Reasoning about OS Performance

COS 316: Principles of Computer System Design
Lecture 5

Nicolaas Kaashoek

Modeling & Measuring Performance

Why does performance matter?

- System performance translates to application performance
- Goal: Provide hardware performance

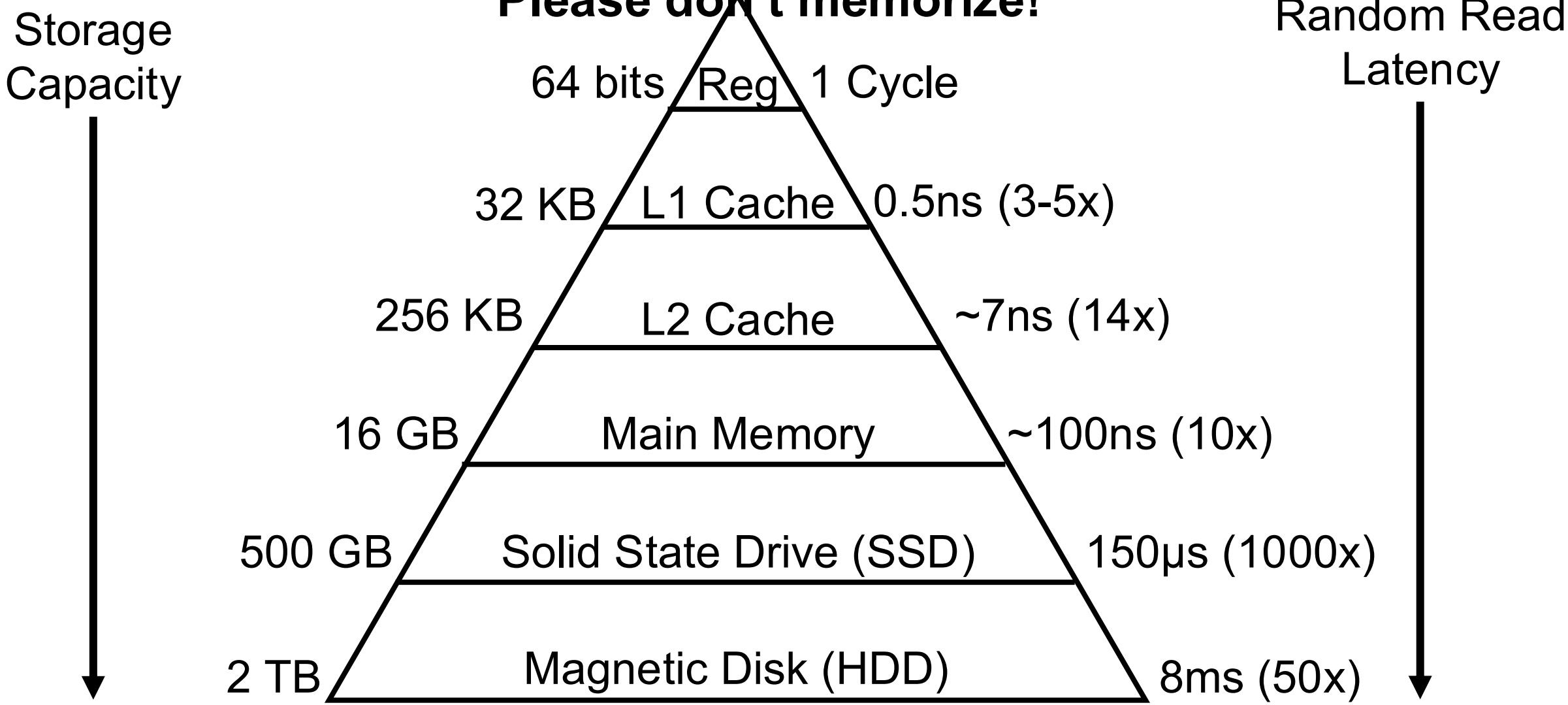
How to think about performance?

