반드시 일등 합시다!

Flash Memory Based Bottom Up Analysis for Smart Phone System

목 차

2012. 10. 16

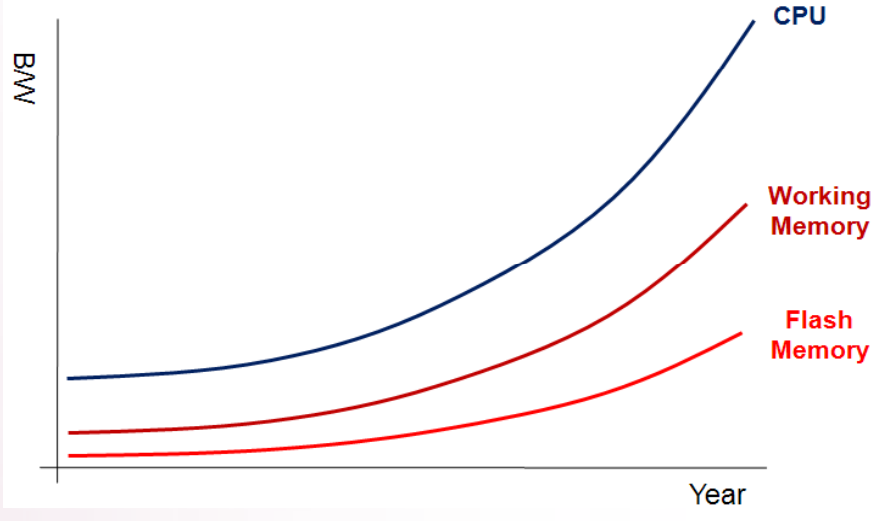
LG Electronics / Mobile Communications Company

LG Life's Good

1. Background

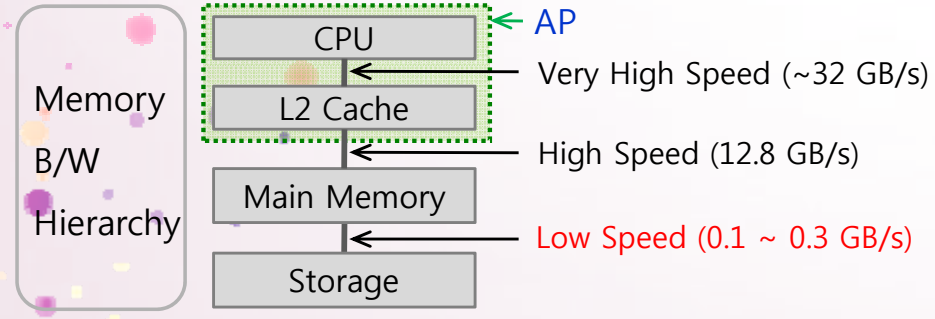
▪ Why Flash Memory is Important ?

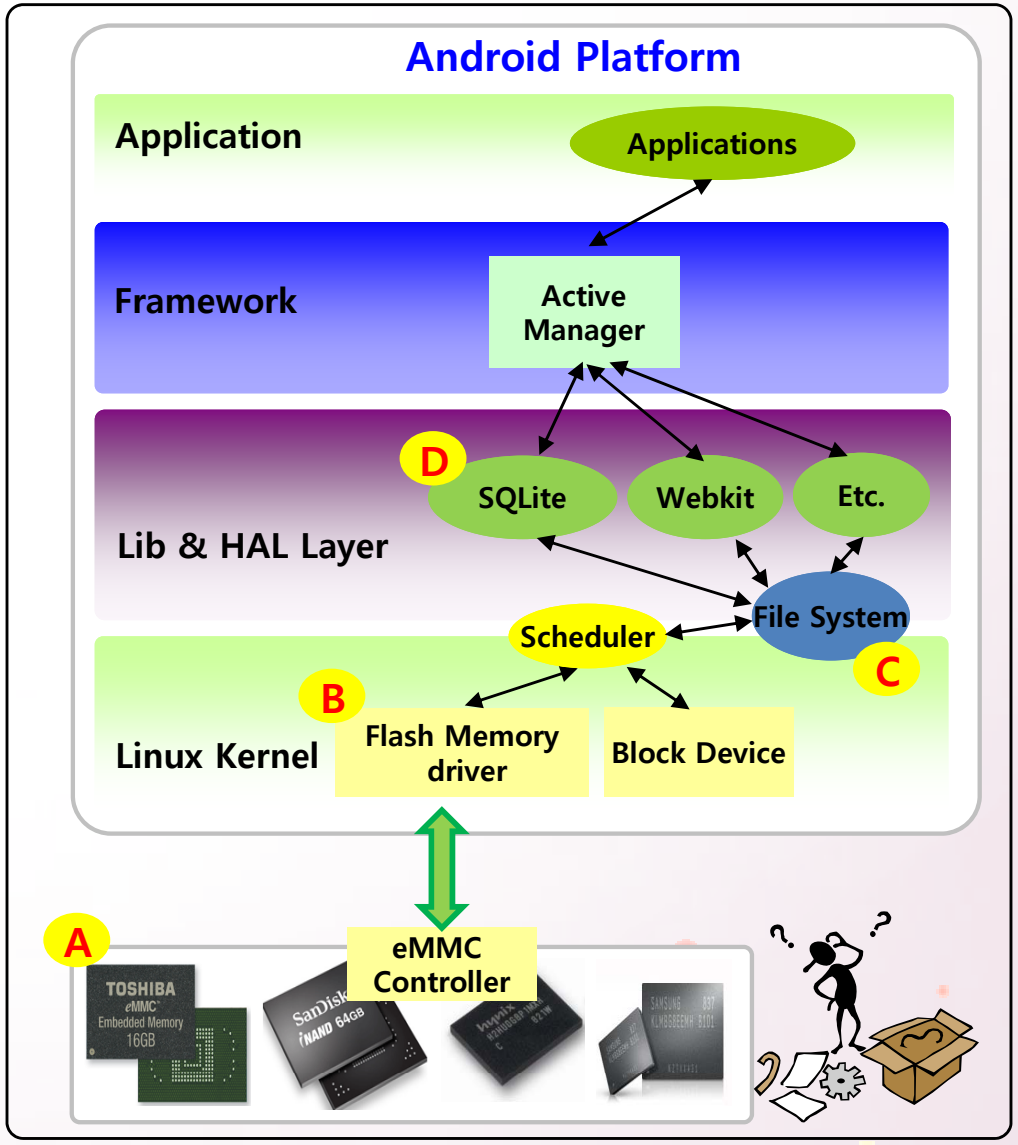
✓ Application Processor 및 Main Memory의 Speed를 Flash Memory가 따라가지 못하고 있는 상황



