ARCHITECTURAL IMPACT OF CUSTOMIZED COMPRESSION METHODS ON THE SWITCH-BASED ALICE DAQ SYSTEM

H. Beker, W. Carena, R. Divià, P. Vande Vyvre, A. Vascotto
CERN, Geneva, Switzerland

M. Schindler
Technical University of Vienna, Austria

M. Fuchs
University of Frankfurt, Germany

Abstract

In this paper we describe how lossless customized compression can significantly reduce the implementation and operational costs of the ALICE DAQ system.

1. The ALICE DAQ environment

ALICE will study phase transition in collisions of heavy ions at ultra-relativistic energies at the projected Large Hadron Collider at CERN. It will produce data volumes of up to 2.5 Gbyte/s [1] with typical trigger rates of 80 Hz and average event sizes of 30 Mbyte.

The parallel architecture shown in Figure 1 distributes this load over a number of parallel read-out systems, termed Local Data Concentrators. Each is exposed only to the traffic of a part of the detector. A number of parallel event builders process and store the data of all sub-detectors on permanent storage for later off-line analysis. However, each event builder sees only a fraction of all events. While this architecture is completely scalable to almost any data rate, for cost reasons we will reduce the throughput as much as possible in order to limit the number of parallel systems and the amount of data to be stored.

The presented methods are not for general use but aimed in particular at experimental data. As they are adapted to our data they are far superior to generic algorithms (Huffman, arithmetic coding, LZW) both in compression ratio and in speed so as a to keep up with the required throughput. They are now available as commercial hardware with (de)compression speeds of up to 40 Mbyte/s. Our approach uses these algorithms after transforming the data into a more "compressible" data space. Here we concentrate on the compressibility-

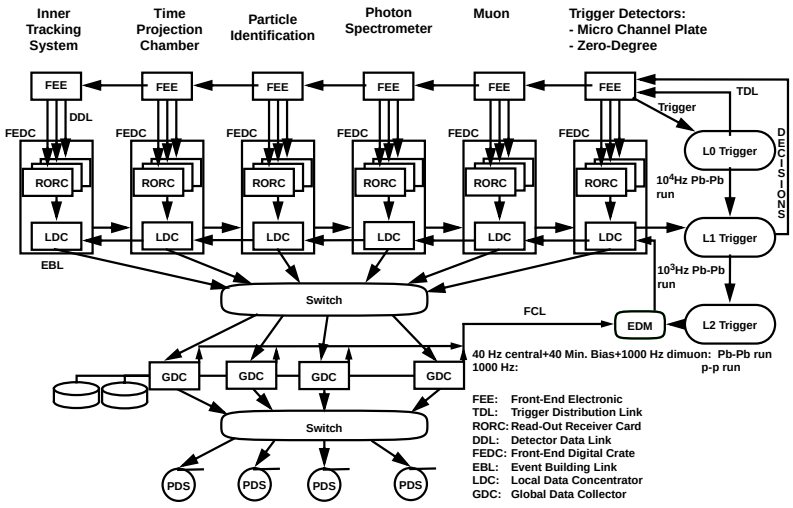

Fig. 1. Schematics of DAQ.

enhancing transformations specific to physics data.

2. General remarks on compression

A distinction can be made between *lossy* and lossless compression methods. Lossy compression is usable where it is permissible to lose quality in transmission and storage while retaining the general content of the data such as in picture, video and audio transmission or storage (e.g. JPEG, MPEG). Here we only discuss lossless methods which permit a bit-by-bit reconstruction of the original data.

Lossless data compression uses the fact that certain data symbols or symbol strings in the input data are more frequent than others. Variable-length codes are assigned to the input data: i.e. shorter and longer codes are assigned to more or less frequent symbols respectively. An intuitive example is the Morse code which assigns "short" to the frequent letter "e" but "short short short long long" to the infrequent symbol ":".

The Shannon theorem defines the information content of a message in the following way:

Given a message made up **N** *symbols – out of a set of* **n** *different symbols – the information content of the message measured in bits is the following:*

$I = N \sum_{i=1}^{n} -p_i log_2(p_i)$

where p_i *is the occurrence probability of symbol i.*

The definition of *symbol* depends on the application; it might be an ASCII code, 16- or 32-bit words, words in a text etc.

Two entropy encoders – Huffman and Arithmetic Coding – compress data close to their Shannon information content. They are described in [2] and are available both as hardware and software.

These encoders only consider single symbols and not their order. This article can be compressed much better when we introduce meta symbols for sequences such as "the", "compression" and "symbol'.

The Lempel/Ziv/Welch algorithm autonomously searches for frequent substrings and codes them arithmetically. It achieves compression ratios which are typically higher than 60%. On modern high-end microprocessors the implementation of the popular program *gzip* achieves a throughput of about 1 Mbyte/s in compression, which is not sufficient for our application.

