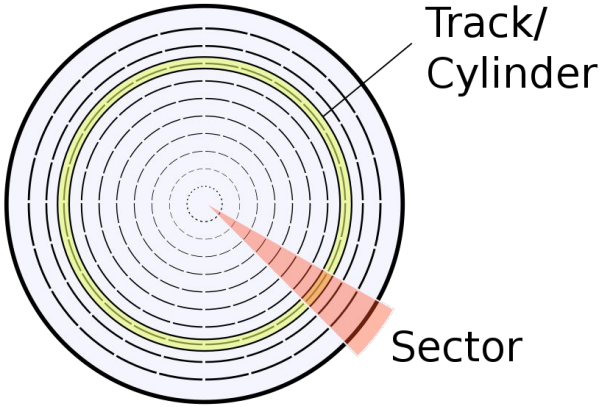
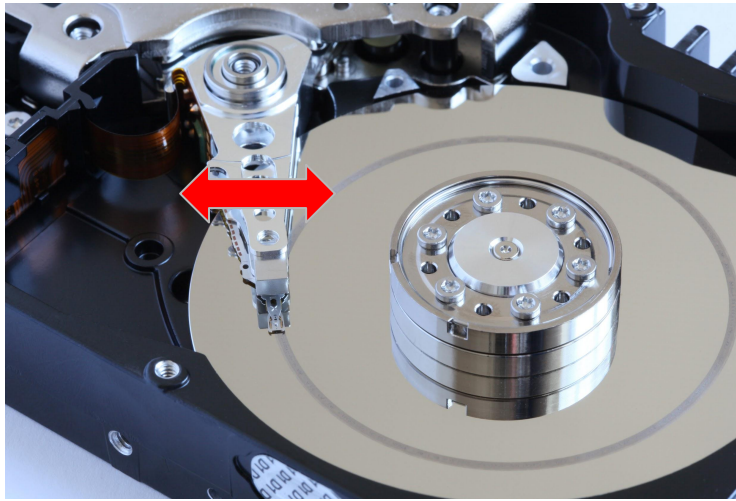
Disk Block Allocation

[Disk] Data Block Allocation

- Data on device are stored in **blocks** or **clusters of blocks**
  - HD: common block size 512B (but also larger blocks upto 4K)
  - SSD: page size 2K - 16K





File Allocation Strategies

- Problem to address: files may grow/shrink
- Static Allocation: pre-allocate a fixed number of blocks (bigger than the requested file size to allow room for growth)
- Dynamic Allocation
  - Contiguous
  - Chained/Linked
  - Indexed
- Previous concepts in *memory allocation* apply to *disk block* allocation

