
SUSE®
We adapt. You succeed.

Isn't one type of car enough?

Isn't one type of car enough?

Isn't one type of car enough?

- Old – Basic Functionality
- Small – Space efficient
- Slow – low risk
- Charged with emotions

Isn't one type of car enough?

- Old – Basic Functionality
- Small – Space efficient
- Slow – low risk
- Charged with emotions

- New – Fancy Functions
- Large – less space efficient
- Fast – high risk
- Charged with emotions

Isn't one type of filesystem enough?

- Functionality
- Efficiency
- Performance
- Reliability
- Charged with Emotions

Why btrfs is the Bread and Butter of Filesystems

… and why you may need other Filesystems as well …

Matthias G. Eckermann
Senior Product Manager
mge@suse.com

LinuxCon 2013 NA
2013-09-12 21:15 UTC

Agenda

- Local Linux Filesystems
- Copy on Write
- Filesystem recommendations
- CoW – and what to do with it
  - Using “snapper”
    to manage Operating System activities
  - Snapshotting on the Desktop
  - Server Side Snapshots
  - Other features – Future

Linux (Local) Filesystems

Major Linux (local) Filesystems

Feature	ext 2/3	reiserfs	xfs	ext4	btrfs
Data/Metadata Journaling	•/•	•/•	○/•	•/•	CoW
Journal internal/external	•/•	•/•	•/•	•/•	CoW
Offline extend/shrink	•/•	•/•	○/○	•/•	•/•
Online extend/shrink	•/○	•/○	•/○	•/○	•/•
Inode-Allocation-Map	table	u.B*-tree	B+-tree	table	B-tree
Sparse Files	•	•	•	•	•
Tail Packing	○	•	○	○	•
Defrag	○	○	•	•	•
ExtAttr / ACLs	•/•	•/•	•/•	•/•	•/•
Quotas	•	•	•	•	Subvol.
max. Filesystemsize	16 TiB	16 TiB	8 EiB	1 EiB	16 EiB
max. Filesize	2 TiB	1 EiB	8 EiB	1 EiB	16 EiB

Major Linux (local) Filesystems

Feature	ext 2/3	reiserfs	xfs	ext4	btrfs
Data/Metadata Journaling	•/•	•/•	○/•	•/•	CoW
Journal internal/external	•/•	•/•	•/•	•/•	CoW
Offline extend/shrink	•/•	•/•	○/○	•/•	•/•
Online extend/shrink	•/○	•/○	•/○	•/○	•/•
Inode-Allocation-Map	table	u.B*-tree	B+-tree	table	B-tree
Sparse Files	•	•	•	•	•
Tail Packing	○	•	○	○	•
Defrag	○	○	•	•	•
ExtAttr / ACLs	•/•	•/•	•/•	•/•	•/•
Quotas	•	•	•	•	Subvol.
max. Filesystemsize	16 TiB	16 TiB	8 EiB	1 EiB	16 EiB
max. Filesize	2 TiB	1 EiB	8 EiB	1 EiB	16 EiB

Copy on Write (1)

