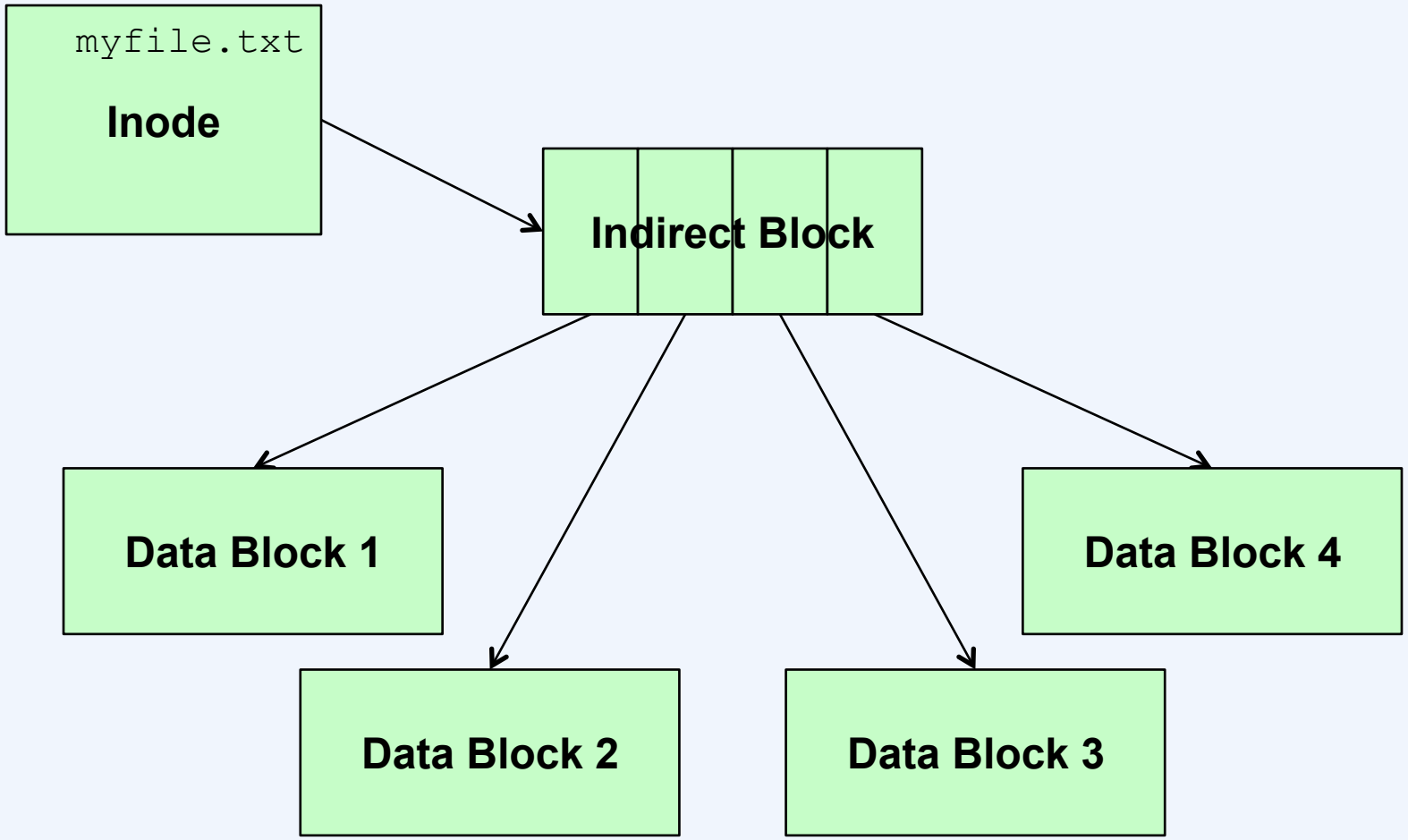