3. Compression of physics quantities

Let us assume that a charge or any other quantity is measures using an 8-bit digitizer. It is distributed approximately exponentially with a mean value equal to one tenth of the dynamic range i.e. 26. Each value between [0..255] is considered a symbol. Applying the above formula to this distribution a mean information content of 6.11 bits per measured value is obtained. This is almost 25% less than the 8 bits required for saving the data as a sequence of 8-bit bytes. If the dynamic range is increased by a factor of four using a ten-bit ADC, it turns out that the mean information content is virtually the same and hence the possible compression gain even higher (39%). The reason for this is that an exponential distribution delivers a value which is more than ten times its mean only every $e^{10} = 22026$ samples. Even quite long codes for such measurements have no appreciable influence on the compression rates. Given that in a realistic architecture 10 bits have to be expanded to 16, the gain is 62%.

3.1. Data-specific compression

The general-purpose algorithms proceed in the following three steps:

1. They create the statistics for symbols and symbol sequences.
2. They create an optimal (or at least good) symbol to code table.
3. They encode the data according to this table.

Steps 1 and 2 are the most time consuming. In experimental data the statistical properties of the data are known a priori. This allows step 3 to be performed immediately with precalculated symbol–code tables.

General algorithms have another basic disadvantage in our application. They make no assumption regarding the structure of their input data. They perform processor-intensive searches for features such as symbol sequences which might not be present at all. They do not know what to consider as a symbol (1-bit, byte, word, long word etc) and usually assume bytes. In our case data will often be 10- or 12-bit quantities.

It is often possible to transform the data bijectively into a space where they

are less evenly distributed and hence compress much better.

3.2. Compression of zero-suppressed data

3.2.1. Effect of zero suppression on data compression

The ALICE detector will contain about 10^9 sensitive elements. If they were to be recorded for each event, this would result in an enormous data volume.

Therefore the analogue front end electronics and connected read-out buses record only data above a certain threshold. Instead of retrieving all data values in a stream, alternating sequences of addresses and measurements are presented to the read-out bus systems. This method is not lossless, but detector designers know the minimum signal which can be distinguished from the inherent detector noise. Only a minor fraction of all connected channels in a given event surpass this threshold. This results in an enormous reduction factor without introducing dead time, as the suppression algorithm can be executed in hardware and in parallel over all detector channels.

However, the resulting output shows much less, if any, statistical correlation, particularly in the address part. If present, the correlation is hard to detect by standard algorithms as data and addresses are packed into compound data words. The limit in general does not coincide with a byte boundary. Therefore, generic compression algorithms perform poorly on zero-suppressed data.

In calibration events in which many or all data channels are to be recorded, the amount of data can actually increase as the addresses are added to the data. Compression will not reduce the highly-redundant address part in these events.

3.2.2. Enhancement of the channel-address compressibility

The measurement values are separated from the added address. Measurement values follow their own statistical distributions; this can be exploited by assigning them variable length codes.

The ALICE pixel detector, for example, produces only addresses and no measurement values. The front-end electronics outputs a list of pixel numbers. 16-bit addresses are required for the 65.536 channels of each detector unit. The binary data value is implicit and zero-suppression lossless; if an address does not appear in the stream, it means that the associated data value is zero.

There are no major inhomogeneities in the occupancy, i.e. the distribution of addresses will be quite uniform between 0 and 65.535. Therefore standard compression methods are of no use.

The addresses are replaced with the address difference of a hit pixel with respect to the previous pixel. This can be accomplished by a very minor change in the multiplexing hardware.

The original addresses can easily be reconstructed from the new sequence and the address differences will be distributed exponentially with a mean of 50 in the case of a 2% occupancy. The Shannon information content of this

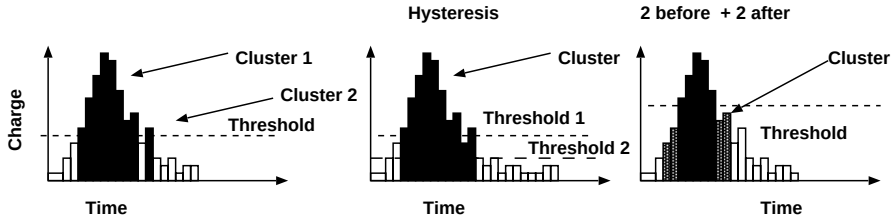

Fig. 2. Cluster definition.

distribution is 9.3 bits per hit instead of 16, resulting in a 42% data compression. A residual correlation in the address differences due to charge sharing between neighbouring pixels will actually allow a 50% compression.

3.3. Enhancement of the compressibility of measurement values

The format and amount of data coming from the time projection chamber (TPC) in ALICE are discussed in [4]. At present, we assume that the data from the TPC will be zero-suppressed. Figure 2 shows how clustered zero suppression can be performed on the TPC raw data. As the time charge buckets are highly clustered it is not necessary to apply an address to every word; the following data structure should suffice:

charge 1, charge 2, ... ,charge N,address of first bucket,# of charge buckets.
next cluster.

