DH IO Modules Project Status Report.

F.Ratnikov, RUTGERS

Introduction

- DH IO Modules project was started in the very beginning of Y2K
- Immediate problems:
  - need direct access to events to keep runsections compact on the output
  - need chain like structure of input and output stream - data branch lifetime is spread over many sequential files - branch driven design
- Framework:
  - was design for essentially sequential access
  - IO was designed as file driven - SeqRootDiskFile
- To fulfill CDF data convention and DH requirements necessary functionality was included into DHMods

RootFileStream

- Interface to ROOT was designed in the spring 2000
  - accept relation *Object ↔Branch name*
  - automatically reconnect object to corresponding ROOT branches when new I/O file is openned
  - support fast (Tbuffer) mode of event read/write
  - is essentially multibranch implementation
  - Pure ROOT interface, no relations with Edm or Framework
- CdfRootFileStream was inhereted from RootFileStream and knew about some Edm details.
- Possibility of multi-branch CDF event structure was discussed that time, conclusion was: it is not necessary
  - multi-branch capability of RootFileStream was used as a particular case of single branch

EventInfo

- External source of information about event is required for Tbuffer access mode. This mode is essential for the FARM concatenator operation
- EventInfo is a class containing information about event that is necessary for I/O module operation without access to the event information (currently run#, rs#, event#, record type, rs range for ERS record)
- This information was naturally put into separate ROOT branch
- EventInfo branch is essentially the “primary” branch.
  - DH IO modules just deliver data of EventRecord from/to corresponding branch as a particular case of any other object and anther branch
  - User is able to associate another object with another branch

AppRootOutputModule

- The AppRootOutputModule is a base module class providing user possibility to define new branch in the event and associate it with any object
- It is implementation for output - no input implementation
- It guarantees synchronization between event branch and user branch
- It guarantees branch is created in only datastreams connected to the data processing path containing this module.

Design of the DH IO Modules Project

- Project has essentially modular structure
  - well defined and well separated interfaces to
    - Edm
    - DFC
    - DH
    - ROOT
- Project has all the necessary hooks to handle multi-branch event structure properly
- Due to lack of the necessary Framework functionality modules perform many tasks not specific to DH itself

DHInput Functionality: Select Input Data List

- Select data by any combination of dataset or fileset or file names
- Full “include” and “exclude” support
- Extra restriction on required run# and runsection# can be applied
- Access both DH data and local private files
  - Accept wildcards for local files
- 100% compatibility with FileInputModule
- Communication with DFC to obtain full list of requested data
- Communication with DH to deliver requested data in the most effective order

Direct Acess to Events

- Process events in natural order, build catalog of events in the file (necessary for correct output file production) FW
  - Fast operation using EventInfo branch (any data except RAW)
  - Slow operation using LRIH information (RAW data)
- Navigation in the file
  - Skip events forward and backwards FW
  - Direct access to the event by run#/event#
  - Inserting necessary BOR records when run# is changed

Fast Copy (Concatenator Mode)

- Read/Write events by ROOT buffers without expanding to separate objects
  - Speed up IO bandwidth by factor of 5
  - Is necessary for concatenating FARM output files

Filtering events

- Input events can be filtered by run# and event#

Output Module Functionality

- Specify output by file name or by dataset name DH
- Assign data file name according to the CDF convention DH
- Collect statistics for the output file FW
- Collect output files in given directory FW
- Can put FILE record into DFC DH
- Split output data into files of given size FW
- Keep runsections compact in the file FW
- Intermediate save the file status to minimize reprocessing in case of job crash FW
- Creates new ERS records when EmptyRunsection condition is detected FW
- Can create many data branches synchronized with primary data branch FW

Crash Recovery

- The goal is to continue data processing after the job has crashed due to any reason
  - with minimal reprocessing overlap
  - keeping DFC consistent at any time

DH

- Sophisticated procedure is developed
- It is semiautomatic
  - The close coordination between Input and Output is required to make procedure mostly automatic
  - Automatic procedure can be implemented on the Framework level where coordination of Input and Output is possible

FW

Still Missed

- Clean up of output from obsolete BOR records
  - several BOR for the same run
  - BOR for the run which all events were filtered out
- Minor issues of current user requests

Conclusions

- **DH IO Modules project has successfully reached all the original goals and satisfy to all project requirements**
- It fulfill to mutually contradictory requirements having high performance for FARM operation and being flexible for user convenience
- Support of the modules mainly includes
  - following changes in the interfering projects (DH, ROOT)
  - satisfying user requests making modules more user friendly
  - keeping documentation up to date
- DH IO Modules include hooks necessary to support multi-branch event structure
- Multi-branch Modules project has essentially different design and structure
- Framework related functionality of the DHMods should be supported by Framework for the new project
- DH specific functionality of the DHMods can be easily moved into new project using original DHMods interfaces