Application requirements for wider adoption of CS technology

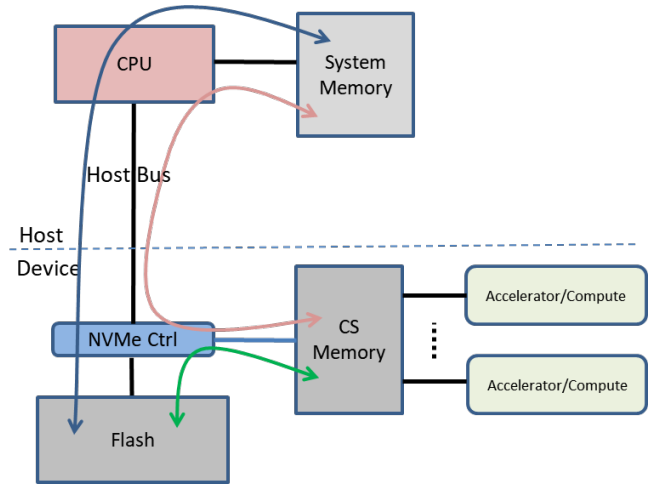
Ramdas Kachare, MSL, Samsung Semiconductor Inc

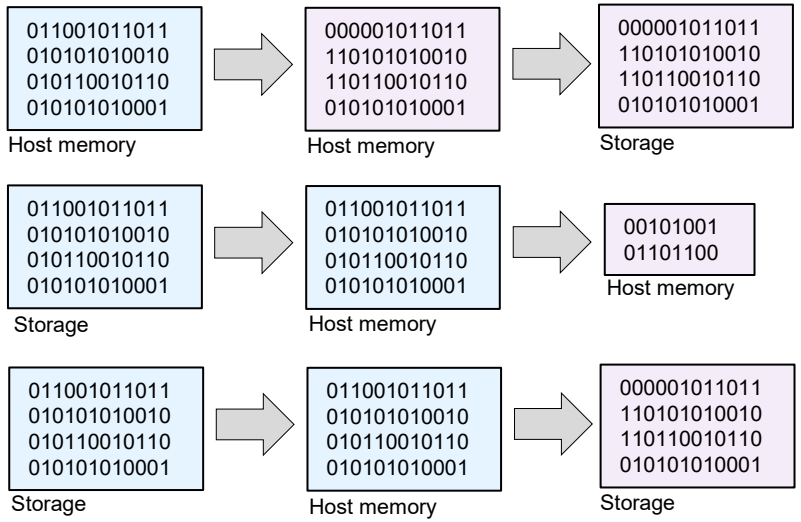
Why Computational Storage

- Data explosion
  - Humongous amount (~60 ZB in 2020), and keeps growing (~20% CAGR, 2020-2025)
- Data driven everything!
  - Improve application productivity using data
- Efficient data processing
  - Compute resources cost – CPU, Memory, Network, energy
  - Performance – latency, throughput, jitter
- Energy consumption
  - Becoming significant portion of overall power consumption

Computational Storage – basic premise

- Three phases for every use case/application
  - Load data for processing
  - Process data
  - Get the results
- Reduce unnecessary data transfers
  - Process data in or closer to storage device when optimal
  - Offload data processing
- Reduce latency of computation as seen by applications
  - Start data processing at the earliest
  - Eliminate data hop
- Moving data to Host for processing is expensive
  - CPU cycles, host bus bandwidth, system memory size/bandwidth
  - Power consumption – processing, cooling





Computational Storage – example applications

- Search in storage
  - Regex, text, objects, files
- Database scan and filter queries
  - Scan heavy
  - Analytics
- Video processing
  - Object detection
  - Transcoding
- Storage services
  - Compression
  - Encryption
  - Media management

Computational Storage – observations and learnings

- Value in near storage compute
  - Efficient utilization of flash bandwidth
  - Reduced system resource costs
  - Lower latency experienced by applications
- One size does not fit all
  - Wide range of use cases, wide range of requirements
  - Value proposition differs for different use cases
- Cost
  - Value gained by user must be higher than the cost incurred including externalities
- Power
  - Value offered must be realizable within user power envelope
- Optimized architectures
  - Multiple architectures
  - Maximize value for different market segments

Application requirements categories

- Development
  - What it would take to develop a CS application?
- Runtime
  - Works fine, meets functional and performance expectations?
- Platform
  - Can it run on current and future Datacenter infrastructure?
- Deployment
  - What are the operational, management needs?

