Utilizing NILFS2 Fine-grained Snapshots

Ryusuke KONISHI

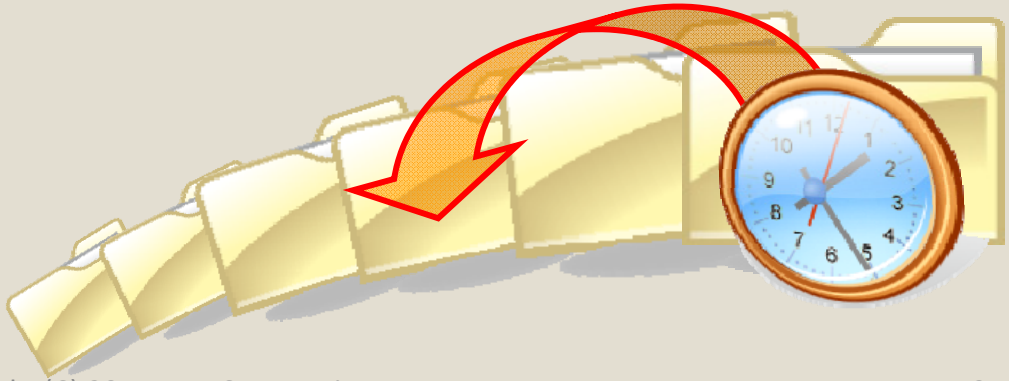
NTT Cyberspace Laboratories
NTT Corporation

