CSCI4180 Tutorial-9: Assignment 3 Review (Part-2)

Zhao Jia

jzhao@cse.cuhk.edu.hk

Nov. 30, 2022

Content

- Workflow of deduplication
- Storage management
- Upload
- Download
- Deletion (NOT required in assignment-3)
- Hints
- Data reduction techniques

Workflow

- Deduplication happens in the upload process
- Workflow
  - File → Chunking → chunks
  - Computing checksum for each chunk
  - Checking the fingerprint index
    - Chunk with the same checksum exists → update the reference
    - Chunk with the same checksum does not exist → store the unique chunk
  - Storage management
    - Chunks → Containers
    - Metadata

Content

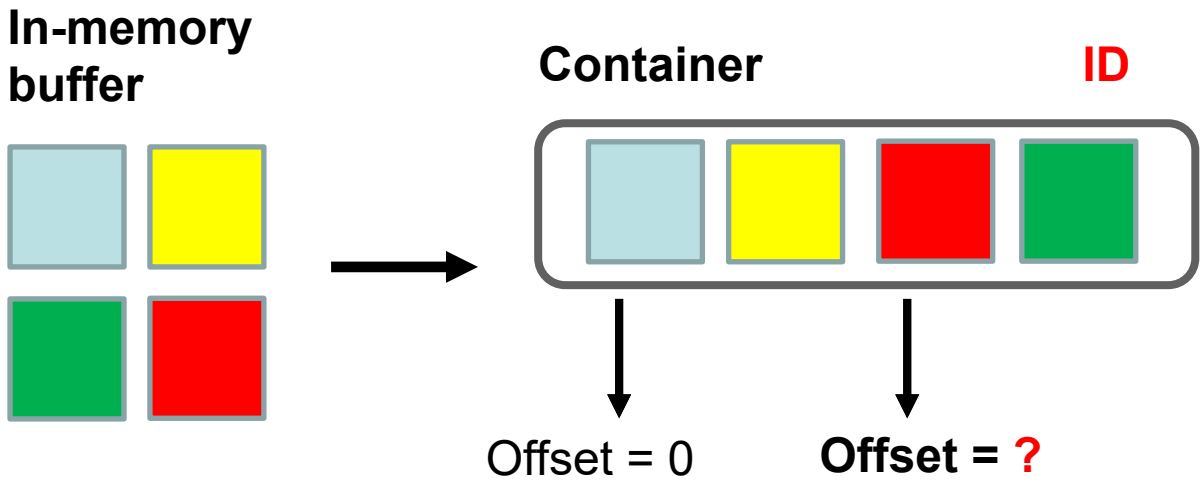
- Workflow of deduplication
- Storage management
- Upload
- Download
- Deletion (NOT required in assignment-3)
- Hints
- Data reduction techniques

Storage management

- Chunks are packed into containers (up to 1MiB each)
  - The client maintains an in-memory buffer (1 MiB)
  - It adds each new unique chunk to the buffer
    - If adding a new unique chunk causes the buffer to go beyond container size limit, the client flushes all chunks in the buffer as a new container and uploads the container to the storage backend
  - Deal with the tail container
    - After the client reaches the end of a file, it should always upload all chunks in the buffer as a new container to the storage backend.
  - Manage the chunk address
    - We need to record the chunk address for download
    - Think about how can we locate a chunk
      - Container ID + offset

Storage management

- Chunks are packed into containers (up to 1MiB each)
  - Manage the chunk address
    - We need to record the chunk address for download
    - Think about how can we locate a chunk
      - Container ID + offset

