Manual de implementación de un esquema de computación distribuida para Geant4

Grupo de Física Nuclear

Autor:

Wilson Javier Almario Rodriguez.

Universidad Nacional De Colombia
Grupo de física nuclear
Bogota, Colombia
Junio 2020

Índice

1. Introducción

1.1. Conceptos claves

1.1.1. Computación Paralela

La computación paralela es el uso simultáneo de más de un procesador para resolver un problema. La computación paralela es una técnica de programación en la que muchas instrucciones se ejecutan simultáneamente. Se basa en el principio de que los problemas grandes se pueden dividir en partes más pequeñas que pueden resolverse de forma concurrente.

1.1.2. MPI

MPI es un estándar de programación en paralelo mediante paso de mensajes que permite crear programas portables y eficientes. El paso de mensajes es una técnica empleada en programación concurrente para aportar sincronización entre procesos y permitir la exclusión mutua.

Su principal característica es que no precisa de memoria compartida, por lo que es muy importante en la programación de sistemas distribuidos.

Al arrancar una aplicación se lanzan en paralelo n copias del mismo programa(procesos), facilita legibilidad y mantenimiento Distinción del código a ejecutar por cada uno: mediante el id del proceso

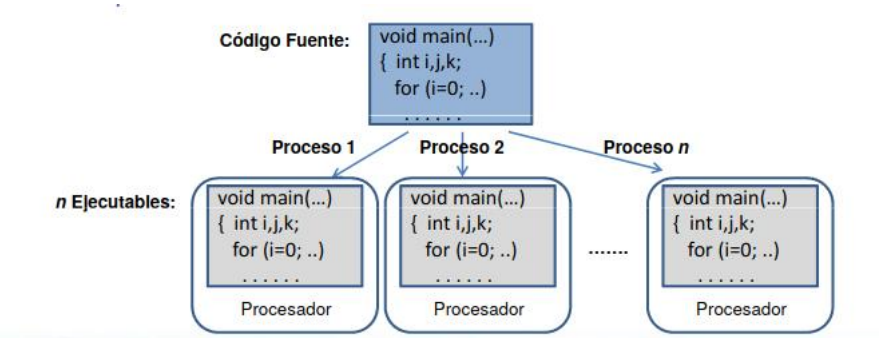


Figura 1: Estructura MPI

1.1.3. NFS

El Sistema de archivos de red (NFS) es una aplicación cliente / servidor que permite al usuario de una computadora ver y opcionalmente, almacenar y actualizar archivos en una computadora remota como si estuvieran en la computadora del usuario. El protocolo NFS es uno de los varios estándares del sistema de archivos distribuidos para el almacenamiento conectado a la red (NAS).

NFS permite al usuario o administrador del sistema montar (designar como accesible) todo o una parte de un sistema de archivos en un servidor. Los clientes pueden acceder a la parte del sistema de archivos que está montada con los privilegios asignados a cada archivo (solo lectura o lectura-escritura). NFS utiliza llamadas a procedimiento remoto (RPC) para enrutar solicitudes entre clientes y servidores. https://searchenterprisedesktop.techtarget.com/definition/Network-File-System

1.1.4. Geant4

Geant4 es un conjunto de herramientas para la simulación del paso de partículas a través de la materia. Sus áreas de aplicación incluyen física de alta energía, nuclear y aceleradora, así como estudios en medicina y ciencia espacial.

1.1.5. G4mpi

G4MPI es una interfaz nativa con bibliotecas MPI. El directorio contiene una biblioteca Geant4 UI y un par de ejemplos paralelos. Al usar esta interfaz, las aplicaciones de los usuarios se pueden paralelizar con

diferentes bibliotecas compatibles con MPI, como OpenMPI, LAM / MPI, MPICH2, etc.

1.2. Metodología y Esquema general de la arquitectura

1. Hacer hostst/cluster del entorno MPI
2. Lanzar el entorno de ejecución de MPI.
3. Ejecutar aplicación mpirun -np 8 myapp
4. Ejecutar comandos MPI G4UI, ubicados en /mpi/

