Conformance, Scalability and Performance Evaluation of NVMe devices

Vinod Eswaraprasad,

Lead Architect, Wipro Technologies

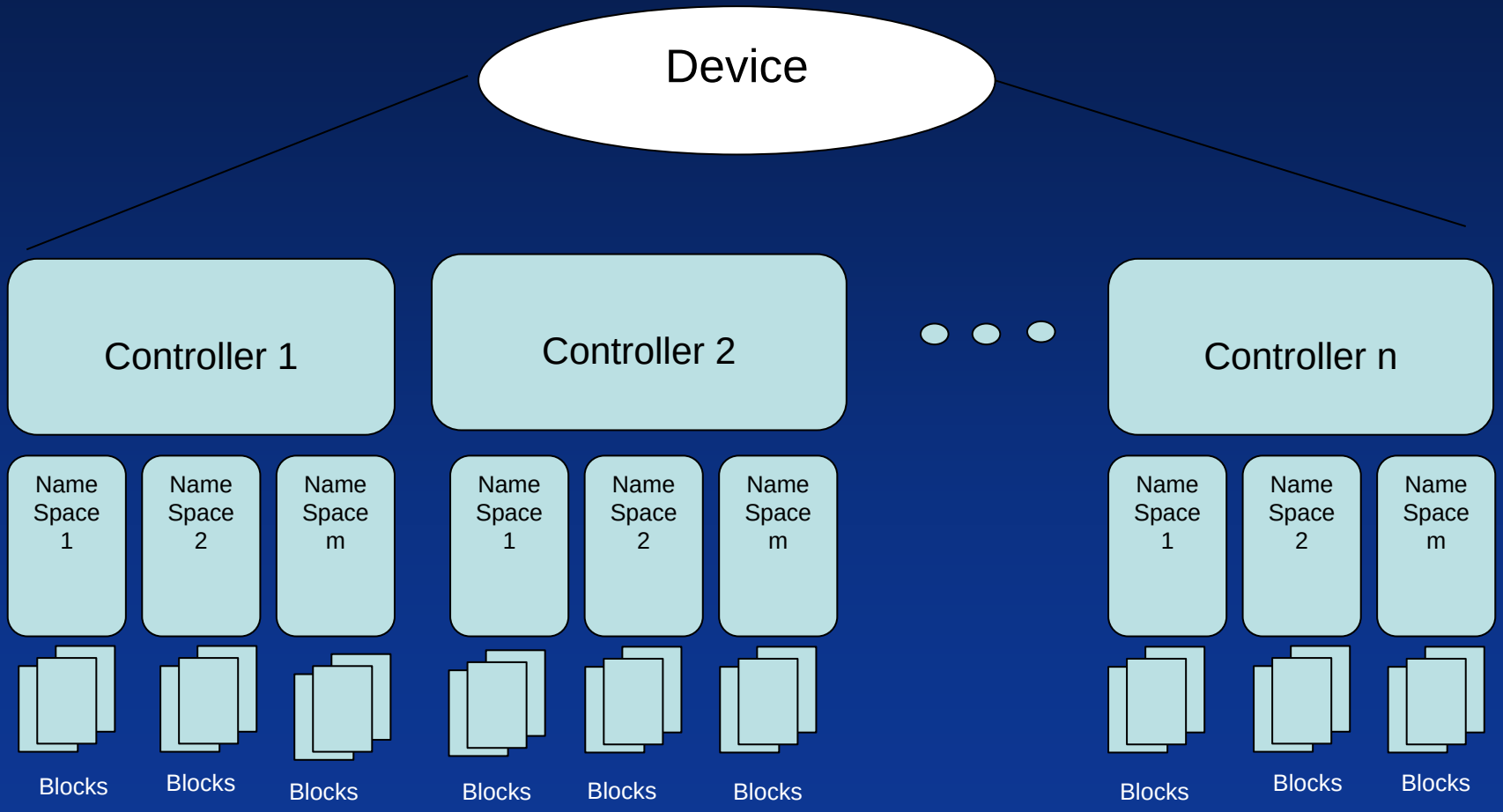
Objective

- NVMe
  - Advanced Host controller interface for PCIe based flash devices - NVMHCI
- Highly scalable and optimized protocol suited for flash storage
- Several devices with vendor specific features
- Common Framework for validation
  - protocol compliance, performance and scalability of specific hardware implementations

