LOOSELESS ENCODINGS AND COMPRESSION ALGORITHMS APPLIED ON POWER QUALITY DATASETS

Jan KRAUS
TU in Liberec, Czech Rep.
jan.kraus@tul.cz

Tomáš TOBIŠKA
TU in Liberec, Czech Rep.
tomas.tobiska@tul.cz

Viktor BUBLA
TU in Liberec, Czech Rep.
viktor.bubla@tul.cz

ABSTRACT

In this paper we are presenting behaviour of various encodings and lossless compression methods for data stores of power quality measuring campaigns and the achieved results with different generated and real datasets. The main purpose of this work is to ease the selection of an optimal encoding for a typical power quality data in the measuring device. The most important criteria for selecting of an optimal method are a compression ratio, its stability, reliability and fitness of such algorithm for an embedded platform with limited memory and computational power. The implemented benchmarking system is designed to be modular so that each part of it can be separately modified or completely replaced while still the whole system is delivering comparable results. Our target platforms are a resource limited, micro-controller based devices so the most attention was paid to choose an algorithm performing sufficiently with reasonable requirements. The experiments have been performed on different artificial and real data from our measuring campaigns.

INTRODUCTION

Quality of supplied electrical energy is being continuously evaluated in all parts of the electrical grid. In the point of measurement the power monitoring devices (PMD) are used to evaluate many quantities according to the requirements of different national and international standards as for example [1]. Most typical applications are built-in devices which are archiving huge amounts of data for possible later inspections and analysis. For example in standards [2, 3] there are specified different quantities and intervals of evaluation for such measurements. The shortest aggregation according to these standards is 150 periods (~3s) but interval as short as 10 cycles (~200ms) is also defined and can be also recorded if requested by user. In booth of those scenarios a huge amount of data is measured reaching easily hundreds of Mb per day.
The majority of the most compact, microprocessor based PMDs are nowadays storing all their data in its internal memory using a plain binary format separately for each record. This approach is usually good enough as the price/capacity ratio with modern storage devices (disk, flash) is constantly dropping. On the other hand each power quality evaluation archive posses indispensable degree of correlation which predetermines the PMD to implement some sort of compression using their rather limited resources.
By using lossless algorithms the information entropy can be dramatically reduced which results in more optimal utilization of the archive capacity. Apart of saving the device storage space it also can increase the speed of data acquisition and improve the responsiveness of booth device and the operator PC as less bytes needs to be handled between booth of them.
In this article we have studied some properties of selected compression algorithms and its behavior. For this purpose we have first collected real measurement archives. We have also created a modular Linux based system for generation of artificial datasets with configurable parameters. This generator was used to prepare data for evaluation of efficiency and stability of the used compression algorithms more thoroughly.

EXPERIMENTAL DATASET GENERATOR

For the purpose of the power quality data generation we have developed our own modular system (figure 1) which simulates the chain of a measuring instrument in a real situation. Such design introduced generator of signal samples, virtual power quality monitor and a compression unit. Each of these modules could be replaced with a different implementation thus providing data with different qualities and parameters.
For generation of signals we have used synthesis from a definable harmonic components with options which can randomly influence phase and amplitude of each component. The generated signals are also variable in time and can contain repeating cycles with the same statistical parameters. Parameters of each dataset can be modified simply by adjustments in corresponding sections of the configuration files. Each channel can be generated independently with a different set of parameters. The overall sampling frequency, generation cycle, sample data types and amount of generated data is also configurable for each generation.
The generated binary dataset with sample values is than processed by an evaluation engine which simulates a real measuring device, calculates results and outputs data in predefined specified format with various data types and encodings. This implementation enables us to perform simulated power quality evaluation of the same generated data and receive different results in many forms at the same time. Those data are used as a start point for benchmarking of different lossless compression algorithms.
The last block is dedicated to compress the generated data set with selected royalty free compression method – RLE, LZ variants, various entropy encodings and block sorting gzip and bzip2. Results of the different experiments are compared in terms of memory consumption, computational time, complexity of

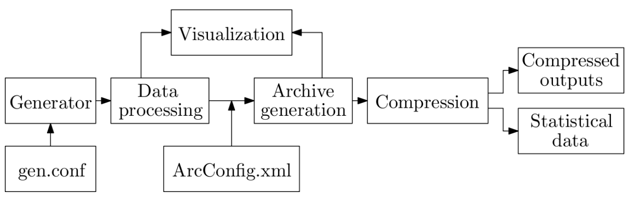

Fig. 1: Schema of implemented dataset generator

implementation and achieved compression ratio. Stability of achieved compression ratios was also evaluated.

