Implementation of Ext4fs Features in NetBSD Kernel

AsiaBSDCon 2017
Hrishikesh Goyal
https://github.com/hrishikeshgoyal/

Personal Info

Implemented Ext4 FS read support for NetBSD kernel,
The NetBSD Foundation, Google SoC 2016

BTech Graduate 2016
Computer Science and Technology
National Institute of Technology
Warangal (India)

Agenda

Filesystem
VFS in NetBSD
Ext4fs Features and NetBSD Kernel
File Data Block Mapping
Extent Support
Double Cache Problem
UBC Interface
HTree Index for DIR Entries

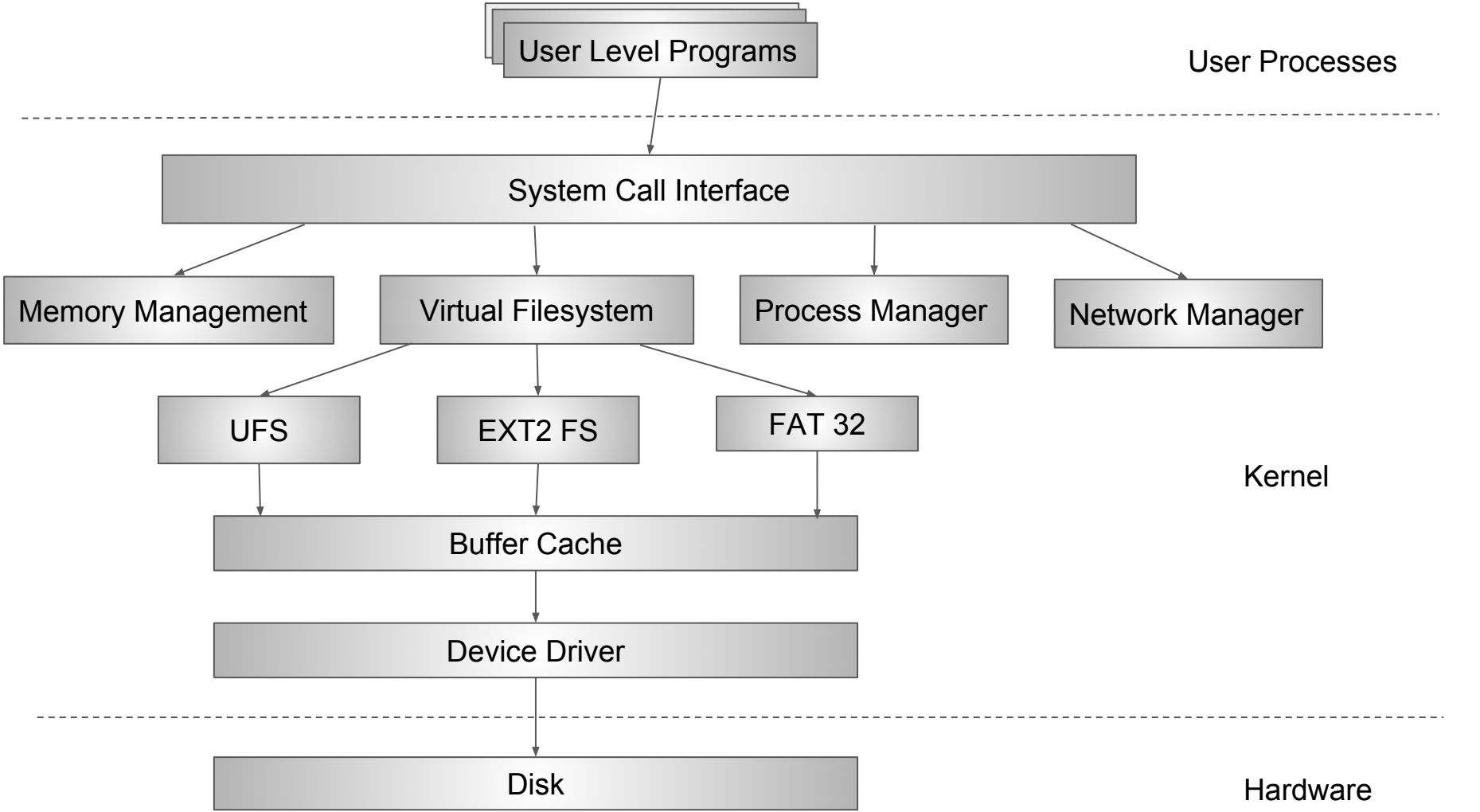
File-system

Concrete

Filesystem is the code of the kernel which controls how data is stored and retrived.

