Python i strojno učenje na resursu Supek

Kvakić, Marko

Python i strojno učenje na resursu Supek

Sveučilišni računski centar (Srce)

6. listopad 2023.

Europska unija
"Zajedno do fondova EU"
EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI
Operativni program KONKURENTNOST I KOHEZIJA

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Sadržaj

- **Python i Lustre**
  - Python, pip i conda
  - Kako Lustre funkcionira?
- **Apptainer**
  - Uvod
  - Izgradnja na Supeku
  - Izvršavanje
- **Strojno učenje na Supeku**
  - Aplikacije i performanse
  - Način implementacije
- **Primjeri**

Europska unija "Zajedno do fondova EU"
EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI
Operativni program KONKURENTNOST I KOHEZIJA
Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

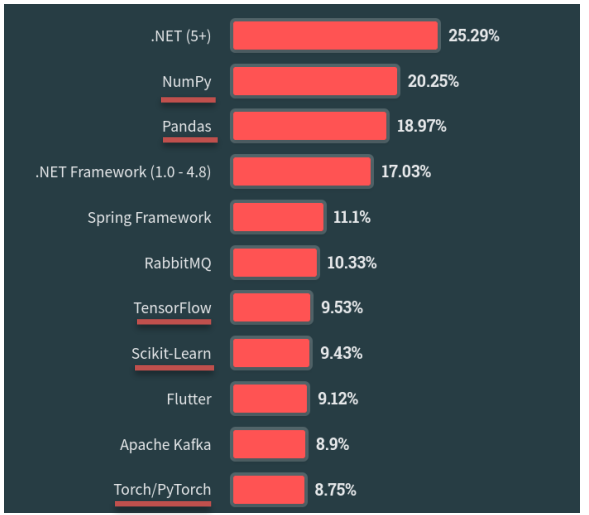
Python i Lustre

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Python i Lustre

Python, pip i conda

- Bogat ekosistem
  - znanstveno računanje
  - podatkovna znanost
  - strojno učenje
- Instalacijski upravitelji
  - pip
  - python –m venv
  - conda
- Osiguravaju "bezbolnu" instalaciju knjižnica i ovisnosti



Slika 1 Rezultati Stack Overflow ankete iz 2023 (izvor) za najkorištenije programske jezike (gore) i 'ostala' programska sučelja (dolje).

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Python i Lustre

Python, pip i conda

```
korisnik@kompjuter:~ python3 –m venv venv-dir

korisnik@kompjuter:~ source venv-dir/bin/activate

(venv-dir) korisnik@kompjuter:~ pip install numpy scipy pandas matplotlib
ipython
Collecting numpy
…
Successfully installed ...

(venv-dir) korisnik@kompjuter:~ pip list
Package            Version
------------------ ------------
asttokens          2.4.0
backcall           0.2.0
contourpy          1.1.1
cycler             0.11.0
decorator          5.1.1
…

(venv-dir) korisnik@kompjuter:~ python moja-skripta.py
...
```

1) Napravi virtualno okruženje **venv-dir**
2) Aktiviraj virtualno okruženje
3) Instaliraj knjižnice
4) Ispiši instalirane knjižnice
5) Izvrši skriptu python

Kod 1 Primjer instalacije knjižnica python korištenjem virtualnog okruženja

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Python i Lustre

Python, pip i conda

```
korisnik@kompjuter:~ find venv-dir -type f | wc -l
13863

korisnik@kompjuter:~ find venv-dir -type d | wc -l
1325

korisnik@kompjuter:~ du -bs venv-dir
373471553 venv-dir

korisnik@kompjuter:~ find venv-dir -type f | xargs ls -sS | ls -sS
34304 venv-dir/lib/python3.9/site-packages/numpy.libs/libopenblas64_p-...
6436  venv-dir/lib/python3.9/site-packages/kiwisolver/_cext.cpython-39...
5576  venv-dir/lib/python3.9/site-packages/matplotlib/ft2font.cpython-...
3348  venv-dir/lib/python3.9/site-packages/matplotlib/backends/_backen...
2624  venv-dir/lib/python3.9/site-packages/numpy.libs/libgfortran-0400...
2364  venv-dir/lib/python3.9/site-packages/matplotlib/_image.cpython-3...
2348  venv-dir/lib/python3.9/site-packages/matplotlib/_qhull.cpython-3...
1780  venv-dir/lib/python3.9/site-packages/matplotlib/_path.cpython-39...
740   venv-dir/lib/python3.9/site-packages/matplotlib/mpl-data/fonts/t...
688   venv-dir/lib/python3.9/site-packages/matplotlib/mpl-data/fonts/t...
```