CPU/ Working Memory/ Flash Memory Bandwidths Gap

NAND Flash의 낮은 B/W 성능이 System 성능 저하 요인
(Booting, Web Browsing, App 실행 및 저장 속도 등)

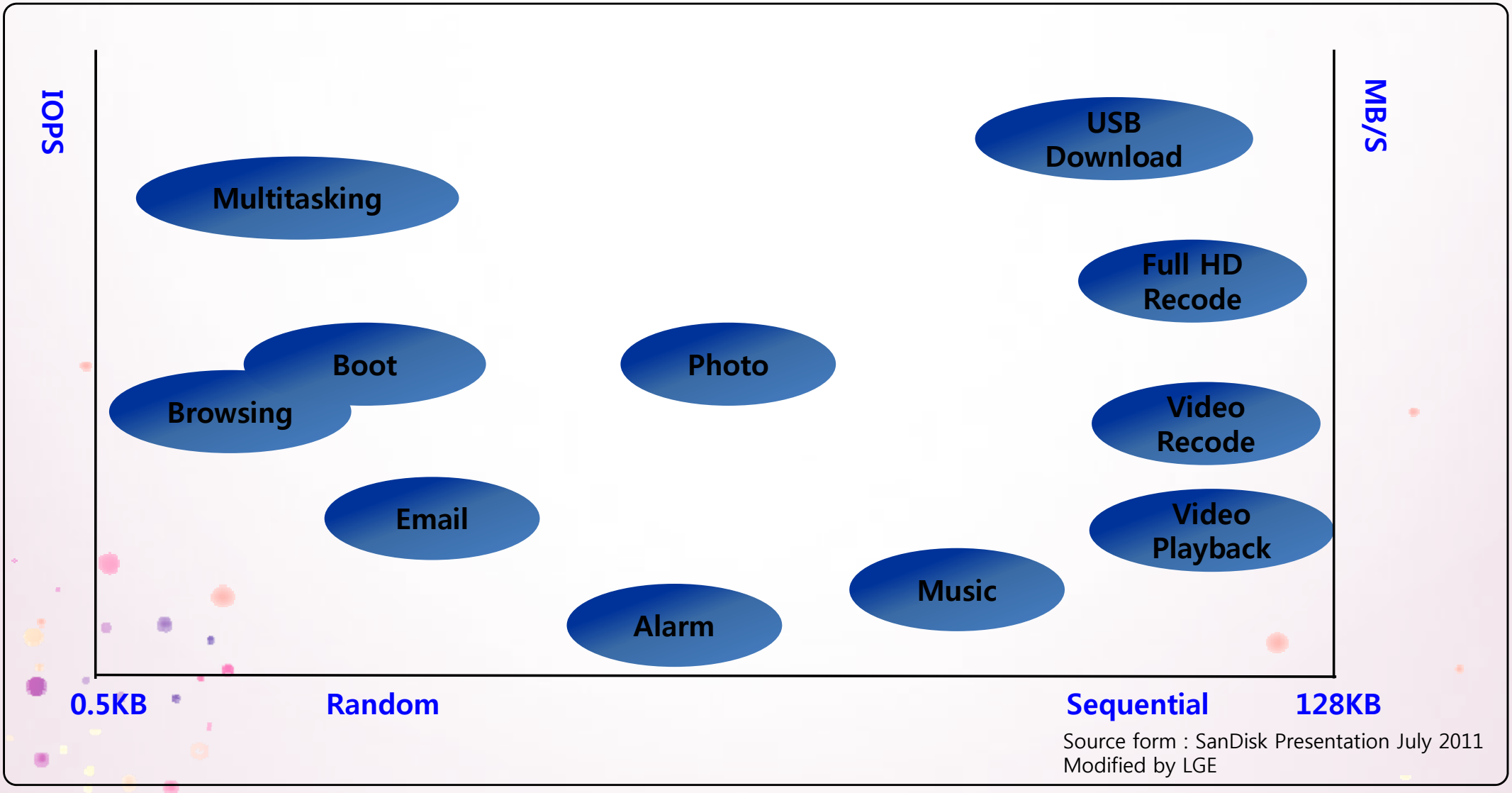




2. Controller & Driver Layer

▪ Flash Memory Performance Effect on Android System A

✓ Application 특성 마다 Bandwidth (Sequential) 혹은 IOPS (Random)의 Dependency를 가짐
✓ Android의 특성상 IOPS의 성능이 System 전체 성능에 Critical한 요소로 인식됨



