GSI Multi Branch System

Introduction

H. Goeringer

November 7, 1996

GSI, Gesellschaft für Schwerionenforschung mbH
Postfach 11 05 52, Planckstraße 1, D-64220 Darmstadt
Tel. (0 6159) 71-0
Chapter 1

MBS Event Serving and Filtering

1.1 PAW

1.2 (Remote) Event Server Clients

The following informations can also be obtained in WWW (http://www.gsi.de/computing/expdv/goopaw.html)

There are three different clients available providing the HBOOK data structures in shared memory (in VMS: permanent global sections) or conventionally in the user's address space:

1. A stand-alone program providing the data in shared memory,

2. A PAW module providing the data in shared memory (AIX only), and

3. A PAW module providing the data in the user's address space.

In the last two cases the client is integrated into PAW. For visualization and further analysis the HBOOK histograms and Ntuples are accessible in case no. 3 only in the current PAW session. In case of shared memory usage, however, the data can be accessed in parallel by any number of additional PAW sessions running on the same node as the client.

All clients work with the CERN software versions 94a and later. Event analysis on the client side is controlled by user written interface programs (in Fortran) defined similarly to those of the APE package:

1. uastart: create HBOOK histograms and Ntuples

2. unaal4: fill HBOOK histograms and Ntuples

3. uastop: user actions before end of analysis

4. uclinfo: user statistics routine (new version only)

The following sections describe the clients on OpenVMS and AIX.
### 1.3 PAW Clients on VMS

All you need to create your VMS client for the (Remote) Event Server is available with the logical name goopaw. For the stand-alone client collecting data in shared memory see

- **make**.gooshrccli.com : command file to create your client
- **link**.gooshrccli.com : link script called in make.gooshrccli.com
- uastarts.for, uanal4.for, uastop.for, uclinfo.for :
  - template programs for the analysis of events
  - and for statistics.

For the PAW client collecting the data in the PAW common PAWC in the user’s address space see:

- **make**.goopawcli.com : command file to create your PAW client
- **link**.goopawcli.com : link script called in make.goopawcli.com
- uastartc.for, uanal4.for, uastop.for, uclinfo.for :
  - template programs for the analysis of events
  - and for statistics.

Copy the templates to your own file system and fill them with your analysis code. See also the extensive in-line documentation in the template files. If further user modules or libraries are needed, copy also the required command files make... and link... and modify them correspondingly. Before the first usage in your session, you have to invoke the command cernlogin to specify the version of the CERN software to be used.

Note that there are two different template files containing the initialization subroutine, namely **uastarts.for** for data collection in shared memory, and **uastartc.for** for data collection in the PAW common PAWC in the user’s address space. These file names are also used in the corresponding command files creating the executables. However, the name of the Fortran subroutine must be **uastart** in both cases.

In the shared memory version, the client is a standalone program without PAW environment. For visualization and further analysis the HBOOK histograms and Ntuples filled in the global section can be accessed in parallel by any number of PAW sessions running on the same node. If the client is invoked repeatedly, the HBOOK data structures are initialized each time in the template procedure uastart thus loosing the previous contents.

In the version with the client integrated into PAW, the HBOOK histograms and Ntuples are accessible only within the current PAW session. Here the HBOOK data structures are initialized by default only once, so that all events received from all client invocations may be accumulated. However, optionally the data structures may be initialized for each call of the client.

**WARNING**: The current VMS releases of the Cern software do not support writing to global sections within PAW. Therefore don’t use the shared memory initialization in PAW, else your PAW session will be corrupted after the first invocation of the client!
1.3.1 PAW as Client for the Event Server in VMS

Having prepared the three Fortran user interfaces `uastartc.for`, `uanal4.for`, and `uastop.for` in your local directory, your PAW executable containing the client for the GOOSY Event Server will be created and invoked with

```
cernlogin
@gopaw:make_goopawcli
goopawcli
```

Within PAW, the client will be invoked with the command

```
PAW > /gsi/goocli/input parameter-list
```

For the parameter-list see the AIX man page `goocli(1L)` or the on-line help information available in your client PAW with

```
PAW > help /gsi/goocli
```

Here several help menus are offered providing information on

- the command interface of the client,
- the event filter (including extensive examples),
- the user interfaces for event analysis control,
- and some general informations, including some hints on the client version writing to shared memory.