1) Broj datoteka u virtualnom okruženju

2) Broj direktorija u virtualnom okruženju

3) Veličina direktorija virtualnog okruženja

4) Ispiši 10 najvećih datoteka

Prosječna veličina = **28 kbytea**

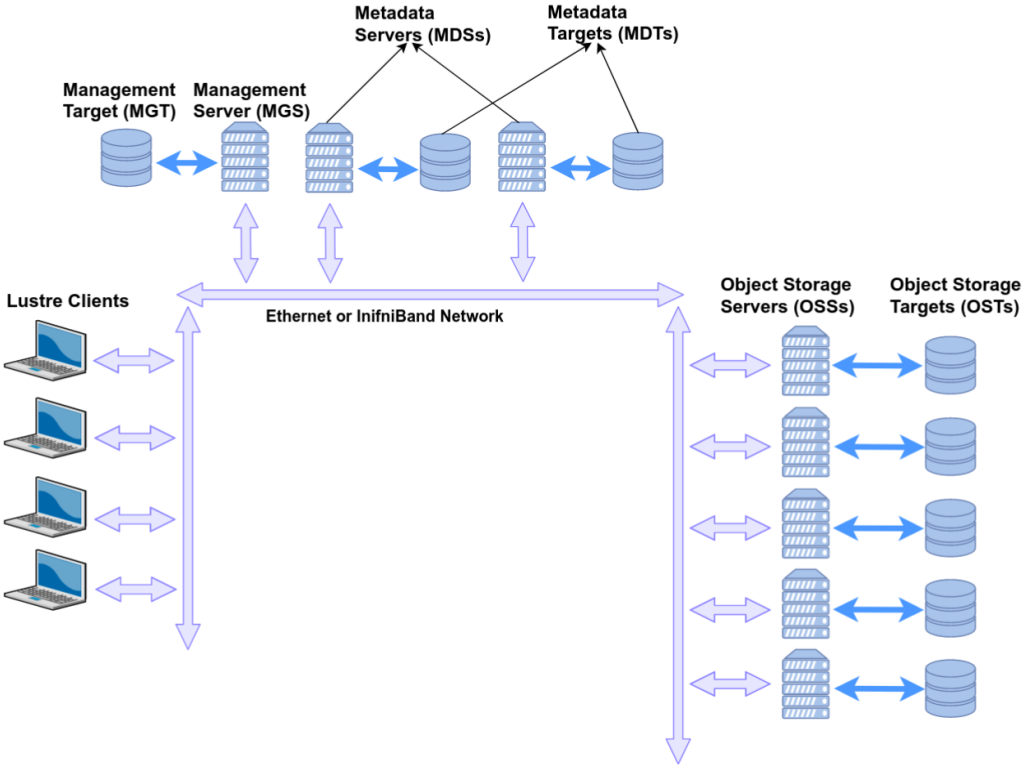
Kod 2 Ispis značajke kreiranog virtualnog okruženja

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Python i Lustre

Kako Lustre funkcionira?

- Paralelni distribuirani datotečni sustav namijenjen HPC okruženju
- Odvajanje virtualnog od fizičkog zapisa raspodijeljenog na više poslužitelja
- Komponente
  - Management – usklađivanje
  - Metadata – virtualni zapis
  - Object Storage – stvarni zapis
  - Client – korisnik
  - Network – veza

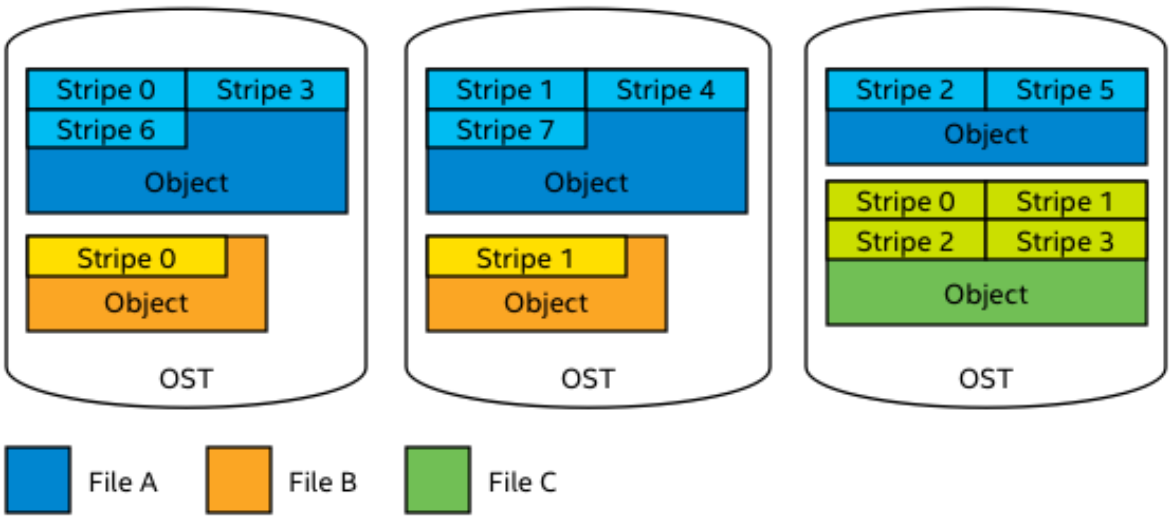


Slika 2 Shema datotečnog sustava Lustre (*Figure 1* u izvoru)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Python i Lustre

Lustre file striping

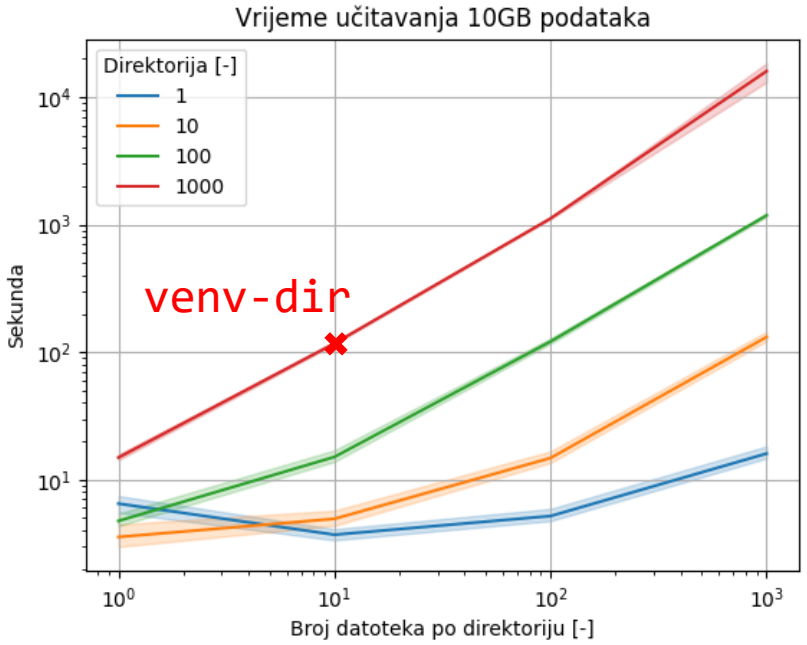


Slika 3 Shema file stripinga u Lustreu (*Fig 1.* u izvoru)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Python i Lustre

