Guía de AliRoot

Sergio Best

2016

Al momento de escribir esta guía, salió un nuevo método para instalar AliRoot. El método es más sencillo que el anterior, pero vale la pena saber por qué es más sencillo.
AliRoot se construye sobre varios otros programas para hacer las simulaciones, los MonteCarlos, la geometría, tales como Root y GEANT. Antes, para instalarlo, era necesario instalar todas estas dependencias. Había libertad para escoger las versiones de las dependencias, pero algunas combinaciones eran recomendadas por los desarrolladores ya que estaba probado que funcionaban.Era un dolor de cabeza.
El nuevo método de AliBuild instala directamente las dependencias necesarias y las maneja, actualiza, de acuerdo a como vayan cambiando las cosas, evitando así los problemas de dependencias. Esto es muy útil para olvidarte de los programas detrás y concentrarte en trabajar con AliRoot. Si más adelante deseas probar cosas nuevas, AliBuild te da la opción de especificar las dependencias a mano.
La mejor guía para instalar AliRoot es la de Dario Berzano. Esta guía fue hecha en el 2016, así que el método de instalación puede haber cambiado. De todas formas es bueno revisar su página. `https://dberzano.github.io/alice/install-aliroot/`. En caso la página ya no esté disponible por alguna razón, haré un pequeño resumen de como instalar Aliroot con AliBuild, pero si está la página de Dario aún, conviene más seguir su guía de instalación. También hay una guía sobre el uso de Git que es bastante útil aprender para manejar las carpetas de AliRoot.

1 Instalando AliRoot

Primero necesitas Pip. Un programa para instalar y manejar paquetes de Python. Si aun no lo tienes o no estás seguro de si lo tienes escribe en terminal.

```
sudo apt-get install python-pip
```

O su equivalente en tu distribución de Linux. Luego usas pip para instalar AliBuild.

```
pip install alibuild
```

Luego creamos una nueva carpeta e instalamos AliRoot y todas sus dependencias allí.

```
mkdir $HOME/alice && cd $HOME/alice
aliBuild init AliRoot,AliPhysics -z ali-master
cd ali-master
aliBuild -z -w ../sw -d build AliPhysics
alienv enter AliPhysics/latest-ali-master
```

Para simplicar más las cosas, pondremos en el .bashrc las siguientes líneas

```
ALICE_WORK_DIR=$HOME/alice/sw
eval "`alienv shell-helper`"
```

Una vez instalado todo y configurado el .bashrc. Para entrar en entorno AliRoot basta con escribir en terminal.

```
alienv load
```

Esto carga las variables de entorno necesarias y te pone en modo AliRoot. Para salir puedes usar la siguiente línea.

```
alienv unload
```

2 Configuración

Aquí definimos varios parámetros que utilizaremos más adelante en la simulación. Todo esto se encuentra definido en el Config.C. Vamos a revisar uno de estos códigos para identificar qué parámetros se definen.

2.1 Config.C

Trabajaremos con el Config.C del MFT. En AliRoot esto se encuentra en la carpeta "aliroot/master/inst/MFT/". Hay una copia del archivo en la carpeta del Git. Solo pondré partes del código para poder explicarlo mejor.

```
LoadLibs();

new TGeant3TGeo("C++ Interface to Geant3");

// Create the output file

AliRunLoader* rl=0x0;

printf("Config.C: Creating Run Loader ...");
rl = AliRunLoader::Open("galice.root",
AliConfig::GetDefaultEventFolderName(), "recreate");
if (rl == 0x0) {gAlice->Fatal("Config.C",
"Can not instatiate the Run Loader");
  return;
}
rl->SetCompressionLevel(2);
rl->SetNumberOfEventsPerFile(1000);
gAlice->SetRunLoader(rl);
```

```
TVirtualMCDecayer *decayer = new AliDecayerPythia();
decayer->SetForceDecay(kAll);
decayer->Init();
gMC->SetExternalDecayer(decayer);
```

Primero hemos cargado las librerías con LoadLibs, puedes revisar la función completa que se encuentra en la última parte del código. Con TGeant3TGeo inicializamos el objeto donde entrará la geometría del detector. AliRunLoader que leerá el galice.root que es donde se encuentra la información necesaria para la simulación. Lo siguiente importante es en SetNumberOfEventsPerFile. Aquí pondras cuántos eventos entran en cada archivo. Cuidado. Si este número es muy bajo, saldrán demasiados archivos. Por el contrario, si es muy grande, será difícil manejarlos. Adáptalo a tus necesidades.