The HBOOK histograms and Ntuples, as booked in `uastart` and filled in `uanal4`, are available only in the common block PAWC in the current PAW session.

WARNING: The current VMS releases of the Cern software do not support writing to global sections within PAW. Therefore don't use the shared memory initialization from the template file `uastarts.for` in PAW. Else your PAW session will be corrupted after the first invocation of the client!

If the client is running, `CTRL-g` invokes the call of a user modifiable (FORTRAN) subroutine `uctinfo` that displays statistical information. The client can be terminated with `CTRL-a`.

1.4 PAW Clients on AIX

The (Remote) Event Server Clients in AIX

All you need to create an AIX client for the Event Servers is available in the directory
1. \texttt{/cern/goopawnew} (new version), or
2. \texttt{/cern/goopaw} (pro version).

The Fortran template files should be copied from there and filled with your analysis code. If you need not add further program modules, you can invoke the make files from these directories directly, e.g. specify

\texttt{make -f /cern/goopawnew/gooshr.make}

in the local directory containing your Fortran sources to create the new version of the stand-alone shared memory client. The following templates and make files are needed for the three types of clients:

1. stand-alone client collecting data in shared memory: gooshr.make: make file \texttt{uastarts.f}, \texttt{uana14.f}, \texttt{uastop.f}, \texttt{[uclinfo.f]} template programs

2. PAW client collecting data in shared memory: goopawshr.make: make file \texttt{uastartc.f}, \texttt{uana14.f}, \texttt{uastop.f}, \texttt{[uclinfo.f]}

3. PAW client collecting data in the user's adress space: goopaw.make\texttt{uastartc.f}, \texttt{uana14.f}, \texttt{uastop.f}, \texttt{[uclinfo.f]}

Only by the new version of the make files (in \texttt{/cern/goopawnew}) supports \texttt{uclinfo} as user interface. Note that there are two different template files containing the initialization subroutine, namely \texttt{uastarts.f} for the stand-alone client, and \texttt{uastartc.f} for the two PAW clients. These file names are also required in the corresponding make files. However, the name of the Fortran subroutine must be \texttt{uastart} in all cases.

The make files check and handle the version of the CERN software as specified with the \texttt{cernlogin} command. All three AIX clients work with the versions 9.4a and later of the CERN software. Without shared memory, the HBOOK histograms and Ntuples are accessible only within the current PAW session. The data structures created and filled in shared memory, however, can be accessed for visualization and further analysis in parallel by any additional number of PAW sessions running on the same node.

A PAW executable can be created with only one of the two clients integrated.

For more information see

1. The Shared Memory Clients in AIX, 3. the extensive in-line documentation in the template Fortran files in \texttt{/cern/goopaw[new].}

If the stand-alone client is invoked repeatedly, the HBOOK data structures are initialized each time thus loosing their previous contents. In case of the PAW clients, by default the HBOOK data structures are initialized only once, so that all events received from all client invocations are accumulated. However, optionally the data structures may be initialized also for each call of the client.
1.4.1 Event Server Clients integrated into PAW

Having prepared the three Fortran user interfaces uastartc.f, uanal4.f, and uastop.f in your local directory, and having specified the cernlogin command

. cernlogin

the PAW executables containing the clients for the GOOSY Event Server will be created and invoked with

1. make -f /cern/goopaw/goopawshr.make
   goopawshr

2. make -f /cern/goopaw/goopaw.make
   goopaw

In the first case, the client writes to shared memory, whereas in the second case, the client writes only to the local PAWC common. The client will be invoked in both cases with the PAW command

PAW > /gsi/goocli/input parameter-list

For the parameter list see the AIX man page goocli(1L) or the on-line help information available with

PAW > help /gsi/goocli

Here several help menus are offered providing information on

the command interface of the client, the event filter (including extensive examples), the user interfaces for event analysis control, and some general informations, including some hints on the client version writing to shared memory.