COMPRESSION ALGORITHMS

In our experiments we have used different entropy, dictionary and block sorting coders [4,7]. Source code of the selected algorithms can be easily obtained, is practically verified and could be reused (has some sort of general public license). Different implementations of the well known principles have been also chosen for its particular fitness in an embedded system with very limited resources.
As another alternative various lossy implementations also exists in practical implementations like for example the patent pending PQZIP. In our experiments we have not been evaluating them as with this kind of compression the information is always deformed and it would be much more complex issue to compare results of compression while the quality of the resulting data would be different in each case.

Entropy Coders

Entropy coders are creating character encodings which are more optimal than the native representation. One of the simplest algorithm of such group is run-length-encoding (RLE) which replaces repeating sequence of bytes in stream with an escape character and a repetition counter. It is very simple and extremely fast with minimal memory requirements but for most cases with PQ data it does not provide sufficient results.
Two more advanced algorithms that represent entropy category are Shannon-Fano(SF) and Huffman Encoding (HUF). Both methods encode each symbol (bytes) into variable length bit stream. More frequent symbols are encoded into shorter sequences. Both methods use a coding table that needs to be stored within the output stream. Difference in booth algorithms is the construction of that coding table. SF recursively divides the set by symbol probabilities while HUF builds a binary tree by joining sub-trees with lowest probability together. While both algorithms have the same asymptotic complexity, HUF typically provides better compression as it creates binary coding that is more optimal.
There also exist another variants to reduce number of passing through the dataset or to construct the coding table from the data stream. A special case of entropy algorithm is an arithmetic coding (AR) which produces near-optimal output (better than HUF) for a given set of symbols. AR is more effective on datasets where small amount of symbols dominate the rest. Characteristics of the entropy coders are:

- fast, optimal coding of single characters
- efficient for noisy data
- low memory footprint
- simple straightforward implementation
- not efficient for small data sets

Dictionary Compression Algorithms

These algorithms are replacing repeating sequences of symbols with shorter sequence, while creating a so called dictionary of such replacements. Some implementations also use referencing to the previous occurrence of such sequence and thus avoiding the need of fixed dictionary. Basic implementation of such a principle is Lempel-Ziv (LZ) algorithm. From LZ the more optimal implementations typically differ by the construction of the dictionary, its size and other minor improvements. Many variants of the basic LZ algorithm exist and many of them used to be protected by patents.
Highly speed optimized variant of LZ algorithm is Lempel-Ziv-Oberhummer (LZO, MiniLZO) which typically achieves much faster and slightly better compression compared to the original LZ77. MiniLZO variant from the same library [6] was used in our tests. It provides binary compatible algorithm which is optimized for embedded systems – it requires less computational power and consumes minimum of operating memory. Characteristics of the dictionary coders are:

- LZ methods are generally applicable
- provide optimal compression for textual data
- simple and fast decompression
- high memory requirements, slower compression
- MiniLZO is especially optimized for embedded environment

Block Sorting Compression

A relatively new and efficient algorithm is block sorting algorithm presented by Burrows-Wheeler. A popular industrial grade implementation of this algorithm is a bzip2 library which we have used in our tests. It is highly portable and ships as standard solution with many embedded operating systems. Additionally it internally supports limited recovery from archive errors. Its main disadvantage is higher requirement for system resources.

