Transforming Storage Controllers with Cost-Effective, On-drive Linux Workloads

Neil Werdmuller
Director of Storage Solutions
Arm
Typical Storage Controller Architectures
Short Overview of Computational Storage
Overview of Computational Storage Drives (CSDs)
Benefits of On-drive Linux and How to Implement it
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:
   Encryption, LDPC, Compression...
   Arm Neon, ML, FPGA...

DRAM ~1GB per 1TB of flash
Storage: 256GB to 64TB... flash

Interfaces: PCIe/SATA/SAS...
What is Computational Storage?

- **Computational Storage**: Computational Storage Services coupled to storage, offloading host processing or reducing data movement.
- **Computational Storage Services (CSS)**: Services that perform computation on data where the service and data are associated with a storage device.
- **Fixed Purpose (FCSS)**: Function that may be configured and used.
- **Programmable: (PCSS)**: CSS that can be programmed to provide one or more CSSs.
- **Computational Storage Device (CSx)**: Drive (CSD), Processor (CSP), Array (CSA).
Moving Data to Compute

Compute waits for data
  Takes time to move data across fabric
  Processing stalled until data is available

Adds latency
  Multiple layers of interface and protocols
  Data copied many times
  Bottlenecks often exist

Consumes bandwidth/power
  Moving data is expensive
  Data copies increase system overall DRAM

Traditional model

1. Request data from storage
2. Move data to compute
3. Compute
4. Move results to storage
Computational Storage Drive (CSD)

Compute happens on the data
Data moved from flash to in-drive DRAM and processed

Lowest possible latency
No additional protocols – just flash to DRAM

Minimum bandwidth/power
Data remains on the drive – only results delivered

Data centric processing
Workloads specific to the computation deployed to the drive

Security
Unencrypted data does not leave the drive

Simplified model with computational storage

1. Request operation
2. Compute
3. Return result
CSD Interfaces

Methods to control and manage a CSD
SNIA developing interoperability standards
SNIA defining NVMe CSD control, advertising, use...

Key interfaces to CSDs
1. PCIe and NVMe: Local server offload
2. NVMe-oF: Offload over fabric e.g. NVMe/TCP
3. On-drive Linux – where the drive is ‘just a server’
Why On-drive Linux?

▪ Huge Open Source developer community

▪ Vast ecosystem of ported and optimised software
  ▪ Protocols such as TCPIP, NVMe…
  ▪ Applications such as databases, machine learning…
  ▪ Standard administration tools
  ▪ Wide range of proven tools for development and debug

▪ Deeply embedded RTOS software more complex to develop/deploy
What is Required to run Linux

Applications processors
- Applications use virtual memory
- Memory Management Unit (MMU)
- Address translation
- Page table manager
- Non-deterministic page table walks
- Non-deterministic interrupt entry
- Program relocation
- Arm Cortex-A portfolio

MMU required for HLOS e.g. Linux

Real-time processors
- Hard real-time
- Deterministic memories
- TCM, LLRAM, Main
- Position-dependent code
- No MMU so cannot run Linux
- Memory Protection Unit (MPU)
- Cortex-R series

Runs bare-metal or RTOS
Evolution to On-drive Linux

Standard PCIe/NVMe SSD

NVMe-oF/TCP
Processes NVMe-oF/TCP commands

Combined NVMe-oF and ‘Just a server’
Frontend processes NVMe-oF commands
TCP/IP packets routed to Linux
Linux apps operate on data in DRAM
Linaro was founded 2010: Together with Arm, Linaro co-maintains the Arm software ecosystem, providing the tools, security and Linux kernel quality needed for a solid base to differentiate on
A few ‘Works on Arm’ partners

www.worksonarm.com
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...

Table 3.2. Recommended Minimum System Requirements

<table>
<thead>
<tr>
<th>Install Type</th>
<th>RAM (minimum)</th>
<th>RAM (recommended)</th>
<th>Hard Drive</th>
</tr>
</thead>
<tbody>
<tr>
<td>No desktop</td>
<td>128 megabytes</td>
<td>512 megabytes</td>
<td>2 gigabytes</td>
</tr>
</tbody>
</table>

Smaller Linux distributions are available

Typical 16TB SSD already has ~16GB DRAM
Low-cost, Low-power Compute

Linux requires an applications processor
Memory Management Unit (MMU) to virtualise 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

...
Arm Flexible Access
Explore a Wide Range of IP Pre-licensing
Defining the next wave of SoC design

Freedom to design and innovate
✓ Access to a wide range of Arm IP
✓ Design rights, tools *and* standard support
✓ Design with any or all IP, at any time

Lower barrier to entry
✓ Pay for the IP used at tape out, not before
✓ Pause, change, stop projects for no fee
✓ Simple, straightforward business terms
Arm Flexible Access
Everything You Need for an SoC

Tools and Models
Faster design and development
Access Arm tooling and models for CPU simulation

IP
Core functionality for any SoC
Arm Cortex CPUs, Mali GPUs, system and security IP, physical IP etc..

