

A New Query Processor for PAW

M.Ballintijn, J.Bunn, O.Couet, N.Cremel

CERN/CN, CH-1211 Geneve 23, Switzerland

This paper starts by indicating the motivation for the rewrite of the Ntuple handling part of PAW. A list of the major improvements and new features is presented as well as a discussion of the compatibility with the old system. Some ideas for future developments are outlined.

1 Interactive Analysis of Ntuples in PAW.

The interactive analysis of high energy physics event data using HBOOK Ntuples[2] has always been one of the major strengths of PAW[1]. With the design and implementation of the PIAF[4] system and the advent of Column Wise Ntuples (CWNs) the amount of event data which can be analyzed interactively has been significantly increased. Other improvements due to the use of CWNs include the support for more data types and new possibilities for structuring the data. CWNs can store boolean, bit field, integer, real, double precision and character string data types. Data can be structured in arrays of one or more dimensions, where the last dimension of each array can be of variable length. This allows for efficient storage of lists of values, lists of vectors etc. The actual length of the array in a given event is defined by an integer variable in the Ntuple.

This paper focuses on the handling of Ntuples in PAW. An Ntuple is a set of events, where for each event the value of a number of variables is recorded. Ntuples are a very convenient tool for analyzing statistical datasets. An Ntuple can be viewed as a table with each row corresponding to one event and each column corresponding to given variable. Typically the events are scattering or collision events recorded by a HEP experiment and variables can be measurements like a scattering angle, the number of tracks or the energy deposited in a calorimeter (but Ntuples are useful for the analysis of any statistical data sample). The interesting properties of the data in an Ntuple can normally be expressed as distributions of Ntuple variables or as correlations between two or more of these variables. Very often it is important to create these distributions from a subset of the data by imposing constraints (also known as 'cuts') on some of the variables. In Table 1 the main com-

Table 1:

Command	Arguments [Optional Arguments]
LOOP	IDN UWFUNC [NEVENT IFIRST]
PROJECT	IDH IDN [UWFUNC NEVENT IFIRST]
PLOT	IDN [UWFUNC NEVENT IFIRST NUPD OPTION IDH]
SCAN	IDN [UWFUNC NEVENT IFIRST OPTION VARLIS]

mands for accessing Ntuples in PAW are listed. For all commands the arguments IDN, UWFUNC, NEVENT and IFIRST can be specified. The range of events to process is specified by IFIRST, the index of the first event, and NEVNT, the number of events. The IDN argument contains the Ntuple identifier and a list of expressions separated by '%', e.g.

```
//LUN1/10.TdcVal-Tzero%iStrip
```

where '/LUN1/10' is the Ntuple to be used and 'TdcVal-Tzero' is plotted versus 'iStrip'

Depending on the command the values resulting from expressions are used in different ways but they are calculated in the same way. The UWFUNC argument is an expression which is used to further select events. If it evaluates to zero the event is skipped, otherwise the value might be used, e.g. as a weight when filling a histogram. The LOOP command is somewhat special because it just calls a user function (COMIS[5]) to process each event.

The part of PAW which evaluates the expressions and performs the appropriate actions on the values is called the Query Processor. An important property of the Query Processor is the language in which the expressions are expressed. In the rest of this paper we will show why the current system needed to be replaced and what the replacement system now offers.

2 Motivation for a new Query Processor

The current version of the system was developed a few years ago to accommodate the features that became available with the Column Wise Ntuples as described above. It also significantly extended the language which is used to formulate the queries.

Sadly enough the software version is unstable and difficult to modify. It suffers from arbitrary limits on the size of expressions and internal tables. It only handles arrays correctly in the simplest cases. Finally it understands only two types, float and string, which makes it impossible to call COMIS functions which take integer or double arguments.

The aim of the new query processor (QP) is to have a more powerful syntax, to fully support all types available in the Column Wise Ntuples and to remove most limitations.

3 New features

In this section the most important features of the new Query Processor (QP) are presented. It is maybe useful to remark that all of the old functionality is also re-implemented and that many bugs that caused erroneous input to be accepted are no longer present. It is hoped that implementation of the new processor will result in easier maintenance, and easier extensibility, and a better system for the user.

3.1 Regular syntax and full type support

The new QP supports all types available in column wise Ntuples: boolean, unsigned integer, signed integer, real and string. The integer and real types are available in multiple sizes, 32 and 64 bit, if the hardware support is available. This is normally only the case for type real. The numerical types can be combined in expressions, in which case automatic type conversions are applied. The new syntax is fully recursive and therefore allows

expressions within expressions. It is also allowed to have indices (re)calculated for each event. The rigorous checking of the syntax and types gives predictable results. In the new system most tables are allocated dynamically and grow when needed. This will alleviate the most common problem with the old system which is the overflowing of fixed length internal tables.

3.2 Shape mapping of (dynamic) arrays

A uniform principle has been adopted for the use of arrays in expressions. The operators are applied element by element, e.g. when adding two arrays they have to have the same shape, and the result is an array of the same shape where each element is the sum of the corresponding elements in the two operands. If one of the operands is a scalar, e.g. multiplying a scalar and an array then the result is again an array of the same shape as the input array with each element the product of the scalar and the corresponding element. It is important to realize how this principle works out in the case of such operators as 'less-than' and boolean 'and' and 'or'. Following the same principle the plot expressions and the weight are processed. This allows for one weight for all plotted values but also for a specific weight for each element of an array. Typical use of these features will be shown in the examples below. The QP checks the proper matching of shapes statically as far as possible. When variable length arrays are used or slices of arrays the proper matching is checked at runtime.

