Block-level deduplication with brute-force search

Block-level deduplication without using hash functions can be achieved by directly comparing the content of blocks. Here's a Python script that reads files in chunks and compares the content of each chunk to identify duplicates:

```
import os

def compare_blocks(file_path1, file_path2, block_size=4096):
    with open(file_path1, 'rb') as file1, open(file_path2, 'rb') as file2:
        while True:
            block1 = file1.read(block_size)
            block2 = file2.read(block_size)
            if not block1 and not block2:
                break
            if block1 != block2:
                return False  # Blocks are different
    return True  # All blocks are identical

def deduplicate_files(directory):
    files = []
    for root, _, filenames in os.walk(directory):
        files.extend([os.path.join(root, filename) for filename in filenames])
    for i in range(len(files)):
        for j in range(i + 1, len(files)):
            file1 = files[i]
            file2 = files[j]
            if os.path.getsize(file1) == os.path.getsize(file2) and compare_blocks(file1, file2):
                print(f'Duplicate found: {file1} and {file2}')
                # Uncomment the line below to delete duplicates
                # os.remove(file2)

if __name__ == "__main__":
    # Specify the directory to deduplicate
    target_directory = "/path/to/your/directory"
    deduplicate_files(target_directory)
```

This script defines two functions:

compare_blocks: Takes two file paths and compares their content block by block. It returns True if the content is identical and False otherwise.

deduplicate_files: Walks through a specified directory, compares each pair of files, and identifies and prints duplicates.

As with the previous script, you can uncomment the line within the if os.path.getsize(file1) == os.path.getsize(file2) and compare_blocks(file1, file2): block to delete duplicates if needed. Always exercise caution when deleting files, and make sure to have backups.

Tasks

1. Select some files in a directory, then run the code. Tune the block size from 256, 512,… to 4096, by increasing powers of two.
2. Measure the run time of the code on same files and plot it against the block size.
3. Count the number of detected duplicates for each block size, and plot it against the block size.
4. Try to use parallel computing modules of python to accelerate the deduplication. Repeat steps 1-1, and 1-2, and compare plots.
5. Do you know any algorithmic and data structure techniques that can accelerate the process? If yes, try and replot.
6. What is the result of your experiment? Prepare at least two paragraphs about what you have found?
7. Make a zip file including the report, codes, and plots.

Please send the task deliverables to mahdi.abbasi@lis-lab.fr by the end of **24/01/24**.

Good luck!