Assignment 3
SPM course a.a. 24/25

April 18, 2025

Miniz-based parallel compressor/decompressor in OpenMP

Miniz is a lossless, open source, high-performance data compression library contained in a single source file that implements the DEFLATE compression algorithm, which is also used in popular formats like ZIP and gzip. Considering the example code provided (minizseq.cpp and the associated header files) that demonstrates how to compress and decompress a set of files sequentially, you are required to implement your parallel compressor using Miniz and OpenMP. Your parallel implementation should efficiently compress all files provided as input arguments, regardless of whether the files are small or large. Small files can be processed independently and sequentially, while a large file can be split and processed in parallel. However, to compress large files in parallel by splitting them into disjoint blocks of a configurable size, pay attention to the fact that DEFLATE compression relies on the context provided by the entire data stream through techniques like LZ77 and Huffman coding. Therefore, the context (i.e., the number and size of each block) should be preserved to enable proper decompression.

Usage example of minizseq:

./minizseq -r 1 -C 0 dir smallfile.dat bigfile.dat

This command compresses all files contained in the directory 'dir' (walking the sub-directories since -r is 1) and the two files provided as input parameters. The command-line option -C 0 instructs the program to preserve (i.e., do not cancel) the original file. A simple way to create files for your performance tests is to use the following example command (it creates bigfile.dat of size 100 MB):

dd if=/dev/urandom of=bigfile.dat bs=2M count=50

The validation and performance tests must be conducted considering both small (up to a few hundred KB) and big files (some MB). **NOTE:** the internal nodes of the spmnuma cluster do not have disks. This means that reading and writing files go through the network file system (NFS). However, the network file system aggressively caches files in memory (if there is space). Therefore, be careful not to use very large files. Consider a file to be large if its size is less than or equal to 128MB. The size of the entire dataset you may use for testing and performance evaluation (comprising all small and big files) should not exceed 512MB.

Requirements

1. Provide a parallel Implementations of the problem using plain C++17 (or above) and OpenMP. Implement your own sequential and parallel version. The parallel implementation should aim to minimize overheads.
2. Concentrate on compression, as decompression takes a smaller amount of time.
3. Provide a performance analysis of your implementations, considering strong speedup on one internal node of the SPM cluster. Consider cases such as many small files and one single (of few) large file(s).

Deliverables

Provide all source files (in plain C++ with OpenMP pragmas where appropriate), scripts to compile and execute your code on the cluster nodes, and a PDF report (max 5 pages) including a brief description of your implementations and the performance analysis conducted. Mention the challenges encountered and the solutions adopted. Submit by email your source code and PDF report in a single zip file named 'miniz_parallel_<YourName>.zip' **by May 2 EOB.** Please use as email subject "SPM Assignment 3".