- Construct a mental model of system design (this lecture)
- Hypothesize design performance (this lecture)
- Measure Performance using a *workload*
- Revise design and repeat

Hardwa Of

Relative Differences are what matter!

Please don't memorize!



Hardware: Magnetic Disks

3 Parts for Performance:

- Seek Speed (Arm to track)
- Rotation Speed (Block under arm)
- Throughput (bits to blocks)

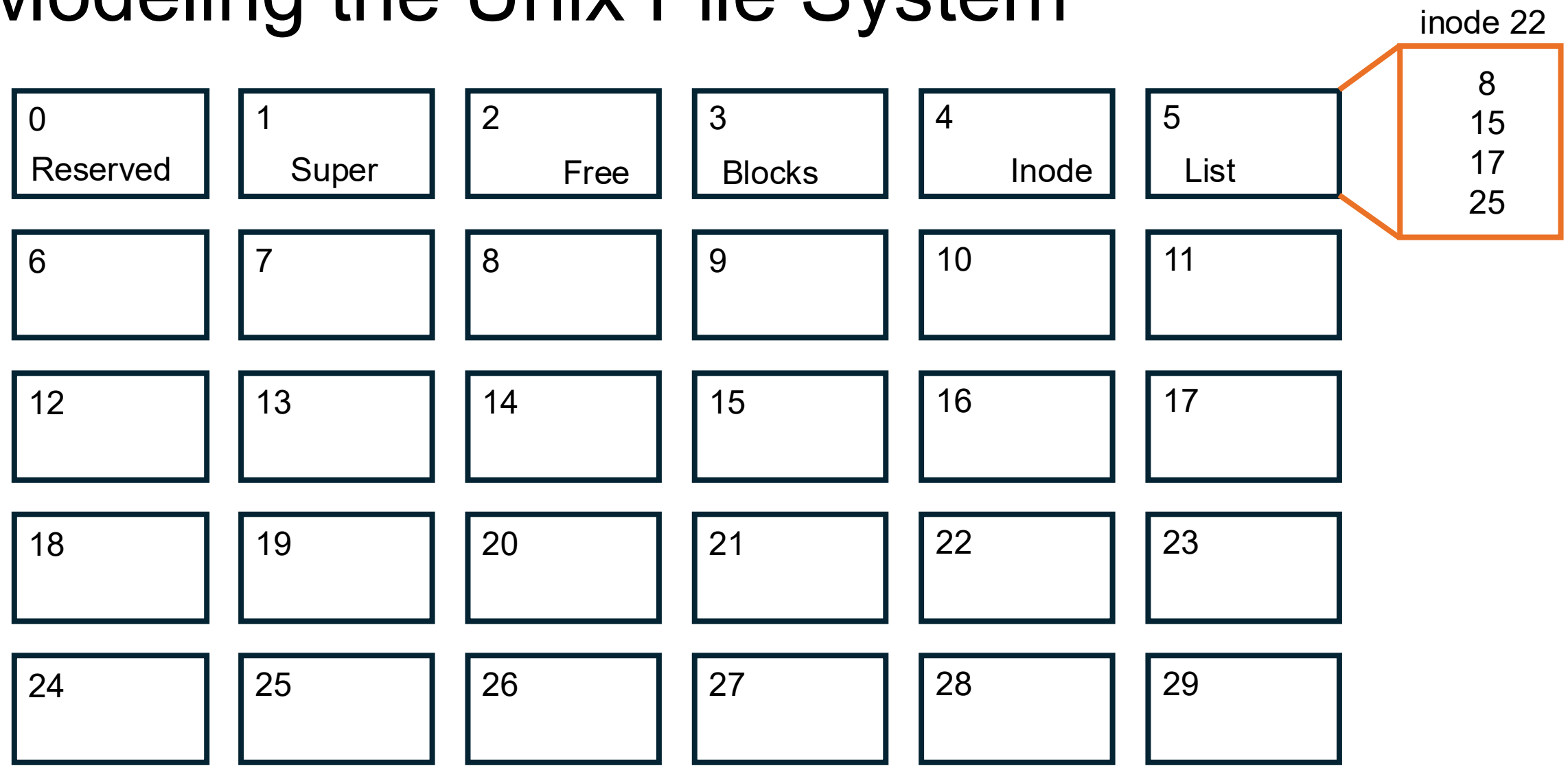
Modern Disk: 4ms seek,

7200 rpm, ~300 MB/s

Throughput vs. Latency

Random vs. Sequential

Modeling the Unix File System



Unix File System Performance - Create

`echo > /foo.` What happens?

1) Allocate an inode for the file
2) Update the newly created inode
3) Update the directory entry for "/"

This is a simplification.

Demo!

Unix File System Performance - Write

`echo a > /foo.` What happens?

1) Find block from free block list
2) Zero the allocated block
3) Write to block
4) Update inode for file

This is a simplification.

Demo!

