File System Performance Tuning For Gdium

Example of general methods

Coly Li
Software Engineer
SuSE Labs, Novell .inc

Novell®

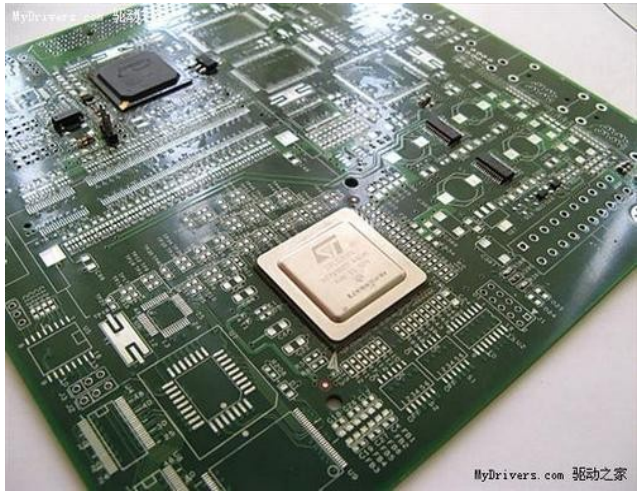
Content

- Brief Introduction to Gdium
- Storage Module of Gdium
- I/O Profiling Methods
- Key Points of File System Performance on Gdium
- General Tuning
- Tuning for Open Office
- Improved Performance Number
- Appendix Info

Brief Introduction to Gdium

- A netbook for education.
- Invested and owned by Dexxon/EMTEC.
- 10” LCD screen, 1024x600 resolution.
- Loongson2f mips64el 900Mhz by STMicroelectronics.
- 512MB DDR2 RAM, 8~16GB flash with USB port (G-key).
- WIFI 802.11b/g, Webcam, 10/100Mbps LAN
- 250 x 182 x 32 mm, 1.2kg (including battery)
- Targeting 4 hours battery life

Brief Introduction to Gdium (Cont.)



- Loongson-2f processor, designed by ICT, MIPS 64bit little endian compatible. Manufactured by STMicroelectronics, adopted by Lemote and Gdium.
- 4 ways issue, out-of-order execution. 64KB L1 iCache and dCache, 512KB L2 cache, all are 4-way set associative mapping.
- Integrated L2 Cache, DDR2 memory controller, I/O controller.

Storage Module of Gdium

- G-Key: USB port flash. 8-16GB, dynamical mapping from LBA to chip location every writing.
- Unknown controller, chip and internal structure
- Device read at 16MB/sec, cache read at 150MB/sec.
- Direct write at 600KB/sec, cache write + sync at 10.5MB/sec (vfat)
- READ size is much larger than WRITE size.
- READ ops is not much more than WRITE ops.

Storage Module of Gdium (Cont.)

- G-Key is quite slower than Winchester Harddisk and MLC SSD, especially writing performance.
- Lovely cheaper than SSD.
- Reasonable performance for web browsing, emailing, documenting, chatting.
- Do not expect highly to compile code, make movie, or depress files.
- Try to improve performance from file system, I/O block (my knowledge domain) to make
  - Launching apps faster
  - More comfortable entertainment experience

I/O Profiling Methods

- I/O profiling can help us to understand I/O accessing pattern on Gdium.
  - blktrace, blkparse, seekwatcher
  - e2block2file, filefrag (ext[234])
  - iogrind (not mentioned in this talk)
- Here are examples of I/O profiling on Gdium with blktrace

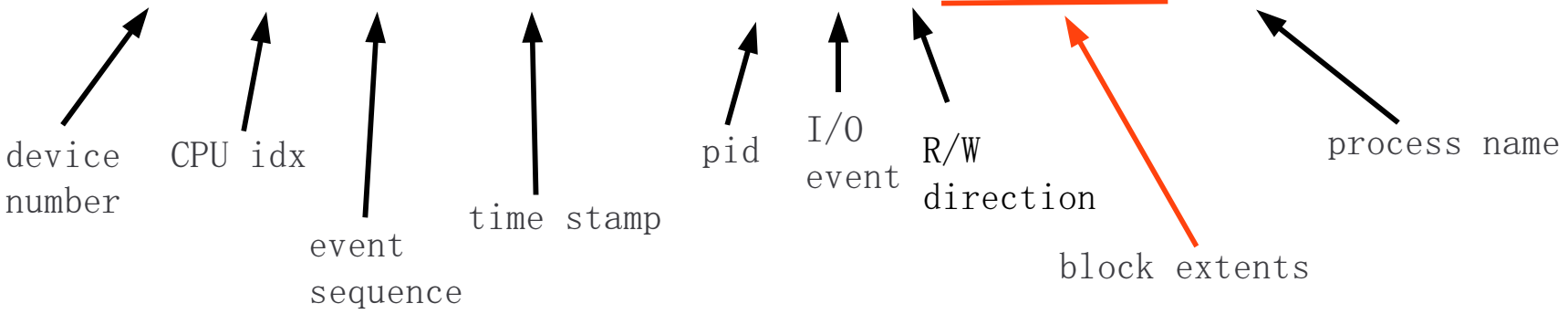
I/O Profiling Methods (Cont.)

