รายงาน

ASSIGNMENT vCowFs

วิชา 0107625 OPERATING SYSTEMS

เสนอ

ดร.อรทัย สังข์เพ็ชร

ดร.อักฤทธิ์ สังข์เพ็ชร

จัดทำโดย

นายนวพงศ์ อารีกุล รหัสนักศึกษา 54010685

นายกฤตภาส คุ้มถนอม รหัสนักศึกษา 57010027

นางสาวจิตตินาท นารถนรกิจ รหัสนักศึกษา 57010176

นายเจตพล พุ่มวัฒนกุล รหัสนักศึกษา 57010216

นางสาวฐิติมา ปาละวัฒน์ รหัสนักศึกษา 57010352

นางสาวณราวรรณ เวชประสิทธิกุล รหัสนักศึกษา 57010379

นายนายธีรชา บุษปพงศ์พันธุ์ รหัสนักศึกษา 57010625

นางสาววรัมพร พงศ์กล่ำ รหัสนักศึกษา 57011066

นางสาวพัชริดา อารีย์สมาน รหัสนักศึกษา 57011567

ภาคเรียนที่ 2 ปีการศึกษา 2559

ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

คำนำ

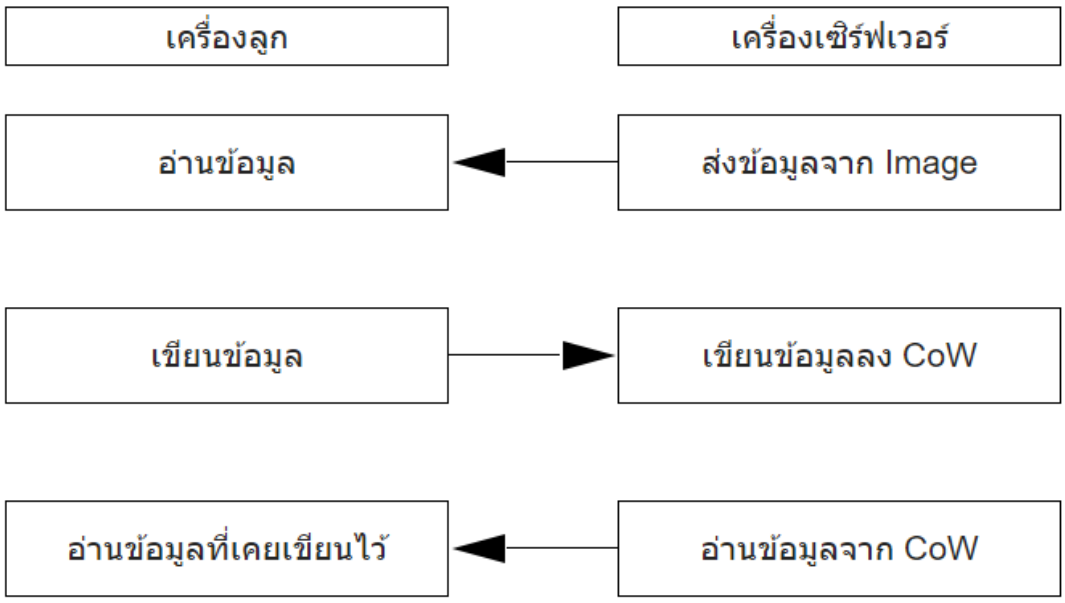
รายงานเล่มนี้เป็นส่วนหนึ่งของรายวิชา OPERATING SYSTEMS รหัสวิชา 0107625 ซึ่งเป็นรายงานที่ใช้ประกอบชิ้นงานการสร้างfile systemขึ้นเองซึ่งfile system ชนิดนี้เป็น versioning copy-on-write file system คือมีความสามารถ จะเรียกไฟล์เวอร์ชันก่อนหน้าขึ้นมาใช้งานได้ โดยรายงานเล่มนี้ประกอบไปด้วย รายละเอียดการออกแบบและรายละเอียดต่างๆในกระบวนการสร้าง file system หวังว่ารายงานเล่มนี้จะมีประโยชน์แก่ผู้ที่สนใจได้ไม่มากก็น้อย