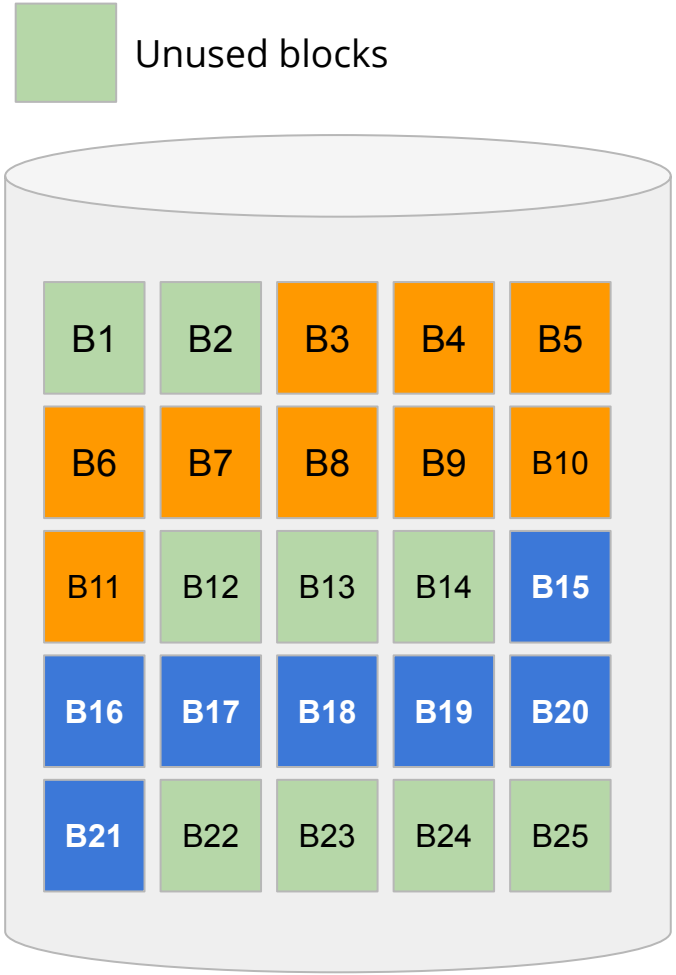
Dynamic Allocation: Contiguous

- Assign a set of contiguous blocks to a file *at the time of creation*
- The directory entry contains: the **starting block** and **number of blocks**
- Block allocation/deallocation as the file grows or shrinks?
- Issue: disk (external) fragmentation
- Effect on disc arm motion?

