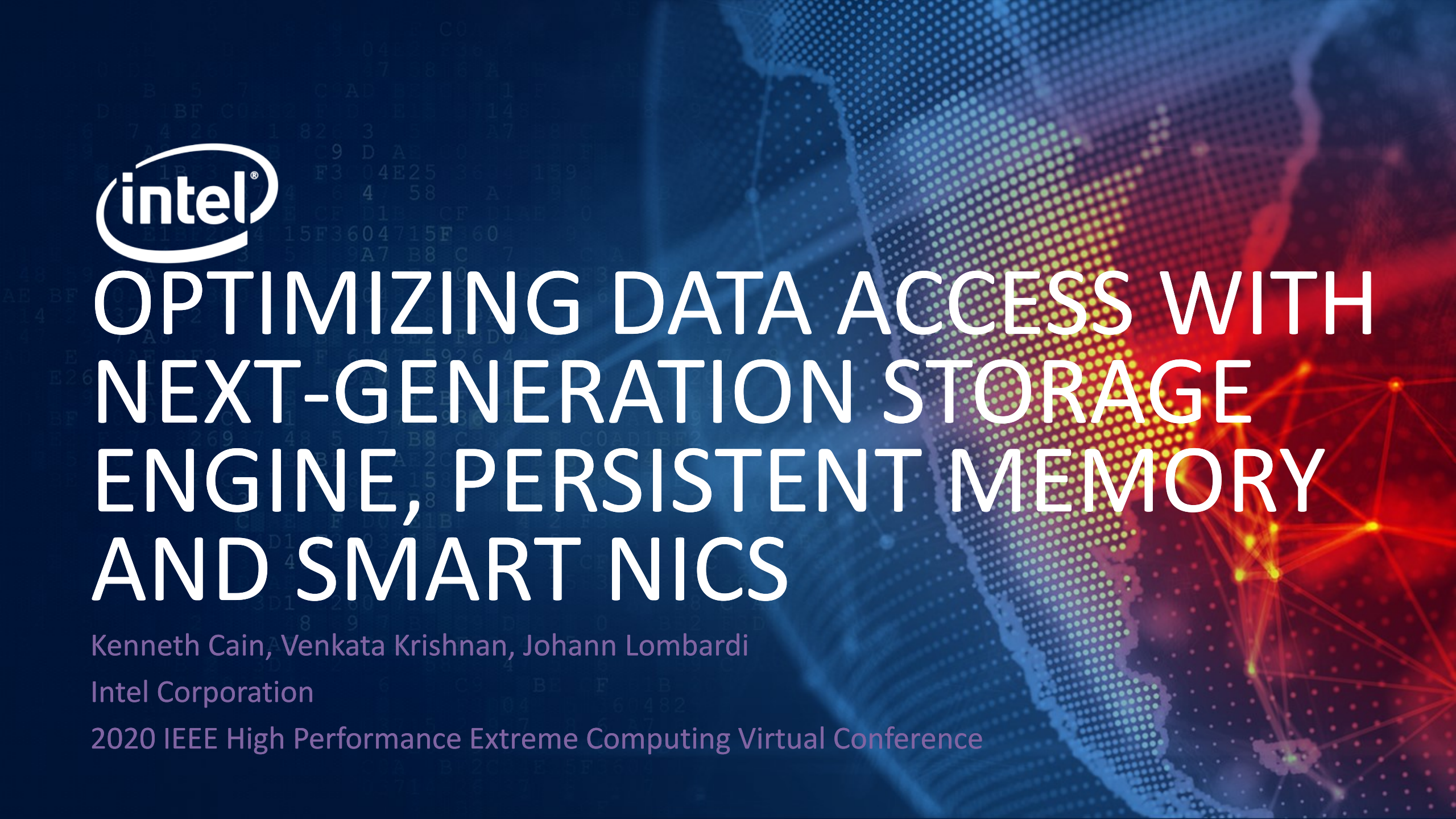


OPTIMIZING DATA ACCESS WITH NEXT-GENERATION STORAGE ENGINE, PERSISTENT MEMORY AND SMART NICS

Kenneth Cain, Venkata Krishnan, Johann Lombardi

Intel Corporation

2020 IEEE High Performance Extreme Computing Virtual Conference

NOTICES AND DISCLAIMERS

Intel technologies' features and benefits depend on system configuration and may require enabled hardware, software or service activation. Performance varies depending on system configuration.

No product or component can be absolutely secure.

Tests document performance of components on a particular test, in specific systems. Differences in hardware, software, or configuration will affect actual performance. For more complete information about performance and benchmark results, visit http://www.intel.com/benchmarks .

Software and workloads used in performance tests may have been optimized for performance only on Intel microprocessors. Performance tests, such as SYSmark and MobileMark, are measured using specific computer systems, components, software, operations and functions. Any change to any of those factors may cause the results to vary. You should consult other information and performance tests to assist you in fully evaluating your contemplated purchases, including the performance of that product when combined with other products. For more complete information visit http://www.intel.com/benchmarks .

Intel Advanced Vector Extensions (Intel AVX) provides higher throughput to certain processor operations. Due to varying processor power characteristics, utilizing AVX instructions may cause a) some parts to operate at less than the rated frequency and b) some parts with Intel® Turbo Boost Technology 2.0 to not achieve any or maximum turbo frequencies. Performance varies depending on hardware, software, and system configuration and you can learn more at http://www.intel.com/go/turbo.

Intel's compilers may or may not optimize to the same degree for non-Intel microprocessors for optimizations that are not unique to Intel microprocessors. These optimizations include SSE2, SSE3, and SSSE3 instruction sets and other optimizations. Intel does not guarantee the availability, functionality, or effectiveness of any optimization on microprocessors not manufactured by Intel. Microprocessor-dependent optimizations in this product are intended for use with Intel microprocessors. Certain optimizations not specific to Intel microarchitecture are reserved for Intel microprocessors. Please refer to the applicable product User and Reference Guides for more information regarding the specific instruction sets covered by this notice.

