SDC19

September 23-26, 2019
Santa Clara, CA

Memory Expansion and Storage Acceleration with CCIX Technology

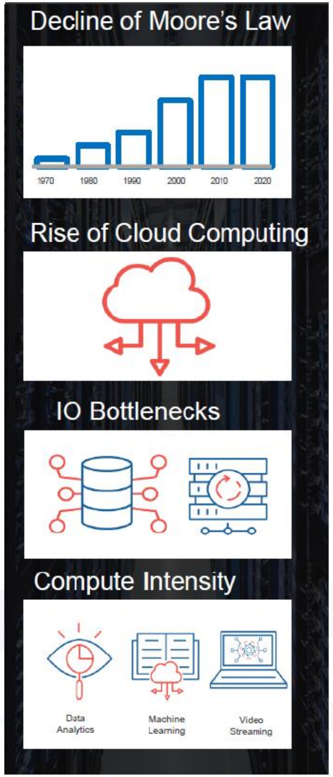
Millind Mittal
Fellow, Xilinx

Agenda

- Brief introduction to CCIX
- Memory Expansion through CCIX
- Persistent Memory support
- Storage with Compute offload
- Q&A

CCIX Context

- Slow down of performance scaling and efficient of general purpose processors
- Increasing “workload specific” computation requirements
  - Data analytics, 400G, ML, Security, compression, ……
- Lower latency requirements
  - cloud based services, IoT, 5G, …..
- Need for open standard for advancing IO Interconnect to enable seamless expansion of compute and memory resources
  - Enable accelerator SoCs to be like a NUMA sockets from Data Sharing perspective



The CCIX Consortium

- 53 Members covering all aspects of ecosystems; Servers, CPU/SoC, Accelerators, OS, IP/NoC, Switch, Memory/SCM, Test & Measurement vendors.
- Specification Status
  - Rev 1.0 - 2018
  - Rev 1.1/Rev1.2 – 2019
  - SW Guide Rev 1.0- Sept, 19
- CCIX Hosts:
  - ARM 7nm test Processor SoC providing CCIX interface (N1SDP)
  - Huawei announced Kunpeng 920
  - A 3rd party ARM SoC, Sample 12/19
- CCIX Accelerator / EP
  - Xilinx VU3xP family CCIX-enabled FPGAs silicon and Alveo boards (U50 and U280) available
  - 7nm chip Versal with CCIX support announced
- SW Enablement
  - In progress ; Key enablement to be completed Sept, 19

Promotors

AMD arm HUAWEI Mellanox TECHNOLOGIES QUALCOMM XILINX

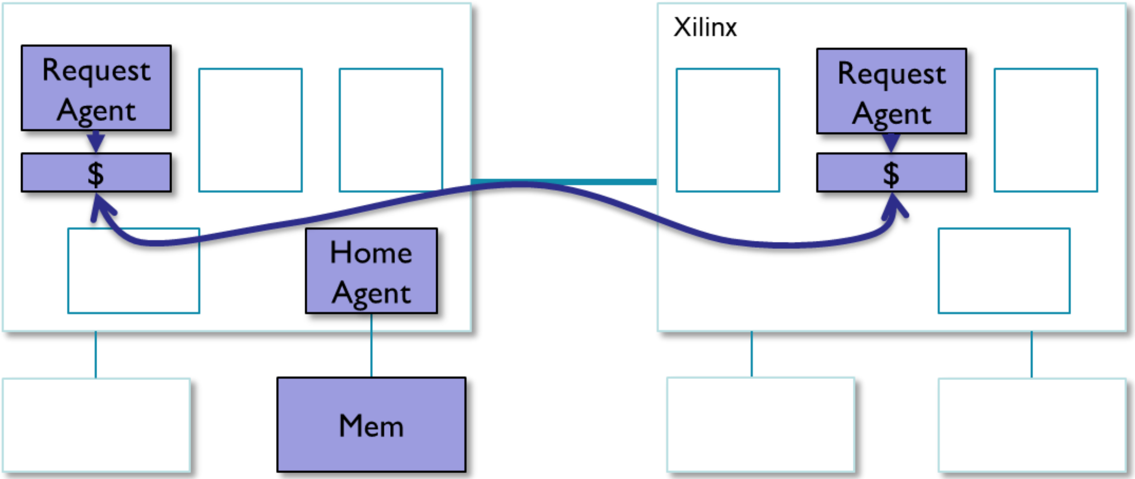
Contributors

AMPERE. Amphenol avery design systems cādence CRAY FUJITSU
GUC IBM IDT iol InterOperability Laboratory KEYSIGHT TECHNOLOGIES Lenovo
Linaro LUXSHARE ICT MARVELL Micron Microsoft
redhat SAMSUNG samtec SK hynix SYNOPSYS TE
Tektronix TELEDYNE LECROY Everywhereyoulook TOSHIBA MEMORY tsmc Western Digital.