“Normal” Write

- Existing blocks of a file are overwritten, when the content changes

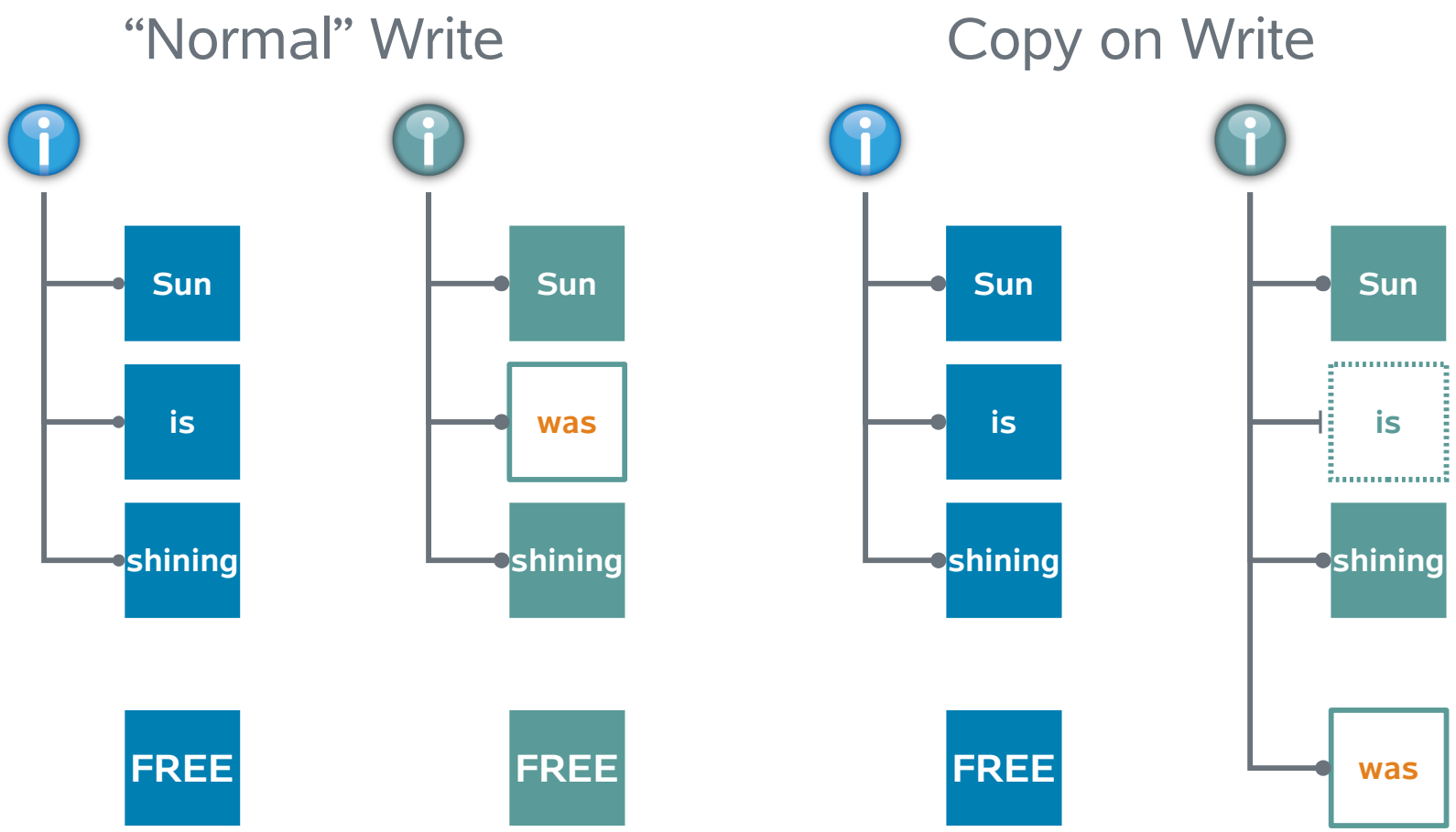
Copy on Write

- If content of a block changes, the changed block is put besides the non-changed block
- Instead the metadata (block-list) changes

Benefit

- Implementation of “transactions” in the filesystem is easy, as old content is still available

Copy on Write (2)



Copy on Write (3)

Disadvantages	Advantages
Performance impact on *specific* workloads, such as storing VMs	Efficient Storage Deduplication Snapshots Integrity beyond Journalling

Btrfs
Main features and concepts

Features

- Extents
  - Use only what's needed
  - Contiguous runs of disk blocks
- Copy-on-write
  - Never overwrite data!
  - Similar to CoW in VM
- Snapshots
  - Light weight
  - At filesystem level

