IMPLEMENTATION OF DATA COMPRESSION IN THE DELPHI EXPERIMENT

N.SMIRNOV, E.TCHERNIAEV

Institute for High Energy Physics,
Protvino, Moscow region, 142284, Russia

This article describes the technical details of implementation of general methods of data compression in the DELPHI experiment. Results of compression for different types of DELPHI data are given.

1 Introduction

In this article we consider an application of general data compression methods to the physics data with the aim of reducing the size of data volume. Thereat, it is necessary to distinguish the terms data reduction and data compression. In the both cases the data volume is reduced, but the implied methods and goals are different.

In case of data reduction, the shrinkage of data is a result of special reconstruction programs which convert the signals of the sensitive parts of detectors to the physical values like momenta, coordinates, particle identifications, etc. The goal of data reduction is not only to shrink the data, but also to facilitate further physics analysis. In case of data compression, the shrinkage of data is a result of more optimum data coding and the implied algorithms do not depend of the data nature. The only goal is the reduction of the sizes of the data files to save disk space.

2 DELPHI data processing chain

At the DELPHI experiment the following kinds of data files are used:

RAW RAW data – files with information from the data acquisition system.

FDST Full DST – files produced by reconstruction program used in DELPHI.

LDST Long or Leptonic DST – the same as FDST but also contain the results of particle identification (for leptonic events only).

SDST Short DST – the same as FDST but some detector specific information was discarded and the results of particle identification were added.

mDST mini DST – similar to SDST but contain the most essential information written in more compact way.

It is clear that the most important for physics analysis LDST, SDST and mDST data should be easily accessable by users, i.e. their copies should be located on disks. At the present time this requires 250 Gbytes of disk space (data of 1991-1994). Such amounts of information can produce definite difficulties even for large computer centres like DELPHI off-line analysis centre at CERN. For collaborating laboratories, keeping all information on disks can be a real problem.

Three solutions of the problem are possible:

- installation of additional disks – the simplest solution, but needs financial resources.
- "intellectual" packing, i.e. attempt to take into account the ranges and precision of values in order to pack them into the minimal number of words. This approach could give the best result but it is very difficult for maintenance because the data format is a subject of frequent changes.
- application of universal data compression techniques. Such an approach is cheap and efficient.

3 Choosing of data compression algorithm

At present, data compression/decompression procedures have become a usual operation for users of all types of computers. There are a lot of different tools for that, but no doubt that the most famous and the most popular one is the deflate/inflate method used by the GZIP program.

The choice of data compression algorithm is determined by two main characteristics of the algorithm: the compression ratio and the decompression time. In the case of GZIP, both characteristics are perfect. It is in "public domain", and used for relatively long time and implemented on many different platforms. So, the choice of the algorithm was not difficult.

We also considered several other algorithms, but they definitely were not so good as the one from GZIP. Let us comment two of them.

The first method is LZW (Lempel-Ziv Weltch) algorithm. It is very fast and relatively effective. It is used, for example, in the compress/decompress programs on Unix. The main advantages of this method are that the algorithm description is short, program codes are compact and readable, its implementation requires small amount of working memory. However, the compression ratio of the LZW method is worse than one of the GZIP method. Moreover it is patented and license politics of the patent owner, UNISYS corporation, is strict and aggressive.

The second method is higher-order arithmetic coding. At the beginning of 1995 the sources of the HA compressor implementing this method had been opened for public access. Usually, the algorithm gives better compression ratio than the deflate/inflate method but it is much slower both for compression and decompression.

4 Implementation of data compression in the PHDST I/O package

For Input/Output in the DELPHI experiment the PHDST package [1] was developed. It provides user-friendly access to data with machine-independent specification of external media. The PHDST package uses the ZEBRA-MZ memory management system to manipulate internal data structures and the ZEBRA-FZ package for computer-independent input/output [2]. The base data structure in ZEBRA is a bank – array with information which can be accessed by its pointer. Such banks can be joined into more complex data structures – lists, trees, nets. In PHDST such structures are used to represent physics events.

Each complex data structure (event) is accompanied by a small array of information, the so called pilot record. The pilot record contains some general information about the event and allows user to decide whether to read for analysis or to skip this event.

Events are usually collected in files on tapes. The typical size of one file is about 200 Mbytes that corresponds to the size of one tape (3480 type cartridge).

One possible solution of including data compression in analysis process is to compress whole files once and decompress them only during the usage in analysis program. Such approach has two significant drawbacks:

- usually tens of programs work with the files simultaneously. As a result many files would be in decompressed state at the same time. This would significantly decrease the effect of file compression.
- very often analysis programs use only few events from a file, not all of them and the decompression of the whole file would decrease essentially the performance of the program.