Source form : SanDisk Presentation July 2011
Modified by LGE

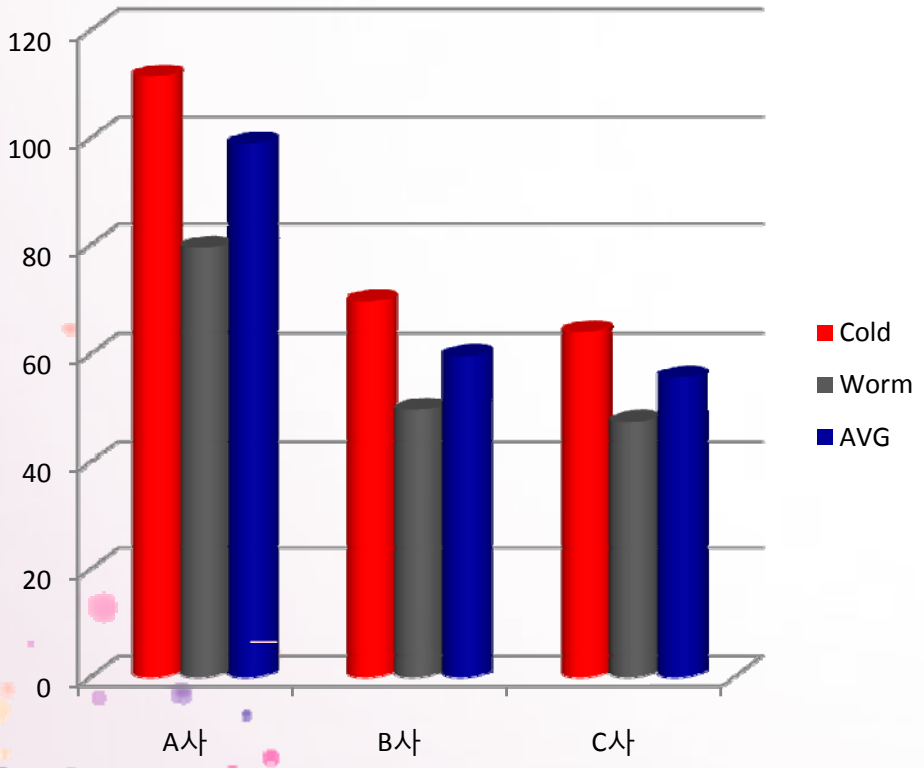
2. Controller & Driver Layer

▪ Performance Gap in Flash Memory A

✓ eMMC Memory의 Performance Gap이 발생 하는 이유?

동일 H/W Platform상에서 Flash Memory만 교체 후 Test

Browsing Speed BMT Test Result



Browsing BMT Test on Android System (10 Times Ave.)
– Android Gingerbread Version

Inside of eMMC

Flash Controller

Flash ROM

Flash RAM

Flash Chip

Flash Controller Role

- Low Level Driver
- FTL (Flash Transition Layer)
  - ① Block Replacement (Address Translation Table)
  - ② Dynamic Wear - leveling
  - ③ Garbage Collection
  - ④ Error Detection & Correction

Flash Controller

- Performance itself: Sequential
- FTL Algorithm : Random

eMMC성능의 가장 큰 원인은 FTL!!

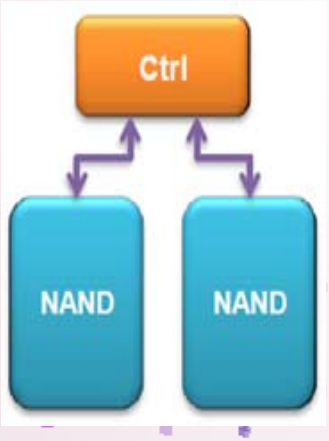
2. Controller & Driver Layer

▪ How to Increase Flash Memory Performance A B

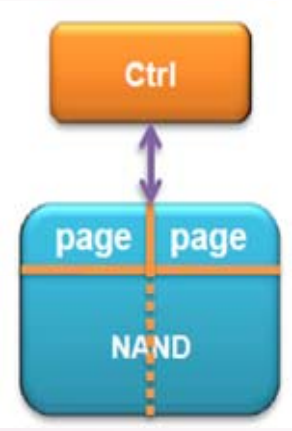
✓ **Flash Memory의 Performance를 증가 시키기 위한 방법?**

Well-Known Approach

① I/O : Data's path increase [Multi-I/O]
② Page size : Buffer size to program at once.
③ Interleave : Hide program operation by using long program time.
④ FTL algorithm
⑤ Physical Bandwidth
- Bus width & Frequency
- DDR
- Low Power Serial Bus (UFS)



