Alluxio

1. memory-centric storage architecture

Tacyon Master와 worker들이 있음...

1. lineage in storage layer

file set이 spark job을 거쳐

1. storage 단위는 block인듯..

FileSystem.java

interface -> 선언만 해놓은 class

class Factory 의 get() -> LineageFileSytem 이거나 BaseFileSystem…

LineageFileSystem.java

BaseFileSystem을 extend해서 사용

lineage(계보) - RDD lineage...DAG?

아닌가.. alluxio lineage가 있네..

File System & context -> context는 그냥 여러 client들 사이에서 공유되어지는 lineage master client의 내용같은거인듯

pool은 무엇인가.. -> 공동으로 사용하는 storage..

lineage가 추가되면 file이 생성됨.

FileOutStream -> lineage job을 통해 output stream이 생성됨

LineageMasterClient.java

alluxio client가 사용. lineage master와 연결시켜주는 역할

createLineage, deleteLineage 등의 함수가 있음

**BaseFileSystem.java**

createDirectory(), createFile(), delete() - directory나 file을 삭제

exist() - 경로가 alluxio path에 존재하는지 검사

free() - alluxio space에서 온 data 지우지만 UFS에서 온 data는 지우지 않음. metadata는 alluxio space에 남아있음

getStatus(), listStatus() - status=metadata. 경로인 경우 경로의 list를 return

loadMetadata() - UFS의 metadata를 alluxio로 로드. data전송은 없음

mount() - ufs의 subtree를 주어진 alluxio path로 마운트. data나 metadata 전송은 없음. path끼리의 connection만 만드는것. storage에 파일시스템을 입히는거라고했던거같은데

unmount() - ufs subtree를 alluxio path에서 독립시킴. alluxio path는 전에 mount 됐었던 경로와 연결됨. subtree의 contents들은 path에서 alluxio에서 제거되지만 UFS의 subtree는 남아있음.

openFile(), rename()

setAttribute() - path의 attribute를 설정.. ex) TTL이나 pin status…

FileInStream.java

file을 읽어오는 API

읽고 cache issue 처리하고

block의 일부분만 읽혀도 cache를 한다.

closeOrCancelCacheStream()

1. reader 두명이 동시에 같은 block을 cache하려고 할때. 한명이 먼저 cancel함..
2. 위와 똑같은데 서로 다른 BlockClient를 필요로 할때..
3. reader 두명이 동시에 같은 block을 cache하려고 할때. 한명이 먼저 완료함..
4. ..

*Caches the entire block even if only a portion of the block is read.*

*Blocks are not cached unless they are fully read.*

*Blocks are not cached unless they are fully read.*

*wow..*

FileOutStream.java

file에 write하는 API

UFS under file system - 범용 플래시 메모리가 아닌것같다… HDFS라고 보면된다고…?

ufs에 쓰고, ..기타등등등

--------------------------------------------------------------------------------------------------------------------------------------

**FileSystemMaster.java**

URF : unread field

processJournalEntry()

journal - 파일 시스템에서 변경사항을 commit하기 전에 저널에 먼저 적는 그런거인

journal을 저장하기 위해서 shared filesystem이 필요. 모든 master들은 shared filesystem을 읽고 쓸 수 있어야 함. journal을 쓰는 것은 leader master만 가능하지만, 모든 master는 읽고 공유할 수는 있음.

inode - file 이나 directory 저장하는 data structure. metadata를 표현하는 방식 중 한개인가..?!

streamToJournalCheckpoint()

check point - inode가 쓰인 후를 checkpoint로 해 mount data를 적도록..

--------------------------------------------------------------------------------------------------------------------------------------

Thrift - 3rd party library

3rd party란...? 제 3자. 하드웨어나 소프트웨어 제품을 제조하고 있는 메이커, 계열 회사, 기술 제휴를 하고 있는 기업 이외의 기업을 총칭. 다른 회사 제품에 이용되는 소프트웨어나 주변 기기를 개발하는 회사.

3rd party library - 다른 회사에서 만든 라이브러리라는 뜻 같은데..

