# Log Structured File Systems

# Motivating Observations

- Memory size is growing at a rapid rate

⇒ Growing proportion of file system reads will be satisfied by file system buffer cache

⇒ Writes will increasingly dominate reads

# Motivating Observations

- Creation/Modification/Deletion of small files form the majority of a typical workload
- Workload poorly supported by traditional Inode-based file system (e.g. BSD FFS, ext2fs)
  - Example: create 1k file results in: 2 writes to the file inode, 1 write to data block, 1 write to directory data block, 1 write to directory inode
    - ⇒ 5 small writes scattered within group
  - Synchronous writes (write-through caching) of metadata and directories make it worse
    - Each operation will wait for disk write to complete.
- Write performance of small files dominated by cost of metadata writes

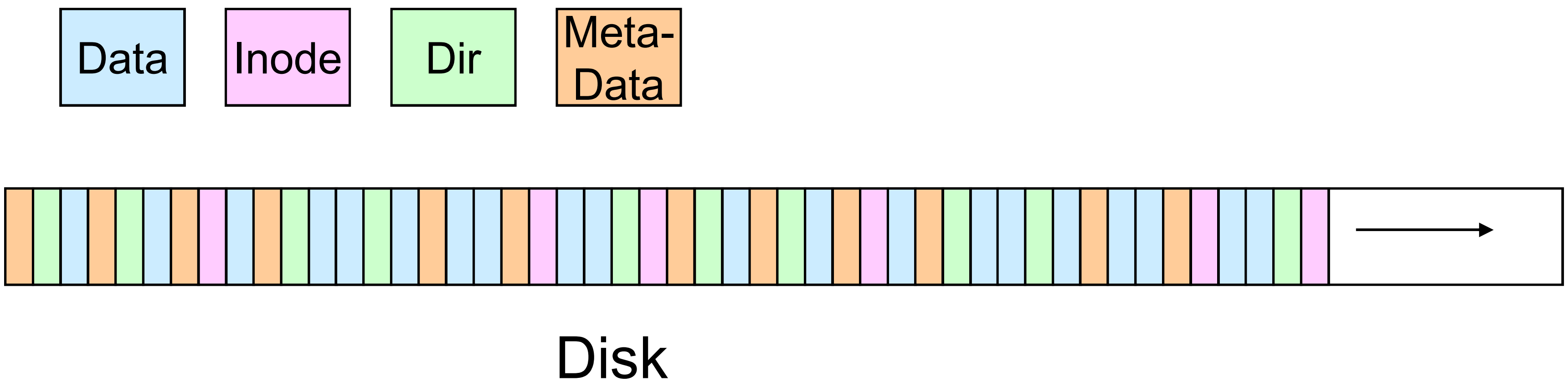
| | | | | | |
|---|---|---|---|---|---|
| Super Block | Group Descrip-tors | Data Block Bitmap | Inode Bitmap | Inode Table | Data blocks |

