SDC 20

BY Developers FOR Developers

Storage Developer Conference
September 22-23, 2020

Flexible Computational Storage Architectures

Neil Werdmuller & Jason Molgaard
Arm

Abstract

- Moving large amounts of data between storage and compute cannot scale given the ever-increasing storage capacities. A shift to computational storage that brings compute closer to the stored data provides the solution. Data-driven applications that benefit from database searches, data manipulation, and machine learning can perform better and be more scalable if developers add computation directly to storage.
- Flexibility is key in the architecture of a Computational Storage device hardware and software implementation. Hardware flexibility minimizes development cost, controller cost, and controller power. Software flexibility leverages existing ecosystems and software stacks to simplify code development and facilitate workload deployment to the compute on the drive. Implementing the hardware and software flexibility into a Computational Storage Drive requires forethought and deliberate consideration to achieve a successful solution.
- This presentation will show how to simplify computational storage architectures. Attendees will walk away with how to reduce power, area, and complexity of their computational storage controller, and leverage Linux and the Linux ecosystem of software to facilitate software development and workload management by taking advantage of computational storage capabilities.

Agenda

- What is driving Computational Storage?
- What are the controller architecture options?
- What is driving adoption of Linux?
- What are the key Linux workloads?
- Conclusions

What is Driving Computational Storage?

Why Computation is Moving to Storage

Bandwidth

Power

Cost

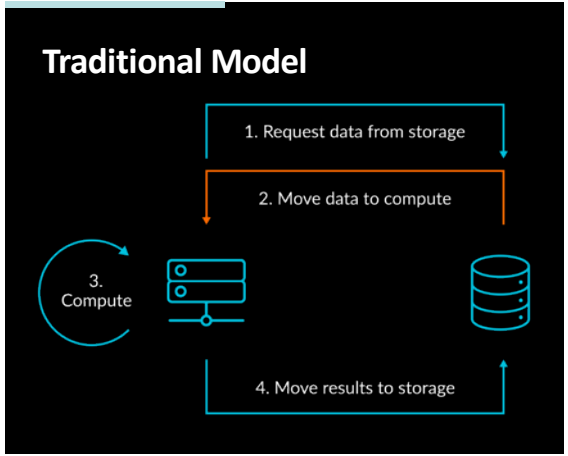
Latency

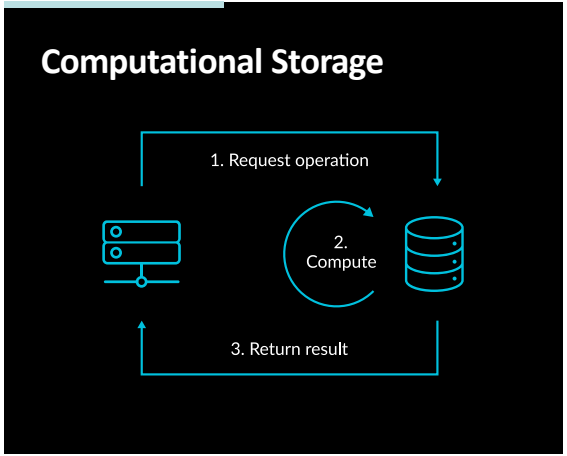
Reliability

Security

Computational Storage

Generating insight where data is stored

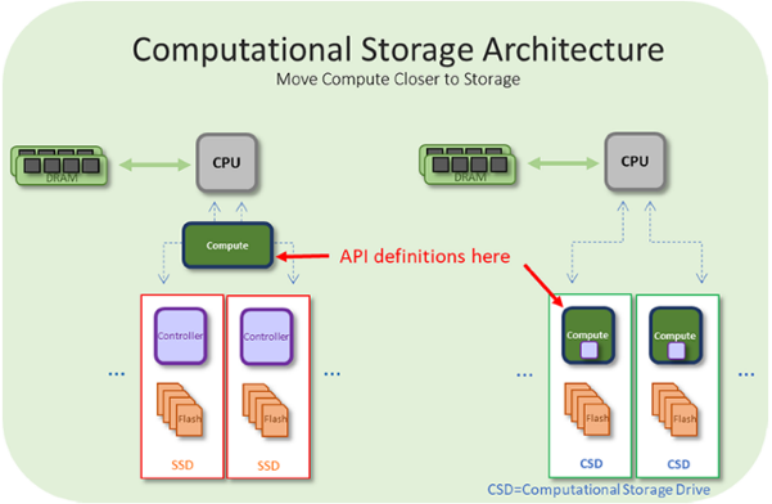




- Energy efficiency
- Low latency
- Security
- Data-centric workloads

SNIA Computational Storage TWG

- CSD Standard required for adoption
  - SSD purchasers require multi-sourcing
- Arm is a founder member in TWG
- Draft standard (0.5r1) available
- However, standards take considerable time to be developed and approved



