FUJITSU

shaping tomorrow with you

How to Be A Filesystem Developer

Aug 10 2015

Qu Wenruo <quwenruo@cn.fujitsu.com>
Fujitsu Limited.

Index

- Why to be a **kernel** developer
- **Why** to be **filesystem** developer
- **Why** to be a **Btrfs** developer
- **How** to be a FS(**Btrfs**) developer
- **Fujitsu** Contribution to Btrfs

Why to be a **kernel** developer

- For Individual
  - **Fame** of Old-school hacker among community
  - Polish the **skill** and more **challenge**
  - More **$$$**

- For Enterprise
  - **Reputation**
  - **Lead** on **latest** technology
  - Easier **maintenance**

Why to be a **filesystem** developer

- Increasing demand on **storage**
  - Big data(GFS)
  - Flash storage(SquashFS/Jffs2/F2FS)
  - Container(Btrfs)
  - VM
  - …

Why to be a **Btrfs** developer

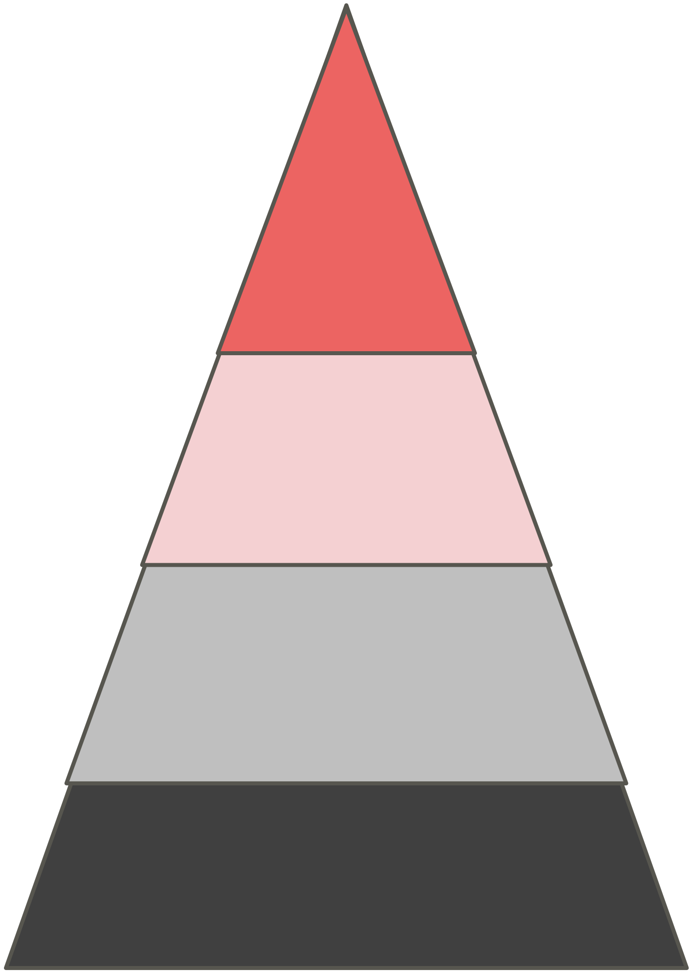
- Bleeding Edge
  - **Feature rich**
    - Already have
      - CoW, compression, deduplication, RAID …
    - Under active development:
      - Inband deduplication
      - Subpage sector size support
      - Separate qgroup accounting for medata and data
      - …
  - **Bugfix**
    - Btrfs is not as stable as traditional fs like xfs/ext4 yet
    - More features -> More code -> More bugs

Why to be a **Btrfs developer**

- Next generation filesystem
  - Btrfs is considered as next generation filesystem
  - Some projects are already using btrfs
    - Systemd
    - Docker
    - OpenSUSE
    - Facebook
    - …
  - Lead on latest technology
    - Example: Oracle Database with ZFS deduplication