Multi- I/O

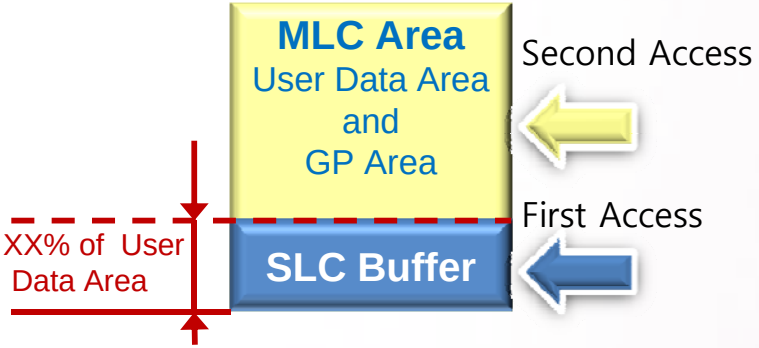


Page Size

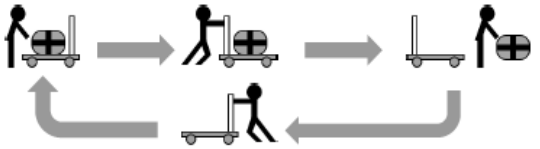
Interleave

More.....

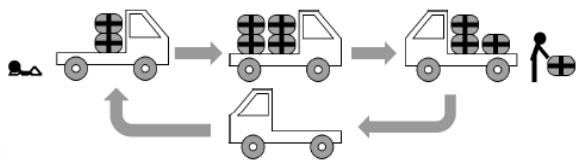
① SLC Buffer and/or SRAM Cache



② New Command

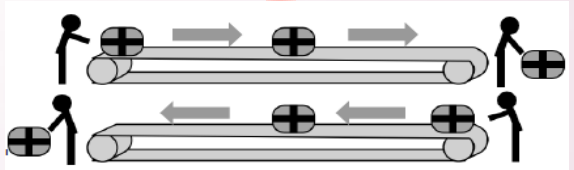


Legacy Command



Packed Command
(eMMC4.5)

Command Queuing
(UFS)



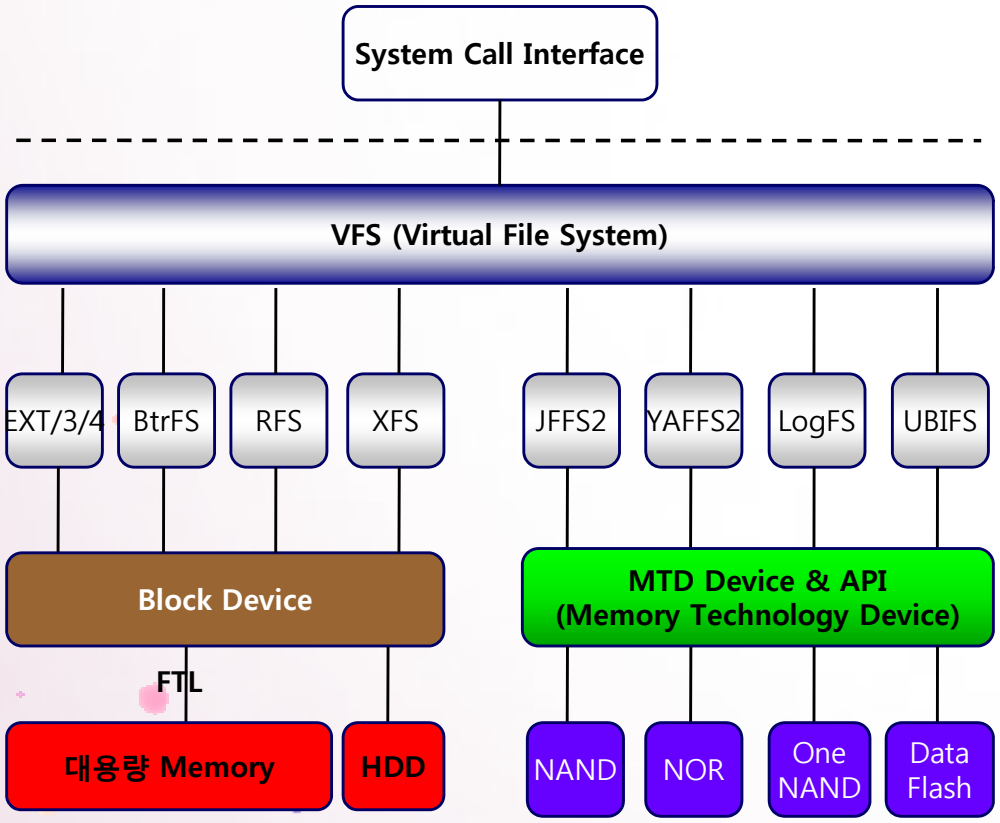
Source form : Toshiba Presentation Material

3. File System Layer

▪ File System in Android Smart Phone C

✓ File System에 의한 System Performance

File System



File System structure

EXT4 Vs BtrFS (Googling Result)

EXT4	BtrFS
현재 Linux에서 범용적으로 쓰이는 기본 File System	Metadata와 data관리에 B-tree를 사용한 Advanced File System
기존의 Ext3를 기반으로 향상 시킨 File System	기존의 File System과 다른 방식으로 개발된 File System
inode의 선 할당으로 인한 전체 용량의 1.5%정도 낭비 발생	작은 파일에 대한 I/O 성능이 좋아 Embedded 환경에 유리 (Booting Time향상)
부분적인 Error Free 제공 (File System의 중요부분인 Metadata의 깨짐만 보호)	Snapshot 지원을 통한 완전한 Error Free 제공

실제 BtrFS로 File System변경 후

-Booting Speed & Browsing Speed Check

-Module 반응 속도 Check

BMT 결과는 EXT4가 우세 함

향후 결과는?

