VSSIM: Virtual Machine based SSD Simulator

Jinsoo Yoo, Youjip Won, Joongwoo Hwang, Sooyong Kang, Jongmoo Choi, Sungroh Yoon and Jaehyuk Cha

Dept. of Electronics and Computer Engineering
Hanyang University
Contents

- Introduction

- VSSIM (Virtual Machine based SSD Simulator)
  - The architecture of VSSIM
  - Algorithm
  - Validation

- Case Studies

- Conclusion

- Demo
SSD Market is Expanding

It is important to **design an SSD Component** for future SSD performance requirements.
Design Components of SSD

- **SSD**
  - # of Channels and Ways
  - # of Channels
  - # of Ways
  - Flash memory (latency, size, …)

- **Flash controller**
  - Flash 0
  - Flash 1
  - Flash 2
  - Flash 3
  - Flash 4
  - Flash 5
  - Flash 6
  - Flash 7
  - Flash 8
  - Flash 9
  - Flash 10
  - Flash 11
  - Flash 12
  - Flash 13
  - Flash 14
  - Flash 15

- **Firmware**
- **DRAM**
- **CPU**
- **SATA / PCIe**
- **HOST**
- **Address Mapping**
- **Wear-Leveling**
- **Garbage Collection**

Host performance (tpmC, Bandwidth, IOPS, …)
SSD Design of S/W & H/W

- # of Channel / Way
- Mapping Algorithm
- Garbage Collection
- Wear-leveling

H/W Prototyping

Pros.
- Accurate

Cons.
- Time Consuming
- Expensive
- Inflexible

S/W based Simulation

Pros.
- Time saving
- Cost efficient
- Flexible

Cons.
- Less accurate
Trace-driven Simulator

Server, PC, and etc.

- read/write
- sector number
- length
- time info

Target Machine Configuration

Trace-Driven Simulator

Simulation Result

- DISKSIM for SSD [Agrwawal08]
- FLASHSIM [Kim09]
- CPS-SIM [Lee09]
Issues in Trace-driven Simulator: Address Space Rescaling

Server, Mass Storage, etc.

Disintegrates the access locality in the trace
Think Time Rescaling for closed loop system

- //Trace [Mesnier07]
- Buttress [Anderson04]
Issues in Trace-driven Simulator: Accurate Replay

Original Trace

- IO Arrivals
- Think Times
- IO Completions

Simulation Target

- IO Arrivals
- Think Times
- IO Completions

Difficulty in accurate replay
Hardware-based Emulator

Pros.
- Accurate test result

Cons.

Cannot Extend
- # of Flash memories
- # of Channels
- # of Ways

Cannot Change
- NAND IO latency
- NAND page size

Remake is needed to use a new NAND flash memory

Inflexible
# Functionality of SSD Simulators

<table>
<thead>
<tr>
<th>Simulator</th>
<th>Type</th>
<th>Host Perf.</th>
<th>Firmware Change</th>
<th># of Channels/Ways</th>
<th>NAND Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>DiSKSIM [Agrawal08]</td>
<td>Trace-driven Offline</td>
<td>No</td>
<td>O</td>
<td>O</td>
<td>O</td>
</tr>
<tr>
<td>NANDSIM</td>
<td>Kernel Code   Online</td>
<td>Yes</td>
<td>X</td>
<td>X</td>
<td>X</td>
</tr>
<tr>
<td>OpenSSD [Lee11]</td>
<td>Hardware-base Online</td>
<td>Yes</td>
<td>O</td>
<td>△</td>
<td>X</td>
</tr>
<tr>
<td>BlueSSD [Lee10]</td>
<td>Hardware-base Online</td>
<td>No</td>
<td>O</td>
<td>△</td>
<td>X</td>
</tr>
</tbody>
</table>

- **O**: Supported
- **△**: Supported with a H/W Limitation
- **X**: Not supported

Can an SSD Simulator support **all these functionality**?
Requirements for a new SSD simulator

