Snapshot Booting on Embedded Linux

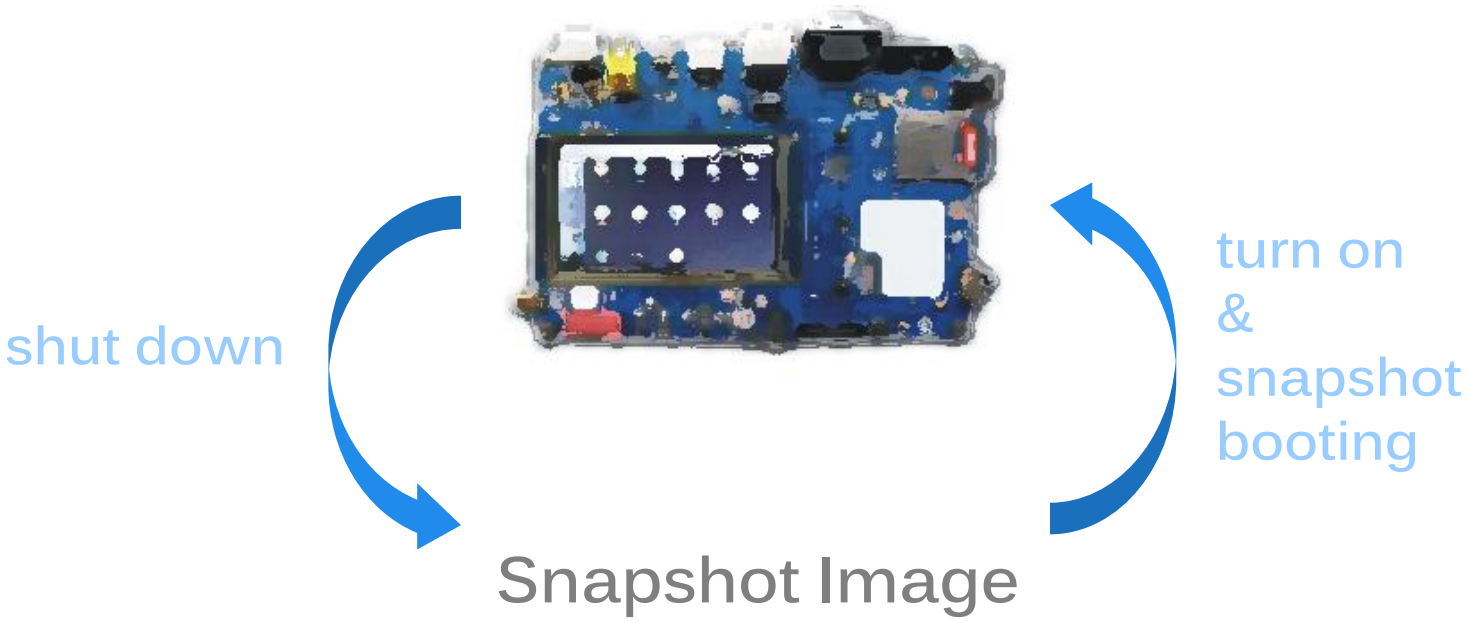
2011. 4.

Kang, Dongwook

ETRI 한국전자통신연구원
Electronics and Telecommunications Research Institute

What is snapshot booting?

- For fast boot-up time, restoring a system to a certain state with a snapshot image



What We Did

- 2009
  - Porting snapshot booting (swsusp in linux-2.6.21) to HUINS-6410 (S3C6410)
  - NAND flash read with DMA (in kernel and u-boot)
- 2010
  - Snapshot restoration in u-boot **
- 2011
  - Supporting Android
  - Supporting SD cards in u-boot