คณะผู้จัดทำ

รายละเอียดการออกแบบ

Copy-on-Write Filesystem Design

Copy-on-Write คือเทคนิคในการใช้ไฟล์หลักร่วมกันจากหลายๆ เครื่อง แต่เมื่อเครื่องใดเครื่องหนึ่งจะเขียนข้อมูล ก็จะไปเขียนในเนื้อที่สำรองแทน ซึ่งเป็นที่มาของคำว่า Copy on Write โดยเนื้อที่สำรองนี้แต่ละเครื่องจะแยกจากกัน ทำให้แต่ละเครื่องสามารถเขียนข้อมูลได้อิสระจากกัน ส่วนการอ่านข้อมูลก็จะอ่านจากทั้งไฟล์หลัก และใน CoW โดยจะอ่านจาก CoW เฉพาะข้อมูลที่เครื่องดังกล่าวเขียนลงไป ส่วนข้อมูลที่ไม่มีการเขียนก็จะอ่านจากไฟล์หลัก



เมื่อเปิดเครื่องใหม่ ระบบจะล้าง CoW ออกทั้งหมด ทำให้ในแต่ละครั้งที่เปิดเครื่องใช้งาน สถานะของระบบจะกลับคืนสู่สภาพเดิม

แนวคิดการทำงานของ Copy-on-Write ด้านอัลกอริทึม

1. เมื่อมีผู้ใช้หลายๆคนเรียกใช้ทรัพยากรตัวหนึ่ง ซึ่งถูกตั้งค่ามาไม่ให้สามารถถูกแยกส่วนได้ ระบบก็จะสร้างพอยเตอร์ที่ชี้ไปยังทรัพยากรนี้ให้ผู้ใช้แต่ละคน
2. แต่หากมีผู้ใช้คนใดคนหนึ่งสั่งเปลี่ยนแปลงข้อมูลนั้นๆ ระบบก็จะทำสำเนาข้อมูลให้ผู้นั้น แล้วแก้ไขที่ตัวสำเนานั้นแทน เป็นการป้องกันไม่ให้ผู้อื่นเห็นข้อมูลที่เปลี่ยนแปลงนั้น

เป็นที่ มาของชื่อ copy-on-write ก็คือ จะทำการ copy ก็ต่อเมื่อต้องการจะ write ข้อมูลลงไปนั่นเอง

ข้อดีของ COW

ไม่จำเป็นต้องทำสำเนาข้อมูลนั้นเสมอเมื่อเรียกใช้ โดยจะทำสำเนาเท่าที่จำเป็นเท่านั้น คือเมื่อมีผู้ใช้ต้องการเปลี่ยนแปลงข้อมูล

ตัวอย่างของ COW ที่พบได้

ในภาษา C++

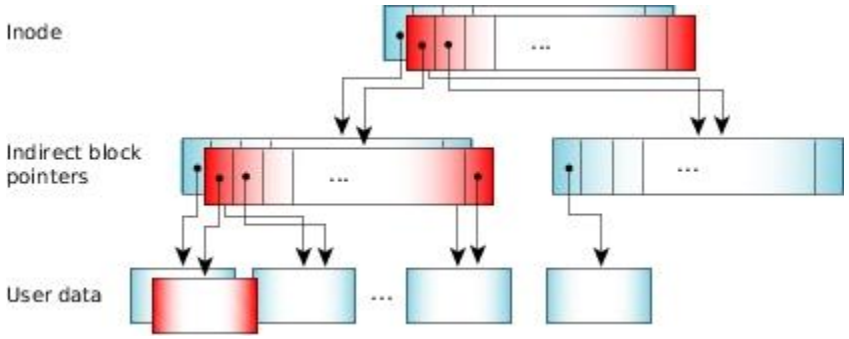
```
std::string x("Hello");


std::string y = x;


y += ", World!";
```

เริ่มแรก y และ x ในบรรทัดที่สองจะใช้ buffer ตัวเดียวกันแต่เมื่อมีการเปลี่ยนแปลงค่า y ในบรรทัดที่ 3 buffer ของ y ก็จะถูกแยกออกมาเป็นอีกตัวหนึ่ง