Outline

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

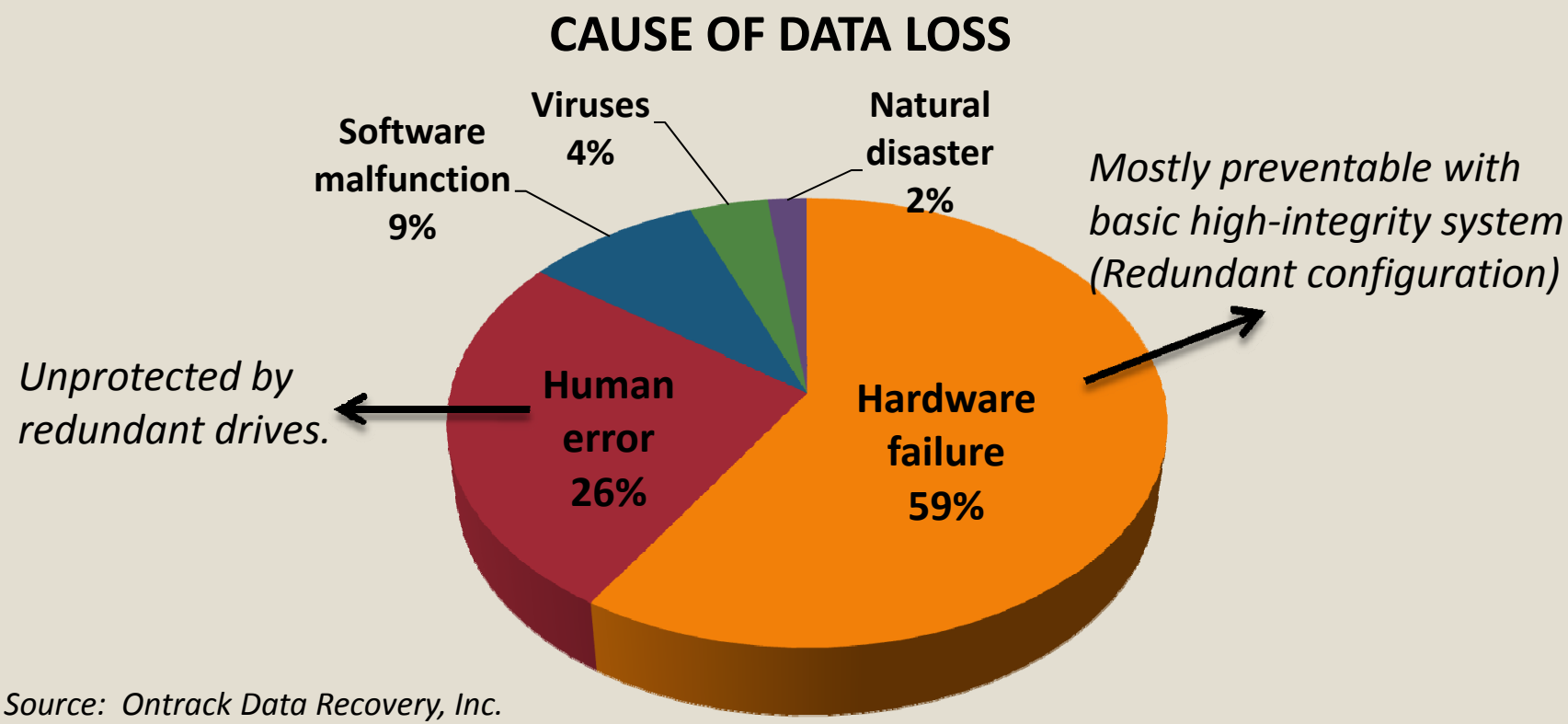
NILFS2 Overview

- **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?

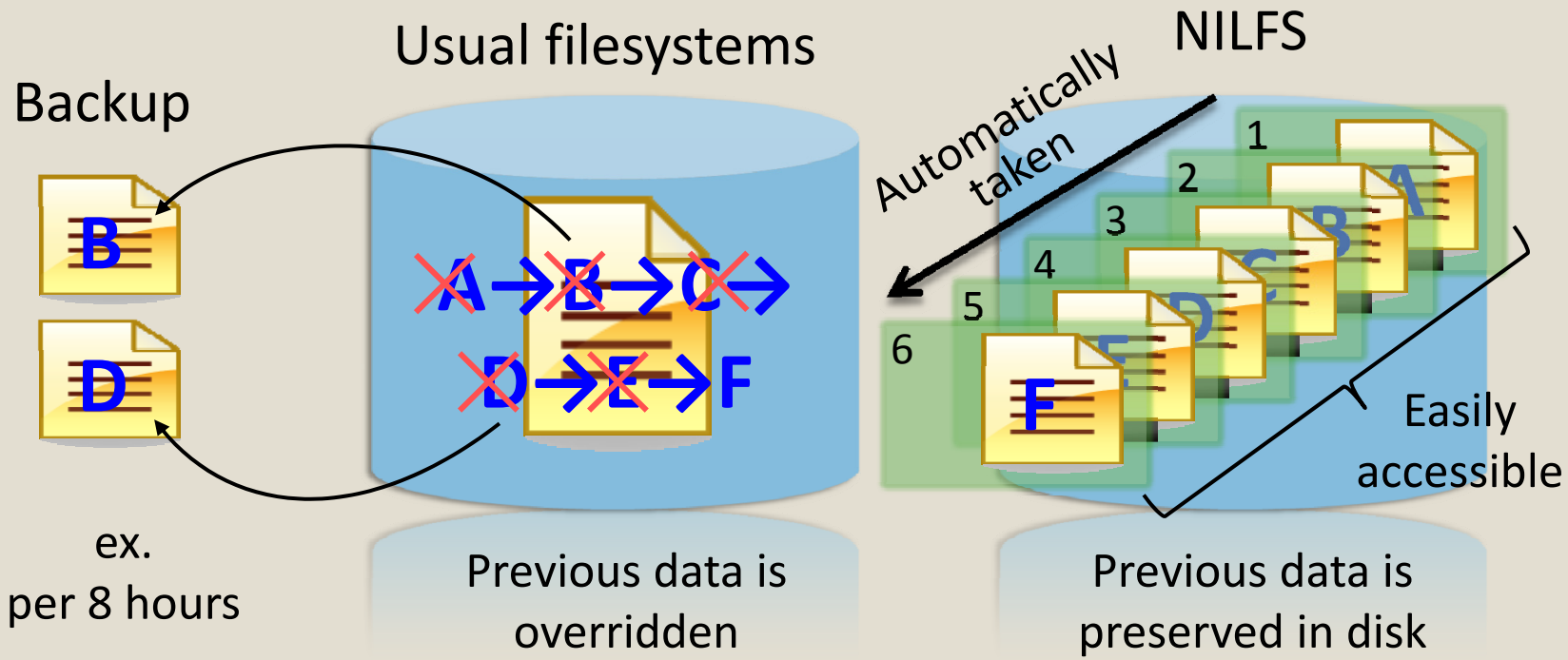
- **Backup is necessary to prevent data loss, but it still accompanies inconvenience and pain.**