Cost reduction scenarios described are intended as examples of how a given Intel-based product, in the specified circumstances and configurations, may affect future costs and provide cost savings. Circumstances will vary. Intel does not guarantee any costs or cost reduction.

Intel does not control or audit third-party benchmark data or the web sites referenced in this document. You should visit the referenced web site and confirm whether referenced data are accurate.

OUTLINE

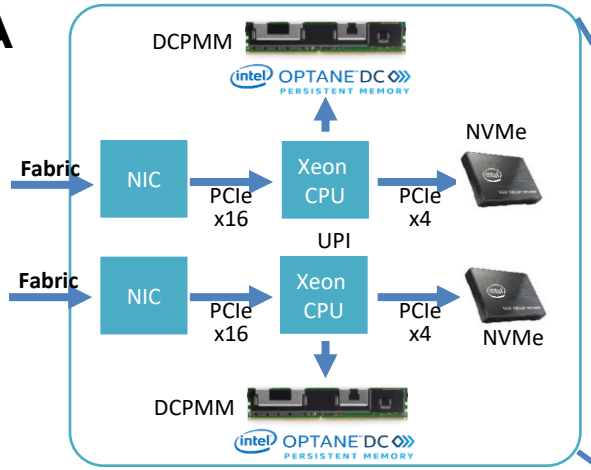
- **Overview: Distributed Asynchronous Object Storage (DAOS)**
  - Architecture
  - Examples of network and compute intensive operation
- **Architecture Research: COnfigurable network Protocol Accelerator (COPA)**
  - HW/SW SmartNIC Framework
- **Architecture Research Efforts Toward DAOS with COPA**


DAOS
Distributed Asynchronous Object Storage

DAOS ARCHITECTURE

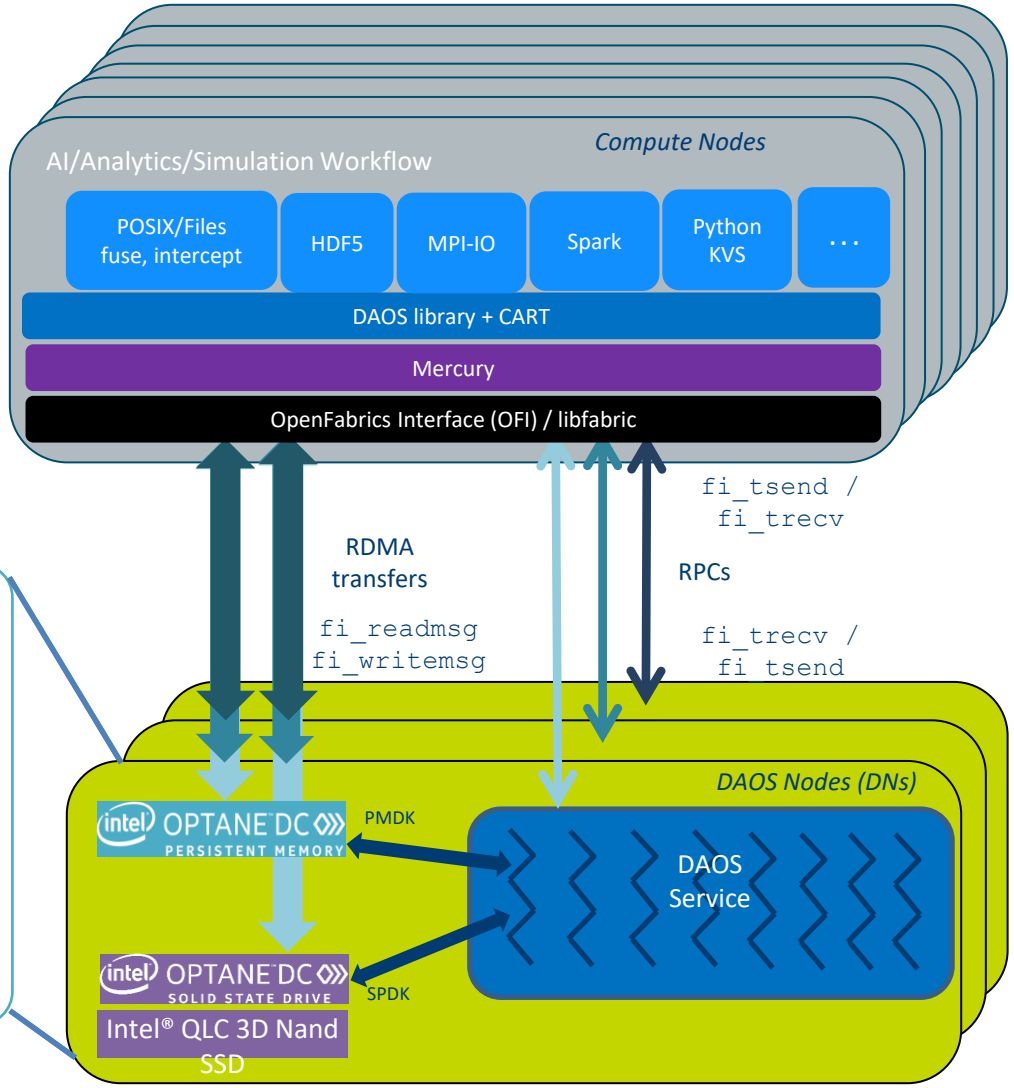
Two-Level Data Placement and High-Performance Communications