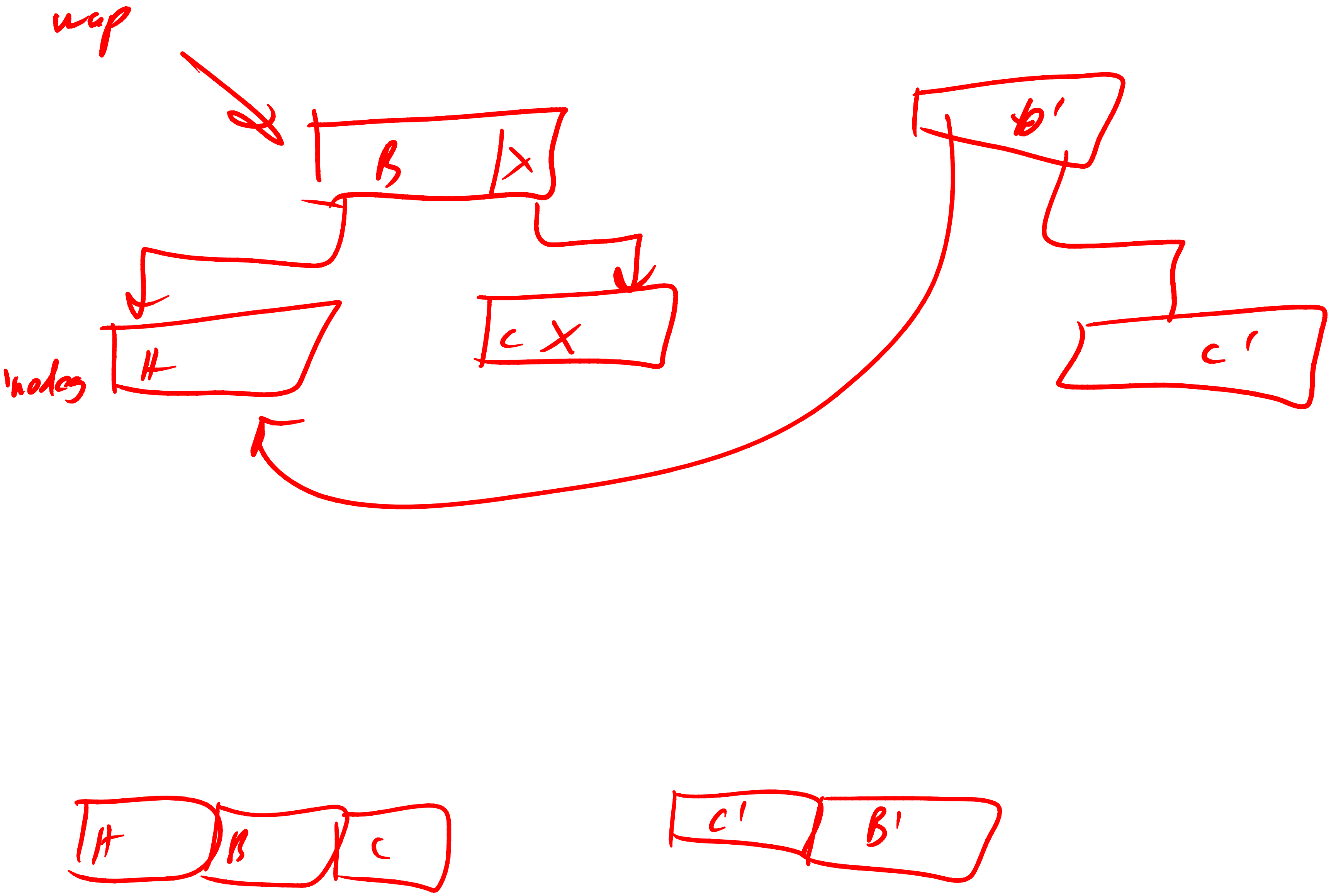
# Motivating Observations

- Consistency checking required for ungraceful shutdown due to potential for sequence of updates to have only partially completed.
- File system consistency checkers are time consuming for large disks.
- Unsatisfactory boot times where consistency checking is required.

# Basic Idea!!!

- Buffer sequence of updates in memory and write all updates sequentially to disk in one go.

