Development of a New Log-structured File System for Linux

Ryusuke Konishi

NTT Cyber Space Laboratories

October 19, 2005

Motivation

- Goals for the Linux file system
  - Reliability and recovery time

    → Improved significantly by journaling file systems.
    But still have problems and recovery can fail.

Recent reliability patches by Hifumi (NTT)	
• Ext3:	3 patches
• ReiserFS:	2 patches

  - Online snapshot: easy recovery of past data
    - Commercial storage systems are costly

Another Approach: Log-structured File System

- LFS appends modified data instead of overwriting
  - Fast recovery: comparable to journaling file systems
  - Suits data salvage and snapshot; improves restorability (e.g. covers operational errors)
  - Fewer seeks; high write performance

Aim of NILFS project:
offering LFS as an alternative for Linux

NILFS = **N**ew **I**mplementation of a **L**og-structured **F**ile **S**ystem

Related Projects

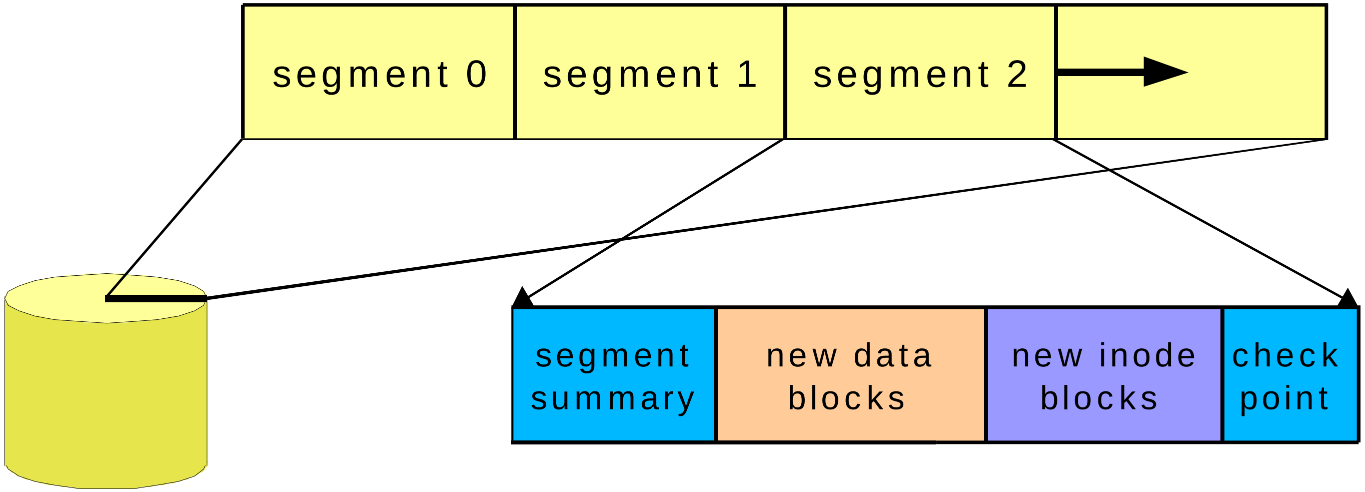
- LinLogFS
  - Implemented for kernel 2.2
  - Abandoned
- LFS with snapshot by Pradeep Padala
  - Similar aim; we desire to cooperate
- Others
  - BSD-LFS (available for 4.4BSD, NetBSD)
  - Commercial products (e.g. NetApp WAFL)

Development Goals of NILFS

- Satisfy OSDL DCL technical capabilities
  - Reliable file system writes
  - Reliable file system operation
- Improve operability with snapshots support
- Satisfy both performance and reliability
  - Taking advantage of LFS on Linux

What is LFS? (1)