- **Placement Level 1: Client Choose Server(s)**
  - **Client-calculated** jump consistent hash based on key, object class
  - Fault domain aware
- **Placement Level 2: Server Choose Media:**
  - Data Center Persistent Memory (DCPMM) or NVMe SSD
  - DCPMM ← app. small, byte-granular data, DAOS metadata
  - NVMe SSD ← app-only bulk data (for high throughput)
- **Communications: iWARP, RoCE, IB, OPA**
  - OS-bypass for low-latency, high message rate I/O
  - Servers initiate RDMA to/from
    - DCPMM (zero copy) + PMDK library for flush
    - DRAM + SPDK library for NVMe SSD I/O
  - Clients (libdaos):
    - **No:** copies, context switches, locking, caching dedicated cores



Dual-Socket DAOS Nodes (DNs)

Intel® Xeon servers with DCPMM & NVMe SSDs

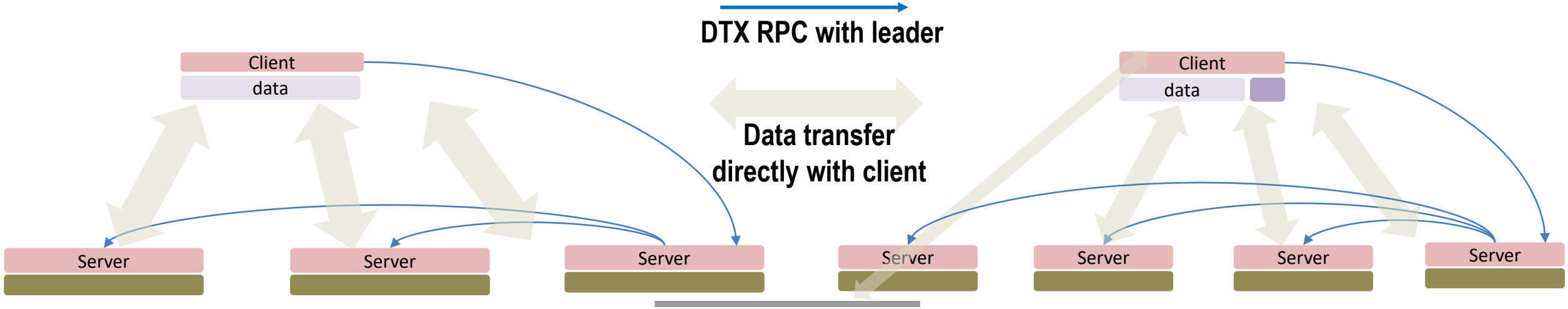


DAOS ARCHITECTURE

Data Protection and Self-Healing / Rebuild

- **Degraded mode – client I/O satisfied by surviving servers**
  - Non-blocking protocol for server fail-out
- **Self-healing / rebuild (online recovery)**
  - Declustered – per object, select alternate server storage to restore original degree of replication
    - Many alternate nodes in parallel – pull object data from surviving servers
  - Throttled – to control impact to serving ongoing client I/O requests
- **Leader server chosen to manage distributed transaction protocol (DTX)**
  - Chosen algorithmically based on key – no single leader node bottleneck

- **Replication**
- **Erasure Code (EC)**



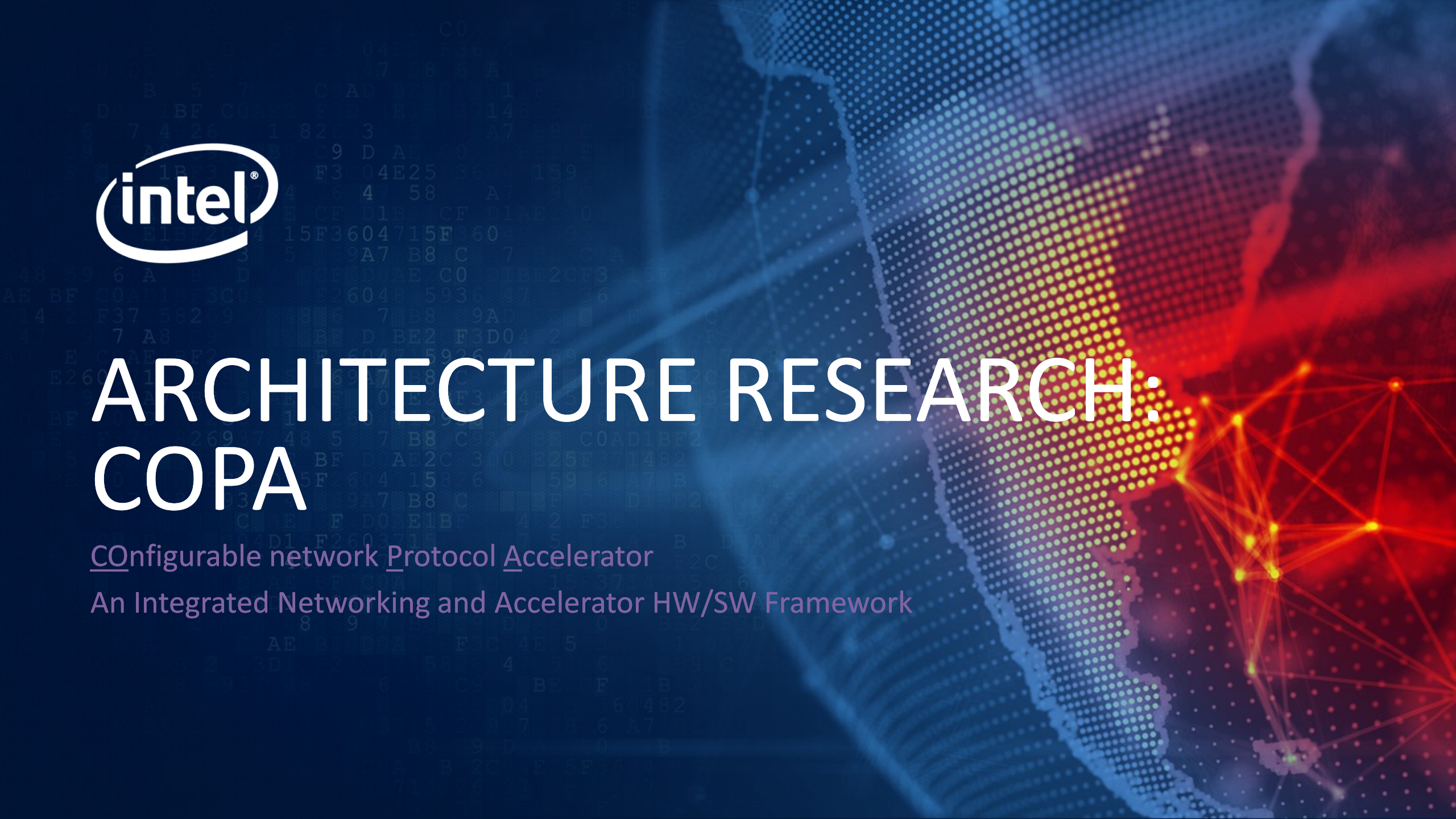
DAOS ARCHITECTURE: END-TO-END DATA INTEGRITY

