Introduction to NAND Flash Aware Hibernation-based Boot

Kyungsik Lee, Software Engineer, LG Electronics
Overview

• Boot time reduction
  – Traditional boot time reduction techniques

• Hibernation boot
  – Hibernation (suspend to disk)
  – Cold vs. hibernation boot time

• Proposal for a new hibernation boot
  – Improving hibernation boot speed.
  – Extending the lifetime of the flash memory.
Boot time reduction
Boot time reduction

• Why is it important?
  – To improve user experience
    Convenience and customer satisfaction
  – Regulatory requirements
    There are critical safety reasons that drive boot time requirements for the automotive industry.
  – Marketing
    Have you seen one-second Android boot?
Traditional techniques

• Measuring
  – Bootchart: a tool for performance analysis and visualization of the Linux boot process.

• Optimizing
  – bootloader, kernel and user space
  – Do it in parallel(independent tasks).
  – Maximizing I/O and minimizing the amount of data.

• Maintainability
  – How maintainable is it?
Hibernation Boot
What is hibernation?

• Hibernation (suspend to disk)
  – Hibernation in computing is powering down a computer while retaining its state.
  – Upon hibernation, the computer saves the contents of its random access memory (snapshot image) to a hard disk or other non-volatile storage.
  – Upon resumption, the computer is exactly as it was before entering hibernation.
Case Study

• Using i.MX 8MQ as a case study for hibernation boot
  – CPU (4 cores) with 3GB memory and eMMMC
  – Bootloader: U-boot
  – Kernel: 4.9(base kernel for hibernation)
  – Android(Oreo)
Cold vs. Hibernation boot time

- **Cold boot**
  - Android is not optimized for fast boot.
- **Hibernation boot**
  - The way the upstream kernel uses for hibernation.
  - Snapshot Image: around 900 MiB
- **Measurement**
  - from power-on to starting Android launch
  - Cold boot: 14.8 sec.
  - Hibernation boot: 11.2 sec.
Optimizing hibernation boot time

- Upstream kernel hibernation
  - Not optimized for fast boot
- Image load time $\gg$ (suspend + resume) time
  - Reducing image size leads to faster image load time.
- Reducing snapshot image size
  - Swap out pages as much as possible.
  - Clear page cache. (sync; echo 3 > /proc/sys/vm/drop_caches)
  - Deduplicate pages and compress.
Deduplicate pages in memory

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>aaa</td>
<td>bbb</td>
<td>ccc</td>
<td>aaa</td>
<td>ddd</td>
<td>aaa</td>
<td>eee</td>
<td>fff</td>
<td>aaa</td>
<td>bbb</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aaa</td>
<td>bbb</td>
<td>ccc</td>
<td></td>
<td>ddd</td>
<td></td>
<td>eee</td>
<td>fff</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>aaa</td>
<td>bbb</td>
<td>ccc</td>
<td>ddd</td>
<td>eee</td>
<td></td>
<td>fff</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Original</th>
<th>Duplicate</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
</tr>
<tr>
<td>1</td>
<td>9</td>
</tr>
</tbody>
</table>
Boot time and Image size

**Boot time**
- Cold: 16 sec.
- Hibernation: 14 sec.
- Optimized: 8 sec.

**Image size**
- Hibernation: 1 KB
- Optimized: 200 KB
Extending the lifetime of flash memory

• Flash memory has a limited lifetime.

• How to maximize lifetime of flash memory in hibernation?
  – Decreasing the write amplification and the amount of data to be written.

• Log-structured block management
  – Dividing up the chosen partition into segments.
  – Writing to segments sequentially and decreasing the write amplification.

• Storage-based data deduplication
  – Reducing the amount of data to be written.
  – Data deduplication and Compression
Clusters and blocks

- A chosen partition is made up of clusters (apart from swap partition).
- A cluster is composed of blocks. A block is 4KB, the allocation unit size.
- Blocks of clusters are allocated when data is written to the clusters.
- Used clusters are reclaimed when they are no longer used and discarded by garbage collector to become idle clusters.
- Clusters are not overwritten until discarded (except header).
Cluster types

- Map: Locate meta and data clusters.
- Meta: A PFN(Physical Frame Number) table
- Data(Un/compressed): where snapshot image data is written.
- Dedupe: A table which has start block addresses of each hot clusters
- Usage count: Store usage count on each blocks
- Garbage collection: A table which has a list of clusters to be discarded
- Idle: To be allocated for use in the future
## Disk layout

- **Clusters (segmentation)**

<table>
<thead>
<tr>
<th>Header</th>
<th>Map</th>
<th>Meta</th>
<th>Dedupe</th>
<th>Usage</th>
<th>GC</th>
<th>Data</th>
<th>Idle</th>
</tr>
</thead>
</table>

- **Data cluster**

<table>
<thead>
<tr>
<th>Chunk Table size</th>
<th>Chunk Table (hash, byte offset, size)</th>
<th>Chunks</th>
</tr>
</thead>
</table>
Storage-based data deduplication

- Deduplication process
  - All pages are hashed first.
  - Pages are compared to the stored copy using hash values and whenever a match occurs, the redundant page is replaced with the entry in the map table that points to the stored page.
  - Unique pages are identified and stored with the hash values in the data clusters.
The amount of data written

- A: Deduplicate(memory)
- B: Deduplicate(A) with storage
- C: Compress(B)
Performance regression

- Image loading speed is getting slower
  - Upon hibernation, the snapshot image is fragmented by the storage-based deduplication process.
  - Accessing fragmented image requires more block I/O frequency.(more random I/O patterns)
Image Loading Performance
Defragmentation

• Selective deduplication
  – Choose hot data clusters based on the number of hot blocks (usage count > a specified threshold).
  – Deduplicate snapshot image with hot data clusters.
  – Cold data clusters will be reclaimed.
  – A slight increase in image size

• Hot and Cold clusters
  – Hot clusters: to be used to deduplicate the new snapshot image
  – Cold clusters: not to be used for deduplication and reclaimed
Usage count on blocks

- Heat map
Image Loading Performance (after)
The amount of data written (after)
Reclaim clusters

• In case of running out of idle clusters
  – Reclaim occurs when the number of idle clusters is below a threshold.
  – Cold and less hot data clusters are reclaimed to get more idle clusters for the next snapshot image.

• Other used clusters
  – Other used clusters are reclaimed after resumed.
Garbage Collection

• Garbage collector
  – A background thread which performs automatic storage management for hibernation.
  – Discarding the reclaimed clusters.
  – Garbage collection will occur at run time when the number of reclaimed clusters is above a threshold.
Questions?