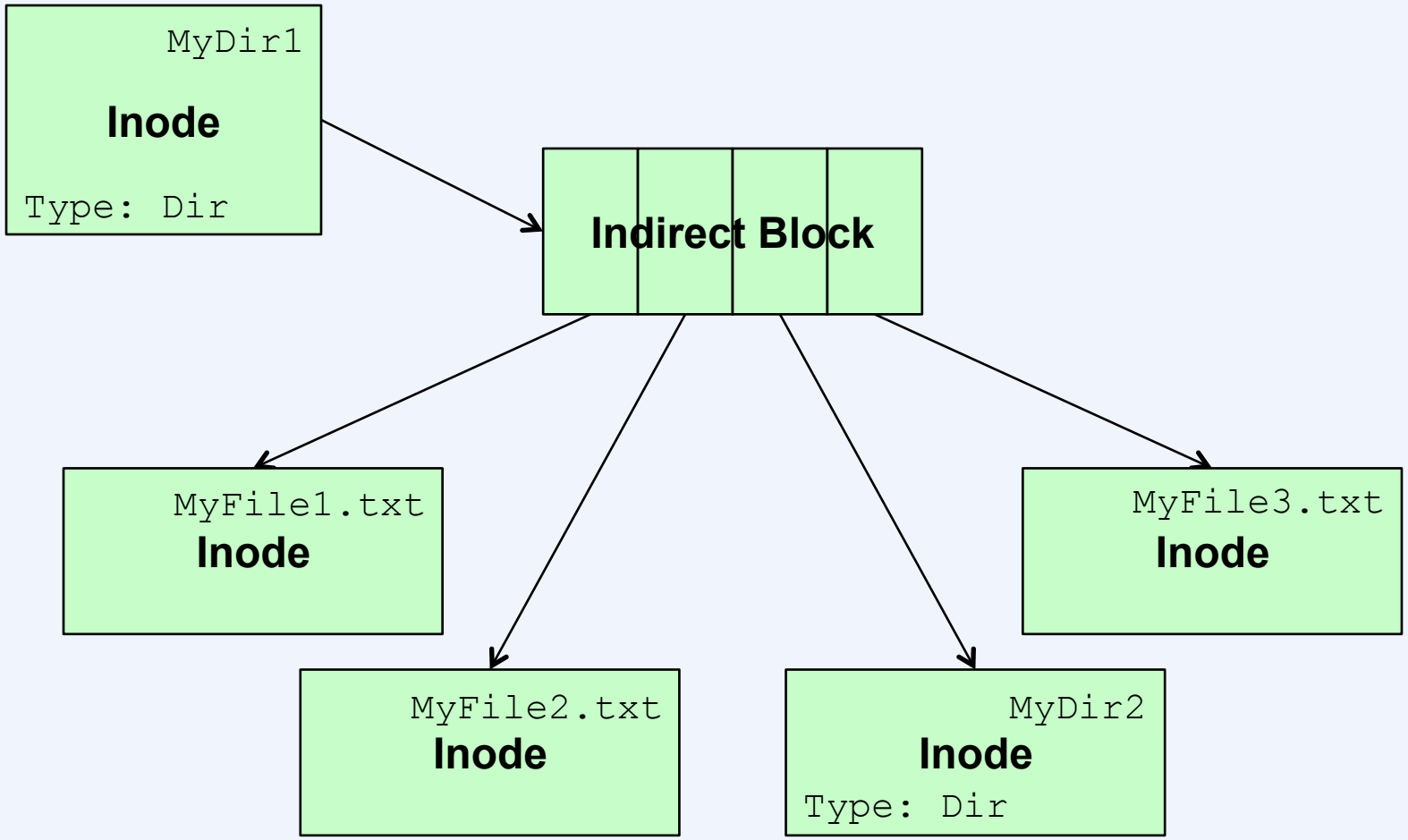
# CS 138: PuddleStore