- Many CSD early developers using Xilinx FPGA – but now migrating to ASIC solution
  - ASIC is lower power and lower cost and easier to program

SNIA Computational Storage Technical Working Group (TWG)

44 participating companies and 144 individual members

SNIA COMPUTATIONAL STORAGE

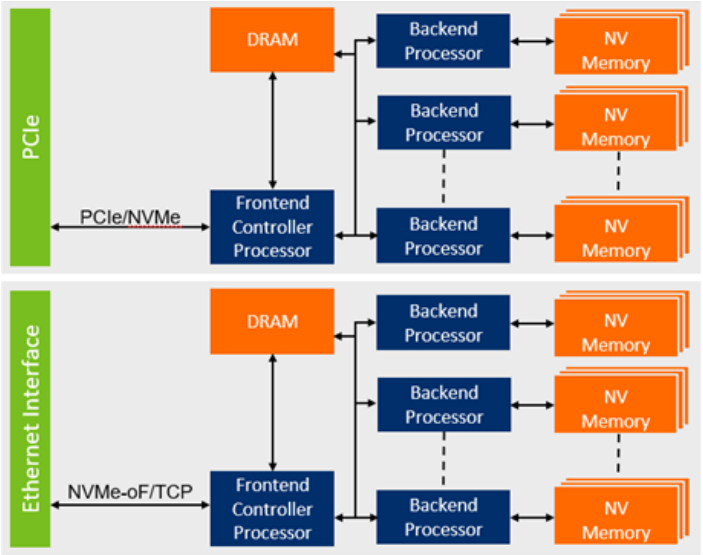
Computational Storage over PCIe and NVMe-oF/TCP

SNIA CS TWG is defining NVMe extensions to deliver Computational Storage
Discovery, configuration and direct usage interaction protocols

Computational Storage PCIe/NVMe SSD
Adds new SNIA CS NVMe protocols

Computational Storage NVMe-oF/TCP

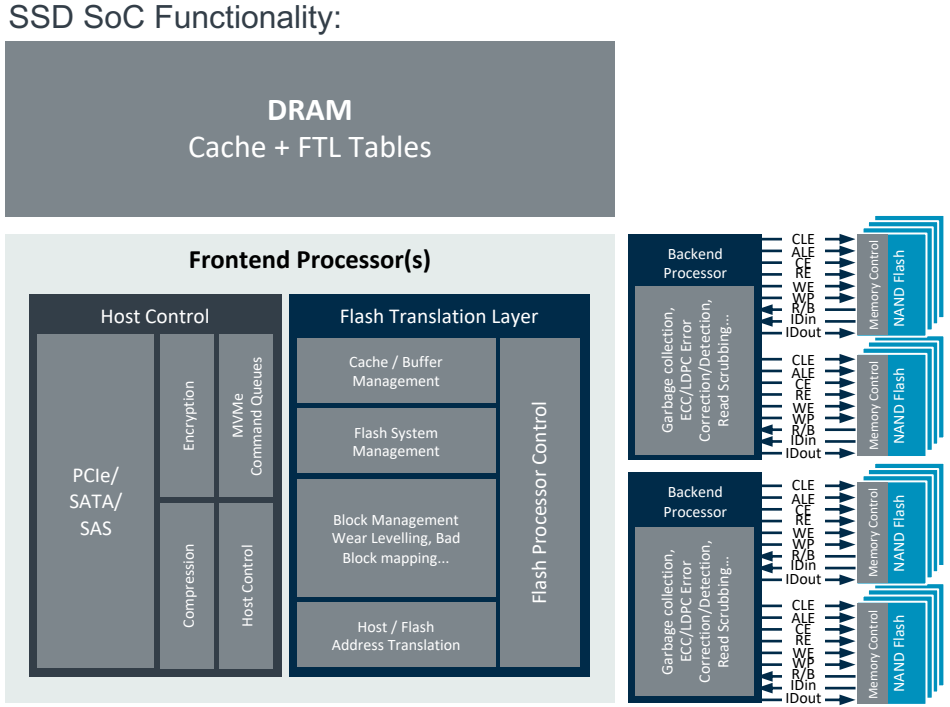
- NVMe over Ethernet fabric transports commands
- Processes standard NVMe-oF/TCP commands
- New SNIA CS NVMe protocols encapsulated



What are the Controller Architecture Options?

