Flexible Computational Storage Solutions

Flash Memory Summit

Neil Werdmuller & Jason Molgaard
Arm

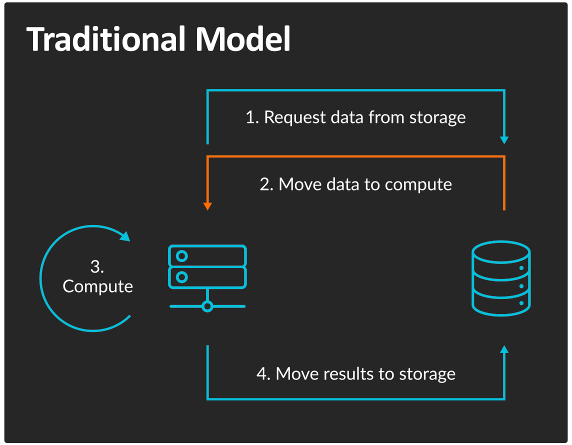
Agenda

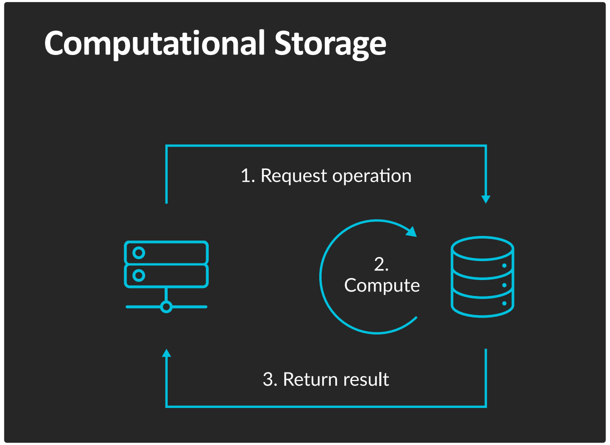
- What is driving Computational Storage?
- What are the controller architecture options?
- What is driving Linux and what are the key workloads?
- Conclusions

What is Driving Computational Storage?

Computational Storage

Generating insight where data is stored





- **Autonomous Computational Storage**
  - Performs compute on stored data
  - Builds results independently in storage
- **Host Managed Computational Storage**
  - Host sends micro-operations to the storage for compute

Energy efficiency

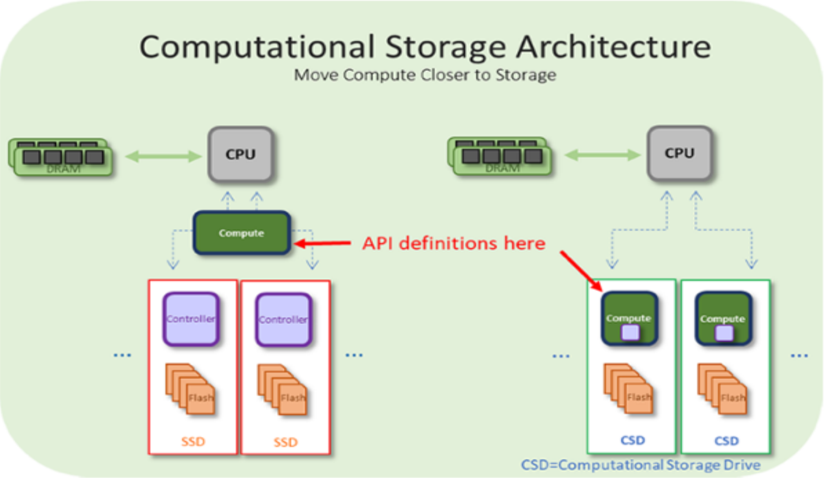
Low latency

Security

Data-centric workloads

SNIA Computational Storage TWG

- CSD Standard required for adoption
  - SSD purchasers require multi-sourcing
- Arm is a founder member in the 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 (TWG)