Lustre performanse



Slika 4 Performanse učitavanja 10GB podataka raspodijeljenih u 1-1000 direktorija i 1-1000 datoteka po direktoriju

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Apptainer i kontejneri

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Apptainer i kontejneri

Uvod

- Kontejneri
  - izoliran skup datoteka koji sadrži sve ovisnosti potrebne za izvršavanje aplikacije
- Apptainer
  - prilagođeni HPC-u
- Svrha
  - stvaranje specifičnih okolina za određenu aplikaciju
  - najbolje za Python i R knjižnice

Slika 5 Apptainer logo

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Apptainer i kontejneri

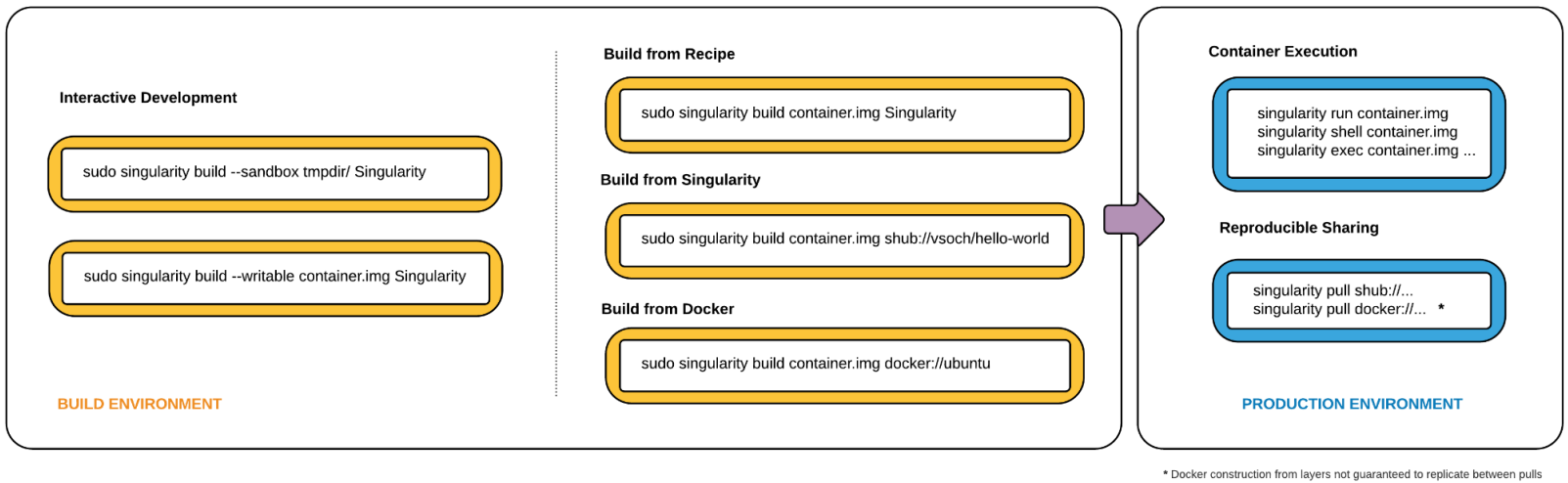
Izgradnja i korištenje na Supeku

- Izgradnja samo na pristupnom **gpu** poslužitelju
  - **`/scratch/apptainer`**
- MPI podrška
  - *bind model*
  - RedHat 8 kompatibilni OS
- GPU podrška
  - **`--nv`** opcija

Europska unija "Zajedno do fondova EU"
EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI
Operativni program KONKURENTNOST I KOHEZIJA
Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Apptainer i kontejneri

Izrada i korištenje kontejnera



Slika 6 “Singularity flow” (Izvor)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Apptainer i kontejneri

Izgradnja jednostavnog sandbox direktorija i konverzija

```
[korisnik@kompjuter ~]$ ssh korisnik@login-gpu.hpc.srce.hr

[korisnik@x3000c0s27b0n0 ~]$ cd /scratch/apptainer
[korisnik@x3000c0s27b0n0 apptainer]$ mkdir ${USER} && cd ${USER}

[korisnik@x3000c0s27b0n0 korisnik]$ apptainer build --sandbox pip/
docker://ubuntu:22.04
[korisnik@x3000c0s27b0n0 korisnik]$ apptainer shell --writable --fakeroot pip/

Apptainer> apt update -y
Apptainer> apt upgrade -y
...
Apptainer> apt install python3-pip –y
…
Apptainer> python3 –m pip install numpy
…
Apptainer> exit

[korisnik@x3000c0s27b0n0 korisnik]$ apptainer build pip.sif pip/
```

1) Spoji se na pristupni GPU poslužitelj

2) Pomakni se u direktorij za izgradnju
3) Napravi korisnički direktorij i pomakni se

4) Napravi sandbox kontejner s Ubuntu 22.04 OS-om
5) Otvori interaktivnu sjednicu

6) Osvježi repozitorije
7) Nadogradi OS

8) Instaliraj python i pip

9) Instaliraj pip knjižnicu

10) Izađi iz interaktivne sjednice

11) Prebaci iz sandbox direktorija u image

Kod 3 Primjer izgradnje kontejnera na Supeku

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Apptainer i kontejneri

Izgradnja jednostavnog kontejnera iz definicijske .def datoteke

```
Bootstrap: docker
From: ubuntu:22.04

%post

  # Ažuriranje OS-a
  apt update –y
  apt upgrade -y

  # Instalacija Pythona i pip modula
  apt install -y python3-pip

  # Instaliraj numpy
  python3 -m pip install numpy
```