Compute in SSD Controllers

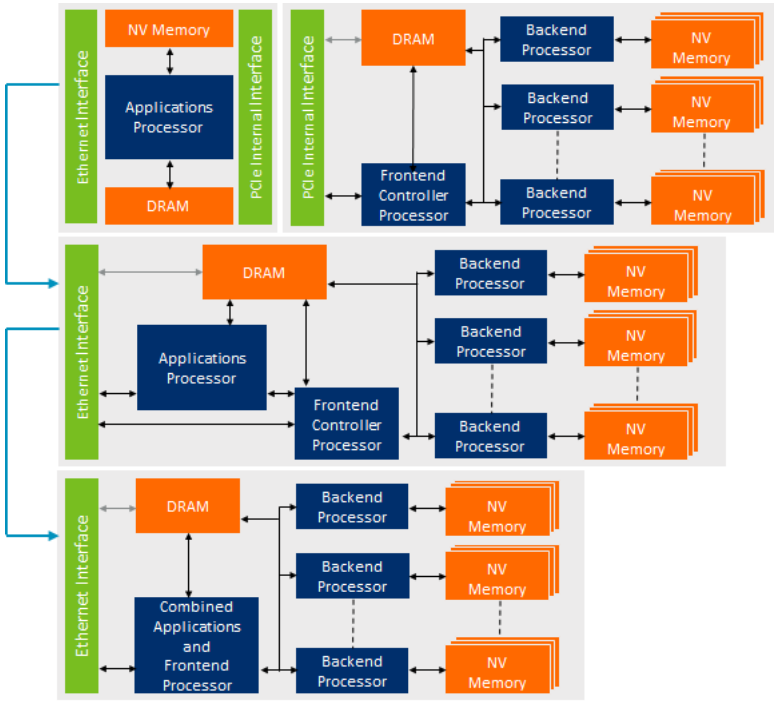
- Frontend: Host I/F + FTL
  - Arm Cortex-R or Cortex-A series
- Backend: Flash management
  - Cortex-R or Cortex-M series
- Accelerators:
  - Hardware accelerators…
  - Encryption, LDPC, Compression…
  - Arm Neon, ML, FPGA…
- **DRAM ~1GB per 1TB of flash**
- Storage: 256GB to 64TB… flash
- Interfaces: PCIe/SATA/SAS…



Options to Add On-drive Linux

Three main options to run on-drive Linux:

1. Add a separate applications processor SoC in-drive
2. Integrate into a single SoC for lower cost/latency
3. Single compute cluster for lowest cost/latency



Linux storage and DRAM requirements e.g. Debian 9 ‘buster’ states system requirements…

Recommended Minimum System Requirements

Install Type	RAM (minimum)	RAM (recommended)	Hard Drive
No desktop	128 megabytes	512 megabytes	2 gigabytes

Smaller Linux distributions are also available

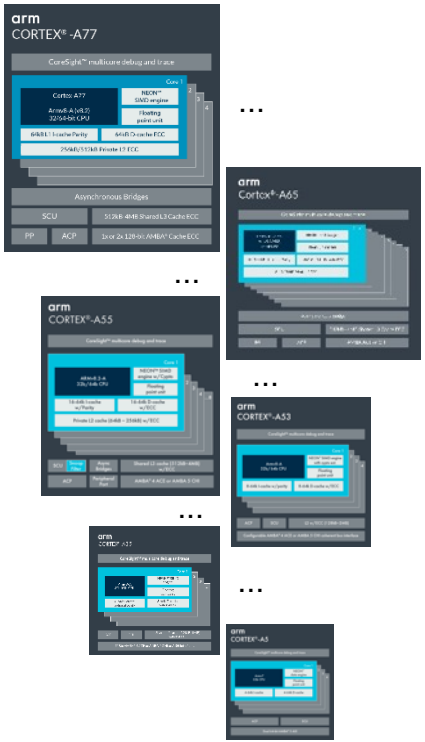
Typical 16TB SSD already has ~16GB DRAM

Applications Processors: Low-cost, Low-power Compute

- Linux requires an applications processor
  - Memory Management Unit (MMU) to virtualize memory
  - Cortex-A series have 21 processors available + strong roadmap
  - From a single processor to many clusters of compute
  - **Meeting every performance point at the lowest possible power**

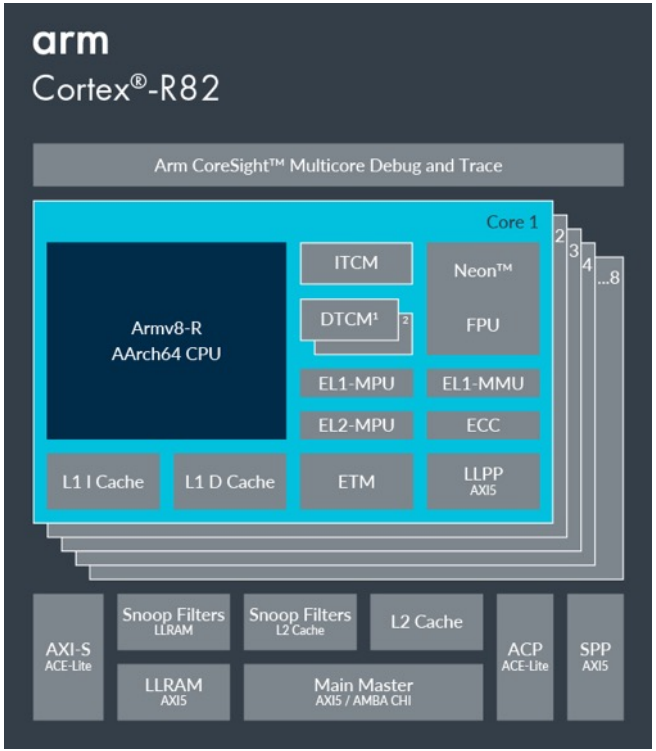
- Some eSSD controllers already use application processors
  - Other controllers, using real-time processors, can easily add them

