CSE 451: Operating Systems

Lab Section: Week 9

Today

- Project 4
- More file system issues

Next week

- No office hours on Tuesday
  - I'm out of town
  - I will be on email
  - makeup office hours after section today (10:30 to 11:30 in the lab)
- Next week's section is a review
  - bring questions!
  - or email me questions/topics in advance

Project 4

- Due next Wednesday, March 9 at 11:59pm
  - in three weeks
- Questions?

Project 4

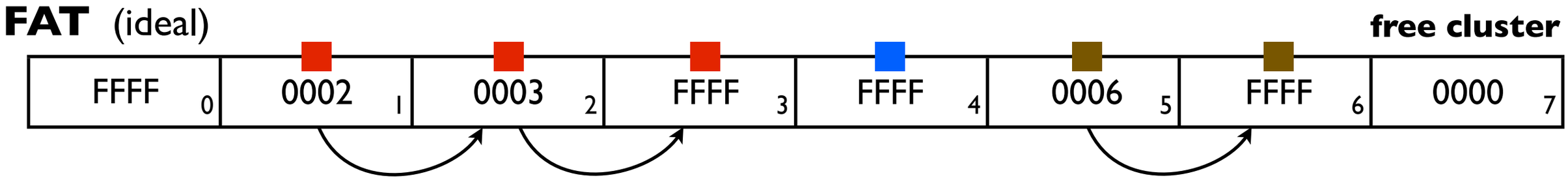
- Gotcha:
  - each FCB has a pointer to its dirent (FCB.DirentOffsetWithinDirectory)
  - you need to update this after sorting the dirents
  - FatDefragDirectory has an example of how to do this

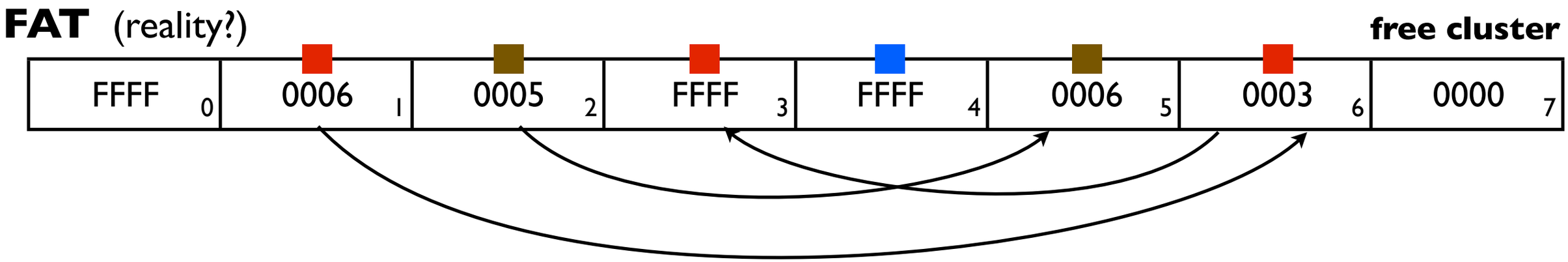
- Gotcha (extra credit):
  - dealing with directories longer than 256 KB (0x40000 KB)
  - complication is the cache manager
    - ... deals with 256 KB at a time (called "views")
    - ... see Chapter 11 of Windows Internals

Today

- ~~Project 4~~
- More file system issues

Fragmentation





how can we optimize data block allocation to minimize fragmentation?

Data block allocation

Say we have a sequence of writes to a file

NtOpenFile
NtWriteFile
NtWriteFile
...
...
NtWriteFile
NtCloseFile

When should we allocate data blocks?

Greedily?

- i.e., one block at a time as we need them
- problems with greedy allocation?

Data block allocation

Say we have a sequence of writes to a file

NtOpenFile
NtWriteFile
NtWriteFile
...
...
NtWriteFile
NtCloseFile

When should we allocate data blocks?

Greedily, with a pre-allocation cache

- idea: reserve a set of contiguous blocks to be allocated *next*
- possible implementations:
  - ... single cache for the system
  - ... cache per cpu
  - ... cache per open file

Data block allocation

Say we have a sequence of writes to a file

NtOpenFile
NtWriteFile
NtWriteFile
...
...
NtWriteFile
NtCloseFile

When should we allocate data blocks?

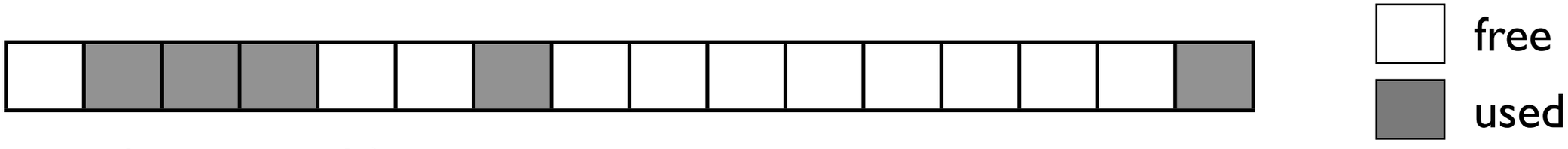
Lazily?

