DEEPHEALTH

Deep-Learning and HPC to Boost Biomedical Applications for Health

HPC-Lab

Eduardo Quiñones, Sergi Albiach
{eduardo.quinones, sergi.albiach}@bsc.es
BSC

Iacopo Colonnelli, Barbara Cantalupo
{iacopo.colonnelli, barbara.cantalupo}@unito.it
UNITO

DeepHealth Winter School – 24 January 2022

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Objectives of the HPC-Lab

1. Extract the parallelism exposed by a set of simple benchmarks and parallelise them with COMPSs
2. Identify the parallelism exposed by three data-parallelism training strategies: *Synchronous*, *Relaxed Synchronous*, *Removed Synchronous*
3. Execute performance experiments on a parallel environment composed of 2, 4, 8 and 16 GPUs

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Agenda

1. Kubernetes Environment Setup
2. Parallel exercises with simple benchmarks
3. Identify the parallelism exposed by training operations
4. Execute performance experiments

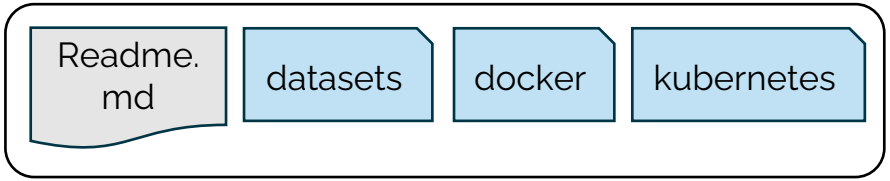
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Kubernetes Environment Setup

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Environment

Execution Environment Setup



From https://gitlab.bsc.es/ppc-bsc/software/deep-health-compss

- **Readme.md → Instructions to deploy the pods and start a run**
- datasets → folder with the datasets that are used in the docker image
- docker → folder including
  - **pyeddl →** folder including source-code of the training operation,
  - **pyeddl/simple_examples →** folder including simple benchmarks to be parallelised with COMPSs
  - **compss →** folder including Configuration scripts
  - Dockerfile and Makefile to create the image
- **kubernetes → folder with yaml file to configure the cluster**

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Deployment

Execution Environment Setup

DEEPHEALTH

1. Go to the "*kubernetes*" folder
2. Execute the command: "**`kubectl create -f compss_deephealth.yaml`**" *
3. Wait until the pods are correctly initialized (**`Running`**)
4. You can see the state of the pods with the command: "**`kubectl get pods`**"

```
NAME                      READY   STATUS    RESTARTS   AGE
compss-5c96c86774-2ck4d   1/1     Running   0          4m16s
compss-5c96c86774-42r5v   1/1     Running   0          4m16s
compss-5c96c86774-fkdtm   1/1     Running   0          4m16s
compss-5c96c86774-gfxgr   1/1     Running   0          4m16s
compss-5c96c86774-jl7jj   1/1     Running   0          4m16s
compss-5c96c86774-q947c   1/1     Running   0          4m16s
compss-5c96c86774-rqlf5   1/1     Running   0          4m16s
compss-5c96c86774-wlc8p   1/1     Running   0          4m16s
compss-rs-dxxps           1/1     Running   0          4m16s
```

- *you'll see the number of worker replicas set to 16 but less workers will be created to fit the maximum number of available pods*

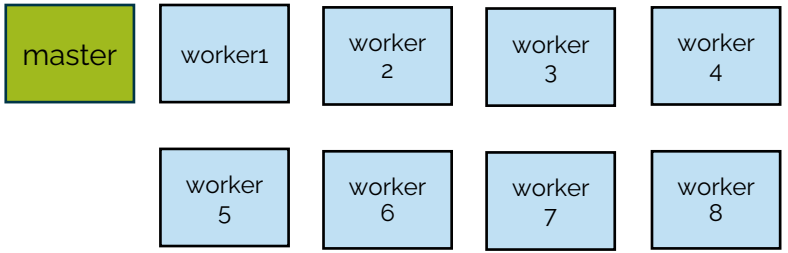
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Deployment

Execution Environment Setup

- The resultant computing infrastructure for the example shown belong are 1 master node and 8 worker nodes (**9 kubernetes pods**)
  - Each pod is composed of **an Intel 12-core and a NVIDIA A40 GPU**
  - Workers are named like: `compss-**********_*****`
  - Master is named like: `compss-rs-*****`