- Arm Neon enables high-performance ML as standard
  - Single Instruction Multiple Data (SIMD) greatly accelerates ML
  - ML processors, FPGAs, ISPs or dedicated hardware easily integrated



www.arm.com/products/silicon-ip-cpu

Cortex-R82: Enabling Next-Generation Storage

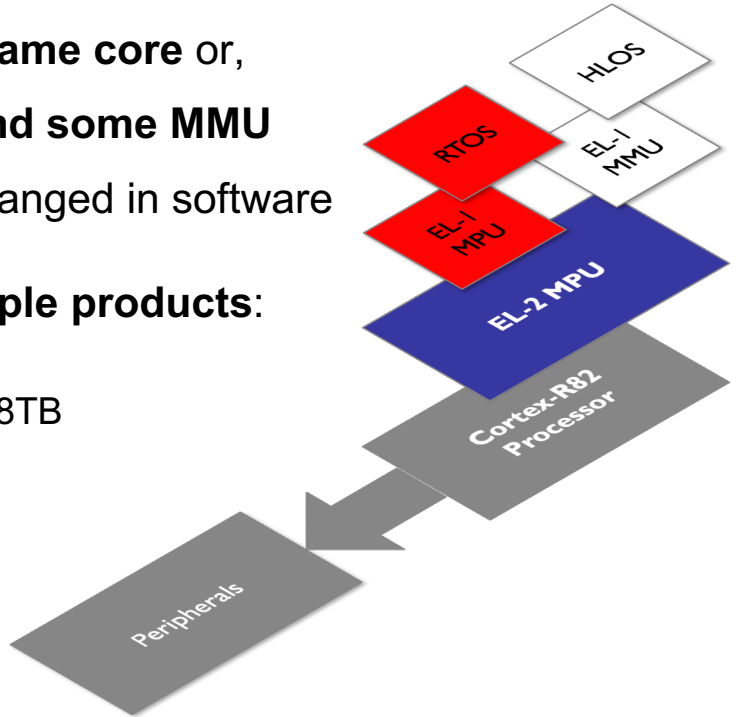


- First 64-bit Cortex-R series processor
  - Large address map for high capacity drives
- Hard real-time needed for controller FTL
  - Specialized hard real time features: lowest latencies and consistent performance
- Memory Management Unit
  - Enabling Linux (or other HLOS) support
- Shared coherent memory across the system
  - Between clusters and even across CXL/CCIX
- Advanced Machine Learning support with Neon

Cortex-R82: Real-time MPU + Linux MMU at EL-1

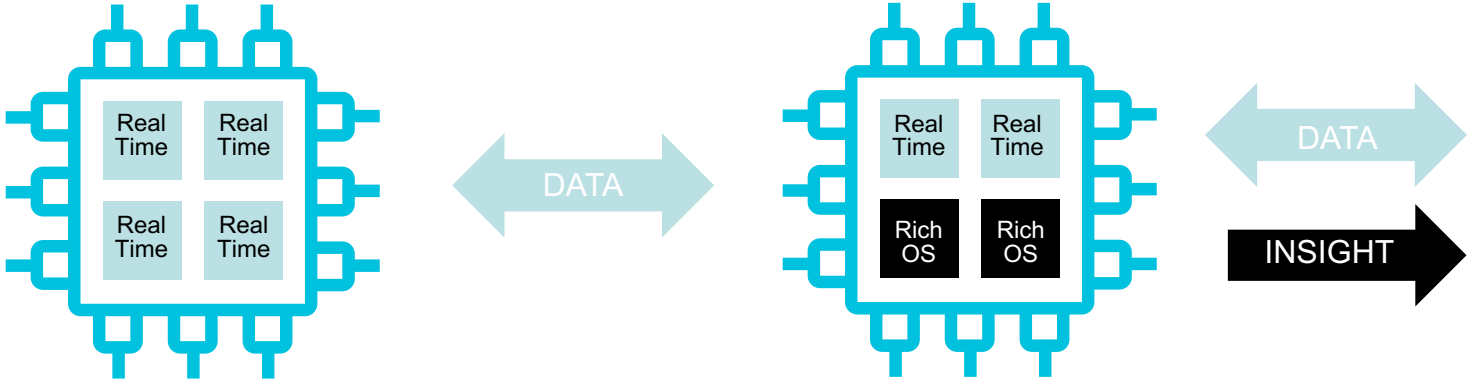
- Separate virtual machines for bare metal/RTOS and MMU contexts
- Enables real-time and Linux to co-exist on **same core** or,
- **Some cores in cluster can be real-time and some MMU**
- Balance of real-time / Linux cores can be changed in software
- Enables a **single controller ASIC for multiple products**:
  1. All real-time eSSD handling 16TB
  2. Half Linux processing / half real-time handling 8TB