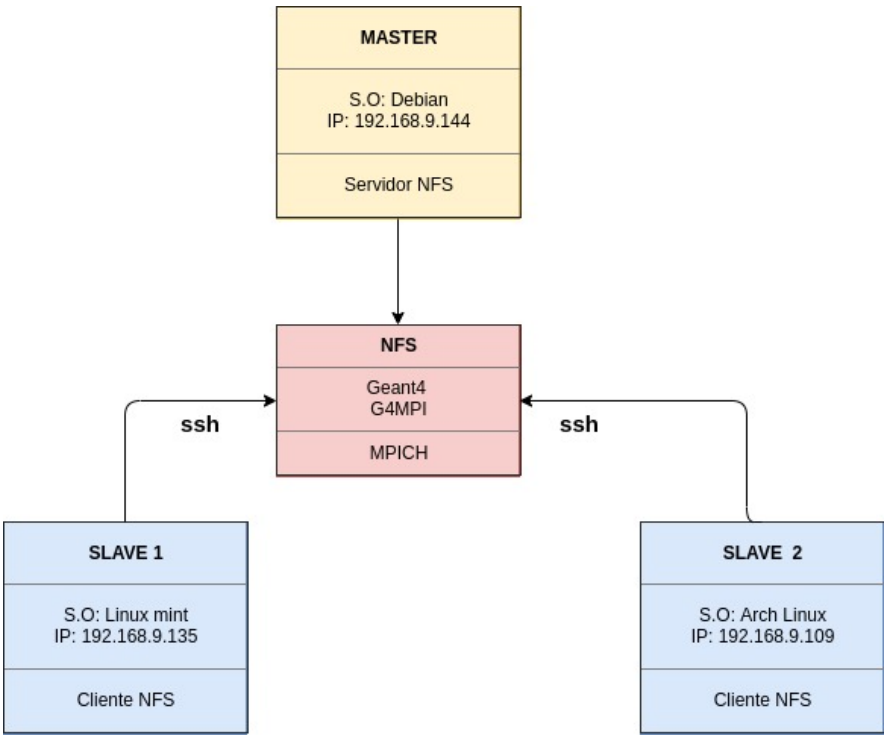


Figura 2: Esquema

2. Instalación de Geant4

2.1. Instalación de prerrequisitos

- **CLHEP** http://proj-clhep.web.cern.ch/proj-clhep/clhep23.html
  leer el archivo README.txt luego crear una carpeta Build y una install dentro de la carpeta build hacer:

```
cmake -DCMAKE_INSTALL_PREFIX=<install_dir> <source_code_dir>
cmake --build . --config RelWithDebInfo
cmake --build . --target install
```

- **Zlib**

```
sudo apt install zlib1g-dev
```

- **QT**

```
sudo apt install qt4-default
```

- **OpenGL**

```
sudo apt-get install libglu1-mesa-dev freeglut3-dev mesa-common-dev
```

- **ray tracer**

```
sudo apt install povray
```

- **X11 libraries**

```
sudo apt-get install xorg-dev
```

- **X11 libraries**

```
sudo apt-get install xorg-dev
```

- **g++ y gcc**

```
sudo apt-get update -y && \

sudo apt-get install -y wget g++-5 gcc-5 cmake libexpat1-dev vim cmake-curses-gui
    freeglut3 freeglut3-dev mesa-utils python libx11-dev libxmu-dev expat && \

sudo apt-get install -y libfontconfig1-dev libfreetype6-dev libxcursor-dev libxext-dev
    libxfixes-dev libxft-dev libxi-dev libxrandr-dev libxrender-dev && \

sudo apt-get install libssl-dev
```

2.2. Instalación de Geant4

1. Creación de carpetas donde estará alojada la instalación

```
$ cd $HOME
$ sudo mkdir -p $HOME/Geant4-10.6/src
$ sudo mkdir -p $HOME/Geant4-10.6/build
$ sudo mkdir -p $HOME/Geant4-10.6/install
$ sudo mkdir -p $HOME/Geant4-10.6/data
```

2. Descarga de Geant4:

```
$ G4_VERSION="10.06.p01"
$ QT_VERSION="4.8.7"
$ sudo wget http://cern.ch/geant4-data/releases/geant4.${G4_VERSION}.tar.gz
$ sudo tar xf geant4.${G4_VERSION}.tar.gz -C $HOME/Geant4-10.6/src
$ sudo rm geant4.${G4_VERSION}.tar.gz
```

3. Compilación:

```
$ cd $HOME/Geant4-10.6/build
$ cmake -DCMAKE_INSTALL_PREFIX=$HOME/Geant4-10.6/install \
      -DGEANT4_USE_OPENGL_X11=ON \
      -DQT_QMAKE_EXECUTABLE=/usr/local/Trolltech/Qt-${QT_VERSION}/bin/qmake \
      -DGEANT4_USE_QT=OFF \
      -DGEANT4_INSTALL_DATA=ON \
      -DGEANT4_INSTALL_DATADIR=$HOME/Geant4-10.6/data \
      -DGEANT4_BUILD_MULTITHREADED=ON \
      -DGEANT4_INSTALL_EXAMPLES=ON \
      ../src/geant4.${G4_VERSION}

$ make -j`nproc`
```

4. Instalación:

```
$ sudo make install
```

5. Cargar varibles de entorno:

```
$ echo "export LD_LIBRARY_PATH=${LD_LIBRARY_PATH}:/usr/local/Trolltech/Qt-${QT_VERSION
    }/lib" >> $HOME/.bashrc
$ echo "source $HOME/Geant4-10.6/install/bin/geant4.sh" >> $HOME/.bashrc
```