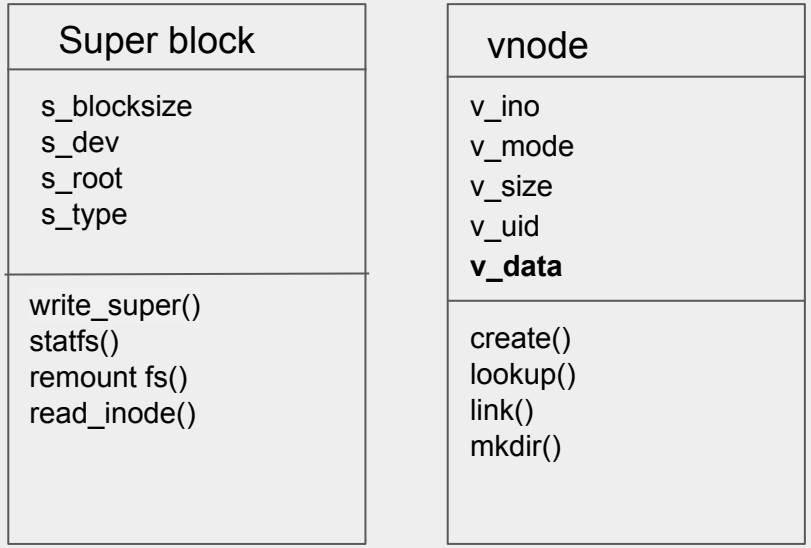
Virtual

Virtual filesystem is an abstraction layer on the top of the concrete filesystem with which rest of the kernel deals.


User Level Programs
User Processes
System Call Interface
Memory Management
Virtual Filesystem
Process Manager
Network Manager
UFS
EXT2 FS
FAT 32
Kernel
Buffer Cache
Device Driver
Disk
Hardware

Virtual Filesystem Layer

1. Vnode Interface
2. VFS Interface



Fourth Extended Filesystem (Ext4 FS)

Introduced a number new features

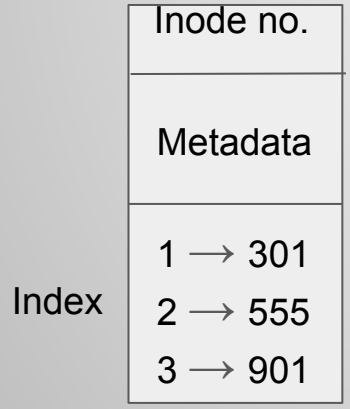
1. Bigger filesystem/file size
2. Subdirectory scalability
3. Extents
4. Multiblock allocation
5. HTree DIR Index (Ext3fs feature)