EXPERIMENTS

In the first experiment we have been testing the parameters of generated data in comparison with real datasets. In the figure 2 one such example is demonstrated where multiple instances (thin) of the same real measurement (thick) for approximately 15 minutes are created with the generator. Based on those experiments we have improved its configuration capabilities and tuned the generator so that the artificially generated voltage and current values had similar statistical parameters as the real data. This is an important improvement from our previous work [5]

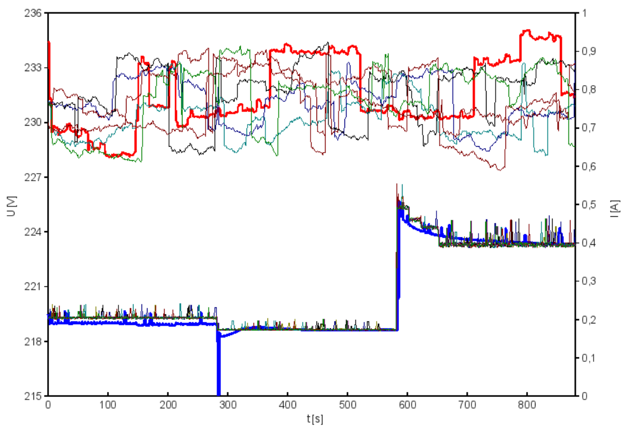

Fig. 2: Real and generated voltage and current samples of a typical measurement campaign

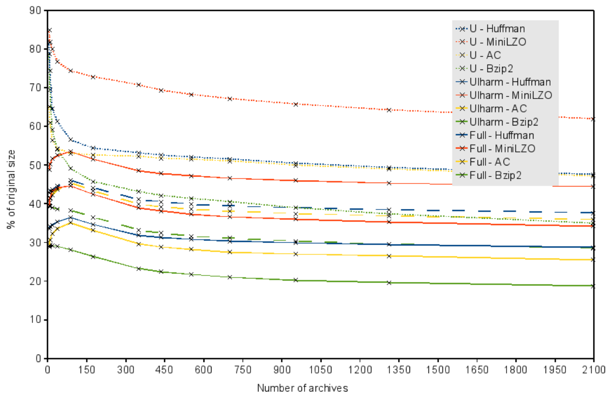

Fig. 4: Compression efficiency by number of archives in the set, delta encoding, different archived quantities

where only randomly generated samples were causing significantly worse compression performance compared to the real measurement.
With multiple randomly generated instances of the same experiment we have been able to test the overall stability of achieved compression ratios for different algorithms. On reasonably large datasets the variation of compression ratio was surprisingly stable (typically below 3%).
Another important factor influencing the compression ratio is a chosen encoding of the quantities in the archive. For the test we have used plain and delta encodings for three different archive sizes. As can be seen in figure 3 the delta encoding was usually achieving better ratio. In some cases the resulting size differed significantly (by more than 10%). It can be also seen that th entropy family of coders (HUF, AC) is much more sensitive to the choice of encoding.

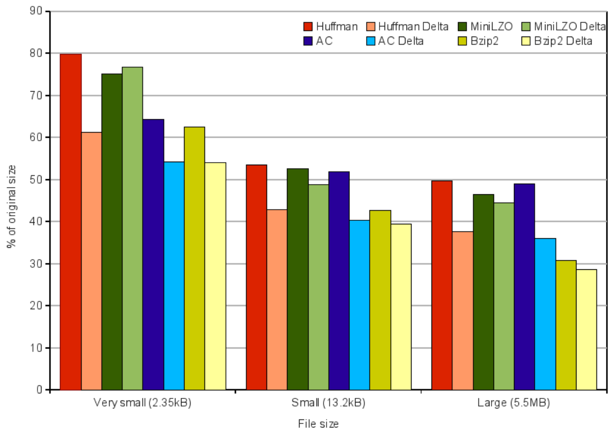

Fig. 3: Compression efficiency on normal and delta encoded datasets

In figure 4 a relation between the size of an archive and the achieved compression ratio is presented for all benchmarked algorithms. Three different combinations of quantities stored in the archive are used – voltage only, voltage and current including harmonics, and full archive including also powers and other quantities.
As can be seen the best compression ratio was achieved by block sorting algorithm bzip2 reaching only 20% of the original file size. The AC and HUF encoders with ratios around 30% might be also interesting for situations where simple and performing algorithm is required. Lastly the LZ variants of compression algorithms represented by MiniLZO do still providing positive compression but they appear to be least suitable for the application on power quality databases.
A relation between efficiency and the size of the archive can be also seen – most of the algorithms are relatively stable for datasets containing above 500 archives in our case. Result of this experiment can help to estimate parameters for implementations of compression in real devices for each supported evaluation interval.
Finally we have measured average time of all the different algorithms on a regular PC to estimate time consumption on the embedded computer. The result can be seen on figure 5. In the real device the actual results may differ based on the used architecture (ARM, Intel Atom, x86 etc.) and available operational memory