And Architectural Motivations Toward Smart NIC and SoC Based Storage Nodes

- **Protection Against Data Corruption: via Checksums**
  - Protect both keys and values against “silent data corruption” over network or in storage media
  - Calculated on client, verified and stored on server (optionally calculated on server for more insight)
  - Xeon clients/servers: checksums via Intel Intelligent Storage Acceleration Library (ISA-L)

- **Architecture Motivations, Disaggregated + Scalable + Reliable Storage Use Case**
  - Computationally intensive: e.g., data protection (Erasure Code) ; end-to-end integrity (checksum)
    - ISA-L for checksum use will consume CPU core 100% and have impact on CPU cache contents
  - Network intensive: e.g., self-healing rebuild with scalable parallel communication among all servers in storage pool
  - Infrastructure size(footprint) and power consumption

- **Suggestive of Need For: Smart NIC, and Smart NIC in SoC Architecture**
  - Need: HW-based storage functions – to free up CPU cycles and cache for client apps & storage service
  - Need: standards-based networking to ease porting of storage software (client + server) to use Smart NIC



ARCHITECTURE RESEARCH: COPA

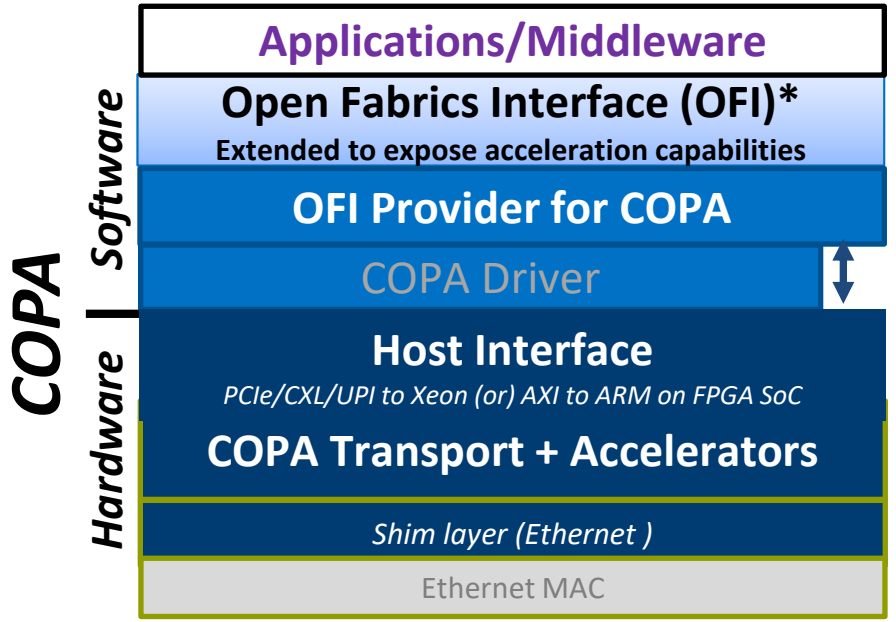
COnfigurable network Protocol Accelerator

An Integrated Networking and Accelerator HW/SW Framework

COPA ARCHITECTURE

COPA provides an integrated networking and accelerator framework on an FPGA with programming simplicity

- Supports full RDMA (PUT/GET) based communication
- Accelerators modules integrated with communication
- Open standards API (libfabric/OFI) + extensions