How to be a FS(**Btrfs**) developer

- Road map to be a Btrfs developer



Kernel Btrfs developer

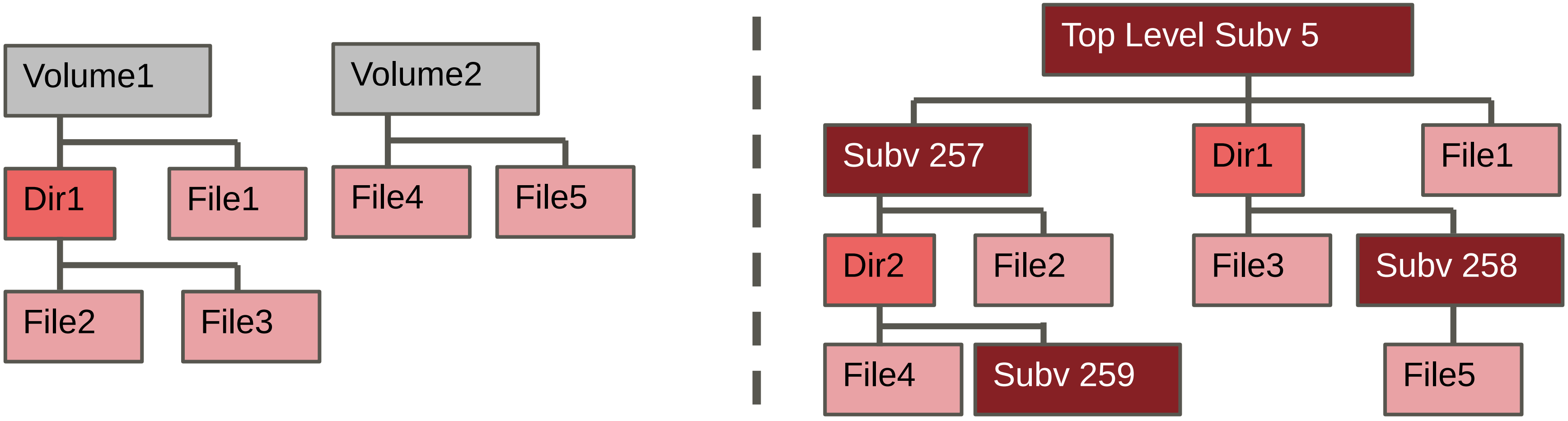
Btrfs-progs developer

Understanding on-disk format

Normal use, bug report and QA

How to be a FS(**Btrfs**) developer

- Normal use bug report and QA
  - Try to **use/test** btrfs
    - Recommend **rolling** distribution to test latest filesystem
      - **Latest** kernel
      - **Latest** btrfs-progs
      - For example **Archlinux**
    - Understanding **Subvolume/Volume** usage



Volume Level Management

Subvolume Level Management

How to be a FS(**Btrfs**) developer

- Normal use bug report and QA
  - Bug report
    - With **detailed** info
      - Kernel/btrfs-progs version
      - Kernel backtrace if needed
      - **Reproducer** if reproducible
  - QA
    - Need a little more **skills**
      - **Compile** latest kernel and btrfs-progs **from source**
      - **Git** (from clone to bisect)