Adopters

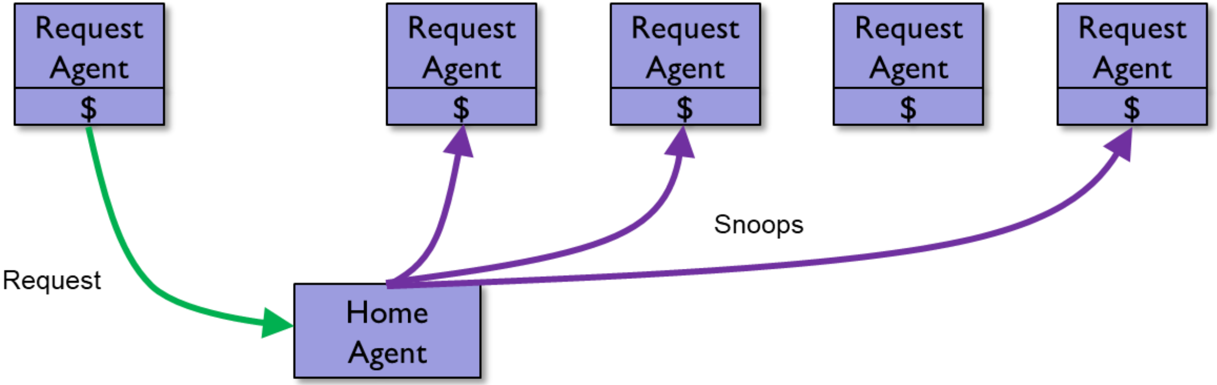
Alibaba (China) Co., Ltd. Arteris, Inc. Baikal Electronics Bitman Technologies, Inc.
Chengdu Higon Integrated Circuit Data Vortex Technologies Design Co., Ltd. Ericsson AB
Iluvatar CoreX Inc. Nanjing Netlist, Inc. Nokia Solutions and Networks Phytium Technology Co., Ltd.
Mentor. A Siemens Business PLDA SAFARI Research Group at ETH Zurich
Shanghai Zhaoxin Semiconductor Co., Ltd. SmartDV Technologies India Private Ltd.
Socionext Inc. Sony Semiconductor Solutions Corporation
Technische Universität (TU) Darmstadt Viavi Solutions, Inc. VMware, Inc.

Use of Caches for System Performance



Role of Home Agent

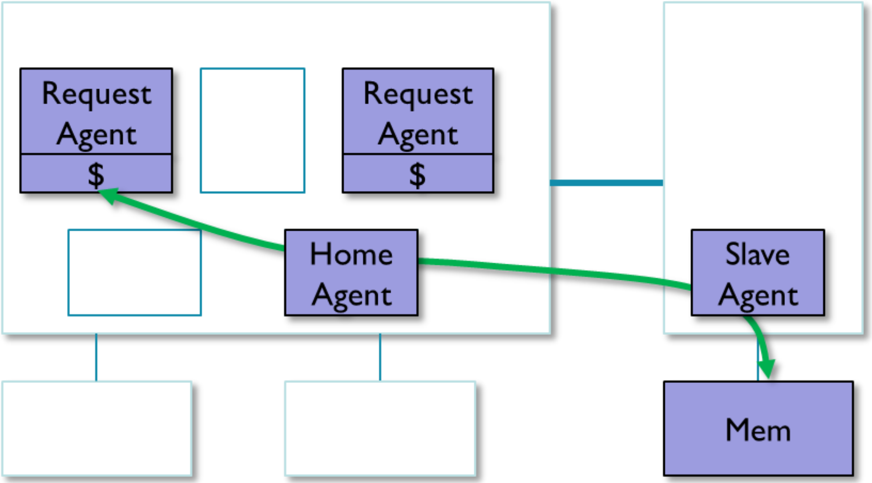
- Home agent controls the cache states when a line is accessed.
  - Ensures one Unique copy when line is being written.
  - Ensures all copies are Shared when multiple copies exist.