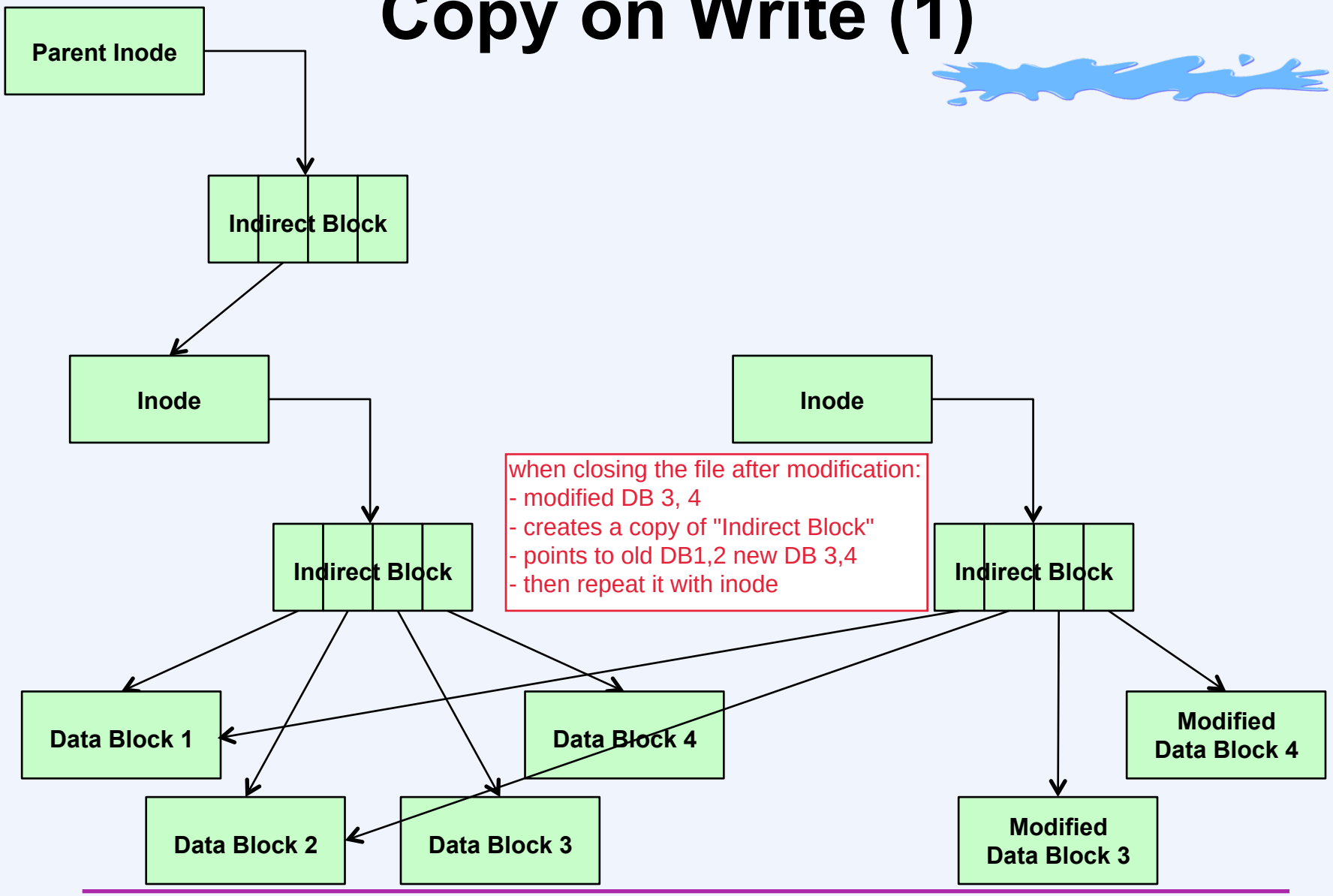
# A File



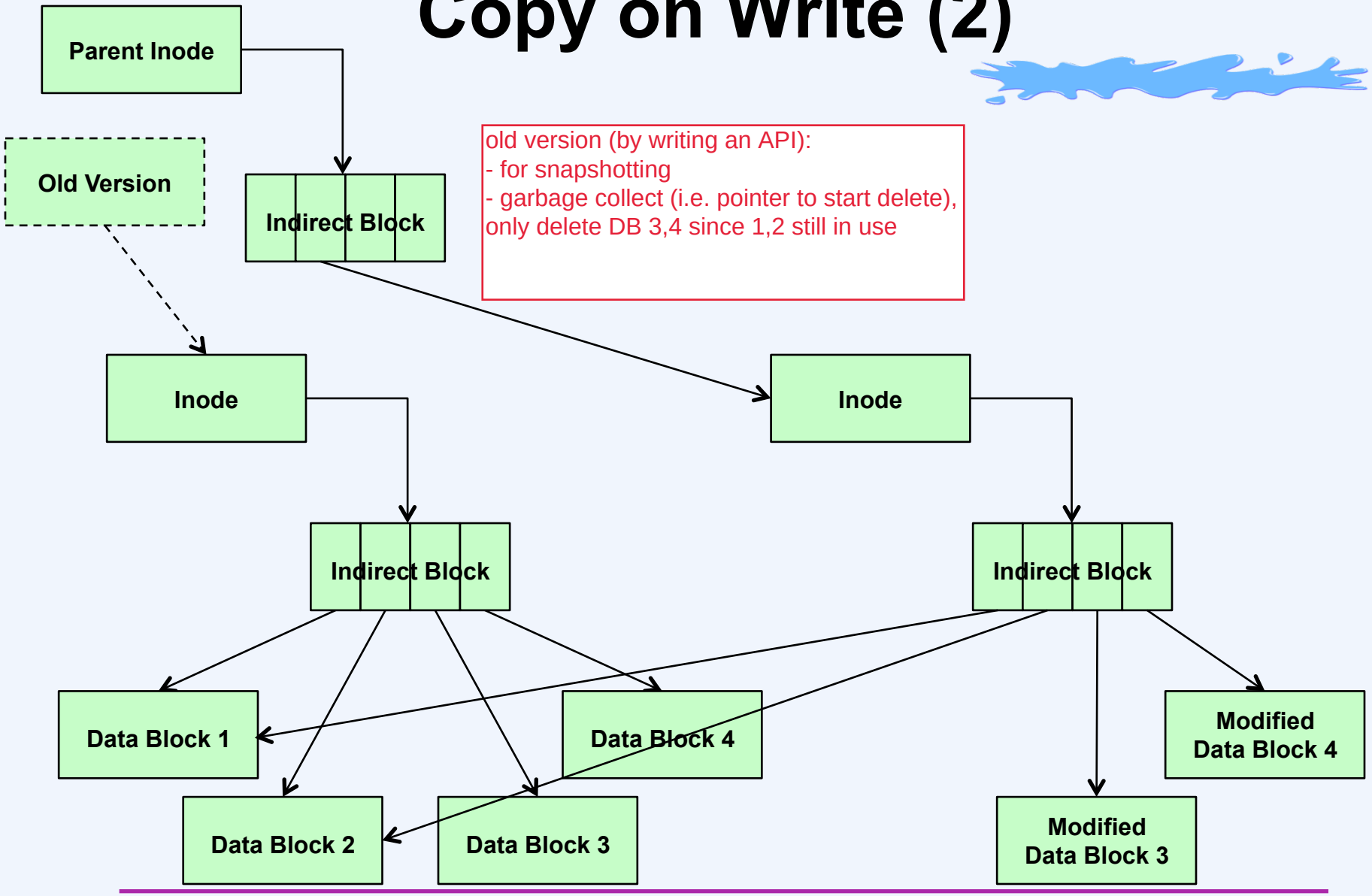
# A Directory



# Copy on Write (1)



# Copy on Write (2)



# Tapestry

- **Inodes, indirect blocks, and data blocks**
  - **stored in Tapestry**
  - **identified by GUIDs**
  - **replicated**