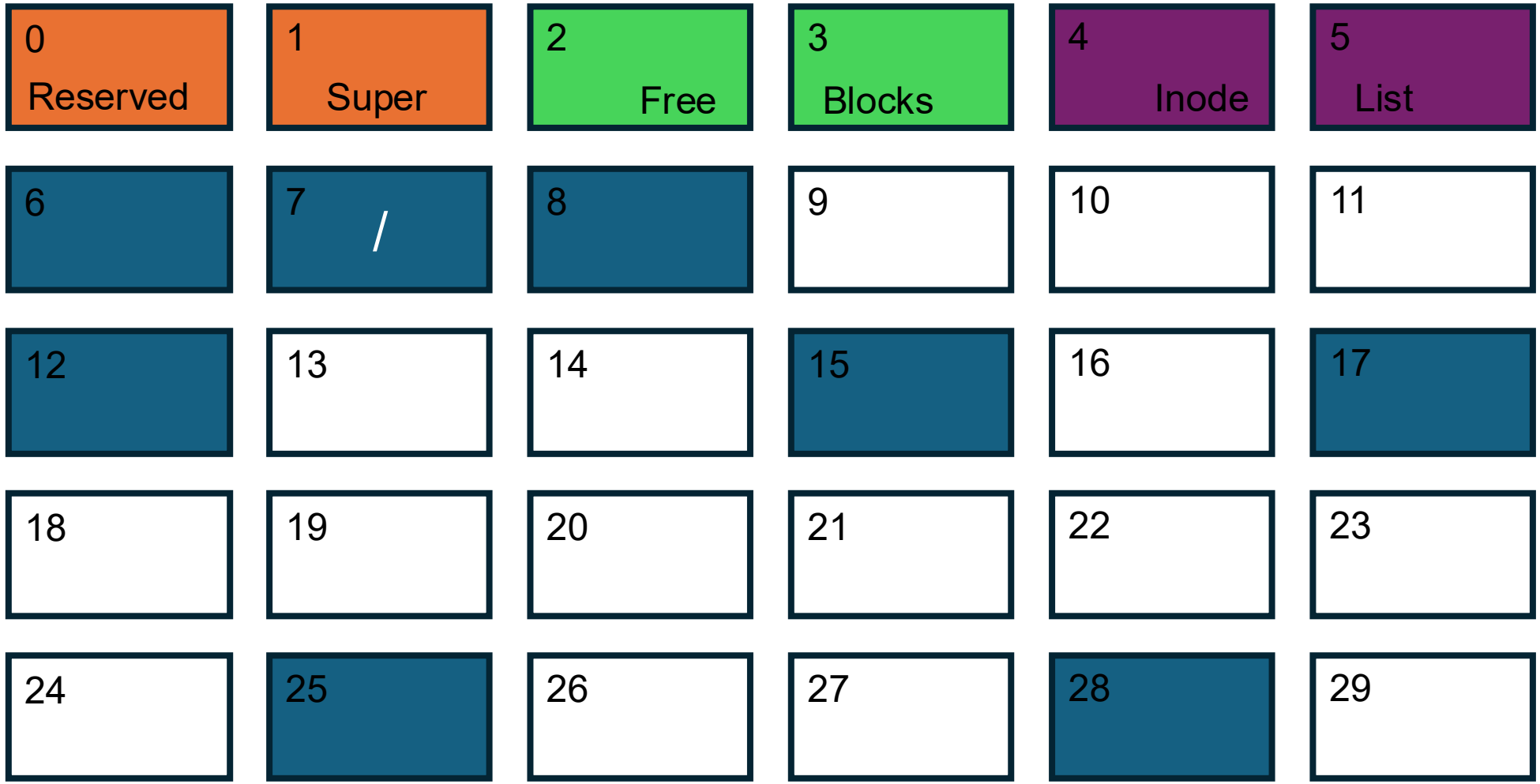
Unix File System Performance – Delete

What does it take to delete a file from the system?

- File can’t be read from or written to anymore
- Space can be reclaimed
- Anything else?

