



次世代文件系统**Btrfs**介绍

繆勰 <miaox@cn.fujitsu.com>

南京富士通南大软件技术有限公司



Agenda

- **What is Btrfs?**
- Why is Btrfs needed?
- Which features does Btrfs include?
- How is Btrfs implemented?
- What did we do for Btrfs?
- How do I use Btrfs?



What is Btrfs?

- Btrfs is a new copy on write file system for Linux aimed at implementing advanced features while focusing on fault tolerance, repair and easy administration. (From Btrfs Wiki)
- Initially developed by Oracle, licensed under the GPL.



Agenda

- What is Btrfs?
- **Why is Btrfs needed?**
- Which features does Btrfs include?
- How is Btrfs implemented?
- What did we do for Btrfs?
- How do I use Btrfs?



Why is Btrfs needed?

- Disadvantages of the traditional file system
 - The limit of the file size and the file system size
 - Very bad augmentability
 - Very bad data integrity
 - No SSD support
 - ...
- The stress of ZFS – the next generation file system on Solaris



Agenda

- What is Btrfs?
- Why is Btrfs needed?
- Which features does Btrfs include?
- How is Btrfs implemented?
- What did we do for Btrfs?
- How do I use Btrfs?



Which features does Btrfs include?

- Extent based file storage
- 64Bits based space management
- Dynamic inode allocation
- COW(Copy On Write) based transaction
- Checksums on data and metadata
- Integrated multiple device support
- Delayed space allocation
- Online Defragment



Which features does Btrfs include?

- Inline file
- Tree Log
- Compression
- SSD support
- Seed Device support
- Snapshot
- Subvolume



Agenda

- What is Btrfs?
- Why is Btrfs needed?
- Which features does Btrfs include?
- **How is Btrfs implemented?**
- What did we do for Btrfs?
- How do I use Btrfs?

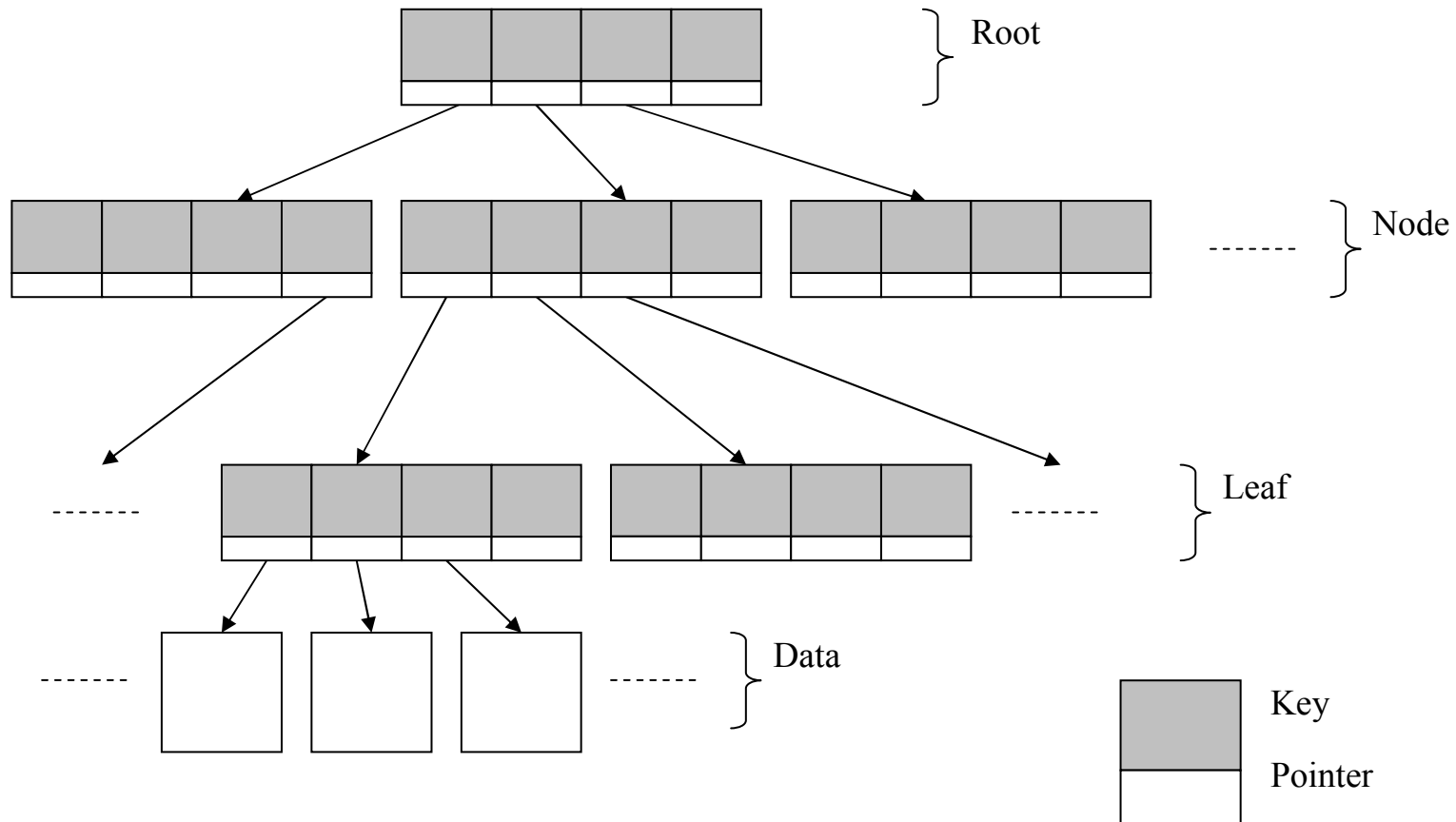


How is Btrfs implemented?

- Basis of Btrfs – B+ tree
- On-disk data structure
- Implementation of the features