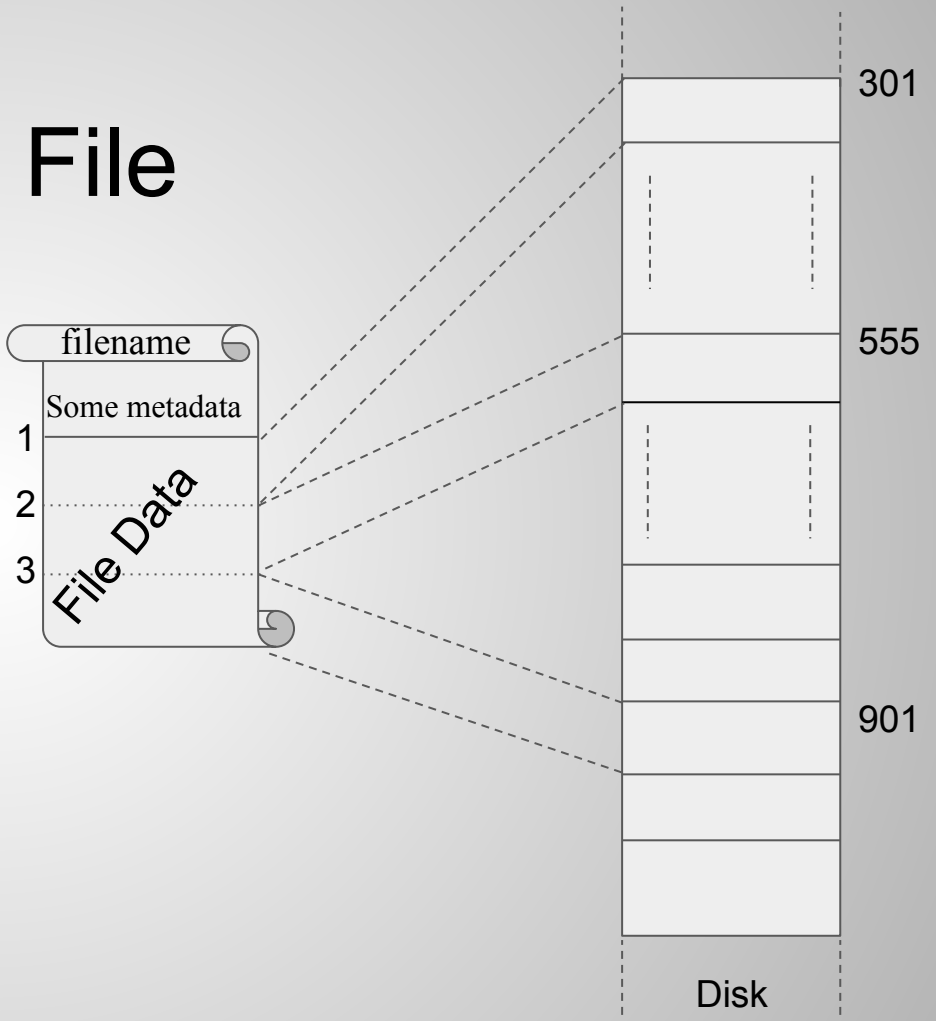
Ext4fs in NetBSD

Ext2fs is the base filesystem and few features from Ext3fs and Ext4fs are supported.

Inode

Inode: An abstract representation of a file in the filesystem.





Ext2 Data Block Mapping (Indirect pointers approach)

Index consist of 15 pointers (60 bytes)

- 12 Direct pointers
- 1 Indirect pointer
- 1 Double Indirect pointer
- 1 Triple Indirect pointer