Development requirements

- Ease of development
  - Will affect wide adoption
  - RTL, S/W, or other skills
  - Compute resource type, size, and capabilities
- IP reuse
  - Users able to reuse their existing IP
- Portability
  - Users be able to move their applications and IP from one provider/vendor to another with ease
- Complex Soft IPs
  - Valuable to be able to offload complex soft IPs

Development requirements – more

- Compute resource type
  - Hammer for every problem or Swiss Army knife?
  - FPGA, GPU, TPU, SoC, Fancy Co-processor?
- Easy system stack integration
  - High level abstraction APIs
  - Standardized methods
  - Minimal latency, performance impact
- Ecosystem – Standards, open source
  - Users may have their own drivers
  - OR, they may like Industry Standard drivers, protocols, interfaces
  - Open source user libraries
- Quick GTM
  - Fast feature enhancements
  - Future proof

Runtime requirements

- Host in Control
  - Orchestration, DMA initiation, error/exception handling
  - DMA execution by device
- Host orchestration efficiency
  - Low overhead of Data loading, scheduling, buffer management
  - Can reduce net value of the solution
- Predictability
  - CS device operation must be predictable
- Error and exception handling
  - Graceful handover to host
  - Limited, pre-determined fallout
  - Sufficient diagnostic data for quick debug

Platform requirements

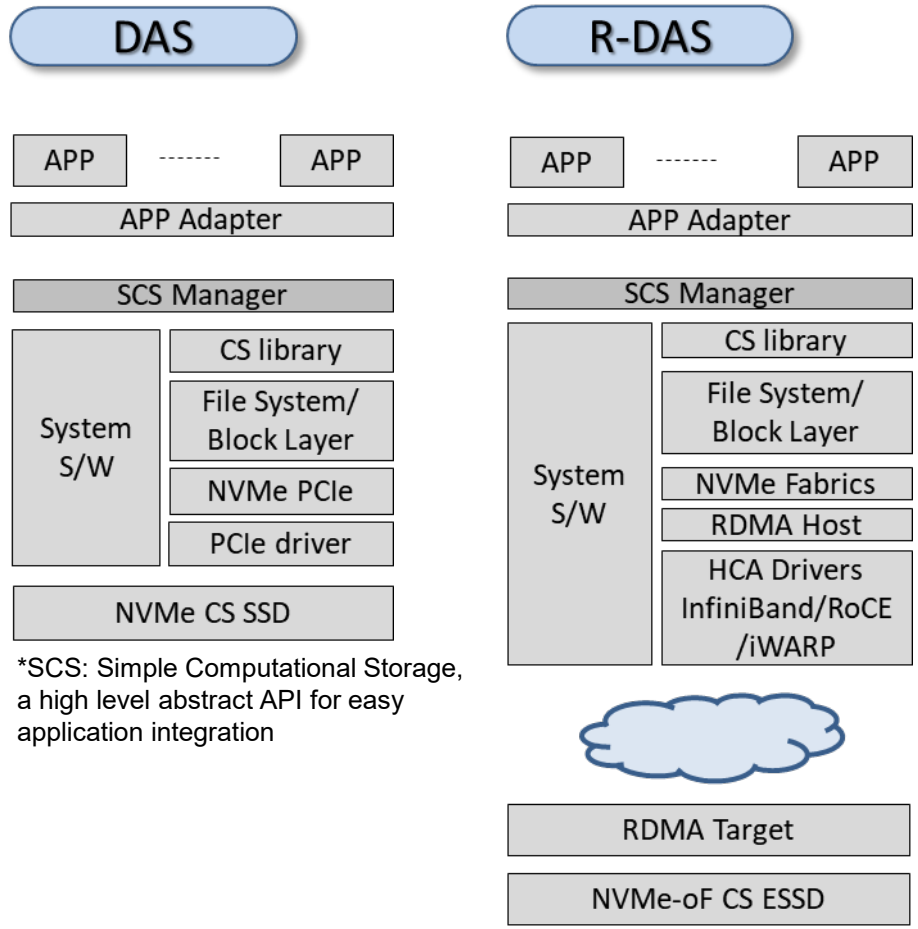
- Server compatibility
  - Dell, HP, Super Micro, IBM, ……
- Right Form factors
  - AIC, U.2, EDSFF etc
- Power
  - Work within given power envelope
  - Smooth throttling
- Thermal
  - Work within given thermal envelope
  - Smooth throttling

Platform requirements – more

- Scalability
  - Multiple CS SSDs in a chassis
  - Support for RAID
- Different OS support requirements
  - Linux, Windows, FreeBSD, ……
- Virtualization support
  - Be able to work in virtualized environments
  - CS memory model