1) Preuzmi s DockerHuba...
2) ... Ubuntu verziju 22.04

3) Osvježavanje repozitorija
4) Nadogradnja OS-a

5) Instalacija Pythona i pipa

6) Instalacija Python paketa

Kod 4 Primjer izgradnje kontejnera korištenjem definicijske datoteke na Supeku

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Apptainer i kontejneri

~/tutorial/6_apptainer/run.sh

Korištenje kontejnera u PBS skripti

```
#PBS -q cpu

cd ${PBS_O_WORKDIR}

apptainer exec pip.sif python3 input.py
```

1) Definiranje reda
2) Premještanje u radni direktorij
3) Pokretanje python skripte putem kontejnera

Kod 5 Primjer skripte PBS za pokretanje kontejnera

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Strojno učenje na Supeku

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Strojno učenje na Supeku

Aplikacije

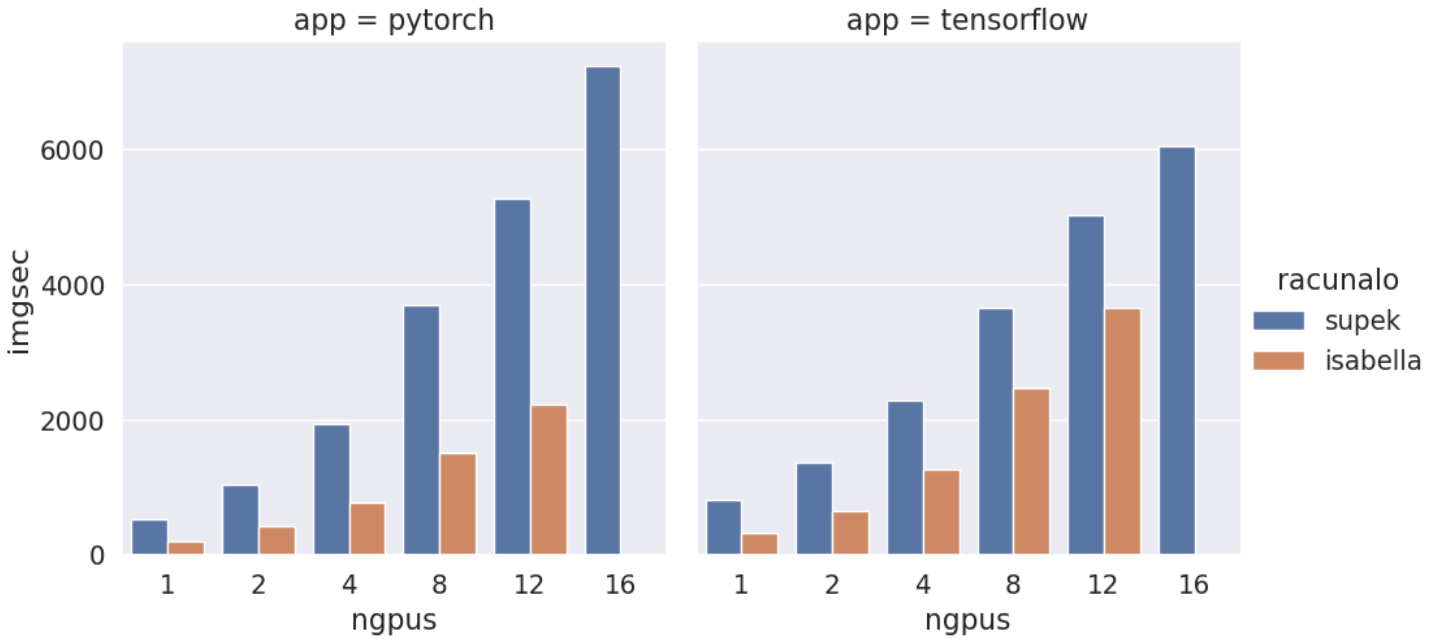
- NVIDIA NGC
  - kontejneri optimizirani za izvođenje na GPU
- Dostupne aplikacije
  - Tensorflow 2.10.1
  - PyTorch 1.8.0, 1.14.0 i 2.0.0
  - Dask 2023.7.0
  - Scikit-learn 0.24.1-2, 1.2.2, 1.30
  - Ray 2.4.0
- U izradi
  - Horovod
  - Rapids
  - Lightning AI

Slika 7 Trenutni repertoar knjižnica za strojno učenje

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Strojno učenje na Supeku

Resnet50 performanse



Slika 8 Brzina treniranja modela ResNet50 [img/sec] korištenjem PyTorcha (lijevo) i TensorFlow a (desno) na klasteru Supek (plavo) i Isabella (narančasto)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Strojno učenje na Supeku

Implementacija

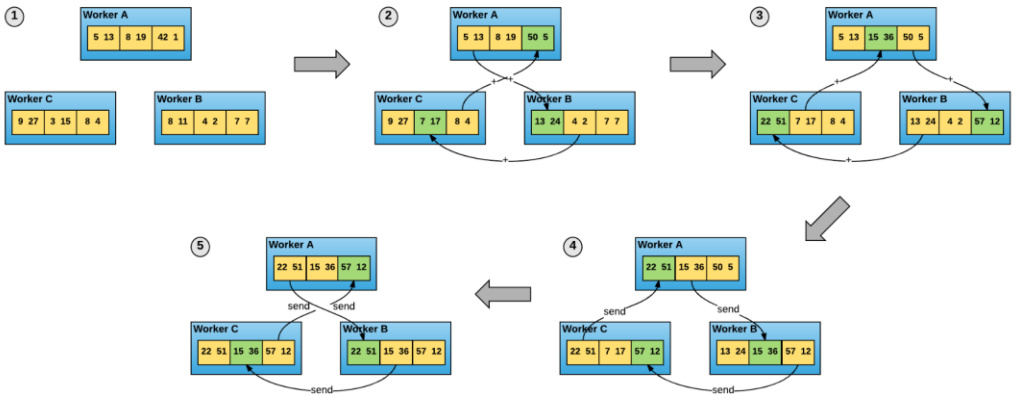
- Modulefiles
  - `$IMAGE_PATH` - definicija staze kontejnera
  - `$PATH` - dodavanje wrappera
- Wrapperi
  - Izvršne shell skripte koje osiguravaju integraciju s PBS-om
- Više inačica ovisno o potrebama
  - **`run-singlenode.sh`** – Tensorflow na jednom čvoru
  - **`torchrun-multinode.sh`** – PyTorch torchrun na više čvorova
  - **`dask-launcher.sh`** – Dask općenito