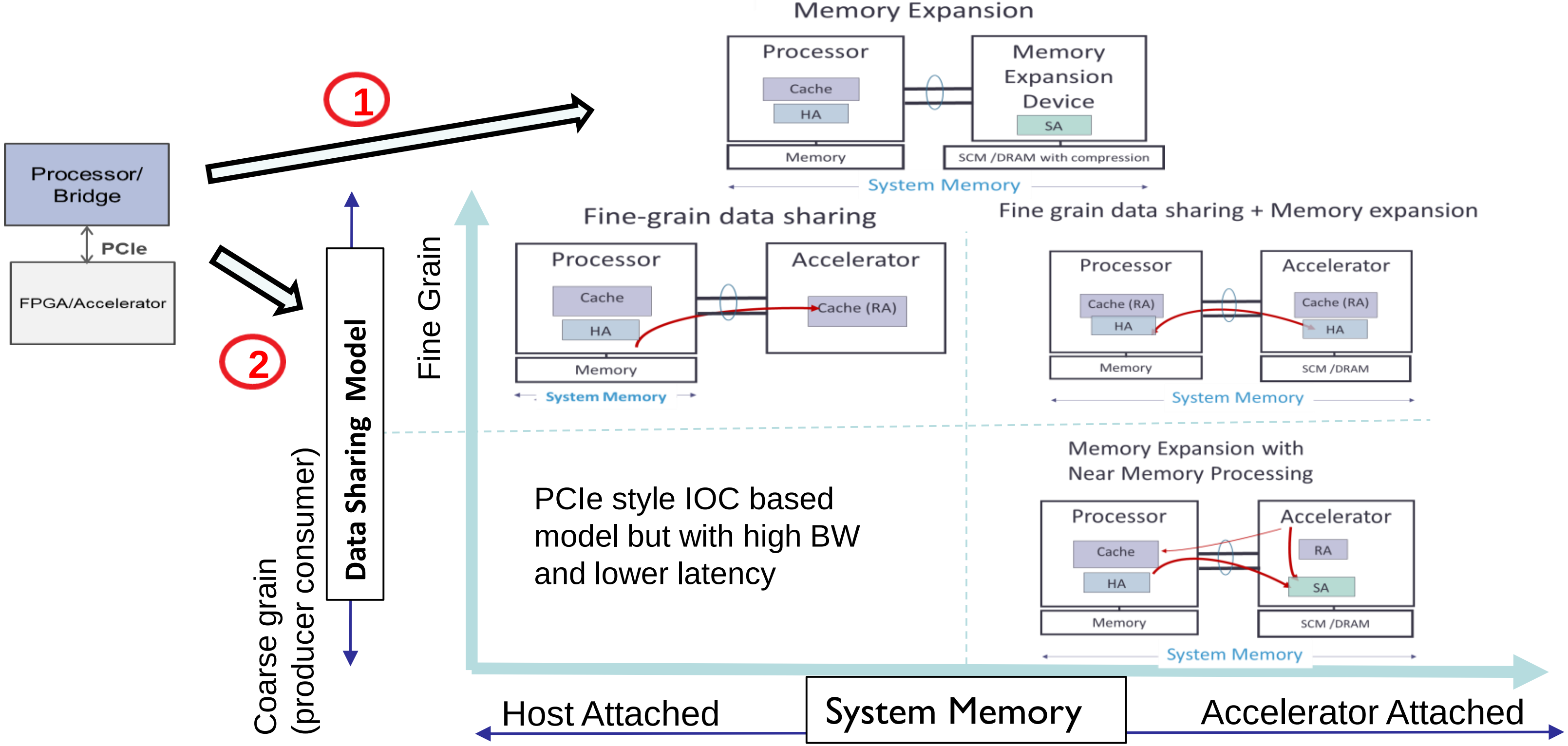
- Each location has a single Home.