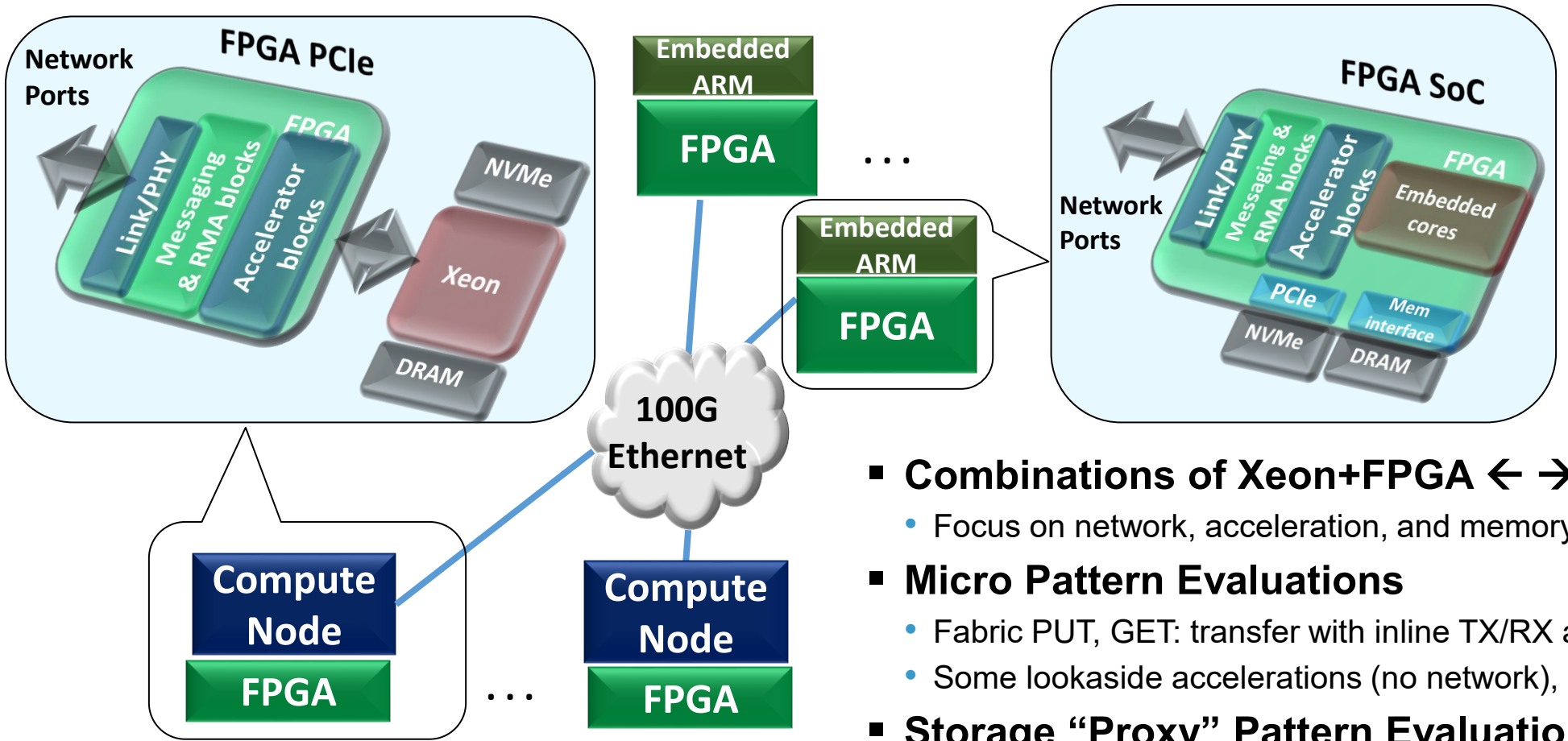


COPA provides streaming (inline) compute during TX/RX and traditional (lookaside) acceleration

- Local invocation by software
- Remote invocation by inbound packet (no CPU/OS involvement)