Work through the steps with your neighbors

Unix File System Performance - Delete



Log-Structured File System - Intro

Rosenblum & Ousterhout, 1992.

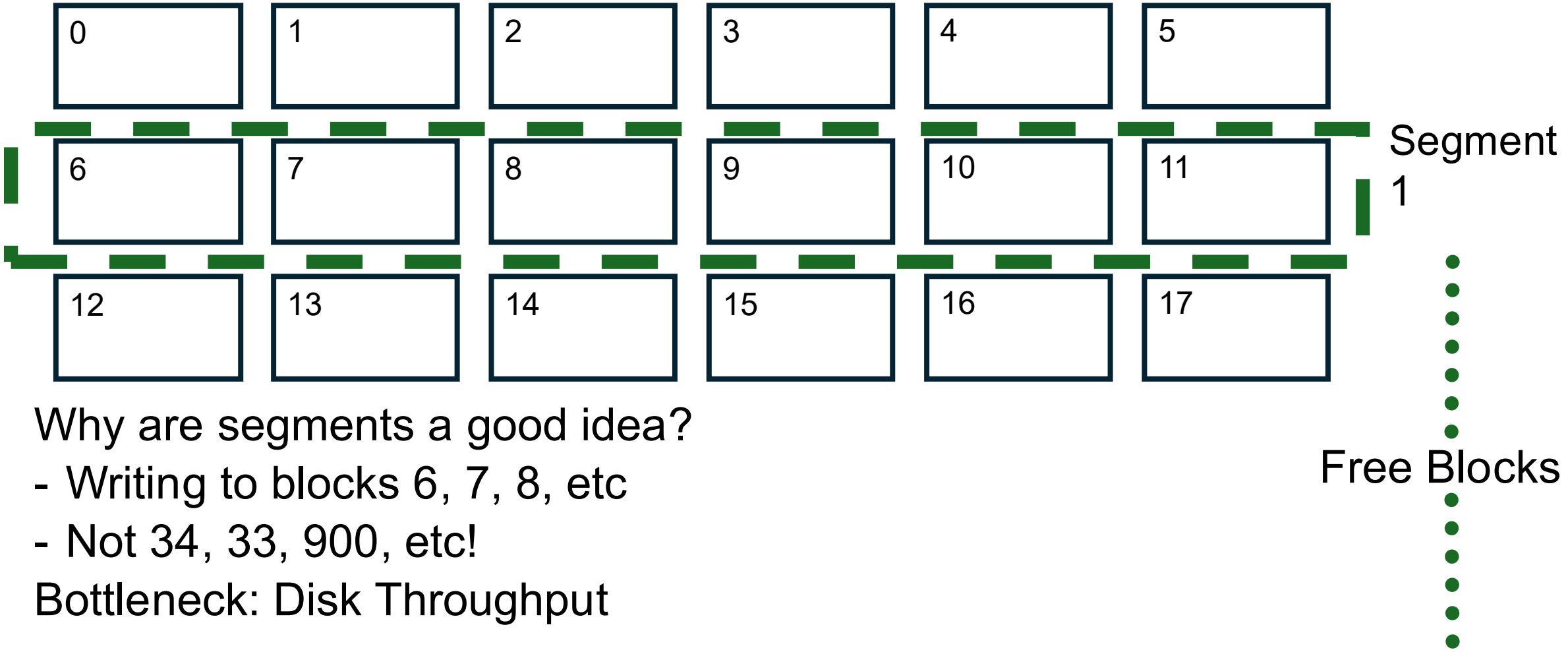
Motivation: turn random writes into sequential ones