ในส่วนของการออกแบบ Copy-on-Write Filesystem Design



เราจะสร้าง pointer ชี้ไปยังข้อมูลที่ถูกอ้างถึงจากเครื่องอื่นๆ โดยที่หากเครื่องใดต้องการเปลี่ยนแปลงข้อมูลหลัก เราจะคัดลอกข้อมูลนั้นๆ เพื่อทำการแก้ไข โดยที่ไม่ได้แก้ที่ไฟล์ต้นฉบับ เมื่อดำเนินการเสร็จ ข้อมูลเดิมและตัวชี้ใหม่จะยังคงอยู่เหมือนเดิม แต่มีชุดใหม่ของบล็อค ตัวชี้ทางอ้อม เพิ่มขึ้นมาแทน

vCow File System Design

- files/directories

    ใน directory จะมี files กี่ files ก็ได้

    ใน directory จะมี directory ซ้อนอีกชั้นหรือมากกว่า 1 ชั้นก็ได้

    ไม่สามารถสร้าง directory ใน file ได้

    File สามารถมีชื่อซ้ำกันได้ถ้าไม่ได้อยู่ใน directory เดียวกัน

    Directory ไม่ควรมีชื่อซ้ำกัน

    Directory จะต้องเก็บlinkของข้อมูลของสิ่งที่อยู่ในตัวเองไว้ เช่น

    Dir1 จะมี link ที่ link ไปหาไฟล์ abc.txt acd.txt asd.txt และ link ที่ไปหา dir2

    ใช้ data structure แบบ

dir1-> abc.txt,acb.txt,asd.txt,dir2

dir2->dir3

dir3->dir4

dir4->abc.txt,aab.txt

- **File access mechanisms**

เลือกแบบ direct/random access

-space allocation

ใช้ linked allocation

-file data structure

Linked list

- **operations บน file และ directory**

int(* getattr)(const char *, struct stat *) เรียกดูสถานะของ File

int(* mknod)(const char *, mode_t, dev_t) สร้าง File ใหม่

int(* mkdir)(const char *, mode_t) สร้าง Directory ใหม่

int(* unlink)(const char *) ลบ File

int(* rmdir)(const char *) ลบ Directory

int(* rename)(const char *, const char *) เปลี่ยนชื่อ File หรือ Directory

int(* chmod)(const char *, mode_t) เปลี่ยน access control ของ File

int(* chown)(const char *, uid_t, gid_t) เปลี่ยน id ของเจ้าของ File

int(* truncate)(const char *, off_t) เปลี่ยนขนาดของ File

int(* open)(const char *, struct fuse_file_info *) เปิด File

int(* read)(const char *, char *, size_t, off_t, struct fuse_file_info *) อ่าน File

int(* write)(const char *, const char *, size_t, off_t, struct fuse_file_info *) เขียน File

int(* release)(const char *, struct fuse_file_info *) ปิด File

int(* opendir)(const char *, struct fuse_file_info *) เปิด Directory

int(* readdir)(const char *, void *, fuse_fill_dir_t, off_t, struct fuse_file_info *) อ่าน Directory

int(* releasedir)(const char *, struct fuse_file_info *) ปิด Directory

int(* fsync)(const char*, int, struct fuse_file_info *) บังคับเขียนข้อมูลในหน่วยความจำลง File

int(* fsyncdir)(const char*, int, struct fuse_file_info *) บังคับเขียนข้อมูลในหน่วยความจำลง Directory

Design Versioning

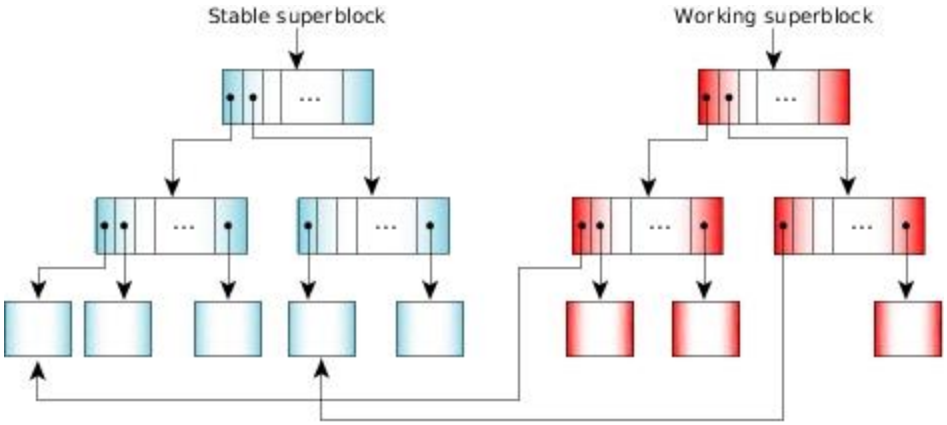
- ทุกครั้งที่มีการสร้างไฟล์ให้สร้าง ver1 ของไฟล์เก็บไว้ใน achive เสมอ
- ทุกครั้งที่มีการเขียนไฟล์ให้เช็คเวลาในการทำ version ใหม่ของชิ้นงาน โดยเทียบจากเวลาในการเริ่มเปิดชิ้นงานกับเวลาปัจจุบันในการทำงาน หากมากกว่าเวลาที่ตั้งไว้ในการทำ version ให้ทำไฟล์ version ใหม่ เก็บไว้ใน achive ไฟล์เดิมเสมอ