Role of Slave Agent

- Slave Agent provides additional memory to a Home Agent
- Slave Agent is only protocol visible when residing on a different chip



CCIX – Open Standard Memory Expansion and Fine-Grain Data Sharing Model with Accelerators



Memory Expansion

Processor · Cache · HA · Memory

Memory Expansion Device · SA · SCM /DRAM with compression

System Memory

Processor/ Bridge

PCIe

FPGA/Accelerator

1

2

Data Sharing Model

Fine Grain

Coarse grain (producer consumer)

Fine-grain data sharing

Processor · Cache · HA · Memory

Accelerator · Cache (RA)

System Memory

Fine grain data sharing + Memory expansion

Processor · Cache (RA) · HA · Memory

Accelerator · Cache (RA) · HA · SCM /DRAM

System Memory

PCIe style IOC based model but with high BW and lower latency

Memory Expansion with Near Memory Processing

Processor · Cache · HA · Memory

Accelerator · RA · SA · SCM /DRAM

System Memory

Host Attached

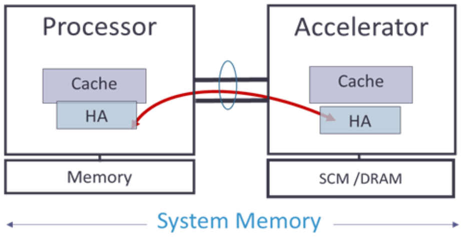
System Memory

Accelerator Attached

CCIX - Key New Attributes

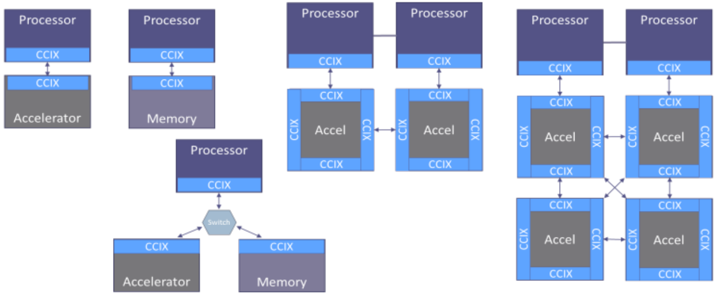
1 Memory expansion and new data sharing models including fine-grain peer processing

Memory Expansion + Fine grain data sharing



2 Flexible topologies

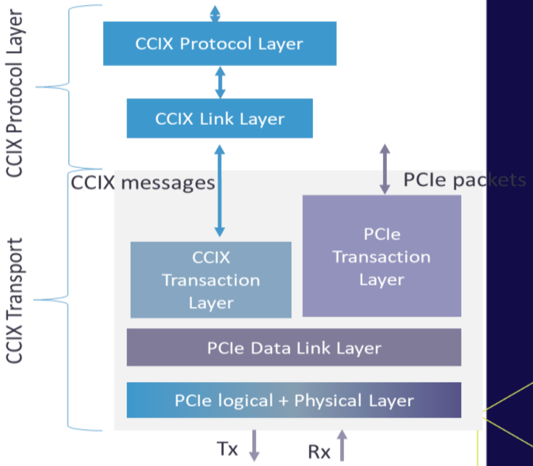
Direct attached, daisy chain, mesh and switched topologies