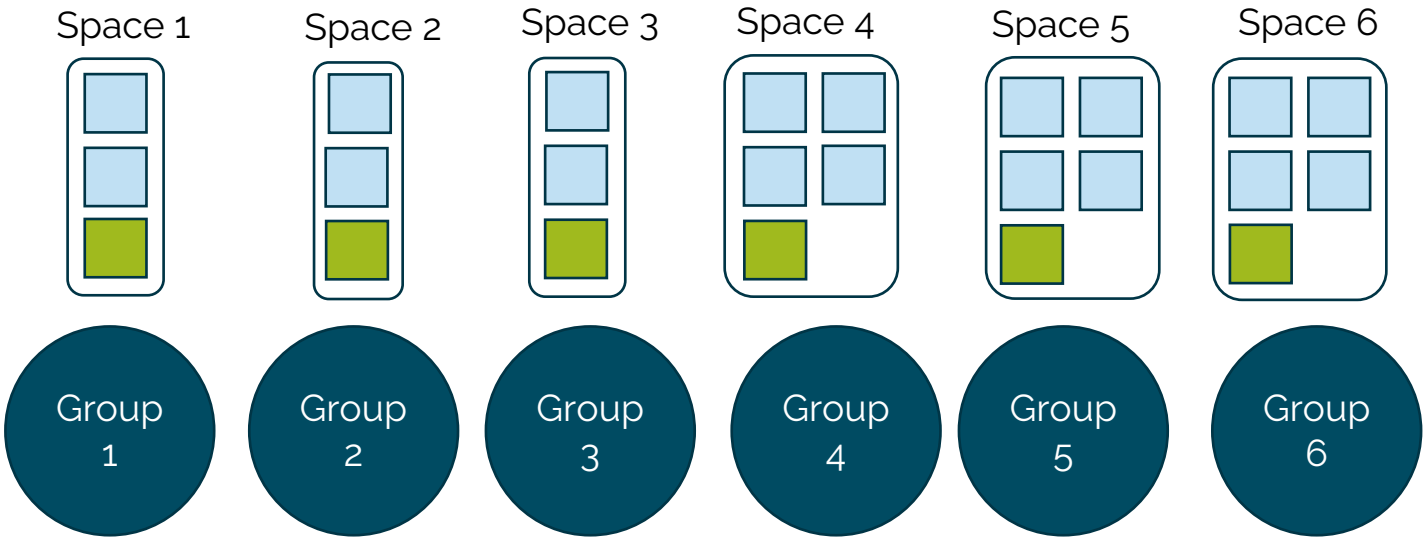
```
NAME                      READY   STATUS    RESTARTS   AGE
compss-5c96c86774-2ck4d   1/1     Running   0          4m16s
compss-5c96c86774-42r5v   1/1     Running   0          4m16s
compss-5c96c86774-fkdtm   1/1     Running   0          4m16s
compss-5c96c86774-gfxgr   1/1     Running   0          4m16s
compss-5c96c86774-jl7jj   1/1     Running   0          4m16s
compss-5c96c86774-q947c   1/1     Running   0          4m16s
compss-5c96c86774-rqlf5   1/1     Running   0          4m16s
compss-5c96c86774-wlc8p   1/1     Running   0          4m16s
compss-rs-dxxps           1/1     Running   0          4m16s
```



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Initial Configuration

Execution Environment Setup



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Deployment

Execution Environment Setup

- Every pod, master or worker, has the same deployed image containing, among other files:
  - The configuration script (**configure_compss.sh**),
  - The folder containing the code that will be executed (**pyeddl, pyeddl/simple_examples**)
  - A run script (**runcompss.sh**)
- The image starts in a conda environment that has been created with the following main packages:
  - Python 3.6
  - PYCOMPSs 2.8
  - PYEDDL 1.1

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Execution: Master Initialization

Execution Environment Setup

DEEPHEALTH

- In order to enter the master pod, the following command should be executed (replace * with master's name):
  "**`kubectl exec -it compss-rs-***** -- /bin/bash`**"
- Now, you are inside the master. The worker's IPs must be added to a "*project.xml*" and "*resources.xml*" files so that COMPSs is able to locate the pods and distribute the tasks. This is done with the command:
  "**`bash configure_compss.sh`**"

IPs of the distributed computing infrastructure (pods)

```
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4713    0  4713    0     0   164k      0 --:--:-- --:--:-- --:--:--  164k
Pods IP's are:
10.1.107.36 10.1.120.31 10.1.2.39 10.1.23.35 10.1.30.41 10.1.42.34 10.1.60.27 10.1.71.22 10.1.88.34
10.1.107.36
10.1.120.31
10.1.2.39
10.1.23.35
10.1.30.41
10.1.42.34
10.1.60.27
10.1.71.22
10.1.88.34
```


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

COMPSs execution

Execution Environment Setup

- The run will be handled by the "*runcompss.sh*" script. It can be modified, if desired, to change the dataset, the network, the number of epochs and the number of workers *. The following options are accepted:
  - **--dataset=** "*mnist*" – "*cifar10*"
  - **--network=** "*simple-mnist*" – "*lenet*" – "*vgg16*"
  - **--num_epochs=** any integer
  - **--num_workers=** 1 – 2 – 4 – 8
  - **--sync_type=** 0 – 1 – 2 (sync, async, full-async)
- Once the parameters are correctly defined, execute the following command to run the code:
  "**`bash runcompss.sh`**"

* *to modify the file in the master you can use vi like: "**vi runcompss.sh**" Press **Insert** to enable modifications, make your changes and press the sequence **Esc** - **:wq** to save and close the edit view.*

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

COMPSs execution

Execution Environment Setup

- Once the parameters are correctly defined, execute the following command to run the code:
  "**`bash runcompss.sh`**"
  - **The first time the code is executed it is usual to face a wait of 5 minutes.** This is because the model is being built first in the master and then in parallel in the workers.

```
Generating Random Table
CS with full memory setup
Building model
Selecting GPU device 0
EDDLL is running on GPU device 0, NVIDIA A40
```

```
EDDLL is running on GPU device 0, NVIDIA A40
CuBlas initialized on GPU device 0, NVIDIA A40
CuRand initialized on GPU device 0, NVIDIA A40
Building the model in distributed computing units...
```


