OpenACC in Colab

Sarah

today

1 Part 1: Introduction

This document summarizes the different stages to run a Fortran code using Colab in the first part. In a second part, how to run it using OpenACC.
The aim of this report is to record a guide but also to list some errors encountered doing so.

2 Part 2: Running a Fortran code in Colab

2.1 Installing Colab

In our particular case, we want to work with Fortran. To do so, we have to set up Colab to run with this specific language. First and foremost, we should install a Fortran kernel. One can refer here to get more details about the process.

2.1.1 Cloning

The first step is to clone the jupyter-fortran-kernel from github.

```
$ git clone git@github.com:ZedThree/jupyter-fortran-kernel.git
```

2.1.2 Installation

Once the folder downloaded, one shlould install the kernel fortran using:

```
$ pip install -e --user jupyter-fortran-kernel
```

Then, launch the following commands:

```
$ cd jupyter-fortran-kernel
$ jupyter-kernelspec install fortran_spec/
```

You should get the message when the installation is completed:

```
[InstallKernelSpec] Installed kernelspec fortran_spec in /home/saras/.local/sha
```

2.1.3 Errors Management

SSH key associated to remote
Here an error occured related to the permission. One should then check the access rights.
Here the error message:

```
Cloning into 'jupyter-fortran-kernel'...
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.
Please make sure you have the correct access rights
and the repository exists.
```

Problem solved: the SSH key existing in the local machine was not associated with my Github account.
Check this link to add the key to your account.

Pip install option –user
The package installer pip for Python is used to launch the installation.

```
$ pip install -e --user jupyter-fortran-kernel
```

In my case, this commands gave an error:

```
ERROR: --user is not a valid editable requirement. It should either be a path t
```

To overcome this, one can simply switch the two options of the pip install command such as:

```
$ pip install --user -e jupyter-fortran-kernel
```

Launching from terminal
After installing the kernel, we can launch the notebook by running the command jupyter-notebook from the terminal.
An error can occur:

```
Access to the file was denied .local/share/jupyter/runtime/nbserver-113142-open
```

The problem can happen for versions of notebook > 5.7.2, a security feature measure was added that prevented the authentication token used to launch the browser from being visible. This feature makes it difficult for other users on a multi-user system from running code in your Jupyter session as you.
A way to solve this problem can be found here.

2.2 Compiling & Executing

In our particular case, it is relevant now to introduce Fortran. Because it is a high-level computer langage and more important it is a compiled langage. That means it cannot be run until the compilation stage.
In the other hand, we work in the Jupyter Notebook which is an interactive environment and interpreted oriented.
A way to overcome this incompatibility is to write the Fortran code inside a file, then, to execute the file as a script. To do so, the fortran kernel is no longer required, and we need to switch to the python kernel to use what we call Magics. They are powerful tools provided by the IPython kernel.

- %%writefile

  Writes the content of the cell into a file that should be specify right after the magics command

```
%%writefile get_age.f95

program get_age
    real :: year, age
    print *, 'What year were you born?'
    read *, year
    age = 2022 - year
    print *, 'Your age is', age
end program get_age
```

%%bash

Allow us to run cells with bash.

```
%%bash

gfortran -ffree-form get_age.f95
./a.out
1984
```

2.3 Results

3 Part 2: OpenACC on Colab

Most of what will be presented here comes from this tutorial.