UBI Fastmap

Thomas Gleixner - linutronix GmbH

Embedded Linux Conference Europe 2012, Barcelona
UBI Overview

IPL/Bootloader
UBI0 (~8GB)
MTD Part 0 (1MB)
MTD Part 1 (~8 GB)
MTD Subsystem
Flash Device, e.g. NAND Flash 8GB

UBIFS

UBI Vol 0 2MB (Kernel image)
UBI Vol 1 ~6GB
UBI Vol 2 ~2GB

mount /
mount /data
UBI

Provides

▶ Volume manager for FLASH
▶ Full device wear leveling
▶ Bad block handling
▶ Data integrity mechanisms
Volume management

UBI Logical Erase Blocks (LEB)

Flash Physical Erase Blocks (PEB)
UBI

Metadata storage

- UBI Headers
- Usable data area
- FLASH Eraseblock
UBI

Metadata
- Erasecount header
- Volume information header
Volume information header

- Volume id
- Logical eraseblock number in volume
- Version counter
UBI

Metadata retrieval

Must be scanned at boot time
UBI

Attach time

- $O(N)$
- Grows linear with FLASH size
Attach time

\[ N = \text{number of eraseblocks} \]
\[ T_p = \text{time to read a single flash page} \]
\[ H_p = \text{number of header pages} \]

\[ T_a = N \times T_p \times H_p \]
UBI attach time

Example I: NAND 64MB 1024PEBs 512B pagesize

N = 1024
Tp = 50us
Hp = 1

Ta = 1024 * 50us * 1 = 51.2ms
UBI attach time

Example II: NAND 4GB 8192PEBs 4K pagesize

\[ N = 8192 \]
\[ T_p = 100\text{us} \]
\[ H_p = 2 \]

\[ T_a = 8192 \times 100\text{us} \times 2 = 1.6384s \]
Can we be smarter?

- Store metadata in a special volume
- but ...
UBI metadata

Where to store metadata?

- No static storage space on NAND
- Metadata update needs to be rare
- No violation of UBI robustness
UBI metadata volume

How to find it?

▶ Split into two volumes
  ▶ Reference volume
  ▶ Data volume
UBI metadata volumes

Reference volume

- contains information about the metadata volume location
- is located within the first N physical erase blocks
- has to be found by scanning
UBI metadata volumes

Data volume
- contains information about all physical eraseblocks
- condenses UBI header data
UBI metadata volumes

Avoid fast updates

- by storing a pool list
- by scanning the erase blocks in the pool list
- by rewriting metadata only when pool list changes
UBI metadata volumes

Pool list

- Configurable number of erase blocks
- Used for current write operations
UBI metadata volumes

Pool list changes

- due to wear leveling
- due to client (e.g. UBIFS) requirements
UBI metadata volumes

Preserve robustness

- by preserving the UBI header semantics
- by fallback to full scanning mode
UBI fastmap

Attach mode scheme

Fastmap reference -> Fastmap storage

Must be scanned at boot time
Attach mode scheme

Must be scanned at boot time
UBI fastmap

Attach mode scheme

Fastmap reference -> Fastmap storage

Must be scanned at boot time
UBI fastmap

**Attach time**

\[ Nb = \text{number of eraseblocks} \]
\[ Ns = \text{number of blocks to scan for reference volume} \]
\[ Np = \text{number of pool eraseblocks to scan} \]
\[ Hp = \text{number of header pages} \]
\[ Sb = \text{size of an eraseblock} \]
\[ Sp = \text{size of a page} \]
\[ Sd = \text{size of metadata per eraseblock} \]
\[ Tp = \text{time to read a single flash page} \]

\[ N\text{totp} = (Ns + Np) \times Hp + Sb / Sp + Sp / Sd \]
\[ Ta = N\text{totp} \times Tp \]
UBI fastmap attach time

Example I: NAND 64MB 1024PEBs 512B pagesize

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Nb</td>
<td>1024</td>
</tr>
<tr>
<td>Ns</td>
<td>16</td>
</tr>
<tr>
<td>Np</td>
<td>16</td>
</tr>
<tr>
<td>Hp</td>
<td>1</td>
</tr>
<tr>
<td>Sb</td>
<td>65536</td>
</tr>
<tr>
<td>Sp</td>
<td>512</td>
</tr>
<tr>
<td>Sd</td>
<td>128</td>
</tr>
<tr>
<td>Tp</td>
<td>50us</td>
</tr>
</tbody>
</table>

\[
N_{totp} = (16+16) \times 1 + \frac{65536}{512} + \frac{1024 \times 96}{512} = 352
\]

\[
T_a = 352 \times 50us = 17.6ms \text{ (UBI: 51.2ms)}
\]
UBI fastmap attach time

Example II: NAND 4GB 8192PEBs 4K pagesize

- Nb = 8192
- Ns = 64
- Np = 256
- Hp = 2
- Sb = 512*1024
- Sp = 4096
- Sd = 128
- Tp = 100us

\[
N_{totp} = (64+256)\times2 + \frac{512\times1024}{4096} + \frac{8192\times96}{4096} = 960
\]

\[
Ta = 960 \times 100\text{us} = 96\text{ms} (UBI: 1.6384s)
\]
UBI fastmap

Summary

▶ Fastmap provides significant speedup
▶ Speedup grows with flash size
UBI fastmap

Further possible optimizations

- Compressed fastmap storage
- Let the bootloader hand the scan table to the kernel
- Implement supplementary NVRAM support
UBI fastmap

**Code**
- Merged in Linux 3.7
- Sponsored by CELF
- Designed and implemented by linutronix