A Flash Scheduling Strategy for Current Capping in Multi-Power-Mode Mode SSDs

Li-Pin Chang, Chia-Hsiang Cheng, and Kai-Hsiang Lin

Department of Computer Science
National Chiao-Tung University, Taiwan

Presented at ASPDAC 2017, Japan
Flash Storage Devices

• High performance, low power consumption, low heat dissipation
  • Ideal storage solution for various computer systems

• Enterprise-scale servers
  • Non-volatile Memory Express (NVMe)

• Personal computers
  • Solid State Disks (SSDs)

• Smart devices
  • Embedded Multimedia Cards (eMMCs)
  • Universal Flash Storage (UFS)
Interface Power Modes

- Embedded storage interfaces may change the peak current limit for power saving and budgeting
- eMMC 5.1
  - 200 mA~900 mA
- USB 3.0
  - High-power mode: 900 mA
  - Low-power mode: 150 mA
- External storage device can be powered by the interface or an AC adaptor
Interface Bandwidth vs. Flash Parallelism

• High storage interface bandwidth
  • A PCIe lane supports up to 1GB/s
    • An NVMe SSD employ multiple PCIe lanes
  • SATA-3 supports up to 600 MB/s
  • eMMC 5.1 supports up to 400 MB/s

• The read throughput of a typical flash memory bank is only 40 MB/s [2]
  • Exploiting flash memory parallelism to fully utilize the bus/interface bandwidth
  • An NVMe SSD has more than 16*4 flash memory banks
  • A recently announced eMMC has 2*4 banks [3]
Peak Current vs. Flash Parallelism

• A page read consumes up to 50 mA on a typical flash memory bank

• In the eMMC, 2*4=8 concurrent page reads incur a peak current 50*8=400 mA

• Much higher than the default power class 200 mA

• Current overloading will cause unexpected voltage dropping
Current-Capping in SSDs

- The max. current supply of a Thunderbolt port is 550 mA
- Switching an SSD from the AC-powered mode to port-powered mode
  - Peak current 760 mA to 544 mA
  - Read thru. 590 MB/s to 315 MB/s
  - Peak curr. 30% ↓ but thru. 47% ↓
- If current capping is disabled
  - The host loses connection to the SSD!
  - Data loss!
Problem Definition

• **Current capping**
  - control the peak current under a limit (cap) at all times

• Goals
  - Maximize SSD internal parallelism
  - Avoid peak current overloading

• In this study, a firmware approach is proposed
Ideas and Challenges

• Observations
  • The current varies during a flash operation
  • The current of different types of operations differs a lot

• Basic idea
  • To adaptively schedule flash operations to suppress the peak current under a cap at all times

• Challenges
  • Need to build accurate models of flash current usage
  • Too slow to check the total current at every time point
  • Flash current usage is subject to the aging process
Current Measurement

• Jasmine OpenSSD platform
• Measure flash current in terms of bank

(a) (b) = measurement pins

SDRAM
SSD controller
channel
plane
chip
bank
bus

DAQ
1 Ohm resistor
DC

OpenSSD
Page Read Current Usage

- flash busy $\rightarrow$ Bus transfer
- Both bus and flash incur tail currents
- Waveform of LSB and MSB pages are similar
Page Write and Block Erase

- Writing MSB pages consumes more energy than writing LSB pages.
- Erasing a block does not involve bus transfer.

![Graphs showing current vs. time for LSB page write, MSB page write, and block erase.](image)
Corner-Based Flash Current Models
• The current usage of each type of flash operation can be modeled using a series of linear functions
  • A corner is the junction of two linear functions \((\text{us}, \text{mA})\)
  • Current model of \(R = \{(0,0), (265,50), (1450,0)\}\)
  • A corner represents a local maximum or minimal of current usage
Model-Based Current Estimation

- Two simultaneous page reads R1 and R2 in the same channel

Bus contention splits the current usage of R2

R1 has no bus tail current
Current-Capping Flash Scheduling

- Between FTL and low-level flash routines
- Decide the actual starting time of bank operations

- Host read and write requests
- Logical page reads
- Flash translation layer
- Flash operations (read, write, erase)
- Current capping layer
- Bank scheduler
- Current models
- Flash operations with starting times
- Flash I/O routines
Flash Scheduling