1.4.2 Shared Memory Clients in AIX

Having prepared the Fortran user interfaces in your local directory, the stand-alone event server client will be created and invoked with

. cernlogin
make -f /cern/goopaw[new]/gooshr.make
gooshr parameter-list

For the parameter list see the AIX man page gooshr(1L)

Similarly the PAW executable containing the integrated client will be created and invoked with:
. cernlogin
make -f /cern/goopaw[new]/goopawshr.make
goopawshr
Paw > /gsi/goocli/input parameter-list

For the parameter list see the AIX man page goocli(1L).

In both cases the client allocates shared memory segments (if not already existing) containing
the HBOOK histograms and Ntuples as booked in uastart and filled in uanal4. You may access
these data from any number of (standard) PAW sessions running on the same host (see Accessing
Shared Memory with PAW).
**F_CLIPAW**

```c
f_clipaw(pc_hostn, pl_portn, pl_nbevt, pc_filtr,
          pl_sampl, pl_echo, pc_tyevt, pl_bflsh)
```

**PURPOSE**

Client to connect GOOSY - PAW - Server. (See also MGOOPS)

**ARGUMENTS**

(see also Description Function)

- **pc_hostn**
  - (char *) Node_name

- **pl_portn**
  - (long *) Port_number > 0
  - or
  - by using the port server

- **pc_hostn**
  - ptr to Server_name or Node_name:Server_name

- **pl_portn**
  - ptr to Port number <= 0

- **pl_nbevt**
  - (long *) Number of events
    - >0: Request number of events from the server
    - =0: Client will be started with the given parameters but no connection established
    - -1: Request an unlimited number of events

- **pc_filtr**
  - (char *) Filter option or File name
    - A: Select all events.
    - F: Filter condition will be prompted.
    - file: File with filter specifications.

- **pl_sampl**
  - (long *) Reduction rate
    - 0,1: Get every event when filter match.
    - n>0: Every n'th event when filter match.

- **pl_echo**
  - (long *) Echo rate
>0 Notify after every n’th event.
0 Disable echo.
n<0 Notify every n percent of requ. events

\textbf{pc\_tyevt} (char *) Display option
\begin{itemize}
\item \textbf{X} Switch off analysis routine UANAL.
\item \textbf{N} No display
\item \textbf{H} Display only event (and subevt) Header
\item \textbf{D} Display evt/seq Header and Data
\end{itemize}

\textbf{pl\_bflush} (long *) Buffer flushing time (timeout = *10)

\section*{Description}
\textbf{FUNCTION}
Connect to a running Server (MGOOPS), send filter condition and request event data. After each event a user analysis routine is called (UANAL) The client connects to an existing server on a VAX-VMS node. The Portnumber of the already running server must be known. The server can be started from a GGOSY environment with a running transport-manager ($TMR$) process:
\begin{verbatim}
G>CREATE PROCESS GPS $GPS
G>START SERVER
\end{verbatim}
(See VAX/VMS Help items MGOOPS, $PS\_SERVER$ and $PS\_PROC$ for further information). The starting server returns his port number and automatically reports his node, portnumber and name to a port server.

To connect a client to a server, you need to specify the node and the port number. Alternatively and to simplify the use, you may specify only the name of the server and port=0. (In case of ambiguity, i.e. several servers with the same name on different nodes, you must specify node:name). The client gets node and port from the port server and connects automatically.

The client requests a number of events (or a continuous stream) from the server. A strong selection on the events may be applied by specifying filter conditions.

Samples of events that match the filter may be taken.
The reception of events may be echoed.

The display of the event (and subevent) header or the full data content may be selected. With the same parameter the user analysis UANAL may be switched on/off.

The server flushes the buffers to the client in a selectable time interval. Even when the server gets no events from his input, or none of the events match the filter, the client gets a information buffer regularly.

Statistical information about the server and the client, i.e. read/written bytes, processed buffers, processed events, filter matching events etc. are contained in every buffer sent by the server.

CTRL\_g invokes the call of a user modifiable (FORTRAN) subroutine UCLINFO that e.g. displays these information. (Response time on CTRL\_g <= buffer\_flushing\_time)

CTRL\_a terminates the client. (Response see above)

For every incoming event, the user written (FORTRAN) analysis routine UANAL is called. UANAL may also be skipped.