How to be a FS(**Btrfs**) developer

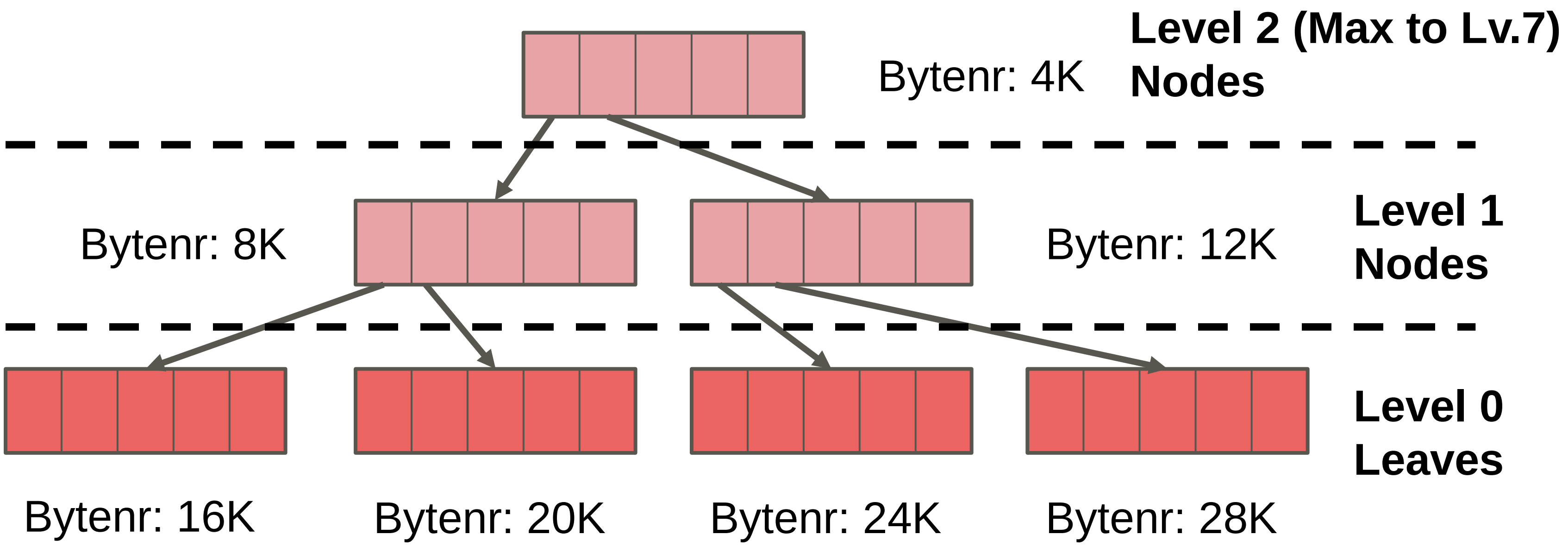
- QA
  - Performance test
    - Phoronix Test Suite
      - All in one, with openbenchmark database, nice chart
    - Sysbench
    - Fio
    - …
  - Function test
    - Fstests(old xfstests)
    - LTP
  - Testing thoery
    - Stress test
    - Regression test
    - …

How to be a FS(**Btrfs**) developer

- Understanding on-disk format
  - Why starting from that
    - On-disk data is **static**
    - No **C codes** involved
    - Existing good **tool** to exam them: **dumpe2fs, btrfs-debug-tree**

How to be a FS(**Btrfs**) developer

- Understanding on-disk format
  - Btrfs stands for **B-tree fs**
  - All metadata is stored in a **B-tree**



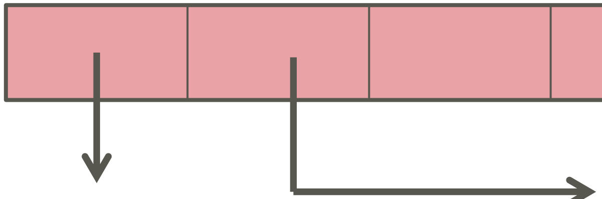
How to be a **Btrfs developer**

Inside btree

Node

- Records pointer (bytenr) to its child lead/node

Node at
Bytenr: **8K**



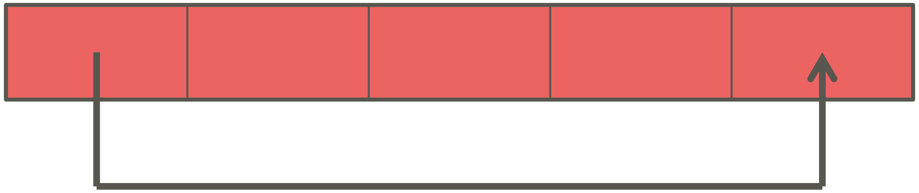
Child leaf/node is at bytenr 16K
First key is **(257, EXTENT_DATA,0)**

