HDF5 dans ma vie…

et dans LSST ?

G.Barrand, CNRS/IN2P3/LAL

HDF5, c'est quoi ?

- du soft dédié pour écrire/lire des données de type scientifique dans un fichier ! Bref qui adresse le "problème de l'IO pour nous".
- (Et donc pas fait, au départ, pour stocker des profils Facebook ou gérer des comptes bancaires)
- Pour l'historique, "qui il y a derrière", l'API, etc... voir le web.
- Donc fait par des scientifiques pour des données de type scientifique. (Il y a même un API FORTRAN ☺)

- (ne pas confondre HDF5 et HDFS...)

Premier contact en HEP, années 2000

- AIDA : (années 2000) : initiative pour définir un API multi langage (C++, java, Python) pour l'analyse de données en HEP. (LAL (1), SLAC (3), CERN (1))
- IO : format de fichier commun : XML. Ok pour des histos mais...

  besoin d'un format binaire, en particulier pour les gros tableaux.
- format ROOT : LAL/Rio C++ : librairie légère d'écriture/lecture pour ce format. SLAC/Freehep idem en java.
- osc-batch avec « root driver » utilisé au travers de Geant4.
- Mais travail pénible de refaire une implémentation pour un format quasiment non documenté. (La seule doc étant les internals de ROOT ! ☹)
- On cherche autre chose…

HDF5

- HDF5 : bon candidat ! Un API java existait déjà.
- écriture d'un driver HDF5 pour écrire/lire les objets HEP standards (histos, profiles, clouds, fonctions, fits, ntuple). Ça marche !
- Mais en fait SLAC/AIDA n'a pas accroché… Ils auraient voulu un système d'IO avec une implémentation en java.
- osc-batch « hdf5 driver » n'a probablement jamais été utilisé pour cause de domination du format ROOT en HEP. (Malgré le fait qu'on pouvait lire les fichiers depuis un prompt ROOT ! Si,si)

HDF5 / ROOT-IO

- HDF5 est un bon package d'IO pour du scientifique (en particulier il n'est pas « intriqué » avec d'autres choses).
- Performances (vitesse, taille de fichier) sur des histos, tableaux, images, aussi bonnes, voir légèrement meilleures que ROOT/IO. Côté vitesse de toute façon on est dominé par l'accès disque !
- Il y a aussi de la compression « zlib » qui est la même et donc le dilemme « taille de fichier versus vitesse » est le même.
- On peut aussi organiser ses données « à la file system » dans un fichier (H5group = TDirectory).
- Astro : pas de problème pour stocker des images et le genre de données mises dans du .fits. Sur un problème de découpage pyramidal de grosses images, les trois formats (fits, root, hdf5) donnent des perfs très similaires.

HDF5 et l'OO

- Il n'y a pas d'outil d'instrumentation automatique de classe C++ utilisateur venant avec la librairie. Il faut écrire son code « adapteur » soi-même. (Forcément là, les ROOTeux se marrent).
- Pour des gens travaillant sur des gros « event model à la HEP » cela peut être pénalisant. Mais pour des gens ayant des « data schéma » figés du genre histos, images, tableaux, ce n'est pas grave. (Puisque on écrit le code « adaptateur » une fois).
- Plus subtil : il n'y a pas non plus d'écriture automatique du « data schema » (de la « sémantique des données ») dans le fichier (l'équivalent des « streamer infos » mis en fin de fichier chez ROOT).
- Mais ce n'est pas rédhibitoire : rien n'empêche de se faire les outils pour ! (Et ce d'autant plus depuis un langage OO ayant de l'introspection).

La suite…

- AIDA meurt au CERN et à SLAC. (Et où on se sent bien seul ☹)
- Geant4 révise sa copie pour les « analysis tools » et veut avoir un « minimum » (histo, ntuple et IO) « simple » qui vient avec Geant4 (no third party tools to install).
- 2010 : iPad & « smart devices » : nouvelle génération de mon soft (C++ et GL-ES) avec un nouveau graphique et aussi une révision de ma copie autour des histos, ntuple, IO : softinex.
- inlib/root : code « pure header » (si,si) : jamais été aussi facile de lire et écrire un histo au format ROOT ! (Linux, MacOS, Windows, iOS, Android).
- Partie de softinex => g4tools embarqué dans Geant4 depuis 2012.
- Mais du coup, dommage collatéral, HDF5 disparaît du paysage.