Filter\_description

Filter criteria for event selection  Filters may be applied on the event (i.e. event header in the case of event type=10) and/or on subevents (if there are).

Several filter may be defined with logical conditions between them. Filter specifications may be grouped in so called blocks with logical conditions between them. Each block applies on a different region of the event (e.g. 1st block on event header, trigger number etc., 2nd block on subevent a etc. ...). In case of a subevent, one must unambiguously define in the first entry of the filter block on which subevent the block shall be applied (by giving e.g. the processor id).
Output selection  The server may send the whole event that has fulfilled the filter criteria or only parts of it. This selection is independent of the filter. E.g. in the filter, the trigger number (event) and a pattern in the subevent b is checked, but only the subevent a and c will be send to the client. The output selection is the first entry of complete filter.

Detailed filter description  In order to maintain software performance, all filter specifications (see topic) are internally translated into bit mask and offset. The mask and offset may be word or longword aligned. This makes necessary that some rules for the filter definitions have to be respected:

The required order is:
1. output selection
2. filter selection
   a) event specific filter
   b) subevent specific filter
Negligible disregard of these rules are automatically corrected and result in a warning message.

A complete filter consists of
1. block with the output selection and
2. one or more blocks with filter criteria.
In some cases, you need only one block for output selection and filter criteria (see topic examples).

Defining a block:
File input and interactive input are identical!
1. length of block, i.e. number of filter entries
2. filter entry(ies)

The filter specification consists of 9 entries (see filter specification).
1: Select filter or output selection for event (1) or subevent (0).
2: Select filter specification (1) else (0).
3: Select output specification (1) else (0).
4: Operation code: see Filter specification
ALL,IDENT,ANY,INCL,EXCL,LT,GE:
filter result = mask opcode object
(the 'object' is the word or longword
at the position 'offset' in the
event data)
5: Logical link between filter
specifications in a filter block:
  0: OR 1: AND
6: Logical link between filter blocks:
  0: OR 1: AND
7: Filter specification:
   0: Take all
   1: trigger
   2: pattern and offset
   4: type
   8: subtype
  12: subtype & type *)
  16: pro cid
  32: contr & subrate *)
  48: contr & subrate & pro cid *)
*) byte/word sequ. from left to right
8: Mask (=bit pattern)
9: Offset

**Filter specification**

filter specification
1.evtsev 2.selflt 3.selwrt 4.opc 5.lnkfl 6.lnkf2
7.fltspec 8.mask 9.offs

1. Select event/subevent
   1 event
   0 subevent

2. Select filter
   0 off
   1 on

3. Select write
0          off
1          on

4.  Object code \([\text{Res} = \text{object opcode mask}]\)
    0          \(!!\) (\text{ALL})
    1          \(==\) (\text{IDENT}) \([\text{object} == \text{mask}]\)
    2          \&\& (\text{ANY}) \([\text{object} & \text{mask}]\)
    3          \&= (\text{INCL}) \([\text{object} & \text{mask} == \text{object}]\)
    4          \(^=\) (\text{EXCL}) \([\text{object} & \text{mask} == \text{mask}]\)
    5          \(<\) (\text{LT}) \([\text{object} < \text{mask}]\)
    6          \(>=\) (\text{GE}) \([\text{object} >= \text{mask}]\)

5.  Logical link between filters in a filter block
    0          OR
    1          AND

6.  Logical link between filter blocks
    0          OR
    1          AND

7.  Filter specification and validity
    0          Take all
    1          trigger
    2          pattern and offset
    4          type
    8          subtype
    12         subtype & type *
    16         procid
    32         contr & subcate *
    48         contr & subcate & procid *
        *) byte/word sequ. from left to right
    0 - 12     valid for events
    2 - 48     valid for subevents
8. Mask (bit pattern)
   decimal or hex (0x....) enter here the
   Word or LongWord (see 9.) req. value for 7.

9. Offset
   decimal or hex (but like 8.)
   enter here the required
   value for 7.(2)
   otherwise 0 (will be set automatically)

Definition (see also 8.):
LW: offset >= 0 index on event or subevent
   (0: 1st LW, ..., etc.)
W: offset < 0 index on event or subevent
   (-1: 2nd W, ..., etc.)

Filter_examples

Filter Examples for interactive or file input
! and /* are allowed comment declarations!