6. Verificar instalación y configuración

```
$ ./geant4-config --help
```

7. Por último crear variables de entorno:

```
echo "source $HOME/geant4/install/bin/geant4.sh" >> $HOME/.bashrc
```

2.2.1. Configurar variables de entorno para Geant4

1. Abrir el archivo bashrc:

```
$ sudo nano ~/.bashrc
```

2. Añadir al final las siguiente líneas:

```
export G4ABLADATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4ABLA3.1
export G4ENSDFSTATEDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4ENSDFSTATE2.2
export G4INCLDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4INCL1.0
export G4LEDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4EMLOW7.9.1
export G4LEVELGAMMADATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/PhotonEvaporation5.5
export G4NEUTRONHPDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4NDL4.6
export G4PARTICLEXSDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4PARTICLEXS2.1
export G4PIIDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4PII1.3
export G4REALSURFACEDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/RealSurface2.1.1
export G4SAIDXSDATA=/home/gfnun/Geant4-10.6/install/share/Geant4-10.6.1/data/G4SAIDDATA2.0
```

2.2.2. Ejecutar Ejemplo B1

1. Ir a la carpeta del código fuente del ejemplo y crear una carpeta para la compilación.

```
$ cd /home/gfnun/Geant4-10.6/src/geant4.10.06.p01/examples/basic/B1
$ mkdir build-B1
```

2. Compilar y ejecutar:

```
$ cmake    DGeant4_DIR   =/home/gfnun/Geant4-10.6/install/lib/Geant4-10.6.1 /home/gfnun/Geant4-10.6/src/geant4.10.06.p01/examples/basic/B1
$ make -j4
$ ./exampleB1
```

3. Configuraciones básicas de cada equipo

3.1. Configuración de usuarios de cada máquina

Todas las máquinas se configuran con el mismo nombre de usuario, así en cada máquina abrimos una terminal y ejecutamos los siguientes comandos:

1. Entrar al modo root:

```
$ ifconfig -a
```

2. Agregar el usuario:

```
$ adduser gfnun
```

   Se solicitara el ingreso de la clave y unos datos adicionales los cuales deberan ingresarse si se desea, de lo contrario solo hacer *enter* y continuar.

3. Comprobar que se creo el usuario:

```
$ cd ~
$ ls /home
```

4. Se debera ver la carpeta del usuario, adicional ingresamos para comprobar:

```
$ ssh gfnun@0.0.0.0
```

5. Salir y se le da permisos de administrador, desde el root:

```
$ exit
$ adduser gfnun sudo
```

3.2. Configurar ip estática

1. Comprobar las interfaces de red que tiene el equipo.

```
$ ifconfig -a
```

2. Edita el archivo de configuración de las interfaces de red con el siguiente comando:

   sudo nano /etc/network/interfaces

3. En este caso se va configurar la interfaz eth0,

```
# Configuraci n de direcci n IP fija para el interfaz eth0
auto eth0
iface eth0 inet static
address 192.168.1.50
netmask 255.255.255.0
network 192.168.1.0
broadcast 192.168.1.255
gateway 192.168.1.1
```

4. Reiniciar las interfaces de red del equipo:

```
$ sudo /etc/init.d/networking restart
```

5. Alternativamente:

```
$ sudo ifconfig eth0 down
$ sudo ifconfig eth0 up
```

3.3. Configurar archivo *Hosts*

Configurar el archivo hosts con las ip de los PC's con los que se va hacer el *clúster* (ver sección configuraciones básicas de cada equipo, para configuración de ip estática).

Para ingresar al archivo:

```
$ sudo nano /etc/hosts
```

El archivo *hosts* debe quedar así. En este caso tenemos dos esclavos y un maestro.

```
127.0.0.1       localhost
127.0.1.1       MrHerbert.gfnun.unal.edu.co     MrHerbert

#MPI Cluster Setup
192.168.9.144 master
192.168.9.135 slave1
192.168.9.109 slave2

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Listing 1: Archivo hosts en el maestro

También se debe configurar el de cada esclavo así:

```
#MPI Cluster Setup
192.168.9.144 master
192.168.9.109 slave1

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Listing 2: Archivo hosts esclavo 1

Para el esclavo 2:

```
#MPI Cluster Setup
192.168.9.144 master
192.168.9.109 slave2

# The following lines are desirable for IPv6 capable hosts
::1     localhost ip6-localhost ip6-loopback
ff02::1 ip6-allnodes
ff02::2 ip6-allrouters
```

Listing 3: Archivo esclavo 1

3.4. Configurar ssh

Lo siguiente es configurar los accesos a través de ssh entre las máquinas para que no solicite clave cada vez que la máquina maestro (master) acceda a cada esclavo (slave1, slave2..etc)

1. En el Master, generar la llave rsa:

```
$ ssh-keygen -t rsa
```

2. Verificar si se tiene el archivo authorized_keys, hacer esto para cada nodo:

```
$ ls -l .ssh/
```

3. Si no se encuentra, crearlo:

```
$ touch ~/.ssh/authorized_keys
```

4. Generar permisos al archivo:

```
$ chmod 600 ~/.ssh/authorized_keys
```

5. Copiar llaves generadas desde el master a los esclavos:

```
$ scp .ssh/id_rsa.pub gfnun@slave1:.ssh/authorized_keys
$ scp .ssh/id_rsa.pub gfnun@slave2:.ssh/authorized_keys
```

6. También se debe copiar en el mismo master para que no solicite clave al hacer ssh sobre el mismo:

```
$ cp ~/.ssh/id_rsa.pub ~/.ssh/authorized_keys
```

7. Para el esclavo dos

```
$ scp .ssh/id_rsa.pub gfnun@slave2:.ssh/authorized_keys
```

   Esto es importante ya que con la implementación de mpi a veces se desea usar el mismo master para ejecutar aplicaciones.