Child leaf/node is at bytenr 20K
Child first Key is (XX, XX, XX)

Leaf

- Records detailed info with its index **KEY**

Leaf at
bytenr: **16K**



Index Key is **(257, EXTENT_DATA, 0)**
Extra data is at XXX offset inside the leaf

How to be a FS(**Btrfs**) developer

- Practice with btrfs-debug-tree
  - With almost every **detail** of btrfs **b-tree**
  - Debug-tree -> do some opertaion -> debug-tree
    - Don't forgot to call '**sync**' before debug-tree
    - To see how btrfs records **files** and **dirs**
    - If **careful** enough, you can also see how Btrfs do **CoW**
    - '**fs tree**' should be the **easiest** start point

  - Reference
    - Extra explain on btrfs features(2015 LinuxCon Japan)
    - About each KEY type and corresponding data structure
      - https://btrfs.wiki.kernel.org/index.php/Btree_Items

How to be a FS(**Btrfs**) developer

- Btrfs-progs developer
  - Why starts from btrfs-progs?
    - Single thread (mostly)
      - No concurrence, no hidden deadlock(mostly)
    - **Direct** metadata operation
      - No extra infrastructure
      - Can use what you learn in previous step
    - **Quick** review
      - Special thanks for **David Sterb**
  - Needed skill
    - **C** programming
    - **GDB** for debugging
    - **Understanding Btrfs B-tree**

How to be a FS(**Btrfs**) developer

Development directions

Btrfs-debug-tree enhancement

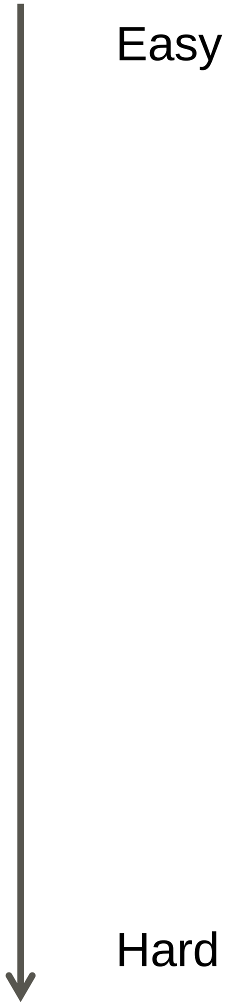
- **Easiest** one
- Can **refer** to existing codes quite easily
- Help you to understand b-tree

Btrfsck enhancement

- More **challenge**
- A little complicated data structure
- May fix your own problem