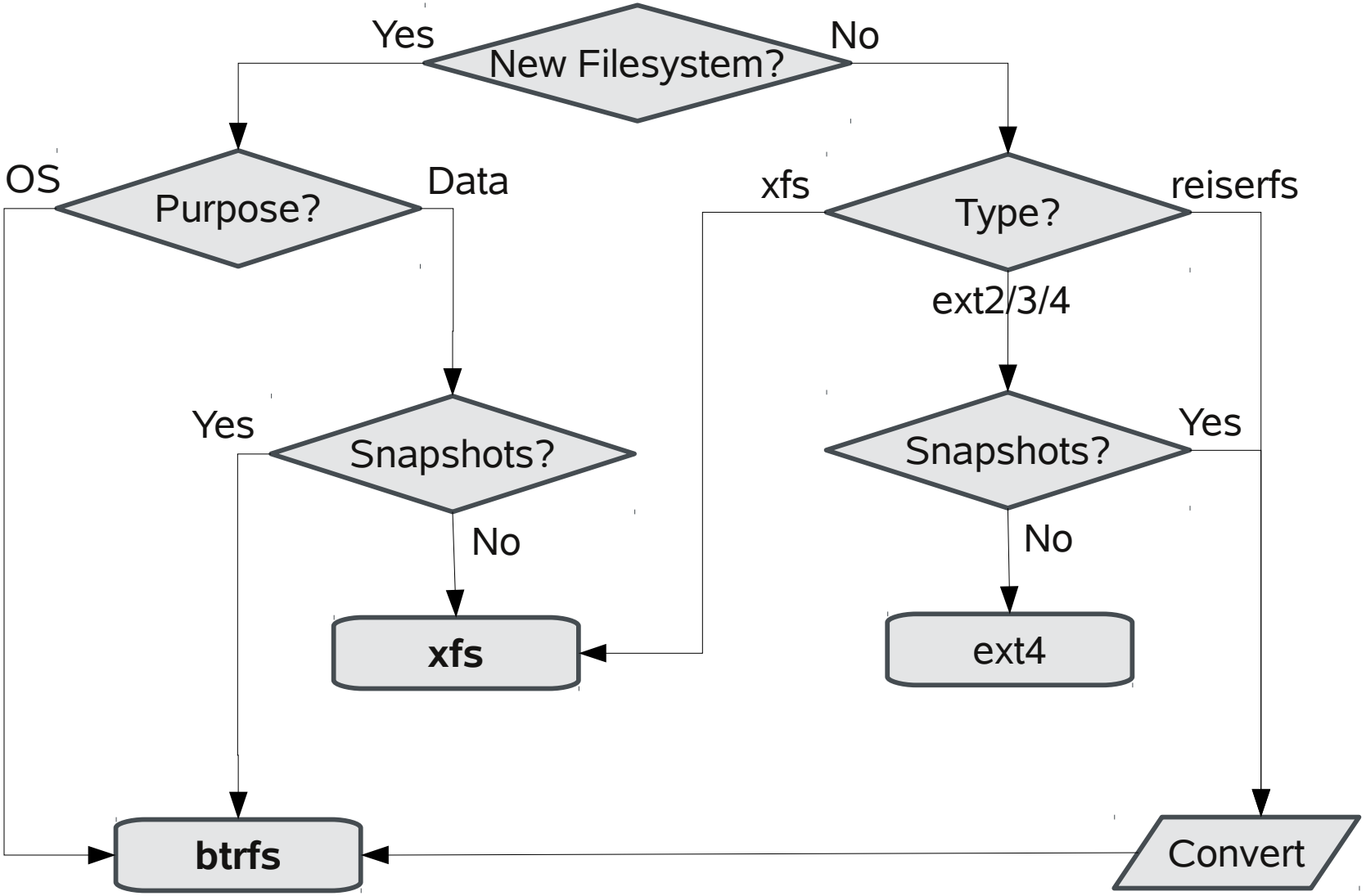
Concepts

- B-Tree
  - Index data structure
  - Fast search, insert, delete
- Subvolume
  - Filesystem inside the filesystem
- Metadata
  - “normal” metadata
  - B-Tree structures
- Raw data
  - Actual content of files

Major Linux (local) Filesystems

Feature	ext 2/3	reiserfs	xfs	ext4	btrfs
Data/Metadata Journaling	•/•	•/•	○/•	•/•	CoW
Journal internal/external	•/•	•/•	•/•	•/•	CoW
Offline extend/shrink	•/•	•/•	○/○	•/•	•/•
Online extend/shrink	•/○	•/○	•/○	•/○	•/•
Inode-Allocation-Map	table	u.B*-tree	B+-tree	table	B-tree
Sparse Files	•	•	•	•	•
Tail Packing	○	•	○	○	•
Defrag	○	○	•	•	•
ExtAttr / ACLs	•/•	•/•	•/•	•/•	•/•
Quotas	•	•	•	•	Subvol.
max. Filesystemsize	16 TiB	16 TiB	8 EiB	1 EiB	16 EiB
max. Filesize	2 TiB	1 EiB	8 EiB	1 EiB	16 EiB

Filesystem recommendations



Why xfs?