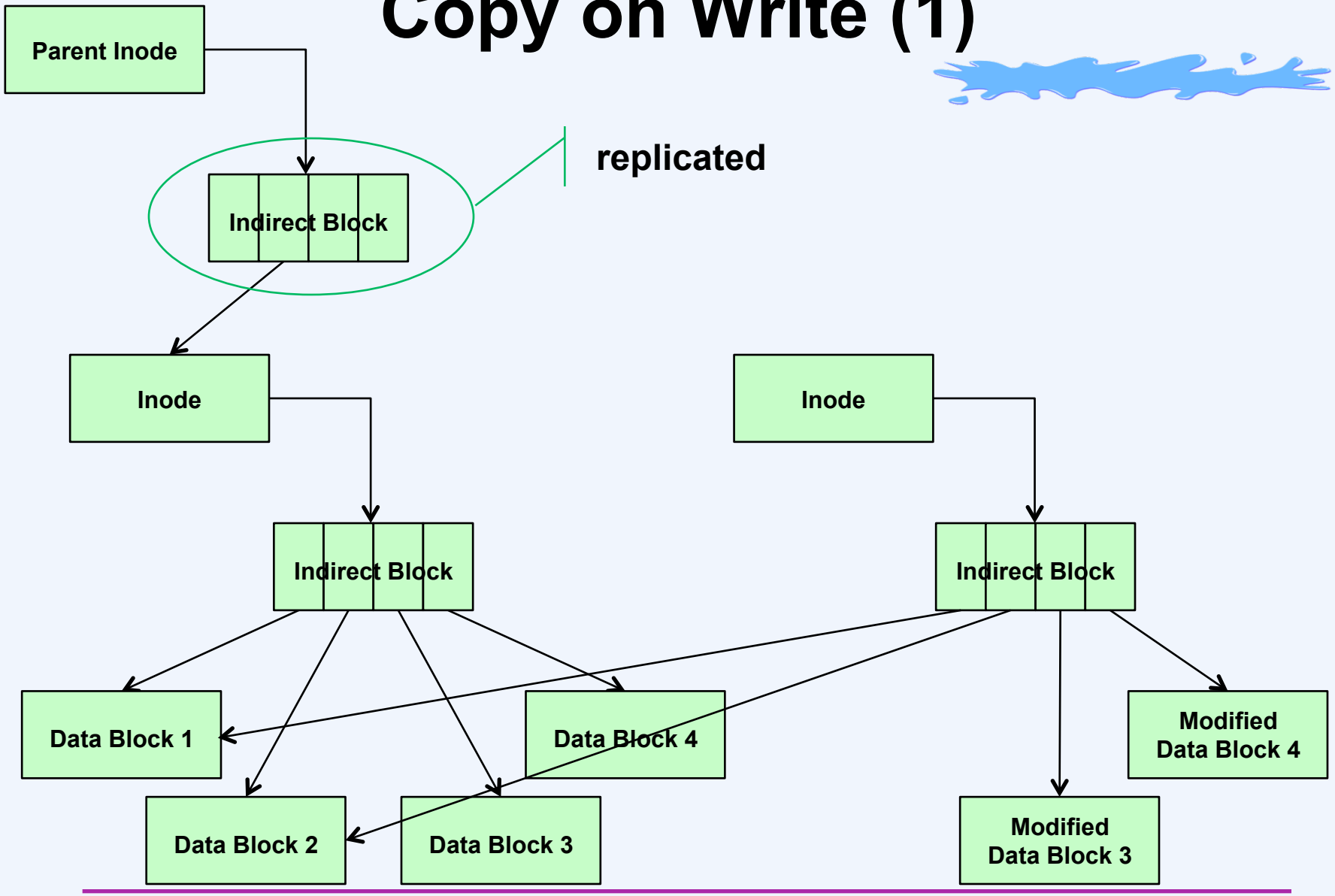
tapestry == index inodes, indirect blocks, data blocks

if you create a new data block/ inodes / indirect blocks, you publish it in tapestry

