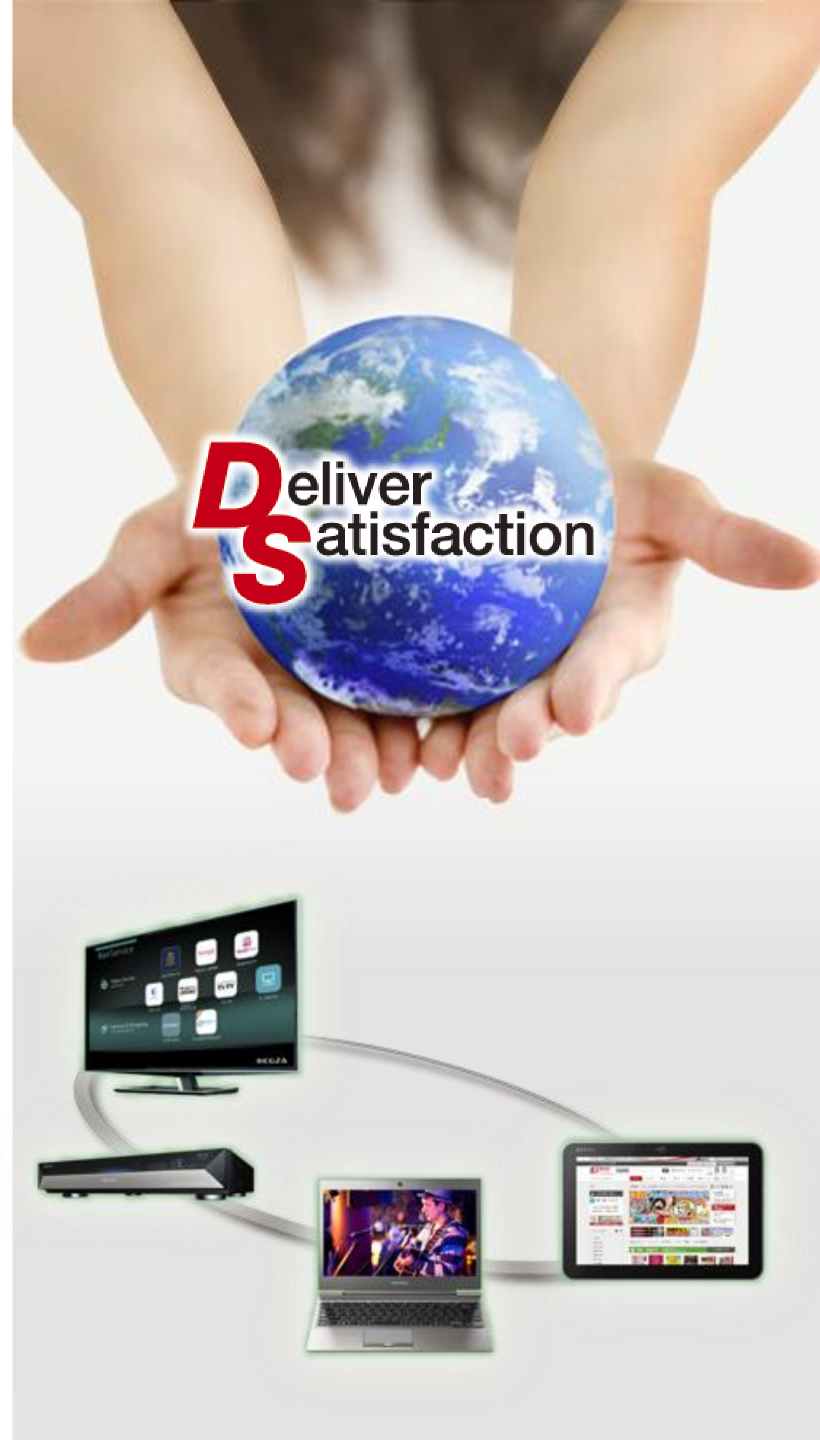
TOSHIBA
Leading Innovation

Evaluation of Flash Filesystems Update

SFTL and BENAND



2012/9/20

Toshiba Corporation
UWATOKO Katsuki

Contents

- SFTL (Simple Flash Translation Layer) NAND Driver
- BENAND (Built-In ECC NAND)
- fastmap of UBI