HDF5 & SOPHYA

- Et où HDF5 réapparait dans mon paysage… ☺
- SOPHYA = Réza & All.
- Recyclage du code osc-batch « hdf5 driver » dans inlib/exlib.
- Write/read SOPHYA::Histo : pas de problème.
- Write/read SOPHYA::Ntuple (all in memory) : pas de problème.
- Plus délicat, le SOPHYA::DataTable, un tableau-paginé…

Le tableau/ntuple paginé

- C'est très HEPien…
- HEP a besoin, du fait de la nature intrinsèquement statistique de sa physique (Heisenberg, Dieu, les dés, etc...) de structure de type tableau mais avec ÉNORMÉMENT d'entrées. Tellement que cela ne tient pas en mémoire. (Par exemple, sortie de simulation à la HEP...)
- On introduit « un machin », le « ntuple », fonctionnant de facto avec un système de stockage. Première version de la chose dans CERN/PAW et nettement améliorée dans CERN/ROOT avec le TTree.
- En gros : lorsqu'on remplit une colonne du tableau/ntuple, en fait on remplit une « page de données » (un TBasket chez ROOT) qui, une fois pleine, est écrite automatiquement dans le fichier attaché à la chose. Idem en lecture, les pages sont importées automatiquement lorsqu'on lit une colonne : l'idée est élégante.
- inlib/root a maintenant cela, et le SOPHYA::DataTable de Réza fait cela aussi.

SOPHYA::DataTable sur HDF5

- SOPHYA::DataTable : tableau-paginé avec du FITS.
- On fait la version HDF5 : il a « suffit » de faire une émulation HDF5 des points d'entrée de cfitsio utilisé par la pagination.
- Pour la pagination on utilise un « dataset » par colonne et l'ajout d'une page au write se fait par un « append » d'un « hyperslab chunké » (si,si) (terminologie HDF5 !).

- (du coup on a du code qui fait du HDF5 mais avec l'API de cfitsio ! Cela peut reservir…)

HDF5 & Geant4

- La collaboration Geant4 est d'accord pour faire du HDF5 (en parallèle au format ROOT et csv) au travers de l'outillage d'analyse g4tools.
- Write/read g4tools::h[1,2,3]d, p[1,2]d : en cours…
- Write/read de ntuple-paginé, avec si possible le même API existant que pour csv et ROOT : en cours aussi…
- Bien sûr on va essayer d'utiliser une même base de code (softinex) que pour SOPHYA.
- Il faudra aussi probablement montrer qu'on peut lire les fichiers depuis CERN-ROOT. (☹)
- Release prévue pour la fin de l'année…

Parallélisme

- multi-cores, multi-machines => besoin d'IO adapté.
- Geant4 : besoin d'avoir un seul fichier en écriture pour plusieurs threads et/ou plusieurs machines.
- softinex/g4tools 2014 : on passe des histos par MPI.
- 2016 : ntuple-paginé pour le format ROOT : un ntuple attaché à un seul fichier dans un thread/rank ; avec le même « booking » plusieurs threads ou machines font des « fill row » qui remplissent les pages (baskets) qui sont envoyées au « thread/rank gérant le fichier ». Ça marche !
- Faire la même chose avec du ntuple-paginé-HDF5.
- Mais HDF5/1.8 a des couches « parallèles » qu'il faut regarder…

Conclusions

Pour un format binaire, HDF5 est un bon package d'IO pour du scientifique.

- Côté OO il manquerait un outillage d'instrumentation de classe pour convaincre « techniquement » un HEPien face à son « event model ». Geant4/analysis-tools pourrait lui montrer que la voie HDF5 est viable pour son « vade-mecum » analytique.
- Astro & LSST : ça marche. Mais la force du FITS tient à son universalité, pourquoi changer ? Pour cause de scénarios de parallélisme ?
- MAIS, pour un format binaire adapté à du scientifique il faut surveiller ce qui se passe du côté web et le « big data », un outsider pourrait émerger de cette activité « buzzwordesque »… (Spark/ Avro ?)