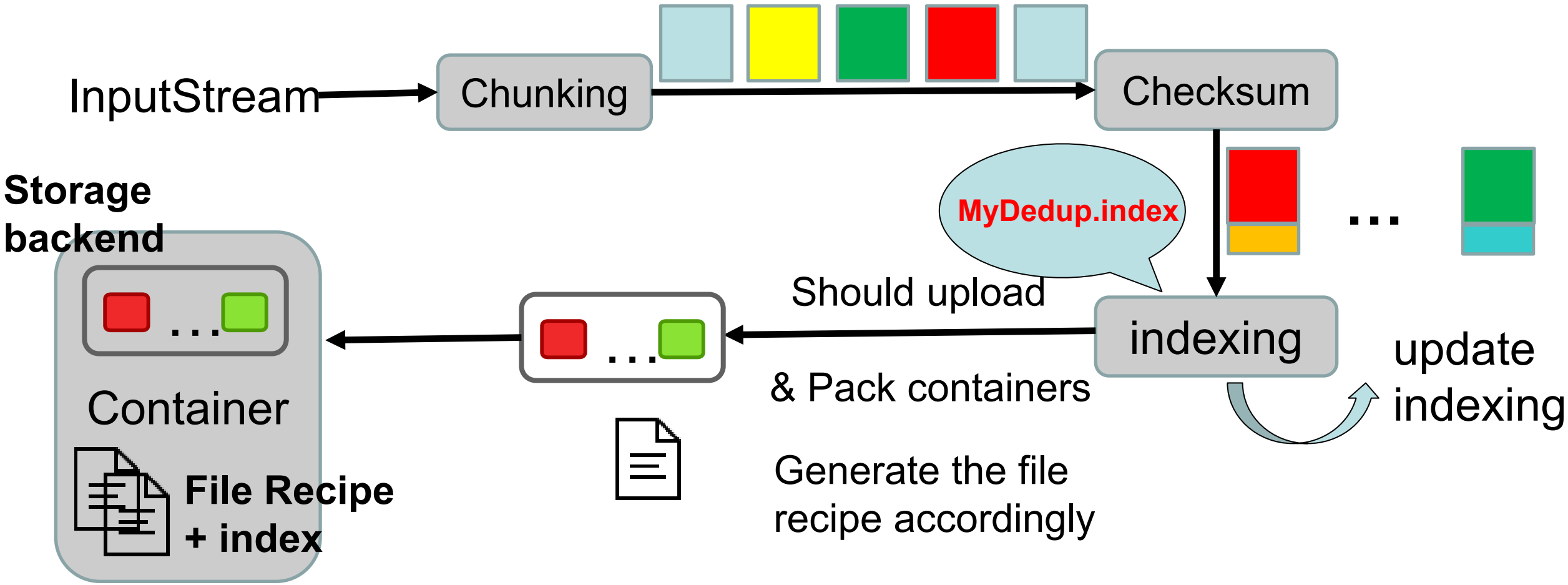


Content

- Workflow of deduplication
- Storage management
- Upload
- Download
- Deletion (NOT required in assignment-3)
- Hints
- Data reduction techniques

Upload

- Upload process
  - The whole workflow



Upload (Cont.)

- Parameters needed in chunking provided
  - min_chunk: minimum size of a chunk -> window size
  - avg_chunk: tells modulus
  - max_chunk: maximum size of a chunk
  - d: base
  - Chunks are identified based on SHA256
- How to decide whether a chunk needs to be uploaded to storage backend?
  - When the chunk is unique
    - which means there is no duplicated chunk stored in backend store.
- Upload/download uint
  - Files are always uploaded/downloaded in units of containers. (1MiB)

Upload (Cont.)

- Statistic for upload
  - Output:
    - Total number of files that have been stored: 1
    - Total number of pre-deduplicated chunks in storage: 2
    - Total number of unique chunks in storage: 1
    - Total number of bytes of pre-deduplicated chunks in storage: 30
    - Total number of bytes of unique chunks in storage: 10
    - Total number of containers in storage: 1
    - Deduplication ratio: 3.00