NVMe - Evaluation Points

- Compliance
  - NVMe 1.0, NVMe 1.1
  - Spec verification -
  - Advanced features
- Performance
  - Throughput
  - Latency
- Scalability
  - Multi-queue environment
  - Arbitration features, priority handling

NVMe – Device View



Protocol Conformance

- Shorter set of commands
  - 12 Admin commands
  - 7 NVM - data movement commands
- Key NVMe Architectural focus
  - Optimized data path
    - No PIO read, maximum one PIO read for command submission
  - Parallel IO operations
    - 64K queues with 64K command depth
  - Virtualization and E2E data protection support

Protocol Conformance – Focus Area

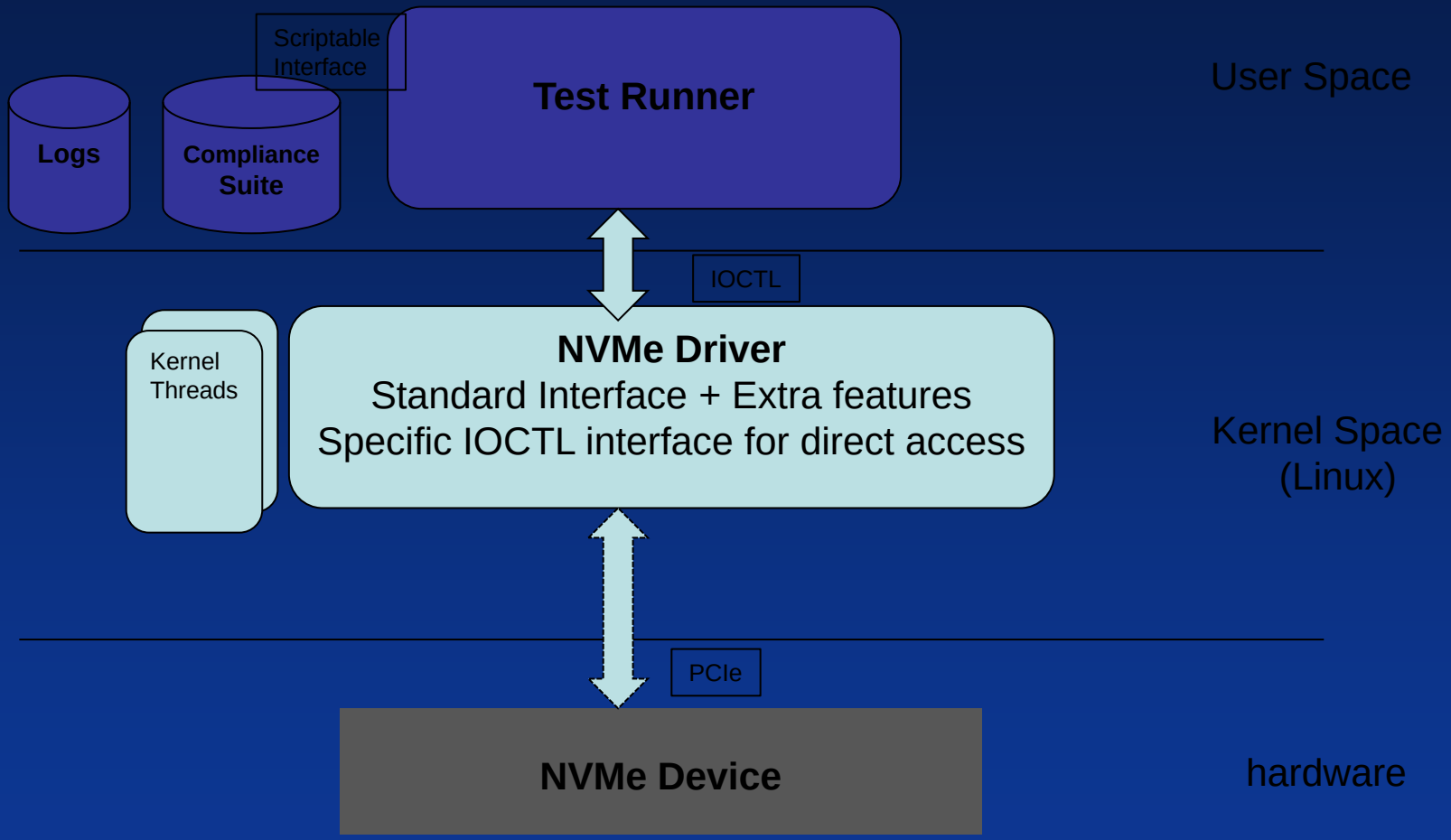
Command Sets - 18	Multiple IO Queues	Multiple Name Spaces
Arbitration Feature	Async. Error Reporting	Error Handling
MSI/MSI-X features	Scatter Gather Support	Atomic Write feature
Metadata buffer support	PCI config, capability	Multi-Path IO
LBA Reservation	Fused Operations	Atomic Write feature