- Does not need to use a trace
- Can measure Host performance in real-time
- Can simulate various SSD architecture:
  - # of channel/way
  - NAND flash memory
  - Page size
  - ...
- Easily changes firmware:
  - Flash Translation Layer (FTL)
  - Write Buffer / Map Cache / TRIM
  - ...
Virtual Machine based SSD Simulator: VSSIM

Application

OS

QEMU

FTL Module

SSD Monitor

IO Emulator Module

Latency Manager

Config. File

VSSIM
WRITE (sector number, sector length)

FTL Module

Mapping Table Update

SSD Monitor

WRITE (sector number, sector length)

Write to RAMDISK

FTL Module

Mapping Table Update

SSD Monitor

WRITE (sector number, sector length)

Write to RAMDISK

FTL Module

Mapping Table Update

SSD Monitor

WRITE (sector number, sector length)

Write to RAMDISK

FTL Module

Mapping Table Update

SSD Monitor

while (channel & flash is occupied)
{
  /* busy loop */
}

FTL Module

Mapping Table Update

SSD Monitor

WRITE (sector number, sector length)

Write to RAMDISK

FTL Module

Mapping Table Update

SSD Monitor

while (channel & flash is occupied)
{
  /* busy loop */
}

FTL Module

Mapping Table Update

SSD Monitor

WRITE (sector number, sector length)

Write to RAMDISK

FTL Module

Mapping Table Update

SSD Monitor

while (channel & flash is occupied)
{
  /* busy loop */
}
Maintains **mapping Information** and **block status**

- Maintain logical to physical **address translation**.

- Garbage Collection

- Wear-leveling

- Issue **NAND read/write request** to IO Emulator Module.
Emulate NAND IO operation

- Perform NAND read/write/erase operation.
- **Introduces** the *appropriate amount of latency* with latency manager.
- Support multi-channel, multi-way operation **using single thread**.
3 Flash memories, 3 Channel

**WRITE1** (0, 0, 0)  **WRITE2** (1, 0, 0)  **WRITE3** (2, 0, 0)  **WRITE4** (0, 0, 1)

<table>
<thead>
<tr>
<th>Flash Type</th>
<th>IO Type</th>
<th>Reg IO Time</th>
<th>Cell IO Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Flash 0</td>
<td>Write</td>
<td>5</td>
<td>2</td>
</tr>
<tr>
<td>Flash 1</td>
<td>Write</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>Flash 2</td>
<td>Write</td>
<td>3</td>
<td>4</td>
</tr>
</tbody>
</table>

**UPDATE Data Structure**

**busy waiting**

**Write to Reg**

**Cell Programming**
SSD Monitor: Real-time Monitoring Tool

- # of Write/Read/Erase requests
- IO Bandwidth (MB/s)
- Total # of sectors written/read

- # of Merge Operations

- # of TRIM Commands

- Write Amplification
Intel X25M vs. VSSIM (configured as X25M)

- 10 channel, 2 way, 4KByte page size

- Seq Read / Write: 512MB File size, 512KB record size
- Rand Read / Write: 512MB File size, 4KB record size
VSSIM validation with X25M using Demerit

- 512MB File size, 512KB record size, Sequential Read/Write
Strength of VSSIM

Real-time Execution
- Do not need trace
- Can measure Host performance
- Display SSD behavior in Real-time

Modularize
- Easily change or fix firmware (Mapping Algorithm, GC, W/L, and etc.)

Can simulate various SSD architecture
CASE STUDIES
## SSD Specification

<table>
<thead>
<tr>
<th>SSD Label</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>Channel</td>
<td>8</td>
<td>4</td>
<td>2</td>
<td>4</td>
<td>2</td>
</tr>
<tr>
<td>Way</td>
<td>1</td>
<td>2</td>
<td>4</td>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td># of Flash</td>
<td>8 EA</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td># of Flash</td>
<td></td>
<td>4EA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flash size</td>
<td>2 Gbyte</td>
<td></td>
<td>4 Gbyte</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Page size</td>
<td>2 Kbyte</td>
<td></td>
<td>4 Kbyte</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Read</td>
<td>60 usec</td>
<td></td>
<td>50 usec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Program</td>
<td>800 usec</td>
<td></td>
<td>900 usec</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Erase</td>
<td>1.5 msec</td>
<td></td>
<td>2 msec</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
In all workloads, 76% of write operations are either less than 4KB or larger than 64KB.