Reducing the number of high cost mask-sets



Enabling Rich Operating System Workloads On-Drive

Generating insight where data is stored

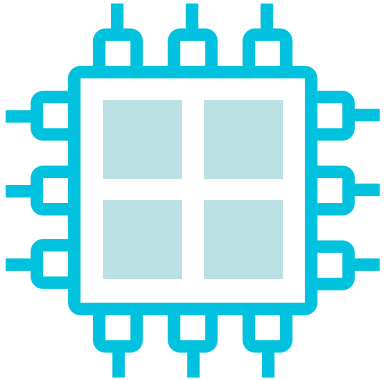


Storage controllers traditionally run real-time workloads to store and access data

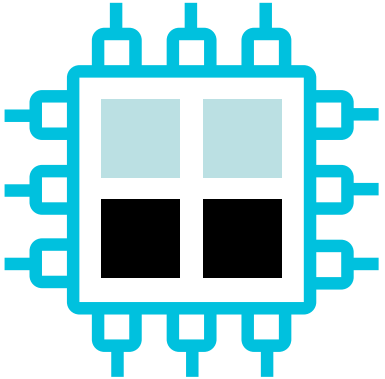
Introducing a Memory Management Unit enables Rich Operating System workloads to generate insight from the data

Designing Flexible Controllers with Cortex-R82

One storage controller tape-out for both pure storage and computational storage



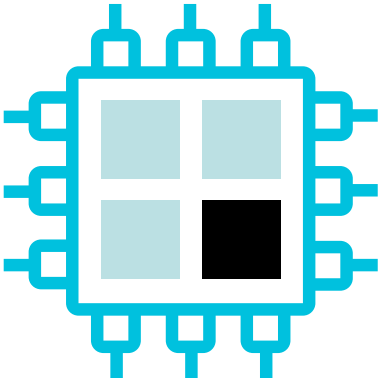
Classic Enterprise Drive



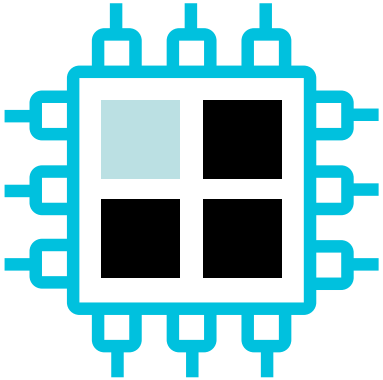
Computational Storage Drive

Flexibility to Change the Balance of Workload

Dynamically adjusting the workload balance on the controller based on external demands



Storage Balance



Computation Balance

Computation workload

Real-time storage workload

What is Driving the Adoption of Linux?

Accelerating Innovation for Storage Developers

Cloud-native software technologies speed development and accelerate innovation

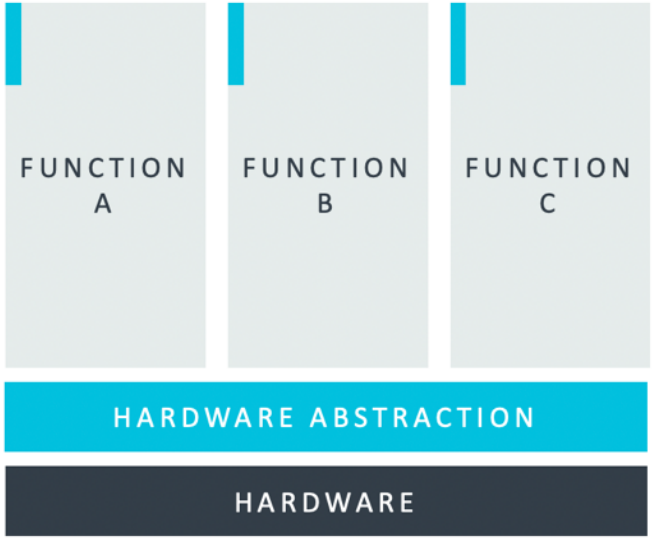
The ability to run Linux on a storage device opens up a range of software tools and technologies to developers

Arm enjoys full support for many Linux distributions and open-source software technologies

Familiarity with these tools and technologies will accelerate innovation, development and deployment

Software Defined System – The Trend

What is ”Software Defined” in this context ?