8. Así no pedirá contraseña cada vez que se va ingresar a cada nodo desde el master, bastara escribir algo así:

```
$ ssh slave2
```

4. Instalación y configuración de servidor NFS

NFS (sistema de archivos de red: «Network File System») es un protocolo que permite acceso remoto a un sistema de archivos a través de la red. Todos los sistemas Unix pueden trabajar con este protocolo.

4.1. Configuración servidor

1. En el master

```
$ sudo apt install nfs-kernel-server portmap
$ mkdir sharedFolder
$ ls sharedFolder/
```

2. Cambiar el propietario de la carpeta a nadie.

```
$ sudo chown nobody:nogroup sharedFolder/
```

3. Ver el estado del servidor:

```
$ sudo /etc/init.d/nfs-kernel-server status
```

4. Detener el servicio:

```
$ sudo /etc/init.d/nfs-kernel-server stop
```

5. Modificar el archivo exports:

```
$ sudo pico /etc/exports
```

6. Añadir:

```
/home/gfnun/sharedFolder/ *(rw,sync,no_subtree_check)
/home/gfnun/Geant4-10.6/ *(rw,sync,no_subtree_check)
/home/gfnun/Geant4-10.5/ *(rw,sync,no_subtree_check)
```

7. Volver a la terminal, y hacer que todos los directorios colocados allí se exporten:

```
$ sudo exportfs -a
```

8. Verificar:

```
$ sudo exportfs
```

9. Salida en la terminal:

```
/home/gfnun/sharedFolder
    <world>
/home/gfnun/Geant4-10.6
    <world>
/home/gfnun/Geant4-10.5
    <world>
```

10. En los esclavos:

```
$ sudo apt-get install nfs-common portmap
$ mkdir -p sharedFolder/
$ ls sharedFolder
$ mkdir -p Geant4-10.6/
$ ls sharedFolder
```

11. En el maestro iniciar el servidor:

```
$ sudo /etc/init.d/nfs-kernel-server start
```

12. Ver el estado:

```
$ sudo /etc/init.d/nfs-kernel-server status
```

13. Ahora compartir las carpetas en cada esclavo, por ejemplo para el esclavo 1 (*slave1*):

```
$ ssh slave1
$ sudo mount master:/home/gfnun/sharedFolder sharedFolder/
$ sudo mount master:/home/gfnun/Geant4-10.6/ Geant4-10.6/
$ sudo mount master:/home/gfnun/Geant4-10.5/ Geant4-10.5/
```

14. Verificar que se esta compartiendo:

```
$ df -h
```

15. La salida debería ser:

```
master:/home/gfnun/sharedFolder               723G  440G  247G  65% /home/gfnun/
    sharedFolder
master:/home/gfnun/Geant4-10.6                723G  440G  247G  65% /home/gfnun/Geant4
    -10.6
master:/home/gfnun/Geant4-10.6                723G  440G  247G  65% /home/gfnun/Geant4
    -10.5
```

16. En uno de los esclavos podemos probar que se comparten los archivos:

```
$ cd sharedFolder/
$ touch test.txt
```

y verificar que se vea desde el maestro.

5. Instalación de Open-MPI

Descargar open-mpi desde https://www.open-mpi.org/software/ompi/v4.0/, se descarga el archivo openmpi-4.0.3.tar.gz. Dejarlo en el directorio:

```
$ pwd
/home/gfnun/
```

1. Descomprimir en la carpeta sharedFolder/ :

```
$ cp openmpi-4.0.3.tar.gz sharedFolder/
$ cd sharedFolder/
$ mkdir openmpi-install/
$ tar -xzf openmpi-4.0.3.tar.gz
```

2. Compilar:

```
$ cd openmpi-4.0.3/
$ mkdir build
$ cd build
$ ../configure --prefix=/home/gfnun/sharedFolder/openmpi-install/ --enable-mpi-cxx --
    enable-orterun-prefix-by-default
$ sudo make -j4
```

3. Instalar:

```
$ sudo make install
$ sudo ldconfig
```

4. Para usar open mpi instalado en la carpeta sharedFolder/ cargamos las variables de entorno:

```
$ export PATH=/home/gfnun/sharedFolder/openmpi-install/bin:$PATH
$ export LD_LIBRARY_PATH=/home/gfnun/sharedFolder/openmpi-install/lib:$LD_LIBRARY_PATH
```

5. Si se desea, se puede dejar en el bashrc:

```
echo "export PATH=/home/gfnun/sharedFolder/openmpi-install/bin:$PATH" >> $HOME/.bashrc
echo "export LD_LIBRARY_PATH=/home/gfnun/sharedFolder/openmpi-install/lib:
    $LD_LIBRARY_PATH" >> $HOME/.bashrc
```

así cargara automáticamente las variables de entorono de mpi cada vez que se inicie sesión.

