UiT

THE ARCTIC
UNIVERSITY
OF NORWAY

INF-2202 (Fall 2015)

ASSIGNMENT #2

Deduplication

Ibrahim Umar & Lars Ailo Bongo

17.09.2015

UNIVERSITY OF TROMSØ

Overview

- Your task is to implement a deduplication sender or receiver using Go language.

- Deduplication is a global compression technique often used by backup systems.
  - Achieves very high compression ratio by identifying redundancy over the *entire* dataset instead of just a *local* window
  - Both sides maintain a big *cache* of previously sent data
  - For redundant data, a short *fingerprint* is sent instead of the data content
  - Deduplication systems need to support *high throughput*

- **Deadline: Monday, 12.10.2015 (end of day)**

Deduplication (sender)

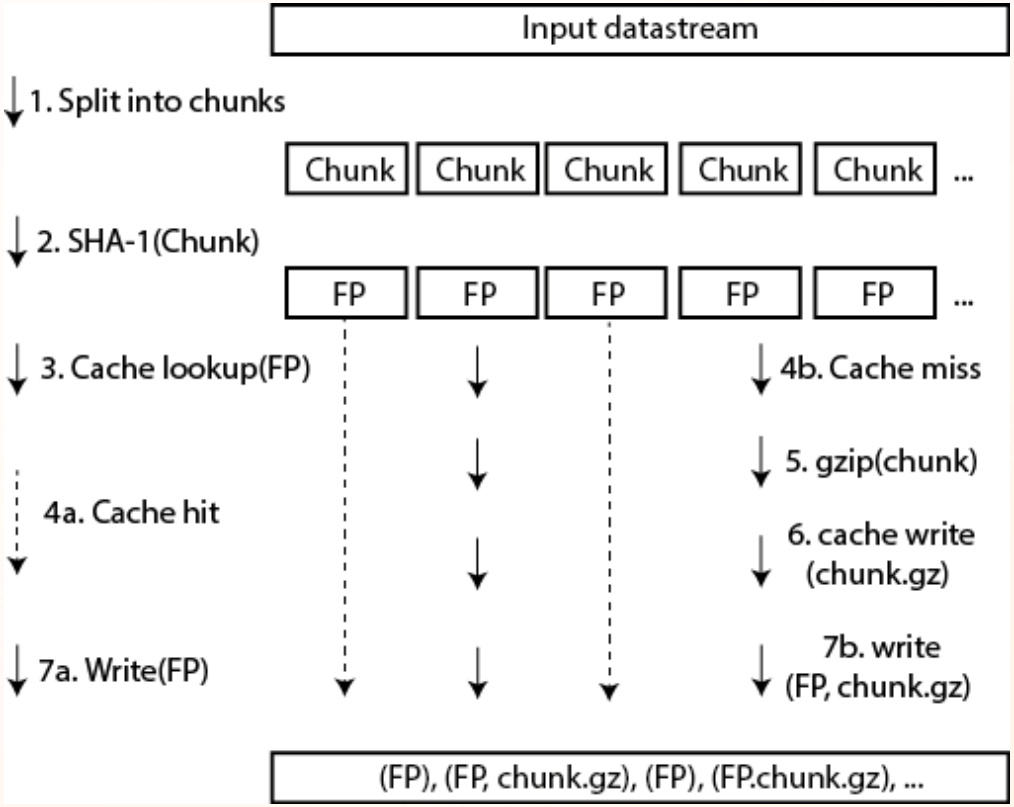

Figure 1: Sender side.

Deduplication (receiver)