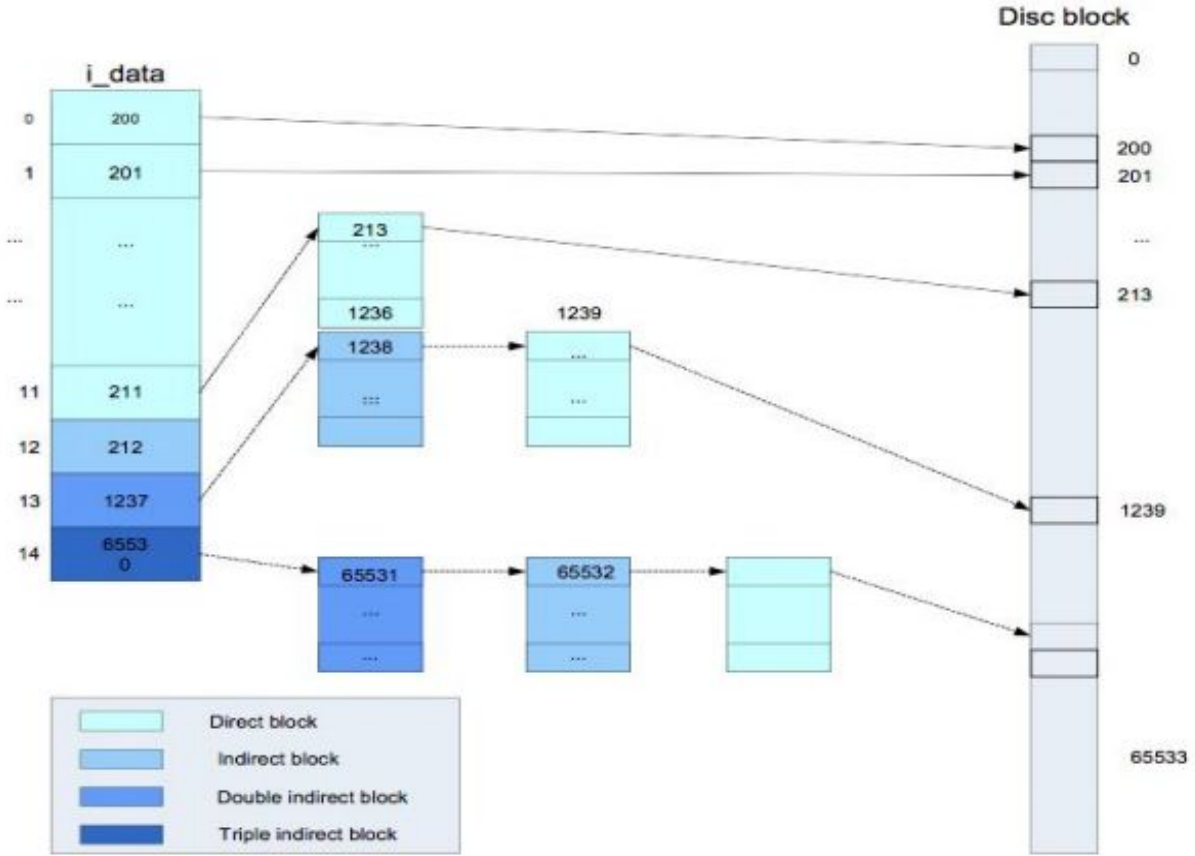