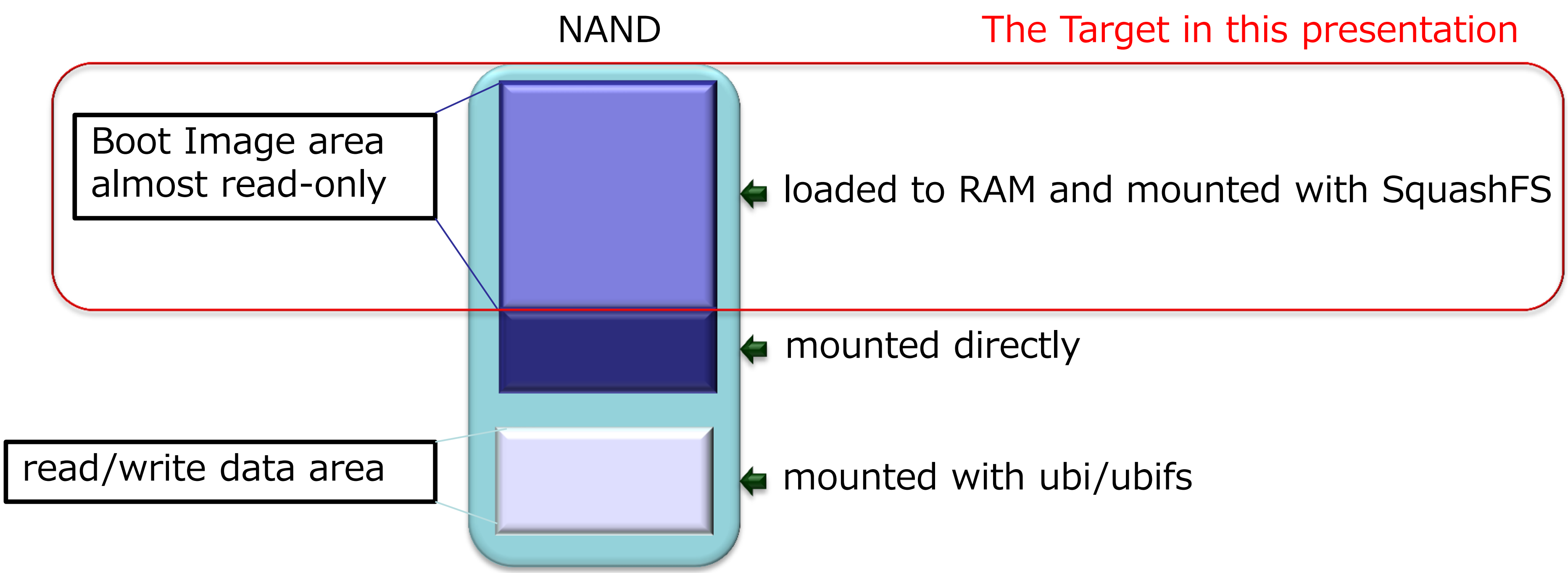
Background

- Boot Sequence of our target.
  - load the boot image from NAND to RAM
  - mount the image on RAM with SQUASHFS

    The reasons are:

    it is faster boot than direct mount in our target.

    it reduces the number of NAND read times and make it predictable for reliability.
  - mount some parts of the boot image directly.



Background

- Requirements of NAND Driver for our target
  - Quick initialization for fast boot
  - Bad Block Management
  - wear-leveling and scrubbing

    but GOOD wear-leveling is not necessary, because it is read-only except for update and the frequency of read is low.
  - Block Device interface for a direct mount

- Other Drivers and status (when we started to develop SFTL)

  mtdblock

  has no bad block management.

  sm_ftl/ssfdc

  These are for SmartMedia™. Therefore there are some limitations of Media Size, Zone wear-leveling etc.

  ubi + (ubifs or gluebi or ubiblk)

  The initialization time of ubi was not match for our target because it was slow. But now it provides “fastmap” which reduces the time. We are considering using this.