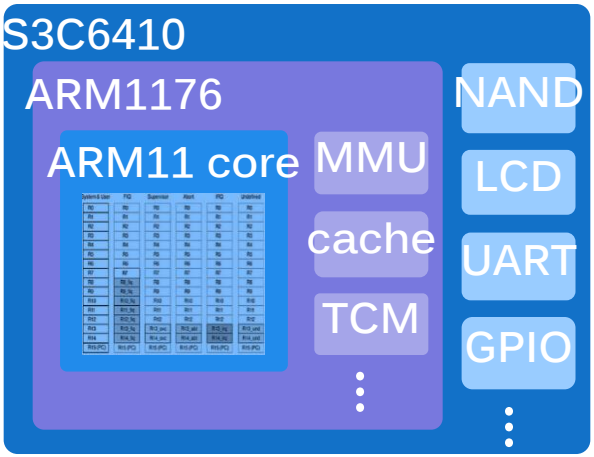
Next Topics

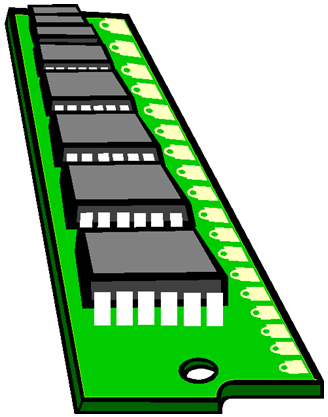
- Inside of a Snapshot Image
- Shutdown / Boot-up Processes
- NAND Flash Read with DMA
- Snapshot Restoration in U-Boot
- When is Snapshot Booting Available?

Inside of A Snapshot Image

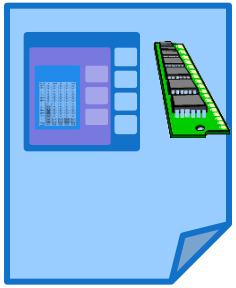
Processor + Device Controllers

Memory





Snapshot



or

Inside of A Snapshot Image

boot_cmd: resume=/dev/mtdbock3
resume_offset= 0x0

swsusp_header

swap table

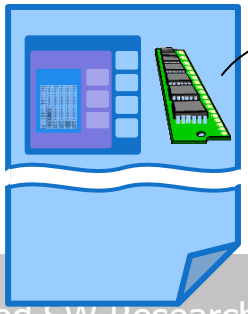
index	swap dev Page #
0	0xXXXXXXXX
1	0xXXXXXXXX
2	0xXXXXXXXX
⋮	
nr_meta_pages	0xXXXXXXXX
nr_meta_pages+1	0xXXXXXXXX
nr_meta_pages+2	0xXXXXXXXX
⋮	
nr_meta_pages + nr_copy_pages	0xXXXXXXXX

swsusp_info

pfn table

index	page frame #
0	0xXXXXXXXX
1	0xXXXXXXXX
2	0xXXXXXXXX
⋮	
nr_copy_pages - 1	0xXXXXXXXX

snapshot image



nr_copy_pages
* 4 kByte

Shutdown Process (1/2)

- Step 1. making snapshot image on memory

```
# echo disk > /sys/power/state

state_store() -> enter_state() -> pm_suspend_disk() ->
    prepare_processes()                 //freeze processes
    swsusp_shrink_memory()              //shrink used memory
    device_suspend() //suspend devices & save device controller regs
    swsusp_suspend()
        local_irq_disable()
        swsusp_arch_suspend()
            save_arch_s3c6410()         // save GPIO, VIC
            save_arch_arm11()           // save cp15 registers
            save_arch_arm11core()       // save arm core registers
                swsusp_save()           // ┬ make snapshot image
                                        // └ (copy all used pages to free ones)
            restore_arch_arm11()        // restore cp15 registers
            restore_arch_s3c6410()      // restore GPIO, VIC
        local_irq_enable()
    device_resume()  //resume devices & restore device controller regs
```

Shutdown Process (2/2)