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Strojno učenje na Supeku

NCCL – Nvidia Collective Communications Library

- "Data parallel" problem *
  - Usklađivanje gradijenata na više čvorova/procesora
- NCCL
  - NVIDIA MPI
  - NVLink – 600GB/s
- AllReduce
  - Ring & Tree **



Slika 9 Ring AllReduce algoritam (*Figure 4.* u izvoru)

* https://siboehm.com/articles/22/data-parallel-training
** https://marek.ai/allreduce-the-basis-of-multi-device-communication-for-neural-network-training.html

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

Europska unija
"Zajedno do fondova EU"
EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI
Operativni program KONKURENTNOST I KOHEZIJA

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

Priprema

- Primjeri
  - Tensorflow i PyTorch
    - single GPU
    - multi GPU
  - Scikit-learn
    - Xgboost
    - Xgboost & Dask
    - Xgboost & Dask**2
  - Ray
    - PyTorch Train
    - TensorFlow Tune
- Priprema/git
  - mkvakic-srce/fer-20231006

```
[korisnik@x3000c0s25b0n0 ~]$ git clone git@github.com:mkvakic-srce/fer-
20231006.git
Cloning into 'fer-20231006'...
remote: Enumerating objects: 70, done.
remote: Counting objects: 100% (70/70), done.
remote: Compressing objects: 100% (44/44), done.
remote: Total 70 (delta 38), reused 58 (delta 26), pack-reused 0
Receiving objects: 100% (70/70), 11.26 KiB | 2.25 MiB/s, done.
Resolving deltas: 100% (38/38), done.

[korisnik@x3000c0s25b0n0 ~]$ cd fer-20231006

[mkvakic@x3000c0s25b0n0 fer-20231006]$ ls -1
README.md
pytorch_distributed.py
pytorch_distributed.sh
pytorch_ray_train.py
pytorch_ray_train.sh
pytorch_singlegpu.py
pytorch_singlegpu.sh
...
```

Kod 6 Priprema direktorija s primjerima korištenjem komande **git**

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

TensorFlow

- Zadano ponašanje - 1 GPU
- Distribuirani proračun
  - korištenjem "strategija"
  - Kompajliranje modela unutar djelokruga (*scopea*) strategije
- Strategije
  - **`OneDeviceStrategy`**
  - **`MirroredStrategy`**
  - **`MultiWorkerMirroredStrategy`**
- Podaci se dijele među procesorima

```
...
layers = [tf.keras.Input(10),                          1) Slojevi
          tf.keras.layers.Dense(10),
          tf.keras.layers.Softmax()]

model = tf.keras.Sequential(layers)                    2) Model

strategy = tf.distribute.MirroredStrategy()            3) Strategija

with strategy.scope():                                 4) Djelokrug
    model.compile()                                    5) Kompiliranje

model.fit(data)                                        6) Treniranje
...
```

Kod 7 Priprema TensorFlow modela sa strategijom **`MirroredStrategy`**

Europska unija "Zajedno do fondova EU" EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI Operativni program KONKURENTNOST I KOHEZIJA

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/tensorflow-singlegpu.*

TensorFlow na jednom GPU procesoru

```
#PBS -q gpu
#PBS -l ngpus=1

module load scientific/tensorflow

cd ${PBS_O_WORKDIR:-""}

run-singlenode.sh tensorflow-singlegpu.py
```

1) Definiranje reda
2) Definiranje resursa
3) Dopremanje TensorFlowa
4) Pomicanje u radni direktorij
5) Pokretanje python skripte putem wrappera

Kod 8 Primjer skripte PBS **tensorflow-singlegpu.sh**

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/tensorflow-strategy.*

TensorFlow na više GPU procesora

```
#PBS -q gpu
#PBS -l select=2:ngpus=1

module load scientific/tensorflow

cd ${PBS_O_WORKDIR:-""}

run-multinode.sh tensorflow-strategy.py
```

1) Definiranje reda
2) Definiranje resursa
3) Dopremanje TensorFlowa
4) Pomicanje u radni direktorij
5) Pokretanje python skripte putem wrappera

Kod 9 Primjer skripte PBS **tensorflow-strategy.sh**

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

PyTorch

- Ručno postavljanje na dostupne procesore (rangove)
- `torchrun`
  - izvorno sučelje
  - implementacija slična MPI
- `accelerate`
  - HuggingFace sučelje
  - viši nivo apstrakcije

```
...
torch.distributed.init_process_group('nccl')
local_rank = int(os.environ['LOCAL_RANK'])

model = torchvision.models.resnet50()
model = model.to(local_rank)

for input, output in dataloader:
    optimizer.zero_grad()
    predicted = model(input)
    loss = loss_fn(predicted, output)
    loss.backward()
    optimizer.step()
...
```

```
1)  Inicijalizacija
2)  Definiraj lokalni rang

3)  Model
4)  Postavi na lokalni rang

5)  Treniraj
```