- Functions enabled by software are abstracted from hardware
- Functions enabled using ubiquitous cloud native Service-Oriented Architecture (SOA) software development model:
  - Functions delivered as services that are self-contained unit of software
  - Mechanism for publishing available services and their characteristics
  - Control of the use of these services

Project Cassini

Initiative to ensure a cloud native software experience across the Arm edge ecosystem

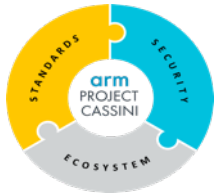
- ✓ Designed to support and enable diverse Arm based hardware
- ✓ Enables the ecosystem to leverage the significant investment being made in cloud native software

Project Cassini is made up of three main components

- ✓ Robust standards leverage software development, but allow for & celebrate hardware diversity
- ✓ Security APIs & certification that can be trusted by developers across workloads and ecosystems
- ✓ Cloud native software stacks and reference use cases

Tangible benefits across the value chain

- ✓ Provides choice and optimized hardware (diversity) for ODM, OEM, and end users
- ✓ Reduces cost and friction of supporting Arm for ISVs and System Integrators

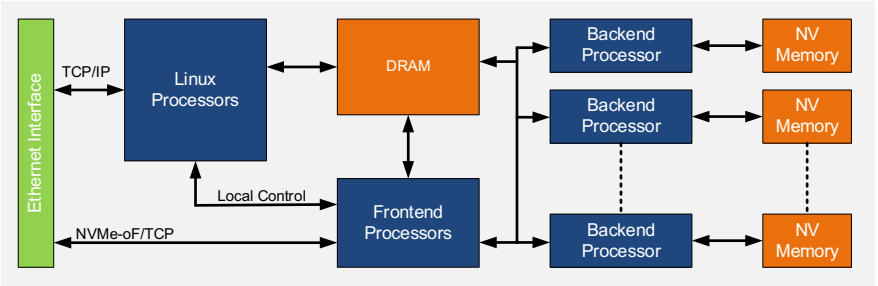


Join us at the Arm Dev Summit for the latest on Project Cassini

project-cassini@arm.com

Extending NVMe-oF/TCP with On-drive Linux

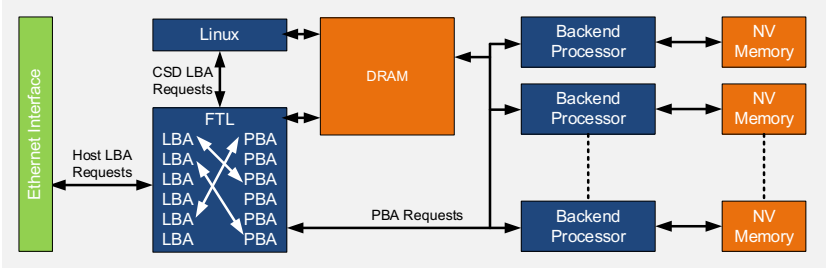
Computational Storage NVMe-oF/TCP
Processes NVMe-oF/TCP commands
Standard SNIA CS NVMe-oF/TCP CSD
+ Standard TCPIP access to Linux ‘server’



- Connects to standard data centre Ethernet switches - lower cost than PCIe switches
- Runs any standard Linux distribution
- Workloads deployed as containers using standard tools e.g. Kubernetes/Docker
- **Linux ‘understands’ the file system - NVMe drive just understands blocks/pages**
- Linux applications operate on files – data just moved from NAND to DRAM
- Managed using the same systems as any other server
- Security managed through standard techniques on Arm such as TrustZone

Autonomous Processing on Drive

- Host sends and receives data to Logical Block Addresses (LBAs) on the drive
- Controller maps LBAs to Physical Block Addresses (PBAs) in the FTL
- Storage also stores the file system that maps files to LBAs
  - A file, such as a JPEG, may be made up of multiple blocks of data
- Computational Storage Drive (CSD) with Linux can mount the file system
- On-drive Linux can now access and operate on complete files, not just blocks



- Enabling any processing that can be done on the host to be performed on the drive
- As files are being stored, processing can be performed autonomously
- Or, once stored the files can be processed in-situ at any point in time