- Step 2. writing snapshot image to NAND flash

```
pm_suspend_disk()
    ……
    swsusp_write()
        snapshot_read_next()   // ┐
        get_swap_writer()      // ├ init & write swsusp_info
        swap_write_page()      // ┘
        save_image()           // ┬ make & write swap table
                               // ├ make & write pfn table
                               // └ write snapshot image
        flush_swap_writer()    // write the last page of swap table
        mark_swapfiles()       // write swsusp_header
    power_down()
```

Boot-up Process

- Step 1. restoring memory and jump

```
System is turned on

init() → software_resume()
    swsusp_check()                          // read & check swsusp_header
    prepare_processes()                     // freeze processes
    swsusp_read()
        snapshot_write_next()               // allocate memory for swsusp_info
        get_swap_reader()                   // read first page of swap table
        swap_read_page()                    // read swsusp_info
        load_image()                  // ┬ read pfn table
                                      // ├ read & restore snapshot image
                                      // └ copy colliding pages to safe pages
    device_suspend()                        // suspend devices
    swsusp_resume()
        local_irq_disable()
        swsusp_arch_resume()
            restore_arch_arm11core()  // ┬ restore colliding pages
                                      // ├ restore arm core registers
                                      // └ jump to swsusp_arch_suspend()
```

Boot-up Process

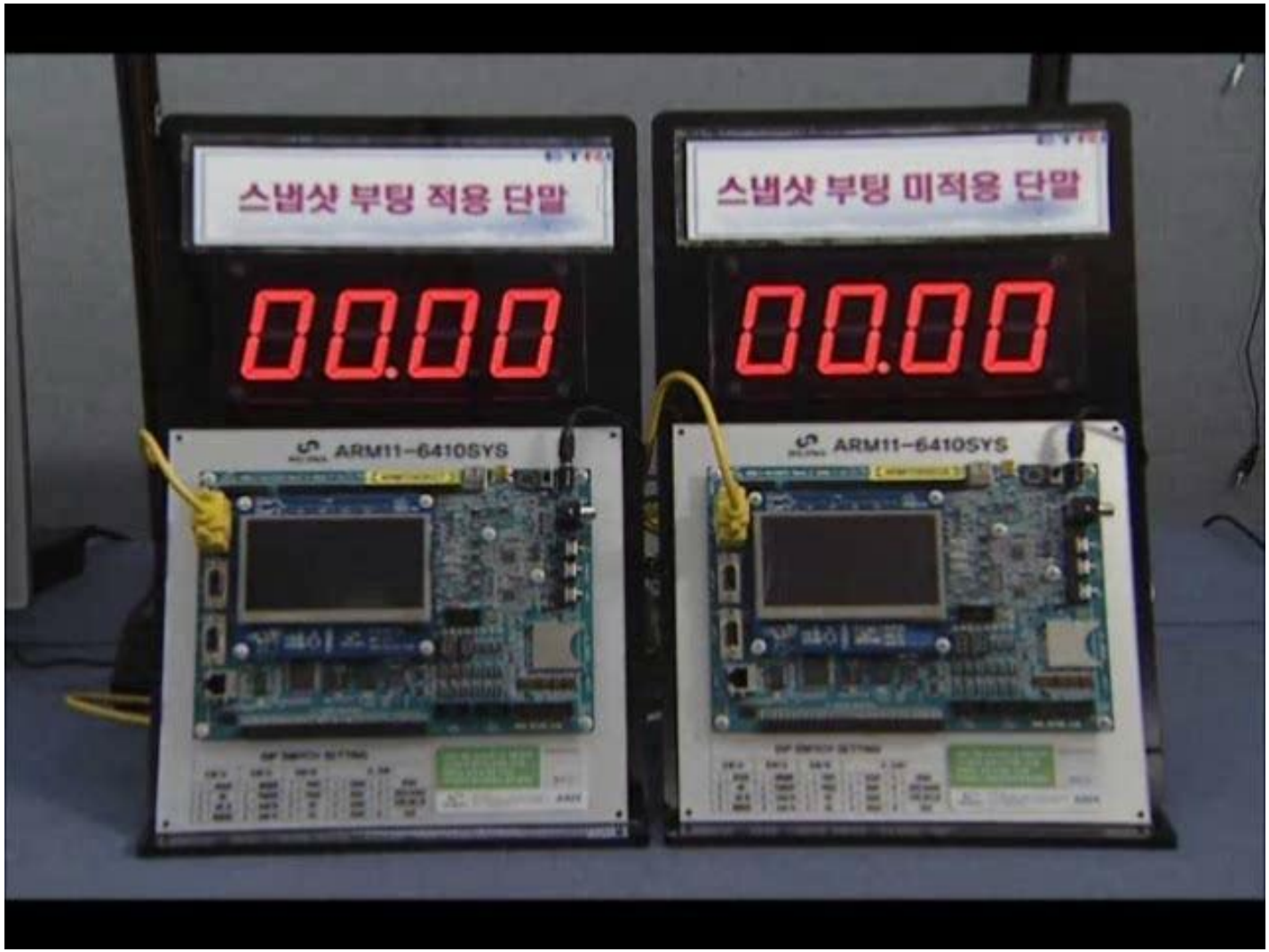
- Step 2. restoring registers

```
        (in swsusp_arch_suspend)
        restore_arch_arm11()            // restore cp15 registers
        restore_arch_s3c6410()          // restore GPIO, VIC
    (in swsusp_suspend)
    local_irq_enable
(in pm_suspend_disk)
device_resume()         // resume devices & restore device controller regs
unprepare_processes()   // wake up processes
```