3.3 Compatibility

The new query processor is largely compatible with the old one as specified in ref. 3. The most important difference is the evaluation order of the selection and plot expressions, which has been reversed. The new system evaluates the selection expression first and only evaluates the plot expressions when necessary. The second difference is related to this, the internal variables VIDN1, VIDN2 and VIDN3 can no longer be supported. The use of these (scalar) variables was anyway not well defined in the case of non scalar expressions. In all cases these variables can be replaced by cuts (see below) or arguments to user functions.

3.4 KUIP vectors

The new query processor allows KUIP vectors to be used in expressions just like Ntuple variables. One of the benefits is the possibility of parameterized expressions and selections. The second benefit is an added level of indirection, an Ntuple variable can be used as an index of a KUIP array. The use of this will be illustrated in the examples.

3.5 Enhanced access to COMIS

Access to COMIS is improved in several ways. Because of the available types, a COMIS routine can be called with logical, real, integer and string arguments. Typechecking is

added to make sure the correct types are passed to the routine, the return type is used in the expression from which the function is called.

3.6 Improved Cuts

The old query processor used cuts as a macro mechanism, a textual substitution was carried out before interpreting the expression. In the new system cuts are treated as function calls. The cut expression is evaluated once for each event and the result is cached. A complex sub-expression, e.g. a call to COMIS or array slice, can now be used in both the selection and the plot expressions without being evaluated more than once.

4 Examples

The following examples illustrate new features of the new QP. The first example is fairly simple:

```
ntuple/plot 10.a%log(b) c+d>42
```

if we assume a, b, c and d to be scalar variables this will create a plot where for each event the point (a,log(b)) is plotted if the value of c plus d for that event is larger than 42. If a, b and c are matrices of dimension (3,4), d still scalar then effectively the following happens:

```
for i = 1 to 3 do
  for j = 1 to 4 do
    ntuple/plot 10.a(i,j)%log(b(i,j)) c(i,j)+d>42
  end
end
```

internally this is of course executed much more efficiently. This is the mechanism by which all non scalar values are combined in expressions. It should be noted that not all built-in functions like log will be expanded this way. It is planned to introduce e.g. vsum which will return the sum of all the elements in the array argument received. For the moment COMIS functions never expand like this. Next the use of a kuip vector is shown:

```
ntuple/plot 10.tdcVal-pedestal(istrip+100*iplane)
```

in this example tdcVal, iStrip and iPlane are scalar Ntuple variables and pedestal is a KUIP vector. For each event the index is calculated and the right element of pedestal is subtracted from tdcVal. A future extension mentioned below will allow this expression to work if all variables are vectors. In the following example cuts and COMIS functions are illustrated:

```
ntuple/cut 1 myfun.f(2*ix,sin(y))
```

```
ntuple/plot 10.x%$1 0.<$1.and.0<x<1
```

the first command defines a cut in which a user supplied fortran function myfun() is to be interpreted using COMIS, it is called with an integer and a real argument. The arguments are expressions using the ix and y variables. It is checked if myfun() is supposed to be called with these types of argument and what the return type is. When the second command is executed, the value of cut \$1 is calculated and used in the selection expression. If the selection evaluates to non-zero a point is plotted using the cached value of \$1. The

type of the value of a cut need not be a scalar (as in this example) but can be any type acceptable to the query processor.

5 Future plans

It is foreseen to add several commands to access Ntuples. The command DUMP will write the selected values to a file in a format suitable for further processing, the command VFILL will create a vector out of the selected values and finally the FIT command will allow the fitting of the selected values. The last one will be clearly the most involved to develop, but will make fitting in PAW a lot more powerful.

The following new features are foreseen for the new Query processor but might be delayed to a later release. The MASK mechanism should be better integrated with the Query Processor. In the current system masks are handled quite differently from the Ntuples, while in principle they can be seen as extensions to the Ntuple they are derived from. Internally the results of cuts could also be cached, a kind of automatic masks, to speedup the looping over the Ntuple.

The new Query Processor allows indices to be calculated dynamically. This allows e.g. an array to be used to map an integer variable to some value. This should be extended to allow an array of integers to be used as indices, producing an array of new values. This would remove one of the remaining limitations on the use of (variable) arrays. A related issue is the use of COMIS functions, which naturally only return scalar values. It should be possible to transform an array into an array using a comis function which takes one or more scalar arguments. The rules for such a mechanism need to be carefully specified.

A full performance analysis of both the new Query Processor as well as the Ntuple package should be performed to look for regions of improvement. The replacement of (part of) the RZ-package should be studied.

Acknowledgments

The primary author thanks René Brun for allowing him the possibility to work on this project and for many useful discussions. Fons Rademakers and Vladimir Berezhnoi contributed to the project by giving many useful remarks and explaining the inner workings of PAW and COMIS.

References

1. R. Brun et al. 'PAW, The Complete Reference', Program Library Q121, CERN, 1993.
2. R. Brun, 'HBOOK Users Guide', Program Library Y250, CERN, 1993.
3. CN/AS, 'A new version of PAW', in CERN Computer News Letter 210, 1993.
4. F. Rademakers, 'The Parallel Interactive Analysis Facility', CERN/CN/94/9.
5. V. Berezhnoi, 'COMIS - Compilation and Interpretation System' Program Library L120 CERN, 1993.