Btrfs-convert debug

- Most **complicated**
- Needs to refer to **kernel** codes
- Not **solved** yet



Easy

Hard

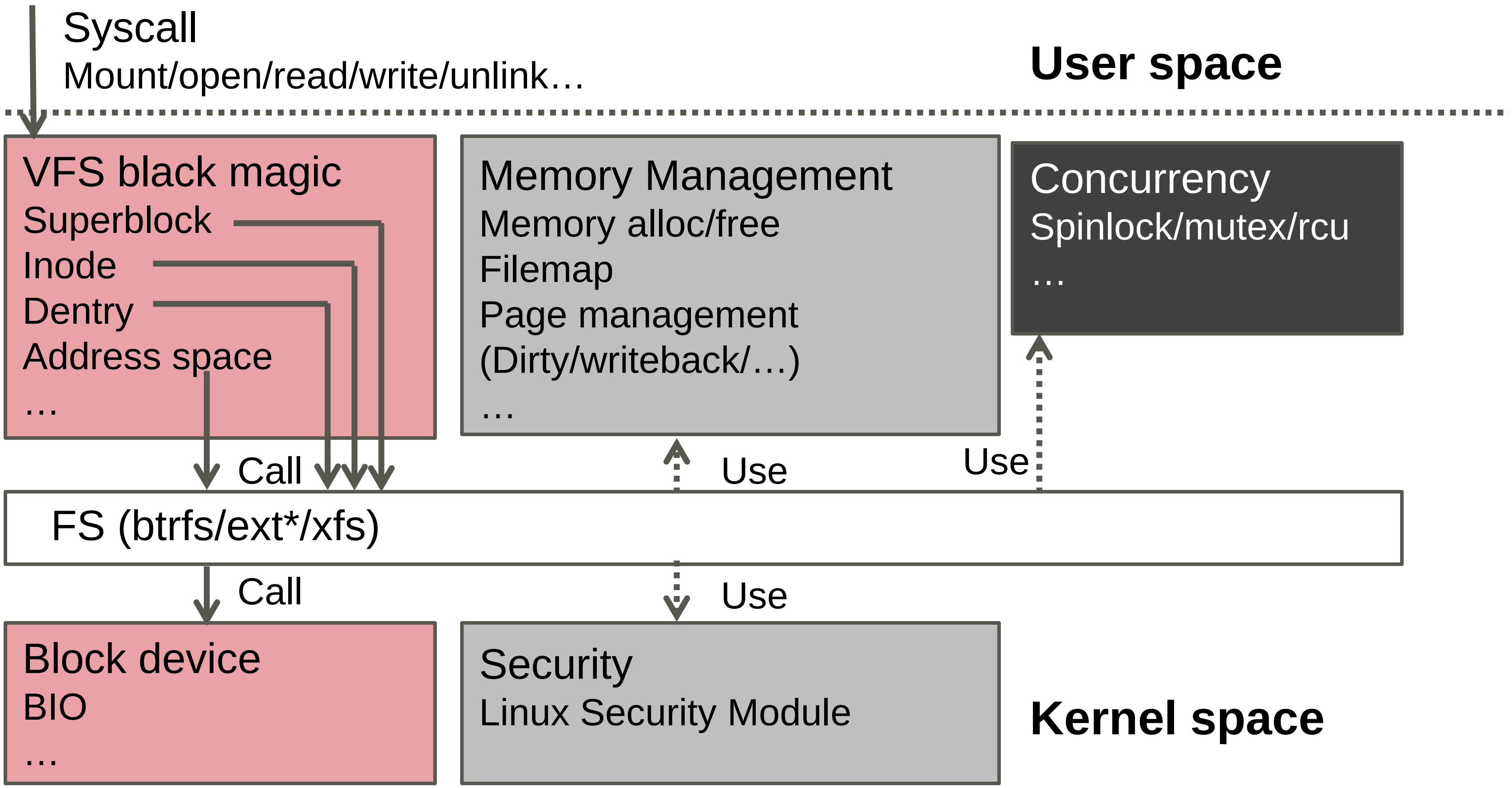
How to be a FS(**Btrfs**) developer

- Btrfs kernel developer
  - The hardest part, a lot of challenges
    - Extra kernel **facility** (from VFS to memory management)
    - Kernel trace/**debugging**.
    - **Concurrency** (Hell of **deadlock**)
    - Old, bad commented codes.
    - …
  - Needs **much more time** to test
    - Just make it run, without panic/BUG_ON/warning
    - Function test
    - Performance test
  - But also huge **accomplishment** when patch is merged

How to be a FS(**Btrfs**) developer

FUJITSU

- Challenges(Kernel **facilities**)



Syscall
Mount/open/read/write/unlink…

User space

VFS black magic
Superblock
Inode
Dentry
Address space
…

Memory Management
Memory alloc/free
Filemap
Page management
(Dirty/writeback/…)
…

Concurrency
Spinlock/mutex/rcu
…

FS (btrfs/ext*/xfs)

Block device
BIO
…

Security
Linux Security Module

Kernel space

- Kernel Doc => LWN => Google => RTFC(fs) => RTFC(facility)