Features of SFTL

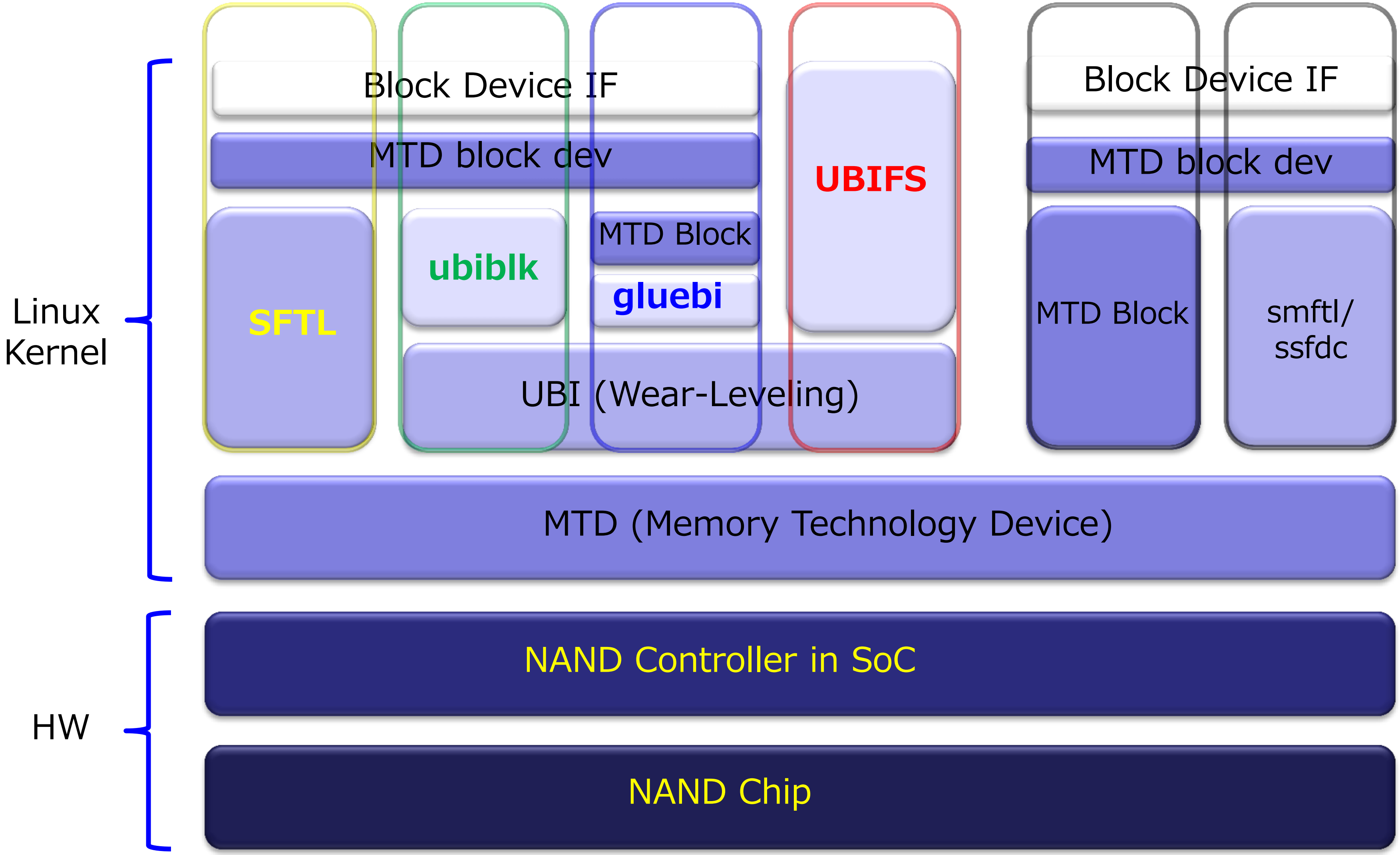
- SFTL (Simple Flash Translation Layer)
- Block Device Interface (using MTD block dev in MTD)
- Provides wear-leveling and Scrubbing
  - no static wear-leveling
- One erase block size cache
  - Sequential access is fast, even if a size of request is not Block size.
- uses 6 bytes in OOB for a logical address, status, version.
- Erases a erase block just before a write
  - for maintenance
  - easy to analyze after boot issue.
- We sent the patch to Linux MTD ML at Dec. 2012.

The maintainer suggested that:

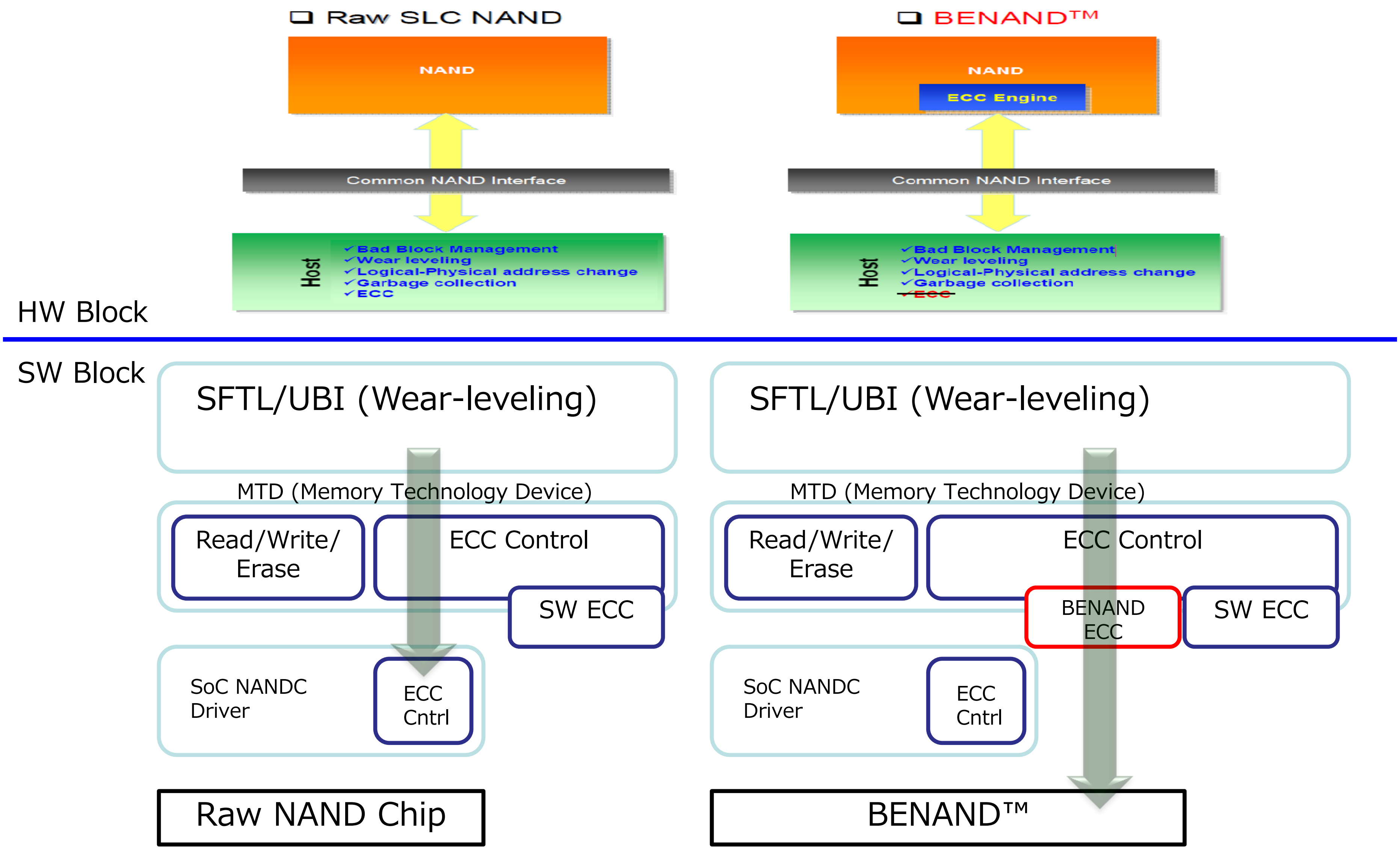
> Using OOB for status/data, not only for bad block status, is a bad idea nowadays. we have to re-design this.

