UiT

NORGES
ARKTISKE
UNIVERSITET

Inf-2202 (Fall 2016)
Assignment 2

Deduplication

Tim Teige & Lars Ailo Bongo

20.09.2016

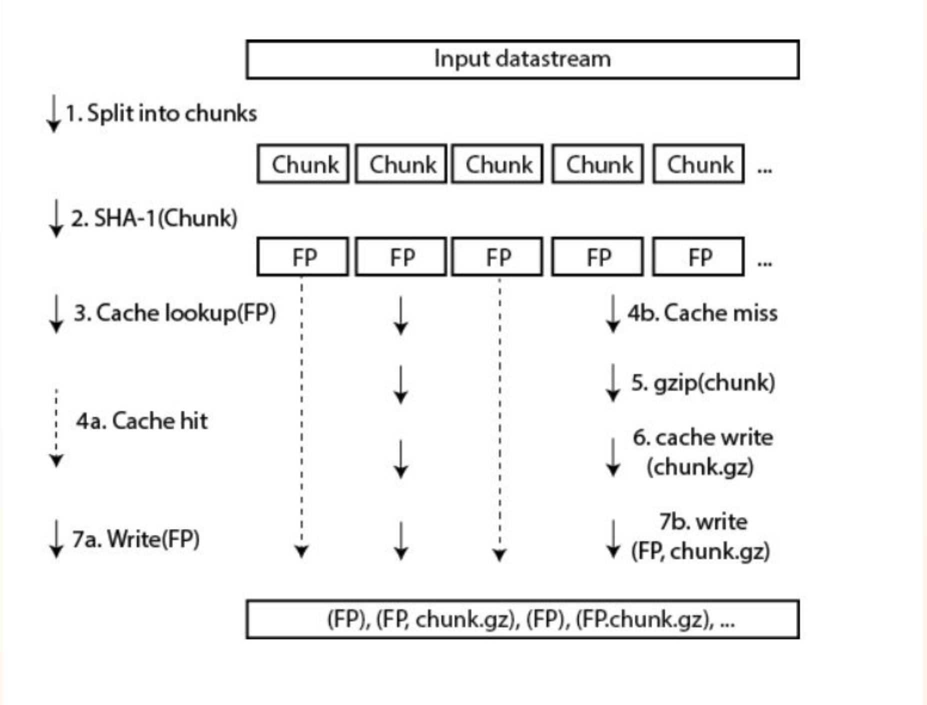
Overview

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

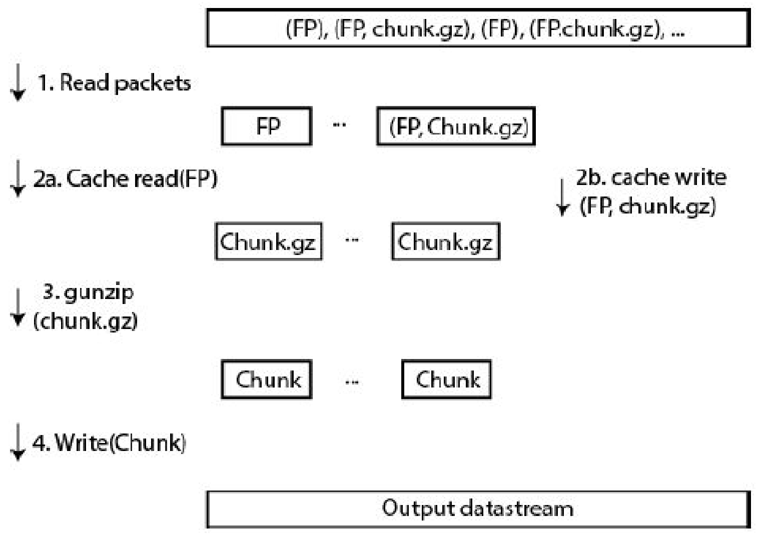
- Deduplication is a global compression technique often used by backup systems.
  - Achieves very high compression ratio by identifying redundancy over the entire dataset instead of just a local window.
  - Both sides maintain a big cache of previously sent data.
  - For redundant data, a short fingerprint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

- **Deadline: Monday 11.10.2016**

Deduplication (Sender)



Deduplication (Receiver)



Datasets

- You will use the UniProt database. These are available at:
  - http://ftp.ebi.ac.uk/pub/databases/swissprot/release/uniprot_sprot.dat.gz
  - http://ftp.ebi.ac.uk/pub/databases/swissprot/release/uniprot_trembl.dat.gz
- Note that these are compressed and will have to be decompressed in order to do deduplication
- You will 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 alternative datasets, or a synthetic dataset. If so, your report must discuss the workload selection and workload properties.

What you need to implement

- Chunking
  - Code to make chunks out of the data is provided as “parser.go”
- Fingerprints
  - The fingerprints should be 160-bit SHA-1 hashes.
- 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.
- Local compression
  - You should compress the data before sending over the network using a local compression algorithm (see http://golang.org/pkg/compress/).

What you need to implement

- 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.
- Compression engine
  - You should implement a concurrent compression engine using Go. Please use available libraries.
- 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 fingerprints 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

- I will send an invite to the new github classroom on the mailinglist.
  If you do not receives this mail by the end of the day (20.09)
  Send a mail to tim.teige91@gmail.com and ask for the invite.

- Work in the repository on your code.
  - If you continuously push to the repository the T.A.
    Will see the progress and be able to help out better

- Place the report inside <root>/report and the code at <root>

- If you wish to work in groups of 2, send me an email with the
  full names of both members. Work on the code in your own repository.

Grading

- Based on your submitted code and report, a PASS or fail grade will be give

- 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