# I/O Performance Improvement
## - Using ext2 in Android -

2011.04.14

Hansung Chun

# Motive

- Samsung Galaxy S
  - `/system:` OneNAND + STL + RFS, 300MB
  - `/data:` **moviNAND + RFS, 1918MB**
  - `/dbdata:` OneNAND + STL + RFS, 100MB
  - `/cache:` OneNAND + STL + RFS, 30MB

- lag-fixes for Samsung Galaxy S replace /data with …
  - OCLF: moviNAND + RFS + ext2 (loop mount)
    - http://forum.xda-developers.com/showthread.php?t=760571
  - z4mod: moviNAND + ext2 (native)
    - http://forum.xda-developers.com/showthread.php?t=845337
  - Voodoo lagfix: moviNAND + ext4 (native)
    - http://project-voodoo.org/
  - …

# Motive

- The reason for the lag
  - Different I/O workload of smart phones
  - moviNAND may not be designed for such workload
  - RFS seems somewhat inefficient
    - designed for FAT compatibility and data integrity
- The reason lag-fix is effective
  - loop mount: write buffering of ext, which reduce amount of work for RFS
  - native: bypassing RFS

# Basic idea

- Can we apply the loop mount method for RFS to other file systems?
  - How about YAFFS2?
  - Is YAFFS2 adequate for such workload?
  - If not, it will show improved performance
  - Let's try this!

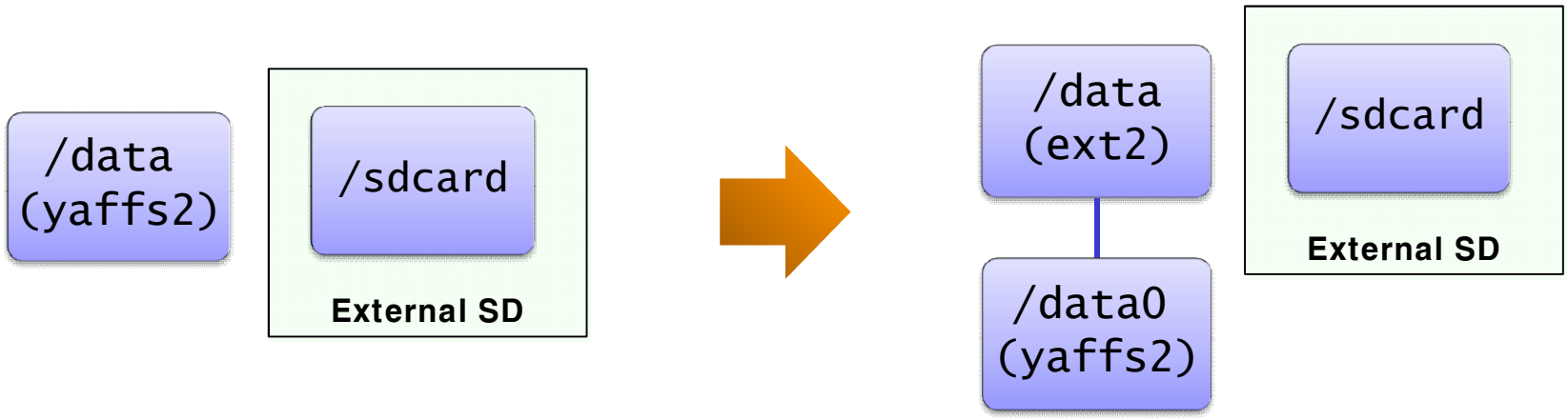
# Basic idea

- Selected device – Google Nexus One
  - Qualcomm Snapdragon 1GHz processor(QSD8250)
  - 512MB DRAM
  - 512MB MLC NAND flash memory
  - Android 2.2.1
- File system configuration of Nexus One
  - `/system`: mtdblock3, YAFFS2, 145MB
  - `/data`: mtdblock5, YAFFS2, 196MB
  - `/cache`: mtdblock4, YAFFS2, 95MB

# Basic idea

- Original Nexus One
  - MLC NAND flash memory
  - Use YAFFS2 as file system for /system, /data and /cache
- Modified Nexus one
  - Replace YAFFS2 with YAFFS2+ext2 (for /data)
  - YAFFS2 – as a wear leveling layer
  - ext2 – as a file system (in userspace view)