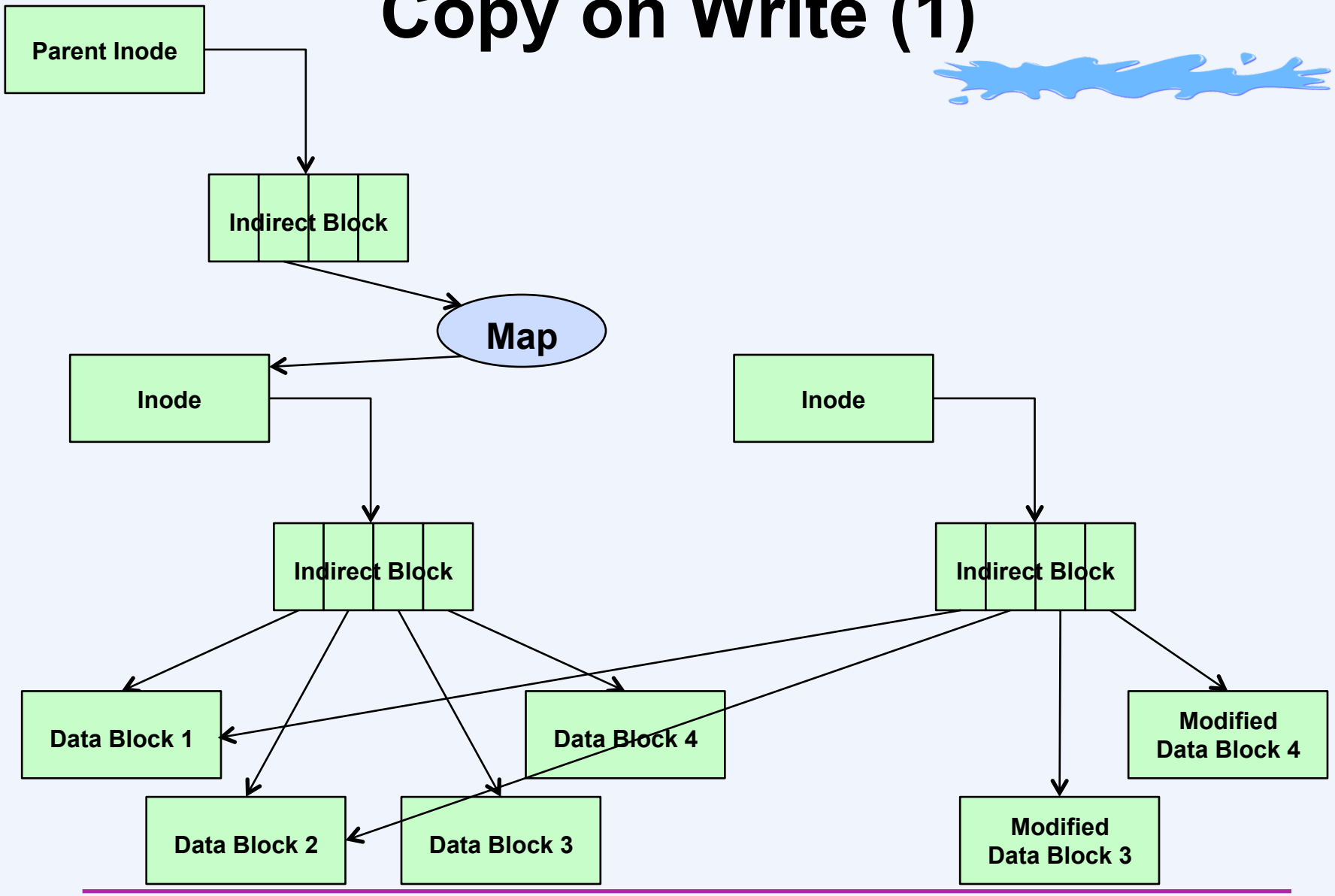
GUID == hash of content of the block, very likely to be unique

Tapestry has to have API implemented to be able to replicate these published blocks