NAND Flash Read with DMA

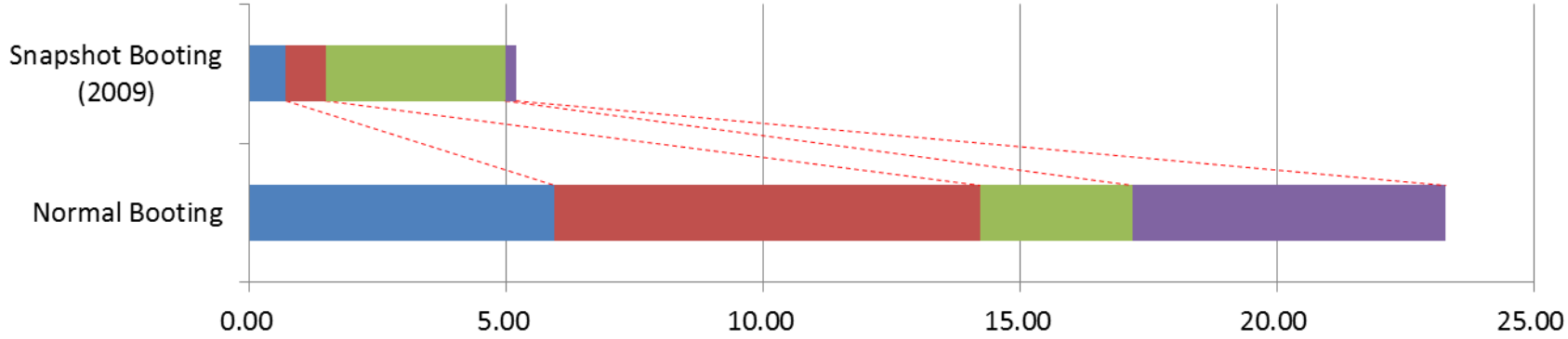
- **With DMA, its read performance improved**
  - 3.3 MByte/sec → 5.5 MByte/sec
- **This is meaningful**
  - Snapshot image loading time is the largest part
  - This time is decided by read speed and snapshot size

Experimental Results **



Experimental Results

- **HUINS-6410 Development Board**
  - S3c6410, 128M DDR, 128M SLC NAND
  - Linux-2.6.21, tinyX, matchbox
  - Snapshot image is 17MB