- Maturity
  - comes from IRIX
  - ported to Linux > 10 years ago
- Track record for
  - Performance
  - Scalability
  - Stability
- Active Development community
  - Checksums
  - Self-identifying metadata

CoW – and what to do with it

btrfs Maturity

Mature / “Enterprise ready”	Not (yet) mature
Copy on Write	Inode Cache
Snapshots	Auto Defrag
Subvolumes	RAID
Metadata Integrity	Compression
Data Integrity	Send / Receive
Online metadata scrubbing	Hot add / remove
Manual Defragmentation	Seeding devices
Manual Deduplication	Multiple Devices
Quota Groups	“Big” Metadata

Using “snapper” to manage Operating System activities

What is snapper?

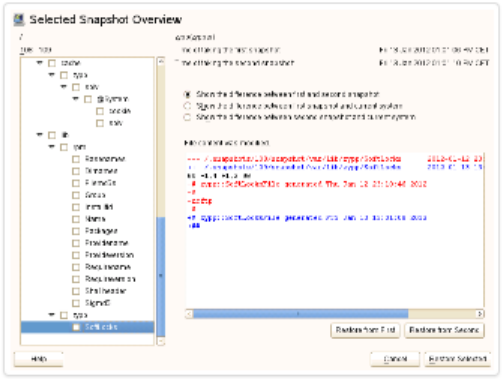
- Tool to manage btrfs snapshots
- Functions:
  - create, modify, delete
  - status (=compare), diff
  - undochange
  - cleanup
- Integration with
  - Package management stack (SUSE: zypper, yum)
  - Systems management stack (SUSE: YaST)
- DBUS service

Snapper Home Tutorial Documentation FAQ

Travel back in time and compare...

The ultimate snapshot tool for Linux

Download »



Watch it in action

Greg Kroah-Hartman and Matthias Eckermann play sysadmins and screw a web server configuration.

Arvin Schnell (lead developer) at FOSDEM 2012

Contribute

Snapper is opensource. Port it to your distribution or integrate it with an application.

Fork us on github »

Tweet

Thanks to Snapper, you can mess up system configuration changes or package installations or updates without having to restore from an old backup and risking to lose some files. Just revert to the snapshot before your problematic change and you're fine. *Linux User & Developer Magazine*

http://www.snapper.io/

snapper demonstration for administrative tasks

Snapshotting on the Desktop

/home/$USER

Using snapper for User data

Requirements

- /home/$USER is a btrfs subvolume
- “snapperd” with DBUS interface
- snapper configuration per user

Additional options

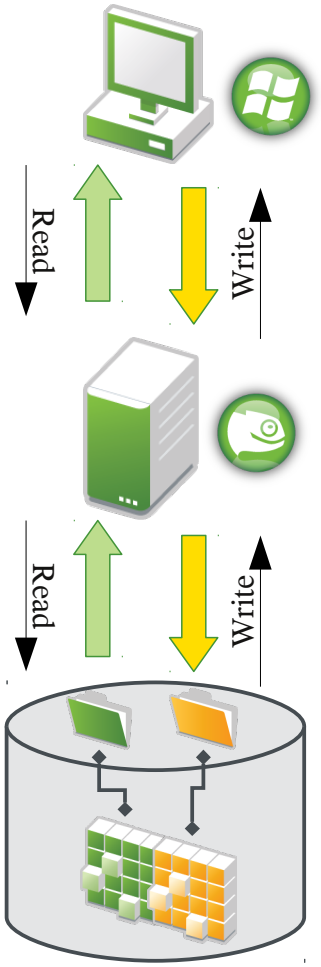
- Automated snapshotting on login/logout
  - Requires pam-snapper
- Automated snapshotting on Suspend

