Task 4d Multi-threading in Geant4 (Optional)

INTRODUCTION

Getting prepared

In this task, you will use the same source code as in task 4c (and you will not change a single line of code :-)). Therefore, make sure that you have completed everything and that your code is functional.

PRACTICE

Exercise 4d.1: Observe the output in sequential mode

The only change you have to do in your setup is to change the version of Geant4 you are compiling against - from **sequential** to **multithreaded**. Before you do so, observe the output from running your application:

```
% ./task macros/basiclong.mac
```

Observe the "run summary" and the total User/Real times used for the simulation. Write the numbers down or keep the whole output for future reference in a separate file using redirection:

```
% ./task macros/basiclong.mac > output-sequential.txt
```

Exercise 4d.2: Enable the multi-threaded compilation.

Look at the beginning of **main()** function in main.cc. As you can see, there is a call to the `G4RunManagerFactory`

```
auto* runManager  =
   G4RunManagerFactory::CreateRunManager(G4RunManagerType::Default);
```

which will instantiate the most appropriate type of RunManager (sequential, MT, ...). Until now, we used a sequential installation of Geant4, so the `G4RunManagerType::Default` corresponds to the sequential type.

Let's use now the Geant4 build with the MT enables, so that `G4RunManagerType::Default` corresponds to the MT type. In a build directory (you are strongly suggested to create a new one, called `task4-build-MT`), you should run `cmake` again, this time substituting the Geant4 installation path with a new one (observe that the paths differ only in two letters):

```
% cd ~/tasks-build
% mkdir -p task4-build-MT
% cd task4-build-MT
% cmake -DGeant4_DIR=/usr/local/geant4/geant4-11.0.1MT-install/lib64/Geant4-11-0-1 ~/geant4-exercises/task4
% make
% ./task macros/basiclong.mac
# or ./task macros/basiclong.mac > output-multithreaded.txt
```

(*Notice*: if you are in a MT installation and you want to force the sequential RunManager, just replace `G4RunManagerType::Default` with `G4RunManagerType::Serial`)

Do you observe any changes in the output? If you have set up everything correctly, you should be able to see:

1) Certain messages are duplicated (prepended with either G4WT0 or G4WT1) - this is output from the two parallel **worker threads**.

2) There are now multiple analysis output files (with ntuples), each of them containing "_t0" or "_t1" in their name. For performance reasons, each of the threads writes to a separate file. In the case of histograms, the bin contents are summed after the run (and you will find them in the file with the expected name); in the case of ntuples, no merging is done (and you have to look into multiple files for results).

Do you observe any speed-up? If you are working in the virtual machine, there should not be any (it is defined to have just one core). In fact, perhaps even a little degradation (due to the multithreading overhead) is to be expected.

Exercise 4d.3: Set up the virtual machine to use two cores (Optional)

If you want to observe the speed-up effect of the threads, your (virtual) machine really needs to run with multiple cores. How to do that? (**Note:** If you use a native installation of Geant4 on your computer, skip these instructions)

1) Shutdown the virtual machine (from inside). It is not possible to change the parameters when it's running.

2) In the "Oracle VM VirtualBox Manager" window, select the "CentOS-INFN_2022_Geant4.11" machine. Right click on it and open "Settings". Find the "System" tab, "Processor" sub-tab.

3) The slider "Processor(s)" enables you to select the number of cores that you want available to the virtual machine. Currently, it is set to 1. Try moving the slider to have more virtual CPUs (note that a warning occurs when you go beyond the real number of physical cores). Unless you have a really old machine, 2 CPUs should be a safe choice.

Note: You may also consult the VirtualBox documentation page here: https://www.virtualbox.org/manual/ch03.html

4) Once you did the changes, start the virtual machine again and return back to the task.

Try running the same command as previously:

```
% cd task4-build
% ./task macros/basiclong.mac
```

What do you observe now? Is there any change in the execution (User/Real) times?

