Utilizing NILFS2 Fine-grained Snapshots

Ryusuke KONISHI

NTT Cyberspace Laboratories
NTT Corporation
Outline

• Nilfs2 overview
• Fine-grained Snapshots - Why?
• Use-case scenario and applications
• Work in progress on Snapshots
• Current status and future plan
NILFS2 Overview

• A mainlined filesystem (since kernel 2.6.30)
• A log-structured filesystem
  – Filesystem itself is a big journal
  – Ensure consistency and quick recovery from unexpected power failure.
• Stand for fine-grained and “any time” snapshots
  – Creates a number of checkpoints every time user makes a change.
  – Can change arbitrary checkpoints into snapshots later on.
  – Snapshots are concurrently mountable and accessible.
Fine-grained Snapshots - Why?

- Backup is necessary to prevent data loss, but it still accompanies inconvenience and pain.

CAUSE OF DATA LOSS

- Hardware failure 59%
- Human error 26%
- Software malfunction 9%
- Viruses 4%
- Natural disaster 2%

Unprotected by redundant drives.

Mostly preventable with basic high-integrity system (Redundant configuration)

Source: Ontrack Data Recovery, Inc.
Including office PC. The data is based on actual data recoveries performed by Ontrack.
Solution with NILFS

• **Buffer filesystem history in disk.**
  • User can even restore files mistakenly overwritten or destroyed just a few seconds ago.

Usual filesystems

Backup

 ex. per 8 hours

Previous data is overridden

NILFS

Automatically taken

Easily accessible

Previous data is preserved in disk
Disk write in NILFS

- **Only modified blocks are incrementally written to disk (in CoW)**
  - Even for metadata and B-tree intermediate blocks as well as data.

Application view

File A (modified)

File B (appended)

Disk usage

- B-Tree intermediate blocks
- Metadata blocks (inodes, ...)

Modified or appended blocks
Garbage Collection

• Creates new disk space to continue writing logs (essential for LFS)
• NILFS2 employs a unique GC which can reclaim disk space keeping selected checkpoints.
  – This makes checkpoints long-term storable in arbitrary granularity that user demands.

A checkpoint which user marked as SNAPSHOT are preserved

Recent checkpoints are preserved, too.
Command Line Programs

Tools are included in nilfs-utils (or nilfs-tools for Debian/Ubuntu) package

- **Snapshot management programs**

  - `lscp` list checkpoints
  - `lscp -s` list snapshots
  - `mkcp -s` make a snapshot
  - `chcp` change an existing checkpoint to a snapshot (or vice versa)
  - `nilfs-clean` manually trigger garbage collection

```bash
$ lscp
  CNO   DATE       TIME      MODE FLG  NBLKINC ICNT
1  2011-05-08 14:45:49  cp     - 11          3
2  2011-05-08 14:50:22  cp     - 200523         81
6  2011-05-08 20:42:00  cp     - 37       1653
7  2011-05-08 20:42:42  cp     - 272146       2116
8  2011-05-08 20:43:13  cp     - 264649       2117
9  2011-05-08 20:43:44  cp     - 285848       2117
10  2011-05-08 20:44:16 cp     - 139876       7357
...```
Use-Case Scenario

• Casual data protection
  – Prevent data loss against operation mistake, even if you have NOT taken snapshot.

• Versioning
  – Make change history on files browsable.

• Tamper detection and recovery
  – Filesystem itself preserves full-time and overall range of change history
    -- track changes using the filesystem.

• Upgrade / Trouble shoot
  – Can revert system state against unexpected troubles. NILFS does not need taking a snapshot before every upgrade nor conf-file editing.
TimeBrowse Project

- A GNOME Nautilus extension applying NILFS
  - Allow browsing change history of documents and restore its arbitrary version.

http://sourceforge.net/projects/timebrowse

Browsable document change history.

User can confirm content of each version through a thumbnail image.
Snapshot Appliance

- **Example: in-house shared storage server**
  - Files are restorable even if other users edited or deleted (like Wiki).
  - Seamlessly accessible from Windows clients.
  - We actually have one and a half years operation record.

Snapshots become browsable on Windows by configuring volume shadow copy VFS module in Samba.
Tamper Detection

Typical approaches

**Notification**
- Real-time
- Example: inotify

**Database + Rule set**
- Rich auditing capabilities
- Tripwire, AIDE, etc...

Fine-grained snapshots
- Can closely track the evidence of intrusion and tampering after the fact, as well as their progress.
- Quick and accurate restoration from the local disk.

Tripwire is a registered trademark of Tripwire, Inc.
Development Focus

Establish fine-grained snapshots and make it ready for use

Enhance support for remote backup and disaster recovery
- Efficient delta extraction, restoration, de-dupe.
- Data security (e.g. shredding), anti-tampering.
• **Problem** (user’s demand)
  - It takes too long to find out changes on filesystem for thousands of snapshots. Users want to shorten the time:
    - Incremental remote backup
    - Search index rebuild
    - Tamper detection

• **Current effort**
  - Proposing experimental API which quickly looks up changed inodes between two checkpoints.
WIP - Snapshot diff (2/4)

• Approach
  • Compare b-trees of “ifile” (metadata storing NILFS2 inodes), then scan modified inodes in the ifile blocks whose disk addresses differ.
WIP - Snapshot diff (3/4)

• **API (testbed)**
  - **NILFS_IOCTL_COMPARE_CHECKPOINTS**
    - Acquire inode numbers of modified inodes.
  - **NILFS_IOCTL_INO_LOOKUP**
    - Lookup pathname of the inodes by inode number.
    - Implementing this ioctl has impact on disk format, and also hard links are not handled at present.

• **Command line tool**