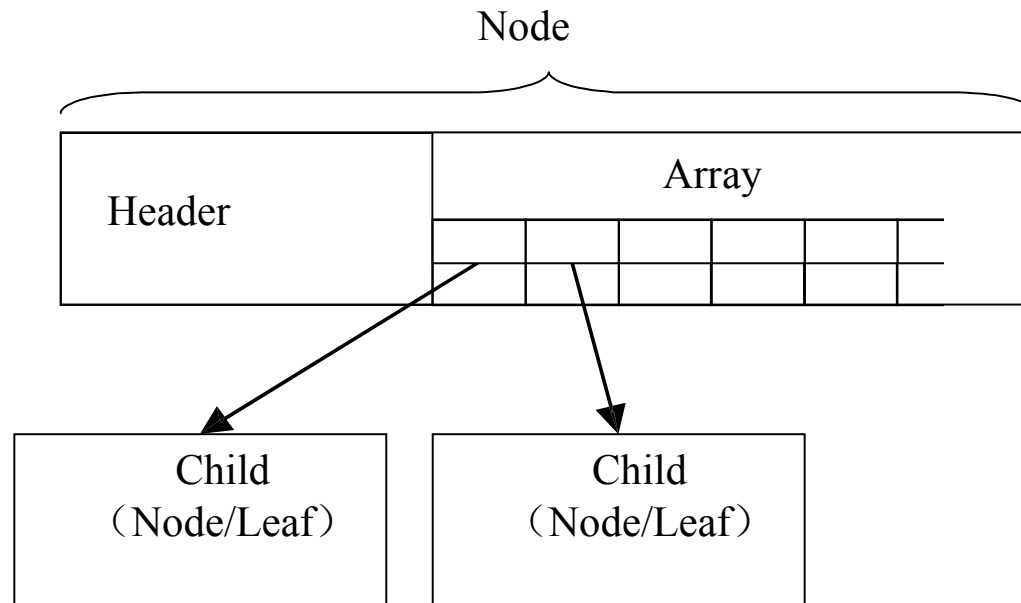
Basis of Btrfs – B+ tree

- Components of B+ tree



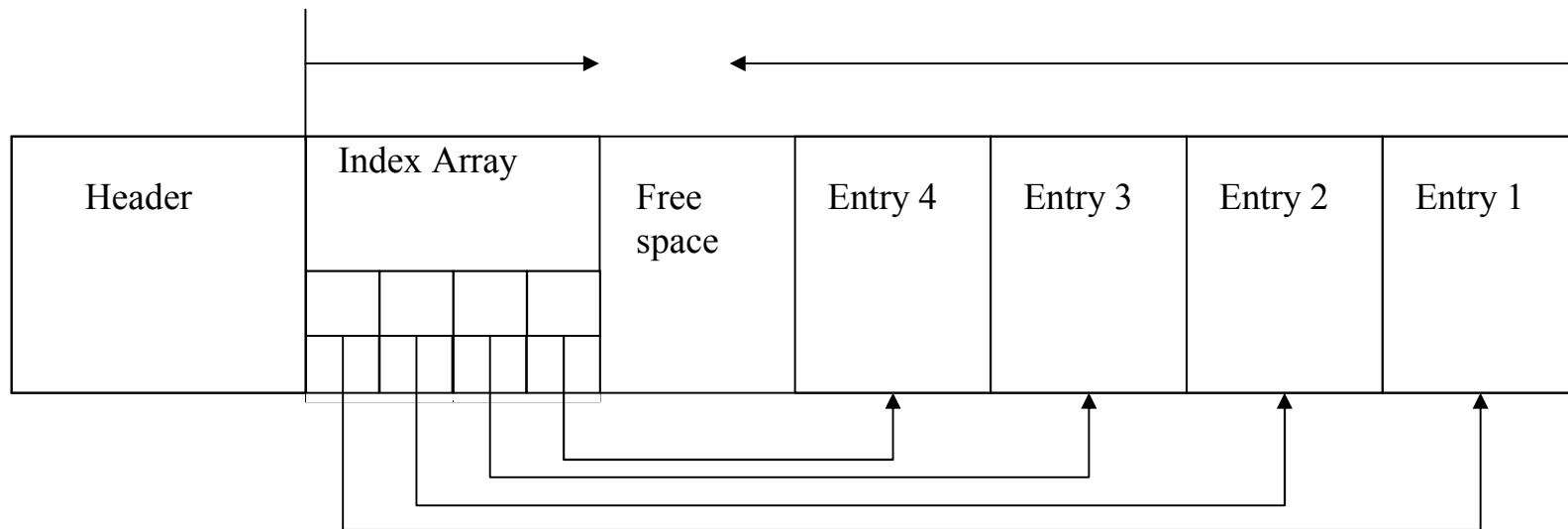
Basis of Btrfs – B+ tree

- Components of the internal node



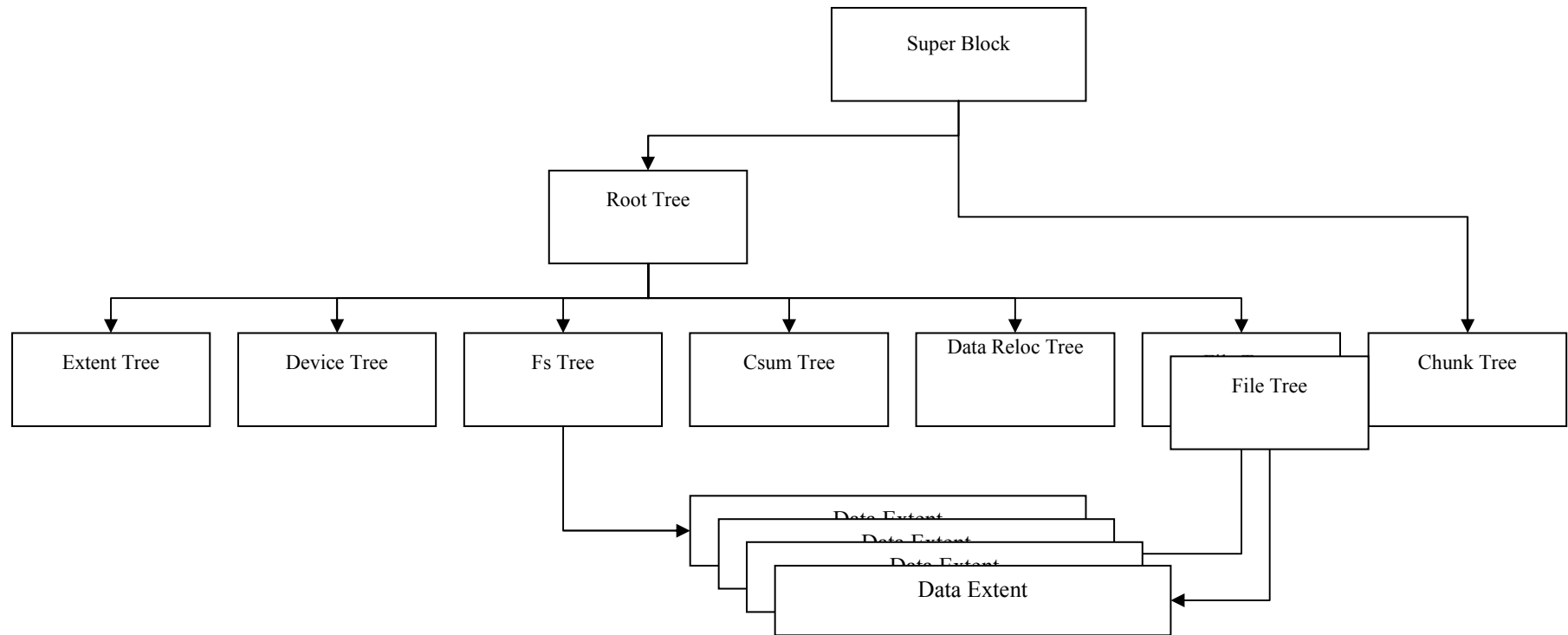
Basis of Btrfs – B+ tree

- Components of the leaf



On-disk data structure

- Main construction





On-disk data structure

- Root Tree
 - `btrfs_root_item`
 - `btrfs_root_ref`
 - ...
- Chunk Tree
 - `btrfs_dev_item`
 - `btrfs_chunk`
- Device Tree
 - `btrfs_dev_extent`



On-disk data structure

- Extent Tree
 - `btrfs_block_group_item`
 - `btrfs_extent_item`
 - `btrfs_tree_block_ref`
 - `btrfs_extent_data_ref`
 - `btrfs_shared_block_ref`
 - `btrfs_shared_data_ref`



On-disk data structure

- Fs/File Tree
 - btrfs_inode_item
 - btrfs_inode_ref
 - btrfs_dir_index
 - btrfs_dir_item
 - btrfs_extent_data



On-disk data structure

- Csum Tree
 - `btrfs_extent_csum`

- Data Relocation Tree
 - `btrfs_inode_item`
 - `btrfs_extent_data`



Implementation of the features

- Extent based file storage
 - [start, len]
 - Extent + Bitmap
- 64Bits based space management
 - 64Bits vs 32Bits(Ext3)/48Bits(Ext4)
- Dynamic inode allocation
 - Insert a inode item into Fs/File tree