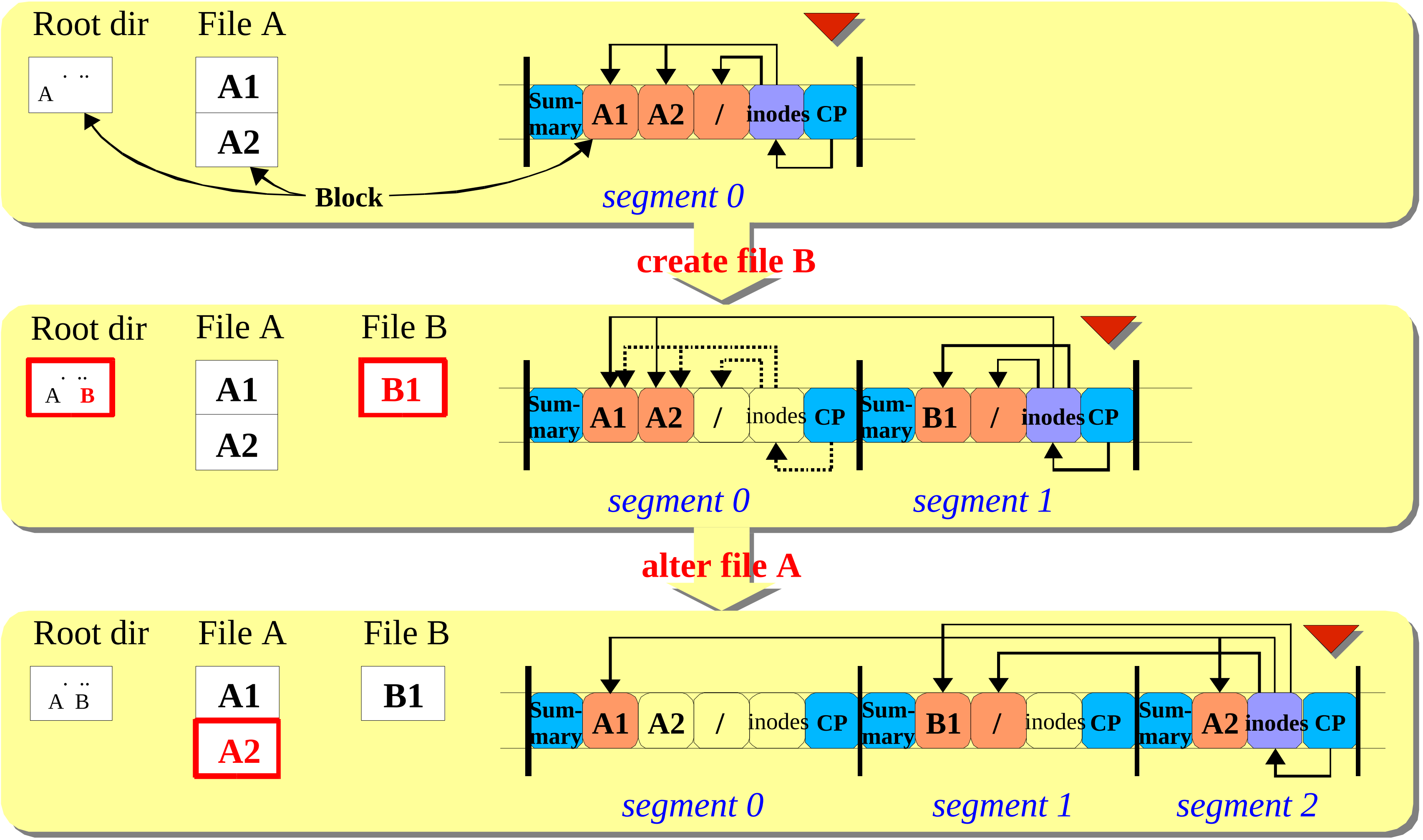
- Disk layout



- Disk write in LFS
  - Modified data and meta-data are written in empty segments
  - Occurrence of Intermediate state is avoided by DB-like techniques (i.e. check-pointing)