3. File System Layer

▪ File System in Android Smart Phone C

✓ File System Journaling에 의한 System Performance Effect

Journaling

- 비 정상적인 종료로 인한, **파일시스템 손상을 최소화**하기 위함
- 오랜 시간이 걸리는 fsck[1]를 사용하는 기존 파일 시스템의 문제점을 해결하기 위해 고안
- Journal을 관리하는 방법으로 file system 손상을 피함

Metadata

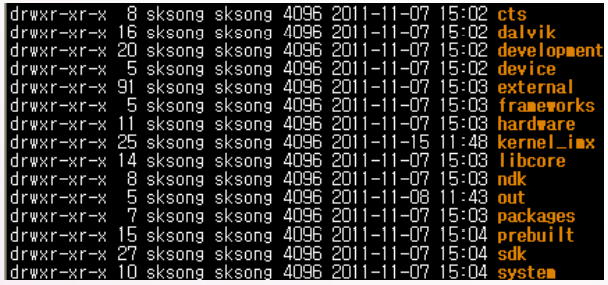
- 속성정보, data를 정의하고 설명해주는 data.
- Data의 종류에 따라 metadata 정보가 달라질 수 있음.

[사진 metadata]

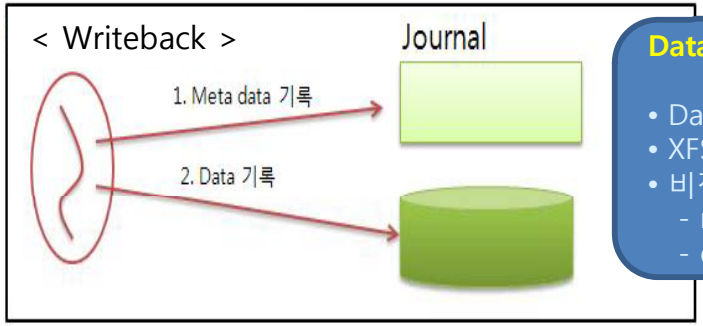
- 카메라 모델명
- 촬영일자
- 렌즈
- 조리개 값 & 셔터스피드
- 초점거리 & 측광방식
- 해상도

[파일 metadata]

- 파일 위치
- 크기
- 소유자
- 권한
- 시간 정보

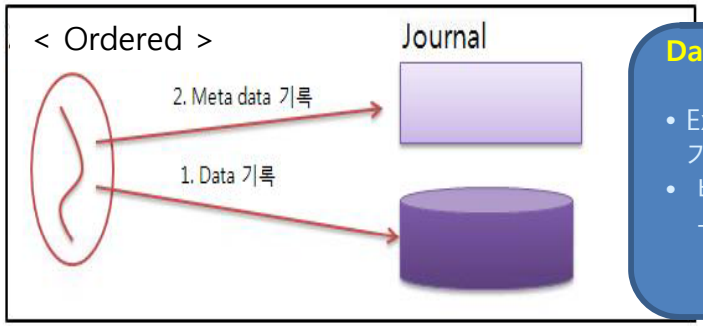


Journaling mode



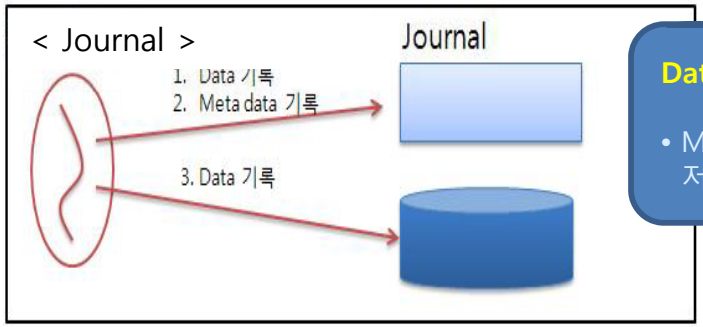
Data integrity : 최저

- Data 복구율 매우 낮음
- XFS file system의 기본 설정
- 비정상 종료 시
  - metadata 보존
  - data 손실 가능



Data integrity : 중간

- Ext4 file system의 기본 설정
- 비정상 종료 시
  - metadata가 journal에 기록되지 않게 되는 경우 data 손실 가능



Data integrity : 최고

- Metadata와 data를 두 번 저장하여 성능 저하 발생

[1]File system check : file system이 비정상적으로 종료되면, 운영체제가 이를 감지해 수행하는 검사
.

3. File System Layer

▪ File System in Android Smart Phone C

✓ File System Journaling에 의한 System Performance 향상

Test
[Journaling On/Off에 의한 Performance 성능 향상도 측정]

Data 영역 partition 실행 후, journaling off option 실행.