3 Layered architecture model

- Support different transports in future

CCIX Architecture Model

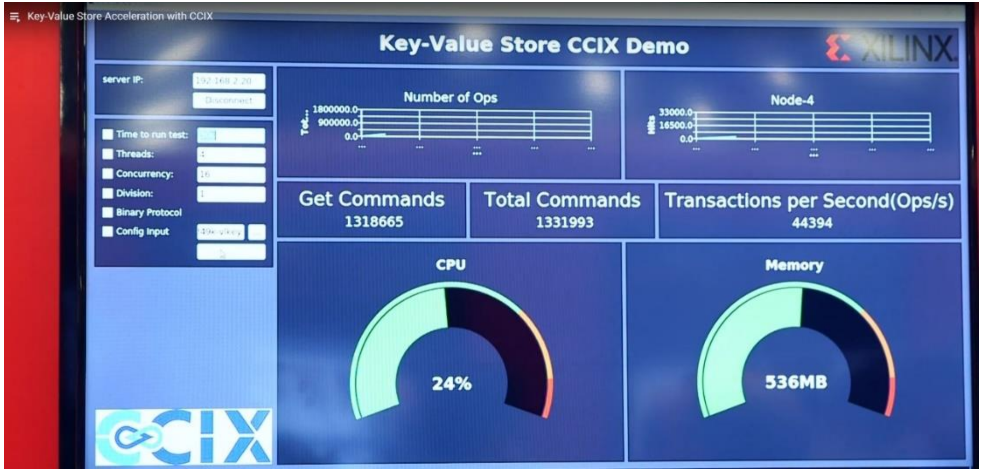


© Copyright 2019 Xilinx

Memory Expansion Through CCIX

Memory Expansion Through NUMA

- Demonstrated Extended memory through NUMA over CCIX at SC18
- KVS Database (Memcached) was enhanced to make use of NUMA expansion model over CCIX
- Key allocations are done in Host DDR, where as corresponding values were allocated on remote FPGA memory
- Expansion memory can also be a persistent memory connected over CCIX link



Key-Value Store Acceleration with CCIX

Key-Value Store CCIX Demo

server IP: 192.168.2.20

Disconnect

□ Time to run test:

□ Threads: 4

□ Concurrency: 16

□ Division: 1

□ Binary Protocol

□ Config Input 49k-vkey

Get Commands 1318665

Total Commands 1331993

Transactions per Second(Ops/s) 44394

CPU 24%

Memory 536MB

https://www.youtube.com/watch?v=drIu4vlubxE&list=PLRr5m7hDN9TLI3vuw1OqLbF7YcGi3UO9c&index=9

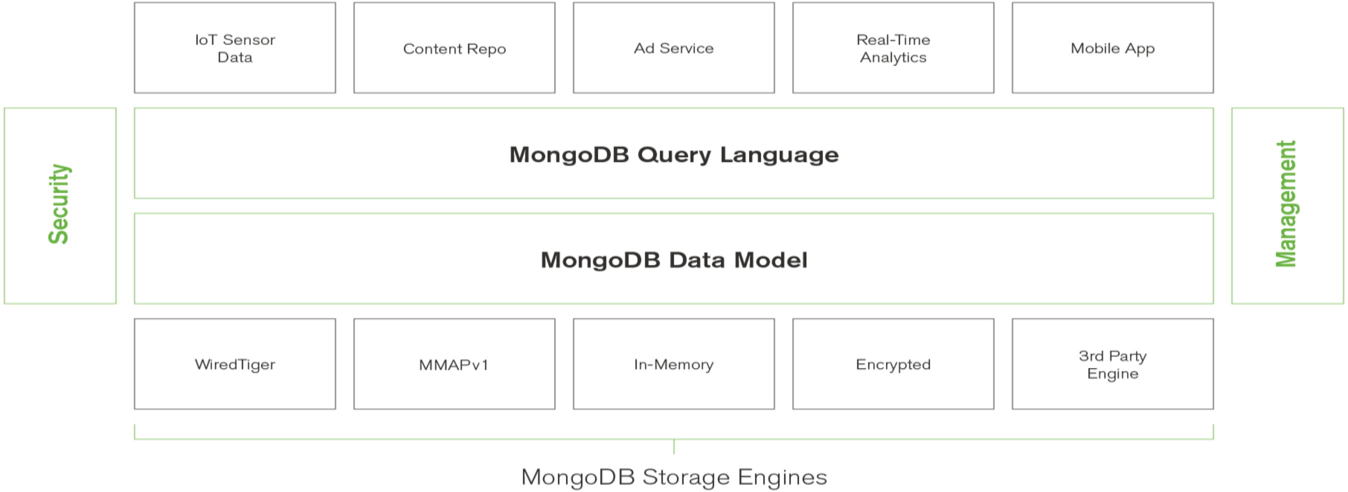
CCIX Based PMEM Solutions

- Expansion memory in CCIX can be utilized as PMEM with load store and cache coherency
- The FPGA can bring in additional benefits for storage acceleration like Compression/Encryption on expansion memory
- PMEM aware application
  - Application needs to be rewritten with PMDK API support
  - Lot of popular databases like Redis Aerospike and MongoDB already have source code with PMDK API support
- Near memory compute will benefit HPC applications
- Extend the custom distributed filesystem for zero copy network transmission and storage