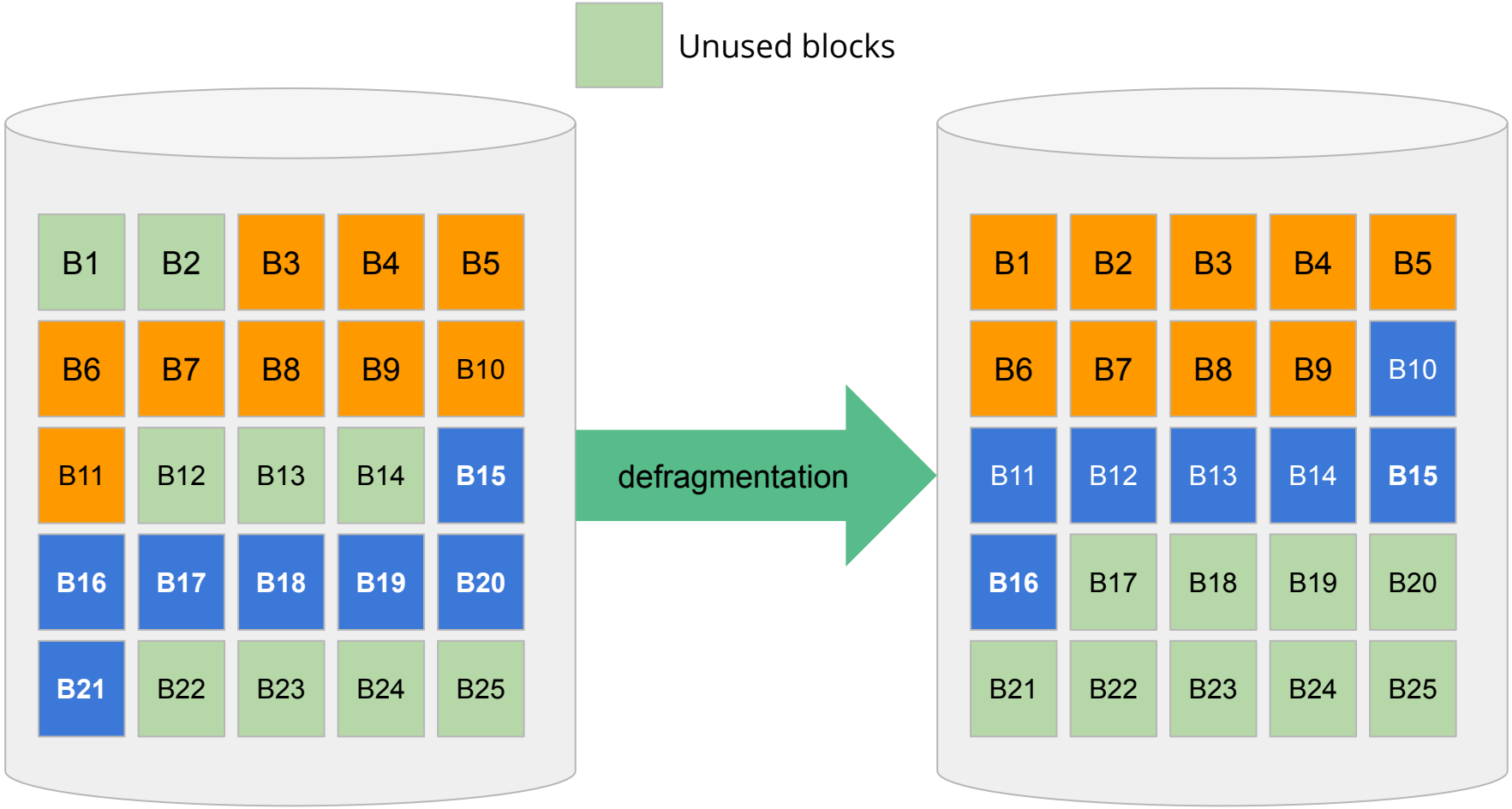
Contiguous Allocation

File	Start Block	Number of Blocks
one.txt	3	9
two.pdf	15	7

How to allocate a new file that requires 5 contiguous blocks?



Contiguous Allocation: Defragmentation



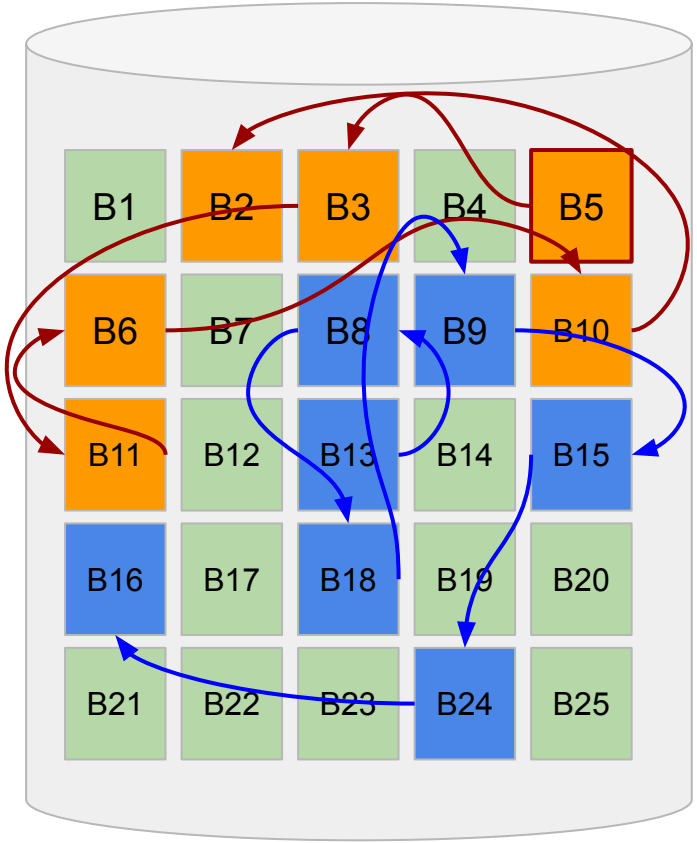
Dynamic Allocation: Linked/Chained

- Reserve a few bytes in each data block for a pointer to the next block
- The directory entry holds the starting block and the number of blocks (or the last block)
  - Head and tail of a linked list
- No external fragmentation
- **Random access is impossible**
  - Accessing the Nth block requires reading the first (N-1) blocks
- Wasted space in each block for the “next/chain” pointer

Linked/Chained Allocation

File	Start Block	Last Blocks
one.txt	5	2
two.pdf	13	16

One.txt: B5, B3, B11, B6, B10, B2
Two.pdf: B13, B8, B18, B9, B15, B24, B16



Effect on disk arm motion?