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

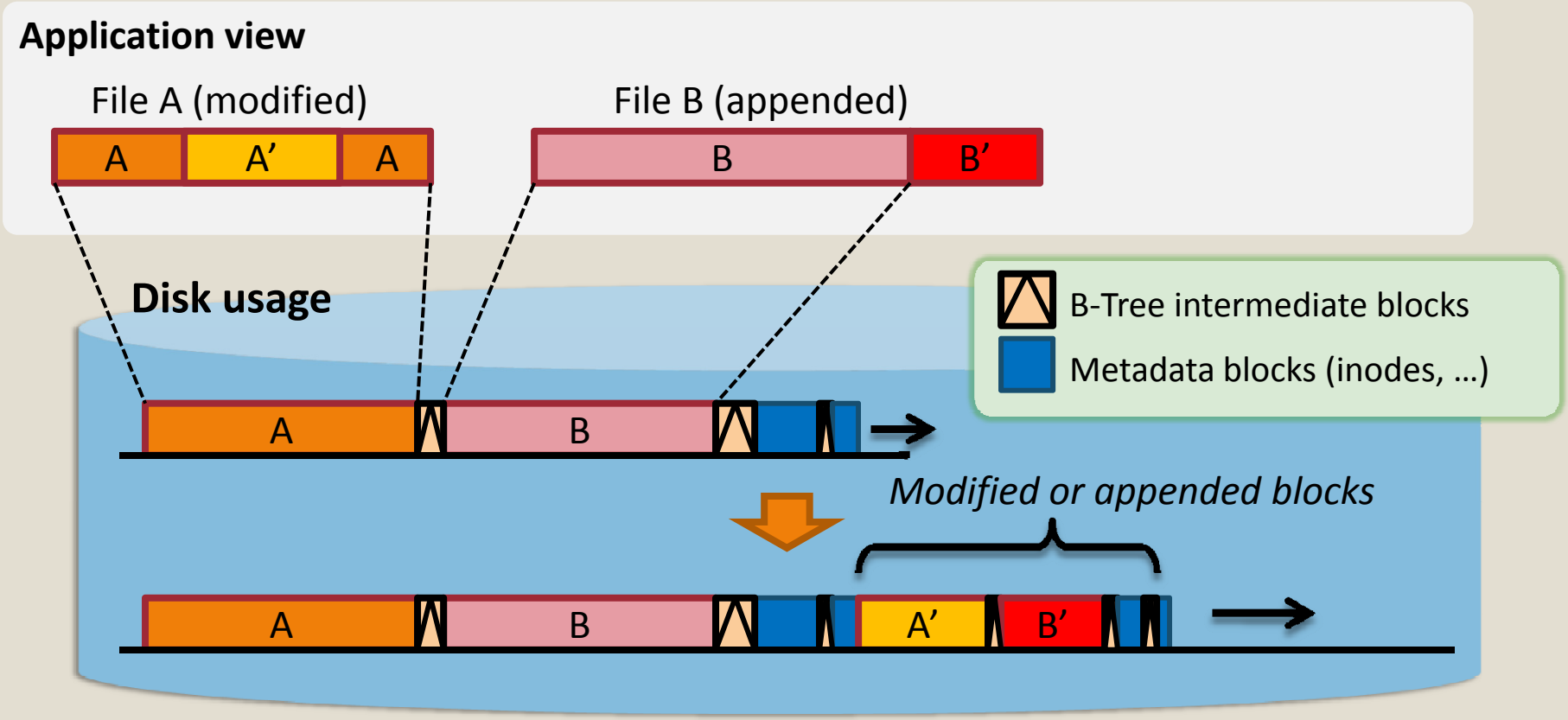
Solution with NILFS

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



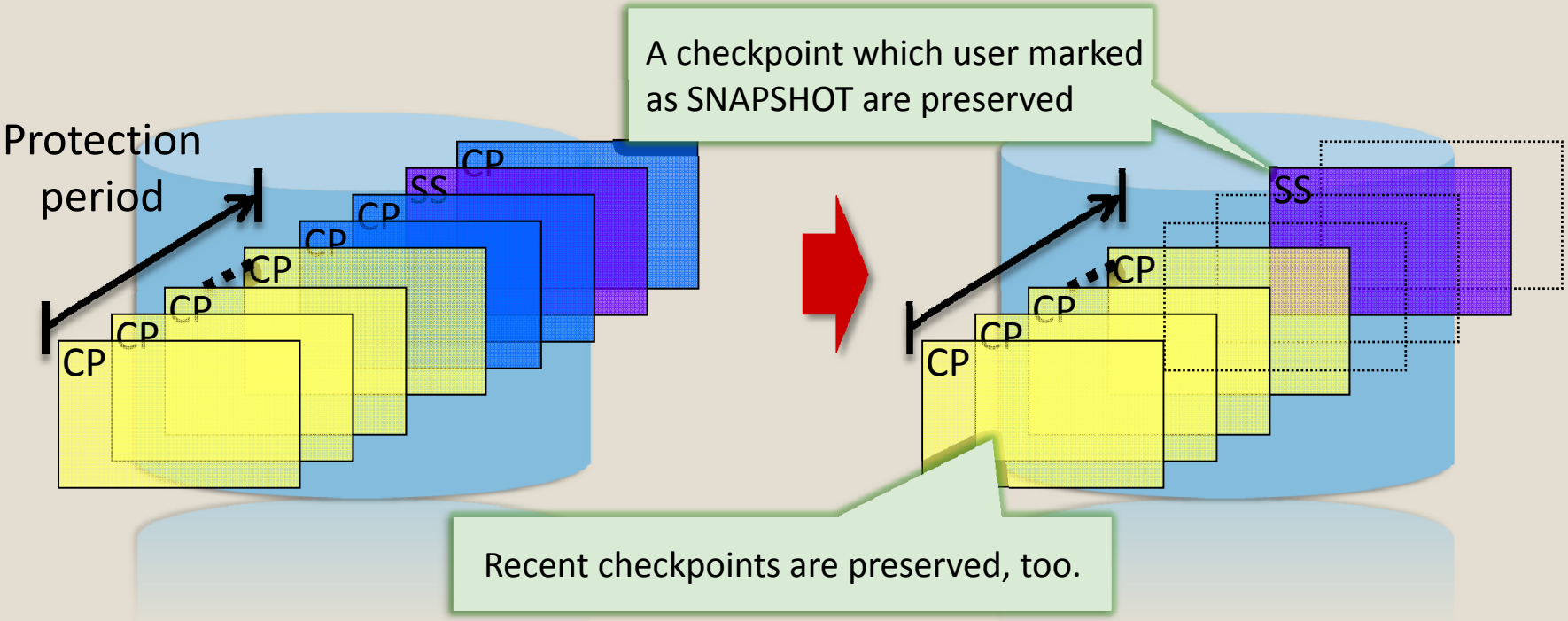
Disk write in NILFS

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



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.



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**

```
$ 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
         3  2011-05-08 20:40:34  cp     -       136         61
         4  2011-05-08 20:41:20  cp     -    187666       1604
         5  2011-05-08 20:41:42  cp     -        51       1634
         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
        ...
```