5.1. Configurar Cluster

Crear un archivo en donde vamos a listar los nodos del *clúster* (nodos se refiere a cada máquina del *clúster*). En este archivo se indica los procesos a usar por máquina.

A continuación se presentan ejemplos del archivo:

- Creación del Archivo:

```
sudo touch my_hosts
```

- Añadir las siguientes líneas, indicando el número de hilos a usar por cada nodo:

```
master slots=1
slave1 slots=2
slave2 slots=2
```

- Ejecutar ejemplo:

```
$ cd ..
$ cd examples
$ sudo mpicc hello_c.c -o hello_c
$ mpiexec -np 4 --hostfile my_hosts ./hello_cslave1
slave2
```

5.2. Ejemplo de Comandos MPI

```
mpirun -np 6 --hostfile my_hosts openmpi-4.0.3/examples/hello_c | sort
mpirun -H slave1,slave2 openmpi-4.0.3/examples/hello_c | sort
mpirun -H master,slave1,slave2 openmpi-4.0.3/examples/hello_c | sort
mpirun -H master,slave1,slave2,slave2 openmpi-4.0.3/examples/hello_c | sort
mpirun -H master,slave1,slave2 -npernode 2 openmpi-4.0.3/examples/hello_c | sort
```

6. Compilar G4mpi

G4MPI es una interfaz nativa con bibliotecas MPI. El directorio contiene una biblioteca de IU de Geant4 y un par de ejemplos paralelizados. Con esta interfaz, las aplicaciones de los usuarios se pueden paralelizar con diferentes bibliotecas compatibles con MPI, como OpenMPI, LAM / MPI, MPICH2, etc.

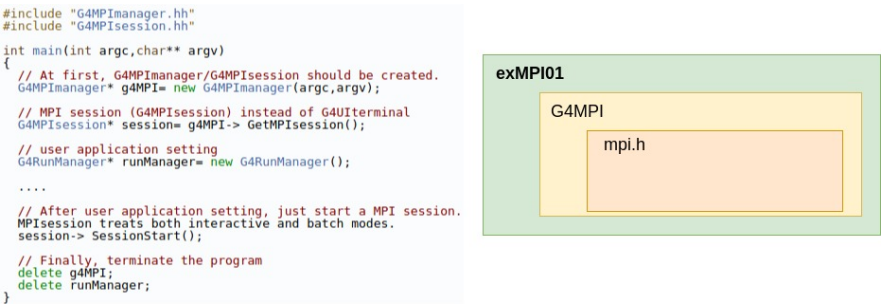


Figura 3: Esquema

6.1. Compilando G4mpi

```
mkdir build-mpi
cd build-mpi
cmake -DGeant4_DIR=/home/gfnun1/geant4/install/lib/Geant4-10.6.1 \
     -DCMAKE_INSTALL_PREFIX=/home/gfnun1/geant4/geant4-mpi \
     /home/gfnun1/geant4/src/geant4.10.06.p01/examples/extended/parallel/MPI/source
make
make install
```

6.2. Compilando Ejemplo exMP01

```
mkdir build
cd build
cmake -DG4mpi_DIR=<where-G4mpi-wasintalled>/lib[64]/G4mpi -DCMAKE_CXX_COMPILER=mpicxx \
      -DGeant4_DIR=<your Geant4 install path>/lib[64]/Geant4-V.m.n <path-to-source>
      (V.m.n is the version of Geant4, eg. Geant4-9.6.0)
make
make install
```

```
mkdir build
cd build
sudo cmake -DG4mpi_DIR=/home/gfnun/geant4/geant4-mpi/lib/G4mpi-10.6.1 -DCMAKE_CXX_COMPILER=
    mpicxx \
      -DGeant4_DIR=/home/gfnun/geant4/install/lib/Geant4-10.6.1 /home/gfnun1/geant4/src/
    geant4.10.06.p01/examples/extended/parallel/MPI/examples/exMPI01
make
make install
```

6.3. Ejecutando Ejemplo con open-mpi

```
$ mpirun -np 3 --hostfile /home/gfnun1/my_hosts ./exMPI01
```

7. Ejecutando ejemplos de geant4 en el sistema distribuido

7.1. Ejemplo exMPI01

En esta sección se va ejecutar el ejemplo exMP01, ubicado en:

```
cd /home/gfnun/geant4.10.06/src/geant4.10.06.p01/examples/extended/parallel/MPI/examples/
    exMPI01
```

La documentación nos indica como debemos compilar el ejemplo.

```
mkdir build
cd build
cmake -DG4mpi_DIR=<where-G4mpi-wasintalled>/lib[64]/G4mpi -DCMAKE_CXX_COMPILER=mpicxx \
      -DGeant4_DIR=<your Geant4 install path>/lib[64]/Geant4-V.m.n <path-to-source>
      (V.m.n is the version of Geant4, eg. Geant4-9.6.0)
make
make install
```