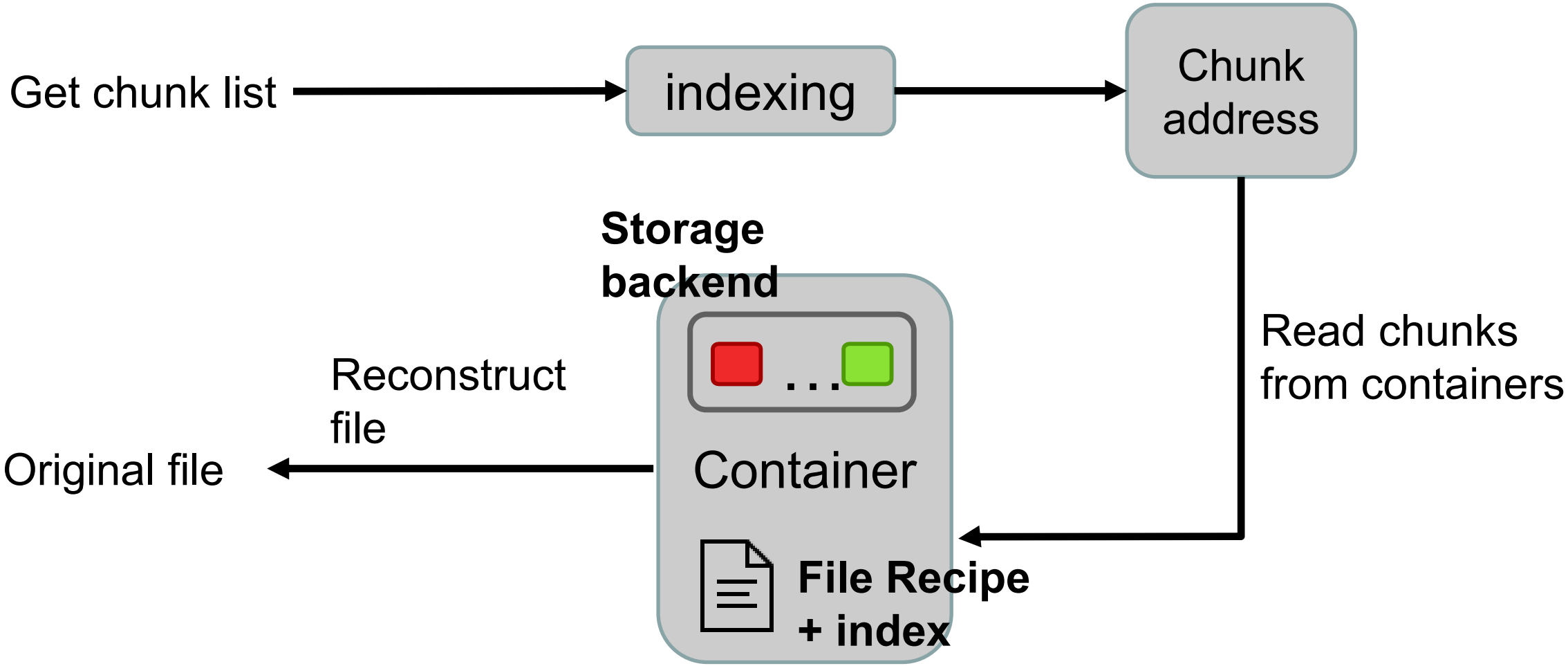
Content

- Workflow of deduplication
- Storage management
- Upload
- Download
- Deletion (NOT required in assignment-3)
- Hints
- Data reduction techniques

Download

➢ Download process

- The whole workflow



Download (Cont.)

- How to get chunk list?
  - We record map between chunk list and file in the file recipe
- How to manage the fingerprint index?
  - 1. We maintain a metadata file called “mydedup.index”
  - 2. At the beginning of each operation
    - we reconstruct the indexing data structure from this metadata file.
  - 3. At the end of each operation,
    - we update the indexing data structure and store it into our metadata file
    - we can rebuild the indexing data structure for the next operation.