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

COMPSs execution

Execution Environment Setup

- After the building, the model is trained and tested against the test set

```
(pyeddl_pycompss_env) root@compss-master:~# bash runcompss.sh
MasterIP is: 10.1.88.35
[  INFO] Using default execution type: compss

---------------- Executing eddl_train_batch.py -------------------------

WARNING: COMPSs Properties file is null. Setting default values
[(378)    API]  -  Starting COMPSs Runtime v2.8 (build 20201111-1150.r6c55999b5b159648fc29
2c885d43d51cd4648f12)
Generating Random Table
CS with full memory setup
Building model
Selecting GPU device 0
EDDLL is running on GPU device 0, NVIDIA A40
CuBlas initialized on GPU device 0, NVIDIA A40
CuRand initialized on GPU device 0, NVIDIA A40
Building the model in distributed computing units...
Building done!
--------------------------------------------------------------------------------
model
--------------------------------------------------------------------------------
input2    |  (3, 32, 32)         =>   (3, 32, 32)         0
conv2d1   |  (3, 32, 32)         =>   (20, 32, 32)        1520
relu1     |  (20, 32, 32)        =>   (20, 32, 32)        0
maxpool2d2|  (20, 32, 32)        =>   (20, 16, 16)        0
conv2d2   |  (20, 16, 16)        =>   (50, 16, 16)        25050
relu2     |  (50, 16, 16)        =>   (50, 16, 16)        0
maxpool2d4|  (50, 16, 16)        =>   (50, 8, 8)          0
reshape1  |  (50, 8, 8)          =>   (3200)              0
dense1    |  (3200)              =>   (500)               1600500
relu3     |  (500)               =>   (500)               0
dense2    |  (500)               =>   (10)                5010
softmax4  |  (10)                =>   (10)                0
--------------------------------------------------------------------------------
Params: 1632080

DisLib array: ds-array(blocks=(...), top_left_shape=(6250, 3, 32, 32), reg_shape=(6250, 3,
 32, 32), shape=(50000, 3, 32, 32), sparse=False)
DisLib array: ds-array(blocks=(...), top_left_shape=(6250, 10), reg_shape=(6250, 10), shap
e=(50000, 10), sparse=False)
MODEL TRAINING...
Num workers:  8
Number of epochs:  10
Number of asynchronous epochs:  1

Training epochs:  1  to 1
```

```
Training epochs:  1  to 1

Elapsed time for epoch range (1-1):  16.36  seconds

Training epochs:  2  to 2

Elapsed time for epoch range (2-2):  7.8  seconds

Training epochs:  3  to 3

Elapsed time for epoch range (3-3):  7.48  seconds

Training epochs:  4  to 4

Elapsed time for epoch range (4-4):  7.23  seconds

Training epochs:  5  to 5

Elapsed time for epoch range (5-5):  7.89  seconds

Training epochs:  6  to 6

Elapsed time for epoch range (6-6):  6.29  seconds

Training epochs:  7  to 7

Elapsed time for epoch range (7-7):  6.78  seconds

Training epochs:  8  to 8

Elapsed time for epoch range (8-8):  5.8  seconds

Training epochs:  9  to 9

Elapsed time for epoch range (9-9):  9.06  seconds

Training epochs:  10  to 10

Elapsed time for epoch range (10-10):  6.86  seconds
Total elapsed time:  81.56  seconds
Evaluating model against test set
Evaluate with batch size 10
[XXXXXXXXXXXXXXXXXXXX] 1000 softmax4[loss=1.690 metric=0.428]
[(229995)    API]  -  Execution Finished

------------------------------------------------------------
(pyeddl_pycompss_env) root@compss-master:~#
```


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Termination

- The execution of a run can be stopped by pressing **Ctrl+C**. Make sure that after this kind of abrut stopping you execute the command "**compss_clean_procs**", just in case any compss process did not end correctly.
- To exit the pod just type "**exit**"
- To end the deployment, exit the pod and in the "*kubernetes*" folder execute the command:
  "**`kubectl delete -f compss_deephealth.yaml`**".
- Take into account that this process may take several seconds, so it is recommended to wait until no pods are deployed before starting a new deployment. Remember that you can see the state of the pods with the command: "**`kubectl get pods`**"

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Parallel exercises with simple benchmarks

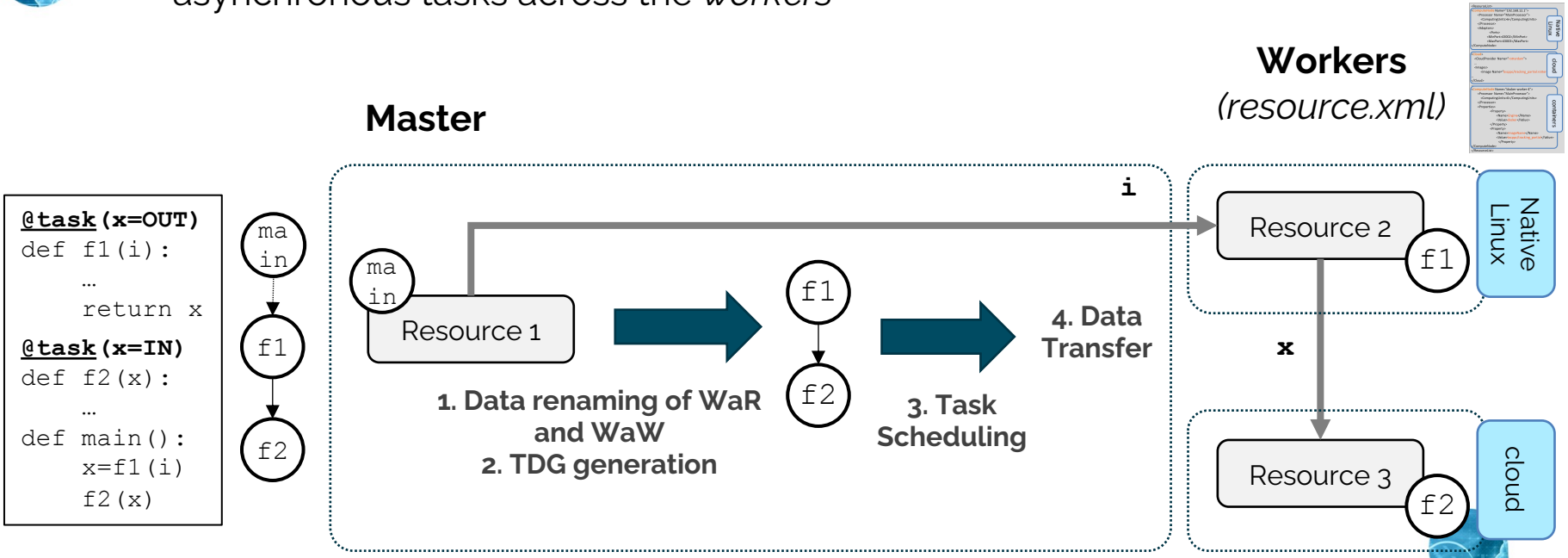
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Memory Model and Parallel distribution

