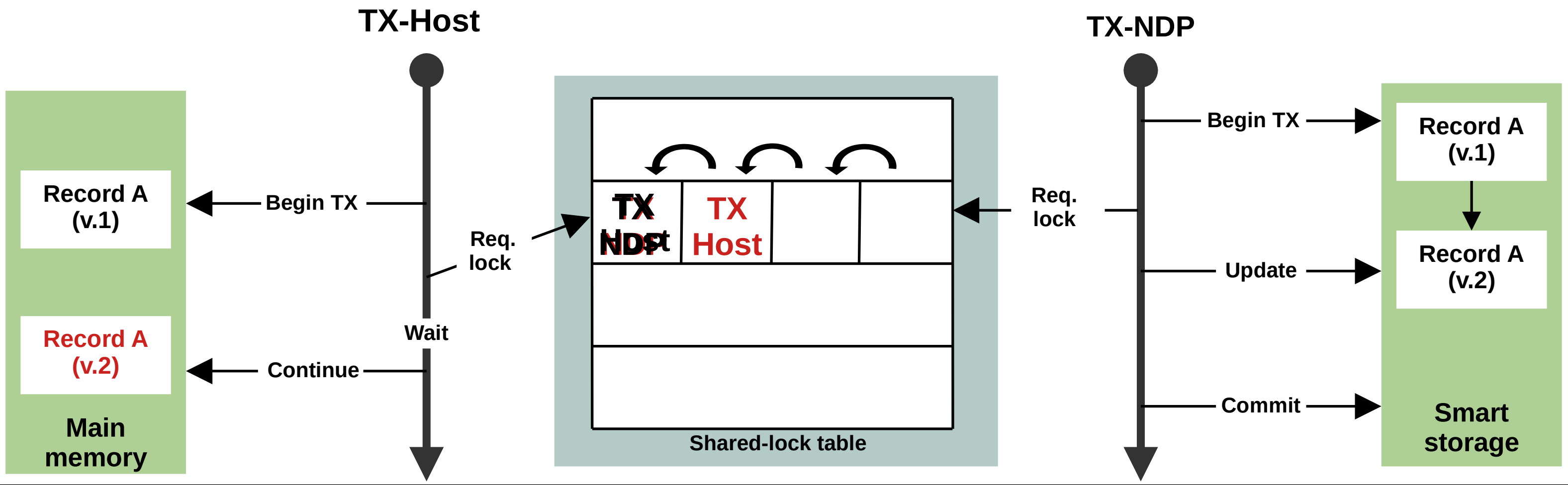


**Multiple version branches cause unresolvable inconsistencies!**

# Database Application: Proposed Approach

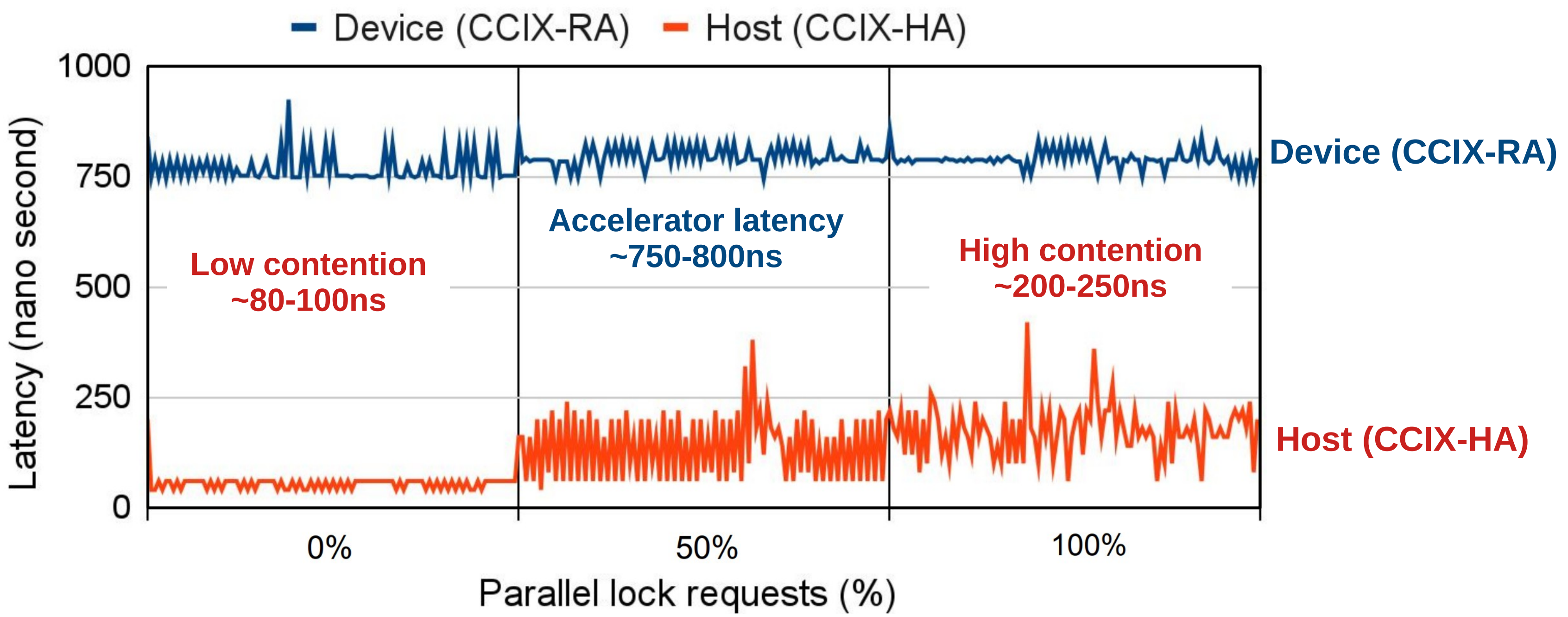
- Goal: Enabling **synchronization mechanisms** between DBMS and smart storage
- neoDBMS architecture
- Shared lock-table
  - Handle write/write conflict
  - CCIX-based solution





# Database Application: Evaluation

- Continuously creating lock requests
  - Smart storage: Alveo U280
  - NeoDBMS: N1-SDP platform



**Host and smart storage synchronization is enabled with very low overhead!**

## Thanks for your attention!