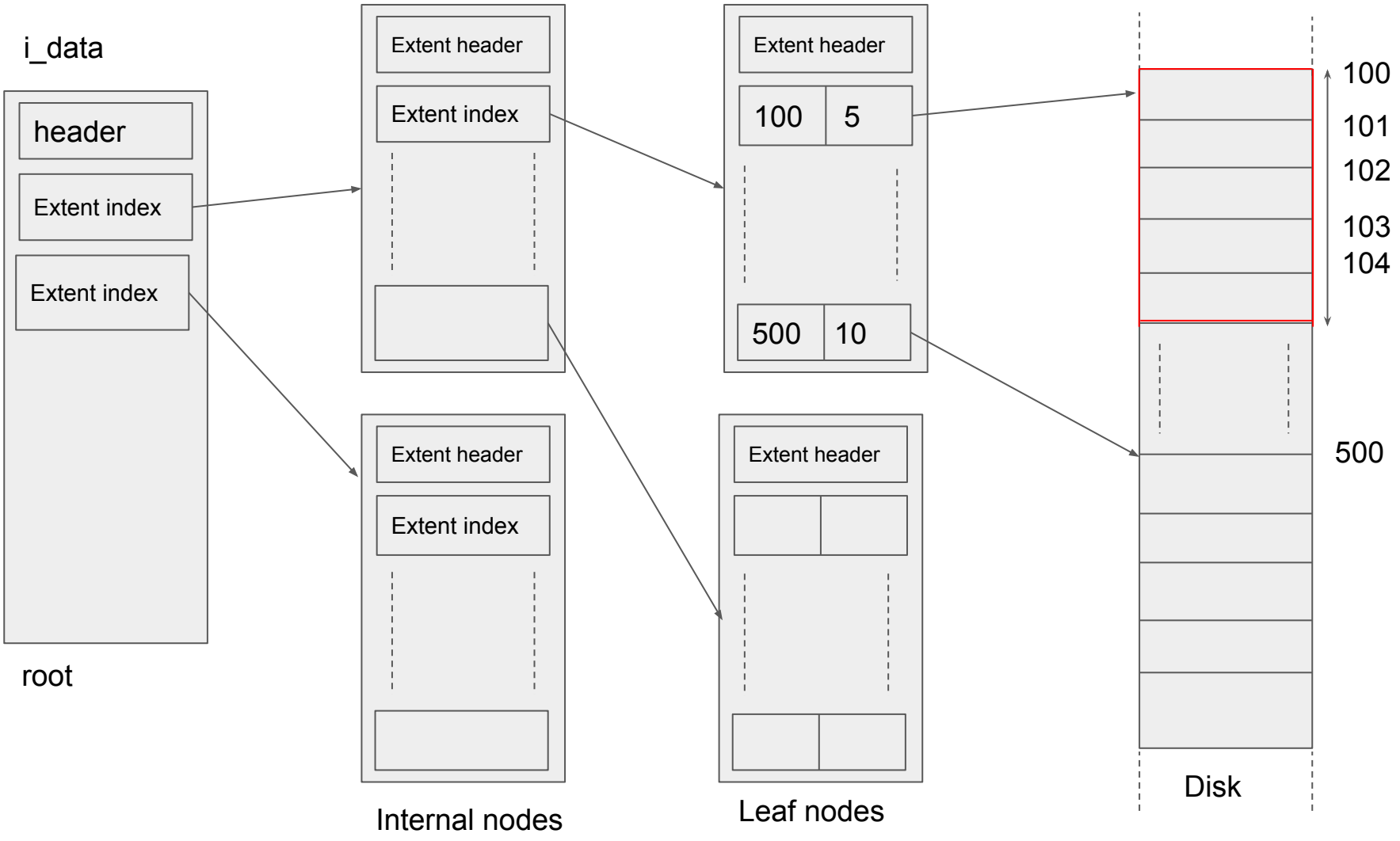