- blktrace
  - See blktrace(8) for detailed information.
  - Capability is built in Linux Kernel since 2.6.17.
  - Package is included in most of Linux Distro.
  - Command line inserted in /etc/rc<runlevel>.d somewhere root partition is mounted as RW.
  - Place output file into a non-profiled partition.
  - The output is binary encoded file, can be converted to readable text by blkparse.

I/O Profiling Methods (Cont.)

- blkparse
  - Produce formatted output of event streams of block

```
8,2    0    20691   158.428881734     3  C   R 22559432 + 32 [0]
8,0    0    20692   158.441929445  2254  A   R 17838800 + 160 <- (8,2) 4686544
8,2    0    20693   158.441930875  2254  Q   R 17838800 + 160 [bash]
8,2    0    20694   158.441938681  2254  G   R 17838800 + 160 [bash]
8,2    0    20695   158.441948104  2254  P   N [bash]
8,2    0    20696   158.441951815  2254  I   R 17838800 + 160 [bash]
```



- R/W direction: R, W, RM, WM, N
  - N -> for Plug and Unplug events
- Block extents: start block + blocks number

I/O Profiling Methods (Cont.)

- e2block2file
  - Reverse mapping blocks to files
  - Need a block-extents file and accessing device
  - Block extents file is a list of extents which are accessed during profiling, it can be generated from blkparse output by
    - blkparse ./sda2.blktrace.0 | awk '{print $8$9$10}' | grep ^[0-9].*\+[0-9].*$
  - Resulted output is

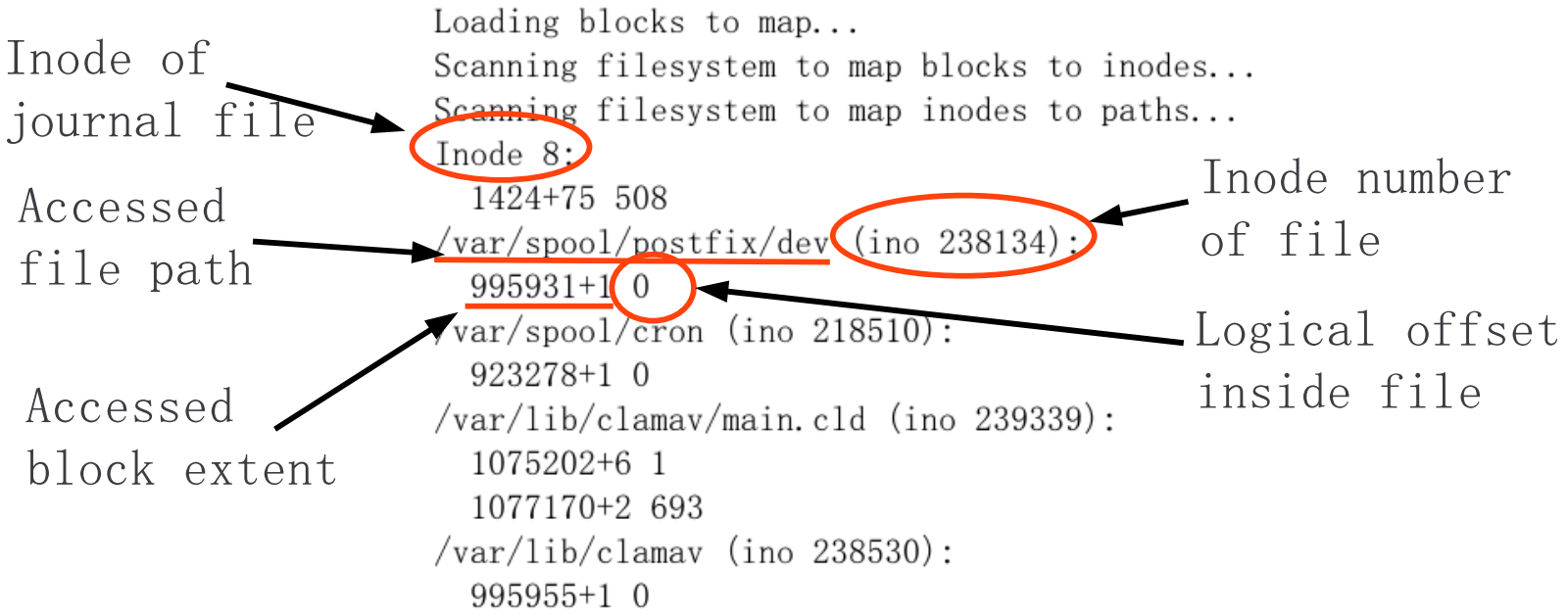
    ... ...

    26305248+8

    25048176+128

I/O Profiling Methods (Cont.)

- Run: e2block2file extents_file /dev/sda2
- The output looks like:



I/O Profiling Methods (Cont.)