# Implementation Detail




Original Nexus One

Modified Nexus One

# Implementation Detail

- Create YAFFS2 image contains ext2 image
  - ext2 image fills YAFFS2 space
  - `# dd if=/dev/zero of=data.img bs=1M count=192`
  - `# mke2fs -t ext2 data.img`
  - `# mkyaffs2image ./data userdata-ext2.img`
- Flash YAFFS2 image
  - `# fastboot flash userdata userdata-ext2.img`

# Implementation Detail

- Mount with loop device
  - init.rc (before)

```
mount yaffs2 mtd@userdata /data nosuid nodev
chown system system /data
chmod 0771 /data
```

# Implementation Detail

- Mount with loop device
  - init.rc (after)

```
mount yaffs2 mtd@userdata /data0 nosuid nodev
chown system system /data0
chmod 0771 /data0
chown system system /data0/data.img
e2fsck -y /data0/data.img
mount ext2 loop@/data0/data.img /data nosuid nodev noatime
chown system system /data
chown 0771 /data
```

# Adding fsck

- ext2 needs to be checked by ‘e2fsck’ at boot time
- First, ported ‘e2fsck’ to Android
- But Android’s `init` does not support execution of arbitrary process
  - “`system/core/init/readme.txt`” describes usage of exec
  - init defines ‘exec’ command, which is not implemented yet.

- Modification
  - Implemented new command ‘e2fsck’
  - Used this command in “`init.rc`”
    - `e2fsck -y /data0/data.img`
  - Message generated by `e2fsck`: `/data0/e2fsck.log`

# Modifying system shutdown process

- Android's system shutdown process
  - 2 points for system shutdown
    - `toolbox` reboot command
    - The code which is called when power button is pressed
  - Simple process – `sync()`, `reboot()` system call
  - Write can be occur between two system calls
  - At boot time, sometimes YAFFS2 checkpoint can be invalid
    - Requires scanning flash memory , which lead to long boot time
    - Scanning time is proportional to size of used space
    - This problem is against YAFFS2 + ext2
    - We've already filled YAFFS2 space with ext2 image!

# Modifying system shutdown process

## Boot time

- If storage space is full…

| checkpoint | boot time |
|---|---|
| valid | about 35 sec |
| invalid | about 55 sec |

## Solution

- **do** `umount("/data")` **at shutdown time**
- When the YAFFS2 checkpoint is saved
  - `sync()`
  - `remount()`
  - `unmount()`