Software Structure

Targets for a performance measurement in this presentation



BENAND (Built-In ECC NAND)



BENAND (Built-In ECC NAND)

- Easy to port BENAND support to MTD NAND
  - add ECC layout for BENAND
  - add a check routine of ECC status after read.

- We have a plan to send the patch to Linux MTD ML.

```
drivers/mtd/nand/nand_base.c      |  73 +++++++++++++++++++++++++++++++++++++++++++
include/linux/mtd/nand.h          |   3 +
2 files changed, 75 insertions(+), 1 deletion(-)
```

Environment

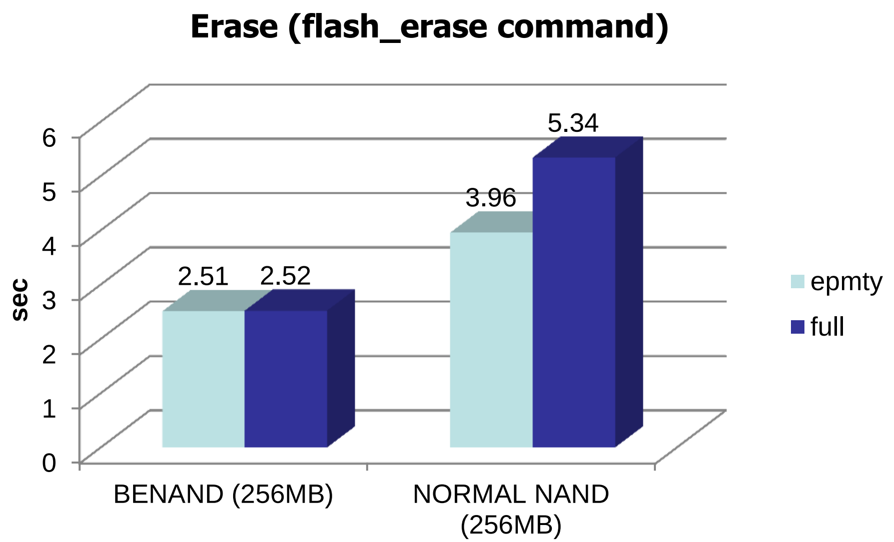
- CPU Cortex-A9 528Mhz, SMP (3 core)
- NAND Controller w/ HW ECC, w/o DMA
- Kernel Linux 3.0.32-ltsi