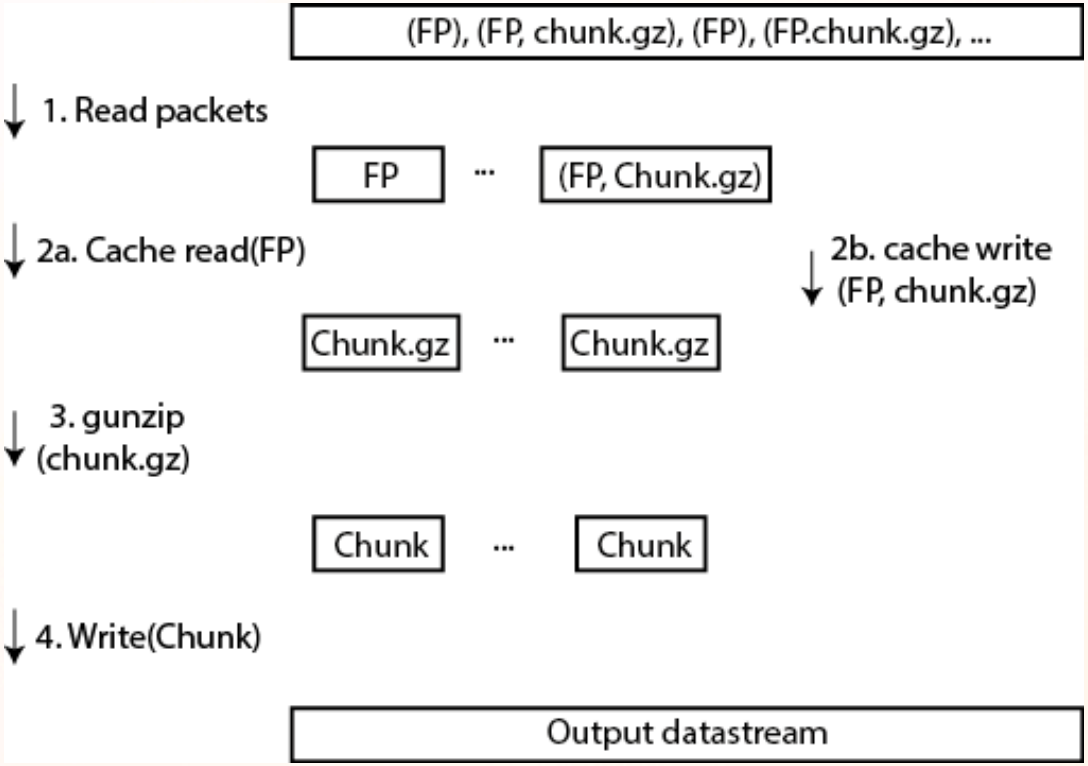

Figure 2: Receiver side.

Test Data

- You will get access to 14 versions of the UniProt database. These are available on:

  ifilab102.stud.cs.uit.no:/data/inf2202

  (note: use sftp or scp to access the files and **do not** run your code on ifilab102)

- You need to make a model for how you will access the data and how much time this will take.

- This model should take into account the dataset size, network bandwidth, and other students.

- You may use an alternative dataset, or a synthetic dataset. If so, your report must discuss the workload selection and workload properties.

What do you need to implement?

1. **Chunking**
   - Code to make chunks out of the data is provided as “parser.go”
2. **Fingerprints**
   - The fingerprints should be 160-bit SHA-1 hashes.
3. **Cache**
   - We assume that the cache can hold all non-redundant chunks and that it fits in DRAM. However, the actual size of the cache may be too large for the computers you have available. If that is the case you must simulate a cache.
4. **Local compression**
   - You should compress the data before sending over the network using a local compression algorithm (see http://golang.org/pkg/compress/).

What do you need to implement? (2)

5. **Protocol**
   - You need to design a protocol for sender-receiver communication. The protocol may send chunks out of order, but it is expected that the input datastream and output datastream are identical.
6. **Compression engine**
   - You should implement a concurrent compression engine using Go. Please use available libraries.
7. **Evaluation**
   - You should do a performance evaluation of your system. To do this you must set goals, select metrics, instrument the code, design the experiments, and report the results.

Requirements

1. Create a model for accessing the dataset.
2. Model, design, and either implement or simulate the chunk cache.
3. Implement deduplication using SHA-1 fingerprints and local compression.
4. Design a protocol for sending fingerpints and chunk data.
5. Implement a concurrent compression engine in Go.
6. Conduct a performance evaluation of your system.
7. Write a report that discuss your models, design, simulation (if any), implementation, experiment methodology, and experiment results

GitHub workflow

1. Fork the assignment repository using GitHub

2. Create a directory in the forked repo using your UIT userID as the name (`<root>/abc123`)

3. Code(s) should be placed under `<root>/abc123/codes` while report is inside `<root>/abc123/report`

4. Work your solution and report using the forked private repo (but please do not make the forked repo public)

5. I will pull all of your repos at **23.59** on **12.10.2015** (no need to make a pull request)

Grading

- Based on your submitted code and report, a PASS or FAIL grade will be given

- Therefore, be sure to adhere to the previously described requirements!

Disclaimer

- Please do not publicize or share your solution or codes anywhere without our permission
- Please refrain yourself to copy other students code(s).
- On the contrary, group discussions and brainstorming for ideas are strongly encouraged