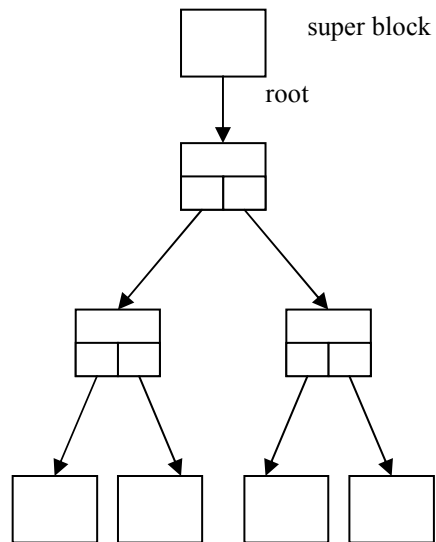


Implementation of the features

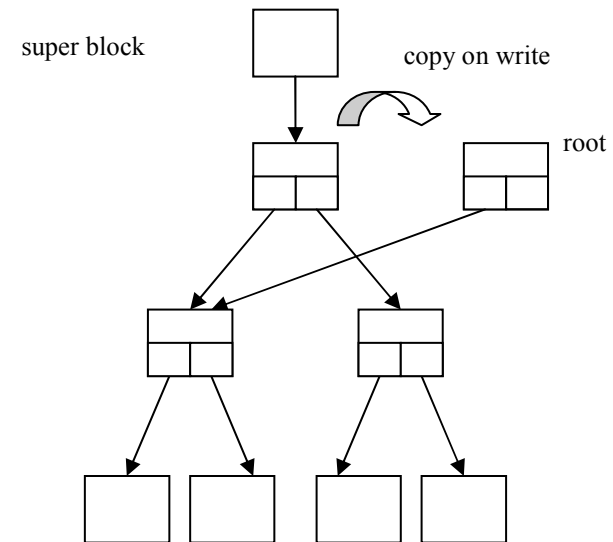
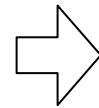
- COW based transaction
 - COW: Copy on Write. Copy the data before writing. Used for forking child task in the kernel
 - guarantee the data integrity on crash, be similar with JBD layer of Ext3/4

Implementation of the features

- COW based transaction



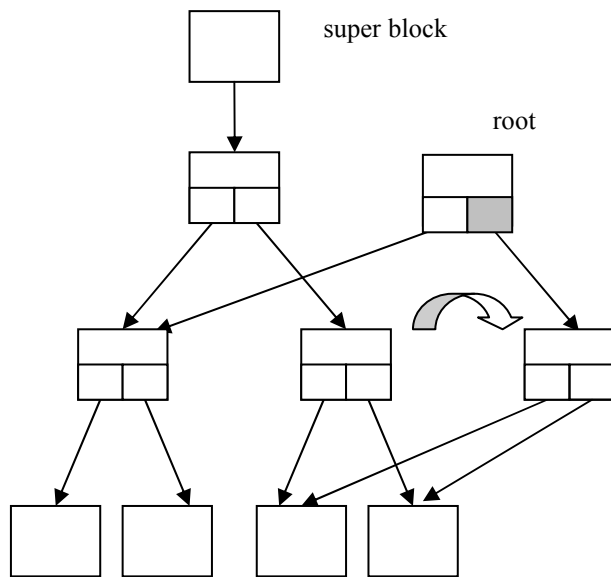
step 0) initial status(start transaction)



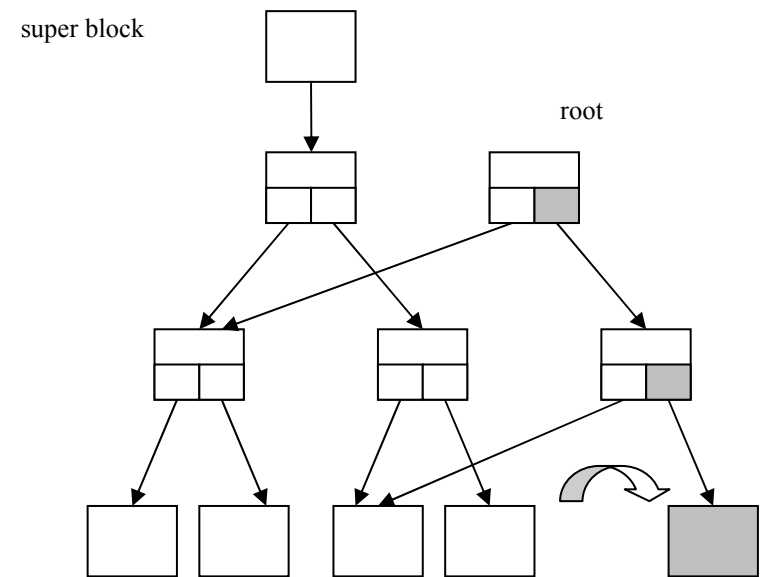
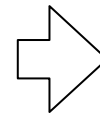
step 1) Update the root

