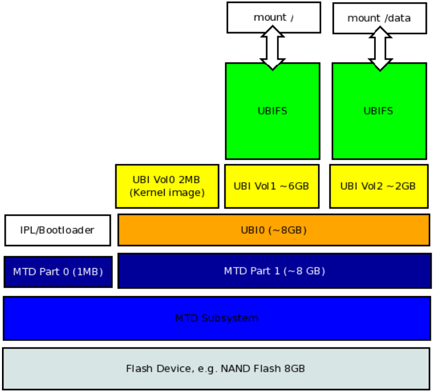
UBI Fastmap

Thomas Gleixner - linutronix GmbH

Embedded Linux Conference Europe 2012, Barcelona

UBI

Overview



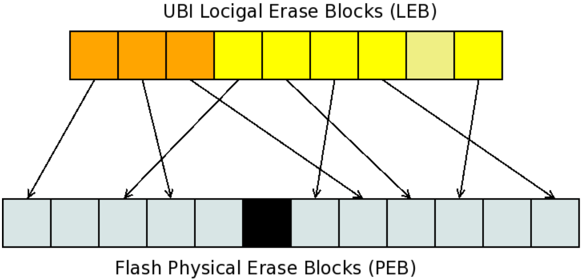
UBI

Provides

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

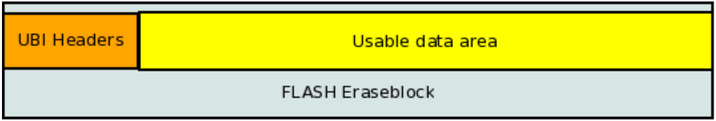
UBI

Volume management



UBI

Metadata storage



UBI

Metadata

- Erasecount header
- Volume information header

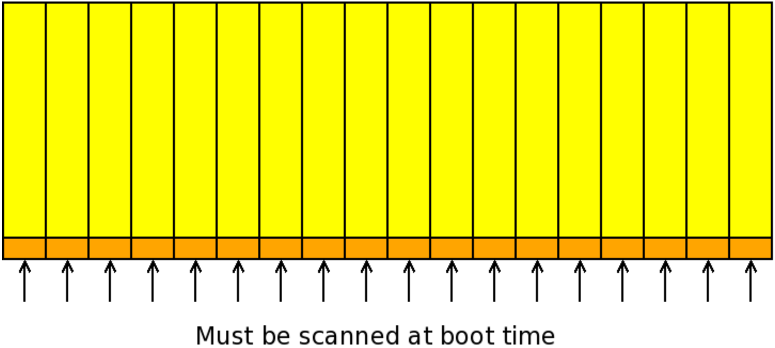
UBI

Volume information header

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

UBI

Metadata retrieval



UBI

Attach time

- O(N)
- Grows linear with FLASH size

UBI

Attach time

```
N = number of eraseblocks
Tp = time to read a single flash page
Hp = number of header pages

Ta = N * Tp * Hp
```

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
Tp = 100us
Hp = 2

Ta = 8192 * 100us * 2 = 1.6384s
```

UBI attach time

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

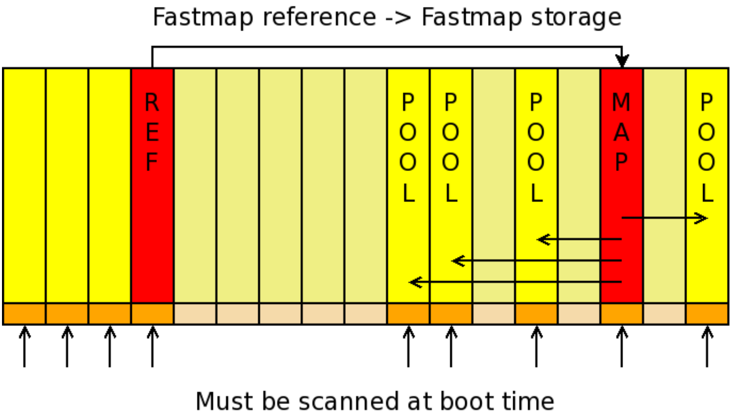
UBI metadata volumes

Preserve robustness

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

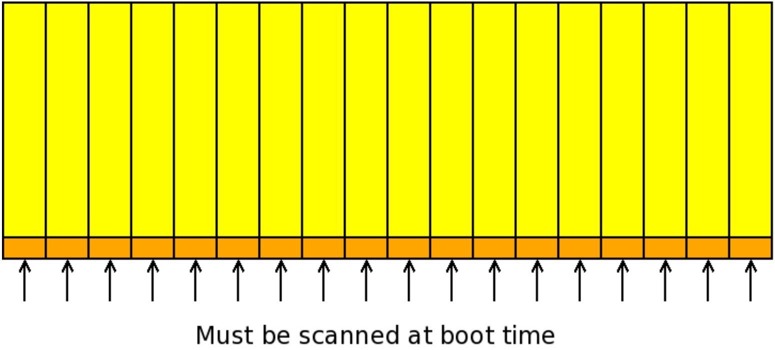
UBI fastmap

Attach mode scheme



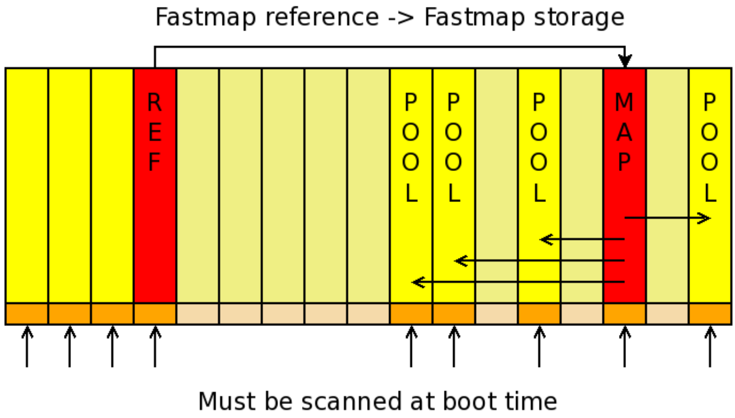
UBI

Attach mode scheme



UBI fastmap

Attach mode scheme



UBI fastmap

Attach time

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

Ntotp = (Ns + Np) * Hp + Sb / Sp + Sp / Sd
Ta = Ntotp * Tp
```

UBI fastmap attach time

Example I: NAND 64MB 1024PEBs 512B pagesize

$Nb = 1024$	$Sb = 65536$
$Ns = 16$	$Sp = 512$
$Np = 16$	$Sd = 128$
$Hp = 1$	$Tp = 50us$

```
Ntotp = (16+16)*1+65536/512+1024*96/512 = 352
Ta = 352 * 50us = 17.6ms (UBI: 51.2ms)
```

UBI fastmap attach time

Example II: NAND 4GB 8192PEBs 4K pagesize

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

```
Ntotp = (64+256)*2+512*1024/4096+8192*96/4096 = 960
Ta = 960 * 100us = 96ms (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