--------------------------------------------------------------------------------------------------------------------------------------

Alluxio FS 바꾸기…

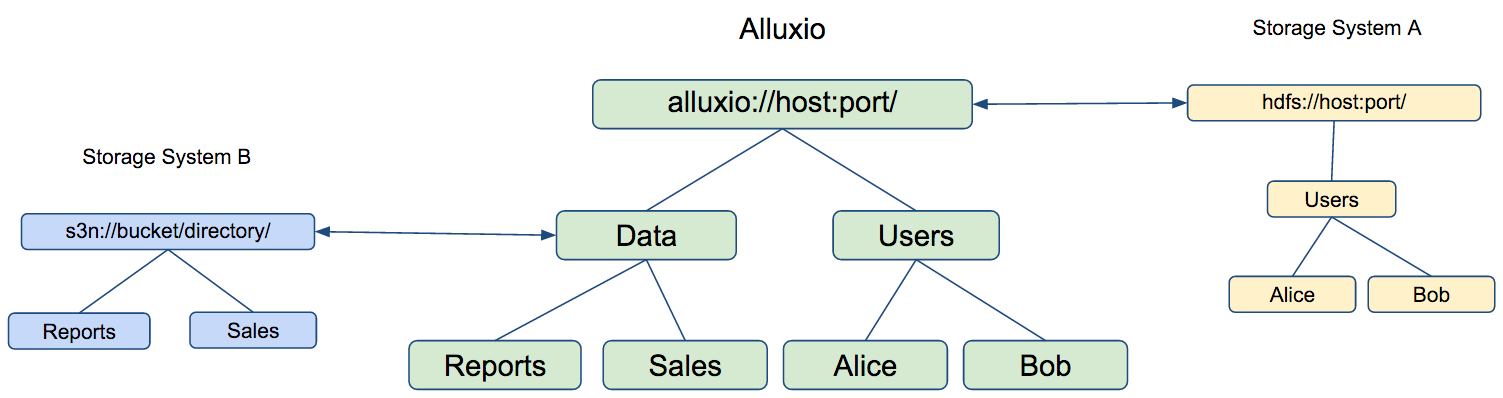
under storage?? - underlying storage 기본기억장치

UFS- unix file system.

HDFS랑 S3A

underfs module

mount - 물리적인 장치 정보를 논리적으로 dir에 인식!



초록색이 underlying file storage같은데..

alluxio storage

1. MEM
2. SSD
3. HDD

alluxio.underfs.address(alluxio namespace가 mount되는 곳) -> primary storage

# 리눅스에서 ssd에 원하는 파일시스템을 마운트하는법

1. mkdir로 디렉토리를 만들고

mkdir /mnt/c

2. vfat이란 파일시스템으로 마운트하기

mount -t vfat /dev/hda1 /mnt/c

<https://blog.naver.com/sihal7/110085919388>

**mount -t glusterfs** *[-o* <options>] <volumeserver>:<volumeid> <mountpoint>

**mount -t glusterfs**

*[-o* <options>] <path/to/volumefile> <mountpoint>

**-t, --types** *vfstype*

The argument following the **-t** is used to indicate the filesystem type. The filesystem types which are currently supported include: *adfs*, *affs*, *autofs*, *cifs*, *coda*, *coherent*, *cramfs*, *debugfs*, *devpts*, *efs*, *ext*, *ext2*, *ext3*, *ext4*, *hfs*, *hfsplus*, *hpfs*, *iso9660*, *jfs*, *minix*, *msdos*, *ncpfs*, *nfs*, *nfs4*, *ntfs*, *proc*, *qnx4*, *ramfs*, *reiserfs*, *romfs*, *squashfs*, *smbfs*, *sysv*, *tmpfs*, *ubifs*, *udf*, *ufs*, *umsdos*, *usbfs*, *vfat*, *xenix*, *xfs*, *xiafs*. Note that coherent, sysv and xenix are equivalent and that *xenix* and *coherent* will be removed at some point in the future - use *sysv* instead. Since kernel version 2.1.21 the types *ext* and *xiafs* do not exist anymore. Earlier, *usbfs* was known as *usbdevfs*. Note, the real list of all supported filesystems depends on your kernel.

dev/hda - 장치 파일의 master device on the first [ATA channel](https://en.wikipedia.org/wiki/ATA_channel)

$ cat /proc/filesystems

$ cat /etc/fstab

!!