Exercise 4d.4: Experiment with the number of threads (Optional)

The multi-threaded mode by default starts two threads. However, this number is configurable - via a macro command and also via environment variable (called `G4FORCENUMBEROFTHREADS`). We will experiment with the latter approach.

Try running the application this way:

```
% export G4FORCENUMBEROFTHREADS=1
% ./task macros/basiclong.mac
```

What happens? Are there still several threads running in parallel? Did we return to the sequential mode?

Now try to set the value of G4FORCENUMBEROFTHREADS to different values (reasonably small integers larger than 0), e.g. 1, 2, 3, 4 (and 16 just of curiosity). Observe carefully the User/Real time measures and tell which of them changes a lot - you can think why. You may also produce some plots in your analysis tool of choice.

Note: The running times and availability of more cores are dependent on your particular laptop so the numbers can differ a lot from the ones presented here and also from your colleagues'.

SOLUTIONS

Exercise 4d.4

Example output:

```
### Run 0 starts.
EventAction: absorber energy/time scorer ID: 0
EventAction: scintillator energy/time scorer ID: 1
 Run terminated.
Run Summary
  Number of events processed : 10000
  User=19.54s Real=20.7s Sys=0.66s
```

Exercise 4d.4

Example output of 2 threads output (single core):

```
### Run 0 starts.
G4WT1 > EventAction: absorber energy/time scorer ID: 0
G4WT1 > EventAction: scintillator energy/time scorer ID: 1
G4WT0 > EventAction: absorber energy/time scorer ID: 0
G4WT0 > EventAction: scintillator energy/time scorer ID: 1
 Run terminated.
Run Summary
  Number of events processed : 10000
  User=21s Real=21.36s Sys=1.59s
```

This is even slower than the sequential mode - no gain from having two cores :-(

Exercise 4d.4

Example output of 2 threads output (more cores):

```
### Run 0 starts.
G4WT1 > EventAction: absorber energy/time scorer ID: 0
G4WT1 > EventAction: scintillator energy/time scorer ID: 1
G4WT0 > EventAction: absorber energy/time scorer ID: 0
G4WT0 > EventAction: scintillator energy/time scorer ID: 1
 Run terminated.
Run Summary
  Number of events processed : 10000
  User=21s Real=11.36s Sys=1.59s
```

This is starting to get better. Athough the speed-up is not perfectly 2.0, it's rather close.

Exercise 4d.4

Example output of 1 thread output in multi-threaded mode. Observe that actually, the simulation slowed down.

```
### Run 0 starts.
G4WT0 > EventAction: absorber energy/time scorer ID: 0
G4WT0 > EventAction: scintillator energy/time scorer ID: 1
 Run terminated.
Run Summary
  Number of events processed : 10000
  User=21.56s Real=22.07s Sys=0.51s
```

Example output of 4 threads output (on a 4-core computer):

```
### Run 0 starts.
G4WT1 > EventAction: absorber energy/time scorer ID: 0
G4WT1 > EventAction: scintillator energy/time scorer ID: 1
G4WT3 > EventAction: absorber energy/time scorer ID: 0
G4WT3 > EventAction: scintillator energy/time scorer ID: 1
G4WT2 > EventAction: absorber energy/time scorer ID: 0
G4WT2 > EventAction: scintillator energy/time scorer ID: 1
G4WT0 > EventAction: absorber energy/time scorer ID: 0
G4WT0 > EventAction: scintillator energy/time scorer ID: 1
 Run terminated.
Run Summary
  Number of events processed : 10000
  User=22.92s Real=6.35s Sys=1.95s
```

Example of output of 8 and 64 threads output. The processor has "only" 4 cores so no speed-up is to be expected. In fact, the execution time even gets a bit slower.

```
### Run 0 starts.
...
Run Summary
  Number of events processed : 10000
  User=22.99s Real=6.42s Sys=2.09s

Run Summary
  Number of events processed : 10000
  User=25.61s Real=7.53s Sys=1.87s
```