Implementation of the features

- COW based transaction



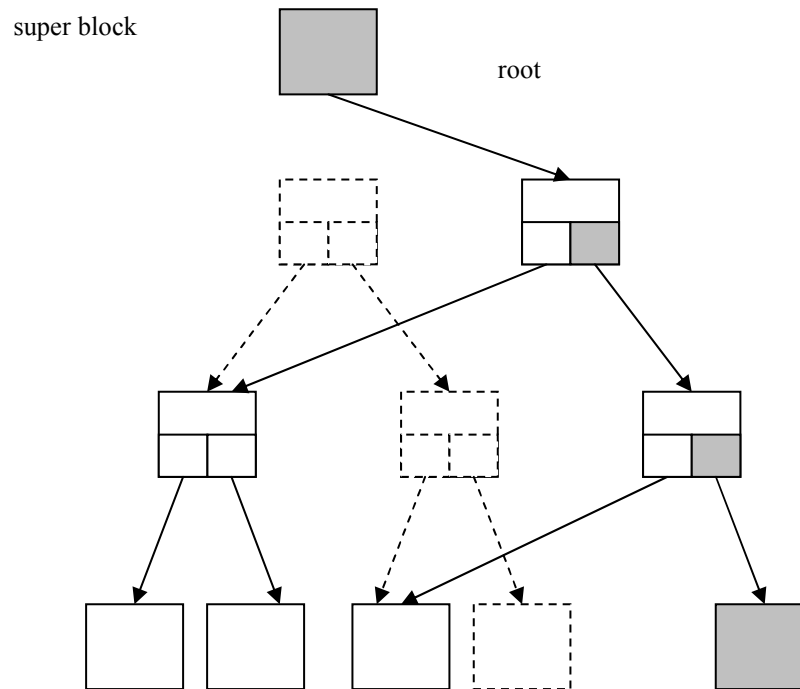
step 2) Update the node



step 3) Update the leaf

Implementation of the features

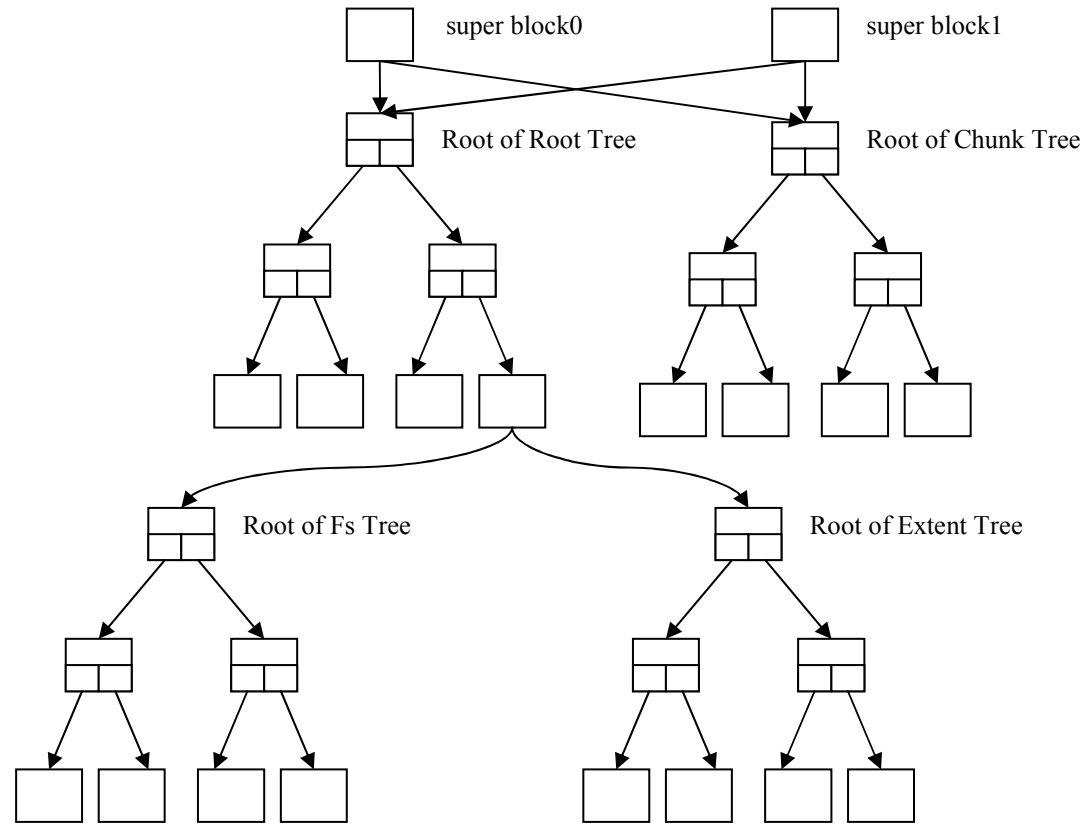
- COW based transaction



step 4) Update the super block(submit the transaction)

Implementation of the features

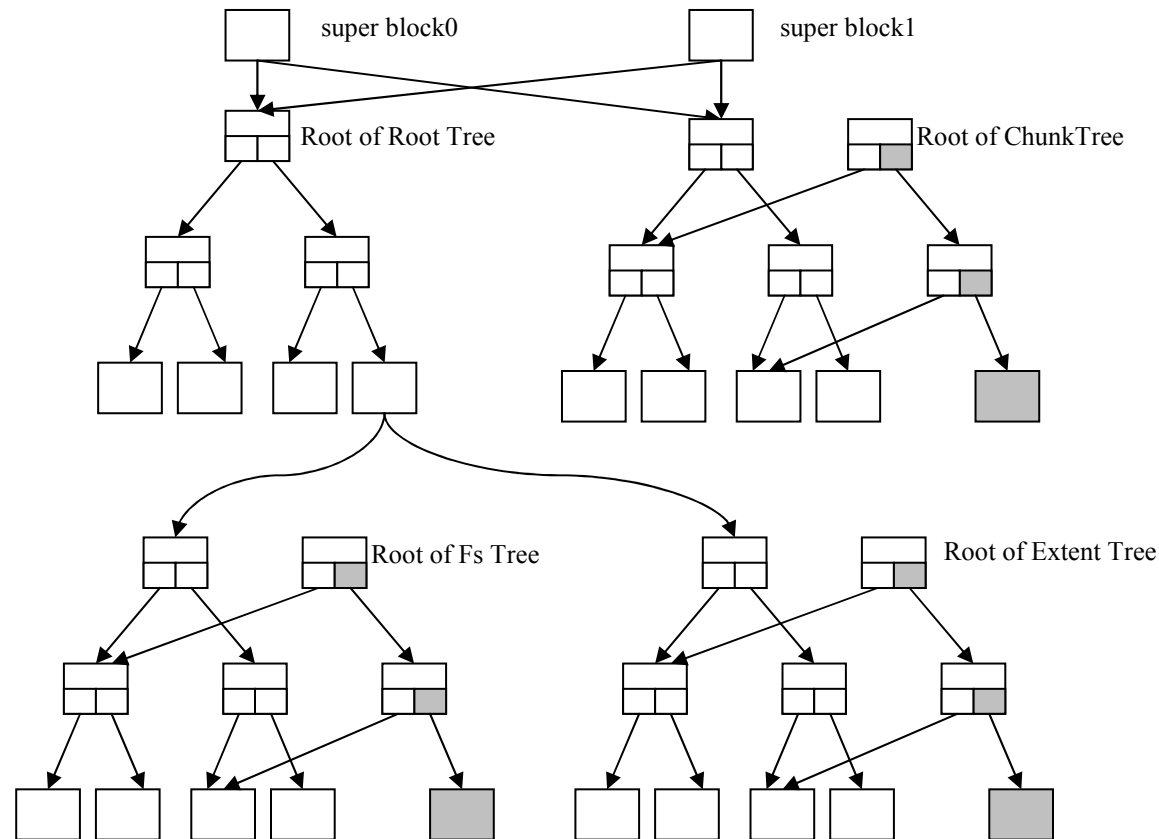
- COW based transaction



step 0) initial status(start transaction)