snapper demonstration for a user

Server Side Snapshots

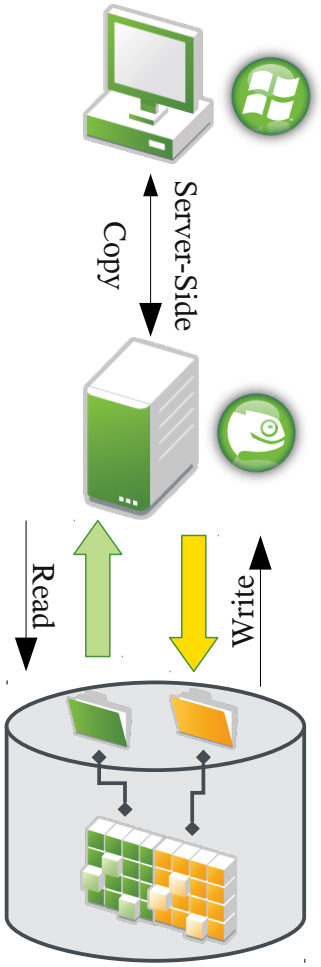
Btrfs as a Samba backend Server Side Copy

Traditional File Copy



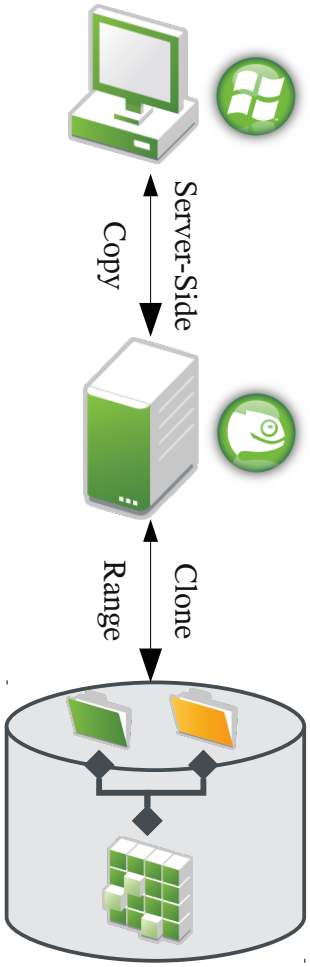
- File data takes disk and network round-trips
- Duplicate data stored on disk

Server-Side Copy



- Network round-trip avoided
- Server copies file data locally
- Duplicate data stored on disk

