Iftekhar Hakim Knowsee
ID-1705045

(1) Number of block = $\frac{120\,GB}{4\,KB} = \frac{120 \times 2^{30}\ byte}{4 \times 2^{10}\ byte}$

$= 30 \times 2^{20}$

For FAT 32, each entry in table is 4 byte.

So, space needed in memory

$= 4 \times (30 \times 2^{20})$ byte

$= 120 \times 2^{20}$ byte

$= 120MB$

(2) If we use i-node instead of linked list allocation, reading will be faster. In linked list allocation, random-access is very slow, because of reading previous disk blocks before the required disk block. But, in i-node, we will bring it to memory. Chain will still be followed but disk accessing for previous blocks will not be needed. Chain will be followed only to get the address of target block.

If we use i-node, data ~~s~~ appears in blocks will be in power of 2, ~~causing~~ the less-efficiency problem of peculiar size in linked list allocation

(3). As long as we need file, we need a whole system to manage it. Files being very important for persistent and long term storage, file system is necessary to maintain it. A file system will keep track of how files are allocated, ~~how~~ how they are protected, ~~fit~~ etc. File system will bring out the meaning of byte appear in disk. There are many variants of managing files in disk. A file system can only specify which are in use. Otherwise, ~~to~~ bytes in disk will ~~sym~~ simply appear as garbages.

④ ~~Hard~~ Symbolic link:-

Suppose, a file in directory Y needs to be linked in directory X.

Symbolic link ~~with~~ system will create a file with type LINK in directory X. In this file, pathname to ~~file~~ target that file in directory Y will be written. ~~Directory~~. There will be directory entry in X for that LINK-type file.

Whenever user wants to access that shared file in X, system understands that the file is in symbolic link. It grabs the pathname from the LINK-file and parse it. It then goes for some disk accesses to get the actual address of ~~the~~ targetted shared file.