Implementation of the features

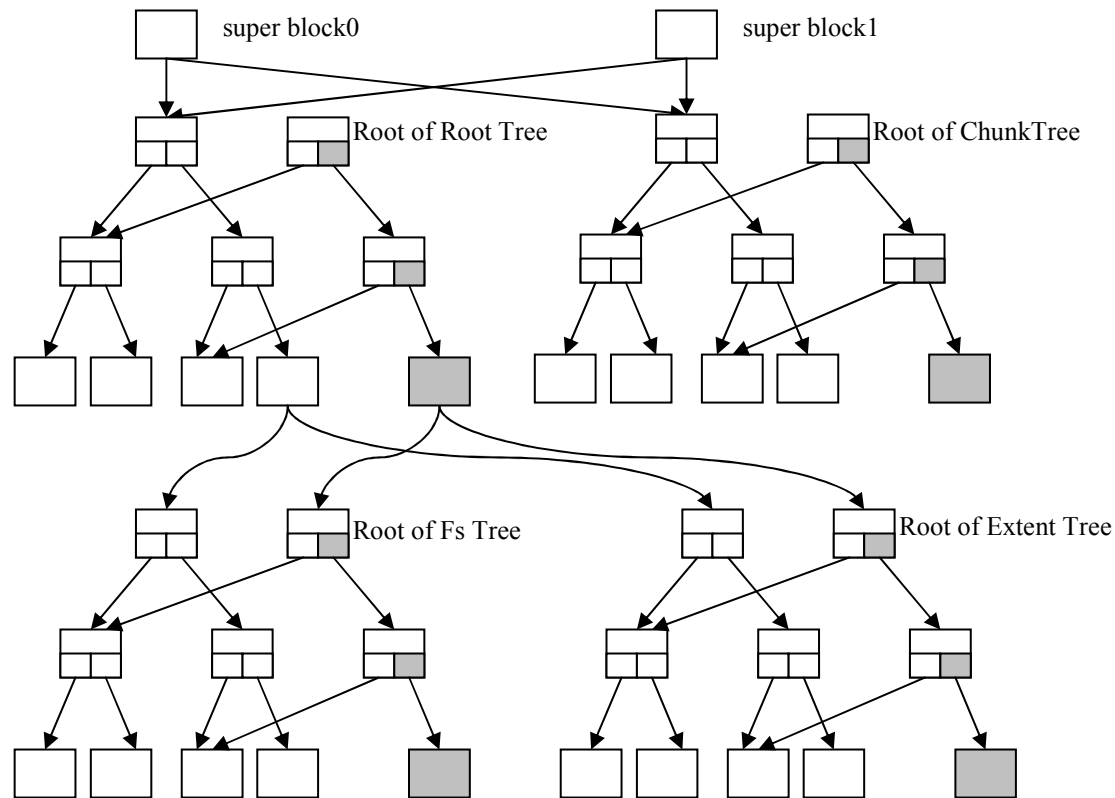
- COW based transaction



step 1) Update metadata in fs tree, extent tree and chunk tree

Implementation of the features

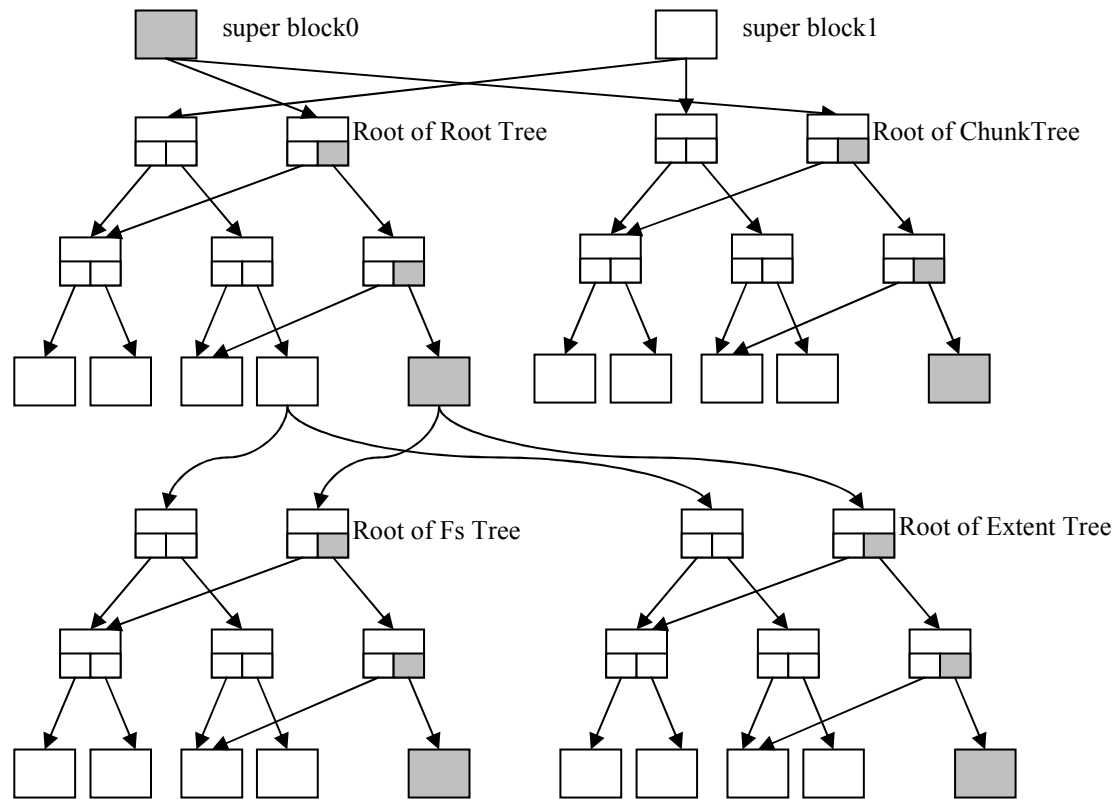
- COW based transaction



step 2.1) Update root tree(The 1st step of transaction commit)

Implementation of the features

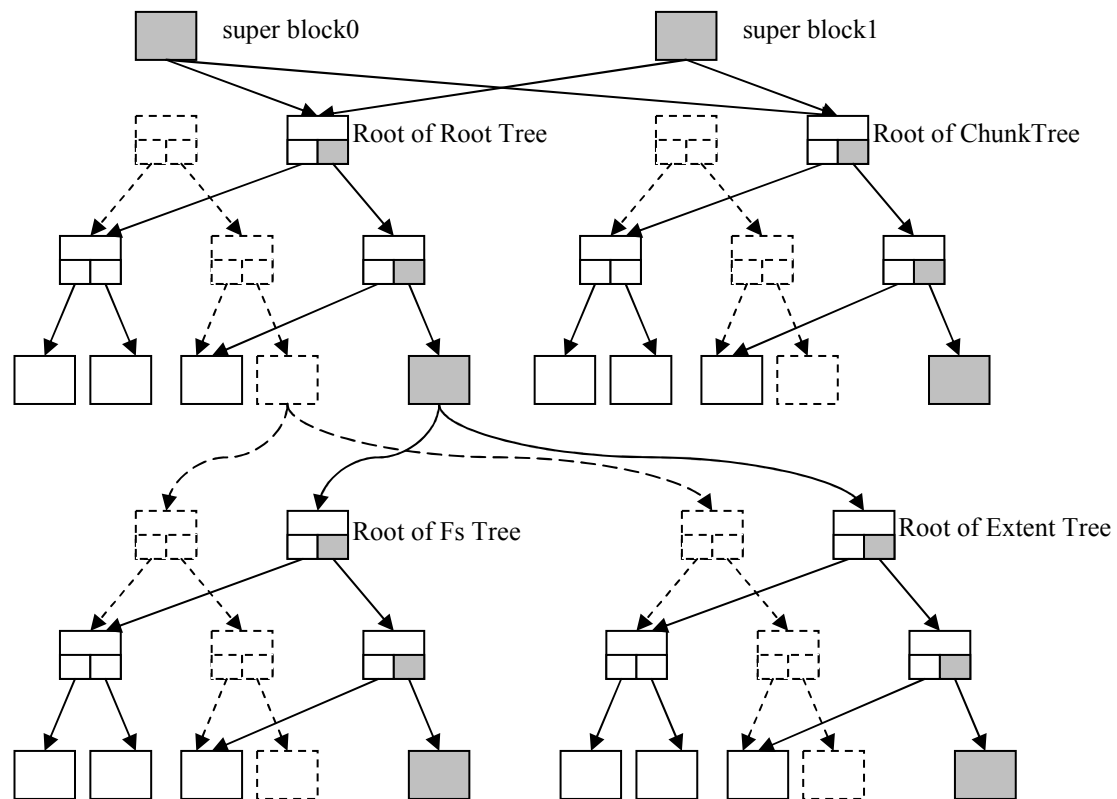
■ COW based transaction



step 2.2) Update the 1st super block(The 2nd step of transaction commit)

Implementation of the features

- COW based transaction



step 2.3) Update the 2st super block(The 3st step of transaction commit)



Implementation of the features

- Checksums on data and metadata
 - Data
 - Read: read checksum data from csum tree before submitting I/O request, do checksum after I/O request ends
 - Write: calculate checksum data before submitting I/O request, insert the checksum data into csum tree after I/O request ends