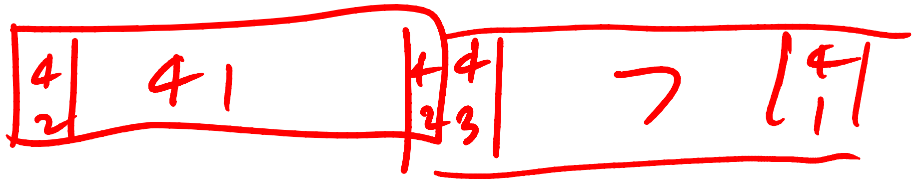




# Issues

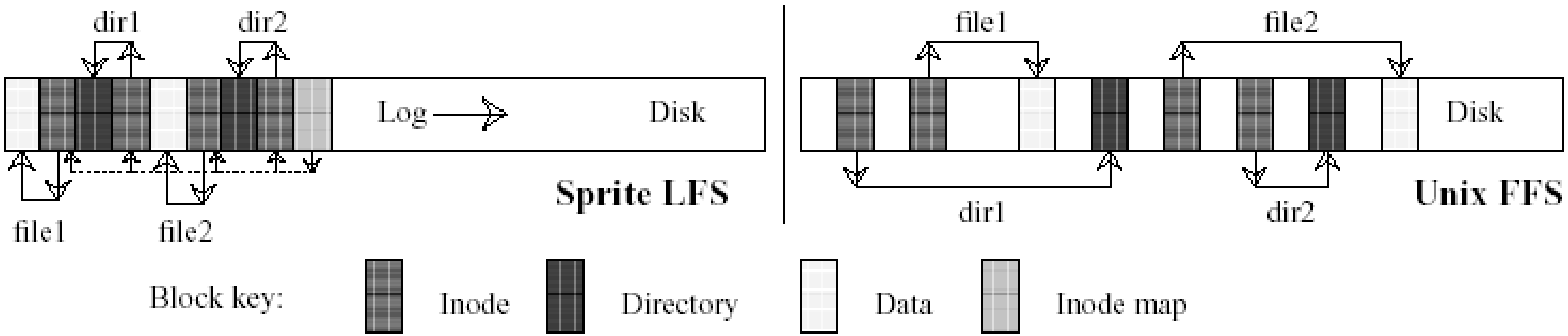
- How do we now find I-nodes that are scattered around the disk?

⇒ Keep a map of inode locations
  - Inode map is also “logged”
  - Assumption is I-node map is heavily cached and rarely results in extra disk accesses
  - To find block in the I-node map, a two fixed location on the disk contains address of block of the inode map
    - Two copies of the inode map addresses so we can recover if error during updating map.



# LFS versus FFS

- Comparison of creating two small files



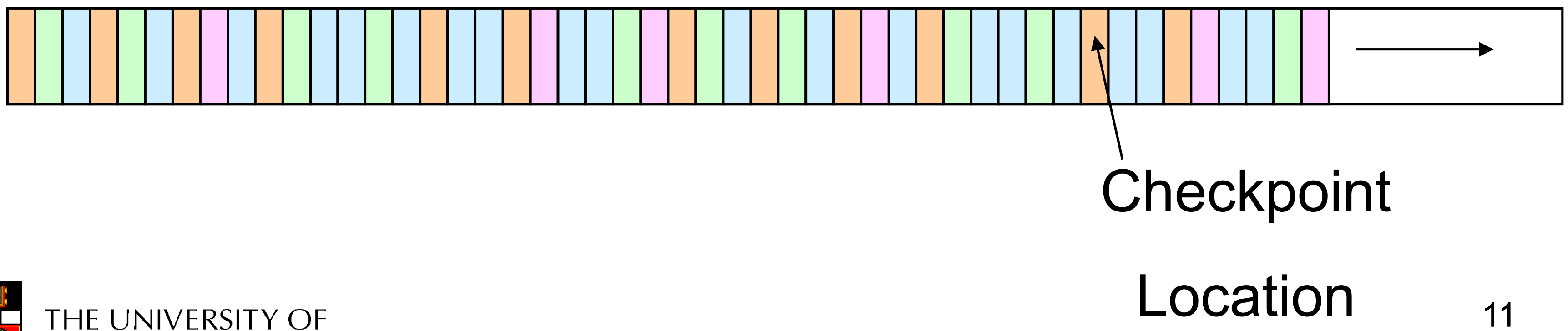

THE UNIVERSITY OF
NEW SOUTH WALES

# Issue
# Disks are Finite in Size

- File system “cleaner” runs in background
  - Recovers blocks that are no longer in use by consulting current inode map
    - Identifies unreachable blocks
  - Compacts remaining blocks on disk to form contiguous segments for improved write performance

# Issue
# Recovery

- File system is check-pointed regularly which saves
  - A pointer to the current head of the log
  - The current Inode Map blocks
- On recovery, simply restart from previous checkpoint.
  - Can scan forward in log and recover any updates written after previous checkpoint
  - Write updates to log (no update in place), so previous checkpoint always consistent



# Reliability

- Updated data is written to the log, not in place.
- Reduces chance of corrupting existing data.
  - Old data in log always safe.
  - Crashes only affect recent data
    - As opposed to updating (and corrupting) the root directory.

# Performance

- Comparison between LFS and SunOS FS
  - Create 10000 1K files
  - Read them (in order)
  - Delete them
- Order of magnitude improvement in performance for small writes