Kod 10 Priprema PyTorch modela sa **`torchrun`**

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/pytorch-singlegpu.*

PyTorch na jednom GPU procesoru

```
#PBS -q gpu                                          1) Definiranje reda
#PBS -l ngpus=1                                      2) Definiranje resursa

module load scientific/pytorch                       3) Dopremanje PyTorcha

cd ${PBS_O_WORKDIR:-""}                              4) Pomicanje u radni direktorij

run-singlegpu.sh pytorch-singlegpu.py                5) Pokretanje python skripte putem wrappera
```

Kod 11 Primjer skripte PBS **pytorch-singlegpu.sh**

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/pytorch-torchrun.*

PyTorch na više GPU procesora

```
#PBS -q gpu                                          1) Definiranje reda
#PBS -l select=2:ngpus=1                             2) Definiranje resursa

module load scientific/pytorch                       3) Dopremanje PyTorcha

cd ${PBS_O_WORKDIR:-""}                              4) Pomicanje u radni direktorij

torchrun-multinode.sh pytorch-torchrun.py            5) Pokretanje python skripte putem wrappera
```

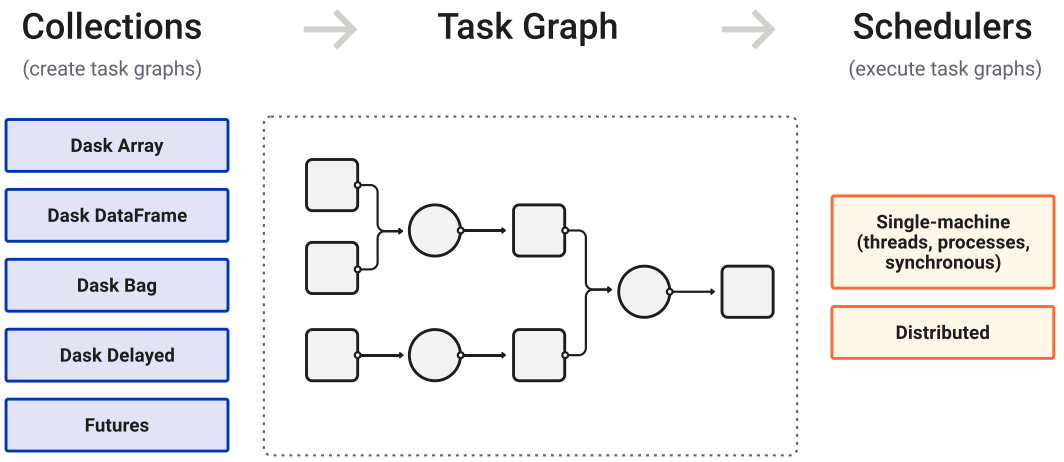
Kod 12 Primjer skripte PBS **pytorch-torchrun.sh**

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

Dask

- Namijenjeno paralelizaciji Python koda i distribuiranoj obradi OOM podataka
- Razlaganje programa na jednostavne operacije koristenjem **dask** grafova
- Komponente
  - Array – tenzori
  - DataFrame – strukturirani podaci
  - Bag – nizovi
  - Delayed – custom funkcije
  - Futures – eager Delayed



Slika 10 Shema aplikacije na klasteru Dask (izvor)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

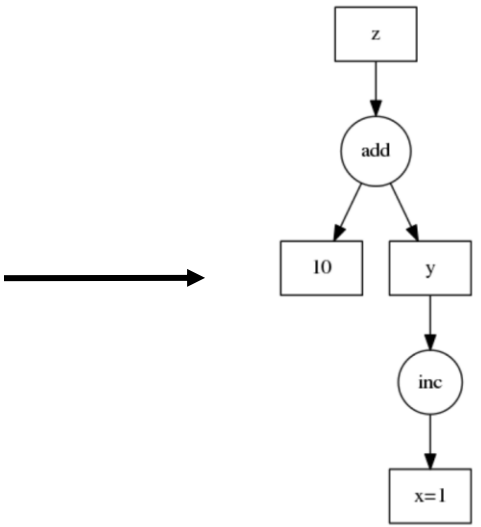
Primjeri

Dask grafovi

```
def inc(i):
    return i + 1

def add(a, b):
   return a + b

x = 1
y = inc(x)
z = add(y, 10)
```




```
d = {'x': 1,
     'y': (inc, 'x'),
     'z': (add, 'y', 10)}
```

Slika 11 Osnovni graf Dask (*Fig 1.* u izvoru)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

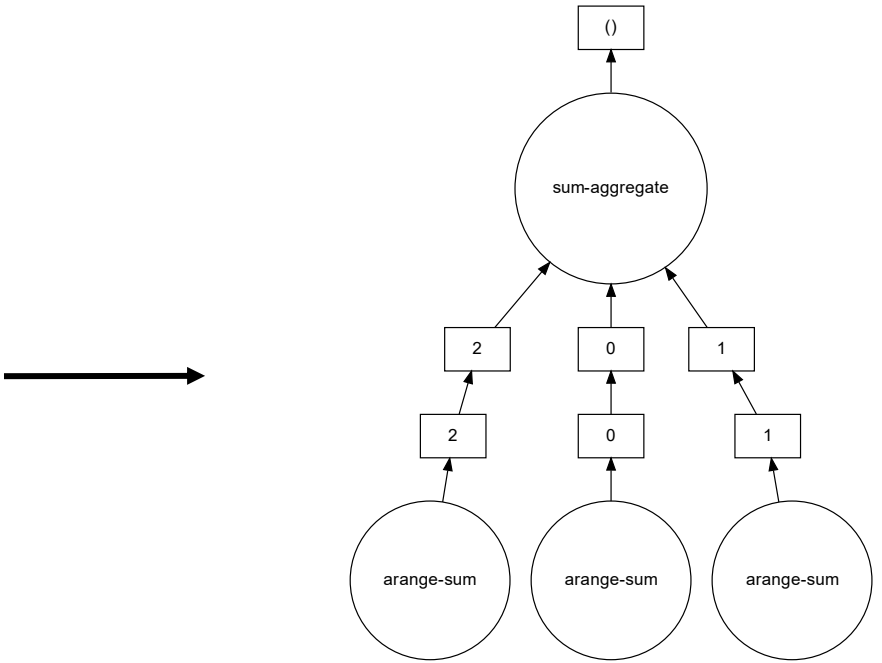
Dask Array

```
>>> import dask.array as da

>>> x = da.arange(15, chunks=(5))
>>> x.dask
{('x', 0): (np.arange, 0, 5),
 ('x', 1): (np.arange, 5, 10),
 ('x', 2): (np.arange, 10, 15)}

>>> z = (x + 100).sum()
>>> z.dask
{('x', 0): (np.arange, 0, 5),
 ('x', 1): (np.arange, 5, 10),
 ('x', 2): (np.arange, 10, 15),
 ('y', 0): (add, ('x', 0), 100),
 ('y', 1): (add, ('x', 1), 100),
 ('y', 2): (add, ('x', 2), 100),
 ('z', 0): (np.sum, ('y', 0)),
 ('z', 1): (np.sum, ('y', 1)),
 ('z', 2): (np.sum, ('y', 2)),
 ('z',): (sum, [('z', 0), ('z', 1), ('z', 2)])}

>>> z.compute()
1605
```

