Computación Concurrente, Paralela y Distribuida

Leandro Rodríguez Liñares – David Olivieri

Curso 2024/25

Práctica 1: Introducción

Instalar Anaconda

Descargar e instalar Anaconda

- Descargar anaconda:

  https://www.anaconda.com/download

  (no es necesario registrarse)

- Ejecutar el instalador:

```
$ chmod +x Anaconda3-2024.10-1-Linux-x86_64.sh
$ ./Anaconda3-2024.10-1-Linux-x86_64.sh
```

Importante: contestar “yes” al final de la instalación

El camino en que se instala será: **`/home/my_user/anaconda3`** (6.5GB)

Instalar MPI

- Crear y activar un nuevo entorno en Anaconda

```
$ conda create --name mpi
$ conda activate mpi
```

- Instalar MPI:

```
(mpi) $ conda install ipython jupyter numpy mpi4py openmpi
(mpi) $ conda install -c conda-forge gcc
```

- Comprobar funcionamiento:

```
#!/usr/bin/env python

from mpi4py import MPI

comm = MPI.COMM_WORLD
my_rank = comm.rank
num_processes = comm.size

if my_rank != 0:
   data = "Saludos del proceso {}".format(my_rank)
   comm.send(data, dest=0)

else:
   print("Hola, soy el proceso %d/%d y recibo:" % (my_rank, num_processes))
   for source_rank in range(1,num_processes):
      data_in = comm.recv(source = source_rank)
      print("  "+data_in)
```

```
(mpi) $ chmod +x CheckMPI.py
$ mpirun -np 3 ./CheckMPI.py
Hola, soy el proceso 0/3 y recibo:
  Saludos del proceso 1
  Saludos del proceso 2
$ mpirun --oversubscribe -np 6 ./CheckMPI.py
Hola, soy el proceso 0/6 y recibo:
  Saludos del proceso 1
  Saludos del proceso 2
  Saludos del proceso 3
  Saludos del proceso 4
  Saludos del proceso 5
```

```c
#include <stdio.h>
#include <string.h>
#include "mpi.h"
int main(int argc, char* argv[])
{
    int my_rank;
    int p;
    int source; int dest;
    int tag = 0;
    char message[100];


    MPI_Status status;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
    MPI_Comm_size(MPI_COMM_WORLD, &p);


    if (my_rank != 0) {
        sprintf(message, "Saludos del proceso %d", my_rank);
        dest = 0;
        MPI_Send(message, strlen(message) + 1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
    }
    else {
        printf("Hola, soy el proceso %d (hay %d procesos) y recibo:\n", my_rank, p);
        for (source = 1; source < p; source++) {
            MPI_Recv(message, 100, MPI_CHAR, source, tag, MPI_COMM_WORLD, &status);
            printf("%s\n", message);
        }
    }


    MPI_Finalize();


    return 0;
}
```

```
(mpi) $ mpicc CheckMPI.c -o CheckMPI
$ mpirun -np 4 ./CheckMPI
Hola, soy el proceso 0 (hay 4 procesos) y recibo:
Saludos del proceso 1
Saludos del proceso 2
Saludos del proceso 3
```

Instalación Anaconda: 11GB

Instalar Numba, Cython, NVidia CUDA y Tensorflow

- Crear y activar un nuevo entorno en Anaconda

```
$ conda create --name gpu
$ conda activate gpu
```

- Instalar NVidia CUDA y herramientas:

```
(gpu) $ conda install numba scipy matplotlib
cudatoolkit ipython jupyter cython
```

- Comprobar funcionamiento de GPU:

```
(gpu) $ python3 CheckGPU.py
Found 1 device(s).
Device: 0
  Name: NVIDIA T400 4GB
  Compute Capability: 7.5
  Multiprocessors: 6
  CUDA Cores: 384
  Concurrent threads: 6144
  GPU clock: 1425 MHz
  Memory clock: 5001 MHz
  Total Memory: 3901 MiB
  Free Memory: 3025 MiB
```

Python vs Numba vs Cython

PythonNumbaCython.py

```python
# Cython Function
import pyximport; pyximport.install()
from sum_series import sum_series_cython


#Python Function
def sum_series_python(x):
     y = 0
     for i in range(x):
          y += i
     return y

# Numba Function
from numba import njit
@njit(cache = True)
def sum_series_numba(x):
     y = 0
     for i in range(x):
          y += i
     return y
```

sum_series.pyx

```python
# Cython Function
def sum_series_cython(int x):
     cdef int y = 0
     cdef int i
     for i in range(x):
          y += i
     return y
```