	Normal	Snapshot
Boot Loader	5.94	0.70
Kernel	8.29	0.78
Initial Script / Snapshot Loading	2.96	3.50
Application / Process & Dev Resuming	6.09	0.21
Total	23.28	5.19

Snapshot Restoration in U-Boot

- We can skip kernel loading and initializing
- We can load snapshot image faster **
  - No memory collision
- But, instead of kernel, u-boot need to initialize devices
  - timer, vectored interrupt controller, and other devices

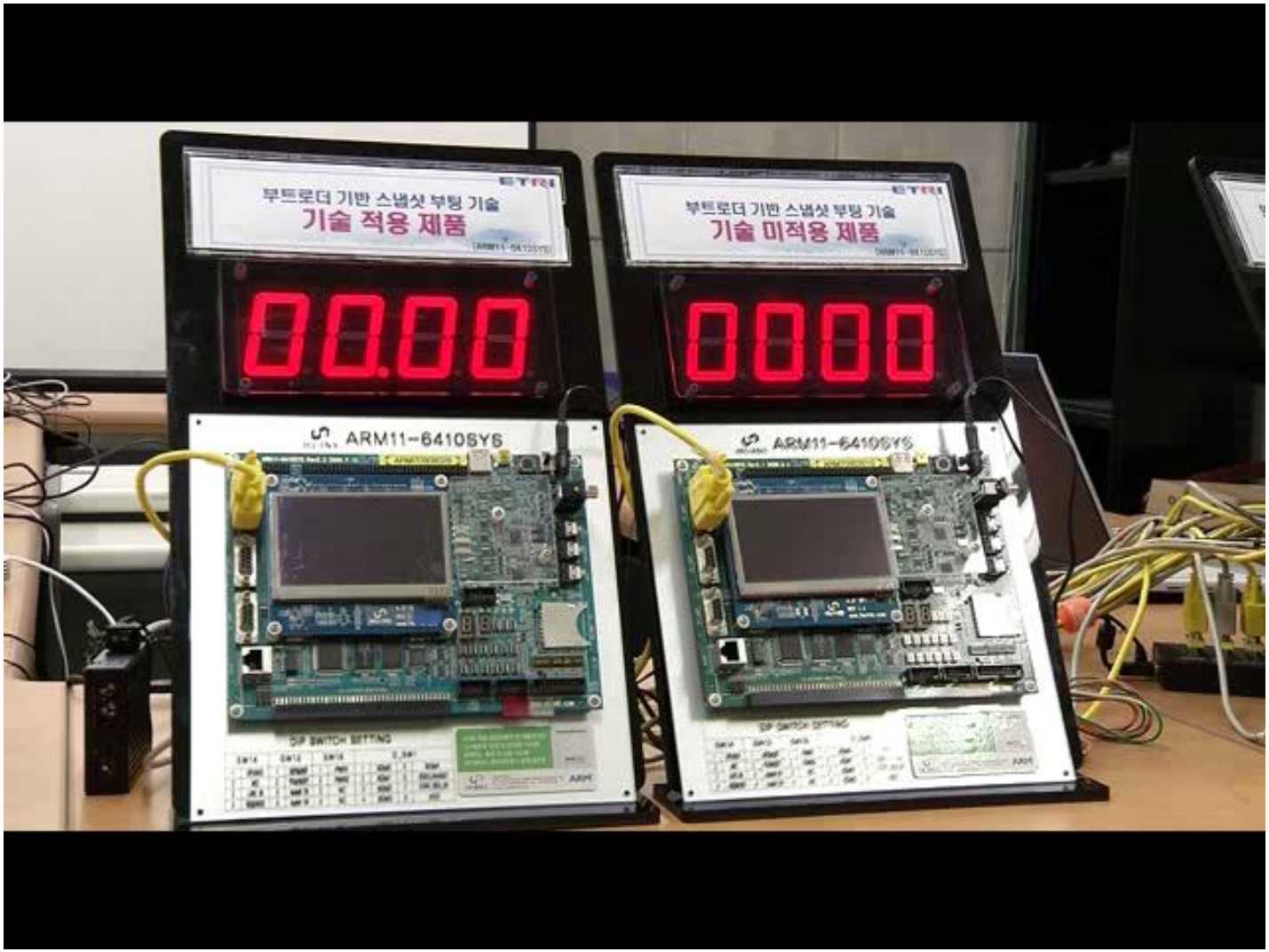
Restoration Process in U-Boot

```
u-boot# bootss 4000000

do_bootss()
    read_swsusp_hdr()          // read swsusp_header
    copy_image                 // restore snapshot image
    setup_peripherals()        // init clock, VIC, timer
    jump_to_resume()           // jump to restore_arch_arm11core()
```