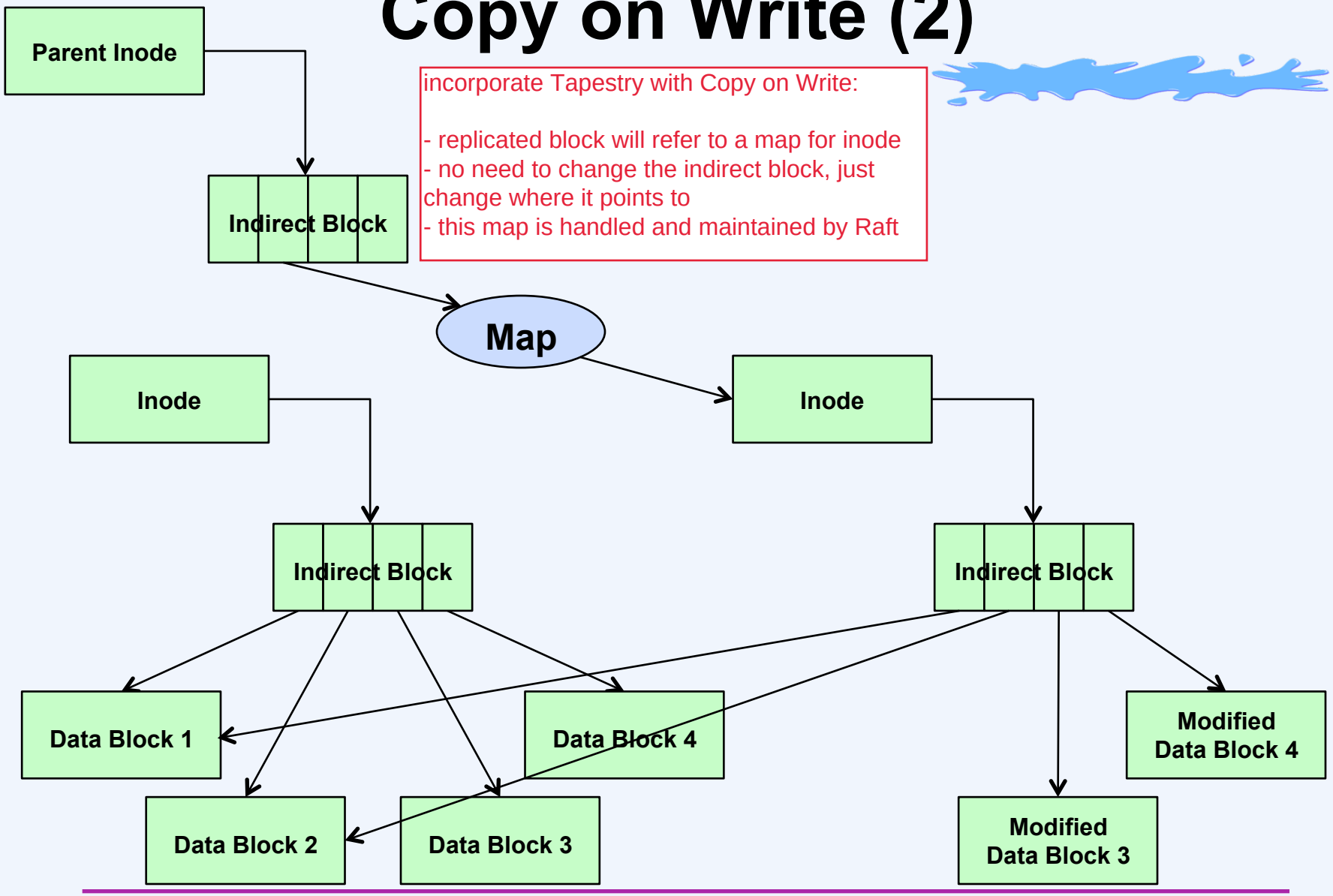
# Copy on Write (1)



# Copy on Write (1)



# Copy on Write (2)



# Raft

- The map is handled by Raft
  - inodes referred to by *Active GUIDS* (AGUIDs)
  - mapped to *Version GUIDs* (VGUIDs)

# Membership Server

- Well known server
  - i.e., all clients know how to contact it
- Identifies Tapestry and Raft nodes
- Identifies root directory's AGUID
- Facilitates adding nodes to Raft and Tapestry

client to connect to cluster:

starts at the root in order to find Raft n Tapestry
root's AGUID == points at root's inode
use that inode to go into Raft replicated map
then use AGUID to map to VGUID of root inode

hardwire into clients the addr of well known membership server
well known membership server knows where the Tapestry, Raft nodes, and root dir are
then just randomly return one

# Required API

- Open return file inode
- Read
- Write
- Create
  - both file and directory
- List
  - contents of directory
- Remove
  - deletes file or empty directory

# Tapestry and Raft

- Your choice
  - use your own
    - you own your project
      - (please don't post it publicly on github!)
  - use the TA version
    - you don't own your project
    - you must sign an NDA

# Final PuddleStore

- You put all this together
  - we give you (most of) the B design
    - if you implement it completely: you get a B
  - if you improve it (reasonably well): you get an A
    - you're encouraged to discuss your design with other teams
- Due May 9

# Design Document

- Due Friday, April 22
  - hand it in earlier, you'll get it back earlier
- Describes A-level features (if any)
  - what is required to implement them?
- Describes API
- How will you test the API?
- What else will you do to test your code?