- Profile result for Gdium G-Key
  - seekwatcher chart, exmaple boot_trace.png
  - Seekwatcher movie, exmaple boot_tracemovie.mpg
  - Blkparse output, example sda2parse
  - E2block2file output, example boot_block2map
  - RW sequence from sda2parse, example boot_RW_sequence

- These methods can also be used on fire fox and open office.

Key Points of File System Performance on Gdium

- No seeking on flash media
  - Reducing seeking does not help on performance any more
  - Quite a lot rules change now
- Meta data I/O merge
  - Merging possible meta data I/O still helps on performance
- Journaling overhead VS. fsck time
  - Is it worth to write data twice on flash ?
  - How long does it take to fsck a 16GB flash ?
  - Is journaling necessary for us ?

Key Points of File System Performance on Gdium (Cont.)

- Will journaling hurt flash life cycle ?
  - Writing times for flash chip is finite.
  - If flash controller can not do dynamic <LBA, chip location> mapping, the flash chip will pass away much earlier by frequent journaling.
- Meta data I/O invoked by file data I/O
  - Updates super block, directory, block group, inode or other meta data.
  - e.g. on ext[234], file data R/W might invoke multiple meta data R/W for indirect/doulb-indirect/triple-indirect index blocks.

Key Points of File System Performance on Gdium (Cont.)

- Read or write performance preferred ?
  - Gdium is a netbook, not flie server, not downloading machine.
  - Information gathering is preferred --- serving read is prior than write.
  - Interaction might be important than overall throughput.
- Bad blocks in file system
  - Repeat R/W bad blocks shooting down performance
  - On cheap USB stick, there is very chance to happen.

Key Points of File System Performance (Cont.)

- Meta data structure
  - Different meta data structure results different performance number
  - Indirect index blocks VS. extents
  - Linear dentries VS. dentries htree
  - In-line data or not
- Proper I/O scheduler in block layer
  - Flash is not RAM, noop scheduler is not a best choice
  - Check anticipatory, deadline and cfq schedulers.
  - Understand the I/O pattern on specific application

Key Points of File System Performance (Cont.)

- Dynamic linking in program starting up
  - Learned from fire fox and open office I/O profile, huge number of I/O spent on .so file loading.
  - It's possible to preload necessary data into memory in stolen time.

General Tuning

- No journaling
  - Avoid extra I/O for check pointing
  - For 16GB flash, fsck is quite faster than expected even without journaling.
  - **Decision: Use non-journaling file system (ext2, vfat, ...)**
- Avoid indirect index blocks
  - For write-once data (e.g. base system), extent helps to avoid extra I/O than indirect index block or file allocation table.
  - **Decision: Use Ext4 (not ready, dropped)**
-

General Tuning (Cont.)

- Minimize I/O in dentry searching
  - Non-linear searching avoids extra dentries reading in large directory file.
  - For small or medium size directory file, linear searching is acceptable.
  - Hashed path in bash helps file searching
  - e.g. in /usr, block size 4KB:
    - 1 block sized directories 98%
    - 2 blocks sized directories 0.008%
    - > 4 blocks sized directories 0.005%

General Tuning (Cont.)

- e.g. still in /usr,
  - 84% directories have no more than dentries
  - 0.1% directories have more than 256 dentries
- **Decision: most of directories are small, non-linear searching is not necessary.**

use ext[234] or vfat

General Tuning (Cont.)

- Use better I/O scheduler
  - Flash W is quite slow, might starve R (web browsing).
  - Noop scheduler does not work as well as it does on RAM disk.
  - **Decision: cfq or deadline, cfq is preferred.**

- Tuning decision:
  - **Replace ext3 with ext2**
  - **Replace ext3/vfat with ext4, when ext4 can work without journaling.**
  - **Use cfq as I/O scheduler**

Tuning for Open Office

- Use similar profiling methods
  - Seekwatcher: oo_trace.png oo_trace.mpg
  - Blkparse: oo_parse
  - E2block2file: oo_block2file
- Before tuning, when oo starts on Gdium,
  - Accesses 465 files, Loading blocks from 96 .so files
  - When opening od[tpgx] files, more file data and meta data I/O
- Decision
  - **Preload dependent files (font, .so, etc.) into memory**

Improved Performance Number

- Start oo
  - Before tuning, 450 inodes and 96 .so files fetched, 19 seconds
  - After tuning, 290 inodes and 57 .so files fetched, 12 seconds.
- Remove 65536 files (data=journal)
  - Before tuning, 36.6 seconds.
  - After tuning, 3.8 seconds

Improved Performance Number

- Surfing http://news.sina.com.cn with FireFox
  - Before tuning, web page display slowly, a little starving.
  - After tuning, web page display faster, little starving.

- **The number is not bad :)**

Appendix Info

- Thanks to
  - Jens Axboe for blktrace
  - Chris Mason for seekwatcher
  - Jan Kara for e2block2file.
- Alexander Beregalov is working on a patch, which tries to make ext4 work without journaling.
- Everyone wants btrfs
  - Not mentioned in this talk.
  - let's talk about it over beer :-)

Thank you !