การทำ versioning

Superblock คือ global root block ที่บรรจุ inode สำหรับ file system bitmap และไฟล์ inode filesystem นี้จะแบ่งออกเป็น 2 superblocks ได้แก่

- Stable superblock ที่เป็น original version ของทุกๆ blocks

- Working superblock ที่เป็นข้อมูลที่ถูก modify แล้ว



Snapshot เป็น consistent view ของ file system

– เรียกได้ว่าเป็น supper block ที่เรามอบหมาย (commit) ให้ทำงาน-

เพื่อสร้าง snapshot ตัว file system ต้องทำตามขั้นตอนดังนี้

1. lock ตัว filesystem เพื่อให้แน่ใจว่าอยู่ในสถานะ stable ไม่มีการเปลี่ยนแปลงเพิ่มเติม

2.เขียน copy block ทั้งหมดลงบนดิสก์โดยไม่สนใจลำดับ ซึ่งตรงนี้จะทำให้เราสามารถ optimize ได้

3.ทำให้ข้อมูล synchronize กับดิสก์

4.สร้าง superblock - บันทึก location ใหม่ของ bitmap และ inode

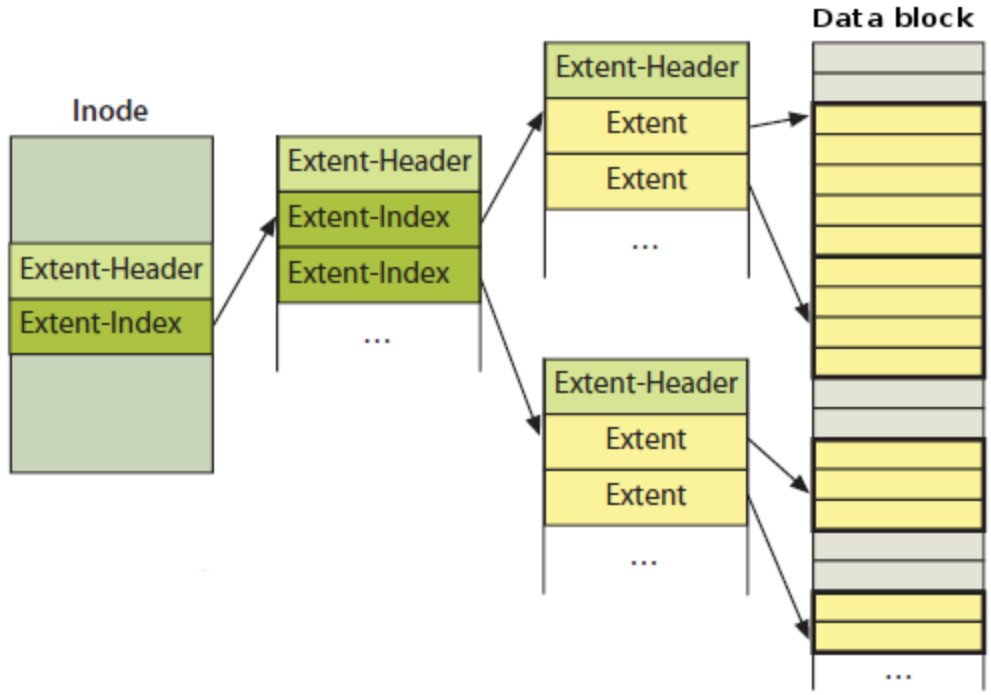
- เพิ่มเลข sequence number

- คำนวณ CRC

5.เขียน superblock ลงบนดิสก์

6.สับเปลี่ยน working view กับ committed view ตัว copy block เวอร์ชั่นเก่าจะโดน free และพร้อมสำหรับการใช้งาน

-อัลกอลิธึ่มสำคัญใน vCowfs



For larger files, Ext4 builds an extent tree

-โครงสร้างข้อมูลของการเก็บข้อมูลบน Disk/ Image File

Storage ที่ใช้ เรียกว่า Image File โดยเป็นไฟล์ที่มีอย่างน้อย 1 MB ซึ่งถูกเก็บอยู่บน File System ที่มีอยู่แล้วของระบบปฏิบัติการหลัก คือ ext4