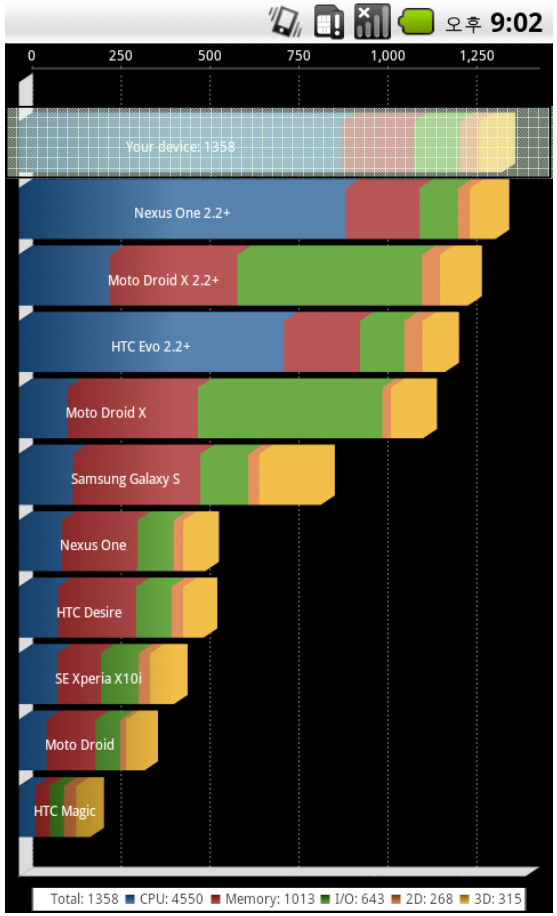
# Modifying system shutdown process

## Modification

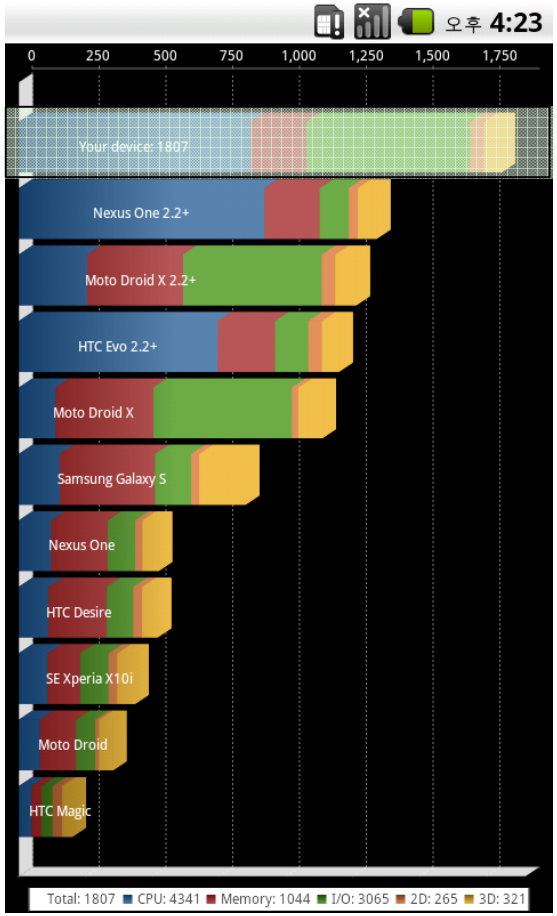
- modified Android's `init` to unmount file system at shutdown time
- modified 2 shutdown points to signal to `init`
- At shutdown points, capability of process only meets `CAP_SYS_BOOT`, not `CAP_SYS_ADMIN`
  - can't use system call `kill()`, `unmount()` …
- `init` does following operations
  - close open files, sync, process kill, unmount
- `init` opens and holds files at `/data/property/*` to handles property service

# Performance Evaluation

- Quadrant Benchmark




YAFFS2




YAFFS2 + ext2

# Performance Evaluation

- Quadrant Benchmark
  - Total score: full benchmark
  - I/O score: custom benchmark with I/O only selected

| | Total score | I/O score |
|---|---|---|
| YAFFS2 | 1340 | 620 |
| YAFFS2+ext2 | 1800 | 2440 |
| YAFFS2+ext3 | 1280 | 410 |
| YAFFS2+ext4 | 1280 | 410 |

# Performance Evaluation

- IOzone
  - 1MB file, 4KB record, without fflush & fsync
  - `# iozone -i 0 -i 1 -i 2 -i 4 -i 6 -i 7 -s 1024k`

| | write | rewrite | read | reread | random read | random write | record rewrite | fwrite | frewrite | fread | freread |
|---|---|---|---|---|---|---|---|---|---|---|---|
| YAFFS2 | 2636 | 2504 | 153912 | 118559 | 108234 | 2548 | 2659 | 2555 | 2660 | 126140 | 81631 |
| ext2 | 41942 | 126608 | 171181 | 146515 | 120685 | 115303 | 121573 | 40670 | 109297 | 138078 | 165266 |
| ext3 | 19162 | 112974 | 169452 | 145892 | 119823 | 100756 | 105175 | 14690 | 86036 | 86036 | 153918 |
| ext4 | 35771 | 63547 | 132095 | 118148 | 98111 | 59283 | 60894 | 33386 | 56487 | 112601 | 127586 |

# Performance Evaluation

- IOzone
  - 4MB file, 4KB record, without fflush & fsync
  - `# iozone -i 0 -i 1 -i 2 -i 4 -i 6 -i 7 -s 4096k`

| | write | rewrite | read | reread | random read | random write | record rewrite | fwrite | frewrite | fread | freread |
|---|---|---|---|---|---|---|---|---|---|---|---|
| YAFFS2 | 2406 | 2431 | 109205 | 132492 | 251796 | 2246 | 2483 | 2374 | 2444 | 122017 | 161514 |
| ext2 | 64034 | 130433 | 105599 | 244918 | 293055 | 345885 | 128804 | 90870 | 109387 | 145413 | 201525 |
| ext3 | 39267 | 48488 | 98112 | 225946 | 192290 | 133151 | 115107 | 43143 | 40196 | 89477 | 277299 |
| ext4 | 60322 | 88533 | 131843 | 216479 | 255154 | 199124 | 77181 | 62425 | 62425 | 128067 | 197369 |