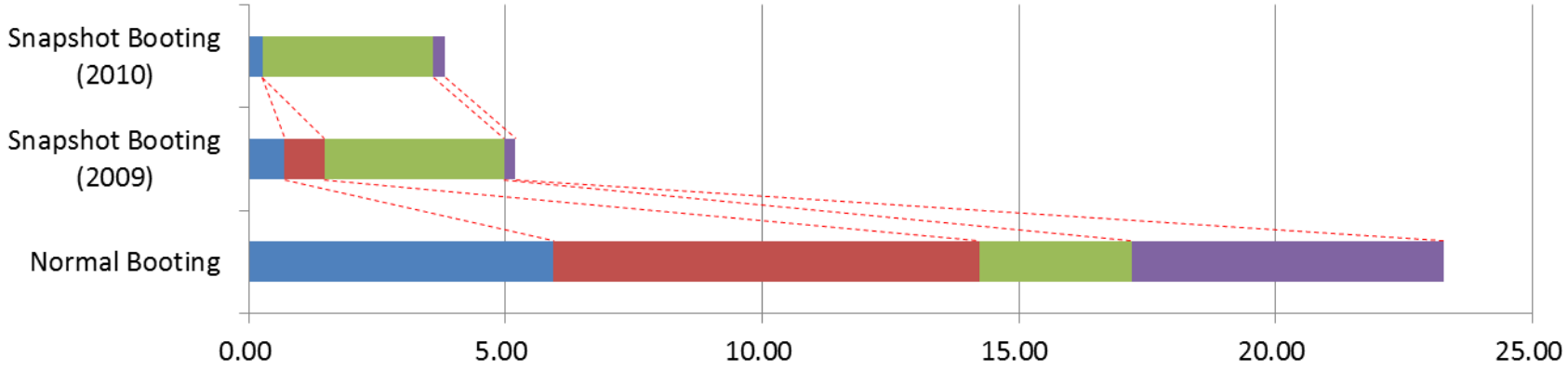
```
        restore_arch_arm11core()  // ┬ restore collided pages
                                  // ├ restore arm core registers
                                  // └ jump to swsusp_arch_suspend()
```

Experimental Results



Experimental Results

- Huins 6410sys
  - S3c6410, 128M DDR, 128M SLC NAND
  - Linux-2.6.21, tinyX, matchbox,
  - snapshot image size = 17MB



	Normal	Snapshot (2009)	Snapshot (2010)
Boot Loader	5.94	0.70	0.27
Kernel	8.29	0.78	0
Initial Script / Snapshot Loading	2.96	3.50	3.33
Application / Process & Dev Resuming	6.09	0.21	0.23
Total	23.28	5.19	3.83

When is Snapshot Booting Available?

- **When slow shutdown is OK**
  - In aforementioned case, about 10 seconds increase
- **When free memory is enough**
  - About half free pages are required
- **When NAND flash is enough**
  - In Android, the snapshot can be over 200 MB
- **When snapshot image is not too big**
  - Its size decides the boot-up time

QnA

dkang@etri.re.kr

http://www.launchphotography.com/Antarctica_Penguins.html