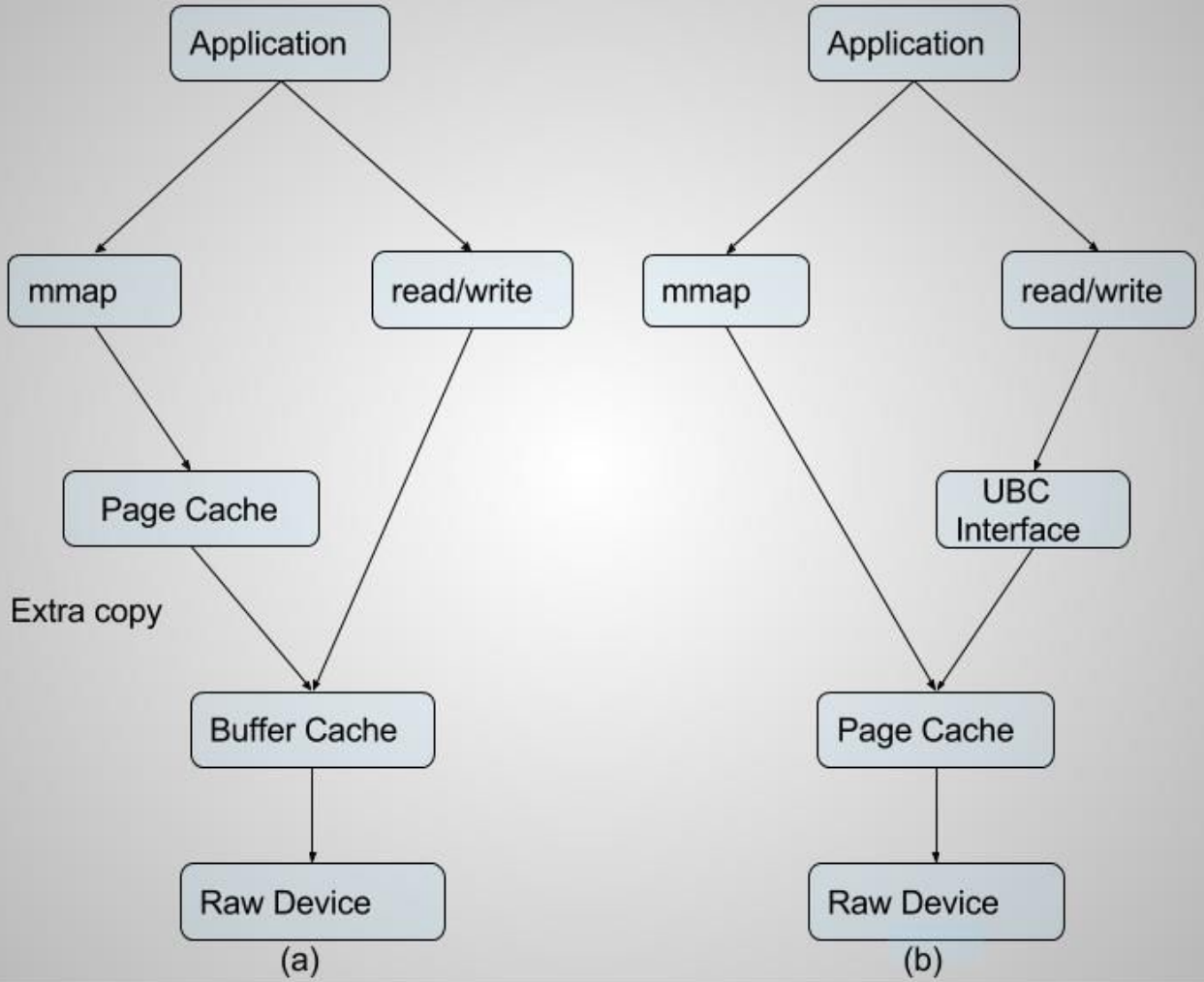
Figure 1 Indirect block mapping

EXT4 EXTENT