How to be a FS(**Btrfs**) developer

- Challenge (kernel **facilities**)
  - **Modern** filesystem also implement quite a lot **optimization**.
    - Delay allocation
      - At **buffered** write time, only early check is done, no space is **allocated** until write time
    - Page cache
      - These unwritten data is stored in **page cache** by MM.
      - FS need to keep page cache **up to date** under a lot of operations(fallocate, truncate, unlink…)
    - …
  - Tons of **minor** features
    - Direct IO
    - Fsync
    - …
  - Solution?
    - Read the **“Funning”** Code, **Again and again**

How to be a FS(**Btrfs**) developer

- Challenges(Kernel trace/**debugging**)
  - Hard to debug compare to user-space program
    - **Recompile** takes a lot of time
    - Kernel panic is hard to **capture**
    - Hard to set **breakpoint/watchpoint**
    - …
  - Solutions
    - Use **ccahe/distcache** , and only recompile given **module**
    - Use **Kdump** to capture crash
    - Use **VM with gdb** to set kernel breakpoint/watchpoint
      - Or old fashion pr_info()
    - …

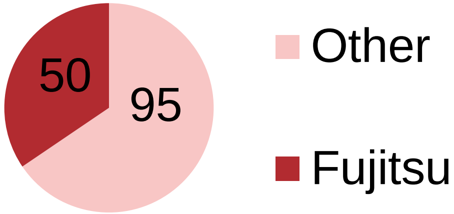
How to be a FS(**Btrfs**) developer

- Challenges(**Concurrency**)
  - Kernel is designed for performance, not education.
  - Concurrency is everywhere, tons of **lock, mutex, workqueue, wait_event**
  - **Lockdep** is the best solution.
    - Need to **enable** in kernel config
    - It’s **runtime** detection, needs tests to trigger it.
    - With quite good output **explaining** how it will cause **deadlock**
    - But not perfect, only detect spinlock/rwlock/mutex and so on. **Not support** for **wait_event**
  - ‘echo w > /proc/sysrq-trigger’ as **fallback** method
    - Just read the **“Funning”** code

