Parallel Huffman Coding

High Performance Computing for Data Science - UniTn 2022/2023

Andrea Cristiano: 229370

Gabriele Padovani: 229207

# Introduction

Nowadays, the transmission through the network as well as the storage of a large amount of digital information has become a fundamental part of the human activity. Near those, as part of the process itself, it is possible to find algorithms of compression and decompression which play an essential role. For many applications the algorithms must be lossless, the information must be reconstructed as it was before the compression, and as fast as possible, some of them can operate in real time situations. Huffman coding[1] is likely one of the simplest approaches for data lossless compression. Parallelization instead, in many cases allows processes to reach better performances with respect to their serial version. On the one side, the project aims to develop an application that relays on Huffman coding algorithm to deal with a lossless compression of texts with variable length.  
On the other side, it aims to exploit the parallel paradigm[2] to improve the performances of the algorithm itself. For this purpose MPI[3] library and OpenMP[4] API are used. The application is entirely written in C and the source code is available on GitHub[5]. The document is organized as follows: the next section will briefly introduce the Huffman coding algorithm with all its main characteristics. This is followed by two parts, which will describe the approaches used to parallelize respectively the compression and the decompression phases. Section 3 will introduce the main obtained result, by focusing on specific parts of the algorithm as well as by giving a general overview of this last.  
The final section will report the conclusions of the project.

# Huffman coding

Huffman coding is one of the most used lossless compression algorithms. It can be used to compress many kinds of digital information: text, image, video, audio, etc. Its first version was developed by David A. Huffman[6] which published[7] it for the first time in 1952. The main aspect that made the algorithm so popular is its higher compression efficiency, achieved thanks to the variable-length used to encode the characters. Character with hight frequency will receive a short binary code, while less frequent characters will receive a long one.

The following two sections will explore the encoding and decoding steps. These are mainly focused on the specific case of text, but they can easily be generalized for other kinds of digital information format.

# Compression

The compression procedure consists of the following main steps:

1. Every process reads a piece of the text and counts the number of occurrences for every character, creating a tuple character - frequency.
2. All the processes send their dictionary to the master process, in this case the process 0.
3. The latter merges its dictionary with the others to obtain a complete dictionary.
4. The dictionary is then sorted by using the Odd-Even sort algorithm.
5. Once the dictionary is sorted, it is used to create the so-called Huffman tree.
6. From the Huffman tree, the encoding dictionary is created, which is then shared with all the other processes.
7. After receiving the encoding dictionary, every process encodes its piece of text. It also creates an array that contains the specific dimension of each block that compose the encoded text. Finally, it sends the array and the encoded text to the master process.
8. The master process merges the encoded texts and also the arrays and saves it with the Huffman tree into a file.

# Decompression