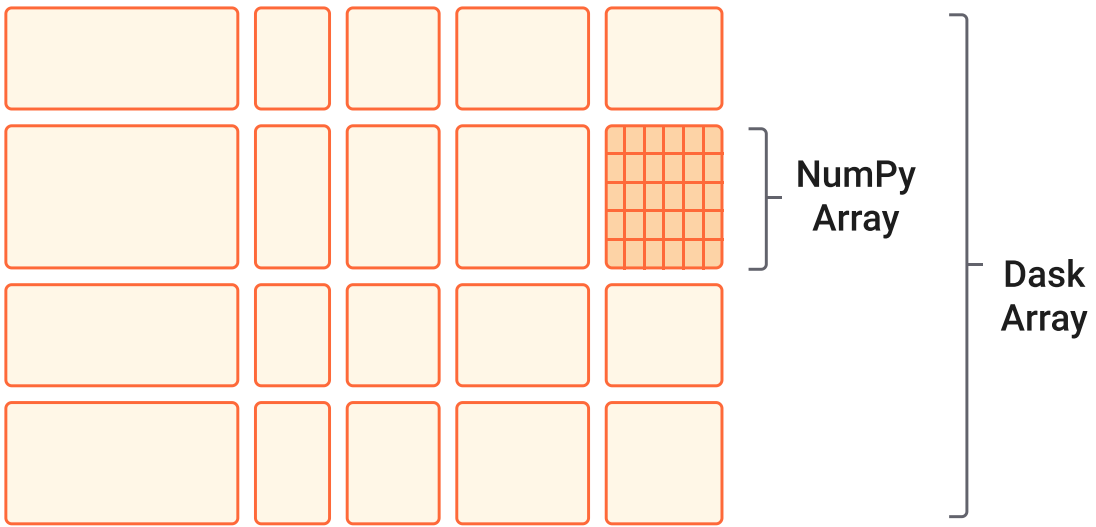


Slika 12 Dask Array i operacija zbrajanja u obliku grafa (*Example: arange* u izvoru)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

Dask OOM podaci



Slka 13 Distribucija podataka na klasteru Dask u slučaju Dask Arraya (izvor)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/sklearn-threads.*

Scikit-learn putem multi-threadinga

```
#PBS -q cpu
#PBS -l select=1:ncpus=16

module load scientific/dask

cd ${PBS_O_WORKDIR:-""}

$IMAGE_PATH python sklearn-threads.py
```

1) Definiranje reda
2) Definiranje resursa

3) Dopremanje Daska

4) Pomicanje u radni direktorij

5) Pokretanje python skripte putem wrappera

Kod 13 Primjer skripte PBS **sklearn-threads.sh**

Europska unija "Zajedno do fondova EU"
EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI
Operativni program KONKURENTNOST I KOHEZIJA
Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/sklearn-dask.*

Scikit-learn + Dask na više čvorova

```
#PBS -q cpu
#PBS -l select=2:ncpus=16:mem=50GB

module load scientific/dask

cd ${PBS_O_WORKDIR:-""}

dask-launcher.sh sklearn-dask.py
```

1) Definiranje reda
2) Definiranje resursa
3) Dopremanje Daska
4) Pomicanje u radni direktorij
5) Pokretanje python skripte putem wrappera

Kod 14 Primjer skripte PBS **sklearn-dask.sh**

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/sklearn-dask-dask.*

Dask na više čvorova

```
#PBS -q cpu
#PBS -l select=2:ngpus=1

module load scientific/dask

cd ${PBS_O_WORKDIR:-""}

dask-launcher.sh sklearn-dask-dask.py
```

Kod 15 Primjer skripte PBS **sklearn-dask-dask.sh**

1) Definiranje reda
2) Definiranje resursa

3) Dopremanje Daska

4) Pomicanje u radni direktorij

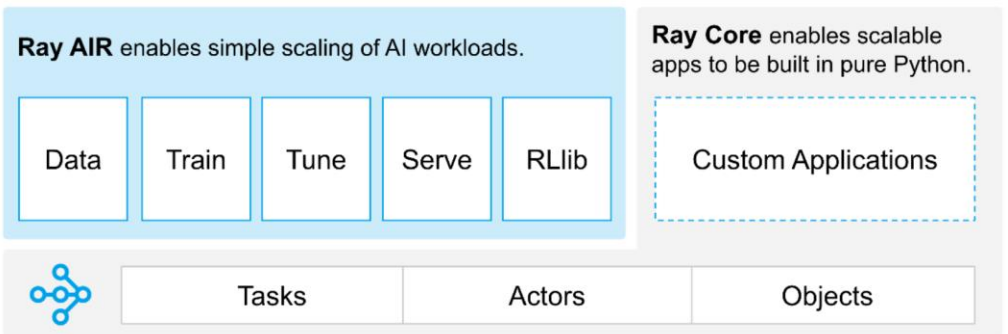
5) Pokretanje python skripte putem wrappera