@ android/device/.../format_disk/format_first.sh

/system/bin/mke2fs –t ext4 –b 4096 /dev/block/mmcblk0p28
/system/bin/tune2fs –O ^has_journal /dev/block/mmcblk0p28

```
Filesystem revision #:    1 (dynamic)
Filesystem features:      has_journal ext_attr resize_inode dir_index filetype needs_recovery extent
 flex_bg sparse_super large_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         unsigned_directory_hash
Default mount options:    (none)
Filesystem state:         clean
```

```
Filesystem revision #:    1 (dynamic)
Filesystem features:      ext_attr resize_inode dir_index filetype extent flex_bg sparse_super large
_file huge_file uninit_bg dir_nlink extra_isize
Filesystem flags:         unsigned_directory_hash
Default mount options:    (none)
Filesystem state:         not clean
```

Test Result [Dual Core/ICS Version/ AndroBench/ 16GByte/ eMMC 4.41]

※ Journaling off 시 Read에 대한 약간의 성능 저하가 있으나 ***write에 대한 성능 향상*** 이 월등히 좋음.

3. File System Layer

▪ File System in Android Smart Phone C

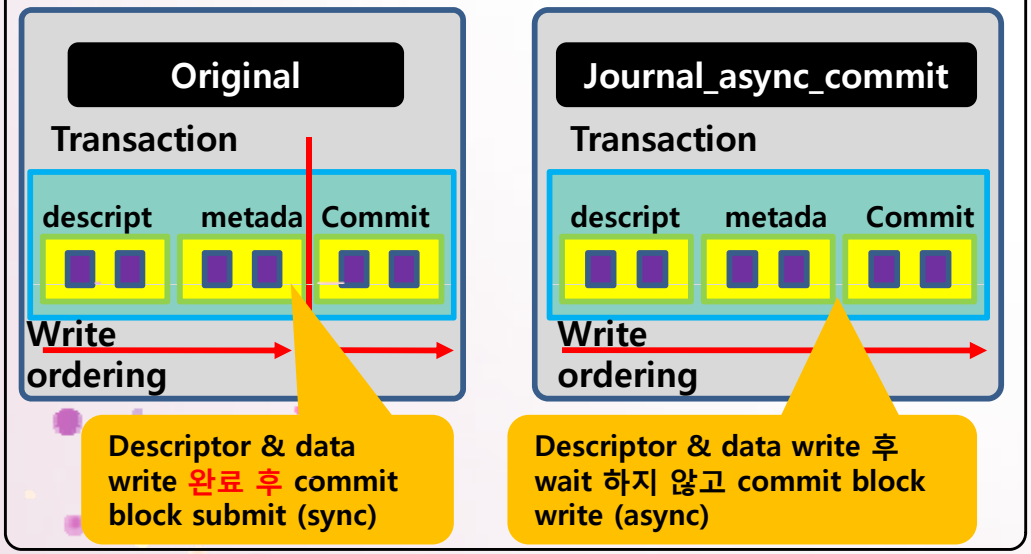
✓ Mount Option에 의한 System Performance 향상

noatime

atime이 set 되어 있는 경우 read만 하는 경우에도 inode의 acces에 의한update발생 하여 eMMC write가 됨.
noatime이 set될 경우 file의 change/ write 시에만 access time이 update되므로 performance 향상 효과가 발생.

journal_async_commit

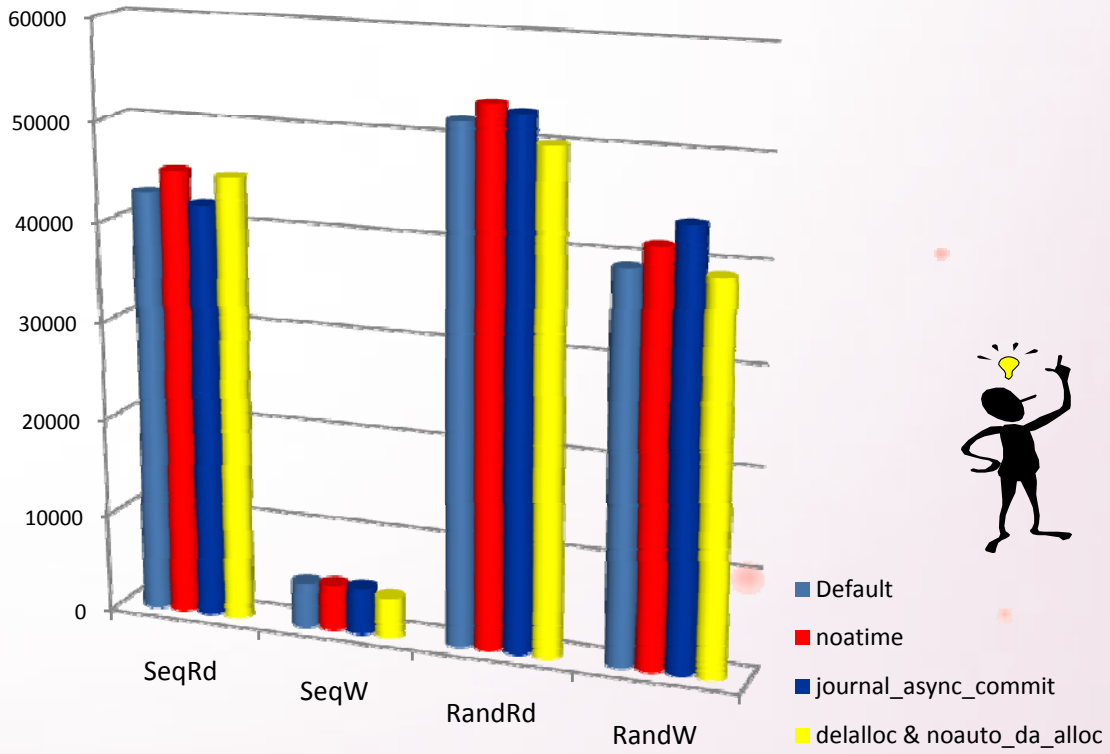
내부적으로 journal_checksum 함께 enable 함.
revoke & descriptor block의 write가 완료하는 동안 commit block의 write가 함께 이루어 지게 되어 (asynchronous하게 동작) 나중에 commit block의 write가 완료될 때까지 따로 blocking wait 할 필요가 없기 때문에 performance 좋아짐. (parallel)



delalloc ↔ nodelalloc

Block allocation을 매번 write시 하지 않고 delay하고 있다가, 나중에 disk에 write해야 하는 순간에 allocation 함.
Sequential write 인 경우에 특히 효과를 볼 수 있고, Data fragmentation 또한 방지 할 수 있음.