Improved Linked/Chained Allocation

- Extract all the “next” pointers from the data disk, place them in a designated disk blocks
- DOS FAT: File Allocation Table
  - FAT for the entire file system is stored in (**small number of**) contiguous disk blocks
  - FAT16: 16-bits (2 bytes) per entry => ***1024 entries can fit into 4 blocks of 512 bytes***
- Random Access performance is improved
  - Reading the Nth block requires only linear traversal (of N-1 links) **within the FAT** (not the actual data disks)
  - Fewer disk I/O compared to pure chained/linked allocation

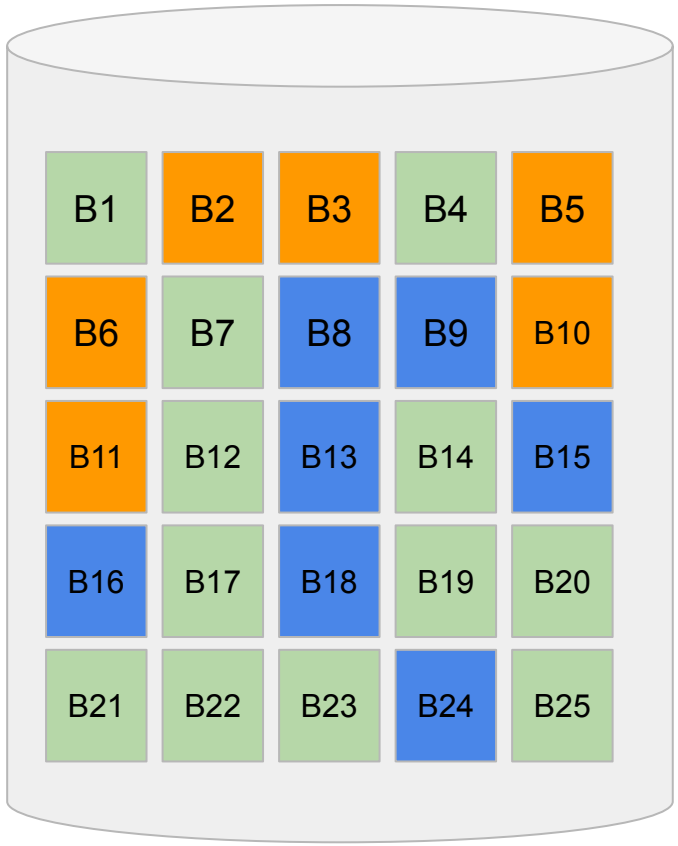
Improved Linked/Chained: MS-DOS FAT

Directory Entries

File	Start	Last
one.txt	5	2
two.pdf	13	16

File Allocation Table (FAT12, FAT16, FAT32)

1	0
2	-1
3	11
4	0
5	3
6	10
7	0
8	
9	
10	2
11	6
12	0
13	
14	0
15	
16	
17	0
18	
19	0
24	



