

ESTUDIO COMPARATIVO DE CANDIDATOS A DPD (Derived Physics Data)

R.Vives, E.Oliver, M.Villaplana

Mayo 2007

Contents

1	Introducción al Event Data Model de ATLAS	1
2	Características generales y estrategias para el análisis de datos	2
2.1	Características del análisis usando Athena	2
2.2	Características del análisis usando Root	2
3	Análisis pormenorizado de cada tipo de formato de datos candidatos a DPD	2
3.1	AOD: Análisis Object Data	2
3.1.1	Características	2
3.1.2	Ventajas	2
3.1.3	Inconvenientes	2
3.1.4	pAOD: Transient/Persistent AOD	3
3.1.5	¿Qué da lugar a las AANT?	3
3.2	AANT: Athena Aware Ntuple	3
3.2.1	Características	3
3.2.2	Ventajas	3
3.2.3	Inconvenientes	3
3.2.4	¿Qué da lugar a las SAN?	4
3.3	SAN: Structured Athena Aware Ntuple	4
3.3.1	Características	4
3.3.2	Ventajas	4
3.3.3	Inconvenientes	5
3.3.4	Consideraciones prácticas	5
4	Conclusiones	5

1 Introducción al Event Data Model de ATLAS

El Event Data Model de ATLAS (EDM) estructura los datos en una serie de niveles de detalle según sea el uso que de ellos se pretende. Por ejemplo, los Event Summary Data (ESD) almacenan la información que viene de los algoritmos de reconstrucción, datos sobre el calorímetro y sobre el tracking system que permiten tareas de calibración y alineamiento, pero sólo son accesibles en ciertos lugares (Tier 0) debido a su tamaño. Por otro lado, los Analysis Object Data (AOD) contienen una parte más pequeña y concisa de la información

pero son más accesibles para el análisis. En el siguiente nivel los analistas deberían poder trabajar con datos que permitan una mayor personalización y movilidad, datos que sea posible analizar posteriormente dentro y fuera de Athena, son los Derived Physics Data (DPD).

2 Características generales y estrategias para el análisis de datos

En gran medida la dificultad existente para definir el tipo de formato idóneo para los Derived Physics Data estriba en la divergencia entre la filosofía de programación y análisis usando Athena y la filosofía propia de Root.

2.1 Características del análisis usando Athena

Athena representa el aspecto más colectivo del análisis en ATLAS. Por eso está tan estructurado y tan lleno de patrones, clases y objetos preestablecidos, pues pretende aportar todo aquello que para los analistas estará en común, evitando idealmente replicación de esfuerzos, compatibilizar versiones y ahorro de tiempo para el desarrollo de programas (si bien se está viendo como el tiempo que se ahorra está siendo empleado en aprender a utilizar estos recursos).

2.2 Características del análisis usando Root

Root representa la simplicidad, el aspecto más individual del usuario que desarrolla su propio código a su manera e independientemente de toda la gran maquinaria de ATLAS. Es un poco más caótico, más libre, fruto más de la experiencia de muchos usuarios que lo han ido enriqueciendo, sin una planificación tan detallada. Los nombres de sus objetos y métodos a menudo no son tan coherentes como en Athena y puede haber funcionalidades duplicadas, pero es muy útil porque funciona y es robusto pues es fruto de la experiencia.

3 Análisis pormenorizado de cada tipo de formato de datos candidatos a DPD

3.1 AOD: Análisis Object Data

3.1.1 Características

Los AOD habrá que mantenerlos para permitir análisis más complejos, especialmente los que se basen en herramientas de Athena y deban acceder a *conditions data*, *back navigation* o *schema evolution*.

- Tamaño por evento: 100 kb.
- Contiene estructuras complejas.

3.1.2 Ventajas

- Puede replicar la estructura de nuestro Event Data en Root.
- Tiene las mismas clases para ESD/AOD/DPD. Gran unificación. Muy atractivo.
- Completamente accesible en Root.
- En pAthena, el formato actual del AOD como DPD funciona.