- Win7: Windows7 Installation
- MS: MS Office Installation
- PS: Photoshop Installation
- VD: Video File Copy
- MP3: 100 MP3 File Copy
- Ubt: Ubuntu 10.04 Installation
- Xen: Xen Compile.

IO Size Distribution (Write, # of Cmd)
FTL should be designed to handle multiple mapping granularity.
SSD performance and # of channels

- SSD-A: 8 Ch 1 Way
- SSD-B: 4 Ch 2 Way
- SSD-C: 2 Ch 4 Way
- SSD-D: 4 Ch 1 Way
- SSD-E: 2 Ch 4 Way

"Increasing Way Parallelism" can be a possible solution for increasing performance.

As # of ways is doubled and # of channels is reduced, the performance decreases by 4~8%
512MB File size, 4KB record size, Random Write (IOZONE)

Channel Parallelism becomes less beneficial as Flash page size becomes larger.
The Basics of Hybrid Mapping FTL

FAST [Lee06]

SATA Command:

( sector number, length )

Sequential Write Detector

Sequential Write

Random Write

Seq Log Block

Rand Log Block

LAST [Lee08]

SATA Command:

( sector number, length )

Sequential Write Detector

Sequential Write

Random Write

Hot/Cold Identification

Seq Log Block

Rand Log Block

Hot Rand Log

Cold Rand Log
Write Request:
Write Page 2
Write Page 4
Interleaving log block writes

Log block is striped across the channel or way: Log block fragmentation.
Not interleaving log block writes

Under utilizing Channel Parallelism
Hybrid Mapping and Multi-channel / multi-way SSD

4 Channel 1 Way SSD

- **FAST**: Interleaves only random log blocks.
- **LAST**: Interleaves only cold random blocks.

Hybrid mapping does not fit in **Multi-ch/Multi-way SSD**.
Write Amplification Factor as a Performance Metric

WAF = \frac{\text{# of page writes which actually happen into flash memory}}{\text{# of page writes from the host}}

WAF is a fair metric for SSD performance and endurance.

Well, I think the WAF may not be a right performance indicator.
Sequentiality Detection & Performance

**FAST**
- **Yes**
  - Sequential?
    - Yes: Sequential Log Blocks
    - No: Random Log Blocks
  - Random Log Blocks
- More Parallelism, Higher Merge Overhead

**LAST**
- **Yes**
  - Sequential?
    - No: Random Log Blocks
    - Yes: Sequential Log Blocks
  - Sequential Log Blocks
- Less Parallelism, Lower Merge Overhead

A Page Write
IO parallelism not only improves performance, but also increases WAF.
Over-provisioning Factor

Flash Memory

Visible to the host

Invisible to the host (Log block)

Data block

Random Log block

Sequential Log block

Over-provisioning Factor = \[
\frac{\text{# of Log blocks}}{\text{# of Data blocks}}
\]

FTL_{30\%}^{(10, 5)}

(# of Rand log blocks, # of Seq log blocks)
Importance of Hot/Cold Identification

- Windows 7 Installation

To make the over-provisioning effective, **Hot data identification** is critical.

- # of blocks / Flash = 4,096 blocks
- # of pages / Block

Same performance with 1/6 of OP
VSSIM: A novel SSD Simulation Tool

- Can measure **Host performance**
- Display SSD behavior in **Real-time**
- Can simulate **various SSD** architecture
- **Easily change** or fix firmware
- **Validate** VSSIM with Real-SSD

VSSIM is publicly available at  [http://esos.hanyang.ac.kr/vssim](http://esos.hanyang.ac.kr/vssim)
Thank you

Q & A