45 participating companies and 213 individual members

Computational Storage over PCIe and NVMe-oF/TCP

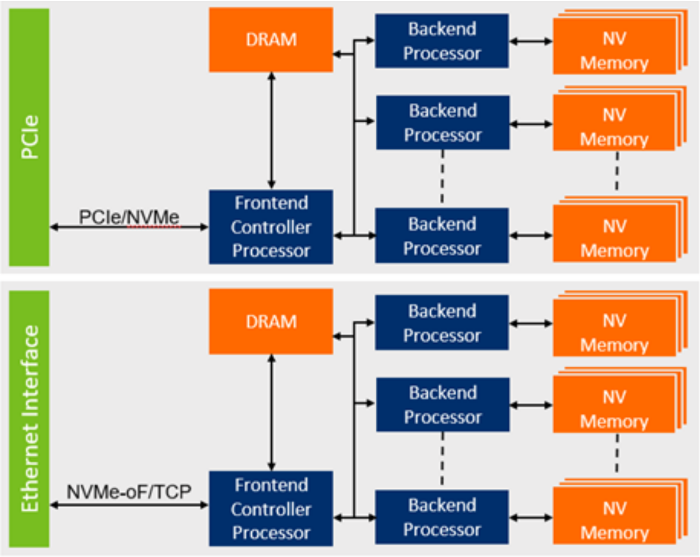
SNIA CS TWG is defining NVMe extensions to deliver Computational Storage discovery, configuration and direct usage interaction protocols

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



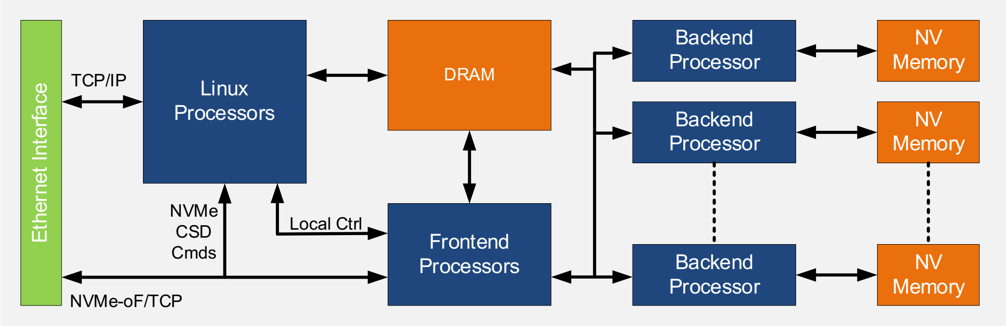
Extending NVMe-oF/TCP with On-drive Linux

Computational Storage NVMe-oF/TCP

Processes NVMe-oF/TCP commands

Standard SNIA CS NVMe-oF/TCP CSD

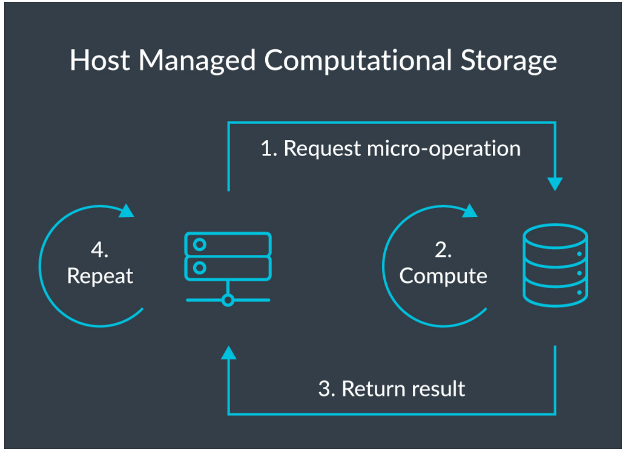
+ Standard TCPIP access to Linux 'server'



- Connects to standard data centre Ethernet fabric
- 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 adopts existing systems

Host Managed CSD with eBPF

- Programs defined in a hardware-agnostic bytecode and downloaded by the host to a device for later execution (eBPF is bytecode definition used)
- eBPF based on using Linux kernel
- Programs provided by a device, typically implemented in hardware (fixed-function) or firmware
- Host requests specific tasks/programs to be run on the data on the drive
- CSD returns results of each program



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

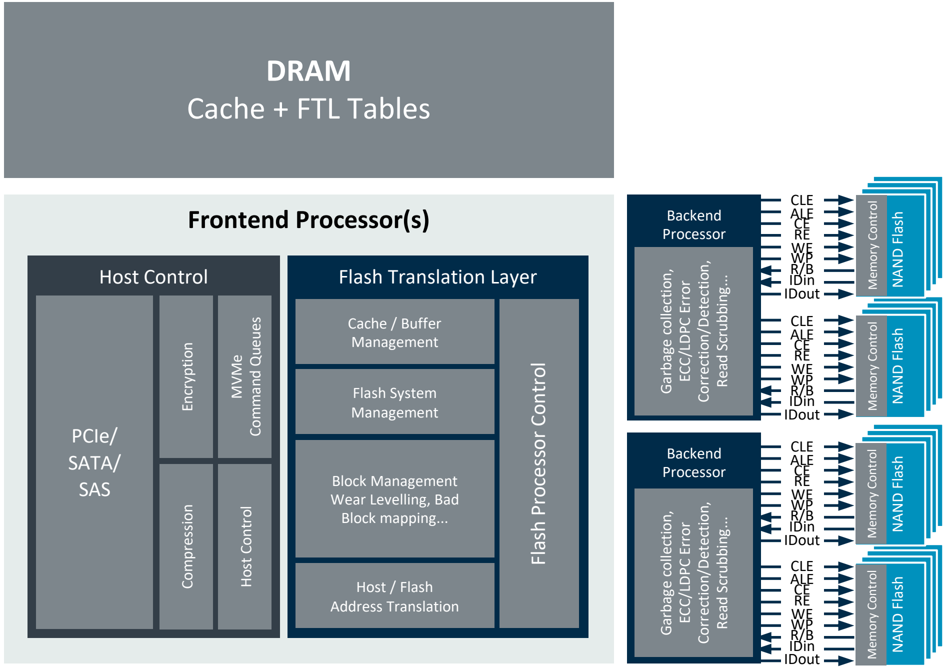
- Accelerators:
  - Hardware accelerators…
  - Encryption, LDPC, Compression…
  - Arm Neon, ML, FPGA…