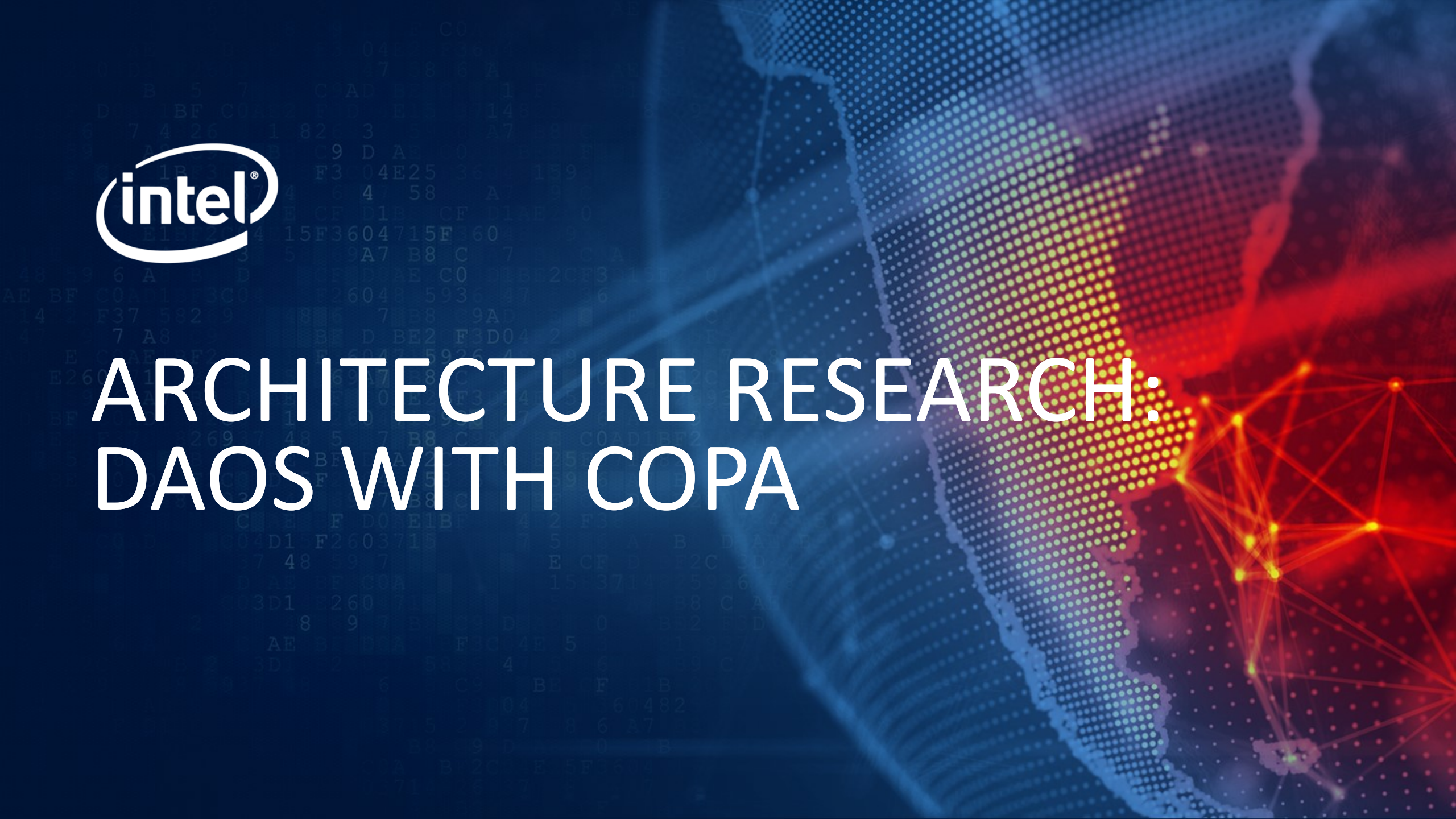
SYSTEM COMPONENTS

FPGA SoC and FPGA PCIe (Currently Stratix 10) on 100GbE Network



- **Combinations of Xeon+FPGA ← → FPGA SoC Exercised**
  - Focus on network, acceleration, and memory (not NVMe SSD here)
- **Micro Pattern Evaluations**
  - Fabric PUT, GET: transfer with inline TX/RX acceleration (CRC64)
  - Some lookaside accelerations (no network), e.g., CRC64
- **Storage "Proxy" Pattern Evaluations**
  - Update and Fetch (RPCs, PUT/GET transfer, inline/lookaside CRC64)

V. Krishnan, O. Serres and M. Blocksome, "COnfigurable Network Protocol Accelerator (COPA) † : An Integrated Networking/Accelerator Hardware/Software Framework," *2020 IEEE Symposium on High-Performance Interconnects (HOTI)*, Piscataway, NJ, USA, 2020, pp. 17-24, doi: 10.1109/HOTI51249.2020.00018.



ARCHITECTURE RESEARCH: DAOS WITH COPA

DAOS AND COPA PRELIMINARY INTEGRATION

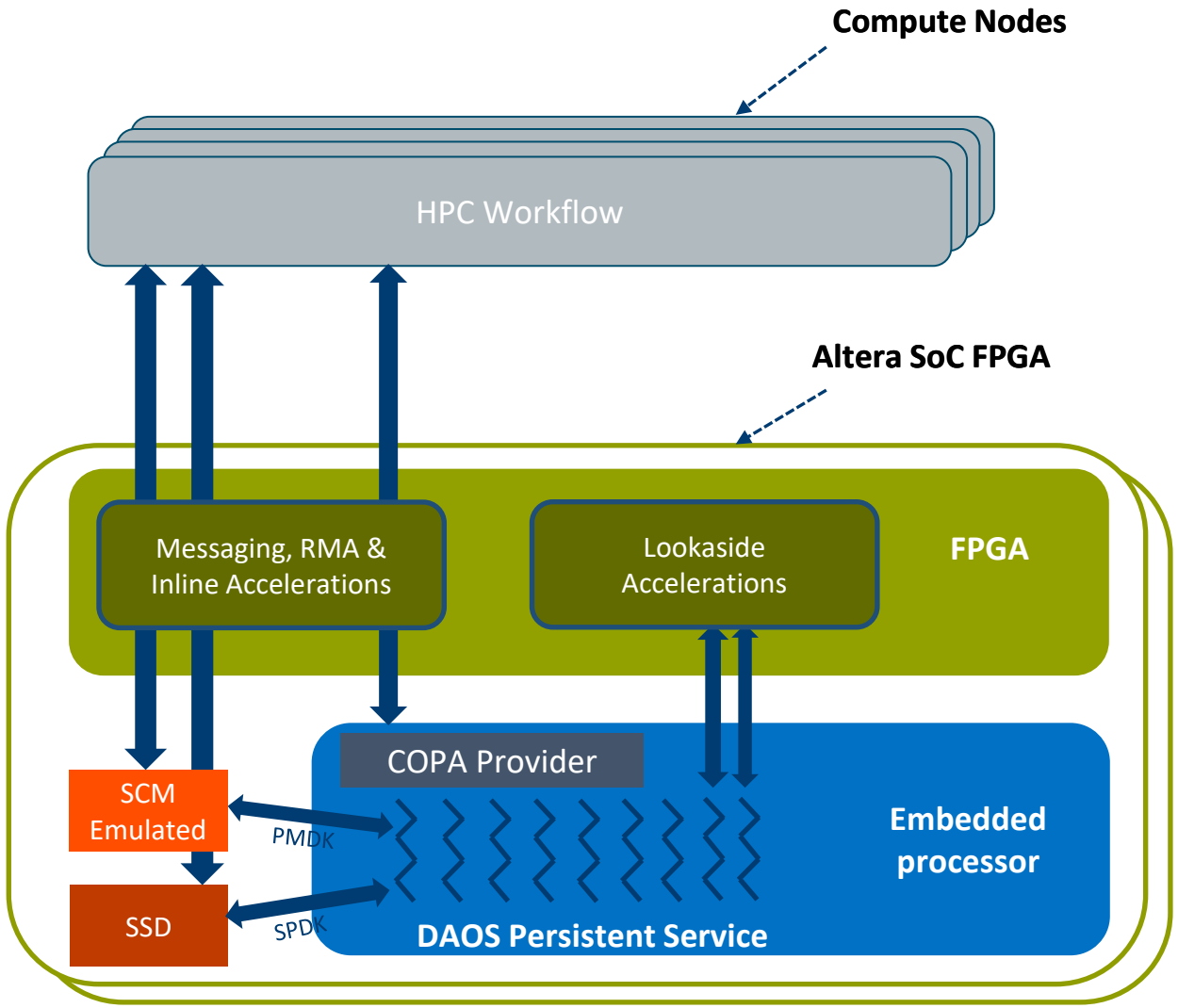
And Conclusions

DAOS and COPA Integration

- **Clients on Xeon hosts with COPA**
- **Service Embedded in SoC with COPA**
- **Fabric agnostic SW port (via OFI, COPA provider)**
- **Full Stack Run Including Server NVMe SSD I/O**
- **Storage Function Acceleration by COPA**
  - Lookaside HW CRC64 on clients + servers
  - Instead of SW CRC (ISA-L)