The Rules:

- Replace file/inode layer with a log
- Log is made of segments (contiguous blocks)
- Log is *append-only,* all writes to head of log
- Reads go to in-memory cache

This lecture: overview. Read the paper if you're curious for more!

Log-Structured File System – The Log



Why are segments a good idea?

- Writing to blocks 6, 7, 8, etc
- Not 34, 33, 900, etc!

Bottleneck: Disk Throughput

LFS Performance – Create 1

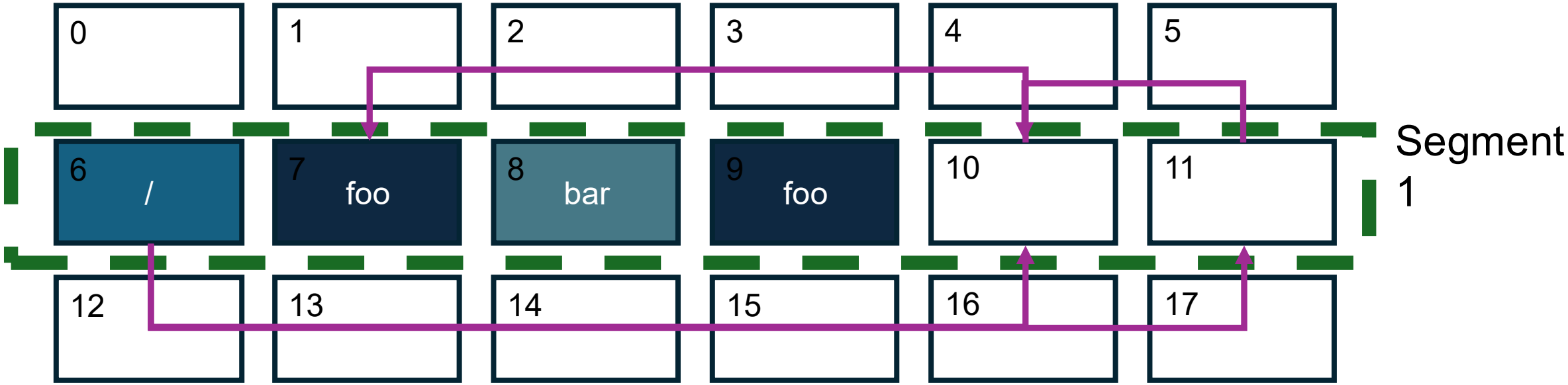
`echo > /baz`. What happens?

- Just need to add to current segment

What if segment is full? Open a new one.

Problem: How do we make sense of blocks?