Para este caso el ejemplo se compila así:

```
$ mkdir build
$ cd build
$ sudo cmake -DG4mpi_DIR=/home/gfnun/Geant4-10.6/geant4-mpi/lib/G4mpi-10.6.1 -
    DCMAKE_CXX_COMPILER=mpicxx \
      -DGeant4_DIR=/home/gfnun/Geant4-10.6/install/lib/Geant4-10.6.1 /home/gfnun/Geant4
    -10.6/src/geant4.10.06.p01/examples/extended/parallel/MPI/examples/exMPI01
$ sudo make
$ sudo make install
```

Ejecutar:

```
$ cd build
$ mpiexec -np 4 ./exMPI01
```

7.2. Ejemplo exMPI04

Compilar, para compilar se debe hacer desde el compilador de open mpi instalado en la carpeta sharedFolder/

```
$ mkdir build
$ cd build
$ sudo cmake -DG4mpi_DIR=/home/gfnun/Geant4-10.6/geant4-mpi/lib/G4mpi-10.6.1 -
    DCMAKE_CXX_COMPILER=/home/gfnun/sharedFolder/openmpi-install/bin/mpicxx \
      -DGeant4_DIR=/home/gfnun/Geant4-10.6/install/lib/Geant4-10.6.1 /home/gfnun/Geant4
    -10.6/src/geant4.10.06.p01/examples/extended/parallel/MPI/examples/exMPI04
$ make
$ make install
```

Ejecutar con Open mpi:
Exportar variables, en este caso la instalación de open mpi se realizó en sharedFolder:

```
export PATH=/home/gfnun/sharedFolder/openmpi-install/bin:$PATH
export LD_LIBRARY_PATH=/home/gfnun/sharedFolder/openmpi-install/lib:$LD_LIBRARY_PATH
```

De esta manera se ejecuta en la máquina local:

```
$ cd build
$ mpiexec -n 4 ./exMPI04
```

8. Configurar una aplicación para uso con MPI

Para este ejemplo se va usar la aplicación g4pntest, ubicada en este repositorio en la carpeta $00_g4_apps/$

8.1. Configuración sesión con G4mpi

1. Agregar las librerias necesarias para el uso de G4mpi, en PNtest.cc:

```
#include "G4MPImanager.hh"
#include "G4MPIsession.hh"
```

2. Luego en la función main() del archivo PNtest.cc se añade las siguientes líneas de código, para este ejemplo se añade después de definir la semilla:

```
  // --------------------------------------------------------------------
  // MPI session
  // --------------------------------------------------------------------
  G4MPImanager* g4MPI = new G4MPImanager(argc, argv);
  // MPI session (G4MPIsession) instead of G4UIterminal
  // Terminal availability depends on your MPI implementation.
  G4MPIsession* session = g4MPI-> GetMPIsession();
  G4String prompt = "[40;01;33m";
  prompt += "G4MPI";
  prompt += "[40;31m(%s)[40;36m[%/][00;30m:";
  session-> SetPrompt(prompt);
```

3. En este caso la aplicación tiene activado Multithreading (MT), se debe dejar sin MT:

```
//#ifdef G4MULTITHREADED //Modified vy Javier
//  G4MTRunManager* runManager = new G4MTRunManager;
//  runManager-> SetNumberOfThreads(4);
//#else
  G4RunManager* runManager = new G4RunManager;
//#endif
```

4. Para la sesión se comentan o se quitan las siguientes líneas de código:

```
/* ORIGINAL
  // get the pointer to the User Interface manager
  G4UImanager* UI = G4UImanager::GetUIpointer();

  if(argc == 1)
    {
#ifdef G4UI_USE
      G4UIExecutive* ui = new G4UIExecutive(argc, argv);
#ifdef G4VIS_USE
      UI->ApplyCommand("/control/execute vis.mac");
#endif
      ui->SessionStart();
      delete ui;
#endif
    }
  // Batch mode
  else
    {
      G4String command = "/control/execute ";
      G4String fileName = argv[1];
      UI->ApplyCommand(command+fileName);
    }
*/
```

5. Se inicia la sesión:

```
session-> SessionStart();
```

6. Por último se elimina la sesión de mpi agregando la siguiente línea de código antes del return 0:

```
  delete g4MPI;
  delete runManager;
  return 0;
```

8.2. Configuración CMakeLists.txt

1. Para el paquete de G4mpi agregar:

```
find_package(G4mpi REQUIRED)
```

2. Para incluir los directorios, se deja así:

```
include_directories(${CMAKE_CURRENT_SOURCE_DIR}/include
            ${Geant4_INCLUDE_DIR}
            ${G4mpi_INCLUDE_DIR})
```