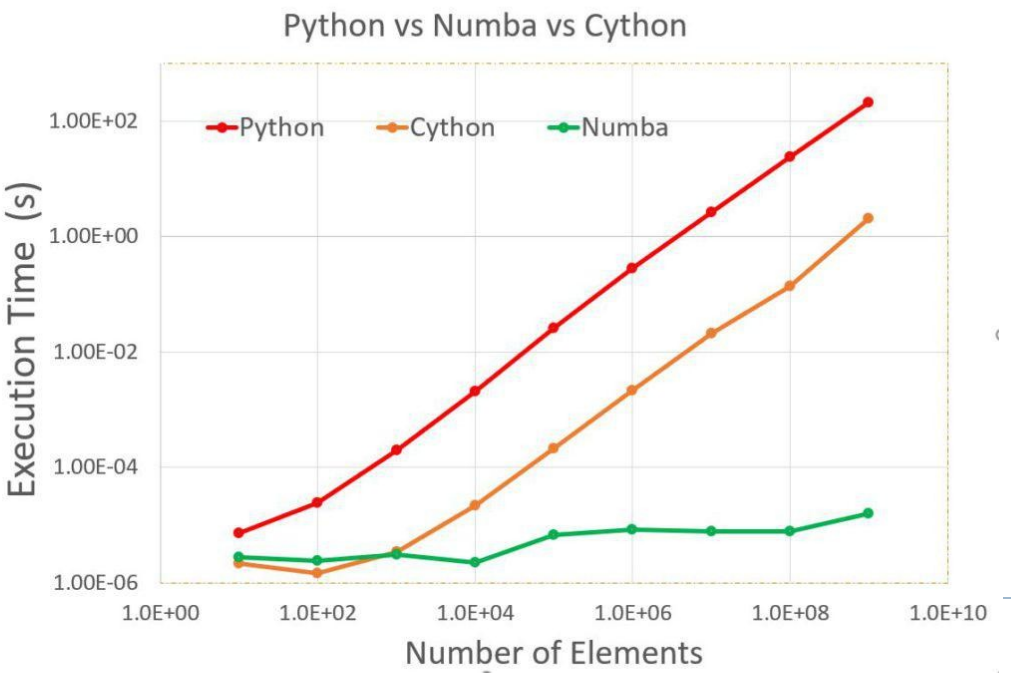
```
(gpu) leandro@T1700:~/Tmp$ ipython
...
In [1]: %run PythonNumbaCython.py
...
In [2]: %timeit sum_series_python(1000000)
32.1 ms ± 433 µs per loop (mean ± std. dev. of 7 runs, 10 loops each)
...
```

1000 elementos

- python 25.2 µs
- cython 301 ns (S = 83.7)
- numba 146 ns (S=172.6)

1000000 elementos

- python 31.8 ms
- cython 261 µs (S = 121.8)
- numba 151 ns (S=210596)



Probar NVidia CUDA

```python
from numba import cuda
import numpy as np

blocks_per_grid = 1000
threads_per_block = 1024
n = threads_per_block*blocks_per_grid

def add_cpu(x, y, out):
    out = x + y;

@cuda.jit
def add_gpu(x, y, out):
    idx = cuda.grid(1)
    out[idx] = x[idx] + y[idx]

h_x = np.ones(n)  # [1...1]
h_y = np.ones_like(h_x)
h_out = np.ones_like(h_x)

d_x = cuda.to_device(h_x)
d_y = cuda.to_device(h_y)
d_out = cuda.device_array_like(d_x)

add_cpu(h_x, h_y, h_out)

# Un hilo para cada elemento
add_gpu[blocks_per_grid, threads_per_block](d_x, d_y, d_out)

print("Número de hilos:", n)
print(d_out.copy_to_host()) # Resultado: [2...2]
```

```
(gpu) $ ipython
...
In [1]: %run Cuda_add.py
Número de hilos: 1024000
[2. 2. 2. ... 2. 2. 2.]
In [2]: %timeit add_cpu(h_x, h_y, h_out)
1.69 ms ± 601 ns per loop (mean ± std. dev. of 7
runs, 1,000 loops each)
In [3]: %timeit add_gpu[blocks_per_grid,
threads_per_block](d_x, d_y, d_out)
359 µs ± 138 ns per loop (mean ± std. dev. of 7 runs,
10,000 loops each)
```

- 1 bloque (1024 hilos):

  S = 0.026
- 100 bloques (102400 hilos):

  S = 2.025
- 10000 bloques (10240000 hilos):