- **DRAM ~1GB per 1TB of flash**

- Storage: 256GB to 64TB… flash

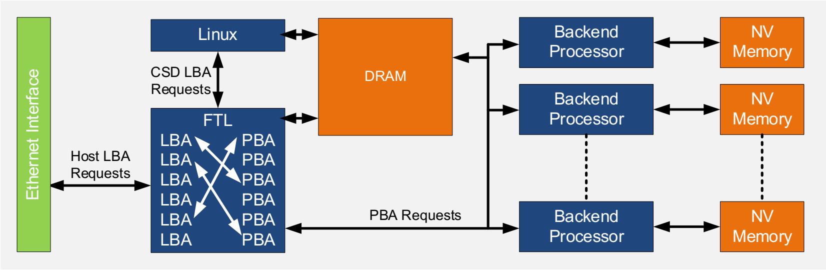
- Interfaces: PCIe/SATA/SAS…

SSD SoC Functionality:



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

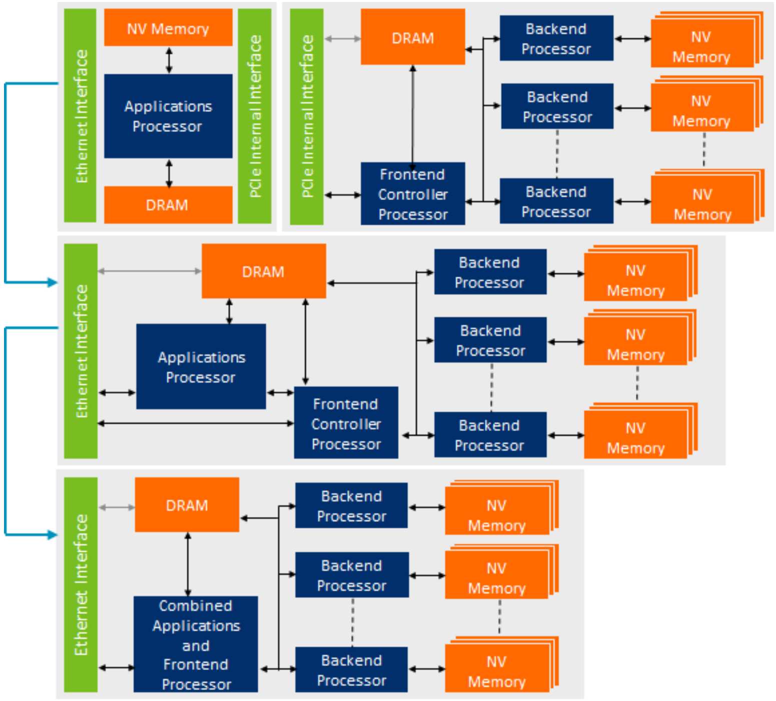


- ➢ Enabling any processing that can be done on the host to be performed on the drive
- ➢ As files are being stored, processing can be performed autonomously
- ➢ Or, once stored the files can be processed in-situ at any point in time