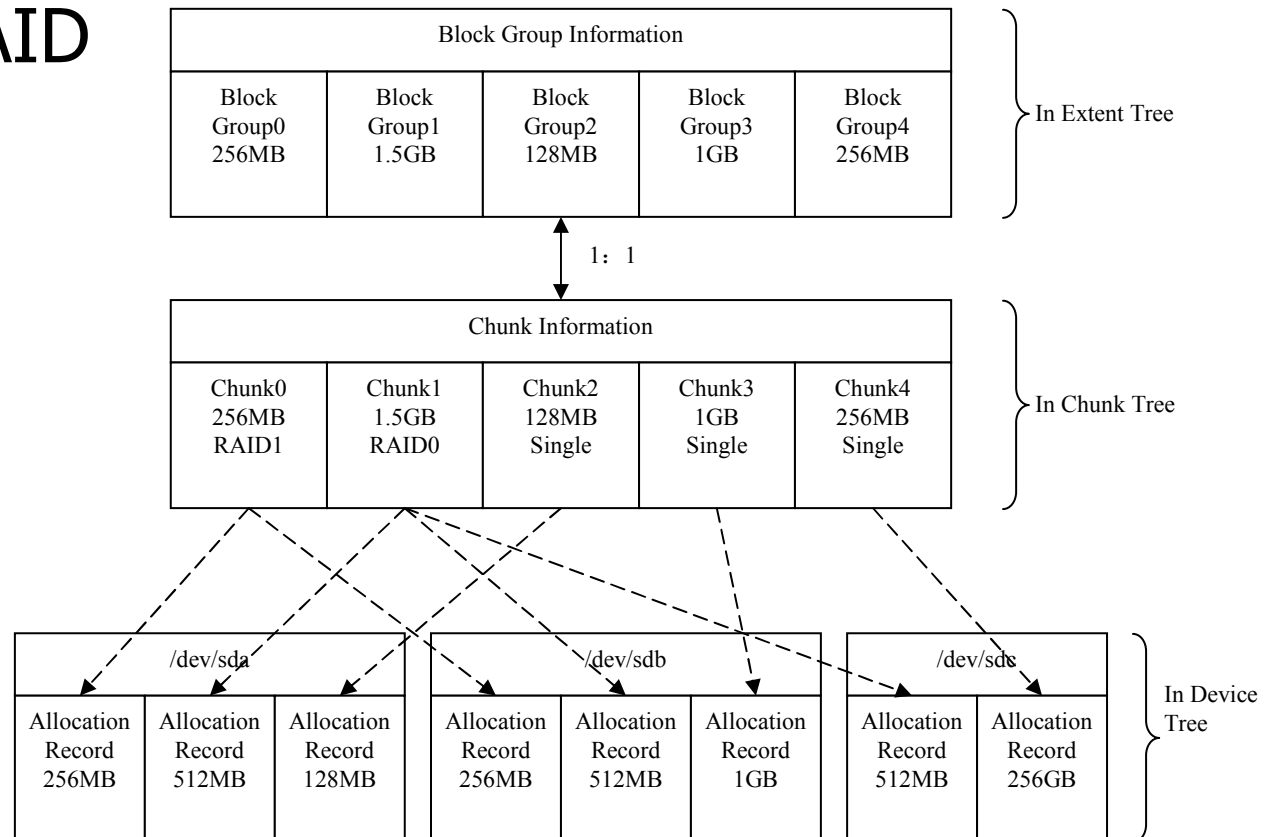
Implementation of the features

- Checksums on data and metadata
 - Metadata
 - Read: read metadata of a leaf or node, and get the checksum data from the header of the leaf/node, do checksum
 - Write: calculate checksum data of the leaf/node, put the checksum into the header of the leaf/node, write the leaf/node into the disk

Implementation of the features

- Integrated multiple device support

- Soft RAID





Implementation of the features

- Integrated multiple device support
 - Add/remove devices
 - Two lists: one is used to manage the devices that in the file system including read-only devices; the other is used to manage the allocable devices
 - Open/Close the device
 - Initialize the device/relocate the data



Implementation of the features

- Delayed space allocation
 - Space reservation
 - Allocate the space when doing write-back, not before writing data into the cache.
 - Advantage: reduce the fragment.



Implementation of the features

- Online Defragment
 - Use the delayed space allocation
 - Read the fragment into the cache, and mark the dirty flag
 - Write back the dirty data in the cache

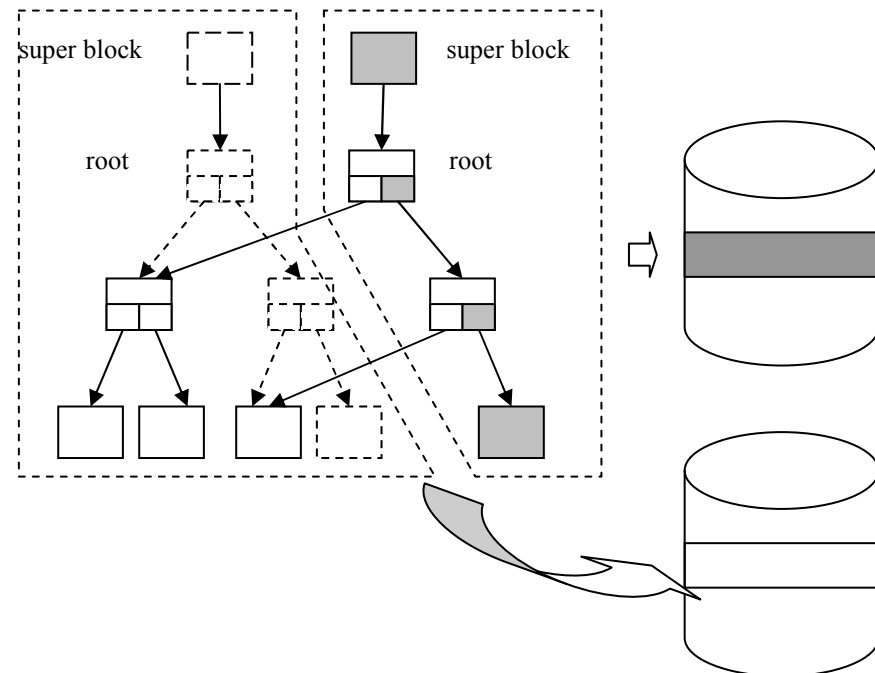


Implementation of the features

- Inline file
 - Store the file data into Fs/File tree after the relative `btrfs_extent_data`
 - Read the file data when reading the metadata
- Compression
 - zlib/LZO
 - Read: uncompress the data after ending the I/O request and then copy the uncompressed the data into the cache.
 - Write: compress the data before the space allocation when doing write-back.