  S = 5.496

Instalación Anaconda: 14GB

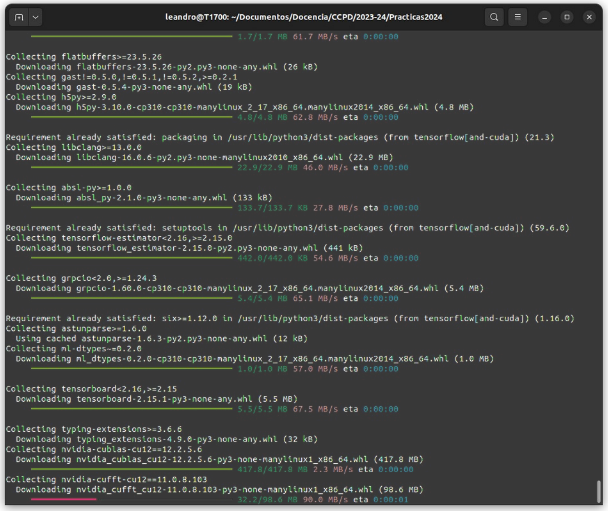
Instalar Tensorflow

- Instalar Tensorflow:

```
(gpu) $ python3 -m pip install tensorflow[and-cuda]
....
```

- Probar instalación:

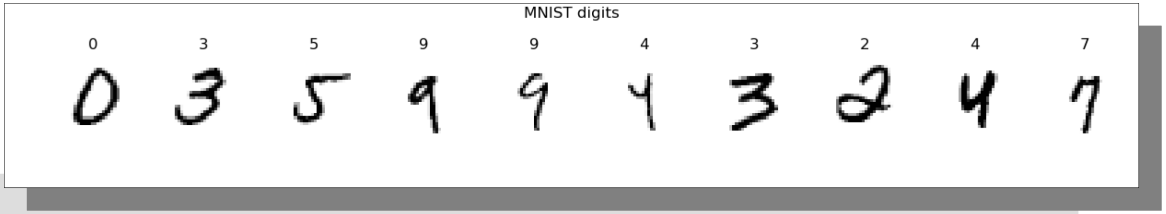
```
(gpu) $ ipython
...
In [1]: import tensorflow as tf
...
In [2]: print("Num GPUs available: ",
len(tf.config.list_physical_devices('GPU')))
...
Num GPUs available:  1
```


leandro@T1700: ~/Documentos/Docencia/CCPD/2023-24/Practicas2024

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.7/1.7 MB 61.7 MB/s eta 0:00:00

Collecting flatbuffers>=23.5.26
  Downloading flatbuffers-23.5.26-py2.py3-none-any.whl (26 kB)
Collecting gast!=0.5.0,!=0.5.1,!=0.5.2,>=0.2.1
  Downloading gast-0.5.4-py3-none-any.whl (19 kB)
Collecting h5py>=2.9.0
  Downloading h5py-3.10.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.8/4.8 MB 62.8 MB/s eta 0:00:00

Requirement already satisfied: packaging in /usr/lib/python3/dist-packages (from tensorflow[and-cuda]) (21.3)
Collecting libclang>=13.0.0
  Downloading libclang-16.0.6-py2.py3-none-manylinux2010_x86_64.whl (22.9 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 22.9/22.9 MB 46.0 MB/s eta 0:00:00

Collecting absl-py>=1.0.0
  Downloading absl_py-2.1.0-py3-none-any.whl (133 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 133.7/133.7 KB 27.8 MB/s eta 0:00:00

Requirement already satisfied: setuptools in /usr/lib/python3/dist-packages (from tensorflow[and-cuda]) (59.6.0)
Collecting tensorflow-estimator<2.16,>=2.15.0
  Downloading tensorflow_estimator-2.15.0-py2.py3-none-any.whl (441 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 442.0/442.0 KB 54.6 MB/s eta 0:00:00

Collecting grpcio<2.0,>=1.24.3
  Downloading grpcio-1.60.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (5.4 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.4/5.4 MB 65.1 MB/s eta 0:00:00

Requirement already satisfied: six>=1.12.0 in /usr/lib/python3/dist-packages (from tensorflow[and-cuda]) (1.16.0)
Collecting astunparse>=1.6.0
  Using cached astunparse-1.6.3-py2.py3-none-any.whl (12 kB)
Collecting ml-dtypes~=0.2.0
  Downloading ml_dtypes-0.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (1.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.0/1.0 MB 57.0 MB/s eta 0:00:00

Collecting tensorboard<2.16,>=2.15
  Downloading tensorboard-2.15.1-py3-none-any.whl (5.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 5.5/5.5 MB 67.5 MB/s eta 0:00:00