COMPSs Task Model

DEEPHEALTH

- The *master* maintains the *memory consistency* and *distributes* the asynchronous tasks across the *workers*



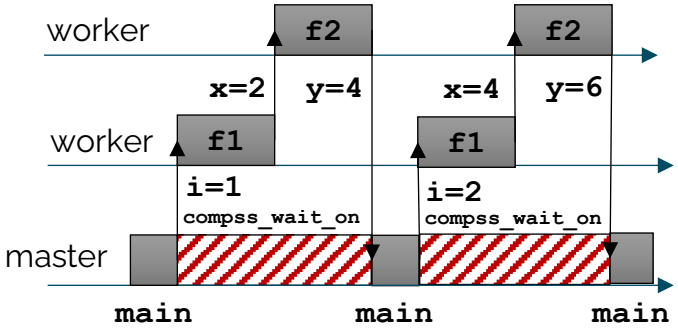
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Memory Model and Parallel distribution

COMPSs Task Model

- **`@task`** (python decorator) to describe parallel units
- **`compss_wait_on`** (COMPSs runtime call) for coarse-grain synchronization
- **`IN,OUT,INOUT`** data dependencies (python decorator) for fine-grain synchronization

```
@task(x=OUT)
def f1(i):
    return i*2
@task(x=IN)
def f2(x):
    return x+2
def main():
    for i in [1..2]
        x=f1(i)
        y=f2(x)
        compss_wait_on(y)
```



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Exercise 1

Parallel exercises with simple benchmarks

DEEPHEALTH

1. Identify the parallelism exposed by the **simple1.py** (available in *pyeddl/simple_examples* folder) by:
   a. Identifying the parallel units
   b. Identifying the synchronization dependencies
2. Parallelise the program using **4 COMPSs tasks**
3. Edit the "*runcompss.sh*" script to invoke **simple1.py** and run the script as follows:
   - "**`bash runcompss.sh`**"

```
def f1(val):
  return val + 1

def f2(val):
  return val + 2

def f3(val):
  return val + 3

def f4(val1, val2, val3):
  return val1 + val2 + val3

if __name__ == "__main__":
  val1 = f1(1)
  val2 = f2(2)
  val3 = f3(3)

  sum = f4(val1, val2, val3)
  print(sum)
```


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Exercise 2

Parallel exercises with simple benchmarks

DEEPHEALTH

1. Identify the parallelism exposed by the **simple2.py** (available in *pyeddl/simple_examples* folder) by:
   a. Identifying the parallel units
   b. Identifying the synchronization dependencies
2. Parallelise the program using **3 COMPSs tasks**
3. Edit the "*runcompss.sh*" script to invoke **simple2.py** and run the script as follows:
   - "`bash runcompss.sh`"

```
def f1(val):
  return val + 1

def f2(val):
  return val + 2

def f3(val):
  return val + 3

def f4(val1, val2, val3):
  return val1 + val2 + val3

if __name__ == "__main__":
  val1 = f1(1)
  val2 = f2(2)
  val3 = f3(3)

  sum = f4(val1, val2, val3)
  print(sum)
```


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Exercise 3

Parallel exercises with simple benchmarks

DEEPHEALTH

1. Identify the parallelism exposed by the **simple3.py** (available in *pyeddl/simple _examples* folder) by:
   a. Identifying the parallel units
   b. Identifying the synchronization dependencies
2. Parallelise the program
3. Edit the "*runcompss.sh*" script to invoke **simple3.py** and run the script as follows:
   - "**`bash runcompss.sh`**"

```
def increment(val):
  return val + 1


if __name__ == "__main__":
arr = [1,2,3,4,5]

for i in range(len(arr)):
  arr[i] = increment(arr[i])

print(arr)
```


The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Exercise 4

Parallel exercises with simple benchmarks

1. Identify the parallelism exposed by the **simple4.py** (available in *pyeddl/simple _examples* folder) by:
   a. Identifying the parallel units
   b. Identifying the synchronization dependencies
2. Parallelise the program
3. Edit the "*runcompss.sh*" script to invoke **simple4.py** and run the script as follows:
   - "**`bash runcompss.sh`**"

```
def initialize_variables():
…

def multiply(a, b, c):
  import numpy as np
  c += a*b

if __name__ == "__main__":
  …

  initialize_variables()

  for i in range(MSIZE):
    for j in range(MSIZE):
      for k in range(MSIZE):
        multiply(A[i][k], B[k][j], C[i][j])


  for i in range(MSIZE):
    for j in range(MSIZE):
      print ("C" + str(i) + str(j) + "=" +
             str(C[i][j]))
```