Log-Structured File System – Lookup 1



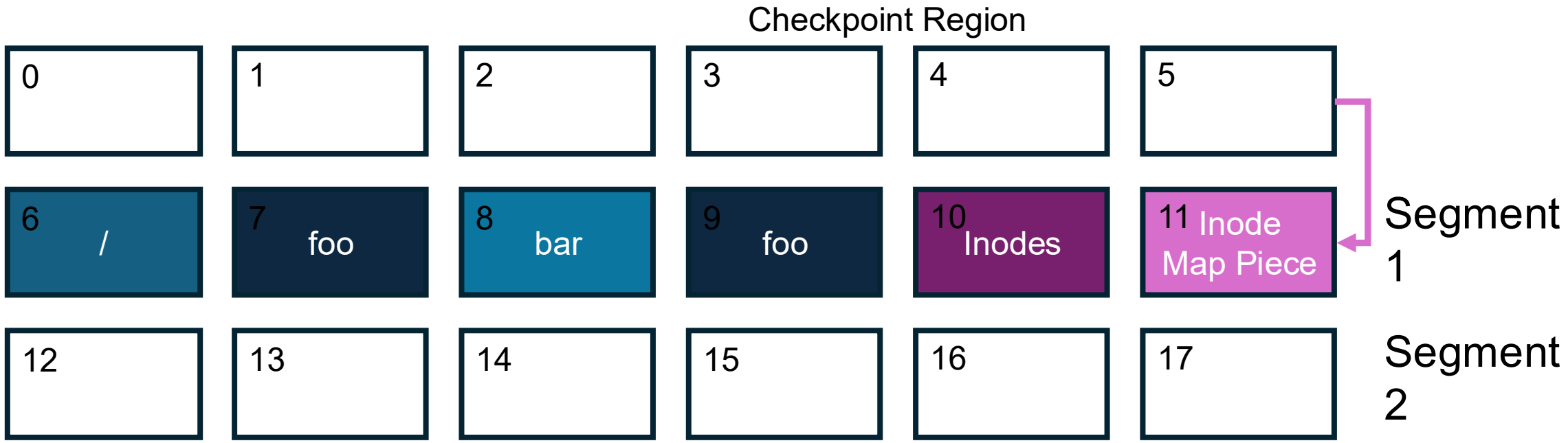
Remember inodes? Let's reuse those.

LFS adds one additional indirection: inode map

- Map tells us where most recent copy of inode is

Map stored in pieces, in segments. Why?

Log-Structured File System – Lookup 2



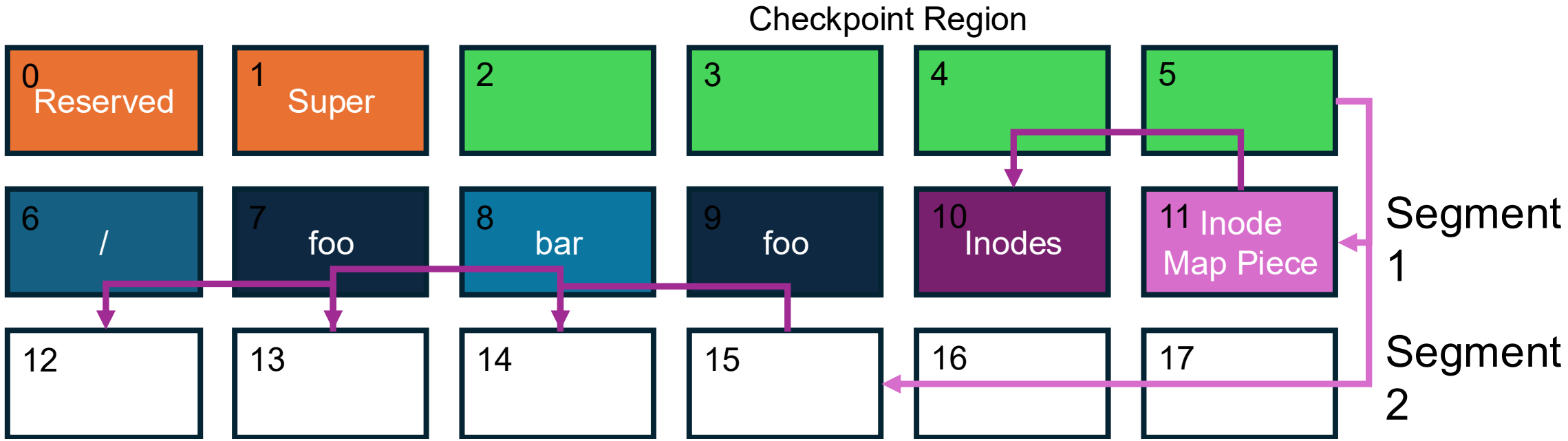
We need some fixed data to tell us about disk