En este caso no están seteando una semilla para el generador de números aleatorios, pero es bueno colocarla. Con gRandom->SetSeed(0) toma la hora del reloj de la computadora. Si quieres comparar con otra computadora, pon en ambas el mismo seed.
Finalmente en el último recuadro definen e inicializan el generador de decaimientos, Pythia. Añadiéndolo al generador de Monte Carlo.

```
gMC->SetProcess("DCAY",1);
gMC->SetProcess("PAIR",1);
gMC->SetProcess("COMP",1);
gMC->SetProcess("PHOT",1);
gMC->SetProcess("PFIS",0);
gMC->SetProcess("DRAY",0);
gMC->SetProcess("ANNI",1);
gMC->SetProcess("BREM",1);
gMC->SetProcess("MUNU",1);
gMC->SetProcess("CKOV",1);
gMC->SetProcess("HADR",1);
gMC->SetProcess("LOSS",2);
gMC->SetProcess("MULS",1);
gMC->SetProcess("RAYL",1);
```

Aquí se definen qué procesos estarán involucradas en la simulación. Decayment, Pair Production, Annihilation, Cherenkov, etc.

```
Int_t iABSO  = 1;
Int_t iDIPO  = 1;
Int_t iHALL  = 1;
Int_t iMUON  = 1;
Int_t iPIPE  = 1;
Int_t iSHIL  = 1;
Int_t iT0    = 0;
Int_t iVZERO = 1;
Int_t iMFT   = 1;
Int_t iEMCAL = 0;
Int_t iFMD   = 0;
Int_t iFRAME = 0;
Int_t iITS   = 0;
Int_t iTOF   = 0;
Int_t iTPC   = 0;
Int_t iTRD   = 0;
Int_t iZDC   = 0;
```

Aquí se definen qué detectores estarán involucradas en la simulación.

3 Simulación

Originalmente en la simulación tenemos toda la información de qué partículas pasaron por qué sitios. Pero eso no es lo que medimos realmente con un detector, así que la data que obtengamos de la simulación debemos transformarla a información que un detector nos puede dar. Vamos a perder información en el proceso, pero está bien, porque en la vida real, un detector pierde mucha información de los eventos que ocurren en su interior. La información pasa por los siguientes pasos. Cada paso se puede encontrar como archivos después de la simulación.

3.1 Kinematics

Aquí se almacenan todas las partículas generadas durante la simulación. La información está completa en este paso.

3.2 Hits

Ahora se simula el paso de las partículas por las piezas del detector usando GEANT. Cuando una partícula atraviesa un material, deposita energía que es lo que se mide. Se almacena la información de la energía depositada, posición, tiempo y partícula que impactó.

3.3 SDigits

A pesar de su nombre, este paso aun no esta digitalizado. Solo se suma la energía depositada en una celda del detector. Aquí perdemos información de los hits individuales que golpearon la celda. Tal como en la vida real.

3.4 Digits

Ahora, el detector no va a decirte: "Depositaron 3 Joules de energía aquí". El detector solo pasa información en 0's y 1's. La energía depositada se traduce a un voltaje (un valor analógico) que luego se pasa a un valor digital usando un ADC (Analogue to Digital Converter). Ciertas cosas a tomar en cuenta: La electrónica real produce ruido de fondo, esto se agrega a la data en este punto. También, si una señal es muy débil, se elimina.

3.5 Raw Data

Más allá de eso, un detector trabaja con ventanas de tiempo. Así que los Digits se organizan en conjuntos por el tiempo en que ocurrieron. Este paso se puede obviar.