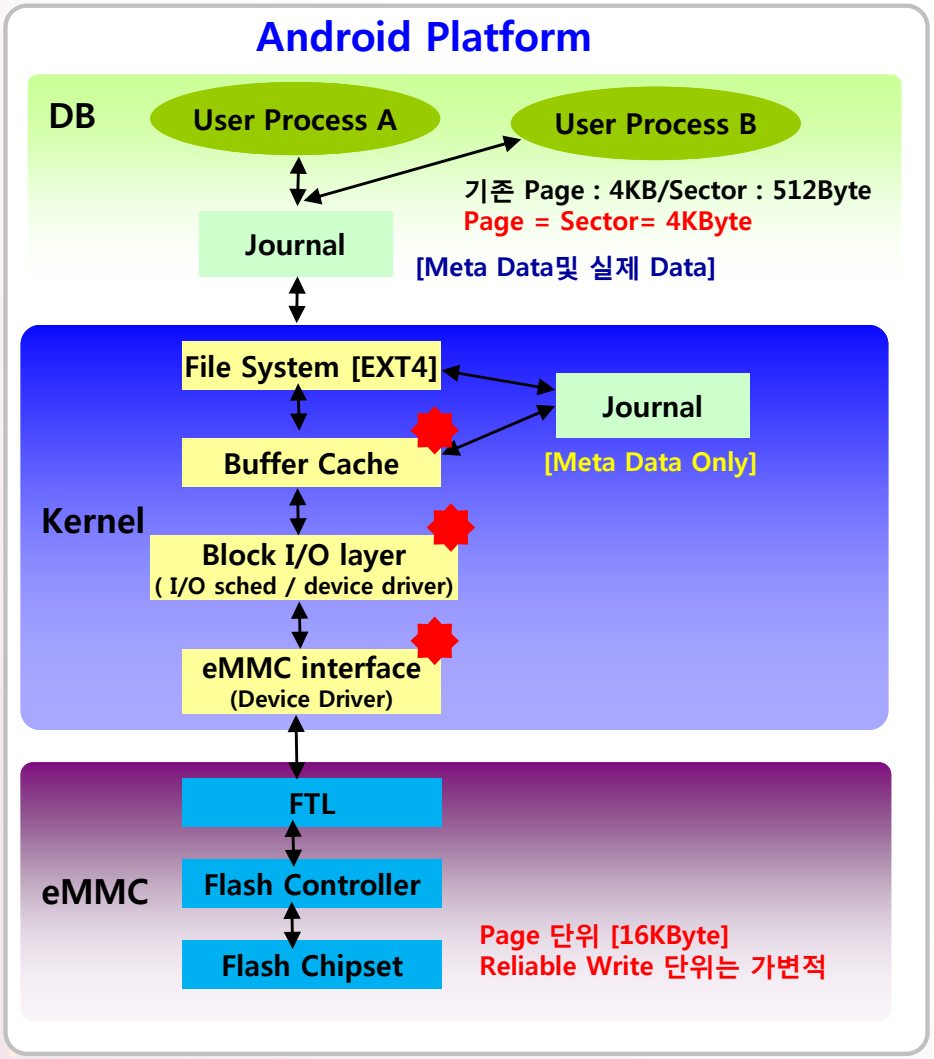
Test Result [Dual Core/ICS Version/ filebench]

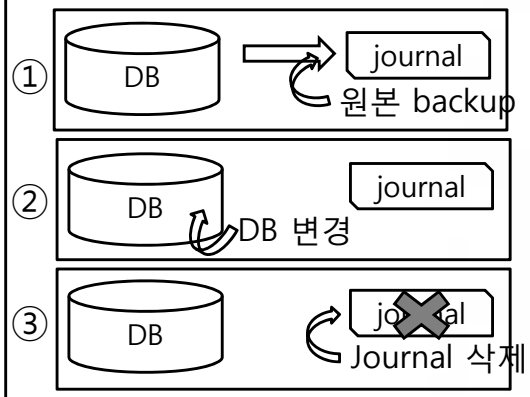


4. DB Layer

▪ DB in Android Smart Phone D

✓ DB Journaling에 의한 System Performance Effect





✓DB의 무결성 보장을 위해
journal 파일을 사용

✓Journal file에 대한 생성, 쓰기,
삭제로 인한 별도 I/O overhead 발생

어떠한 조건을 만족해야
2번, 3번 항목처럼
DB만 변경할 수 있을까?