3. Añadir ejecutable y enlace con las librerias Geant4:

```
target_link_libraries(PNtest ${G4mpi_LIBRARIES})
```

8.3. Configuración archivos .csv

1. La función RunAction::BeginOfRunAction(const G4Run* aRun), se deja así, para la generación de cada archivo .csv en cada nodo o cada hilo:

```
void RunAction::BeginOfRunAction(const G4Run* aRun)
{
  G4cout << "### Run " << aRun->GetRunID() << " start." << G4endl;
  G4RunManager::GetRunManager()->SetRandomNumberStore(false);

  // ntuples will be written on each rank
  G4int rank = G4MPImanager::GetManager()->GetRank();
  std::ostringstream fname;
  fname<<"out_test-rank"<<rank;


  G4AnalysisManager* analysisManager = G4AnalysisManager::Instance();
  analysisManager->SetVerboseLevel(2);
  analysisManager->SetFileName(fname.str());

  analysisManager->OpenFile();
  analysisManager->CreateH1("Edep","Energy (keV)", 1000, 0., 1000.); // Id = 0
}
```

2. Por último la función void RunAction::EndOfRunAction(const G4Run* aRun) se deja así:

```
void RunAction::EndOfRunAction(const G4Run* aRun)
{
  //if(!IsMaster()) return;
  //G4int nofEvents = aRun->GetNumberOfEvent();
  //if (nofEvents == 0) return;
  if(!IsMaster()) return;
  G4int nofEvents = aRun->GetNumberOfEvent();
  if (nofEvents == 0) return;

  G4cout << "RunActionMaster::EndOfRunAction" << G4endl;
  const G4int rank = G4MPImanager::GetManager()-> GetRank();

  G4cout << "======================================================" << G4endl;
  G4cout << "Start EndOfRunAction for master thread in rank: " << rank<<G4endl;
  G4cout << "======================================================" << G4endl;

  if ( !G4MPImanager::GetManager()->IsExtraWorker() ) {
```

```
18
19      //Save histograms *before* MPI merging for rank #0
20      if (rank == 0) {
21        auto analysisManager = G4AnalysisManager::Instance();
22        analysisManager->Write();
23        analysisManager->CloseFile(false); // close without resetting histograms
24      }
25
26
27        //Merge of g4analysis objects
28        G4cout << "Go to merge histograms " << G4endl;
29        G4MPIhistoMerger hm(G4AnalysisManager::Instance());
30        hm.SetVerbosity(1);
31        hm.Merge();
32        G4cout << "Done merge histograms " << G4endl;
33      }
34
35     //Save g4analysis objects to a file
36     //NB: It is important that the save is done *after* MPI-merging of histograms
37     //One can save all ranks or just rank0, chane the if
38    if (true){
39    // if (rank == 0){
40      auto analysisManager = G4AnalysisManager::Instance();
41      if (rank == 0) {
42        analysisManager->OpenFile("out_test-merged");  }
43      analysisManager->Write();
44      analysisManager->CloseFile();  }
45
46     G4cout << "=================================================" << G4endl;
47     G4cout << "End EndOfRunAction for master thread in rank: " << rank << G4endl;
48     G4cout << "=================================================" << G4endl;
49
50
51    // auto analysisManager = G4AnalysisManager::Instance();
52    // analysisManager->Write();
53    // analysisManager->CloseFile();
54  }
```

8.4. Configuración Clúster

Para configurar el clúster y hacer uso de este con open mpi creamos un archio de texto plano llamado hostfile. Para este caso se tiene disponible 5 computadores con 8 hilos cada uno, es decir el clúster tiene 40 hilos disponibles. Si se desea usar todos los hilos disponibles el archivo debería erse así:

```
1  master slots=8
2  slave1 slots=8
3  slave2 slots=8
4  slave3 slots=8
5  slave4 slots=8
```

Se puede administrar la cantidad de hilos a usar y en que computadores,por ejemplo:

```
1  master slots=2
2  slave1 slots=2
3  slave2 slots=2
4  slave3 slots=2
5  slave4 slots=2
```

En este caso configuramos el *clúster* para usar 2 hilos por cada computador lo que da un total de 10 hilos a usar por el *clúster*.

8.5. Comando Ejecución con Open mpi

A continuación el paso a paso para la ejecución de la aplicación:
Cargar variables de entorno:

```
$ source ~/Geant4-10.6/install/bin/geant4.sh
$ export PATH=/home/gfnun/sharedFolder/openmpi-install/bin:$PATH
$ export LD_LIBRARY_PATH=/home/gfnun/sharedFolder/openmpi-install/lib:$LD_LIBRARY_PATH
```

Compilar, para este caso por ejemplo el nombre de la carpeta donde se encuentra el proyecto es g4pntest$_mpi$:

```
$ cd g4pntest_mpi
$ mkdir build
$ cd build
$ cmake -DG4mpi_DIR=/home/gfnun/Geant4-10.6/geant4-mpi/lib/G4mpi-10.6.1 -DCMAKE_CXX_COMPILER
    =/home/gfnun/sharedFolder/openmpi-install/bin/mpicxx \
        DGeant4_DIR   =/home/gfnun/Geant4-10.6/install/lib/Geant4-10.6.1 ../
$ make -j`nproc`
```