Btrfs Enhanced Server-Side Copy



- Data avoids network **and** disk round-trips
- No duplication of file data

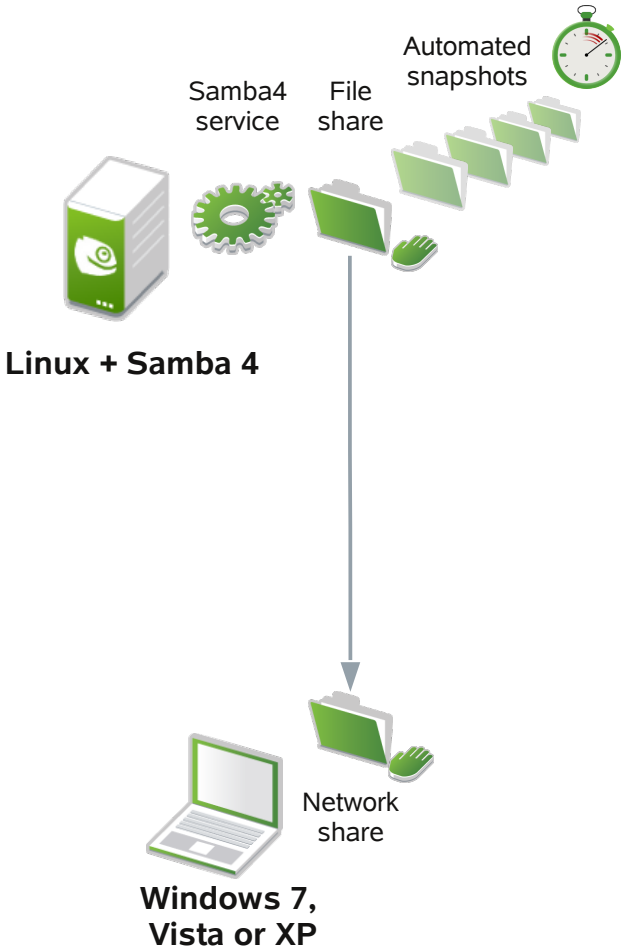
Server Side Snapshots

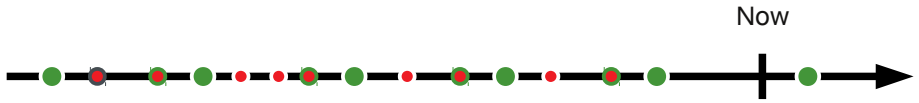
Btrfs as a Samba backend "Recovery Point"

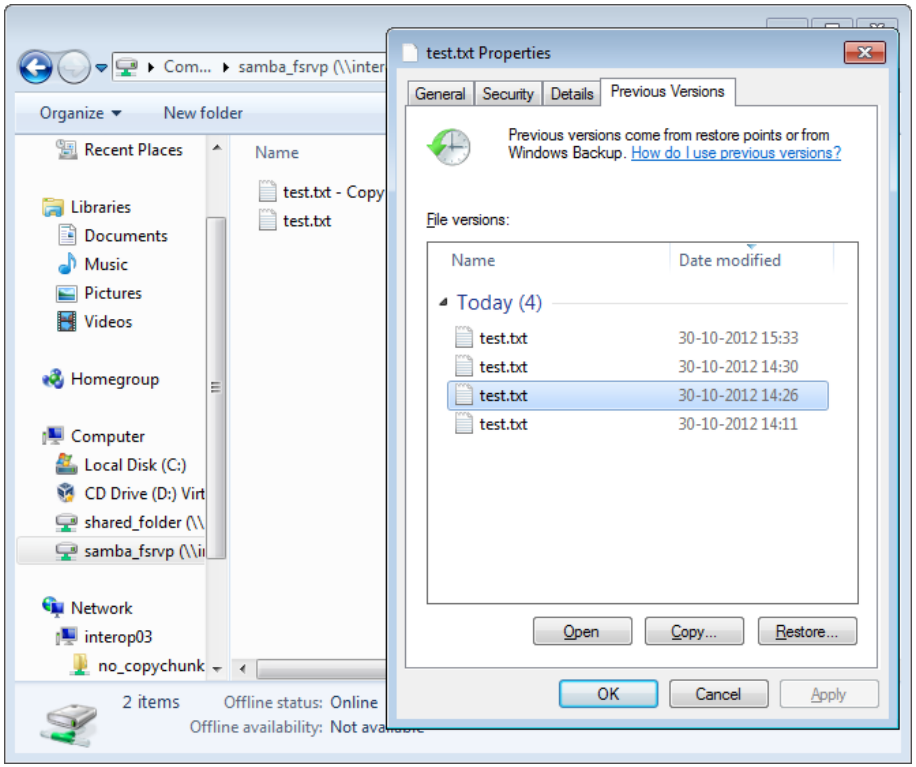
Samba4 and btrfs, snapper

Prototype Samba implementation of “Recovery Point”

- Automatic snapshots by Snapper
- Previous versions of “test.txt” in Explorer
- File “test.txt” is changed
- File “test.txt” is created