Compliance validation framework

- Standard NVMe driver – with extra features
- Easy enhancement for vendor specific features

- User level test runner
  - Scriptable feature (in progress)
- Same software stack used for
  - Performance and scalability

Compliance validation framework



Direct access to Hardware

- Test runner has more direct access to hardware features, than typical clients.
- The 64 byte command descriptor can be created by the test program send to the driver
- The 16 byte command completion directly made visible to user space
- Enhanced IOCTL interfaces
  - Queue creation - Admin and IO
  - Command submission – Admin and NVM

Queue creation and scalability

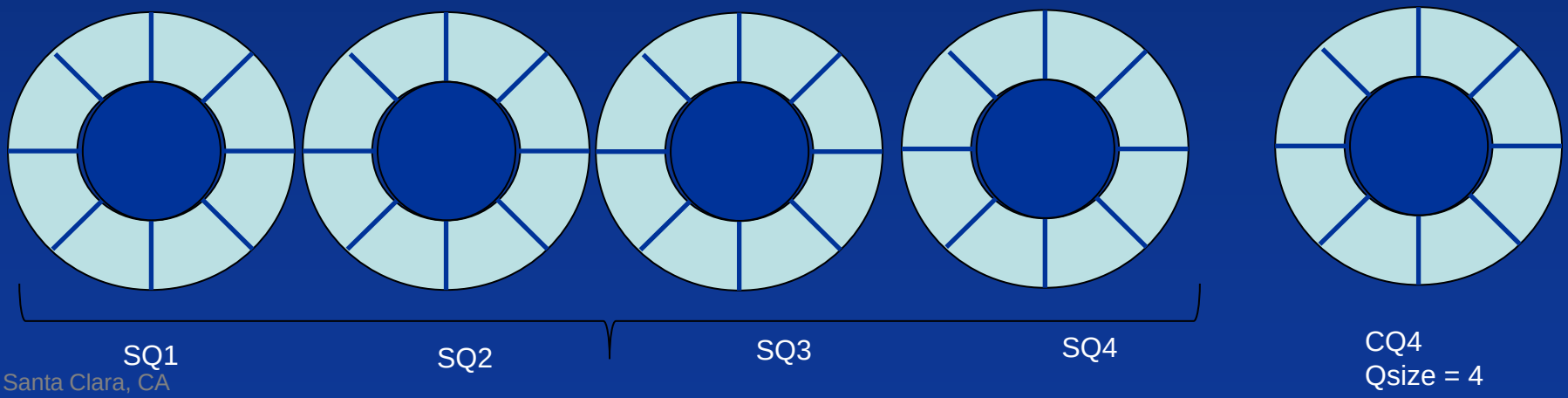
- Parallel operations are possible through multiple IO submission queues
- Test runner can create IO submission queue dynamically
- The Submission queues can have single or shared Completion queue
  - Ability to scale up to 64K
  - User level thread pools to handle Queue usage
- Doorbell register accessibility
  - Functionality of DBSQTail and DBCQHead boundaries

Validate arbitration mechanism

- 3 types of arbitration mechanisms
  - Round Robin
  - Priority based – Weighed round robin
  - Vendor specific
- Arbitration burst
  - Select “n” commands from a selected queue
  - Performance analysis
- How do we setup commands in SQ for deterministic execution sequence

Arbitration Mechanism – determinism

- Multiple IO Submission Queues and single IO completion queue
- Fill the completion queue with initial 4 IO commands, do not process completion queue
- This makes controller stall – do not process SQ commands
- Now fill the SQs, when controller is idle…