Different scheme of data compression was implemented in DELPHI. The data compression has been applied to each event separately. It has been used the possibility of FZ to write not only on external media (disks, tapes) but also into internal memory of the program. This allowed to implement the following scheme (Fig. 1):

1. Instead of writing out the event into external file, it is saved in the special array;
2. Information in this array is encoded by some compression procedure, for example GZIP deflate;
3. The result is placed into ZEBRA bank;
4. This bank is put into the external file.

The procedure of reading is inverse to the procedure of writing.
Let us consider each step in more details.

1. The output stream must be initiated with

```
CALL FZFILE ( LUN, LREC, 'MXUO' )
```

where	'MXO'	–	specify memory mode and exchange format for output,
	'U'	–	suppresses the byte inversion, this has a sense only for machines with the right to the left byte numbering.

2. Two subroutines mmzip and mmuzip for in-memory compression/decompression were implemented. They are based on the code of the GZIP program and can compress and decompress data from one array to another. They are written in C, but intended to be called from FORTRAN and have the following user interface:

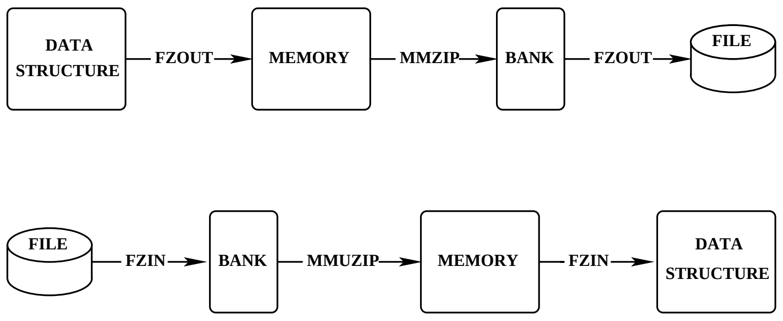

Figure 1: Data compression/decompression principal schemes

```
memory to memory compression
    CALL MMZIP  (srcsize, src, tgtsize, tgt, irep)

memory to memory decompression
    CALL MMUZIP (srcsize, src, tgtsize, tgt, irep)
```

where	src	–	input array with the size srcsize
	tgt	–	output array with the max size tgtsize
	irep	–	reply

3. The pilot record and the data structure itself are compressed separately and ZEBRA bank with pilot record is created. First few words of the pilot record contain information about the compression algorithm used, lengths in bytes of coded pilot record and data structure, and other service information.

4. The usual output procedure: compressed bank is written to the disk by the FZOUT routine which outputs data in computer independent format, so the data can be transferred to and then be read on any computer platform by FZIN routine.

The advantages of this approach are summarised below:

- Compression/decompression procedures are completely independent from data format (Full/Long/Short-DST ...).
- Data are still stored in machine independent format, for this reason there is no problem for home laboratories.
- Flexibility and transparency: one compression algorithm can be easily replaced by another, all modifications are located in the I/O package, thus there is no changes in user codes.
- Event by event structure of the data file remains unchanged; it allows to use direct access to events.
- Pilot record and d/s itself are compressed separately, so user can decide whether to read the event or to skip it.

5 Results of data compression in the DELPHI experiment

The results of applying of data compression to different data files used at DELPHI are shown in Table 1. Compression ratio is defined as

$$Compression\_ratio = \frac{Original\_size - Compressed\_size}{Original\_size} * 100\%$$

Table 1: Compression ratios for different types of data

Data type	RAW	FDST	LDST	SDST	SDST (MC)	mDST
Compression ratio	47%	46%	49%	40%	44%	27%

Timing of event input/output gave the following results:

- Creating and writing of compressed data is approximately 4-5 times slower than writing the same data without compression. But this procedure is made only once for all data during the production.
- Reading of compressed data is 30% slower than reading of ordinary data. So, the delay is negligible with respect to the time used by physics analysis part of a user program.

One remark should be made. We compared the size of file produced by the procedure described above, i.e event-by-event compression and the size of file produced by the program GZIP, i.e. compression of the whole file. The later is usually 10% less. This decrease results from the small size of some (mainly leptonic) events. We found that the compression factor grows with event size up to 64 Kbytes. Above 64 Kbytes the compression factor does not depend on event size. Typical event size is 25 Kbytes for hadronic event (SDST). It creates a possibility for further improvement by clustering of few events in one hyperevent with the size slightly more than 64 Kbytes and compression of such hyperevents.

Acknowledgements

We are grateful to all our colleagues at the DELPHI experiment for numerous suggestions for the package development. There were many fruitful discussions with V.Perevozchikov and Yu.Belokopytov. Our special thanks go to the DELPHI Analysis Software Group coordinator M.Mazzucatto for his permanent attention to the package. This work was partially supported by INTAS (contract INTAS-93-3602).

References

1. V.Perevozchikov and N.Smirnov, PHDST Package Description. User's Manual. DELPHI 92-118 PROG 189 Rev.3
2. J.Zoll at al., The ZEBRA system. CERN Program Library Q100/Q101