• Bank status
  • Ready $\rightarrow$ Selected $\rightarrow$ Scheduled $\rightarrow$ Busy $\rightarrow$ Ready

• Bank selection
  • Channel with fewest busy banks to reduce bus contention
  • Favoring read over other operations for better read response

• Bank scheduling
  • A corner list $L=$\{\} initially
  • Scheduling an operation = adding new corners to $L$
  • No corners should have current value $>$ current cap
Flash Scheduling in Action

- Current cap = 100 mA
- R1, LSB W1 can safely start at time 0
Flash Scheduling in Action

- **R2** is scheduled for time 990 to avoid cap
Flash Aging

- Aging barely affect peak current
- Aging marginally speeds up MSB page write, but significantly slows down block erase
  - Erase current model evolves as flash ages
Experimental Setup

- **Workloads**

<table>
<thead>
<tr>
<th>Workload</th>
<th>Volume size</th>
<th>Read (%)</th>
<th>Avg. req. size</th>
<th>Seq (%)</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>PC</td>
<td>40 GB</td>
<td>31.20%</td>
<td>11.6 KB</td>
<td>22.80%</td>
<td>Mixed</td>
</tr>
<tr>
<td>TBL</td>
<td>30 GB</td>
<td>4.10%</td>
<td>7.95 KB</td>
<td>17.50%</td>
<td>Mixed</td>
</tr>
<tr>
<td>NB</td>
<td>20 GB</td>
<td>54.80%</td>
<td>18.8 KB</td>
<td>33.90%</td>
<td>Mixed</td>
</tr>
<tr>
<td>SVR1</td>
<td>400 GB</td>
<td>48.70%</td>
<td>11.1 KB</td>
<td>7.10%</td>
<td>Random</td>
</tr>
<tr>
<td>SVR2</td>
<td>400 GB</td>
<td>11.50%</td>
<td>19.2 KB</td>
<td>75.50%</td>
<td>Sequential</td>
</tr>
<tr>
<td>TPCC</td>
<td>180 GB</td>
<td>66.70%</td>
<td>8.1 KB</td>
<td>0.01%</td>
<td>Random</td>
</tr>
</tbody>
</table>

- **Flash organizations**

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bank organization</td>
<td>4 ch*4 bank</td>
<td>Overprovision</td>
<td>10%</td>
</tr>
<tr>
<td>Page, block size</td>
<td>32 KB, 4 MB</td>
<td>Native cmd. queue</td>
<td>32 requests</td>
</tr>
<tr>
<td>Write buffer size</td>
<td>4 MB</td>
<td>Current caps</td>
<td>200 ~ 1200 mA</td>
</tr>
</tbody>
</table>
Experimental Setup

- Dynamic current capping (DCC)
  - Corner-based model
- Idle insertion \(^{[4]}\)
  - Insert an idle between each request
- Count-base \(^{[1]}\)
  - Square waveforms

<table>
<thead>
<tr>
<th>Cap (mA)</th>
<th>Idle ((\mu)s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1200</td>
<td>40</td>
</tr>
<tr>
<td>1000</td>
<td>110</td>
</tr>
<tr>
<td>800</td>
<td>169</td>
</tr>
<tr>
<td>600</td>
<td>236</td>
</tr>
<tr>
<td>400</td>
<td>357</td>
</tr>
<tr>
<td>200</td>
<td>727</td>
</tr>
</tbody>
</table>
Results

• Response
  • The advantage become significant when capping lower 600 mA
  • The response time of TBL is shorter than NB due to the concentration of write requests
Results

• Throughput
  • DCC reached the highest throughput from capping higher than 800 mA
  • Throughput of DCC higher than other algorithm under any cap current
Results

• Scheduling overhead
  • TB & SVR with frequent high-current write request restrict the flash parallelism
  • Preserving less than 250 us to schedule a single request
Results

• Implement on OpenSSD Jasmine platform and successfully limit the current under 200 mA
• Throughput on OpenSSD is close to simulator

[Graph showing current and throughput over time]
Conclusion

- Rich flash parallelism may incur high instantaneous peak current
  - Performance vs. Power budgeting/throttling
- This study introduces a firmware-level scheduling strategy for current capping
  - Realistic current models of flash operations
  - Fast corner-based flash scheduling
  - A proof-of-concept based on OpenSSD
- Future work
  - Evolving current models with consideration of flash aging and process variation
Thanks for listening
Q&A