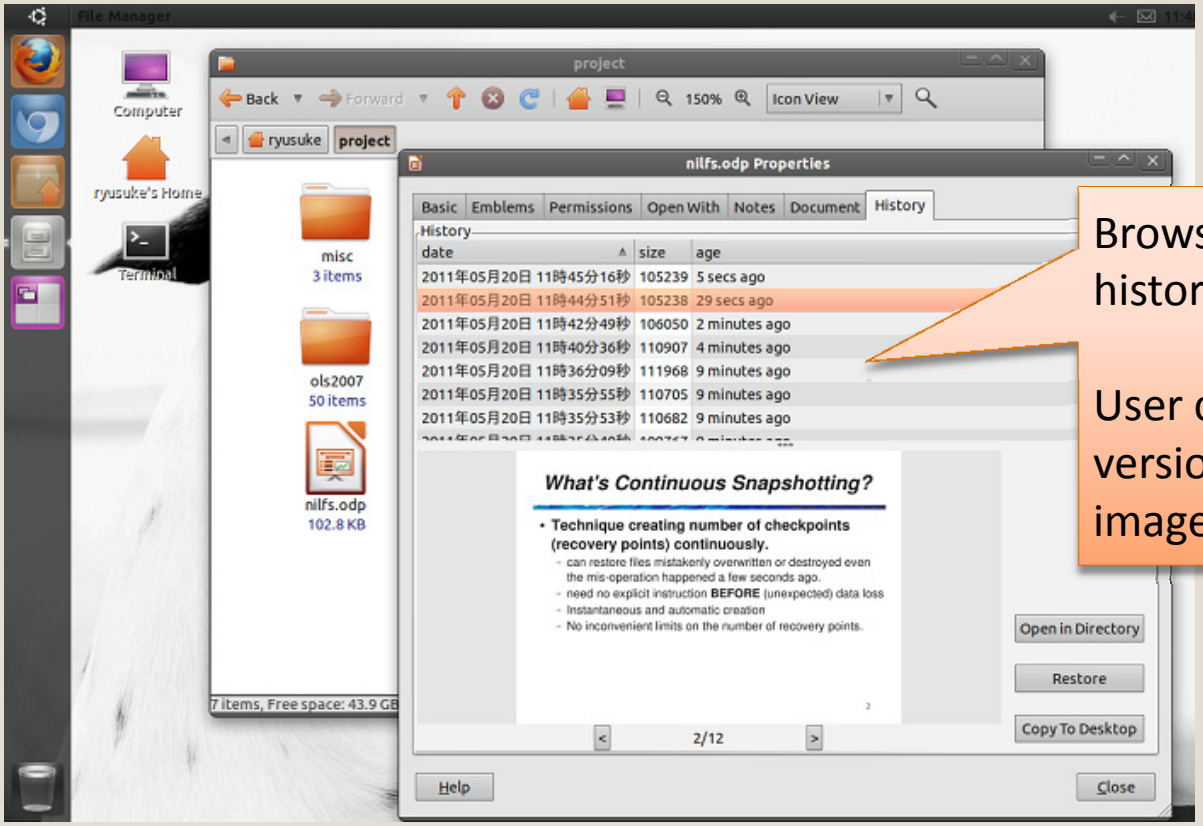
Use-Case Scenario

- **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

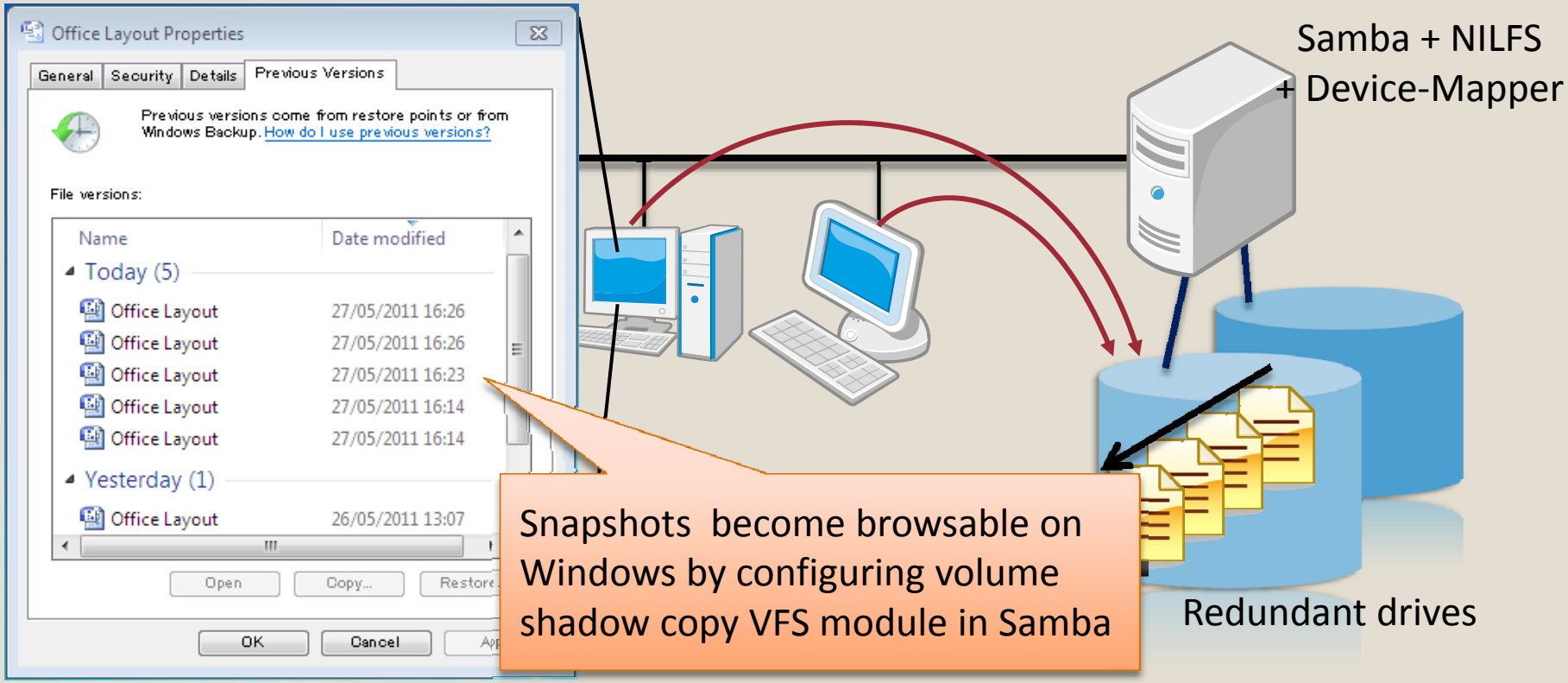


Browsable document change history.

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.