Conclusions and Potentials / Vision

- **DAOS Potential Enhancements**
  - use streaming / inline acceleration for checksums
  - engage more storage function offloads (e.g., Erasure Code)
- **Advanced HW Features Can Enable FPGAs as Autonomous Nodes. E.g.,:**
  - Hard Processor Cores
  - Compute Express Link (CXL) – perf., cache/memory coherence
  - Advanced memory – DDR, HBM, Persistent Memory
  - …



BACKUP

DAOS RESOURCES

Resource	URL
Source Code on GitHub	https://github.com/daos-stack/daos
Documentation	https://daos-stack.github.io/
Community Mailing List	https://daos.groups.io/
DAOS Solution Brief	https://www.intel.com/content/www/us/en/high-performance-computing/overview.html

DAOS PERFORMANCE

IO-500 Benchmarks

- **IOR**
  - Easy: any IOR pattern to show best-case performance without any explicit caching
  - Hard: single shared file with transfer 47008 bytes!
  - Separate Write and Read/verify runs.
- **mdtest**
  - Easy: private directory per process with empty files
  - Hard: shared directory with 3901-byte files
  - Separate write, read, stat, and delete runs
- **Find**
  - scan namespace created with IOR and mdtest

DAOS PERFORMANCE

IO-500 "Wolf" Testbed Configuration

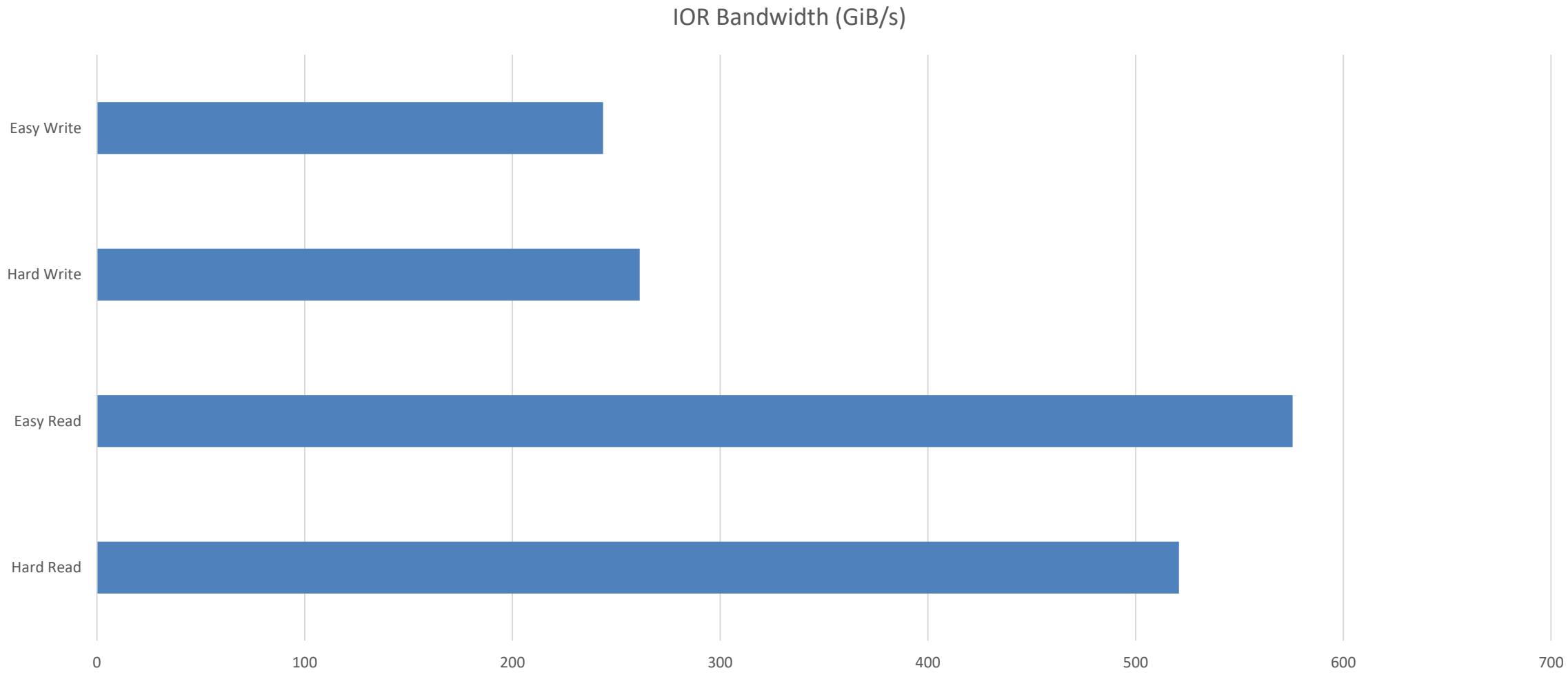
- **Cluster Summary**
  - 10x, 50x compute nodes (10 node, open challenges)
    - 42 ranks per node for 10 node challenge
    - 32 ranks per node for open challenge
  - 30x storage nodes
  - Dual-rail Omni-Path® fabric

- **Compute node (CN) specifications**
  - Mix of:
    - Broadwell, Haswell, Cascade Lake
  - 2x Intel® Omni-Path® 100 adaptors

- **Storage node (SN) specifications**
  - 2x Cascade Lake CPU
    - Xeon® Platinum 8260L @ 2.4GHz
    - 24 cores per CPU
  - 12x Optane® DC Persistent Memory DIMMs
    - Configured in app-direct/interleaved mode
  - 2x Intel® Omni-Path® 100 adaptors