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…

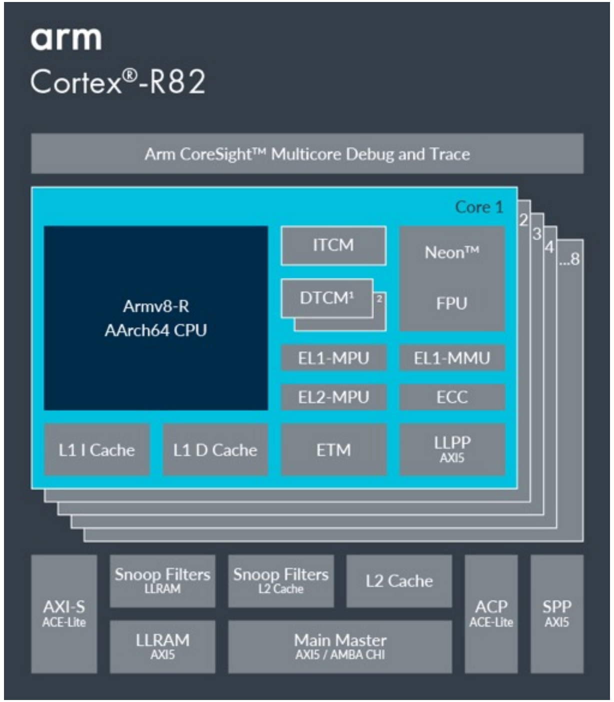
Recommended Minimum System Requirements

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

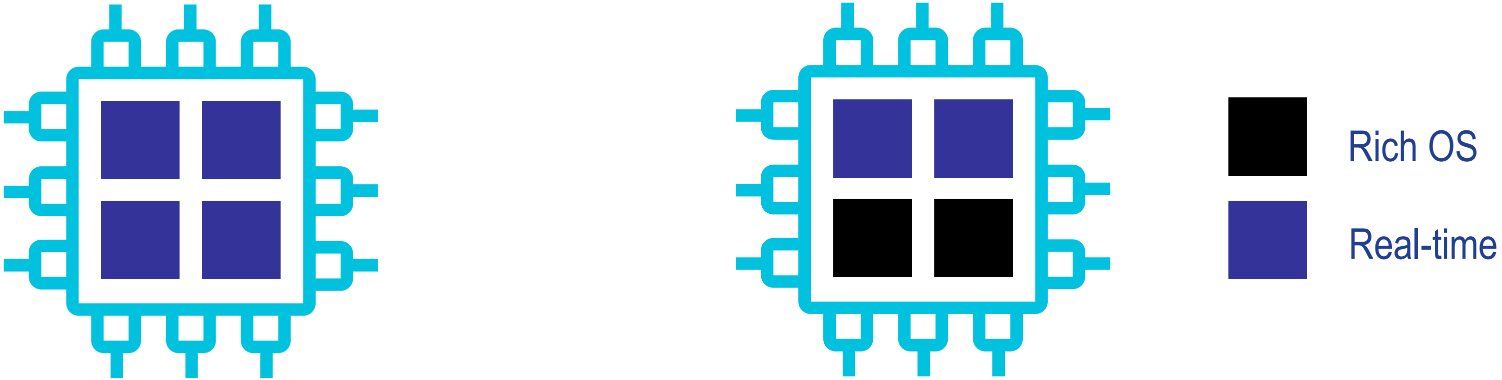
Cortex-R82: Enabling Next-Generation Storage