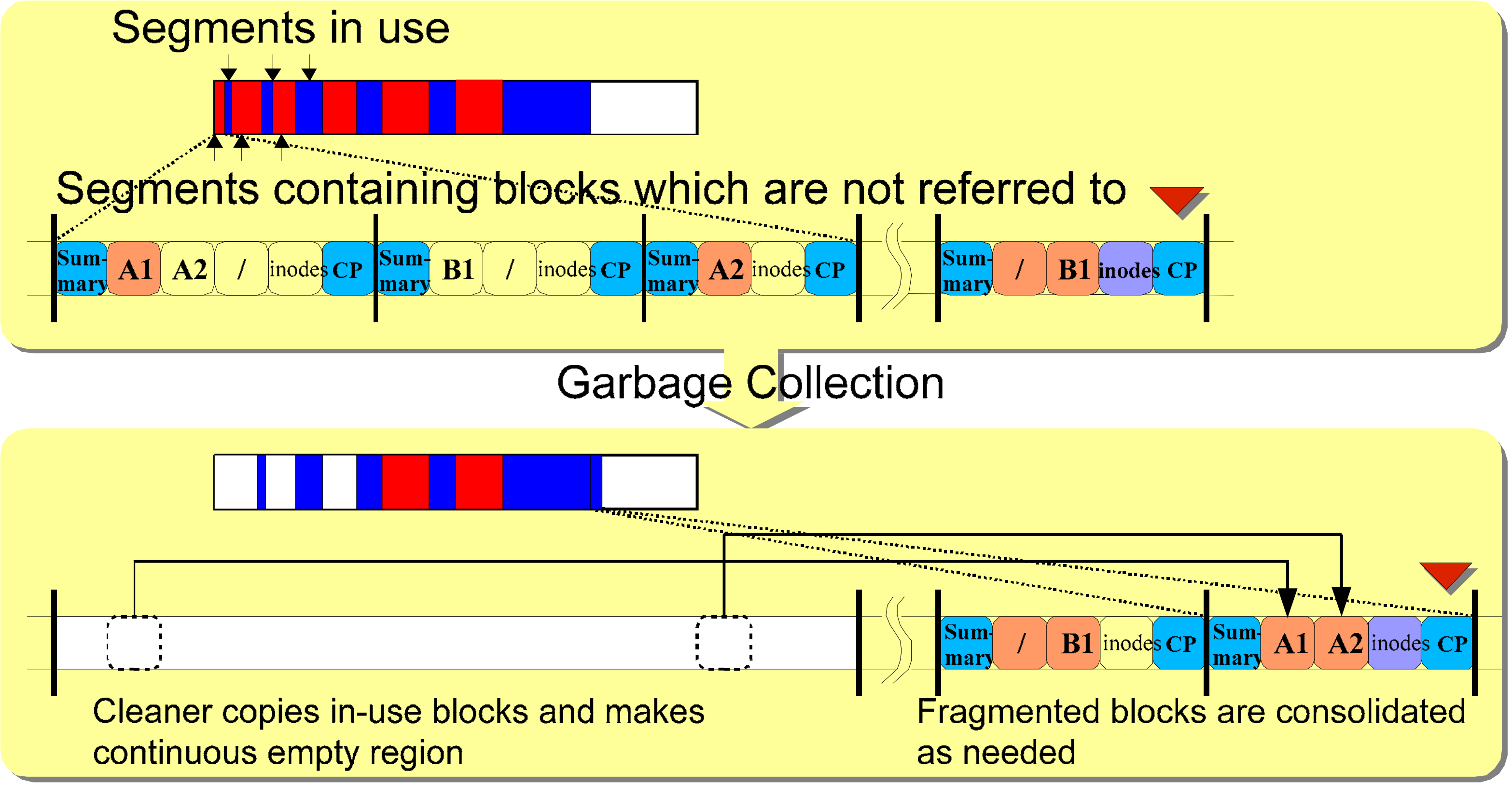
What is LFS? (2)

- LFS Write



What is LFS? (3)

- Garbage Collection (Cleaner)
  - Reuse of segments while writing
  - A key challenge in LFS with snapshot



Cleaner copies in-use blocks and makes continuous empty region

Fragmented blocks are consolidated as needed

Features of NILFS (1)

- Simultaneously mountable snapshots
  - Allow users and tools to enter past directory trees (i.e. time domain extendable namespace)
  - Can clip consistent state; help online backup
- Immediate recovery after system crash
  - Safer recovery without overwriting meta data
- Complies with Linux FS-semantics