3.1.3 Inconvenientes

- No es Athena-aware en el sentido del TAG. (se cree que se puede añadir esta funcionalidad pero todavía no se ha demostrado).
- El usuario necesita las clases AOD y el diccionario. (desde la Release 13 los diccionarios se cargan automáticamente).
- Hay que correr Root dentro de Athena (el formato AOD no es compatible en Root Stand-alone; incluso en el caso de un portátil con la misma versión de Linux y el mismo compilador usado en Athena, el usuario no puede copiar y usar con éxito las librerías compartidas).
- Restricción muy severa a slc4.
- El formato actual del AOD como DPD no consigue generar las librerías requeridas en los *'worker nodes'*.
- El formato actual del AOD no funcionaría como DPD.

3.1.4 pAOD: Transient/Persistent AOD

Esta separación diferencia entre el AOD que se usa para almacenamiento (*persistent*) y el utilizado para análisis (*transient*). El *transient AOD* contiene datos y métodos mientras que el *persistent AOD* sólo contiene los datos. Durante la lectura de los datos se realiza una conversión de *persistent* a *transient* de forma que el usuario sólo ve el *transient AOD*. Esta separación permite almacenar los datos de forma eficiente y leerlos rápidamente. También permite *'schema evolution'* (habilidad para leer datos antiguos cuando los AOD/ESD ya han cambiado).

3.1.5 ¿Qué da lugar a las AANT?

El análisis de datos sobre un AOD da como salida un fichero Root que no se puede reanalizar con Athena. Si hay algún error o se quieren añadir o modificar variables antes de pasar a Root, hay que volver a compilar y correr el programa de análisis y debido al tiempo necesario para ello eso resulta muy poco versátil.

3.2 AANT: Athena Aware Ntuple

3.2.1 Características

Athena Aware NTuple (AANT) tiene un TTree plano y referencias a datos de los eventos como RDO, ESD y AOD. El TTree es leíble en Root, lo cual procura un acceso rápido a algunas cantidades para un *fast analysis*. El *full analysis* se hace con Athena. Los usuarios pueden acceder a los datos de los eventos utilizando las referencias. De esta manera, AANT proporciona un entorno consistente tanto para *fast* como para *full analysis*.

- Tamaño por evento: 56 kb.
- Contiene tipos simples de datos.
- Se puede producir en Reconstruction como CBNTAA y en análisis sobre AOD/ESD (con EventView, HightPt View, TopView, etc)
- Nunca desaparecerán pues siempre serán útiles para sacar los resultados finales de un análisis y para exámenes rápidos y simples de datos.

3.2.2 Ventajas

- Es Athena-aware en el sentido del TAG. (*'Event selection'* desde el AANTTree y *'back navigation'* hasta el original AOD, ESD y RDO)
- El usuario puede copiar las AANT a su portátil y, con sólo tener Root instalado, analizarlas a su manera.

3.2.3 Inconvenientes

- No puede replicar la estructura de nuestro Event Data en Root.
- No se puede releer en el Athena Event loop.

3.2.4 ¿Qué da lugar a las SAN?

Las Structured Athena-aware Ntuples (SAN) fueron introducidas con la intención de tender un puente entre las AANT y los AOD. Si pudieran guardarse, por ejemplo, objetos como el *Electron* de los AOD en una *branch* de AANT, el código de análisis sería más fácil de desarrollar, mantener y, posiblemente, podría compartir las herramientas de análisis de los AOD. Lamentablemente, debido a la complejidad de la clase *particle*, la cual se usa en la gran mayoría de AOD, actualmente es imposible usar un *Electron* de AOD almacenado en una *branch* de AANT. Por esta razón un *Electron* de SAN, aún compartiendo muchas de las características de su homónimo en los AOD, no implementa todos sus aspectos.

3.3 SAN: Structured Athena Aware Ntuple

3.3.1 Características

SAN tiene un papel importante y debería mantenerse como una Root-Tupla común para análisis simples (*minimal physics análisis*). Las clases SAN están pensadas para ser simples, clases de datos basadas en Root con el mínimo de dependencias con las herramientas del software de Athena.

- Tamaño por evento: 88 kb.
- Contiene estructuras complejas de datos.

