Navigating SSD Tail Latency: Profiling, Analysis, and Optimization

Guanying Wu (gwu@siliconmotion.com)

Principal Engineer

Silicon Motion Technology Corp.

Legal Notice and Disclaimer

- The content of this document including, but not limited to, concepts, ideas, figures and architectures is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Silicon Motion Inc. and its affiliates. Silicon Motion Inc. assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this document.
- Nothing in these materials is an offer to sell any of the components or devices referenced herein.
- Silicon Motion Inc. may have patents, patent applications, trademarks, copyrights, or other intellectual property rights covering subject matter in this document. Except as expressly provided in any written license agreement from Silicon Motion, Inc., the furnishing of this document does not give you any license to these patents, trademarks, copyrights, or other intellectual property.
-
- Silicon Motion, the Silicon Motion logo, MonTitan, the MonTitan logo are trademarks or registered trademarks of Silicon Motion Inc.

Problem Statement

- Solid-state drives (SSDs) grapple with significant tail latency issues.
  - NAND tRead ranges from 40~100 us, but the tail latency (e.g., 6 9's) scales up to the **millisecond** range.
- Major factors:
  - Garbage collection, wear leveling, NAND ECC, NAND contention,
  - Firmware overhead, and task scheduling policies.
- Focus of Presentation:
  - QoS Simulation model for identifying major contributors of long tail latency.
  - Optimization techniques for tuning SSD QoS.

Simulation Model Design

Limitations of Existing Tools

Tools	Limitations
Behavioral model	Lacks timing for IO latency analysis
Cycle-accurate/FPGA-emulation model	Too detailed and slow to simulate; algorithm development as complex as FW development; often results in trial-and-error
Analytical model	Effective for data-path performance and power estimation, but struggles with control path
ASIC	Gets mired in complexity and details of HW/FW implementations, and struggles to identify major factors contributing to tail latency without causing significant errors

Simulation Model: Design Goals

Our simulation model has been designed with specific goals in mind, aiming to:

- Facilitate fast QoS algorithm development.
- Establish a Queuing network model with realistic timing.
  - Striking a balance between no-timing and excessive timing.
- Ensure Completeness, including key components that impact timing:
  - Firmware (FW).
  - IO processing.
  - Background operations: Garbage Collection (GC), Wear Leveling (WL), NAND algorithms, System info maintenance.
  - CPU, DDR, NAND, HW engines.

SiliconMotion Simulation Model: Design Method

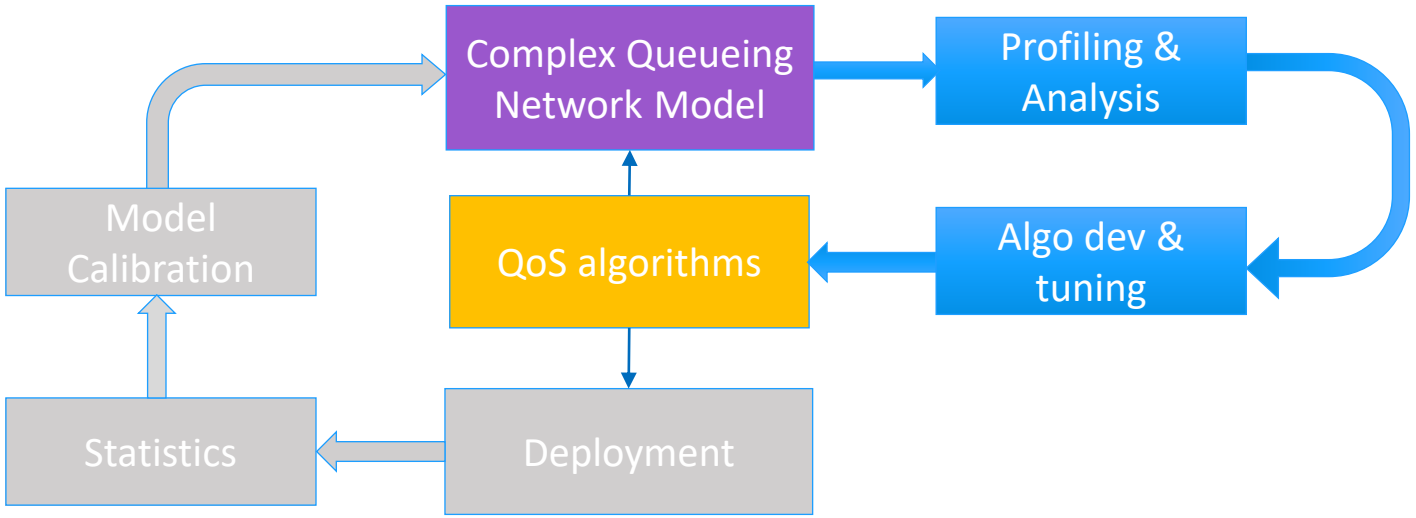
- White Box approach:
  - Status of components and queues are fully visible and collectible.
  - Identify the bottleneck.
  - Log the entire lifetime events of any request.
  - Understand latency distributions.
  - Unearth the root cause of anomalies.
- Easy error injection.
- Enable Fast development:
  - Using modern C++ and SystemC for quick algorithm exploration.