Storage with Compute Offload

Storage Engine IOPs Improvement for MongoDB

- Demo planned in SC19
- MongoDB Storage Engine layer options



IoT Sensor Data

Content Repo

Ad Service

Real-Time Analytics

Mobile App

Security

MongoDB Query Language

MongoDB Data Model

Management

WiredTiger

MMAPv1

In-Memory

Encrypted

3rd Party Engine

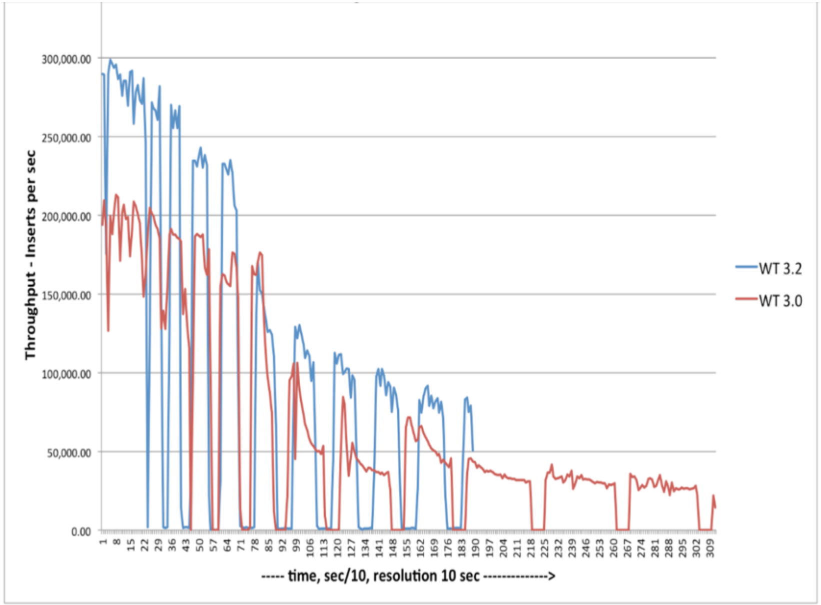
MongoDB Storage Engines

© Copyright 2019 Xilinx

Analysis and Inference

- WiredTiger is an performance, scalable, production quality, NoSQL, Open Source extensible platform for data management

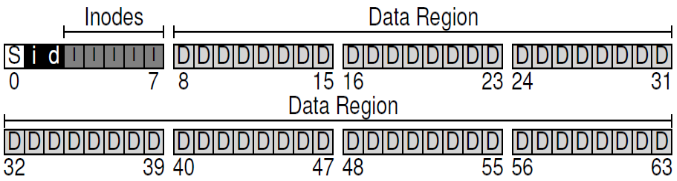
- Run two performance bench marking tests & collected call stacks
  - https://github.com/johnlpage/POCDriver
  - https://github.com/mdcallag/iibench-mongodb
- Major hot spots were identified as
  - WiredTiger IO operations (IO intense)
  - Compression (CPU intense)



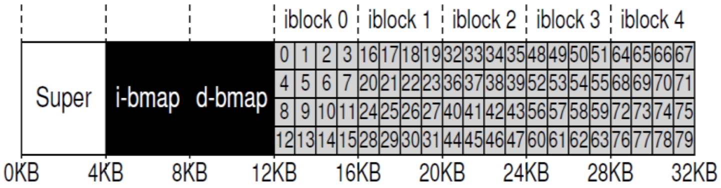
WiredTiger Storage Engine (http://source.wiredtiger.com/)

EXT4 File System Shared Data-structures

- File System Structures
  - Superblock: Describes & maintains state for FS
  - Inode: Every object managed with FS (file/directory). Includes metadata
  - Dentry: Translates between names and inodes
- File : Represents an open file