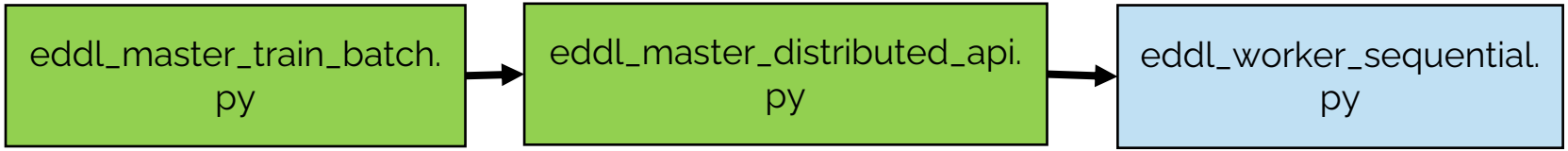
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Identify the parallelism exposed by training operations

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Code Structure

Identify the parallelism exposed by training operations



- ***eddl_master_train_batch.py,*** loads the dataset, builds the model and invokes the ***fit*** processes:
  - *fit_sync*, corresponding to *Synchronous Parameter Averaging*
  - *fit_async*, corresponding to Relaxing Synchronzation
  - *fit_full_async*, corresponding to *Removing Synchronzation*
- ***eddl_master_distributed_api.py,*** implements the *fit* processes by iterating over a number of predefined *epochs* and *workers*, and invoking *train_batch* and *aggregation_paramaters* processes
- ***eddl_worker_sequential.py,*** invokes the eddl functions *build* and *train_batch*

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

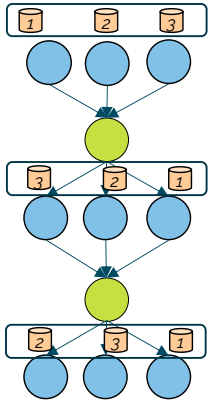
Exercise 5

Identify the parallelism exposed by training operations

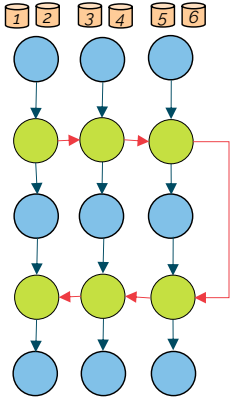
DEEPHEALTH

1. Identify the parallelism exposed by the ***fit_sync, fit_async, fit_full_async*** functions (available in *pyeddl/eddl_master_distributed_api.py* and invoking *pyeddl/eddl_worker_sequential.py*) by:
   a. Identifying the parallel units
   b. Identifying the synchronization dependencies
2. Determine which functions should be executed by the master node and which by the worker node

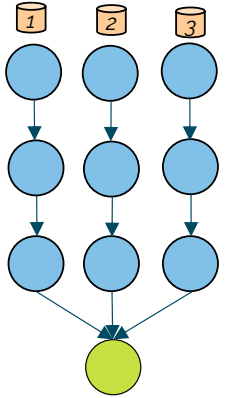
fit_sync (*Synchronous Parameter Averaging*)



fit_async (*Relaxing Synchronzation*)