- 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

Designing Flexible Controllers with Cortex-R82

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

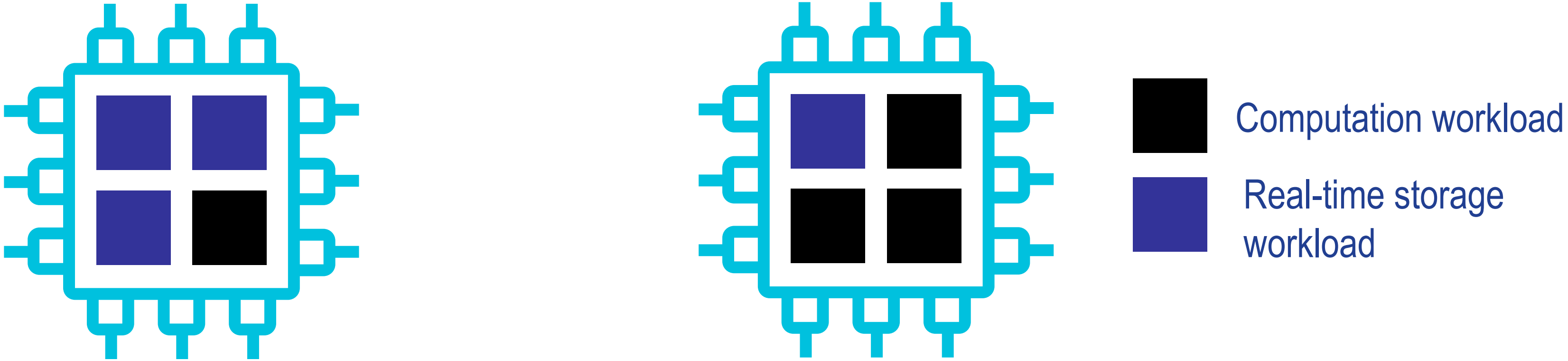


Classic Enterprise Drive

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

CSD with Linux is a Low-cost Edge Server

Compute:
- Arm-based SoC

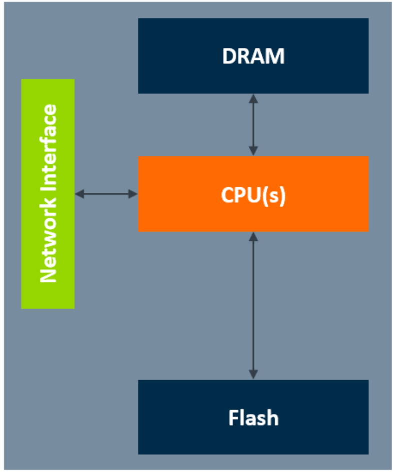
Memory:
- Shared DRAM

Storage:
- Shared Flash

Interface:
- Ethernet…

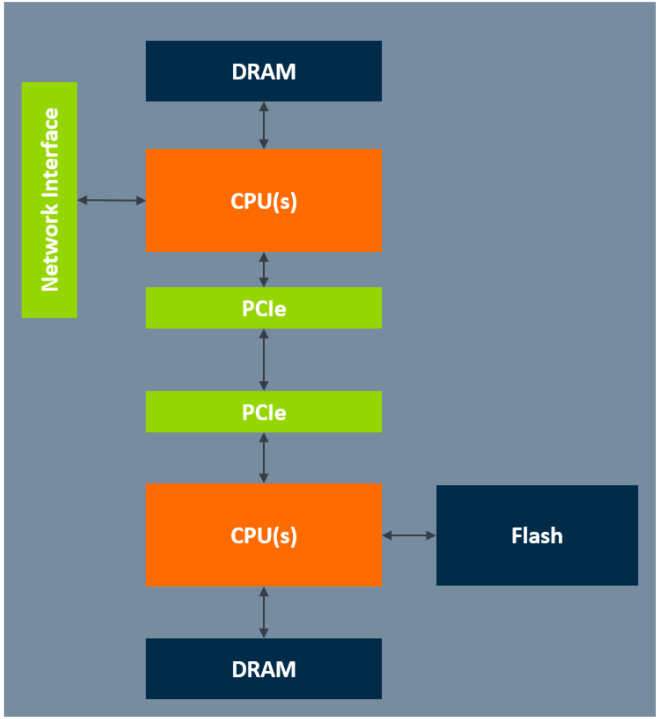
Power:
- Potentially powered over Ethernet (PoE ~15…100W)

SSD Based Edge Server:



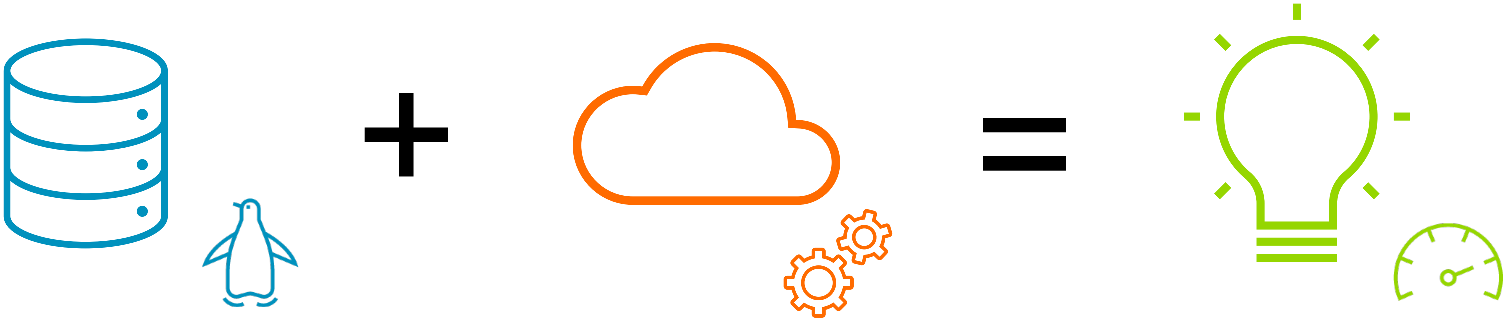
Vs.

Classic Edge Server:



What is driving Linux?
What are the key Workloads?

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

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, cādence, RAPID7, 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

- 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
- ...

Conclusion

Flash Memory Summit

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!

Flash Memory Summit

Everything You Need To Know

For Success