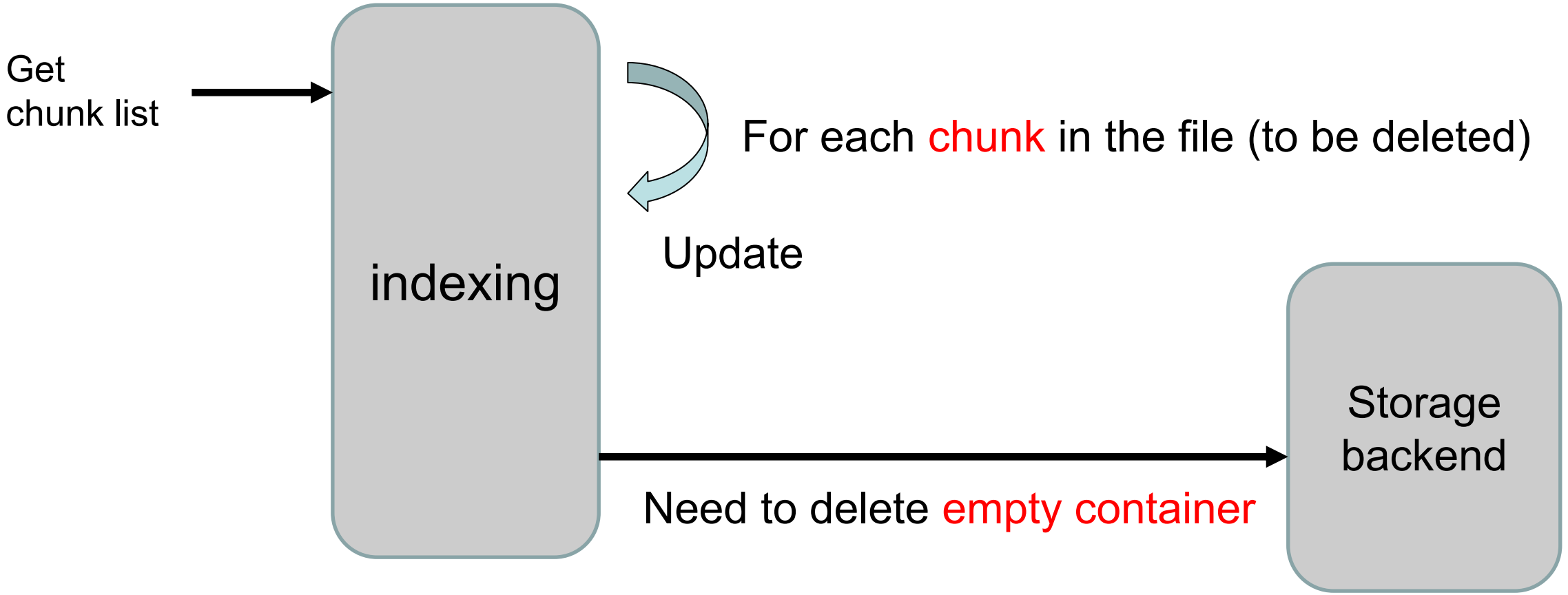
Content

- Workflow of deduplication
- Storage management
- Upload
- Download
- Deletion (NOT required in assignment-3)
- Debug
- Data reduction techniques

Deletion (NOT required in assignment-3)

➢ Delete process

- The whole workflow



Deletion (NOT required in assignment-3) (Cont.)

- We delete data in the unit of Container
  - Delete a file
    - Remove the indexing and metadata of this file
  - Delete a chunk
    - There is no file in our storage that depends on this chunk
    - Its entry in index structure should be removed
  - Delete a container
    - We physically delete a container if there are no valid chunks in it
    - Both the data and entry in index structure should be removed
  - Think about what do we need to maintain in our indexing data structure

Deletion (NOT required in assignment-3) (Cont.)

- We delete data in the unit of Container
  - Think about what do we need to maintain in our indexing data structure
    - Reference count: record how many time a chunk is referred by some files
    - Delete a file
      - For all chunks in this file: Reference count - 1
    - Detect the invalid chunks
      - Reference count = 0
    - Detect the empty containers
      - For all chunks in the container: Reference count = 0

Content

- Workflow of deduplication
- Storage management
- Upload
- Download
- Deletion (NOT required in assignment-3)
- Hints
- Data reduction techniques

Hints (Cont.)