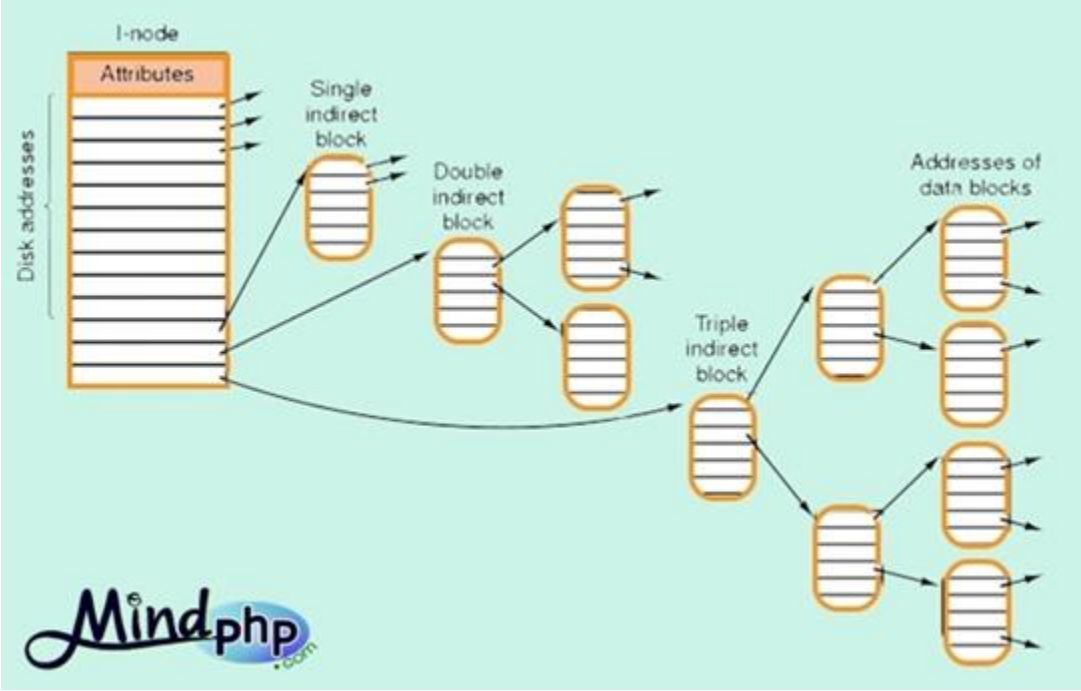
Makefile จะอยู่ในโฟเดอร์ runscript โดยที่ไฟล์ run.sh คือ makefile/mesonninja ซึ่งภายใน run.sh จะมี code ดังนี้

```
#!/bin/bash

        rm -r ../kaii
        mkdir ../kaii
        cd ../kaii
        meson ..
        ninja
```

โดยเราสามารถเรียกใช้ไฟล์ run.sh ได้โดย ./run.sh

Part of Design



Inode ใช้ในระบบ ปฏิการยูนิกส์โดยสร้างตารางเล็ก เรียนกว่าไอโหนด ให้กับแต่ละไฟล์ มีหน้าที่ก็บข้อมูลต่าง ๆ ที่เกี่ยวข้องกับไฟล์นั้นเอาไว้ ค่าจำพวก file permission, file owner และอื่นๆของแต่ละไฟล์ หาก partition หนึ่งมี จำนวน inode อยู่ 1,000 หมายความว่า partition นั้นจะมีไฟล์ได้เพียง 1,000 ไฟล์เท่านั้น แม้จะมี disk space เหลือ แต่หาก inode เต็ม ก็จะไม่สามารถสร้างไฟล์ใหม่เพิ่มได้อีก วิธีแก้ คือใช้ Double Indirect Block เพื่อรองรับไฟล์ที่มีขนาดใหญ่

Inode Attribute

- Create Time
- Modify Time
- ขนาดของไฟล์
- Owner ID เลขไอดีที่บ่งบอกถึงเจ้าของไฟล์หรือไฟล์ต้นฉบับ
- Group ID
- Version หลังจากที่เรา Unmount ไปแล้ว หากเรา Mount อีกครั้ง สามารถหาไฟล์นั้นเจอได้จาก Version

- Access Control บ่งบอกถึงการทำงาน เช่น Read only, Write only หรือ Both of read and write
- ชื่อไฟล์
- Index Array
- ขนาดของ Array