ubiblk	v0.9	9/26/2011
fastmap	RFCv5	5/17/2012
sftl		12/14/2011

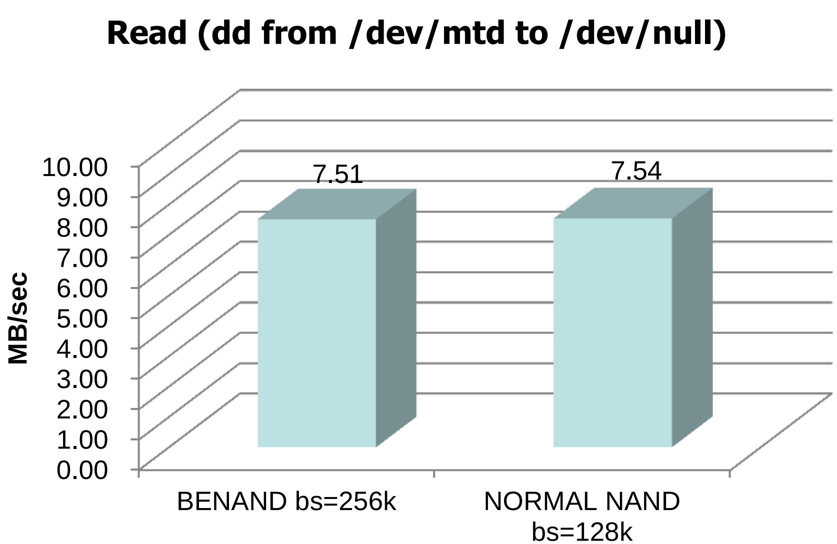
- mtd-utils 1.3.1
- gcc 4.5.1
- NAND

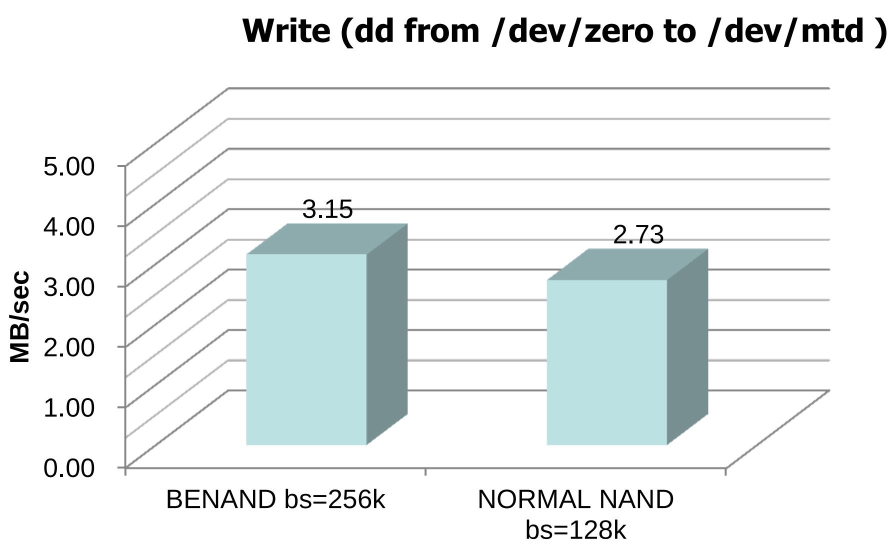
	BENAND (TC58BVG2S0FTA00)	**NORMAL NAND** (TC58NVG1S3ETA00)
Media Size	512MB	256MB
Block Size	256KB	128KB
Page Size	4096B + 128B	2048B + 64B

Basic NAND Performance (Erase/Read/Write)



The erase/write performance of BENAND is better than NORMAL. The result is considered that it is because of the difference of BLOCK size.