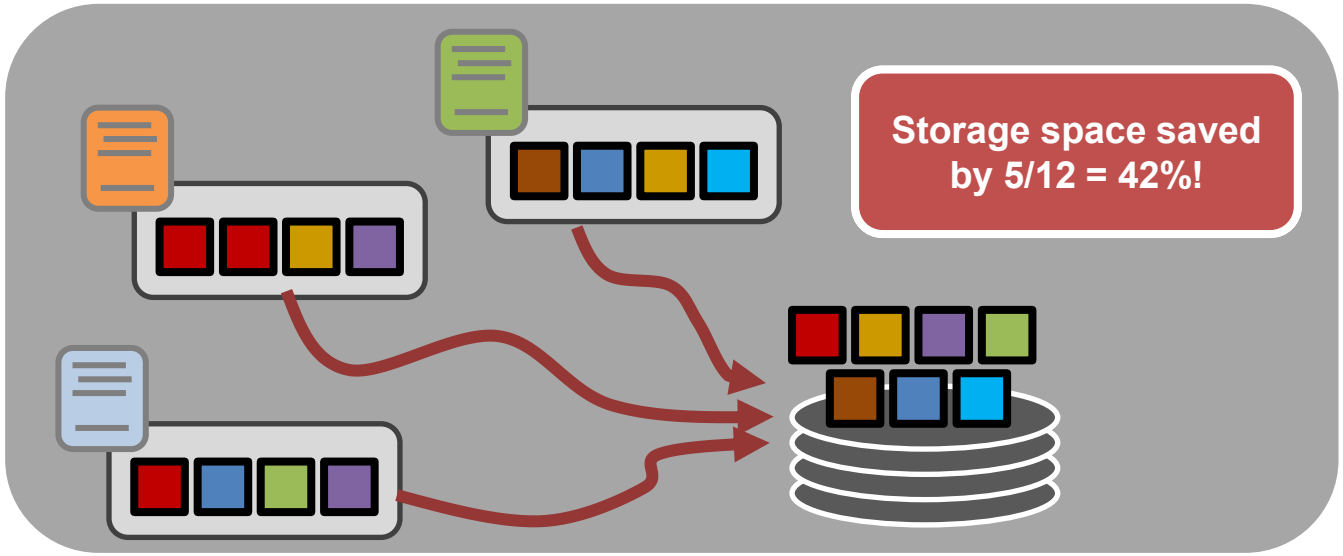
- How to test your program?
  - Create test cases by yourself. This is also an important skill!
  - Can you deal with upload, download, delete operation in sequence without errors?
  - Can you download the file that contains the same contents with the original file that you uploaded?
  - Can you correctly deal with duplicated chunks? (e.g. different file contains same chunks)
  - Can you deal with large files? (e.g. linux image file)
  - Can you correctly pack chunks into containers?
- How about the performance of *your implementation?*

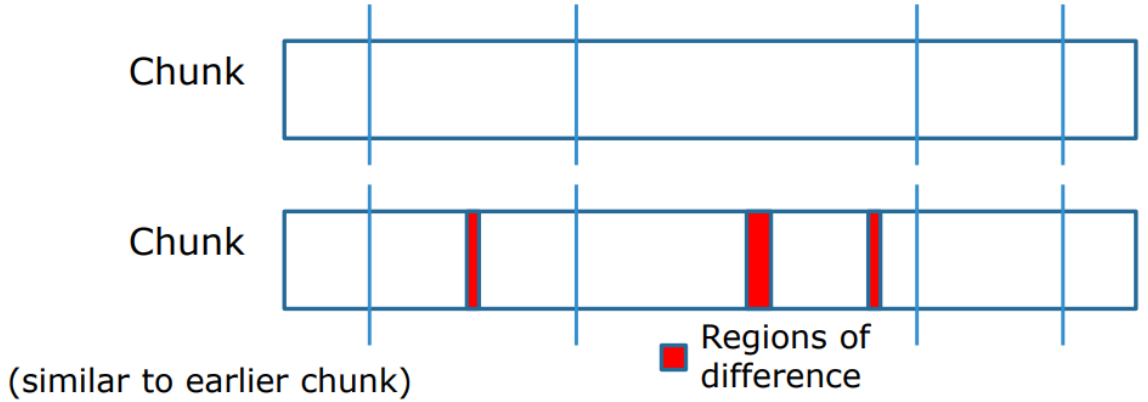
Content

- Workflow of deduplication
- Storage management
- Upload
- Download
- Deletion (NOT required in assignment-3)
- Hints
- Data reduction techniques

Data Reduction Techniques

- Deduplication → Coarse-grained redundancy elimination
- Delta compression → Byte-level compression for similar chunks





- Local compression → Byte-level compression
  - E.g., Huffman coding

Thank you
Q & A