- LFS adds a *Checkpoint Region*
- Region tells us where the map is

To avoid random writes, buffer in memory

Only checkpoint periodically for crash recovery

LFS Performance – Create 2

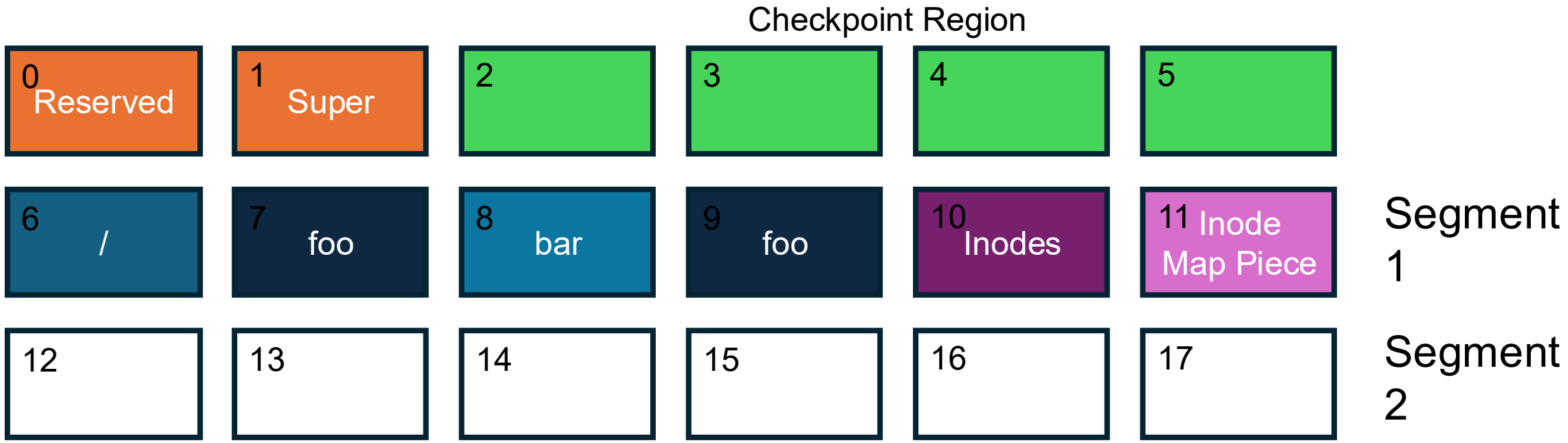


Complete our example: **`echo > /baz`**

- Make a new inode for baz
- Update / to point to that inode
- Update /'s inode to point to newest version
- Write new inode map

Wrote to: 12, 13, 14, 15! **How many ms?**

LFS Performance – Write to File



`echo a > /bar`

- Get a new block: in the log!
- Update Bar's inode
- Write new piece of the map

Wrote to: 12, 13, 14! **How many ms?**

LFS – Garbage Collection

Append-only structures only grow, but not infinite space

When can we reclaim space? How can we reuse the space that has been reclaimed?

These are important questions, but we don’t have time.

