A general introduction to Quantized Indexing, describing the inherent advantages of this algorithm over current technologies, and presenting some test results

In 2004, 1stworks began an effort to further improve in the performance of our hotComm collaboration, conferencing and content delivery software. During this project, our Chief Scientist, who combines an academic background in Theoretical Physics with a consuming interest in more efficient data compression, uncovered a method which solves the problem of arithmetic precision in Enumerative Coding, the underlying problem which has long prevented it from being used for general purpose entropy coding, and for coding complex data structures.

While Enumerative Coding has been acknowledged as the most “desirable” universal coding scheme since the 1970s, to this point it has only been used as a technique for encoding CDs and DVDs, where brute force techniques can be used to create the constrained codes.

The wider and longer term research effort to develop a general purpose implementation had previously only produced the Arithmetic Coder, (invented by Jorma Rissanen, IBM Research, 1976) which, while it is in widespread use today as a general purpose entropy coder, is still essentially only an approximation of the exact codes used in Enumerative Coding.

With the solution to the arithmetic precision problem, we immediately recognized the wider potential for this algorithm, from delivering a competitive advantage in large scale applications such as database and web search and media storage to its obvious advantages in content delivery, particularly for audio and video codec applications where the requirement is to most efficiently process small amounts of dynamically and unpredictably changing data.

In addition to its general advantages in speed, compression efficiency and resilience to uncertain inputs, Quantized Indexing uses only simple instructions, shifts and adds rather than more complex multiply and divide instructions. This fact alone delivers an incremental speed advantage for applications running on the lower powered processors typically employed in portable devices, such as media players and cell phones, while significantly extending battery life.

Simply stated, Quantized Indexing always compresses data to a smaller size than the very best research Arithmetic Coders, while simultaneously delivering a significant and measurable speed advantage. In fact, the same Jorma Rissanen who invented Arithmetic Coding, and is still active in the Data Compression

community, understood the technique immediately and was extremely positive after we provided a link to the Quantized Indexing paper on its publication.

Like all data compression applications, your results will always depend on your particular objectives, but the compressed size and speed advantages of Quantized Indexing are always there to be exploited, and are incremental to any other process which may be under consideration.

For a simple, but immediate, appreciation of the inherent speed advantage of this algorithm, and the power of Combinatorial Mathematics applied to Enumerative Coding, consider this:

In processing any string of data, Quantized Indexing does no work at all on the most probable symbol and less work on the least probable symbol. Even that lesser work comprises only simple instructions, not the more complex and power consuming instructions of an Arithmetic Coder, which is also required to inspect every bit that is presented in order to maintain its probability table.

In our own testing and verification with a large number and variety of inputs, such as the sparse arrays which are typically encountered in search applications by companies such as Google, Yahoo and Oracle, the speed advantage is more than 240 times faster, while the compressed output size advantage is “only” 6%. In addition to its speed and efficiency gains, the stability and predictability of the encoded size of the data permits optimal packaging into fixed size fields, for serializing complex data into separately compressed components more suitable for fast random access without the need to expand other intervening data.

In contrast, when tested with small amounts of dynamically variable data such as that processed in the audio and video codecs of companies such as Real or Microsoft, the tested speed advantage is reduced to “only” 20 times, but the compressed output size can be reduced by more than 50%

In either case, when executed on the lower powered processors typically found in portable devices, the compressed output size advantage remains constant, but the speed advantage is increased by a factor of about 1.7, precisely because no complex instructions are required.

Some of our test results are shown below compared to the most highly regarded and tuned Arithmetic Coder currently available in academic research. The #1’s are the number of 1’s in each block size, N, changing to the proportion of 1’s in each block. The N column shows the compressed output size from the Arithmetic Coder as a % above that produced by our algorithm, while the Speed column shows the execution time advantage of this algorithm as a simple multiple of the Arithmetic Coder performance. As a result, the 110% in Vary means compressed output from Quantized Indexing which is only half the size of the best Arithmetic Coder available today

To relate the data to real world applications, the upper right corner of the table shows the larger values of “n”, meaning bigger blocks of data to be processed such as search vectors, where the speed advantage is most obvious. The bottom line of the table, marked Vary, uses smaller values of “N” and dynamically variable inputs, representing the smaller and more variable data found in audio and video applications, where the compression advantage is more significant.

During testing, we also applied a small amount of optimization to the large sparse array example shown in the upper right corner of the table, which increased the 247 times faster result to over 1000 times faster, additional proof of the potential of this algorithm.

#1's	N:4K	Speed	N:8K	Speed	N:32K	Speed	N:128K	Speed
8	6.846	68.3	6.421	112.8	5.447	199.6	5.966	247.5x
16	4.175	59.7	3.830	78.5	3.389	138.1	3.730	168.0
32	2.297	49.7	2.090	58.9	2.096	95.9	2.220	117.2
N/64	1.370	40.3	0.606	41.0	0.186	41.7	0.073	42.5
N/32	0.897	30.8	0.343	33.7	0.123	34.2	0.049	34.5
N/16	0.505	21.8	0.197	25.3	0.084	24.6	0.040	24.8
N/8	0.359	14.4	0.155	16.7	0.069	16.8	0.045	16.8
N/4	0.288	9.2	0.138	10.8	0.083	10.6	0.068	10.5
N/2	0.509	6.6	0.445	6.6	0.367	6.4	0.332	6.4
Vary	110.899%	21.9	96.736	19.6	71.308	16.5	52.580	14.1

For those with a more academic background, who want to better understand the underlying mathematics of the algorithm, please visit www.1stworks.com/ref/qi.htm which contains several detailed technical reports describing the evolution of this innovative work against the background of combinatorial mathematics and enumerative coding.

For those with the more practical focus of software developers, there is also a sample program implementation, together with the source code, which is available for download from that page so developers can better understand and confirm the inherent efficiencies offered by this new algorithm for themselves.

1stWorks has been issued a US patent, 7,161,507, for Quantized Indexing

For more information concerning Quantized Indexing, please contact kdriscoll@1stworks.com or call him on 508 877 9973 Eastern