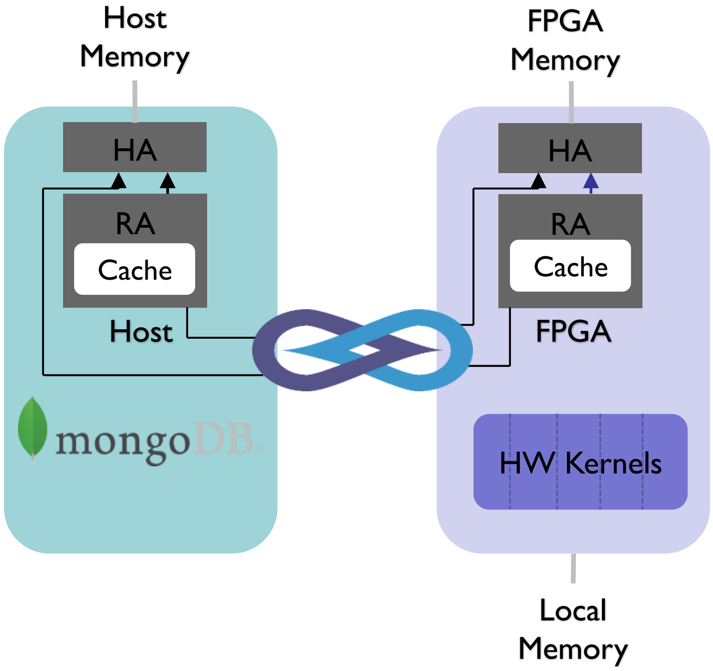


The Inode Table (Closeup)



Accelerated Design Over CCIX

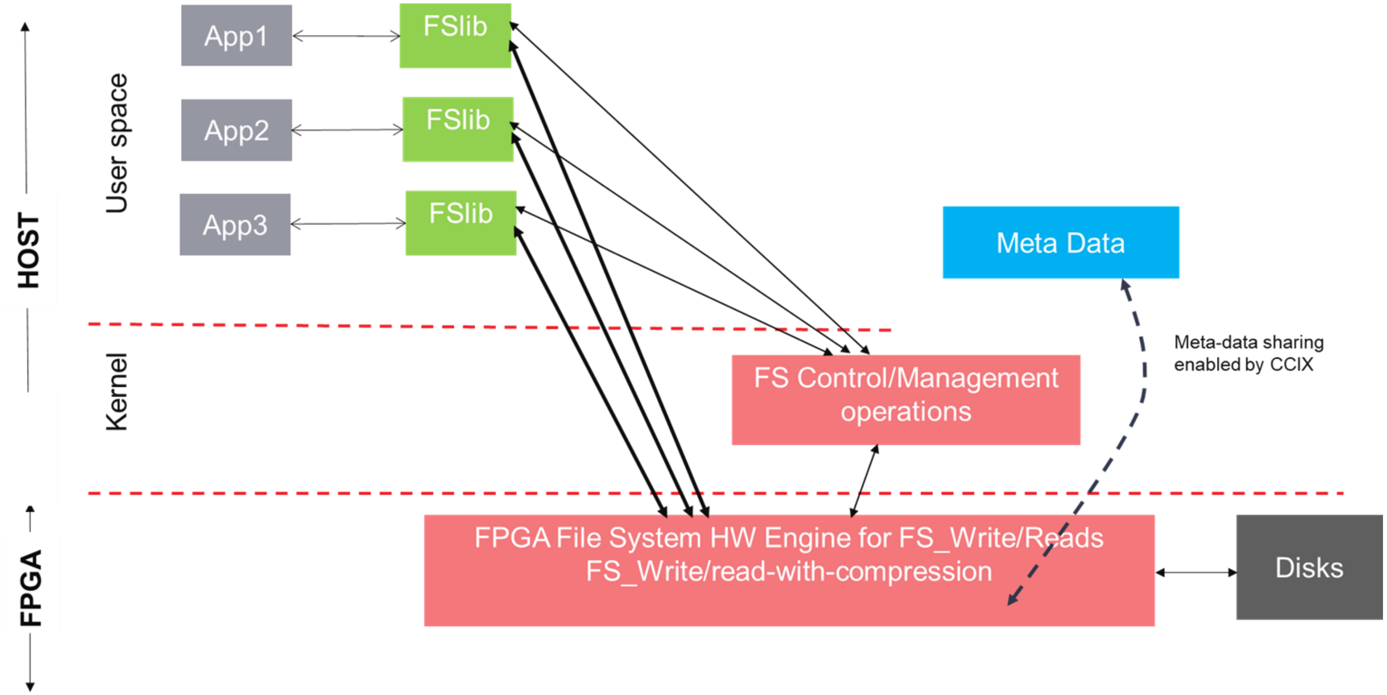
- IOPs are limited due to OS context switch and other SW overheads
- Enable user space calls to FS directly
- Offload performance critical operations (writes/reads) fully to FPGA with interface to storage
  - File system Meta data structures are maintained in shared FPGA memory
  - Actual file data is stored over FPGA connected storage class memory which is faster than SSDs
- Inline efficient Compression
- Seamless acceleration architecture through shared meta-data enabled by CCIX



