High performance Lempel-Ziv compression using optimized longest string parsing and adaptive Huffman window size

Naphtali Rishe, Artyom Shaposhnikov, Alexander Vaschillo, Dmitry Vasilevsky, Shu-Ching Chen

High Performance Database Research Center¹, School of Computer Science, Florida International University, University Park, Miami, FL 33199, (305)348-1706, Fax (305)348-1705, E-mail: {rishen,shaposhn,vasch01,vasil01,chens}@fiu.edu

Keywords: data compression, Lempel-Ziv, Huffman, adaptive window, longest match string searching, asymmetric hashing.

Abstract

We present optimizations that improve the compression and computational efficiency of Lempel-Ziv (LZ77) and Huffman algorithms [Shaposhnikov-93]. The compression performance of the LZ77 algorithm can be improved by using an optimized longest match parsing strategy. Another factor that can be considered is the size of the reference to the matching string, which can vary due to the secondary Huffman compression. We present an efficient parsing algorithm that considers both factors while minimizing the computational requirements. Our second optimization technique optimizes static Huffman encoding by efficiently dividing the input into blocks of varying size with uneven character frequency distributions.

Our parsing method finds the longest match and results in better compression than greedy parsing. The search is optimized using an asymmetric hash function, which works faster than similar methods using hash searching. The output of LZ77 is further compressed by a static Huffman compression with statistically adaptable block size. The algorithm improves Huffman compression by efficiently and accurately estimating the output size of static Huffman blocks.

An experimental program using the proposed optimizations was created and its performance was analyzed using the Calgary data compression corpus. The proposed optimizations result in one of the best performances achievable with LZ77 and Huffman algorithms. The program works approximately 28% faster than GZip and the compression gain is about 4% better. While this is a modest gain, considering that both programs use the same compression algorithms and dictionary size, the improvements are significant and can be utilized to improve many existing data compression systems without requiring the end users to update the decompression software and hardware.

References


¹ This research was supported in part by NASA (under grants NAGW-4080, NAG5-5095, NAS5-97222, and NAG5-6830) and NSF (CDA-9711582, IRI-9409661, HRD-9707076, and ANI-9876409).