Ejecución aplicación:

```
mpirun -np 40 -hostfile hostfile -x G4ABLADATA -x G4ENSDFSTATEDATA -x G4INCLDATA -x G4LEDATA
     -x G4LEVELGAMMADATA -x G4NEUTRONHPDATA -x G4PARTICLEXSDATA -x G4PIIDATA -x
    G4REALSURFACEDATA -x G4SAIDXSDATA -x G4RADIOACTIVEDATA -oversubscribe ./PNtest ../macros
    /THEmacro.mac
```

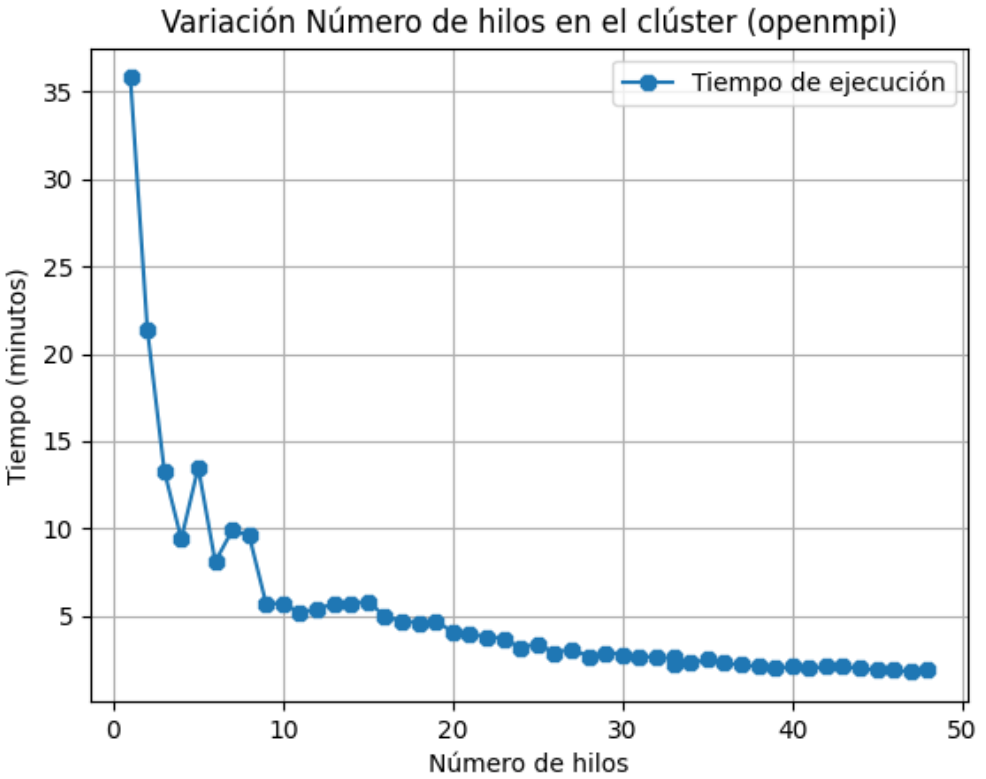
Si se requiere usar menos hilos, por ejemplo 12, el comando de ejecución sería así:

```
mpirun -np 12 -hostfile hostfile -x G4ABLADATA -x G4ENSDFSTATEDATA -x G4INCLDATA -x G4LEDATA
     -x G4LEVELGAMMADATA -x G4NEUTRONHPDATA -x G4PARTICLEXSDATA -x G4PIIDATA -x
    G4REALSURFACEDATA -x G4SAIDXSDATA -x G4RADIOACTIVEDATA -oversubscribe ./PNtest ../macros
    /THEmacro.mac
```

9. Rendimiento *Clúster*

9.1. Variación número de hilos en el *clúster*

- **Simulación:** G4pntest
- **Número de neutrones:** 10e6
- **Energía:** 0.025 eV
- **Variación de Número de hilos con:** open MPI
- **Número de computadores:** 6
- **Número de hilos disponibles:** 48
- **Comando:** $mpirun -np 3 -hostfile ./PNtest THEmacro.mac



9.2. Comparación Clúster vs 1 PC, variando el número de neutrones

- **Simulación:** G4pntest
- **Energía:** 0.025 eV
- **Número de neutrones:** 200e6
- **Número de hilos utilizados en PC:** 8
- **Número de hilos utilizados en el Clúster:** 48

	PC (Geant4MT)	**Clúster (open MPI)**	**Aceleración Clúster**
Tiempo	1 hora 36 minutos	42 minutos	2.3 veces más rápida

9.3. Comparación Resultados de una simulación usando MT y MPI

Simulación con $1x10^6$ neutrones disparados directamente al suelo en un cono con apertura angular de 85 grados:

La energía de los neutrones sigue la distribución de la fuente de 252Cf. El detector es un cilindro con dimensiones $5x100$ cm2 alto x diámetro

- **Simulación:** G4pntest
- **Energía:** 0.025 eV
- **Número de neutrones:** 1e6
- **Número de hilos utilizados con MT:** 60
- **Número de hilos utilizados en el Clúster:** 40