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

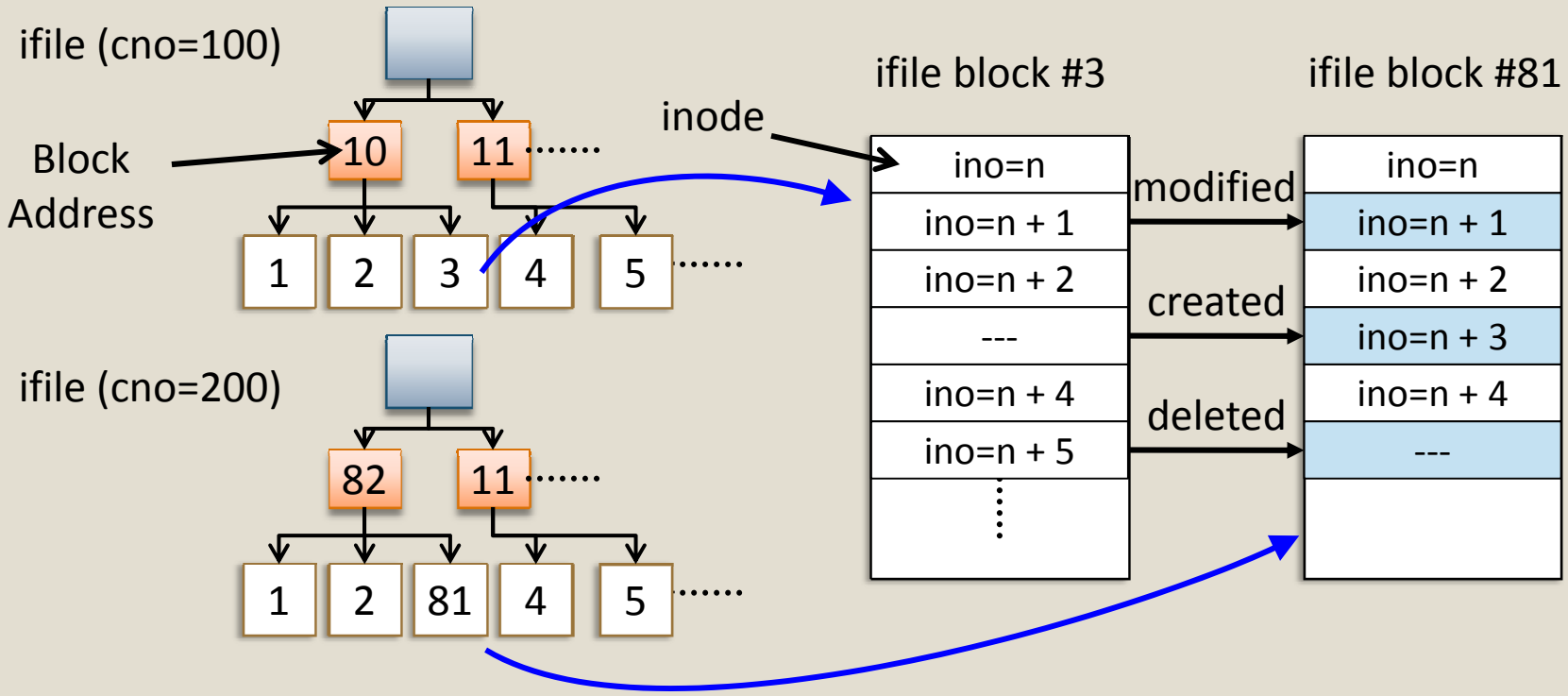
- Efficient delta extraction, restoration, de-dupe.
- Data security (e.g. shredding), anti-tampering.