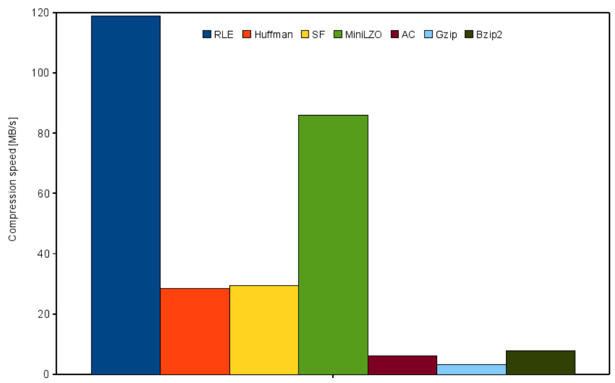

Fig. 5: Comparison of compression performance, different algorithms

capacity. Also the compression libraries which we have used in this benchmark are developed with different level of speed optimization so it is difficult to compare the achieved results without bias. But what can be seen is that even the slowest implementation of bzip2 used in this test is still sufficiently fast compared to the amount of data generated typically by a PMD device per day.

CONCLUSION

According to our previous experiments with power quality data compression a significant improvement can be achieved by using common algorithms to optimize the internal data storage of a modern PMD. We have developed an automatic system capable to generate pseudo-real data for automated and more in-depth analysis of behavior of those algorithms.
To verify its correctness we have simulated multiple measuring campaigns using the generator and compared results with real time data collected directly by a measuring device. For the PMD as well as for the evaluation module of the generator we have used the same ANSI C source code so that we could generate byte arrays of the archive with exactly same structure. Additionally the generator was enhanced by an extra output block to provide different encodings and subsets of the output data.
With the generator we have experimentally verified that the general purpose compression algorithms can significantly improve the utilization of a storage in an embedded PMD. Influence of input encoding on the overall compression was presented on different datasets. Stability of the achieved results have been tested on real and generated datasets. The best performing algorithm in our experiments was bzip2 of block sorting compression. With delta encoding this algorithm was able to reduce size of the full archive to only 20% of its original size.
Our future work should include improvements to the implemented generator to support additional parameters so that the generated voltage and current signals will fit even more closely with the modeled reality. We would also like to use its existing core to create a real measuring device which would support the selected compression natively and to add communication and visualization modules. With such device a more precise analysis of real system requirements would be possible.

REFERENCES

[1] EN 50160, 2000, *Voltage Characteristics in Public Distribution Systems*

[2] IEC 61000-4-30, 2003, *Electromagnetic compatibility (EMC) – Part 4-30: Testing and measurement techniques – Power quality measurement methods,* IEC, Geneva, Switzerland

[3] ERU, 2007, *"Pravidla provozování distribučních soustav – příloha 3: Kvalita elektřiny v distribuční soustavě a způsoby jejího zjišťování a hodnocení"*, Provozovatelé distribučních soustav, Prague, Czech Republic

[4] K. Sayood, 2003, *Lossless Compression Handbook,* Academic Press, San Diego, USA

[5] J. Kraus, 2008, "Optimal Methods for Data Storage in Performance Measuring and Monitoring Devices", *Proceedings of Electronic Power Engineering Conference*, VUT Brno, Czech Republic

[6] M. Oberhumer, 2005, *LZO Real-time Data Compression Library,* [online,15th November 2008], http://www.oberhumer.com/

[7] B. Geelnard, 2006, *Basic Compression Library Manual API version 1.2*, [online, 21th November], http://bcl.comli.eu/