We expect about 12.000 tracks in the TPC. Each charged particle traverses on average 75 circular concentric pad rows. Each of these crossings will involve 4–5 neighbouring pads and about 6–7 consecutive time charge buckets. Two bytes are needed for the encoding of the cluster start and the cluster length. At this point we assume a dynamic range of 10 bits and 8-bit logarithmic encoding. The total amount of data produced after zero suppression will be about 35 Mbyte/s before compression. This is the dominant contribution to our sustained bandwidth of up to 2.5 Gbyte/s.

3.3.1. Treatment of analog measurements

The following considerations will be applicable to a wide range of detectors data:

- Digitization accuracy has to be traded off against data volume. In most practical cases the accuracy is chosen so as to be equal to or double the root mean square of the electronic noise. The introduced *digitization noise* only adds 4 or 16% respectively to the intrinsic fluctuations.
- On the other hand – as already illustrated in the initial example – dynamic range does not influence the information content significantly, as long as the measured quantity is distributed approximately exponentially. Coding measurement values logarithmically presents no gain after entropy encoding.

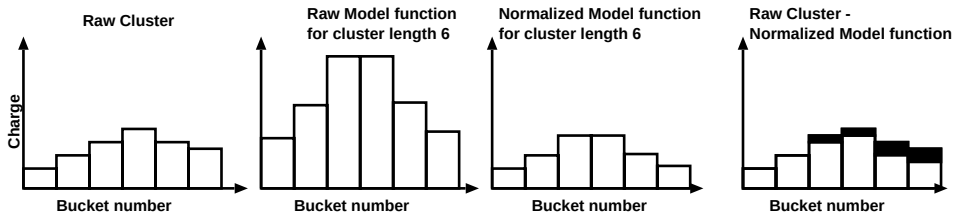

Fig. 3. Lossless waveform modelling.

- Furthermore, compression benefits from on-line equalization of the gains and pedestals of the amplification electronics and digitizers. This narrows the collective distribution of the measurement values and leads to better compressibility.
- Finally the digitizer sensitivity can be limited to the plausible spatial and temporal frequency response of the detector by employing analogue or digital filters to the measurements. Besides reducing noise and hence useless information content, this procedure will be particularly beneficial for the waveform modelling method of the TPC and the silicon drift data illustrated in the following section.

3.3.2. Lossless waveform modelling

The cluster addresses and lengths will be encoded using their own specific code trees. The incremental address-storing approach shown for the pixel detector will further reduce the amount of data from addresses.

Using the assumptions of the initial example which match our TPC data only 25% data reduction on charge measurements can be achieved through entropy encoding. For higher compression rates a detector-specific compressibility-enhancing transformation is perforemed which also applicable for the silicon drift chamber data.

As sketched in Fig. 3, instead of saving only cluster address, length and the time buckets, the mean bucket contents will be added to the cluster data. The time charge buckets are replaced by their residuals with respect to a scaled *model function.*

Scaling involves integer divisions, but rounding errors are not relevant, as the residuals also account for the rounding errors. The residuals have a narrower distribution than the raw measurements and yield better compression more than compensating for the added cluster average.

The model function is obtained a priori by measuring the mean pulse shape of uncompressed data which is known to the encoder and decoder and hence not part of the data stream.

Exact values for the compression ratio will be obtained after evaluation with real data. In the ideal case, the waveform is modelled perfectly and the residuals are only due to the inherent detector noise. If the digitization accuracy chosen is equal to the noise, the number of bits required is 2.15 per residual or 14 for

the whole pulse containing 6–7 samples. After compression, address, cluster length, and cluster mean require 16 bits. This makes about 4 bytes compared with the initial 8–9. When choosing half this accuracy, 3 bytes are for encoding a cluster.

Because of correlation of the noise in the time domain, residuals will form typical sequences and be encoded more efficiently by advanced compression methods.

This method can *losslessly* reduce the TPC data by 50%.

3.4. Data-compression implementation

Data will be compressed as early as possible and no later than in the front-end digital crates in order to relax the bandwidth requirements for all following subsystems, including the automatic off-line tape-handling facility.

As the various transformation steps in the proposed data-modelling schemes are very experiment- and detector-specific, they will be handled in software or in reprogrammable hardware. The final step of encoding can be handled by general-purpose chips. They perform a combination of LZW and arithmetic coding in real time.

4. Conclusions

Customized compression algorithms could allow us to reduce the bandwidth requirements of the ALICE DAQ system by a factor of about two. This has a positive impact on the cost of all subsystems which in most cases scales linearly with the achieved data reduction. Assuming a tape cost in the order of $1 per Gbyte, this will help to reduce the operational DAQ cost stemming mainly from the magnetic tapes by several million dollars a year.

5. References

1. The ALICE Collaboration, *ALICE Technical Proposal for A Large Ion Collider Experiment at the CERN LHC*, CERN/LHCC **95-71**, LHCC/P3
2. M. Nelson, *The data compression book* Prentice Hall, 1991.
3. IBM, *IBM ALDC1-40S Data Sheet*
4. A. Rybicki, *Analytical Previsions of $r\phi$ Resolution of the ALICE TPC*, available from the author (e-mail: arybicki@chall.ifj.edu.pl)