1. 파티션 생성 (fdisk 말고 parted사용할 수도 있음)

fdisk /dev/hda

2. 포맷

mkfs -t ext3/dev/hda8

3. 마운트

mkdir /data

mount -t ext3 /dev/hda8 /data

4. fstab에 등록

cat /etc/fstab

<https://blog.naver.com/koomung/90162447858>

먼저 예전의 IDE방식의 하드디스크는

hda, hdb, hdc~ 이렇게 명명을 합니다.

Hard Disk A, Hard Disk B, Hard Disk C~ 이렇게 순서대로 이름이 정해집니다.

SCSI방식의 하드디스크라면

sda, sdb, sdc ~ 이런순으로 명명이 되구요.

Scsi hard Disk A, Scsi hard Disk B, Scsi hard Disk C~ 이런 약자이구요.

lsblk

lsblk --fs

sudo parted -l

df -h -x tmpfs -x devtmpfs

mount point -> /home/arcs/data

/dev/nvme1n1p1: UUID="59c6dad0-e26c-49fe-a988-3bc2a6014b65" UUID\_SUB="0ae81051-7117-494f-b17b-461cc2c97481" TYPE="btrfs"

<https://www.howtoforge.com/how-to-convert-an-ext3-ext4-root-file-system-to-btrfs-on-ubuntu-12.10>

<https://btrfs.wiki.kernel.org/index.php/Conversion_from_Ext3>

<https://unix.stackexchange.com/questions/53313/how-to-show-the-filesystem-type-via-the-terminal>

--------------------------------------------------------------------------------------------------------------------------------------

FileSystemMasterClient.java

checkConsisytency() -> Alluxio metadata의 consistency를 check.

createDirectory(), createFile()  
completeFile(), delete(), free() -> file 삭제, free, complete

getStatus()

getNewBlockIdForFile()

listStatus()

loadMetadata()

mount() -> UFS path를 alluxio path로 마운트

getMountTable() -> list all mount points

rename()

setAttribute()

scheduleAsyncPersist()

unmount()

--------------------------------------------------------------------------------------------------------------------------------------

RereyHandlingFileSystemMasterClient.java

//Service?

file system master와 interact하기 위한 thrift client wrapper로, alluxio client가 사용함.

thrift clients가 thread safe하지 않기 때문에 thread safety를 제공하고, retry를 제공하기 위한 목적

위의 FileSystemMasterClient를 implement해서 override.

--------------------------------------------------------------------------------------------------------------------------------------

DefaultFileSystem.java

Master that handles all file system metadata management.

locking path가 무엇인가..?? journaling. logging. ufs의 absent path를 cache.

TTL(time to live) bucket...

inode tree의 path를 locking할때 다른 concurrent operation이 inode를 수정할 수 있다.

try(..; ..){

….

}

->?

isFullyInAlluxio() -> inode file block들이 alluxio에 100%있는지

isFullyInMemory() -> inode file block들이 memory에 100%있는지

isInTopStorageTier() ->block이 어떤 워커노드의 top storage level에 있으면 true.

rename방법

1. source inode의 이름을 destination의 이름으로 바꿈
2. source inode를 destination parent로 삽입
3. 필요하다면 UFS operation을 실행

source file이 지속되면 UFS에서 source file을 rename한다...가뭔소리

1. source inode를 source parent에서 삭제
2. 마지막 수정 시간을 source와 destination의 parent inode들에 저장.

mount()

LockedInodePath inodePath = mInodeTree.lockInodePath(alluxioPath, InodeTree.LockMode.WRITE); -> 약간 동시에 수정하면 안되니까 lock을 거는 것일듯

////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

**alluxio client node**

BaseFileSystem.java

FileSystemMasterClient.java

RetryHandlingFileSystemMasterClient.java

FileSystem.java

file\_system\_master.thrift// 여기에 thrift문법으로 자동생성할 함수 코드에 대한 선언 작성

**alluxio master node**

DefaultFileSystemMaster.java

FileSystemMasterClientService.java // 스리프트가 자동으로 생성

DefaultFileSystemMaster.java // 여기에 실제로 구현

ufs -> under file system. alluxio에서 사용하는 fs를 말하는 듯

//////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////////

thrift 돌리는 방법

bin/alluxio thriftGen

mvn install -DskipTests -Dmaven.javadoc.skip -Dcheckstyle.skip -Dlicense.skip -Dfindbugs.skip