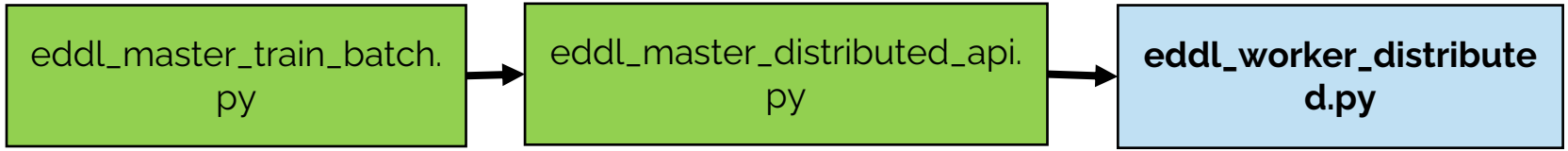
fit_full_async Removing Synchronzation



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Exercise 5: Solution

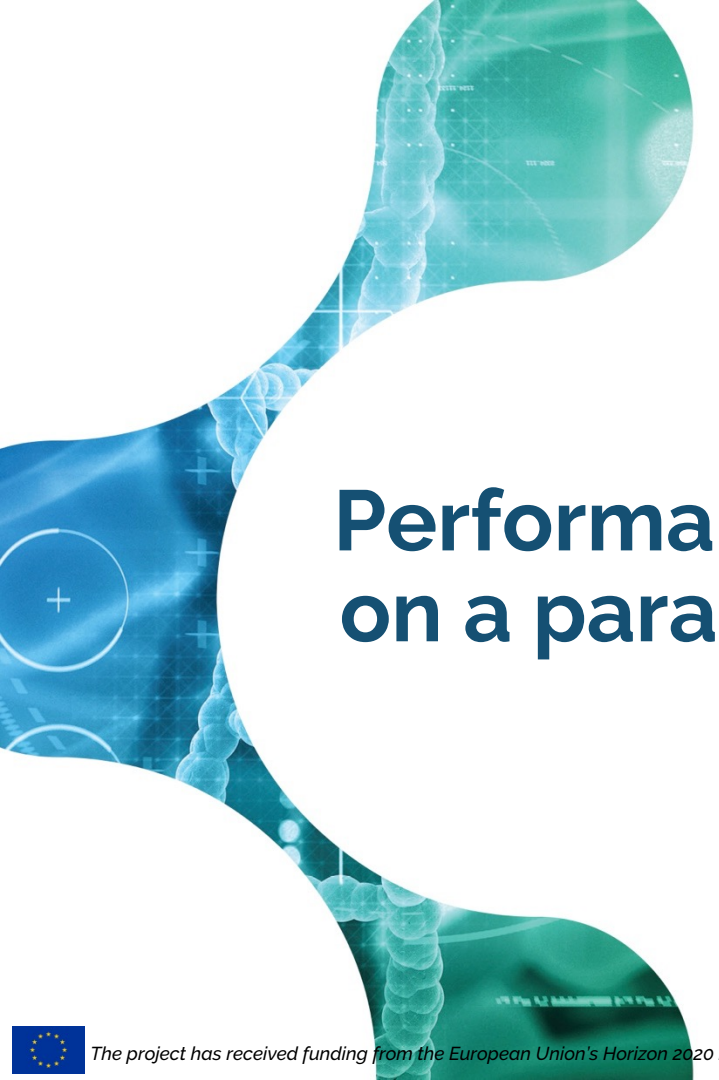
Identify the parallelism exposed by training operations



- ***eddl_master_train_batch.py,*** loads the dataset, builds the model and invokes the ***fit*** processes:
  - *fit_sync*, corresponding to *Synchronous Parameter Averaging*
  - *fit_async*, corresponding to Relaxing Synchronzation
  - *fit_full_async*, corresponding to *Removing Synchronzation*
- ***eddl_master_distributed_api.py,*** implements the *fit* processes by iterating over a number of predefined *epochs* and *workers*, and invoking *train_batch* and *aggregation_paramaters* processes
- ***eddl_worker_distributed.py,*** invokes the eddl functions *build* and *train_batch*

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

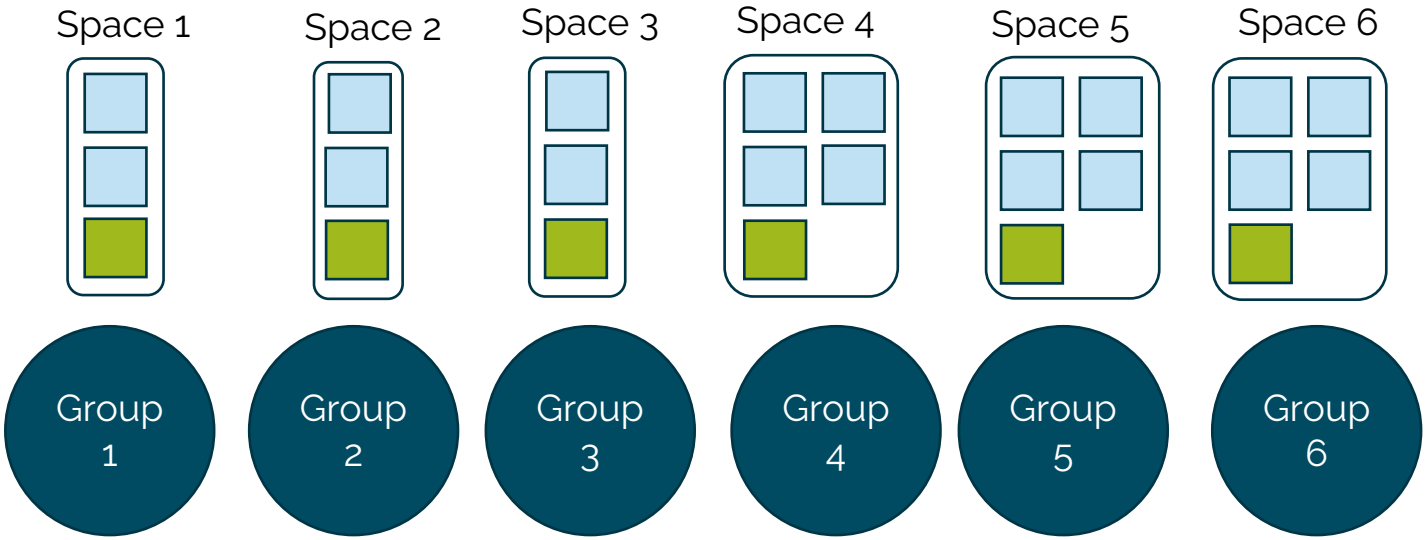
Performance experiments on a parallel environment



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Setup 1 (Initial Configuration)