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

Ray

- Originalno
  - Distribuirano ojačano učenje
- Komponente
  - Ray Core – osnovne elementi
  - Ray Air – distribucija ML
  - Ray Ecosystem – prilagodba postojećim knjižnicama ML
- Osnovni elementi – Ray Core
  - Tasks – funkcije
  - Actors – klase
  - Objects – varijable



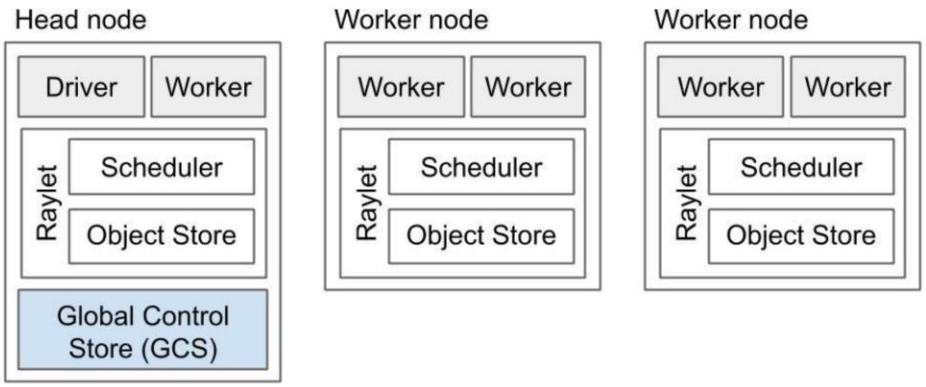
Slika 14 Ray komponente (izvor)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

Ray klasteri

- Head čvor
  - Driver – glavni program
  - Global Control Store – koordinacija klastera
- Worker čvorovi
  - Worker – izvršavanje zadataka i "posjedovanje" referenci
- Raylet
  - koordinacija lokalnih resursa
  - Scheduler – raspoređivanje
  - Object Store – pohrana

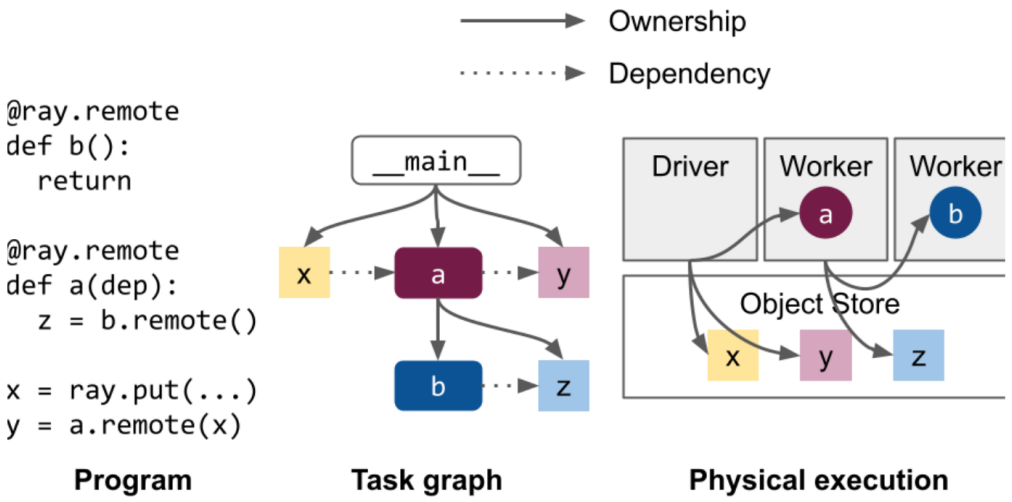


Slika 15 Shema Ray klastera (*Figure 2-3* u izvoru)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

Ray reference i izvođenje programa



Slika 16 Shema Ray posjedovanja i izvršavanja (*Ownership, str. 8* u izvoru)

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/pytorch-ray-train.*

PyTorch putem Ray Traina

```
#PBS -q gpu
#PBS -l select=2:ngpus=1:ncpus=4

module load scientific/ray

cd ${PBS_O_WORKDIR:-""}

ray-launcher.sh pytorch-ray-train.py
```

Kod 16 Primjer skripte PBS **pytorch-ray-train.sh**

1) Definiranje reda
2) Definiranje resursa

3) Dopremanje Raya

4) Pomicanje u radni direktorij

5) Pokretanje python skripte putem wrappera

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Primjeri

~/fer-20231006/tensorflow-ray-tune.*

TensorFlow putem Ray Tunea

```
#PBS -q gpu
#PBS -l select=2:ngpus=1:ncpus=4

module load scientific/ray

cd ${PBS_O_WORKDIR:-""}

ray-launcher.sh tensorflow-ray-tune.py
```

Kod 17 Primjer skripte PBS **tensorflow-ray-tune.sh**

1) Definiranje reda
2) Definiranje resursa

3) Dopremanje Raya

4) Pomicanje u radni direktorij

5) Pokretanje python skripte putem wrappera

Europska unija "Zajedno do fondova EU"
EUROPSKI STRUKTURNI I INVESTICIJSKI FONDOVI
Operativni program KONKURENTNOST I KOHEZIJA
Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.

Hvala na pažnji!

Projekt je sufinancirala Europska unija iz Europskog fonda za regionalni razvoj.