SSD Firmware Implementation Project
- Lab. #2 -

Sang-Phil Lim (lsfeel0204@gmail.com)
SKKU VLDB Lab.

2011-03-31
## Lab. Time Schedule

<table>
<thead>
<tr>
<th>Lab.</th>
<th>Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>#1</td>
<td>FTL Simulator Development Guide</td>
</tr>
<tr>
<td>#2</td>
<td>FTL Simulation Guide</td>
</tr>
<tr>
<td>#3</td>
<td>Project 1 Presentation</td>
</tr>
<tr>
<td>#4</td>
<td>Jasmine OpenSSD platform tutorial #1</td>
</tr>
<tr>
<td>#5</td>
<td>Jasmine OpenSSD platform tutorial #2</td>
</tr>
<tr>
<td>#6</td>
<td>FTL Porting Guide</td>
</tr>
<tr>
<td>#7</td>
<td>Firmware Debugging Guide</td>
</tr>
<tr>
<td>#8</td>
<td>SSD Performance Evaluation Guide</td>
</tr>
<tr>
<td>#9</td>
<td>Project 2 Presentation</td>
</tr>
</tbody>
</table>
Contents

- Introduction of example FTL
- FTL simulation guide
- FTL & Workload
- Sample workload pattern observation
Example FTL - Greedy

• Simple FTL scheme is implemented in an FTL simulator
  – Page mapping FTL
  – Garbage collection policy: **Greedy**
    • That is “Select a victim block which contains minimum valid pages”
    • To reduce average GC overhead
Example FTL - Greedy

• Data structure
  – Page-level mapping table
    • For address mapping
      • *struct pagemap* (map.c)
  – Bit-map table
    • For victim block selection
      • *struct bitmap* (bitmap.c)
  – Misc. metadata
    • for GC/page_alloc
      • *struct vm* (vm_ftl.c)

<table>
<thead>
<tr>
<th>LPN</th>
<th>PPN</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>100</td>
</tr>
<tr>
<td>1</td>
<td>20</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>PPN</th>
<th>validity</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Set</td>
</tr>
<tr>
<td>1</td>
<td>Unset</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Example FTL - Greedy

- Write operation (ftl_write() in ftlsimple.c)

  - Per-page processing
    - Read-modify-write (if not a full page write)
    - Allocate free page
    - Write new data
  - Invalidate old page
  - Update mapping table
  - GC start (if free block < MAX_FREE_BLKS)
Example FTL - Greedy

- Garbage collection \((\text{vm}_\text{gc}_\text{start}())\) in \(\text{vm}_\text{ftl}.c\)

```
while (vm.fb_num < MIN_FREE_BLKS + GC_BLKS)
```

![Diagram showing garbage collection process](image)
FTL Simulation Guide

1. Configure NAND flash
   - TOTALSIZE_GB = 32 (but, for Financial1.diskmon = 1)
   - EXTRABLKS = 1~32% of TOTALSIZE_GB (increasing by a factor of 2)

2. Fill up ‘user space’ with valid data before simulation
   - Call `warm_all()`

3. Aging NAND flash sufficiently
   - Write data at least 2 times larger than provisioning space
   - Ex. Total write req. of workload = 2GB, provisioning space = 2GB, then..
     $ sim trace_file 3
FTL Simulation Guide

- Write cliff

Aging period is mandatory for observing actual performance behavior of FTL.
FTL & Workload

• Most of cases, write pattern of workload can significantly impact on an FTL performance
  – Research goal: “Toward for a Holy-grail FTL”
FTL & Workload

• Impact factors to determine FTL performance
  – Spatial/Temporal locality
  – Alignment/Granularity
  – Range/Order/Interval
  – Read-Write ratio
  – Etc.

More info...

Sample Workload Pattern

- Web32G_NTFS.diskmon
- General32G_NTFS.diskmon
Sample Workload Pattern

• ss_pc3.diskmon
Sample Workload Pattern

• Financial1.diskmon
• tpcc.diskmon
Notice for Lab #3

• In next Lab. – “Project #1 presentation”
  – Describe FTL implementation in detail
    • Including some optimization ideas
  – Analyze your simulation results for each given workload
    • Performance implication
Any Questions?