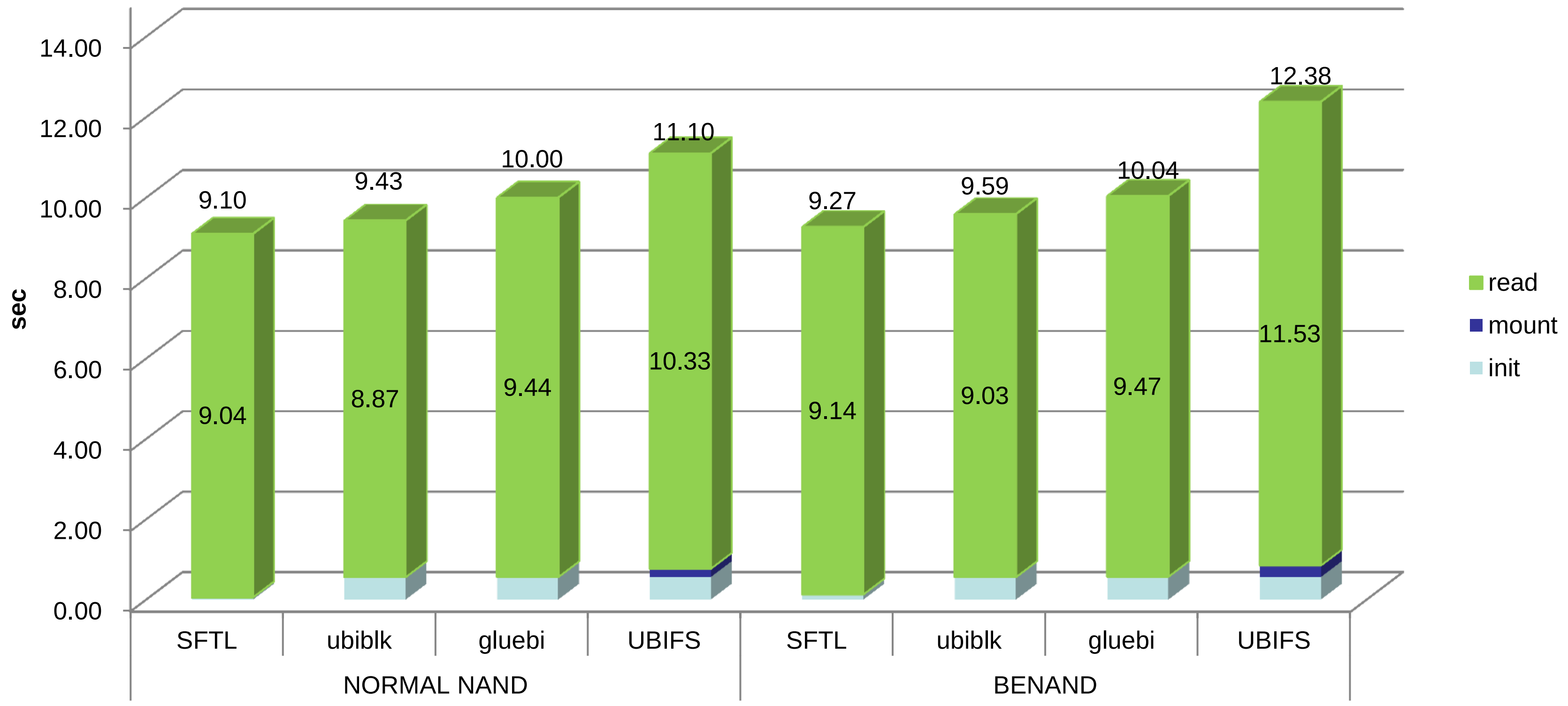




Performance on our target's boot sequence

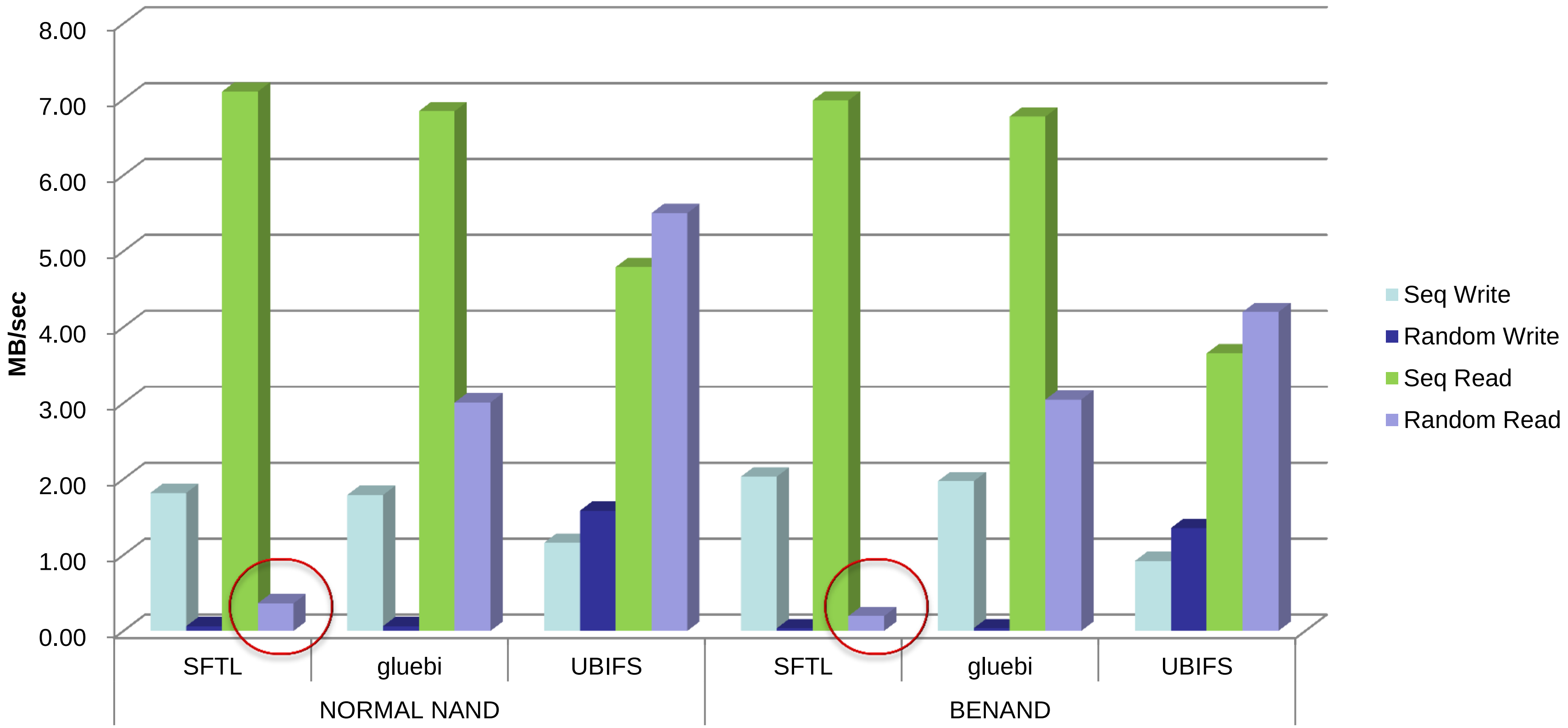
The following graph is the result of our target's boot sequence example.

