Parallel Data Compression Using `gzip`

Annie Feng Song (afs2185)
afs2185@columbia.edu

Introduction

`gzip` is a file format (and also an application) that is widely used for file compression and decompression. It uses the DEFLATE compression algorithm using the Lempel-Ziv coding. Most commonly, `gzip` is used by web protocols (e.g. HTTP compression) due to the fact that it can be implemented as a streaming algorithm.[1]

Implementations

Perhaps the most well known implementation of the `gzip` compression algorithm is the `zlib` library written in C.[2] There is also a parallelized version of the algorithm, `pigz`, written in C.[3]

In terms of haskell implementations, I have found a version that is essentially a wrapper around the `zlib` library.[4] I have also come across a pure-haskell implementation of the decompression using `gzip`.[5]

Project Goals

My goal in this project is to implement a parallelized `zlib` compression application in Haskell. The parallelized implementation will mimic that of `pigz` (which essentially divides the file to be compressed into blocks and compresses the blocks separately). The files compressed using my implementation should do so in a reasonable amount of time and should be able to be decompressed by `gzip`. I will also conduct correctness and performance testing using files of various sizes and include them in my final presentation.

Project Milestones

Due to limited time, I may not be able to finish the project as I've outlined in the previous section. However, here's the general milestones that I intend to achieve sequentially.

---

[1] https://en.wikipedia.org/wiki/Gzip
[2] https://github.com/madler/zlib
[3] https://zlib.net/pigz/
[4] https://hackage.haskell.org/package/zlib
[5] https://github.com/GaloisInc/pure-zlib

1. Create basic program interface and achieve compression using the Haskell `zlib` library
2. Replace compression component with my implementation of the sequential DEFLATE algorithm.
3. Parallelize the work by referring to `pigz` and splitting the files into blocks.
4. Perform benchmark testing and record results.

The test files that I use throughout the project will be included in the final deliverable.