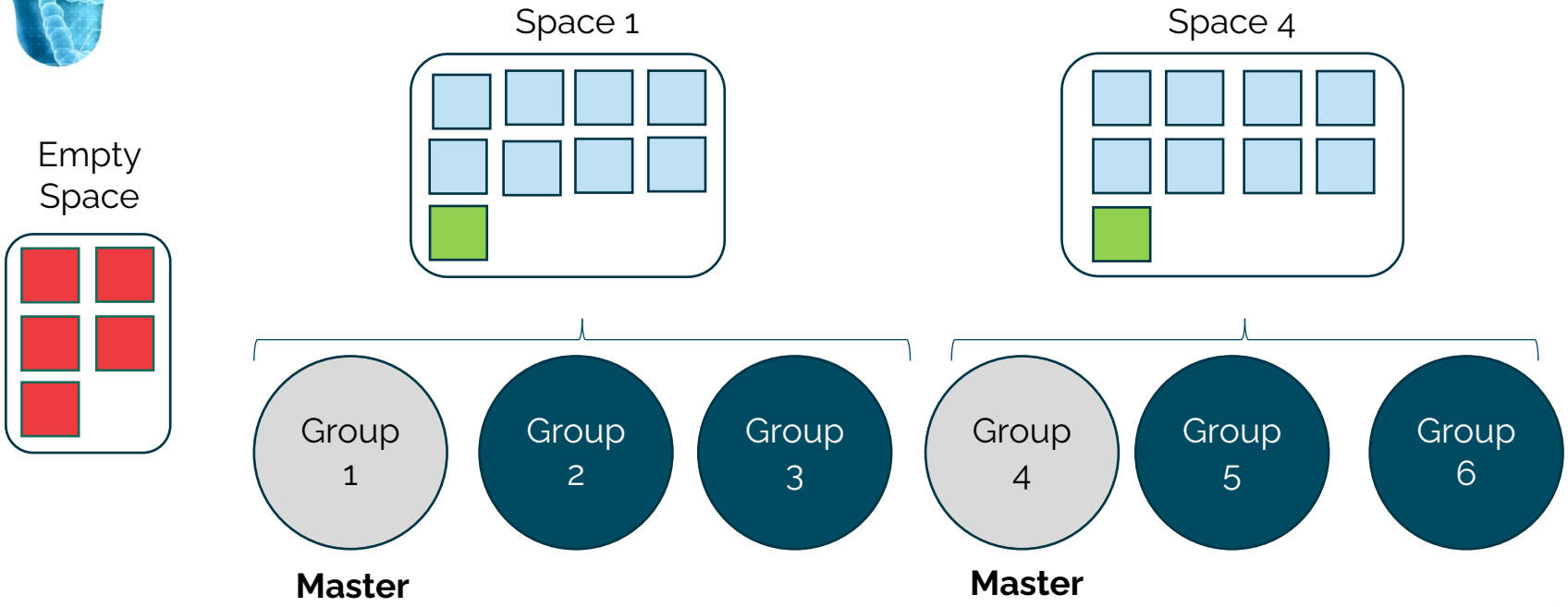
Performance experiments on a parallel environment



The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Setup 2

Performance experiments on a parallel environment

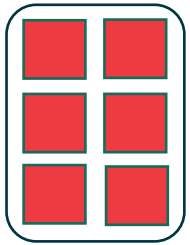


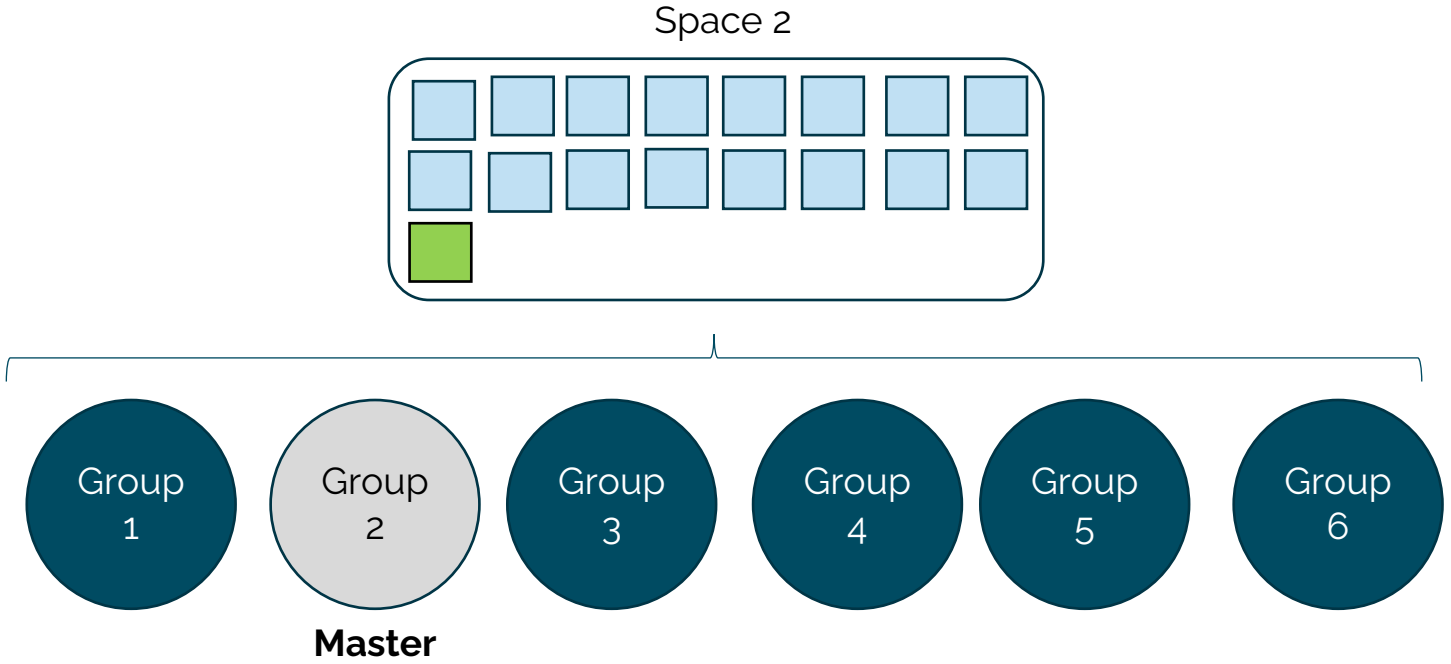
(At every new setup configuration, the environment needs to be configured again; see slides 6 to 10)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Setup 3

