CSCI4180 (Fall 2012)

Assignment 3: Deduplication

Due on December 20, 2012, 23:59:59

1 Introduction

This assignment is about implementing a simple storage application with deduplication. We implement a variable-sized chunking scheme using *Rabin fingerprinting*. You keep an in-memory index structure to identify unique chunk. The design of the index structure is up to your choice. Finally, you will experience the use of an object-based storage backend called *Swift* provided by OpenStack.

2 System Overview

You're going to implement a single *Java* program called *MyDedup*. MyDedup supports the following operations: upload, download, and delete.

2.1 Upload

The upload operation includes the following functions:

- *Chunking*. It reads in the pathname of a file and divides the input file into chunks using Rabin fingerprinting.
- *Identifying unique chunks*. Only unique data chunks will be uploaded to the cloud. We use an in-memory index structure to record and identify unique chunks.

2.1.1 Assumptions

We assume that the index structure is empty when MyDedup is started. MyDedup will run continuously, and will not terminate after a file is uploaded. Uploaded files won't be updated. We identify files using their upload pathnames. Different uploaded files must have different pathnames unless they are deleted. After each file upload, you should report the following details:

- Total number of chunks
- Number of unique chunks
- Both numbers of bytes with and without deduplication (only data chunks are counted)
- Dedup ratio: number of bytes with deduplication divided by number of bytes without deduplication

2.1.2 Chunking

We use Rabin fingerprinting for variable-size chunking. Please refer to lecture notes for details. In particular, we divide chunks by checking if an offset matches an *anchor-point criterion*. An anchor point is determined by the *anchor mask* with b 1-bits. If the bitwise AND operation of the Rabin fingerprint and the anchor mask is equal to zero, then we have an anchor point.

A data chunk is defined by the byte range starting from the first byte right after the previous anchor point (or the beginning of the file) to the end of the current anchor point. While we reach the end of the file, we simply produce a chunk between the last anchor point and the end of the file, even though the length of the new chunk can be very small.

MyDedup takes the following parameters from the command line as input parameters: (i) m, the window size (in bytes), which also defines the minimum chunk size (ii) d, the base parameter, (iii) b, the number of bits of the anchor mask, and (iv) x, the maximum chunk size (in bytes), which limits the size of a chunk if no anchor point is found.

Recall that Rabin fingerprint also takes a modulo parameter q. Here, we fix q = 0x100000000, so that every Rabin fingerprint must be a 32-bit unsigned integer.

Chunks are identified based on SHA-1.

2.2 Download

Given the pathname, MyDedup retrieves chunks from the cloud and reconstruct the original file.

2.3 Delete and Reference Counting

You may keep track a reference count for each unique chunk in your in-memory index to identity how many copies of chunks are referring to the chunk. During upload, the reference count is incremented by one; during delete; the reference count is decremented by one. If the reference count of a chunk is equal to zero, the chunk should be removed from the cloud, and the entry should be removed from your index structure.

2.4 Swift

You will use Swift APIs to access the Swift-based object storage. The TAs will give you details on how to use the APIs.

2.5 Sample Input/Output Format

Upload:

```
upload <m> <d> <b> <x> <file_to_upload>

Report Output:
Total number of chunks:
Number of unique chunks:
Number of bytes with deduplication:
Number of bytes without deduplication:
Deduplication ratio:
```

Download:

```
download <file_to_download>

Report Output:
Number of chunks downloaded:
Number of bytes downloaded:
Number of bytes reconstructed:
```

Delete:

```
delete <file_to_delete>

Report Output:
Number of chunks deleted:
Number of bytes deleted:
```

3 Miscellaneous

Note that the program should be completed within a reasonable timeframe for a given dataset. The time limit is set by the TAs. Marks will be deducted if the program runs too long.

The TAs will also provide specific requirements on how to present the outputs for grading. Please attend the tutorials.

Bonus (5%) The top 3 groups who have the shortest total upload and download time for their programs will receive the bonus marks. You may use different techniques to speed up your program, such as multi-threading, elegant indexing structure, etc. To get the bonus marks, you must first correctly implement all the upload/download/delete operations.

3.1 Submission Guidelines

You must at least submit the following files, though you may submit additional files that are needed:

- MyDedup.java

The assignment is due on December 20, 2012, 23:59:59. Demo will be arranged on the next day. Have fun! :)