3.3.2 Ventajas

En lugar de guardar el p_T , η , ϕ , e , etc como ramas individuales en el AANT, en SAN uno grabaría objetos más complejos (y compactos) que contendrían estas y otras variables.

El usuario tiene la posibilidad, por tanto, de crear sus propios objetos durante cierto análisis en Athena y añadirlos al AANT. Esto ofrece también la posibilidad de añadir los objetos AOD (Electron, Muon, Photon, etc.) a la AANT del usuario.

La creación de SAN tiene varias ventajas. Permite al usuario utilizar las mismas clases tanto para análisis en Root como en Athena. Siendo así, debería ser más fácil mover piezas de código *hacia delante y hacia atrás* en la secuencia del ‘Event Data Model’. Imagina que estás desarrollando un procedimiento complicado de identificación de electrones en Root, y que esta pieza de código puede ser útil para una gran cantidad de gente en ATLAS. Con las NTuplas estructuradas debería ser más fácil subir esta pieza de código a Athena sin tener que modificarla ostensiblemente para que encaje en la estructura de Athena.

Otra ventaja de las SAN es que debería ser más fácil comparar y compartir piezas de código entre los diferentes análisis (ya sean en Root o en Athena) ya que los objetos fundamentales son los mismos que en Root.

‘With the structured NTuple, we achieve some transparency in the ATHENA-ROOT interplay.’[1]

En particular:

- Puede replicar la estructura de nuestro Event Data en Root.
- Es Athena-aware en el sentido del TAG. (‘Event selection’ desde el AANTTree y ‘Back Navigation’ hasta el original AOD, ESD y RDO)

- Incluye TODAS las clases en el paquete, evitando así dependencias externas.
- Es tan simple como copiar el paquete UserAnalysisEvent a tu portátil.
- Es independiente de la plataforma.
- Se facilita un Makefile para producir una librería compartida de todas las clases SAN y el diccionario.
- Es posible producir SAN a gran escala usando *‘Jobs Transformations’*.

3.3.3 Inconvenientes

- Para añadir objetos complejos a las AANT del usuario, se precisa la versión 5 de Root o mayor, de manera que esto no es posible para las *releases* 11.0.X o incluso 11.5.0 o superiores.
- El usuario necesita las clases SAN y el diccionario en su portátil.
- No se puede leer en el Athena Event Loop. (excepto para *‘event selection’*).
- Para generar los diccionarios requiere de gcc-xml (el equipo Reflex del CERN no cubre la mayor parte del rango de plataformas usables en el portátil).
- SAN no tiene un mecanismo para soportar *‘schema evolution’*.
- *‘... we finally believe that SAN can not replace easily the existing POOL-based AOD.’* [2].

3.3.4 Consideraciones prácticas

- He seguido las indicaciones de la *‘twiki page on SAN’* hasta que he dado con un problema que no he podido resolver todavía. Hay un punto en que se copia un fichero comprimido al portátil conteniendo todas las clases SAN necesarias en Root. Al descomprimirlo se crean 5 directorios. En el directorio /Root debe estar presente el fichero Makefile, el cuál debería copiarse al directorio /run y, junto con el código fuente, permitir generar la librería compartida del paquete. En mi caso el Makefile no está presente. He consultado a Ketevi Assamagan por mail pero aún no he obtenido respuesta.
- La posibilidad de producir SAN con EventView está siendo implementada.
- La clase base que usa AnalysisSkeleton en Root se basa en TSelector (reemplazando a MakeClass).

4 Conclusiones

Parece ser que los candidatos con más posibilidades son SAN y pAOD (AOD separado en transient y persistent). Según se menciona en el *‘Report of the AOD Format Task Force’*[2], el prototipo de SAN proveerá las bases para la representación persistent de los AOD. Cabe esperar, por tanto, que los dos candidatos acabaren fusionándose.

‘We should further try as much as possible to share a common interface among the SAN and AOD transient classes. This will facilitate porting analysis code between SAN/ROOT and AOD/Athena analysis environments.’[2]

References

- [1] <https://twiki.cern.ch/twiki/bin/view/Atlas/AddingObjectsToAAN#Introduction>
 [2] <https://twiki.cern.ch/twiki/bin/view/Atlas/AODFormatTaskForce#Mandate>