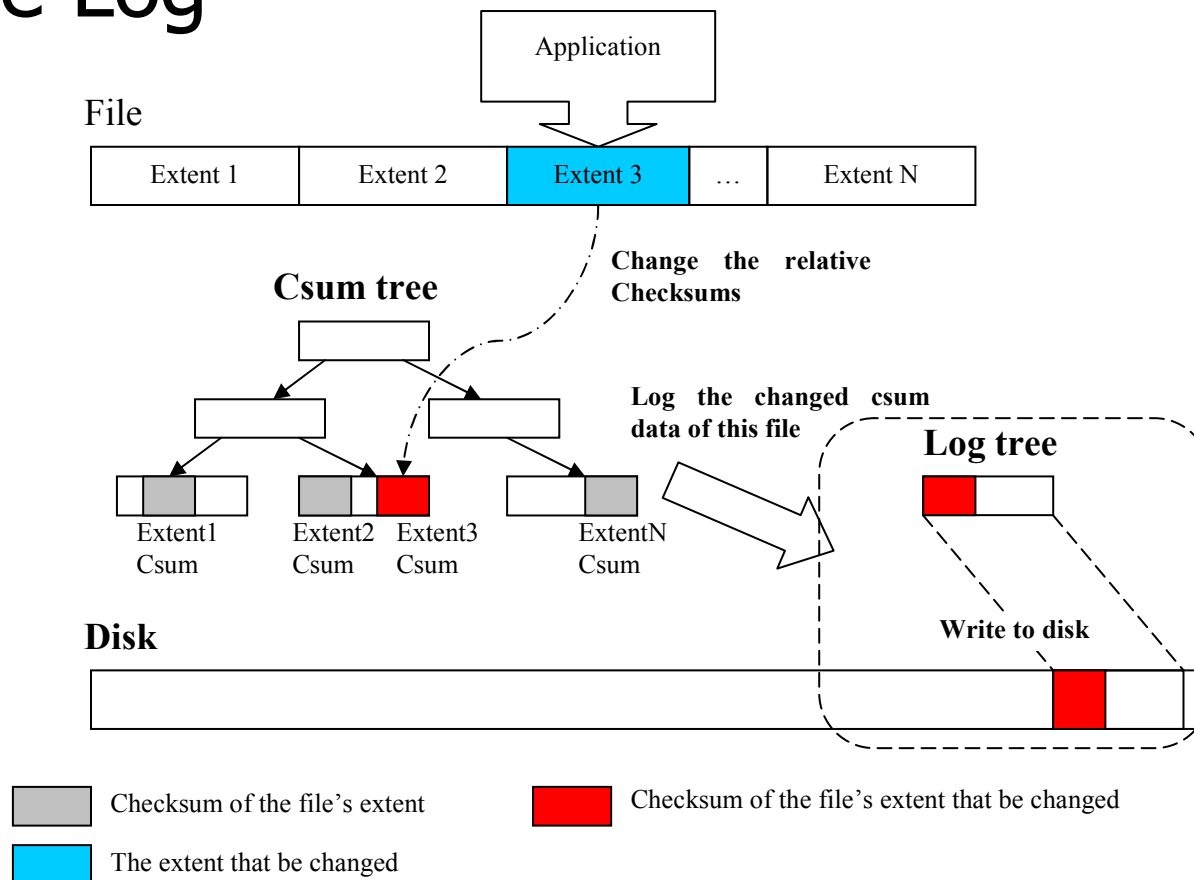
Implementation of the features

- SSD support
 - Free space cluster
- Seed Device support



Implementation of the features

■ Tree Log



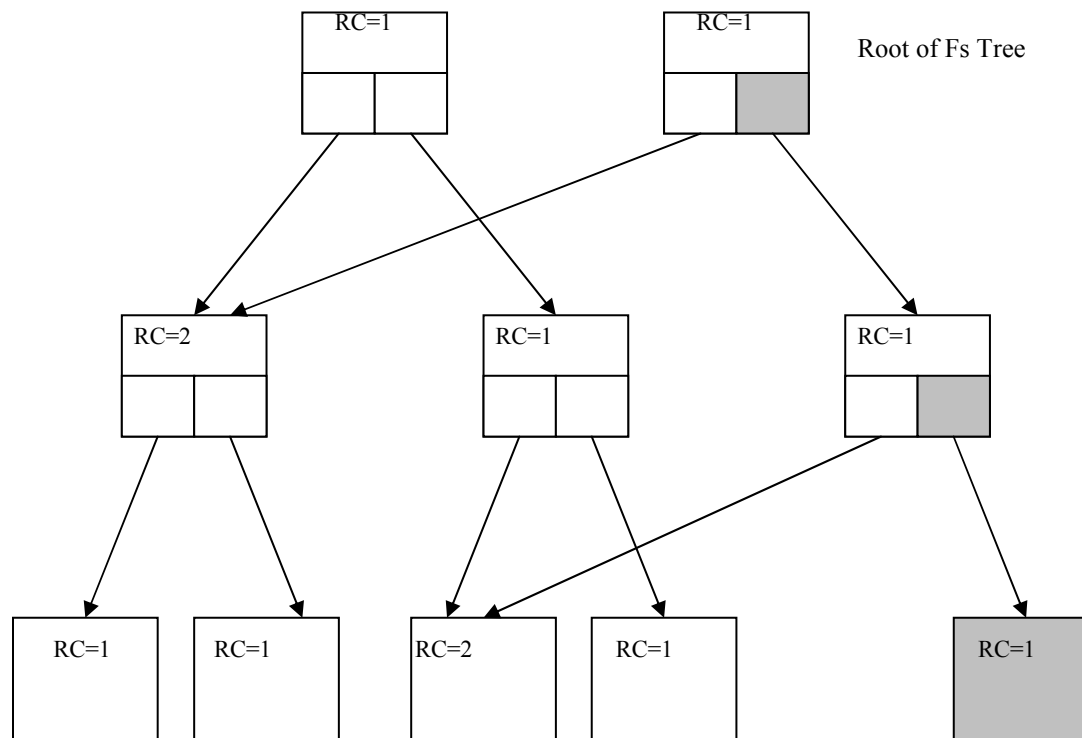


Implementation of the features

- Snapshot
 - Use the reference counter of the extent
 - Create a new file tree and copy the root of the original fs/file tree
 - Insert directory item and directory name index into the parent fs/file tree