Collecting typing-extensions>=3.6.6
  Downloading typing_extensions-4.9.0-py3-none-any.whl (32 kB)
Collecting nvidia-cublas-cu12==12.2.5.6
  Downloading nvidia_cublas_cu12-12.2.5.6-py3-none-manylinux1_x86_64.whl (417.8 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 417.8/417.8 MB 2.3 MB/s eta 0:00:00
Collecting nvidia-cufft-cu12==11.0.8.103
  Downloading nvidia_cufft_cu12-11.0.8.103-py3-none-manylinux1_x86_64.whl (98.6 MB)
     ━━━━━━━━━━━━━                            32.2/98.6 MB 90.0 MB/s eta 0:00:01
```

Instalación Anaconda: 19GB

Probar Tensorflow



```
(gpu) $ python3 ./CheckTF.py
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 512)               401920
 dense_1 (Dense)             (None, 512)               262656
 dense_2 (Dense)             (None, 10)                5130
=================================================================
Total params: 669706 (2.55 MB)
Trainable params: 669706 (2.55 MB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
Epoch 1/10
1875/1875 [==....==] - 5s 2ms/step - loss: 0.1874 - accuracy: 0.9436 - val_loss: 0.1054 - val_accuracy: 0.9683
Epoch 2/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0869 - accuracy: 0.9751 - val_loss: 0.1311 - val_accuracy: 0.9647
Epoch 3/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0606 - accuracy: 0.9829 - val_loss: 0.0810 - val_accuracy: 0.9780
Epoch 4/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0466 - accuracy: 0.9870 - val_loss: 0.0959 - val_accuracy: 0.9777
Epoch 5/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0380 - accuracy: 0.9895 - val_loss: 0.0718 - val_accuracy: 0.9842
Epoch 6/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0295 - accuracy: 0.9923 - val_loss: 0.0963 - val_accuracy: 0.9811
Epoch 7/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0230 - accuracy: 0.9936 - val_loss: 0.1180 - val_accuracy: 0.9801
Epoch 8/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0199 - accuracy: 0.9946 - val_loss: 0.1119 - val_accuracy: 0.9812
Epoch 9/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0148 - accuracy: 0.9959 - val_loss: 0.1149 - val_accuracy: 0.9820
Epoch 10/10
1875/1875 [==....==] - 4s 2ms/step - loss: 0.0132 - accuracy: 0.9963 - val_loss: 0.1293 - val_accuracy: 0.9810
```

Instalar PyTorch

PyTorch

- Instalar PyTorch:

```
(gpu) $ conda install pytorch torchvision torchaudio pytorch-cuda -c pytorch -c nvidia
....
```

- Probar instalación:

```
(gpu) $ ipython
...
In [1]: import torch
...
In [2]: print(torch.__version__)
2.5.1

In [3]: torch.cuda.is_available()
Out[3]: True

In [4]: torch.cuda.device_count()
Out[4]: 1

In [5]: torch.cuda.get_device_name(0)
Out[5]: 'NVIDIA T400 4GB'
```

Probar Pytorch

```
(gpu) $ python3 ./CheckTF.py
...
Extracting /tmp/data/MNIST/raw/t10k-labels-idx1-ubyte.gz to /tmp/data/MNIST/raw

Length of dataset: 60000
Length of first vector in dataset:  torch.Size([784])
Label of first vector in dataset:  5
Length of Train Dataset:  50000
Length of Test Dataset:  10000
Sequential(
  (0): Linear(in_features=784, out_features=512, bias=True)
  (1): ReLU()
  (2): Linear(in_features=512, out_features=10, bias=True)
  (3): Softmax(dim=1)
)
Epoch 01: loss 1.5777 - accuracy 0.8966 - validation accuracy 0.9294
Epoch 02: loss 1.5177 - accuracy 0.9496 - validation accuracy 0.9497
Epoch 03: loss 1.5033 - accuracy 0.9623 - validation accuracy 0.9548
Epoch 04: loss 1.4942 - accuracy 0.9702 - validation accuracy 0.9672
Epoch 05: loss 1.4884 - accuracy 0.9761 - validation accuracy 0.9626
Epoch 06: loss 1.4841 - accuracy 0.9798 - validation accuracy 0.9728
Epoch 07: loss 1.4809 - accuracy 0.9825 - validation accuracy 0.9698
Epoch 08: loss 1.4786 - accuracy 0.9844 - validation accuracy 0.9746
Epoch 09: loss 1.4763 - accuracy 0.9865 - validation accuracy 0.9699
Epoch 10: loss 1.4747 - accuracy 0.9880 - validation accuracy 0.9764
```

Instalación Anaconda: 27GB

Borrar anaconda

```
(gpu) $ conda activate base
(base) $ conda init --reverse –all
...
==> For changes to take effect, close and re-open your current shell. <==

$ rm -rf ~/anaconda3
$ rm -rf ~/.condarc ~/.conda
```