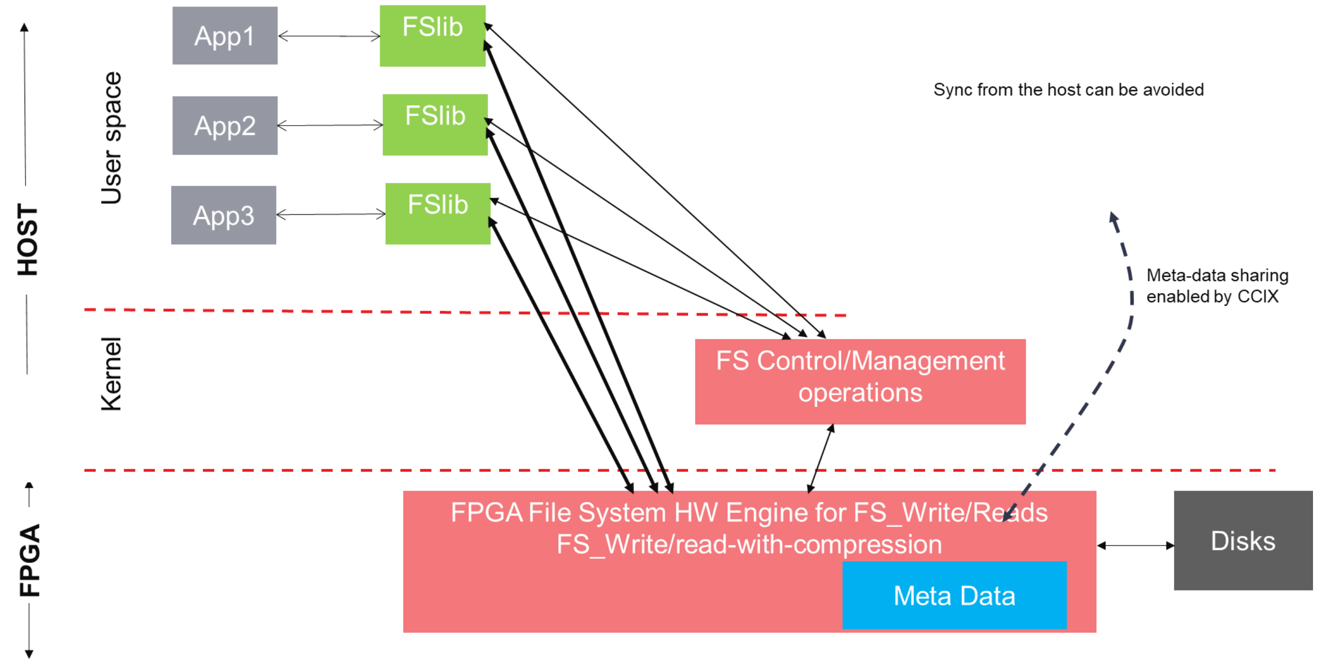
Split File System Operation Distribution Between Host & FPGA

- Instead of full file system offload we propose a split file system with Metadata share over CCIX interface
- CPU Handled operations:
  - fs_open – Creates new file or reopens the existing file
  - fs_exist – Checks whether the file exists
  - fs_rename – Renames existing file
  - fs_terminate – closes the file system
  - fs_create – creates the file system
  - file_size – Returns the file size
  - file_close – closes the file
  - file_truncate – truncates the file to the specified size
- All these operations need not be sent to FPGA as these can read/edit the shared structures
- In total, for a 5 minute WT performance run these functions were called for around 150 times
- FPGA Handled operations:
  - fs_read – Reads a data block from file
  - fs_write – writes a data block to file
- In total, for a 5 minute WT performance there are ~1565000 reads+writes

Split File System Operation Distribution Between Host & FPGA


HOST
User space
Kernel
FPGA
App1
App2
App3
FSlib
FSlib
FSlib
Meta Data
Meta-data sharing enabled by CCIX
FS Control/Management operations
FPGA File System HW Engine for FS_Write/Reads
FS_Write/read-with-compression
Disks

Meta-data in the FPGA Attached Memory


HOST
User space
Kernel
FPGA
App1
App2
App3
FSlib
FSlib
FSlib
FS Control/Management operations
FPGA File System HW Engine for FS_Write/Reads
FS_Write/read-with-compression
Meta Data
Disks
Meta-data sharing enabled by CCIX

Sync from the host can be avoided

Questions?