```
nilfs-diff [options] [device] cno1..cno2
```

*Checkpoint numbers*
WIP - Snapshot diff (4/4)

Time required to compare two directories/snapshots containing linux-2.6.39 source code that one file differs

<table>
<thead>
<tr>
<th>Comparison method</th>
<th>Time (seconds)</th>
</tr>
</thead>
<tbody>
<tr>
<td>diff:1 -Nqr snapshot-a/ snapshot-b/</td>
<td>56.5 ( \times 209 ) faster</td>
</tr>
<tr>
<td>diff:2 -Nqr snapshot-a/ snapshot-b/</td>
<td>10.2 ( \times 38 ) faster</td>
</tr>
<tr>
<td>nilfs-diff</td>
<td>0.27</td>
</tr>
</tbody>
</table>

diff:1 -- modified diff which does not skip comparison even if device numbers and inode numbers equal.
diff:2 -- optimized diff which skips comparison if **inode numbers and ctimes equal**.

Hardware specs: Processor: Xeon 5160 @ 3.00 GHz x 2, Memory: 7988MB, Disk: IBM SAS SES-2
WIP - Revert API (1/4)

• **Problem** (user’s demand)
  • Recovery may fail due to disk space shortage because each file is copied.
  • Restoring many files or media files takes time, which also leads to availability loss in business systems.
    – Recovery of large user data
    – Recovery against system upgrade failures
    – Recovery from tampering

• **Current effort**
  • Recovery of past data without duplication.
• **Approach (preliminary)**
  - Deleted block of NILFS is not actually discarded; just its lifetime is marked ended.
  - Revive blocks that we want to recover, and reuse them.

![Diagram showing the approach of WIP - Revert API](image)

Lifetime of blocks:
- **File A created**
- **File A deleted**
- **Blocks of “File A” reappear by extending their lifetime.**
WIP - Revert API (3/4)

• API
  • in preparation -- Is it reflink?

• Command line tool (testbed)

  `nilfs-revert [options] source-file file-to-be-reverted`
WIP - Revert API (4/4)

Time and disk space required to recover a 2GiB size file

<table>
<thead>
<tr>
<th>Restore method</th>
<th>Time (seconds)</th>
<th>Capacity growth (GiB)</th>
</tr>
</thead>
<tbody>
<tr>
<td>cp</td>
<td>84.6</td>
<td>2.04</td>
</tr>
<tr>
<td>nilfs-revert</td>
<td>1.1</td>
<td>0.016</td>
</tr>
</tbody>
</table>

0.8% overhead comes from update of 32 bytes metadata per disk block

Hardware specs: Processor: Xeon 5160 @ 3.00 GHz x 2, Memory: 7988MB, Disk: IBM SAS SES-2
Current Status

• Not so many enhancement for the kernel code. Only noticeable changes are:
  – Online resize, fiemap, discard, and performance tuning, etc.

• Advancement in userland support
  – Now bootable from GRUB2
  – util-linux-ng (libblkid) recognizes NILFS2 partitions.
  – Palimpsest/udisks (GUI disk utility), parted, and so on.

• nilfs-utils 2.1
  – Contains resize tool and easy-to-use GC tool/library.
TODO items / Future Plan

• Snapshot diff and revert API
• Efficient remote replication and restoration
• Security
  – Past file shredding
  – Transient vulnerability frozen in snapshots
• Remaining essential features
  – Extended attributes, POSIX ACL
  – Fsck
• Performance improvement
  – Log writer, GC, directory lookup, inode allocator, etc...
  – Fast and space-efficient caching of inodes and data pages against many snapshot mounts
• Kernel space Garbage Collector
Questions?
We welcome your contributions

• Mailing-list
  – linux-nilfs <linux-nilfs (at) vger.kernel.org>

• Project information
  – http://www.nilfs.org/

• Development tree
  – git://git.kernel.org/pub/scm/linux/kernel/git/ryusuke/nlfs2.git
Thank you for listening!