1.a) Output: whole event.
   Filter: Take all events with trigger >= 3
   !
   ! output selection
   1 ! block with 1 filter
   1 0 1 0 0 0 0 0
   ! filter selection
   1 ! block with 1 filter
   1 1 0 6 0 0 1 3 0

1.b) is identical with 1.a)
   !
   1 ! block with 1 filter
   1 1 1 6 0 0 1 3 0
2.a) Output: whole event.
   Filter: Take all events with
   trigger = 3 OR
   trigger = 7
   AND
   the first three bits set
   in the 15th LongWord of
   subevent (processor id=20)
   1 ! output selection
   1 0 1 0 0 0 0 0 0
   2 ! event filter selection
   1 1 0 1 0 0 1 3 0
   1 1 0 1 0 0 1 7 0
   2 ! subevent filter selection
   0 1 0 1 1 1 16 20 0
   0 1 0 4 1 1 2 7 15 ! mask and offset decimal

2.b) Filter like 2.a) but
   Output subevent (processor id=20)
   and subevent (processor id=30)
   2
   0 0 1 1 0 0 16 10 0
   0 0 1 1 0 0 16 20 0
   2 ! event filter selection
   1 1 0 1 0 0 1 3 0
   1 1 0 1 0 0 1 7 0
   2 ! subevent filter selection
   0 1 0 1 1 1 16 20 0
   0 1 0 4 1 1 2 7 15 ! mask and offset hexadec.

User_routines

These routines have to be provided and linked
together
UANAL
Module UANAL.FOR
CALLING UANAL(I4EVT, I2STS, I4LEN)
PURPOSE User analysis routine. Here, histogramming etc. has to be done.
PARAMETERS
  I4EVT INTEGER*4 I4EVT(0:I4LEN) event vector (Longwords)
  I4LEN INTEGER*4 I4LEN data length in L.W
  I2STS INTEGER*2 I2STS Return status (0: success)

UCLINFO
Module UCLINFO.FOR
CALLING UCLINFO(I4BUF1, I4BUF2)
PURPOSE User info routine. Will be executed after the first and the last data buffer sent from the server and for each buffer with a CTRL-g keyboard input.

  For details see UCLINFO.FOR template and GOOCINC:S_CLNTBUF.H for available info and statistics data.

PARAMETERS
  I4BUF1 INTEGER*4 I4BUF1(0:* ) and
  I4BUF2 INTEGER*4 I4BUF2(0:* ) ptrs to server and client info data

Implementation
PROCEDURES see PC.PROC
STRUCTURES see PC.PROC Structures
MACROS see PC.PROC Macros
Return type none
File name PC_CLIPAW.C
Version 1.01
Author R.S. Mayer
Last Update 14-Jan-1993

Internals
Utility
Module name F_CLIPAW
File name PC_CLIPAW.C
Home direct. TOOL$SOURCE
Compile lib. TOOL$LIB:PC_GPS.TLB
Link option SYS$LIBRARY:UCX$IPC.OLB/LIB SYS$SHARE:VAXCRTL.EXE/SHARE
Created 01-Sept-1993

Updates
Updates Date Purpose
15-Dec-1993 Input, output buffer structure changed
s_clntbuf_swap obsolete (RSM)
10-Jan-1994 Documentation
02-Feb-1994 Include names and defines modified!!!
03-Feb-1994 New STC-routines
17-Feb-1994 Handling read error (RSM)
25-Feb-1994 UCLINFO reinserted! (RSM)
15-Mar-1994 Server endian, adapted to new stc, RSM
Contents

1 MBS Event Serving and Filtering ................................................................. 1
  1.1 PAW ........................................................................................................... 1
  1.2 (Remote) Event Server Clients ................................................................. 1
  1.3 PAW Clients on VMS ................................................................................. 2
    1.3.1 PAW as Client for the Event Server in VMS ................................. 3
  1.4 PAW Clients on AIX .................................................................................. 3
    1.4.1 Event Server Clients integrated into PAW .................................. 5
    1.4.2 Shared Memory Clients in AIX ......................................................... 5
  F.ClIPAW ........................................................................................................ 7

Index ............................................................................................................. i