Model Architecture

- **Model Architecture**
  - Timing annotated components interconnected through queues.
  - A complex queuing network: buses, CPUs, DDRs, SRAMs, NANDs, data-path processing engines, control path accelerators, etc.
  - Queues carry transactions, similar to Transaction-Level Modeling (TLM) non-blocking.
- **Profiling**
  - Each host command is accompanied by a state transition log and a timestamp.
  - Queuing Delay Analysis: On enqueuing, record the number of customers ahead.
- **Visualization Utilities**
  - Integrated tools for data visualization and analysis.

QoS Study Methodology

Methodology for QoS Study



- Profiling
  - Run the system
  - Collect queue statistics: utilization rate, service time, queue length
- Analysis
  - Locate the hotspots: high utilization components
  - Root cause the abnormal service time
- Algo dev & tuning
  - GC, WL, NAND Mgmt.
  - IO Traffic Scheduling
  - Power Shaping
  - Performance Shaping
- Model Calibration
  - Deploy in real or cycle accurate environment.
  - Calibrate the model

Methodology for QoS Study

Case study: random read QoS

1. Algorithm Proposal
   - Focus on firmware-centered algorithm development
     - Balancing read vs write/erase
     - Prioritizing host vs garbage collection
     - Managing normal read vs retries
     - Handling quick retries vs slow retries
   - The key is fine-tuning parameters to reach an optimal balance
2. Correctness Verification
   - Does the algorithm halt as expected?
   - Verification Methods
     - extensive testing
     - formal verification
3. Performance Validation
   - How much performance improvement is achieved?

SiliconMotion

Flash Memory Summit

Unveiling SSD Tail Latency: Causes and Mitigation Strategies

Exploring the Anatomy of SSD Tail Latency

- The Primary Contributing Factors to Tail Latency:
  - Service Time Variability
    - Inconsistent servicing times for requests leads to spikes in latency.
  - Queuing Delay
    - Delays due to queuing contribute significantly to tail latency.
- Both elements demand our attention for effective mitigation.

Tail Latency: Service Time Variability

- Major Sources of Service Time Variability
  - LDPC error floor
  - Voltage Threshold (Vth) misalignment.
- Mitigation Strategies
  - Triger soft-decoding before entering LDPC error floor
  - Identify the Correct Voltage Threshold (Vth) by background tracking

Tail Latency: Queuing Delays

Sources of Queuing Delays:

- High Utilization Rate:
  - Increased chance of contention
  - Inherent collisions due to randomness
- Head-of-Line Blocking:
  - During Program/Erase cycles
  - During Read Retries
- Imbalanced Utilization:
  - Example: Two parallel processing units available, but only one is utilized

SiliconMotion Tail Latency: Queuing Delay Mitigation

- Priority Management:
  - Prioritize foreground tasks.
- Preemption:
  - Enable preemption of background tasks. Methods include:
    - Cancellation: Similar to Program/Erase cancellation during PLP.
    - Suspension: Implementing Program/Erase suspension.
    - Slicing: Break long-running firmware tasks into shorter steps.
- Preventing Background Starvation:
  - Reserve adequate bandwidth for background tasks to complete.
  - Design policies to ensure timely completion of background tasks.
- Overprovisioning:
  - Allow some level of utilization sacrifice.
  - Reserve resources for foreground tasks.
- Life-Latency Trade-off:
  - Perform garbage collection during idle times.
  - Reserve bandwidth for garbage collection during busy periods.
- Bandwidth-Latency Trade-off:
  - Utilize some NAND bandwidth for Vth tracking to lower Read Retry rate.

Summary

- Developed a QoS Simulation Model that offers realistic timing annotation, complete profiling, and quick algorithm exploration capabilities.
- Proposed a comprehensive methodology for QoS study, including profiling, analysis, algorithm development, and model calibration.
- Analyzed sources of tail latency - Service Time Variability (focusing on read retry) and Queuing Delays. Proposed several mitigation ideas to combat tail latency, centered around priority management, preemption, overprovisioning, and strategic trade-offs.

Our work offers a systematic and effective approach to manage SSD tail latency, ultimately enhancing SSD performance and usability.

SiliconMotion

Flash Memory Summit

Meet us at booth #315

Scan to learn more!