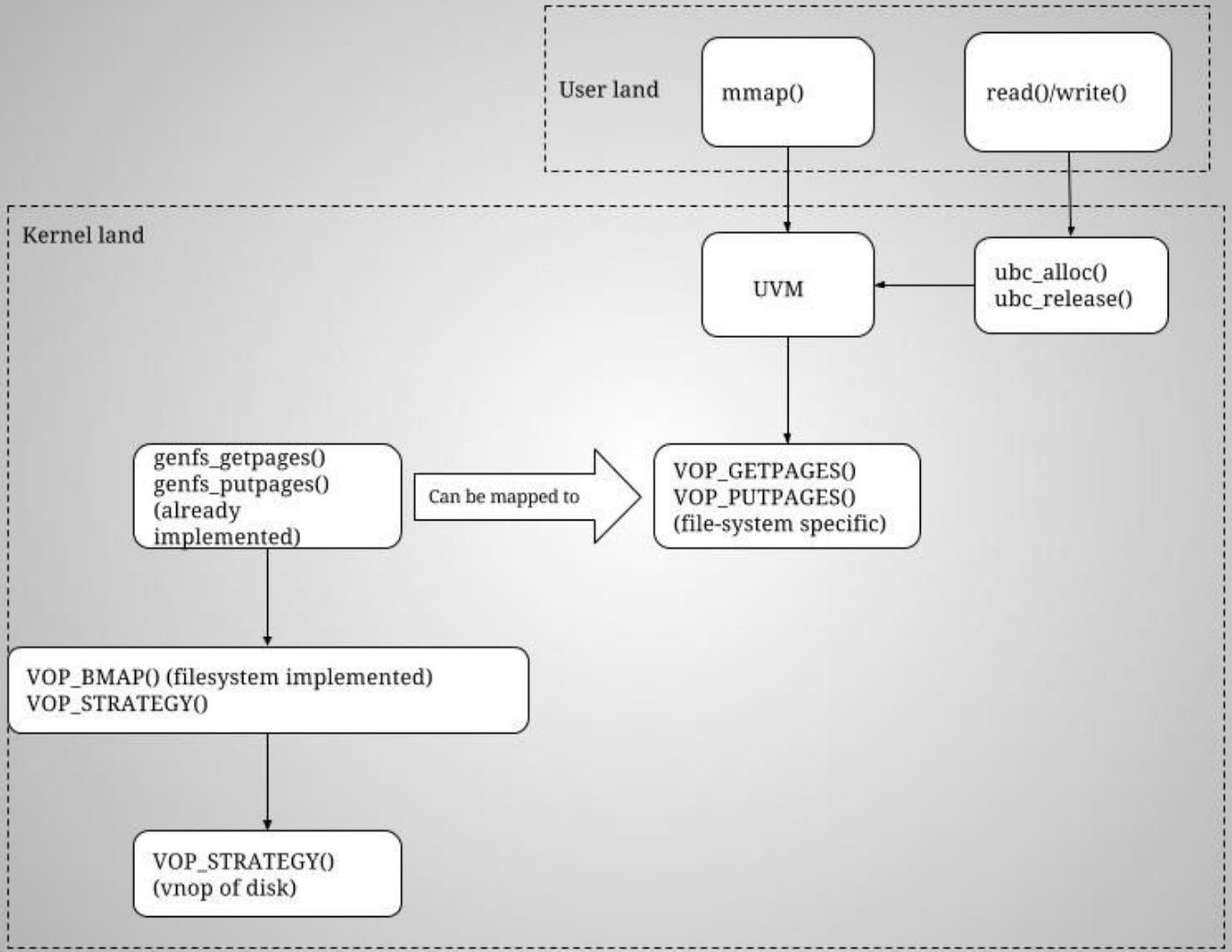
- Based on B+Tree Indexing
- One pointer for a bunch of contiguous file data blocks