- 128MB MTD Partition.
- read 64MB compressed file (squashfs image) to RAM.



Read/Write Performance – tiobench -

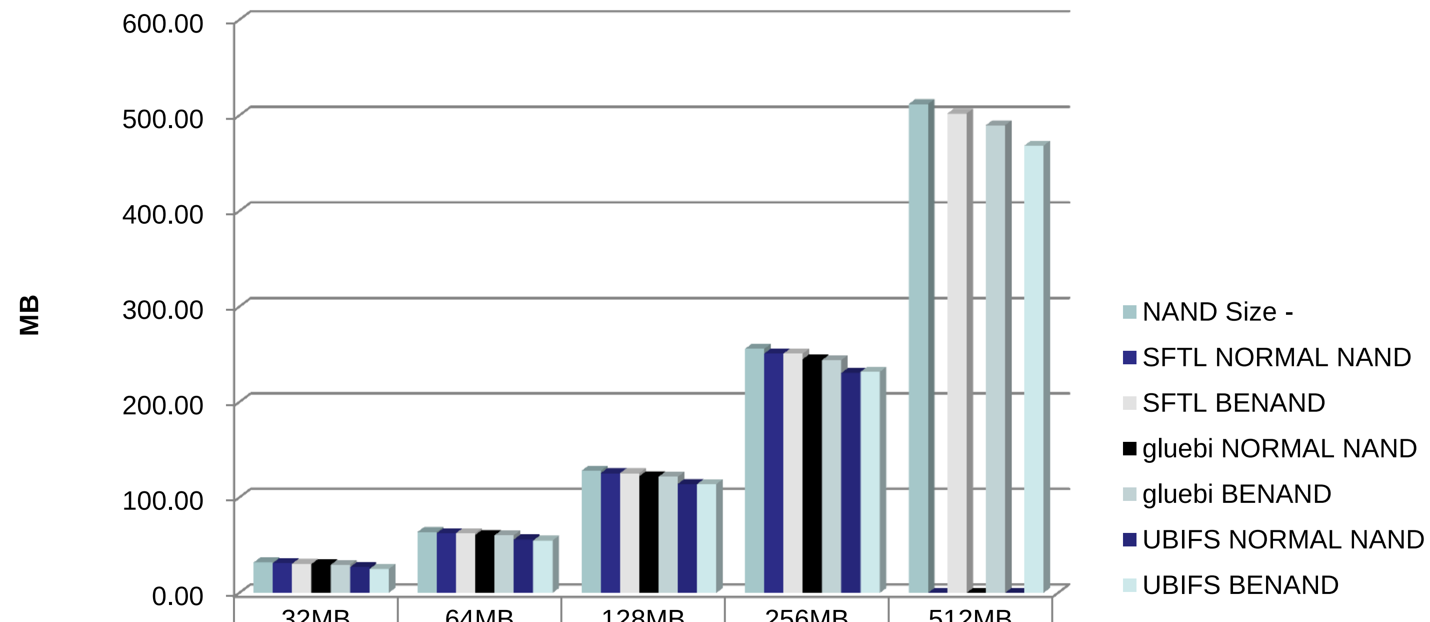
options: unit size 4KB (-b), one thread (-t), writing synchronously (-S)
drop cahe between each tests (modified the source)
ubifs mount with compr="none" option to avoid an influence of contents.



The performance of sequential read/write of SFTL are good. But the performance random is not good, especially random read. Because SFTL has just one erase block buffer for read/write.