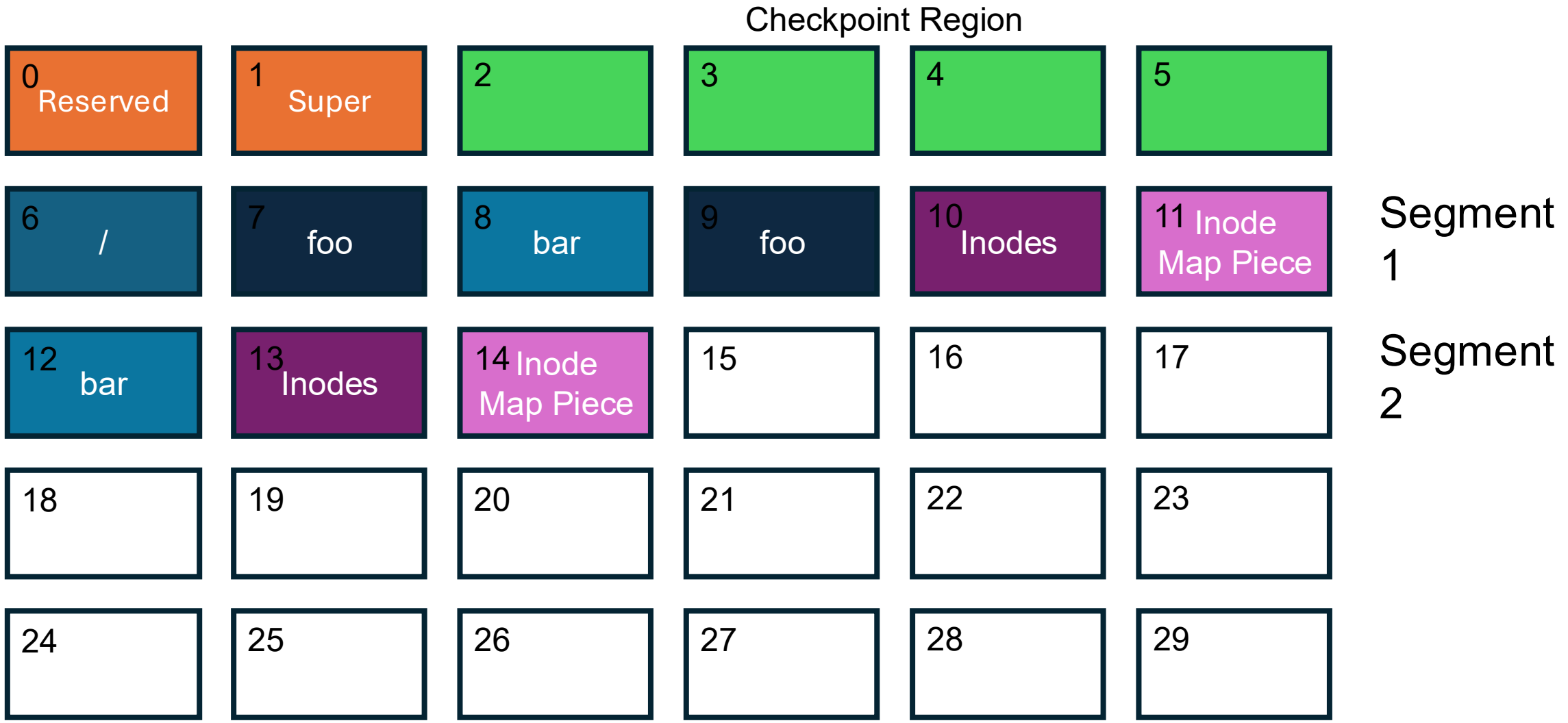
Insight: Copy live data, reclaim dead segments, treat the log as a linked list

LFS Performance – Delete File

Same idea as before. Work with your neighbors to figure out how to delete a file in a log-structured file system!

Add a number. How many writes to disk, are they random or sequential, and how long does it take in total?

LFS Performance – Write to File



LFS vs. Unix File System Tradeoffs

What does a LFS do better than the Unix file system

- Create?
- Read?
- Write?
- Delete?

What does the Unix file system do better than an LFS?

Log-Structured File System – Layering

LFS shows why layering is such a powerful concept.

What did we change about the Unix file system? Only the inode/file layer. The rest stayed the same.

- The same absolute paths still work, and to the application it looks the same
- But we'll get very different performance

*Layers create **modularity***

Tradeoffs in Systems

Tradeoffs are a fundamental concept in systems

Rare that system is *strictly* better than competitor

Instead, we build systems that are better in the situations *that matter*

Critical questions when it comes to evaluating systems!