CSD with Linux is a Low-cost Edge Server

Compute:
- Arm-based SoC

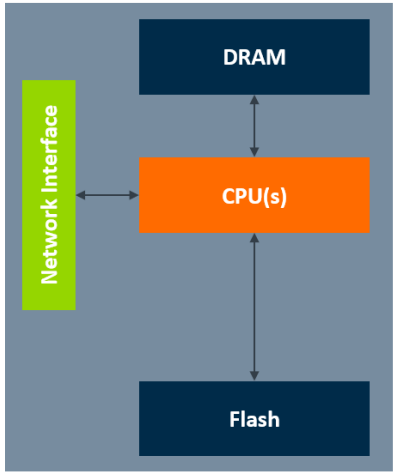
Memory:
- Shared DRAM

Storage:
- Shared Flash

Interface:
- Ethernet…

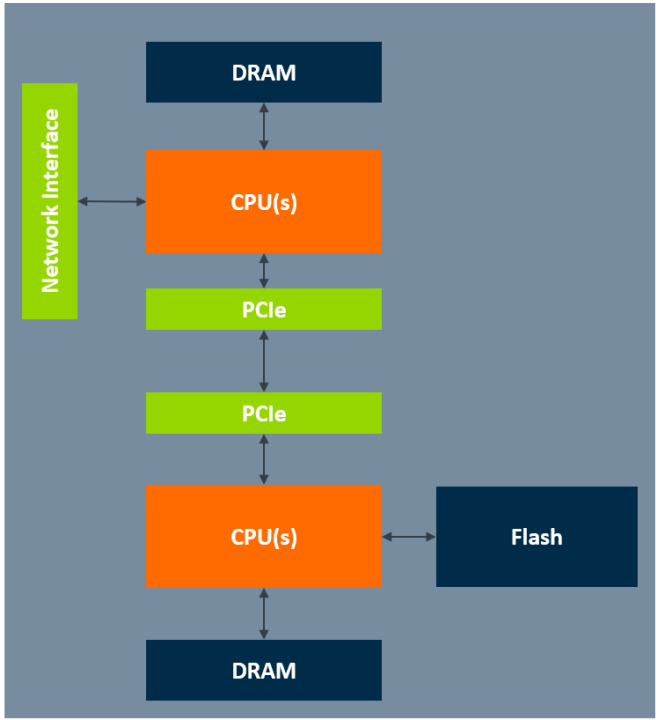
Power:
- Potentially powered over Ethernet (PoE ~15…100W)

SSD Based Edge Server:



Vs.

Classic Edge Server:



What are the key Linux Workloads?

Leverage Arm Cloud Native Software Ecosystem

CI/CD: Drone.io, Travis CI, GitLab, Jenkins, GitHub Actions, Azure Pipelines, AWS CodePipeline