# Performance Evaluation

- IOzone
  - 1MB file, 4KB record, with fflush & fsync
  - `# iozone -i 0 -i 1 -i 2 -i 4 -i 6 -i 7 -s 1024k -e`

| | write | rewrite | read | reread | random read | random write | record rewrite | fwrite | frewrite | fread | freread |
|---|---|---|---|---|---|---|---|---|---|---|---|
| YAFFS2 | 2752 | 2570 | 131585 | 120259 | 108234 | 2440 | 2632 | 2768 | 2768 | 147147 | 128062 |
| ext2 | 716 | 728 | 142779 | 174747 | 119403 | 683 | 103246 | 866 | 822 | 162878 | 139224 |
| ext3 | 571 | 537 | 116098 | 134208 | 103246 | 583 | 51462 | 618 | 700 | 128046 | 113737 |
| ext4 | 524 | 497 | 140988 | 171181 | 128564 | 479 | 17512 | 544 | 504 | 162872 | 140371 |

# Performance Evaluation

- IOzone
  - 4MB file, 4KB record, with fflush & fsync
  - `# iozone -i 0 -i 1 -i 2 -i 4 -i 6 -i 7 -s 4096k -e`

| | write | rewrite | read | reread | random read | random write | record rewrite | fwrite | frewrite | fread | freread |
|---|---|---|---|---|---|---|---|---|---|---|---|
| YAFFS2 | 2453 | 2440 | 197960 | 208407 | 169893 | 767 | 773 | 2421 | 2434 | 189830 | 199133 |
| ext2 | 815 | 884 | 162487 | 174536 | 291761 | 982 | 109474 | 845 | 954 | 154624 | 183602 |
| ext3 | 653 | 760 | 124850 | 204902 | 250873 | 606 | 72745 | 684 | 646 | 80852 | 209374 |
| ext4 | 588 | 538 | 159968 | 187452 | 290513 | 599 | 96281 | 487 | 486 | 154986 | 170758 |

# Performance Evaluation

- Simple micro-benchmark
  - Get the value *n*
  - Create *n* MB file
  - Overwrite *n* MB data on specific block
  - Unit of overwriting is 1KB
- Why micro-benchmark
  - With IOzone, couldn't find the reason why Quadrant score is so different between ext2 and ext3/4

# Performance Evaluation

- Micro-benchmark: create 1MB file, overwrite 1MB data
  - without fsync

| file system | Create(MB/s) | Overwrite(MB/s) |
|---|---|---|
| YAFFS2 | 2.54 | 44.40 |
| YAFFS2+ext2 | 32.74 | 21.25 |
| YAFFS2+ext3 | 17.26 | 18.59 |
| YAFFS2+ext4 | 38.41 | 20.35 |

  - with fsync

| file system | Create(MB/s) | Overwrite(MB/s) |
|---|---|---|
| YAFFS2 | 2.84 | 0.43 |
| YAFFS2+ext2 | 0.64 | 11.91 |
| YAFFS2+ext3 | 0.45 | 11.13 |
| YAFFS2+ext4 | 0.54 | 0.10 |

# Performance Evaluation

- Micro-benchmark: create 5MB file, overwrite 5MB data
  - without fsync

| file system | Create(MB/s) | Overwrite(MB/s) |
|---|---|---|
| YAFFS2 | 2.65 | 38.35 |
| YAFFS2+ext2 | 72.50 | 43.75 |
| YAFFS2+ext3 | 42.80 | 29.95 |
| YAFFS2+ext4 | 79.42 | 40.80 |

  - with fsync

| file system | Create(MB/s) | Overwrite(MB/s) |
|---|---|---|
| YAFFS2 | 2.68 | 0.44 |
| YAFFS2+ext2 | 0.86 | 17.52 |
| YAFFS2+ext3 | 0.58 | 17.19 |
| YAFFS2+ext4 | 0.58 | 0.10 |

# Conclusion

- YAFFS2+ext2 pros & cons
  - can expect high performance under Android's workload
    - especially for write operation
  - lack of journaling
    - may be vulnerable to power cut
  - inefficient structure – loop mount
    - every R/W operations is done twice(1 for ext2, 1 for YAFFS2)
  - To use full space, we fill YAFFS2 space, which cause garbage collection during write operation
    - lower write performance

# Conclusion

- ext2 is simple, light and verified
  - generally use less resource(CPU & memory) than other file systems
  - has been verified for a long time
  - The time consumed for `e2fsck` is acceptable
    - if the size of file system is small
- ext2 can be still useful
  - if it is supported by efficient wear-leveling layer