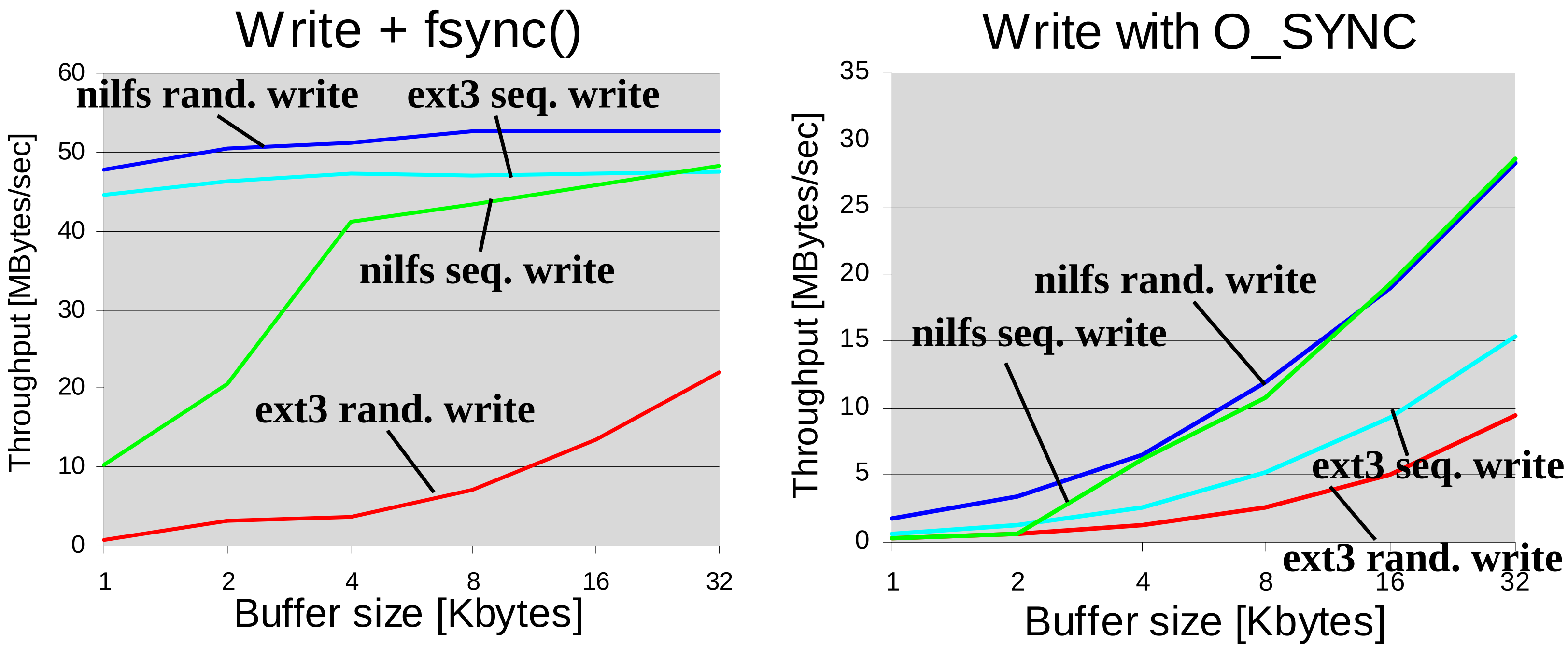
Features of NILFS (2)

- B-tree based file and inode management
  - Enable fast lookup
  - Adopted in modern file systems, but difficult to implement for LFS because blocks are relocatable
- 64-bit data structures
  - support many files, large files and disks.
- Loadable kernel module
  - no recompilation of the kernel is required.

Snapshots: Example of Use

```
# mount –t nilfs /dev/sda3 /home
…
# inspect /dev/sda3
…
nilfs> listcp
 137686    66 Wed Oct  5 14:48:51 MajorCP|LogiBegin|LogiEnd
   …
 150528  1852 Wed Oct 15 14:54:01 MajorCP|LogiBegin|LogiEnd
   …
# mkdir /home-last-week /home-5-minutes-ago
# mount –t nilfs –r –o cp=137686 /dev/sda3 /home-last-week
# mount –t nilfs –r –o cp=150528 /dev/sda3 /home-5-minutes-ago
```

Write Performance



- Measured by iozone; measurement environment and conditions are as follows:

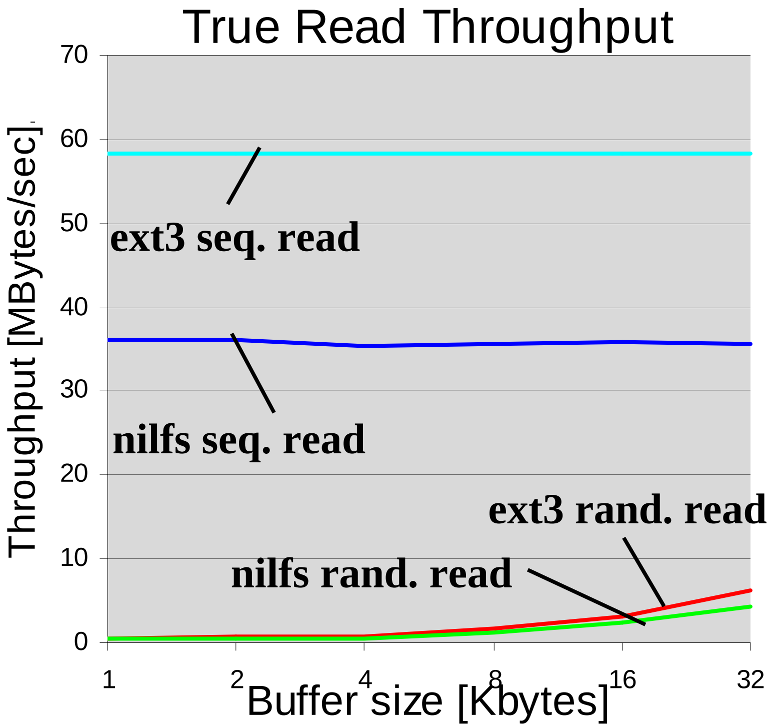
Measurement PC

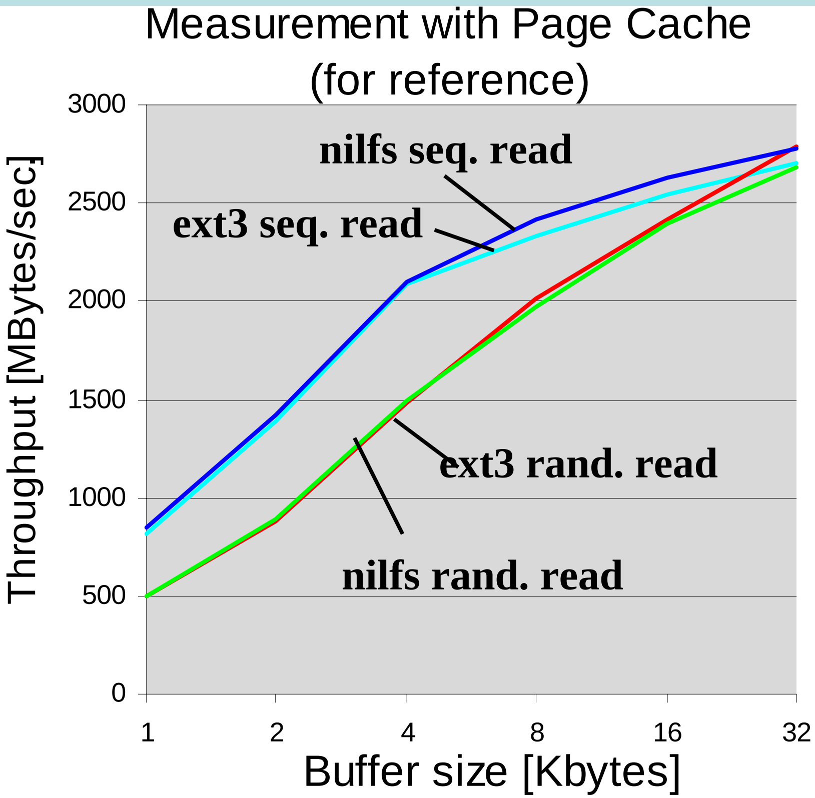
- CPU: Pentium 4 3.0GHz
- Memory: 1GBytes
- Disk: IDE (Ultra-ATA 7,200rpm)

Condition of Comparison

- Ext3 journaling mode: Ordered (default)
- No garbage collection (NILFS)
- Kernel version: 2.6.13
- Random write: Overwrite
- Sequential write: New creation
- File size: 512MB

Read Performance





- Measured by iozone; measurement environment and conditions are as follows:

Measurement PC

- CPU: Pentium 4 3.0GHz
- Memory: 1GBytes
- Disk: IDE (Ultra-ATA 7,200rpm)

Condition of Comparison

- Ext3 journaling mode: Ordered (default)
- No garbage collection (NILFS)
- Kernel version: 2.6.13
- File size: 512MB
- Cache is flushed by umount/mount for true read

NTT

Development Status

- Implemented
  - – Basic operations
    - mount(), umount(), open(), read(), write(), fsync(),…
  - – Basic snapshot functions
  - – Simple roll-backing and roll-forwarding
- Not implemented
  - – Cleaner (GC)
  - – Snapshot management mechanism
  - – B-tree base directory management
  - – Performance tuning (e.g. read ahead)
  - – And all the rest …

Further Development Plans

- Project has just begun
  - “Time-domain” tools (e.g. tls, tdiff, tgrep, tfind, ttar and so on)
  - GUI tools
  - Real “delete” function
    - wipes out past data to enhance security
  - Efficient synchronous write operations
    - fsync(), open() with O_SYNC, and so on
  - GNU GRUB support
  - And more …

Status of Distribution

- Licensed under GPL
- Downloadable from the NILFS Homepage
  - primary: http://www.nilfs.org/
  - mirror: http://nilfs.sourceforge.net/
- NILFS ML (in preparation)
- Positive response
  - Picked up by many domestic and overseas news sites
  - We'd like to activate discussions in LKML

Future Plans

- Continue the development
  - frequent updates
- Involve outside developers
  - Currently several hackers in NTT Lab.
- Promotion
- Shoot for merge into the mainline kernel