OS project 5

2016310936 우승민

이번 과제의 목표는 기존의 xv6 **file system** 에서 **12** 개의 **direct blocks** 와 **1** 개의 **singly-indirect block** 으로 구성되어 있던 **inode** 를 **11** 개의 **direct blocks** 와 **1** 개의 **singly-indirect block**, **1** 개의 **doubly-indirect block** 으로 변경하여 xv6 의 표현가능한 size 의 크기를 키우는 것입니다.

우선적으로 file 의 maximum size 를 키우기 위해서는 **data blocks** 의 수를 늘려야 하기 때문에 **param.h** 에 define 되어있는 **FSSIZE** 를 **1000 -> 20000** 으로 늘렸습니다.

```
#define NPROC        64  // maximum number of processes
#define KSTACKSIZE 4096  // size of per-process kernel stack
#define NCPU          8  // maximum number of CPUs
#define NOFILE       16  // open files per process
#define NFILE       100  // open files per system
#define NINODE       50  // maximum number of active i-nodes
#define NDEV         10  // maximum major device number
#define ROOTDEV       1  // device number of file system root disk
#define MAXARG       32  // max exec arguments
#define MAXOPBLOCKS  10  // max # of blocks any FS op writes
#define LOGSIZE      (MAXOPBLOCKS*3)  // max data blocks in on-disk log
#define NBUF         (MAXOPBLOCKS*3)  // size of disk block cache
#define FSSIZE       20000  // size of file system in blocks

~
"param.h" 14L, 761C written
```

Xv6 를 실행하면 아래와 같이 변경된 것을 확인할 수 있습니다.

```
xv6...
8010a460 2c
cpu0: starting 0
sb: size 20000 nblocks 19937 ninodes 200 nlog 30 logstart 2 inodestart 32 bmap start 58
```

다음으로 **fs.h** 에 define 되어 있던 **inode** 에서 1 개의 **direct block** 을 **doubly-indirect block** 으로 바꾸어 주었습니다.

```
#define NDIRECT 11
#define NINDIRECT (BSIZE / sizeof(uint))
#define DINDIRECT (BSIZE / sizeof(uint) * BSIZE / sizeof(uint))
#define MAXFILE (NDIRECT + NINDIRECT + DINDIRECT)

// On-disk inode structure
struct dinode {
  short type;           // File type
  short major;          // Major device number (T_DEV only)
  short minor;          // Minor device number (T_DEV only)
  short nlink;          // Number of links to inode in file system
  uint size;            // Size of file (bytes)
  uint addrs[NDIRECT+2];   // Data block addresses
};
"fs.h" 58L, 1809C written
```

여기서 **DINDIRECT** 가 제가 추가해준 **doubly-indirect block** 이고 크기는 기존 **NINDIRECT** 가 이중으로 구성되어 있는 것이기 때문에 **NINDIRECT 크기의 제곱**으로 해주었습니다.

또한 **dinode structure** 에서 기존에 **NDIRECT+1** 의 크기로 선언되어 있던 **addrs** 에 1 을 추가로 더해주었습니다. 그리고 이것은 **file.h** 의 **inode structure** 에서도 맞춰주었습니다.

```
struct inode {
  uint dev;           // Device number
  uint inum;          // Inode number
  int ref;            // Reference count
  struct sleeplock lock; // protects everything below here
  int valid;          // inode has been read from disk?

  short type;         // copy of disk inode
  short major;
  short minor;
  short nlink;
  uint size;
  uint addrs[NDIRECT+2];
};
"file.h" 37L, 802C written
```

마지막으로 **fs.c** 에 파일의 내용을 실제 disk 에 저장된 data block 과 mapping 해주는 **bmap** 함수를 수정해 주었습니다. **bmap** 함수는 기존에 mapping 되어있는 block 이 있으면 불러오고 없으면 새로 할당해줍니다.

```
static uint
bmap(struct inode *ip, uint bn)
{
  uint addr, *a;
  struct buf *bp;

  if(bn < NDIRECT){
    if((addr = ip->addrs[bn]) == 0)
      ip->addrs[bn] = addr = balloc(ip->dev);
    return addr;
  }
  bn -= NDIRECT;

  if(bn < NINDIRECT){
    // Load indirect block, allocating if necessary.
    if((addr = ip->addrs[NDIRECT]) == 0)
      ip->addrs[NDIRECT] = addr = balloc(ip->dev);
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;
    if((addr = a[bn]) == 0){
      a[bn] = addr = balloc(ip->dev);
      log_write(bp);
    }
    brelse(bp);
    return addr;
  }
```

위 사진이 원래 사용되던 **bmap** 함수이고 저는 아래 코드를 추가해주었습니다.

```
  bn -= NINDIRECT;

  if(bn < DINDIRECT){
    if((addr = ip->addrs[NDIRECT+1]) == 0)
      ip->addrs[NDIRECT+1] = addr = balloc(ip->dev);
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;
    if((addr = a[bn/NINDIRECT]) == 0){
      a[bn/NINDIRECT] = addr = balloc(ip->dev);
      log_write(bp);
    }
    brelse(bp);
    bp = bread(ip->dev, addr);
    a = (uint*)bp->data;
    if((addr = a[bn%NINDIRECT]) == 0){
      a[bn%NINDIRECT] = addr = balloc(ip->dev);
      log_write(bp);
    }
    brelse(bp);
    return addr;
  }

  panic("bmap: out of range");
}
```

singly-indirect block 까지 할당이 끝났으면 다시 **bn** 을 0 에서부터 시작하게 해주고, **doubly-indirect block** 크기만큼 즉 **128^2** 만큼 새로 할당하게 해주었습니다.

Singly-indirect block 할당해주는 코드와 유사하게 만들어 주었고, 다만 각각의 block 에서 **NIDIRECT** 크기 만큼을 더 할당하게 해주었습니다.

그 후 xv6 를 실행하여 test 코드를 진행하면 block 이 잘 할당된 것을 확인할 수 있습니다.