Implementation of the features

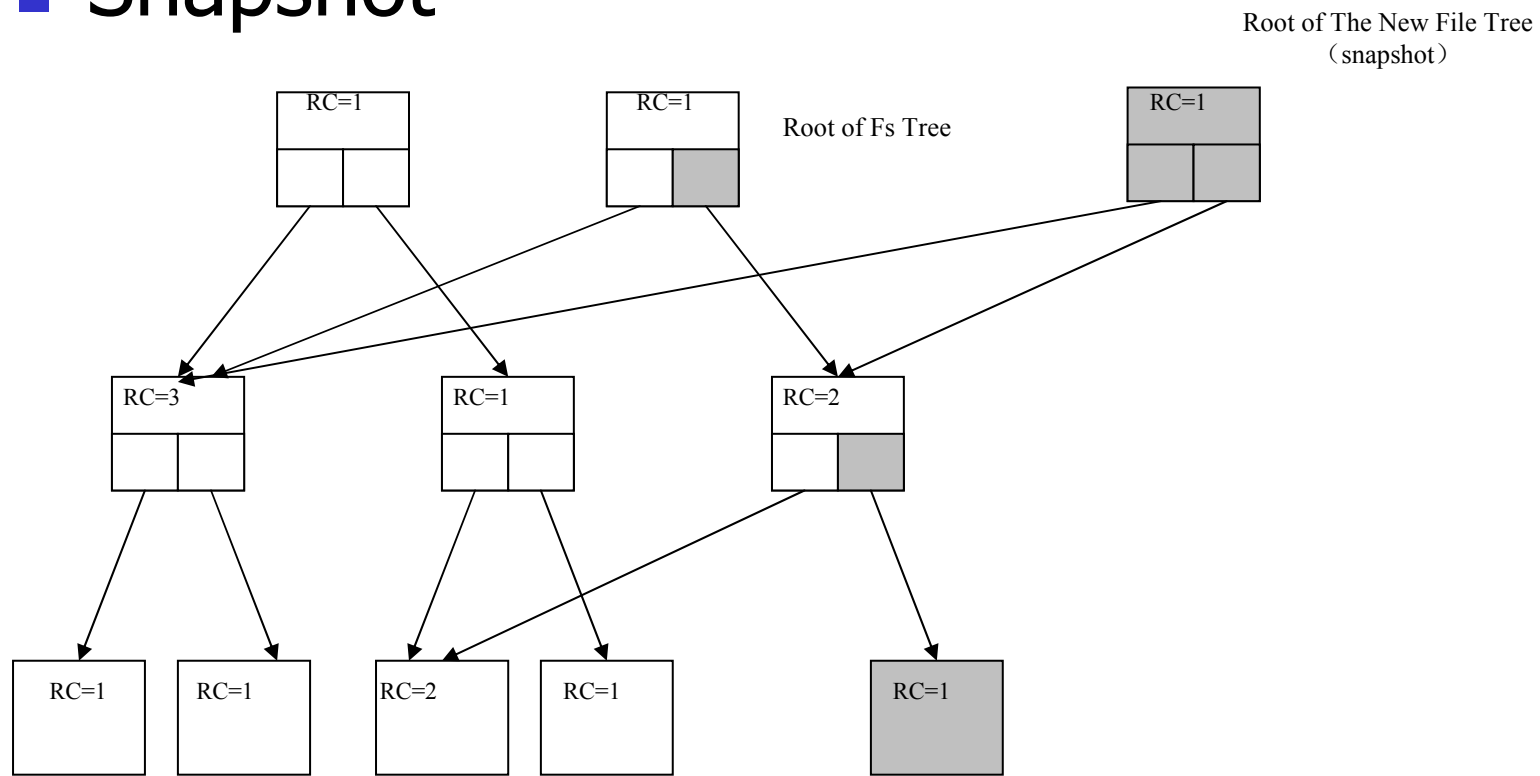
- Snapshot



Step1 before creating snapshot

Implementation of the features

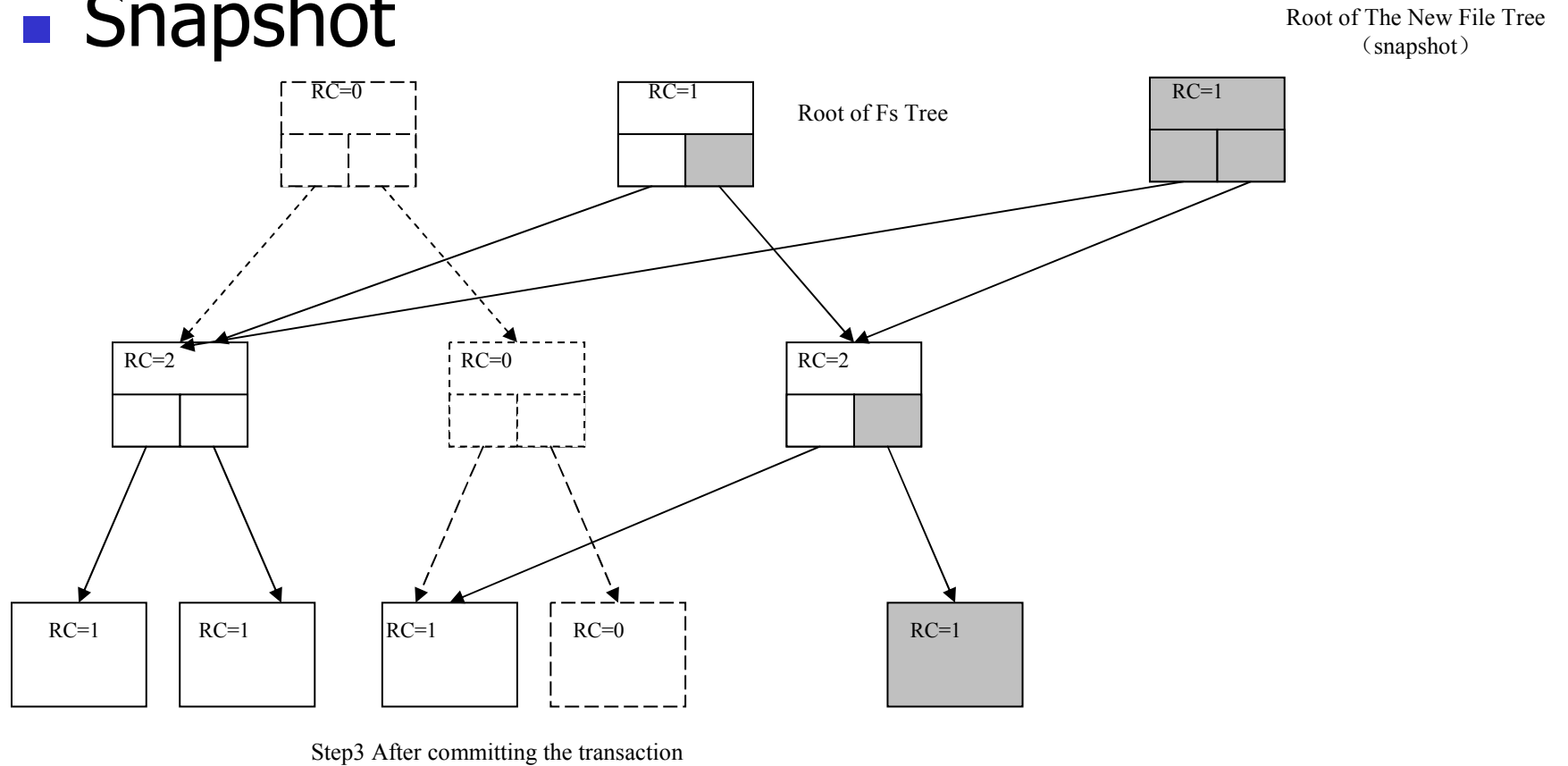
■ Snapshot



Step2 after creating snapshot

Implementation of the features

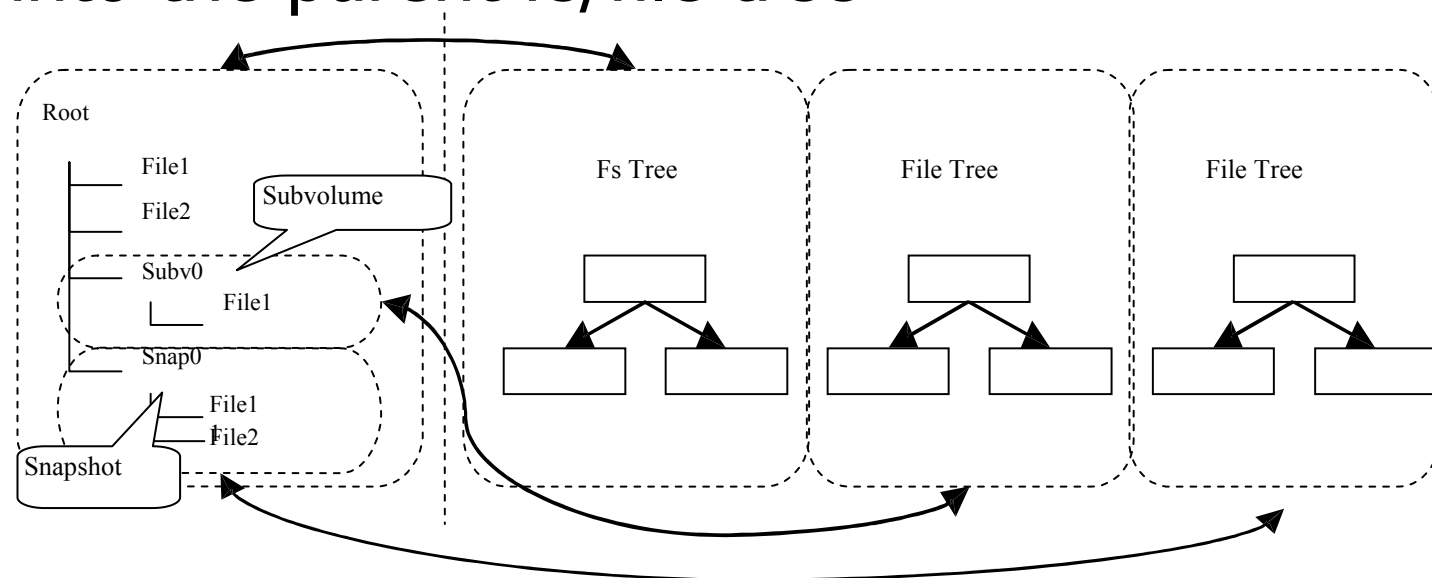
■ Snapshot



Implementation of the features

■ Subvolume

- Insert a new file tree into the root tree
- Insert directory item and directory name index into the parent fs/file tree





Agenda

- What is Btrfs?
- Why is Btrfs needed?
- Which features does Btrfs include?
- How is Btrfs implemented?
- **What did we do for Btrfs?**
- How do I use Btrfs?



What did we do for Btrfs?

- Introduce LZO compression
- Inode ID allocator
- Improve Chunk allocator
 - Utilize the device space better
- Improve tree log
 - By introduce the sub transaction id
- Delayed metadata update
- ...



Agenda

- What is Btrfs?
- Why is Btrfs needed?
- Which features does Btrfs include?
- How is Btrfs implemented?
- What did we do for Btrfs?
- How do I use Btrfs?



How do I use Btrfs?



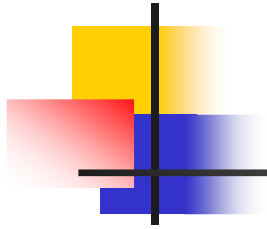
Reference

- 新一代 Linux 文件系统 btrfs 简介 (刘明)
<http://www.ibm.com/developerworks/cn/linux/l-cn-btrfs/>
- Btrfs Wiki
<http://btrfs.wiki.kernel.org/>
- Btrfs Mail List
linux-btrfs@vger.kernel.org



Reference

- Sourcecode (Git)
 - <http://git.kernel.org/?p=linux/kernel/git/mason/btrfs-progs-unstable.git>
 - <http://git.kernel.org/?p=linux/kernel/git/mason/btrfs-unstable.git>



Thanks!

Q/A