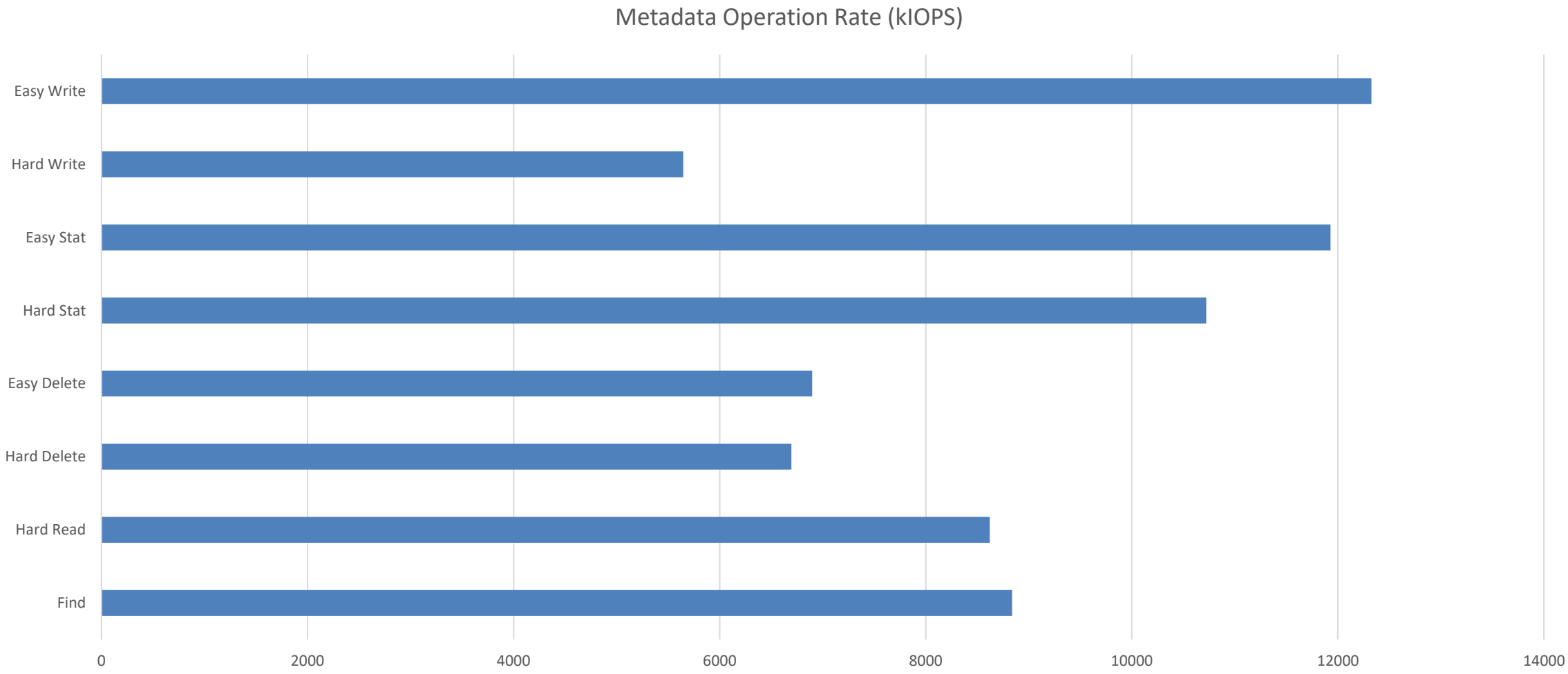
DAOS PERFORMANCE

IOR Bandwidth on Wolf Cluster



DAOS PERFORMANCE

Metadata IOPS on Wolf Cluster

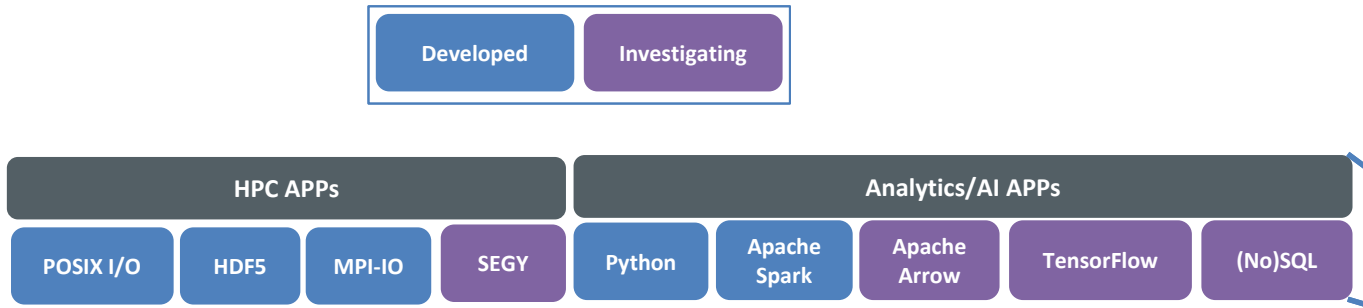


DAOS & IO-500 IN NUMBERS

Metric	Intel DAOS IO-500 Run
Total Number of files created	5.8 Billions
Biggest file size	79.2 TiB
Time to fully read the big file	141 seconds
File scanning rate (including file size retrieval)	10 Millions/s

DAOS ARCHITECTURE

Client Library and Interfaces



- **POSIX I/O – namespace distributed over servers**
  - DAOS Filesystem (libdfs) – apps / frameworks may link directly
  - FUSE Daemon – transparent access to DAOS, involves syscalls
  - I/O Interception Library – OS bypass for read/write operations
- **MPI-IO Support**
  - MPI-IO Driver uses DAOS array API (+ libdfs for collective open)
- **Python Bindings**
  - Export key-value store objects
  - Integrate with dictionaries: iterator, direct assignment, etc.