3.6 runSimulation.C

```
AliSimulation *simulator = new AliSimulation(config);
TDatime dt;
UInt_t seed = dt.Get();
simulator->SetSeed(seed);
simulator->SetRunNumber(runNumber);
simulator->SetTriggerConfig("MUON");
simulator->SetMakeDigits("MUON MFT");
simulator->SetMakeSDigits("MUON MFT");
```

Aquí damos la configuración al simulador, definimos la semilla con la fecha y configuramos el número del Run. Además, de acuerdo a qué detector sea, se convierte a Digits y SDigits de forma distitna. Recuerda que cada detector tiene sus propias condiciones de activación y ruido de electrónica.

```
TStopwatch timer;
timer.Start();
simulator->Run(nevents);
timer.Stop();
timer.Print();
```

La mayor parte es solo para saber cuánto tiempo le toma al simulador hacer todo, pero lo importante está en Run(nevents). Este es el número de eventos que generaras. Ten en cuenta la capacidad de procesamiento de tu computadora y cuánto tiempo tienes disponible.

4 Reconstrucción

Ahora vamos a tomar el camino inverso y veremos si con los Digits o Raw Data podemos reconstruir la identidad de las partículas incidentes.
Para empezar a reconstruir, AliReconstruction forma clusters de Digits. Cuando una partícula pasa por un material, activa varias celdas. Todas estas se pueden agrupar por cercanía espacial y temporal. Luego cada cluster tiene características tales como la energía total, que correspondería a la energía entregada por la partícula incidente, un punto representativo de impacto, qué tan esparcida está la forma del cluster. Todas estas cosas nos dan idea de las características de lo que impactó a nuestro detector. Toda esta información se guarda en los RecPoints.
Luego se reconstruyen, a partir de los clusters, las partículas originales.
Todo esto se ejecuta usando una macro de reconstrucción que veremos a continuación. Tras finalizar, el output será un archivo ESD el cual contiene las partículas reconstruidas. Este es el archivo que analizaremos.

4.1 runReconstruction.C

```
AliReconstruction *reco = new AliReconstruction("galice.root");

// switch off cleanESD
reco->SetCleanESD(kFALSE);
```

Esta macro se parece mucho a la de simulación. La única diferencia es que en este caso cargamos el galice.root. Pero lo importante es que generaremos el ESD, recuerda que aquí estará la información de las partículas reconstruidas.

5 Análisis de datos

Analizaremos de la misma forma un ejemplo de la misma carpeta.

5.1 RunAnalysisTaskMFTExample.C

```
AliAnalysisManager *mgr = AliAnalysisManager::GetAnalysisManager();
  if (mgr) delete mgr;
  mgr = new AliAnalysisManager("AM","Analysis Manager");
```

El encargado del análisis será el AliAnalysisManager.

```
  TStopwatch timer;
  timer.Start();
  mgr->InitAnalysis();
  mgr->PrintStatus();

  if (!strcmp(runType,"grid")) {
    printf("Starting MFT analysis on the grid");
    mgr->StartAnalysis("grid");
  }

  else if (!strcmp(runType,"local")) {
    printf("Starting MFT analysis locally");
    mgr->StartAnalysis("local", GetInputLocalData());
  }
```

Solo queda inicializarlo, darle los parámetros necesarios y dejar que analice nuestra reconstrucción de la simulación. Esto es para poder leer la reconstrucción que se ha hecho a partir del ESD.

References

[1] Alice Software Installation, *https://dberzano.github.io/alice/install-aliroot/* Dario Berzano, 2016

[2] Alice Offline Manual, *http://aliweb.cern.ch/Offline/AliRoot/Manual.html* ALICE@CERN, 2011

[3] EMCal Beginners, *http://aliweb.cern.ch/secure/Offline/sites/aliceinfo.cern.ch.secure.Offline/files/uploads/EMCAL/EMCal_Beginners.pdf* Gustavo Conesa Balbastre, 2010