- a.k.a. *delayed allocation*
- wait before allocating blocks (until NtCloseFile, until the cache fills up, ...)
- **advantage:** we can allocate blocks for the entire file, all at once
- gotcha: NtWriteFile should fail if the disk is full
  (need to update "used block count" greedily, even if blocks allocated lazily)

Data block allocation

How do we find a run of N contiguous blocks?

free block bitmap



linear search?

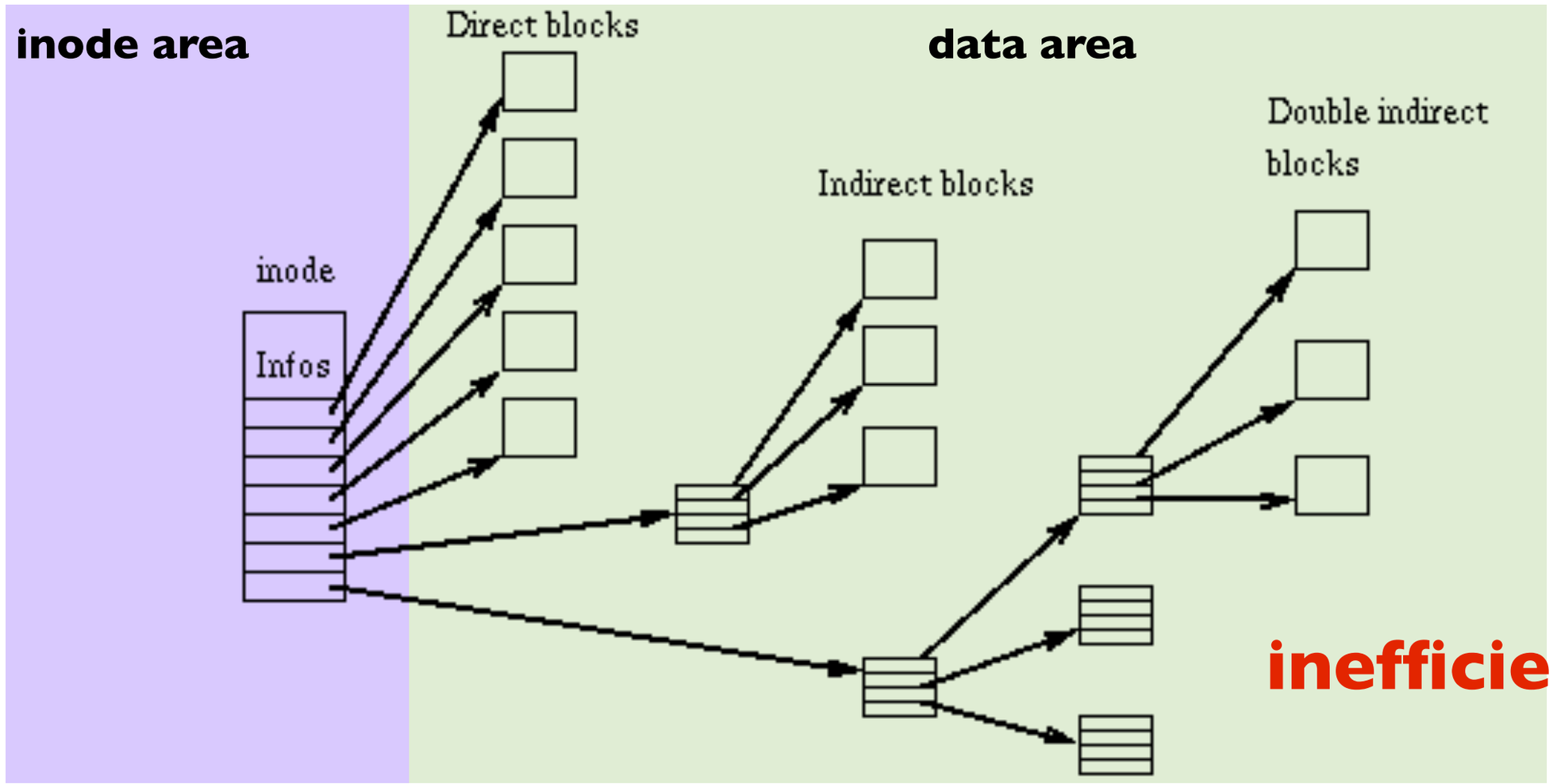
- easy to check for some runs of 8, 16, 32, or 64 (why?)
- but still slow

buddy bitmap #1

buddy bitmap #2

Inodes

Unix FS (many other file systems roughly similar)



inefficient?

image courtesy of Wikipedia

More efficient large inodes

- new idea: *extents*
  - *extent:* a contiguous region of the file represented by a contiguous range of physical blocks on disk
- btree of *extents*
  - replacement for indirect pointers
  - used by ext4 and xfs