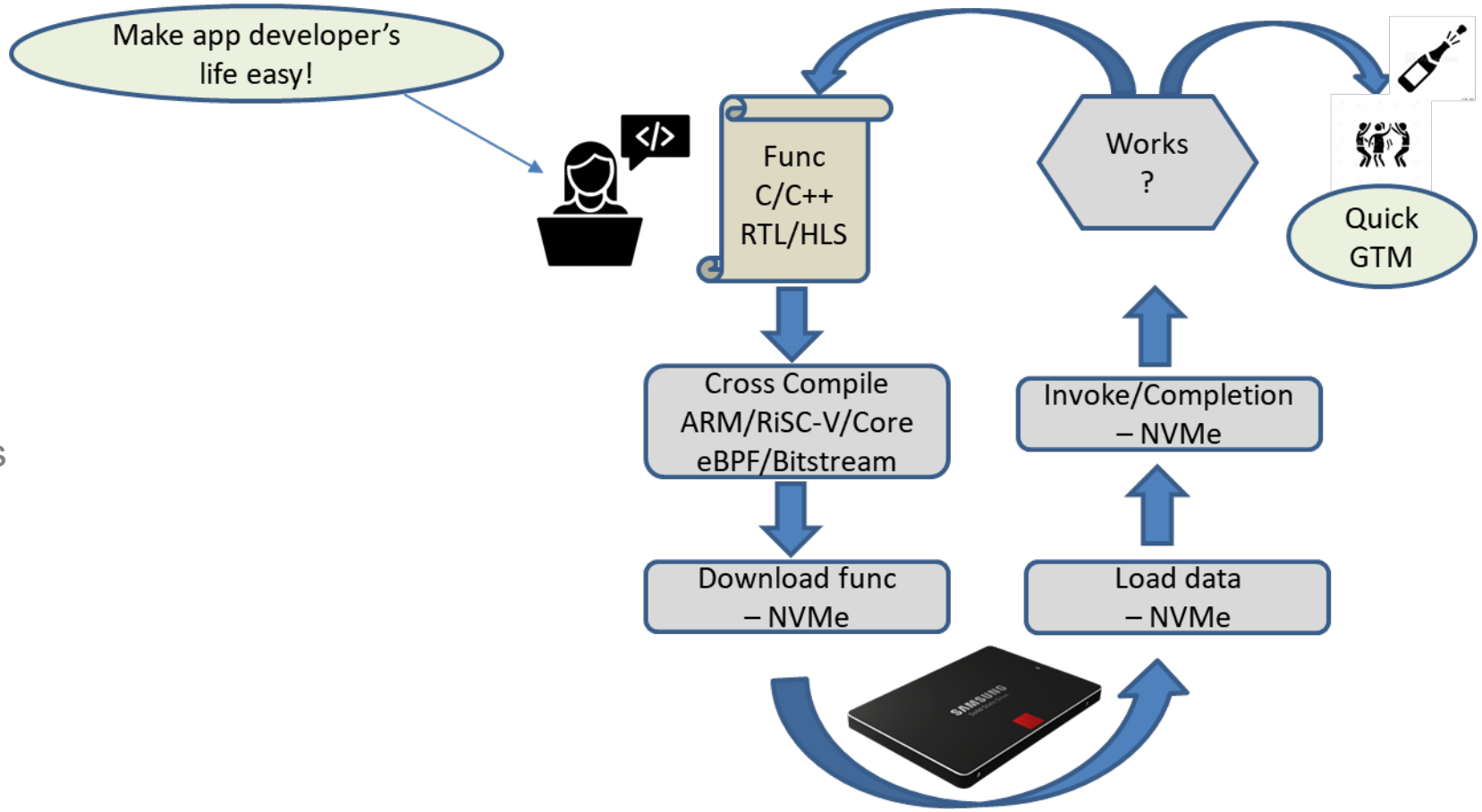
Deployment requirements

- Manageability
  - Discovery, configuration, initialization, monitoring
- Transport Agnostic
  - Be able to work in DAS or RDAS system architectures
- Diagnostic/Upgradability aspects
  - In-field debug, upgrade of compute f/w, shell, kernels etc
- Privacy/Security of offloaded functions
  - Offloaded functions transparent to CS device
  - Offloaded functions not interfering with normal device operation
- Roadmap support
  - New applications? Quick changes to existing applications?



Summary – wide market adoption

- Ease of development
  - Quick and easy iteration
  - Software like development flow
  - Easy debug
- Flexibility of solution
  - Functional app first
  - Transparent performance upgrades
- Standards+ SSD operation
  - Storage IOPs at full performance
  - Concurrent Computational Storage



Call for action

- Collaboration: End users, vendors, service providers, academia
  - Many pieces to the picture!
- Value propositions and validation
  - Where Computational Storage makes sense and where it does not
- System architectures
  - New possibilities to take advantage of Computational Storage technology
- Ecosystem development
  - Tools, solutions
  - Reference designs, examples
  - Standardization
  - Open source

Please take a moment to rate this session.

Your feedback is important to us.