FAT must be saved on the disk

One.txt: B5, B3, B11, B6, B10, B2
Two.pdf: B13, B8, B18, B9, B15, B24, B16

Indexed Allocation

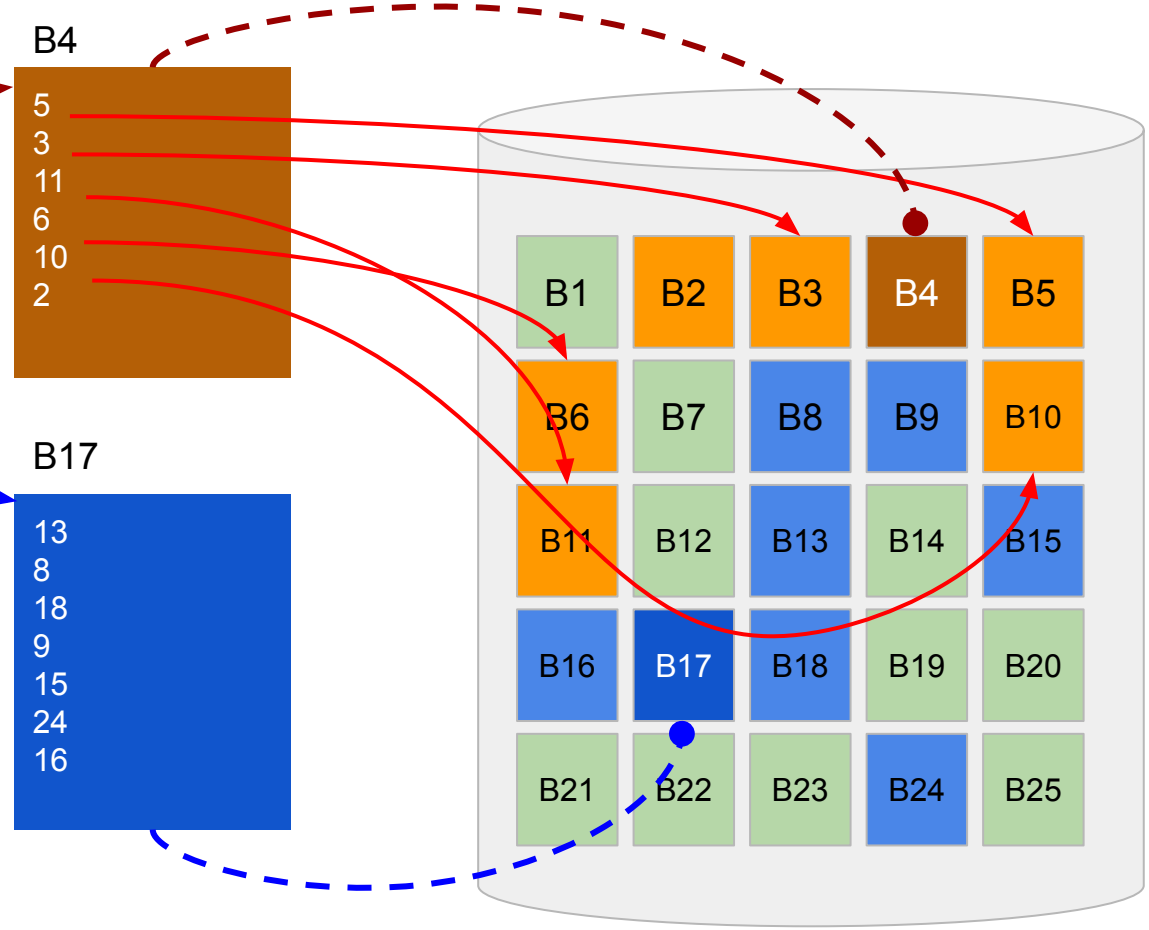
- MS-DOS FAT can be a **bottleneck** of file access; all updates to **any files** in the FS must update **one global copy** of FAT
- Solution: each file should hold its own **index block(s)**
  - The index block records the data block addresses (pointers) used by the file contents
- Directory entry = pair <file name, address of index block>
- Random (non-sequential) access is possible

Many filesystems today use some variant of index blocks.

Indexed Allocation (Recall "page table" in VM)

Directory Entries

File	Index Block
one.txt	4
two.pdf	17



One.txt: *B5, B3, B11, B6, B10, B2*
Two.pdf: *B13, B8, B18, B9, B15, B24, B16*

Indexed Allocation: Limitations

- The maximum file size is limited by the number of pointers that can fit into one index block
  - 512-byte blocks (2^9)
  - 4-byte pointers (32-bit address)
    - 512/4 = 128 pointers per block (2^7)
  - Total disk capacity $2^{32} \times 2^9 = 2^{41}$ = 2 Terabytes
  - Max file size $2^7 \times 2^9 = 2^{16}$ = 64K bytes
- To store larger files
  - Link several index blocks together
  - Multi-level index
  - Combined: use both direct index and multilevel indices (Unix File Systems)

Which Allocation Algorithm?

Block Allocation	File Growth	Fragmentation	Direct/Random Access
Contiguous	May require relocation	Yes	**O**(1)
Chained/Link	Easy	No	**O**(N)
Indexed	Limited by **size of index block**	No	**O**(1)
Multi-level Indexed	Limited by **depth of index block hierarchy**	No	**O**(log depth) depth is constant for UFS