Other features – Future

Conversion to btrfs

- btrfs-convert
- offline in-place migration from
  - ext2/3/4

    and
  - reiserfs
- Keeps metadata of the old filesystem for a roll-back

demonstration: convert reiserfs to btrfs

Continuously Running Systems

Snapshot / Rollback for full system – Based on

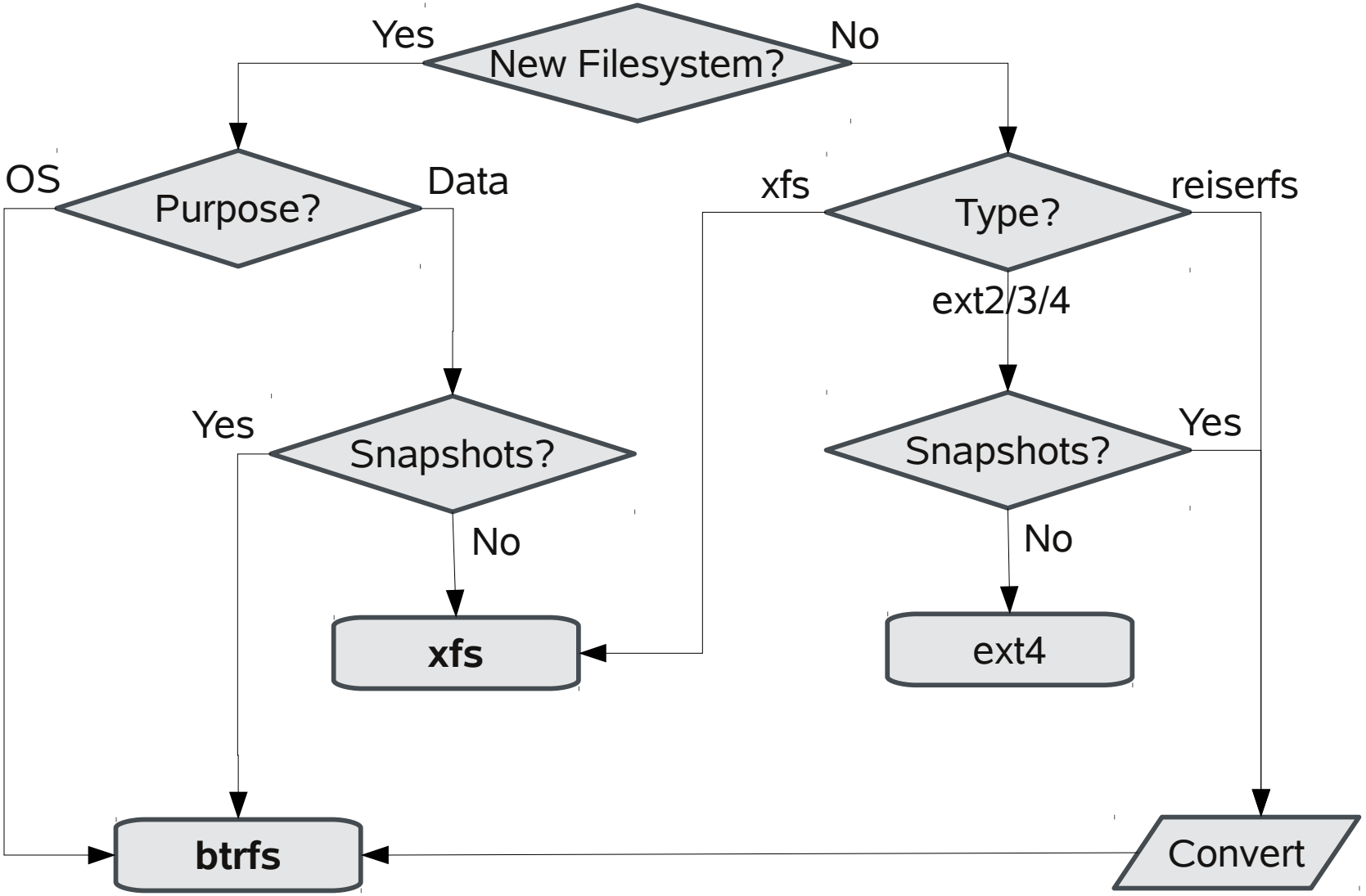
- btrfs
- Snapper
- Bootloader integration
  - Booting directly from a btrfs snapshot
  - Jump back to a former status of the OS, including kernel / initrd

Btrfs – Planned features

- Data de-duplication:
  - De-duplication during writes
  - Manual De-duplication
- Tiered storage
  - e.g.: combine SSD and HDD

Summary

Filesystem recommendations



Go ahead, try btrfs and snapper today!

Your questions!?

Thank you.

Corporate Headquarters
Maxfeldstrasse 5
90409 Nuremberg
Germany

+49 911 740 53 0 (Worldwide)
www.suse.com

Join us on:
www.opensuse.org