WORKLOADS: NGINX+, KEYDB, MySQL, ceph, cadence, RAPIDS, DATADOG

LANGUAGE & LIBRARY: OpenJDK, GOLANG, LLVM, python, node JS

CONTAINERS & VIRTUALIZATION: docker, KVM, RANCHER, vmware, KubeEdge

OPERATING SYSTEM: Red Hat Enterprise Linux, fedora, CentOS, debian, SUSE, ubuntu

NETWORKING: .io, DPDK, OvS Open vSwitch, envoy, Istio, flannel, cilium

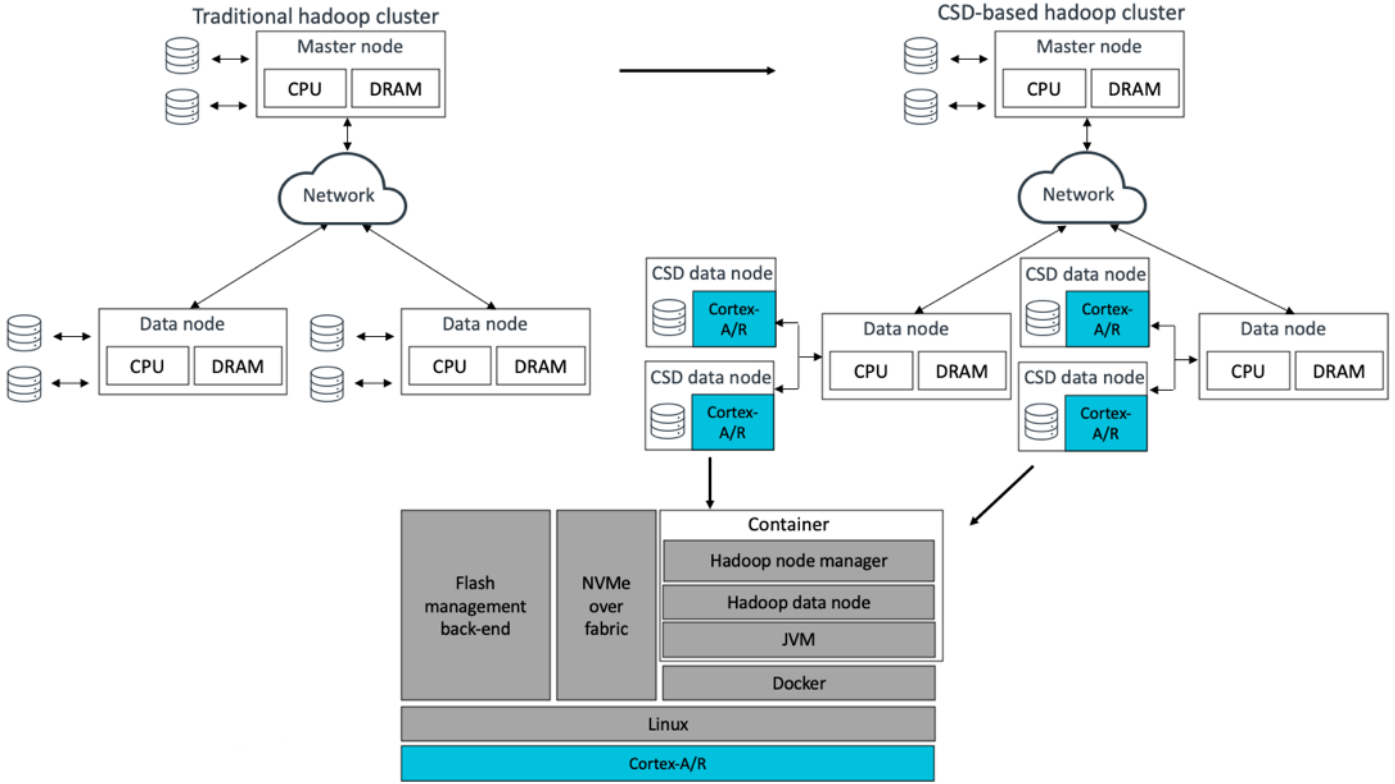
Migrating Linux Workloads to CSDs

- Key workloads and applications for CSDs:
  - Off-load (compression/encryption/encoding/etc.)
  - Database acceleration
  - AI/ML
  - Content Delivery Network
  - SmartNIC + CSD
  - Edge Computing
  - Image Classification
  - Video
  - Transportation
  - Custom workloads (customer specific workloads)

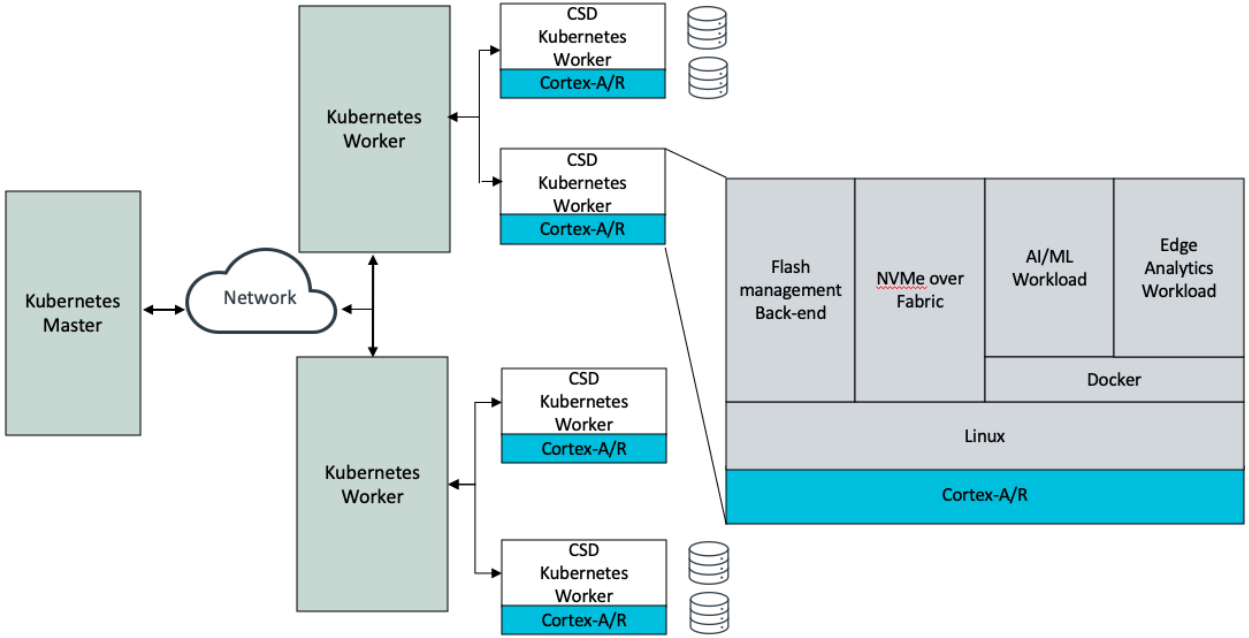
Provisioning Workloads to CSDs

- SNIA Computational Storage standardized protocols
  - Running over NVMe/NVMe-oF (under development)
- Containerization with Kubernetes, Docker…
  - Standard Linux approaches for workload deployment
- eBPF (Linux JIT compiled BPF programs)
  - Run in Linux virtual machine

CSD Enabled Energy-Efficient Hadoop Cluster



CSD Enabled Low-Latency Edge Analytics Deployment



Conclusion

Conclusion

- Computational storage workloads are diverse
  - Huge variety of use cases and applications
  - Best enabled through on-drive Linux
  - Innovation comes from the developer community

- CSD controllers require flexibility
  - Controller designs need to enable multiple products/workloads
  - Fast real-time to support increasing data rates and capacities
  - Dynamically balancing real-time and Linux capabilities

Thank you