Problem: FFS performance not good
- Seeking due to many random writes
- Inodes, data spread across cylinder group

Trend: caches getting bigger
- Computer memories getting bigger
- Disk caches getting bigger
- Big gap between seq and random performance

How does this trend affect reads and writes?
- Reads absorbed by cache
- many writes are delayed and batched
- fsync() writes still need to go to disk
- If we could handles writes, we would get good performance

When is the write performance the most?
- For sequential writes
- What if we turned all write traffic into seq. write traffic

Solution: **write all data just sequentially to the disk**
- Organize the disk as a log
- Append to end of log
- Write both data and inodes to end of log

Problem: how to find the data?
- Solution: indirection: Have **inode map** to point to where inodes are
- Where should inode map be?
- Problem: write to inode, write to inode map - not sequential
- Solution: write inode map along with inodes
- Problem: cant find inode map!
- Solution: **checkpoint region**
  - have a pointer to start of inode map
  - write checkpoint region occasionally

Problem: disk gets full!
- may have old copies of inodes and data that we no longer use
- clean up all this old stuff to make space
- this is done by “garbage collector”
  - read in a segment
    - read segment summary block
  - determine live inodes
  - use that to determine live data blocks
  - write out new segments sequentially

Problem: crash recovery
- use two checkpoint regions
- switch between them atomically
- roll forward capability
Questions

1. What is the recursive update problem? How does LFS solve it?
2. List the steps needed to find the data of a file “/foo” on disk.
3. Explain how LFS knows how to find an inode given its inode number.
4. What are the disadvantages of LFS? Why don’t we all run LFS today?