Performance experiments on a parallel environment





(At every new setup configuration, the environment needs to be configured again; see slides 6 to 10)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

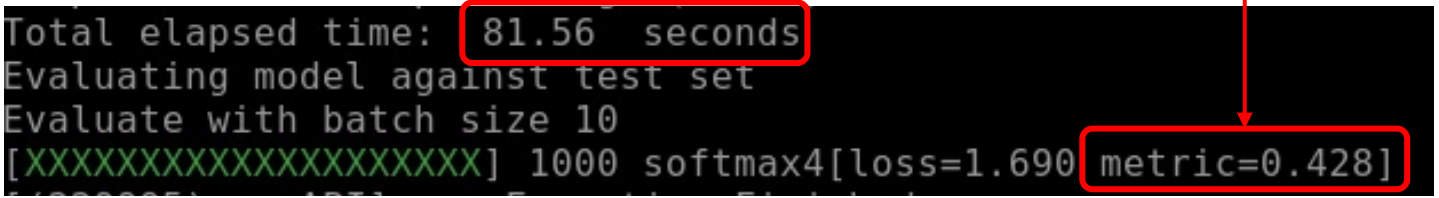
Experiments to be conducted

	Setup 1	Setup 2	Setup 3
Group 1	--num_workers=1/2 --sync_type= 0	--num_workers=8 --sync_type= 0/1	
Group 2	--num_workers=1/2 --sync_type= 1		--num_workers=16 --sync_type= 0/1/2
Group 3	--num_workers=1/2 --sync_type= 2		
Group 4	--num_workers=1/4 --sync_type= 0	--num_workers=8 --sync_type= 2	
Group 5	--num_workers=1/4 --sync_type=1		
Group 6	--num_workers=1/4 --sync_type=1		

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Experiments to be conducted

- Conduct **two executions** per experiment collecting:
  - Model accuracy
  - Total elapsed time



- Report the experiments here:

https://docs.google.com/spreadsheets/d/10CZWpdG9ZnYGkuVOiDofHfAsLgPLNvilAFNDoUlxgl4/edit?usp=sharing

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

COMPSs execution

Execution Environment Setup

- Configure "*runcompss.sh*" script with the following options:
  - **--dataset=** "*cifar10*"
  - **--network=** "*lenet*"
  - **--num_epochs=** 10
  - **--num_workers=** 1 – 2 – 4 – 8 – 16
  - **--sync_type=** 0 – 1 – 2 (sync, async, full-async)
- Execute the train:
  "**`bash runcompss.sh`**"

* *to modify the file in the master you can use vi like: "**vi runcompss.sh**" Press **Insert** to enable modifications, make your changes and press the sequence **Esc** - **:wq** to save and close the edit view.*

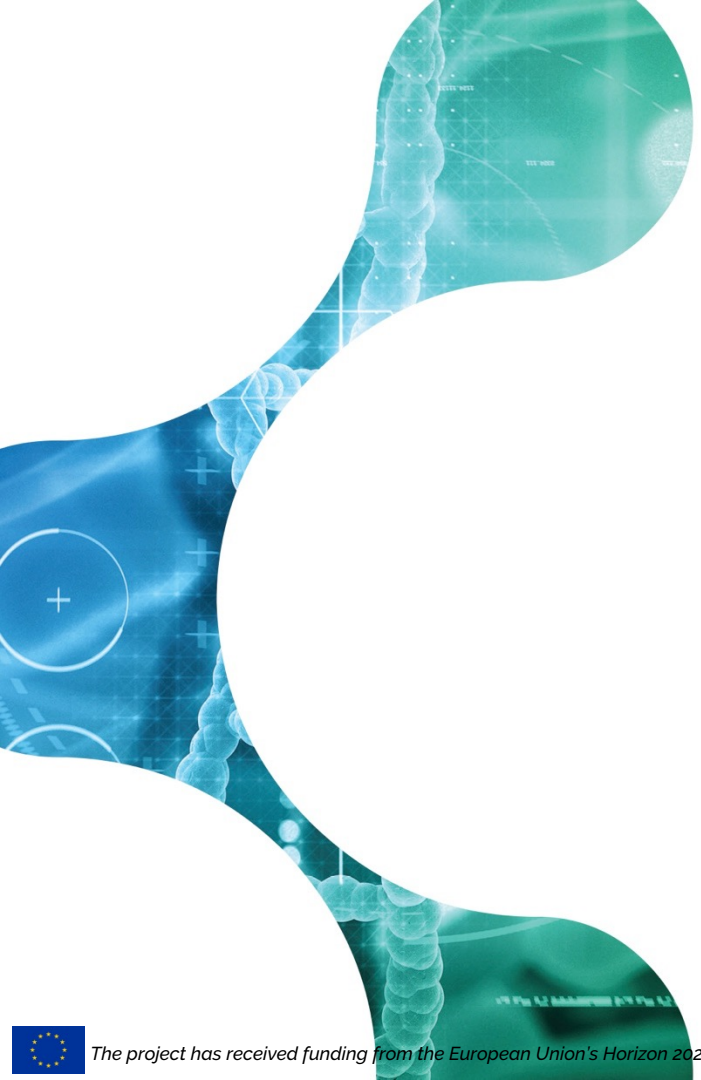
The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.

Evaluation

Let's comment the experiments!

for sure there will be many surprises… ;)

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.



DEEPHEALTH

Thank you!

Eduardo Quiñones

eduardo.quinones@bsc.es

The project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 825111.