WIP - Snapshot diff (1/4)

- **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

Comparison method	Time (seconds)
`diff:1 -Nqr snapshot-a/ snapshot-b/`	56.5
`diff:2 -Nqr snapshot-a/ snapshot-b/`	10.2
`nilfs-diff`	0.27

x 209 faster

x 38 faster

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**.

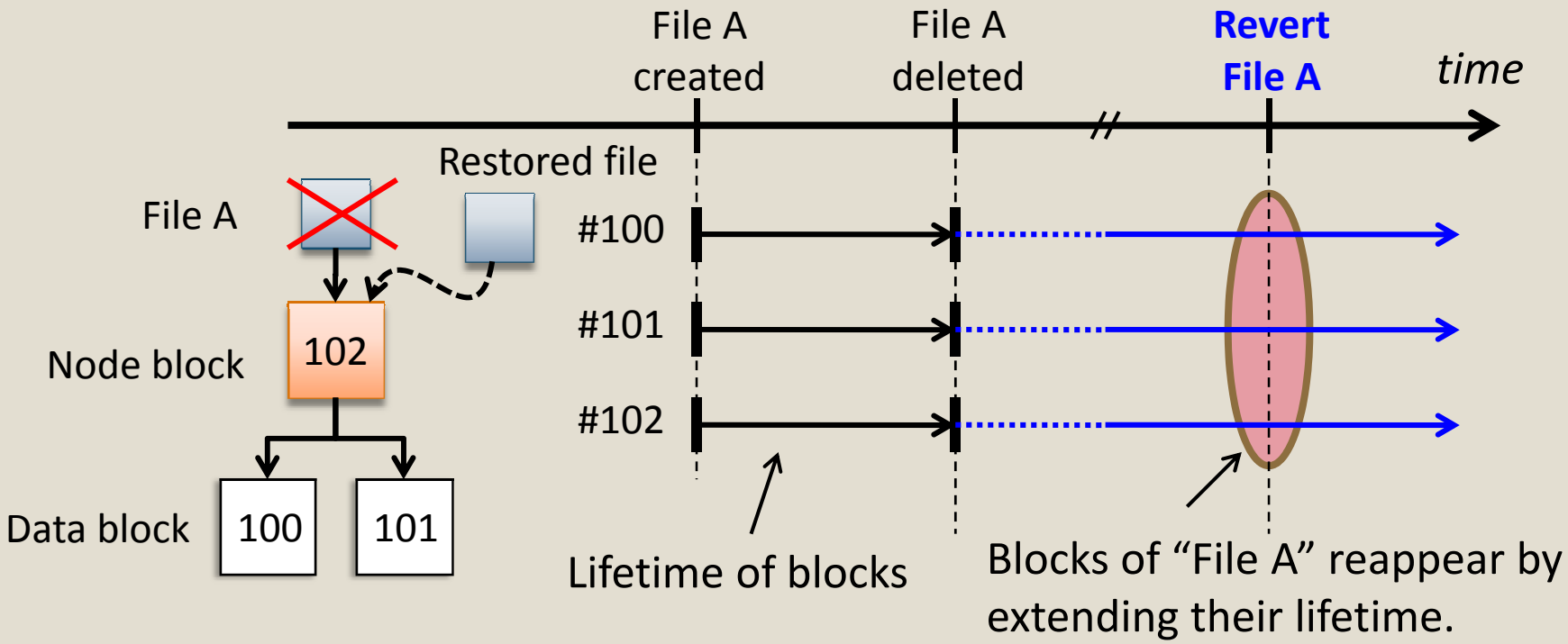
Hardware specs: Processor: Xeon 5160 @ 3.00 GHz x 2, Memory: 7988MB, Disk: IBM SAS SES-2

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.

WIP - Revert API (2/4)

- **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.



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

Restore method	Time (seconds)	Capacity growth (GiB)
`cp`	84.6	2.04
`nilfs-revert`	1.1	0.016

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/nilfs2.git

Thank you for listening !