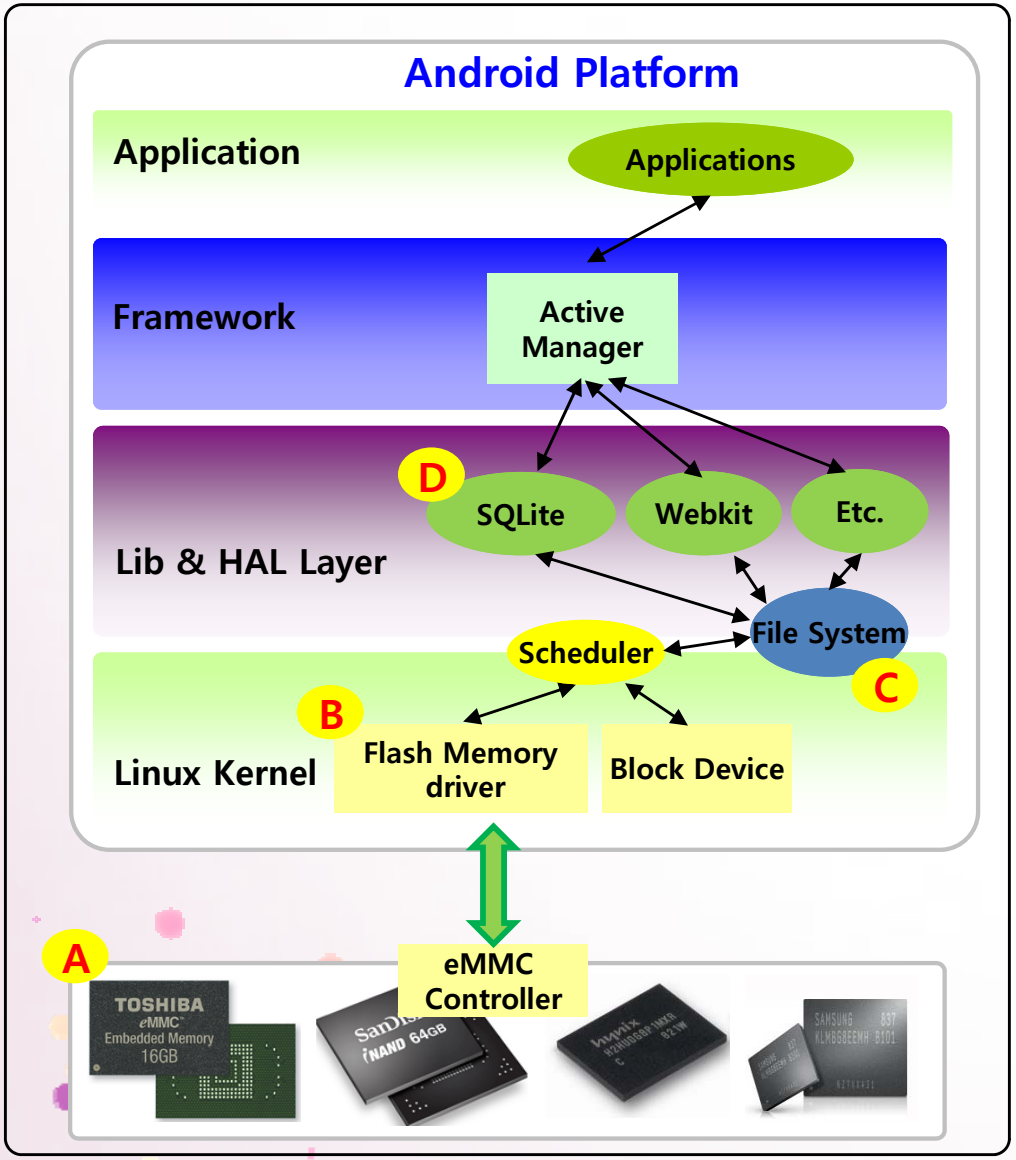
DB Journaling Off를 하기 위한 조건

1. DB의 Page Size가 Sector Size와 동일 할것
2. Database의 Page수정이 단 하나의 Database Page에서만 발생 할 것
3. Sector Write가 Atomic Operation이어야 할 것

1번 2번 항목은 DB의 수정으로 간단히 해결
3번 항목은 File System과 Flash Memory에서 많은 검증 필요

5. Summary

▪ Summary



A Flash Memory itself

1. 각 업체 마다, 사용되는 eMMC Controller가 상이함
2. 이로 인한 FTL차이로 인하여 IOPS의 편차 발생
3. Flash Memory성능을 높이기 위한 여러 가지 Technology가 시도되고 있음
   (Ex. Interleave, Multi-I/O, Low Power Serial Bus [UFS] 등)

B Flash Memory Driver

1. eMMC의 Version Up되면서 성능 향상을 위한 여러 가지 Feature 및 Command가 새로이 추가 됨
   (Ex. eMMC4.5 – HS200, Packed/Discard Command)
   (Ex. UFS – Command Queuing)

C File System

1. EXT4 File System이외에도 BtrFS를 주목 해야 함
2. Journaling에 의한 System Performance 저하
   - File System의 Journaling Off시키면 20%정도 성능 향상
     (Embedded Battery System에 적합)
3. File System Mount Option선택만 잘해도 10%정도 성능 향상

D DB

1. Journaling에 의한 System Performance 저하
   - File System의 Journaling Off시키면 40%정도 성능 향상
     (Full Journaling이기 때문)
2. DB 에서 Journaling Off 하기 위한 조건은 까다로우며 많은 Test필요 함

And More....

Q&A