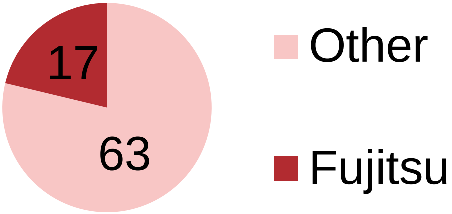
Developments Statistics

- Btrfs Kernel Contribution from Fujitsu

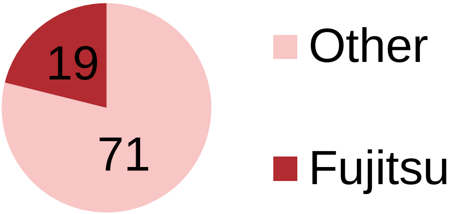
3.18 Commits



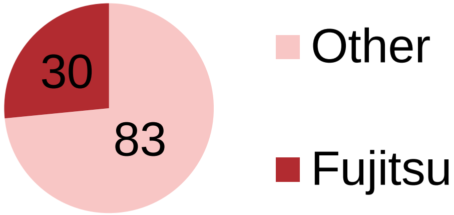
3.19 Commits



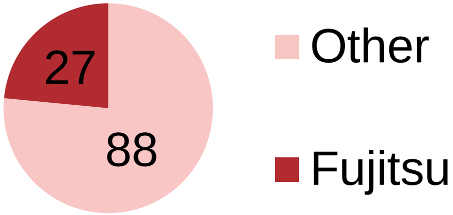
4.0 Commits



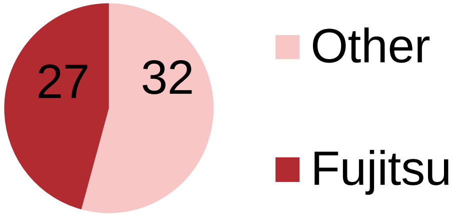
4.1 Commits



4.2 Commits

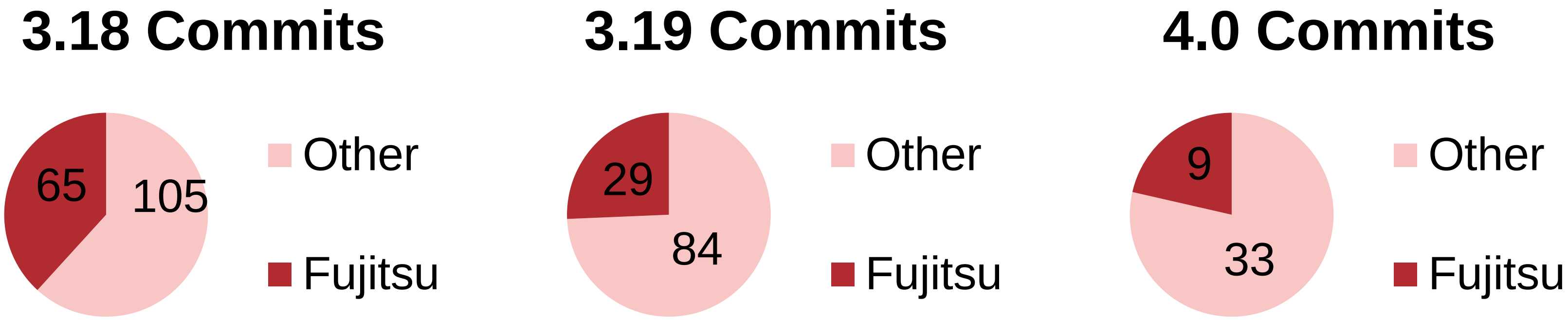


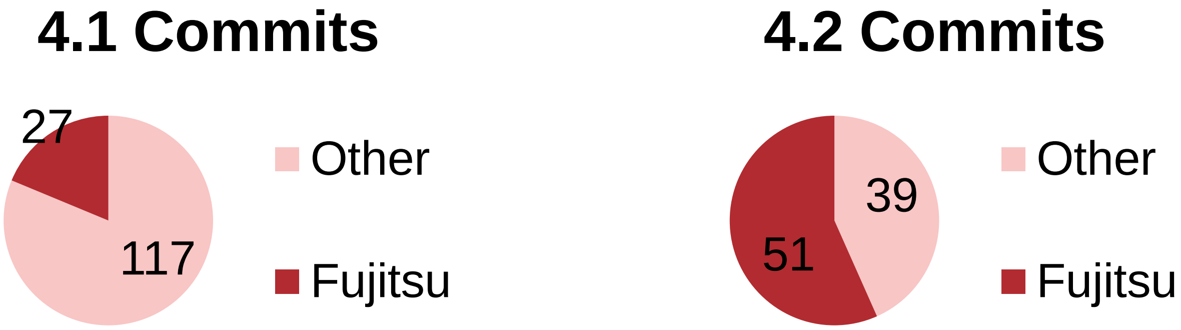
4.3 Commits



Developments Statistics

- Btrfs-progs Contribution from Fujitsu





Index

- Future Plans
  - Quota Reserve space framework rework
  - Inband de-duplication
  - Btrfs-convert rework
  - RAID5/6 readahead

Future Plans

- Quota reserve space framework rework(Submitted)
  - Accurate quota reserve space for write
  - Avoid reserve same space for several times
  - Use a rb-tree tree to record which data range is already reserved
  - Submitted too late and patchset too big, will be delayed to 4.4.

- Inband de-duplication(RFC submitted)
  - In memory extent<->hash tree
  - Controllable overhead and memory usage
  - On-disk extent<-> hash tree planned (Thanks **Liu Bo**)

Future Plans

- Btrfs-convert rework(WIP)
  - Btrfs-convert bugs already located
  - Support separate chunks profile after convert

- RAID5/6 readahead
  - Make it work again
  - May created a unified readahead framework

QA

- QA Time now

FUJITSU

shaping tomorrow with you