i_data
header
Extent index
Extent index
root
Extent header
Extent index
Extent header
Extent index
Internal nodes
Extent header
100
5
500
10
Extent header
Leaf nodes
100
101
102
103
104
500
Disk

Double Cache Problem


Application
mmap
read/write
Page Cache
Extra copy
Buffer Cache
Raw Device
(a)
Application
mmap
read/write
UBC
Interface
Page Cache
Raw Device
(b)

Unified Buffer Cache


User land
mmap()
read()/write()
Kernel land
UVM
ubc_alloc()
ubc_release()
genfs_getpages()
genfs_putpages()
(already
implemented)
Can be mapped to
VOP_GETPAGES()
VOP_PUTPAGES()
(file-system specific)
VOP_BMAP() (filesystem implemented)
VOP_STRATEGY()
VOP_STRATEGY()
(vnop of disk)

Directory

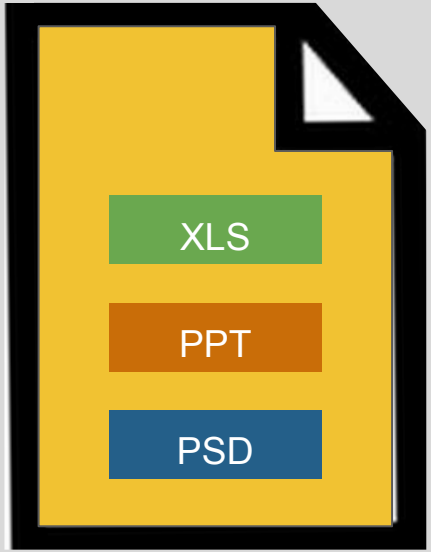
Directory as a file

Directory entries

Ext2fs Directory structure

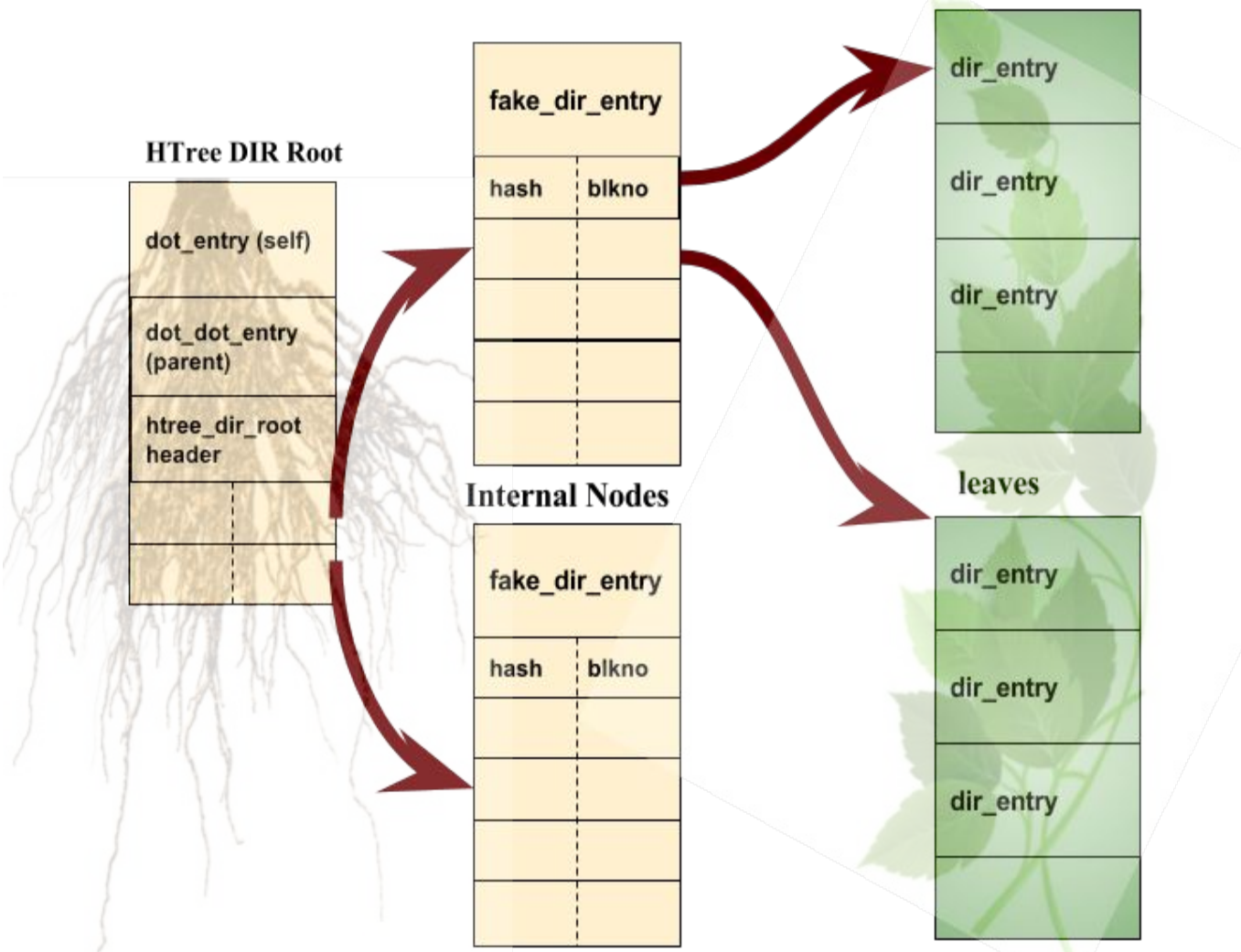
As what it looks like..

What actually it is.. !!!



HTree DIR Index

- Uses hash of the filename as the key of the HTree.
- Directory Index Block(internal nodes)
- Directory Entries Block(leaf nodes)


HTree DIR Root
dot_entry (self)
dot_dot_entry (parent)
htree_dir_root header
fake_dir_entry
hash
blkno
Internal Nodes
fake_dir_entry
hash
blkno
dir_entry
dir_entry
dir_entry
leaves
dir_entry
dir_entry
dir_entry

Thank you

<hrishi.goyal@gmail.com>