Support and Training
Design with confidence
Full support for all included products
Online training for all customer engineers
Arm Flexible Access: What is Included

**Processors**
- Cortex-A53
- Cortex-A35
- Cortex-A34
- Cortex-A32
- Cortex-A7
- Cortex-A5 (UP and MP)
- Cortex-R52
- Cortex-R8
- Cortex-R5
- Cortex-M33
- Cortex-M23
- Cortex-M7
- Cortex-M4
- Cortex-M3
- Cortex-M0+
- Cortex-M0

**Graphics Processors**
- Mali-G52
- Mali-G31
- Also includes Mali DDKs

**CoreLink Interconnect**
- CoreLink NIC-450
- CoreLink NIC-400
- CoreLink CCI-400
- CoreLink CCI-500
- CoreLink CCI-550
- ADB-400 AMBA domain bridge
- XHB-400 AXI-AHB bridge

**System Controllers**
- CoreLink GIC-500
- CoreLink GIC-400
- PL192 VIC
- CoreLink TZC-400
- CoreLink L2C-310
- CoreLink MMU-500
- BP140 Mem. Intf
- BP141 TrustZone Mem. Wrapper

**Peripheral Controllers**
- PL011 UART
- PL022 SPI
- PL031 RTC

**Security IP**
- CryptoCell-312
- CryptoCell-712
- True Random Number Generator

**CoreSight Debug & Trace**
- CoreSight SoC-400
- CoreSight SDC-600
- CoreSight STM-500
- CoreSight System Trace Macrocell
- CoreSight Trace Memory Controller

**Physical IP**
- Artisan physical IP platform, TSMC
- Artisan PIK for Cortex-M33, TSMC 22ULL

**Corstone Foundation IP**
- Corstone-101
- Corstone-201

**IP Products**
- Socrates IP Tooling
- Arm DS-Gold
- Virtual system models (fast & cycle accurate)

Legend: IP Products | Tools & Models

75% of Arm Cortex licenses over past 2 years

*Additional logic IP, standard cell, embedded memory compilers, and interface IP across many foundry nodes also available without a license fee via DesignStart (designstart.arm.com)
When new data is presented to the trained model, large numbers of multiply-add operations are performed using the new data and the model parameters. This process is performed once. In-storage compute can perform this directly on the data, returning results.
For each piece of data used to train the model, **millions of model parameters are adjusted**. The process is repeated many times until the model delivers satisfactory performance. **In-storage compute can perform this directly on the data, without movement.**
Project Trillium: Arm’s ML computing platform

<table>
<thead>
<tr>
<th>Ecosystem</th>
<th>AI/ML Applications, Algorithms and Frameworks</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TensorFlow™, TensorFlowLite™, PyTorch™, ONNX™, Caffe™, Caffe2™, Android™ NNAPI</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Software Products</th>
<th>Software Libraries Optimized for Arm Hardware</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>arm NN, arm COMPUTE LIBRARY, CMSIS-NN</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Hardware Products</th>
<th>Arm Hardware IP for AI/ML</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CPU: arm CORTEX, arm DynamIQ, arm NEOVERSE</td>
</tr>
<tr>
<td></td>
<td>GPU: arm MALI</td>
</tr>
<tr>
<td></td>
<td>NPU: Machine Learning Processor (ML)</td>
</tr>
<tr>
<td></td>
<td>Partner IP: DSPs, FPGAs, Accelerators</td>
</tr>
</tbody>
</table>
Enable faster deployment of ML
• **Any Arm processor with Neon or GPU (OpenCL)**
• Significant performance uplift compared to OSS alternatives (up to 15x)

Optimized low-level functions for Arm
• Most popular ML and CV functions
• Supports common ML frameworks
• Quarterly releases
• CMSIS-NN separately targets Cortex-M

Publicly available now (no fee, MIT license)
developer.arm.com/technologies/compute-library

---

Basic arithmetic, mathematical, and binary operator functions
Color manipulation (conversion, channel extraction, and more)
Convolution filters (Sobel, Gaussian, and more)
Canny Edge, Harris corners, optical flow, and more
Pyramids (such as Laplacians)
HOG (Histogram of Oriented Gradients)
SVM (Support Vector Machines)
H/SGEMM (Half and Single precision General Matrix Multiply)
Convolutional Neural Networks building blocks (Activation, Convolution, Fully connected, Locally connected, Normalization, Pooling, Soft-max)
CSD with Linux is a Low-cost Edge Server

Compute:
- Arm-based SoC

Memory:
- Shared DRAM

Storage:
- Shared Flash

Interface:
- Ethernet...

Vs.

Classic Edge Server:
- CPU(s)
- DRAM
- PCIe
- Network Interface

SSD Based Edge Server:
- CPU(s)
- Flash
- DRAM
- PCIe
- Network Interface
Computational Storage Today

Computational storage *is* happening today
FMS18 there were 4 CSD talks, FMS19 there were 19 CSD talks – full day stream as SDC19
CSDs are available *now* from multiple manufacturers

SNIA CSD standards being developed to deploy/manage workloads
Either over NVMe/PCIe, NVMe/TCP or ethernet or other fabrics
Linux delivers the fastest route for workload development, deployment and management

The drive as ‘just a server’ fully leverages Linux ecosystem
An Enterprise SSD connected via ethernet running Linux *is* a low-cost, low-power server
Computational Storage White Paper

Download our free guide to computational storage on Arm at storage.arm.com

- The benefits computational storage brings to architects, developers and organizations
- How to move from a traditional storage solution to a more intelligent device
- Real-world examples from Arm’s partners
Learn more and download the white paper: storage.arm.com

Contact:
neil.werdmuller@arm.com

Visit: snia.org