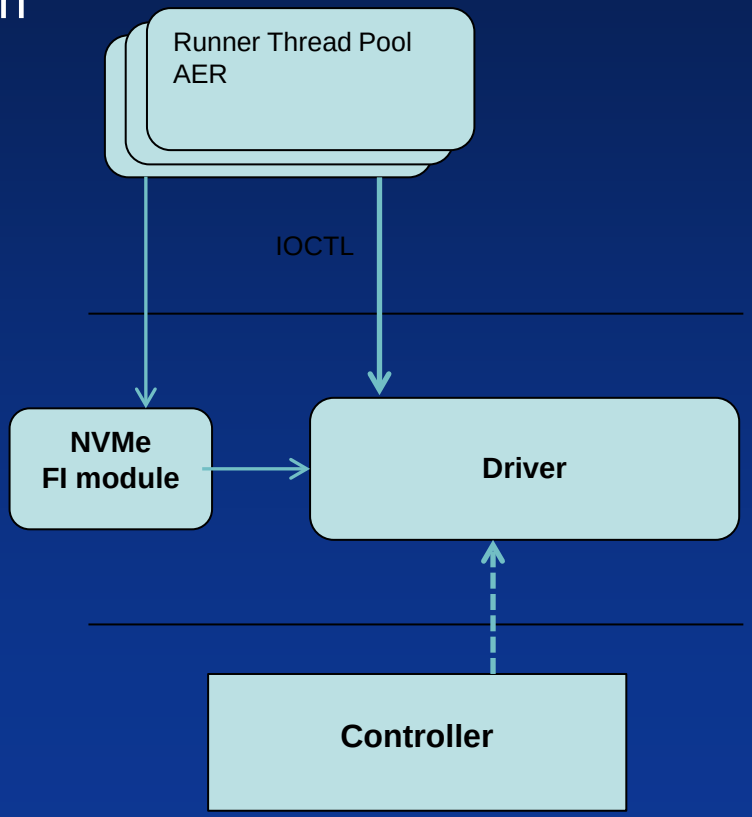


Asynchronous Event Reporting

- Status, error and health information, at the time of occurrence
- Software posts one or more AER command
  - Selectable – Error Status, SMART, Command specific
- The request is completed when one of the selected event occurs
  - The event is masked till host clears the event
- Software clears the event by reading log pages associated with the event
- Controller queues events in case of no outstanding AER
  - Threshold – count and time?

AER – Validation scenario

- User level AER thread pool
  - Threads post AER command, and then gets blocked
  - Special IOCTL to block the thread
  - Map of AER command ID to the waiting thread
  - When the command completes the thread is woken up
  - Thread retrieves the error info by issuing get_log_pages for the event
- Special fault inject IOCTL to generate errors asynchronously
  - Write Uncorrectable blocks
  - Event queuing and clearing validation



SGL, PRP – Buffer options

- Rich buffer management options
- PRP
  - PRP1, PRP2, PRPList
  - Boundary conditions
  - Check for allowed and dis-allowed buffer discontinuities
- SGL
  - SGL segments
- Metadata
- Validate buffer Metadata buffer management

SGL and PRP

- Source data created in chunks
  - Multiple virtual buffers - configurable
  - Map them to the SGL/PRP
    - User level configuration option
  - IOCTL to submit the Command to SQ
- Options for virtual data buffer alignment
  - Cause all possible discontinuities
  - Trigger possible PRP/PRP list combinations
    - Page size, and alignment options
  - Validate logical data correctness
- Use compare command to validate the buffers

Performance

- IOPS
  - Commands submitted and completed per second
  - User thread pool (thread per core) to create distributed IOs
- Throughput
  - MBs moved per second
  - Sequential and Random – Reads/Writes
  - Different IO buffer size
  - DSM options
- Latency
  - Service time of each command in steady state
  - Relation to latency during multiple active IO Queues

Scalability Analysis

- Single IO queues vs. Multiple QIO
  - Command completion latency
  - Histogram for Individual command completion times
  - Configurable number of IO submission queues
    - From 1 to CAP.MQES
- Arbitration and Arbitration Burst Impact
  - Throughput analysis for various AB values
- Load specific interrupt coalescing effect
  - Optimal TIME/THR

Summary

- Scriptable conformance tests
- Easily Customizable for vendor specific features
- Support for NVMe 1.1 commands
- Extensible driver
- Performance analysis framework

Thank You