Available Size

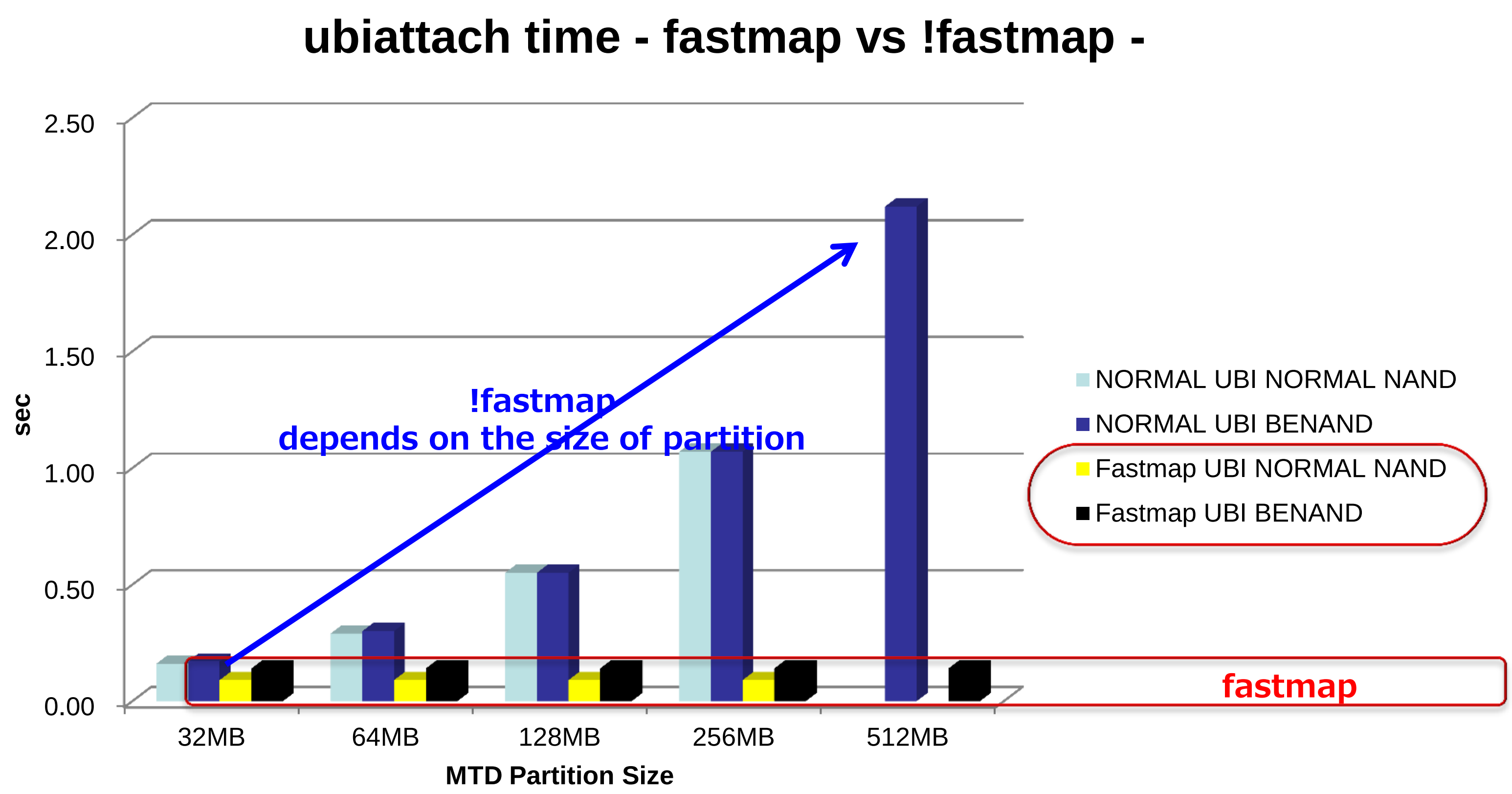
A size of writing compressed data until filesystem full.



	32MB	64MB	128MB	256MB	512MB
NAND Size -	32.00	64.00	128.00	256.00	512.00
SFTL NORMAL NAND	31.38	62.75	125.50	251.00	0.00
SFTL BENAND	30.75	62.75	125.50	251.00	502.00
gluebi NORMAL NAND	30.27	60.91	122.31	245.09	0.00
gluebi BENAND	29.55	60.55	121.82	244.12	489.74
UBIFS NORMAL NAND	27.05	56.09	114.30	230.56	0.00
UBIFS BENAND	24.98	55.00	114.09	231.78	468.38

These are just for reference data, because these depend on parameters of reserved block number for bad block, etc. But in the case of a small partition, especially using a large page NAND, we have to be careful the size.

UBI fastmap



ubiattach time - fastmap vs !fastmap -
2.50
2.00
1.50
1.00
0.50
0.00
sec
!fastmap
depends on the size of partition
NORMAL UBI NORMAL NAND
NORMAL UBI BENAND
Fastmap UBI NORMAL NAND
Fastmap UBI BENAND
fastmap
32MB
64MB
128MB
256MB
512MB
MTD Partition Size

Conclusion

- SFTL
  - SFTL has good performance to our target so far. But we have to reconsider the implement to merge Linux main line.

- BENAND
  - it’s easy to port BENAND to MTD NAND.
  - The performance is almost same as NORMAL NAND. In this presentation, the result is assumed to depend on the ERASE Block size, not BENAND.

- UBI fastmap
  - ubiattach time is very fast. We are considering to adopt this, and keep on evaluating it.

TOSHIBA

Leading Innovation