TRUBA
Türk Ulusal Bilim e-Altyapısı

TÜBİTAK
ULAKBİM

YBH Ortamında Python, Conda Kurulumu ve Kullanımı

İsmail Güzel
TÜBİTAK ULAKBİM
TRUBA

EURO^2

İçerik

- Anaconda / Miniconda kurulumu
- Modül olarak Python kullanma
- Optimize Intel Python
- Paketleri Yönetme
- Slurm betiği ile iş gönderme

Neden Python?

- **Basit ve Anlaşılır Syntax**:
  Python, okunabilir ve anlaşılır bir dile sahiptir, bu da öğrenmeyi kolaylaştırır.

- **Çok Yönlü Kullanım:**
  Web geliştirme, veri analizi, yapay zeka ve daha fazlasında kullanılan geniş bir kütüphane ve arz yelpazesi sunar.

- **Topluluk Desteği:**
  Büyük ve aktif bir topluluğa sahiptir, bu da sorularınızın hızlı bir şekilde cevaplanmasını sağlar.

- **PyPI Stats**: PyPI paketleri için analitik, https://pypistats.org/
  PyPI ile hazır bulunan python paket sayısı : **486 367**

Anaconda / Miniconda

- Conda ürünleri, Continuum Analytics tarafından oluşturulan ücretsiz bir platformlar arası paket yönetim sistemi oluşturur ve aşağıdaki bileşenlerden oluşur.

  - **Conda**, uyumlu dağıtımlardan paket yüklemek için açık kaynaklı bir paket yöneticisi ve sanal ortam yöneticisidir.

  - **Anaconda**, birçok python paketi ve uzantısı içeren bir Conda paket dağıtımıdır. Conda ile birlikte gelir.

  - **Miniconda**, Conda ile birlikte çok daha küçük bir çekirdek paket seti içerir.

Merkezi Anaconda Kullanımı

- **centos7.3 işletim sistemi**

  - `eval "$(/truba/sw/centos7.3/lib/anaconda3/2021.11/bin/conda shell.bash hook)"`

- **centos7.9 işletim sistemi**

  - `eval "$(/truba/sw/centos7.9/lib/anaconda3/2021.11/bin/conda shell.bash hook)"`

- **Konfigürasyon**:

```
conda config --add pkgs_dirs /truba/home/$USER/anaconda
conda config --add envs_dirs /truba/home/$USER/anaconda/envs
conda update -n base -c defaults conda
```

- **Dosya sayısı problemi:**
  Kullanıcı ev dizinine kurulması tavsiye edilmez. Miniconda kurulumu yapılabilinir.

Miniconda Kurulumu

- Ev dizinine kurulum:
  - **İndirme**: wget https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh

  - **Kurulum başlatma** : `bash Miniconda3-latest-Linux-x86_64.sh`

  - **Kurulum esnasında**:
    - `conda init` NO
    - prefix `/truba/home/$USER/miniconda3`

- **Kullanım** :
  `eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash hook)”`

- **Not:** Duruma göre `.local` altında bulunan paketleri önlemek istenebilir.
  `export PYTHONNOUSERSITE=False`

Intel Python Kullanımı

- Intel python kullanımı conda ortamı içerisinde olmaktadır.

- Ev dizinine kurulum:
  - **İndirme**: wget https://registrationcenter-download.intel.com/akdlm//IRC_NAS/f649c85f-f6e1-4f13-b8c1-b75ec1a9ccb9/intelpython3-2024.0.0_251-Linux-x86_64.sh
  - **Kurulum başlatma** : `sh intelpython3-2024.0.0_251-Linux-x86_64.sh`
  - **Kurulum esnasında**:
    - `conda init` NO
    - prefix `/truba/home/$USER/intelpython3`

- **Kullanımı :**
  `eval "$(/truba/home/$USER/intelpython3/bin/conda shell.bash hook)"`

- **TRUBA' da olan intel python kullanım:**

  `eval "$(/truba/sw/centos7.9/lib/intelpython3/bin/conda shell.bash hook)"`

Paket Yönetimi

	pip	conda
Yapısı	Python paketleri için standart	Python ve diğer diller
Paket Kaynağı	pypi	anaconda reposu, conda-forge
Bağımlılık Yönetimi	Bağımlı yükler. Çatışma kontrolü zor.	Çatışmaları daha iyi yönetir.
Çoklu Dil Desteği	Python	Python, R, Julia, Scala, …
Sanal Ortamlar	`virtualenv` ya da `venv`	Entegre yönetim
Yüklenme Hızı	Yavaş, Python dilinde	Daha Hızlı, önceden derlenmiş
Güncelleme ve Kaldırma	`pip install / uninstall`	`conda install / remove`
Depolama Formatı	TXT dosyası: requirement.txt	YAML dosyası: environment.yml

Modül Python Paket Yönetimi: pip

- **TRUBA'da hazır bulunan modüller:**

```
module av 2>&1 | grep python

modül_ismi/versiyon-bagımlı1-versiyon-bagımlı2-versiyon
```

- **pip güncelleme:** `python3 -m pip install --upgrade pip`
- **Sanal ortam paketi kurma:** `python3 -m pip install virtualenv --user`

Modül Python Paket Yönetimi: pip

- **Sanal ortam oluşturma:**

```
export PATH=$PATH:$HOME/.local/bin
virtualenv $HOME/python/proje1
```

- **Paketlerin dizini:**
  `$HOME/.local/lib/python3.11/site-packages`

- **Aktif etme**:
  `source $HOME/python/proje1/bin/activate`
  Sanal ortam içerisinde --user parametresine gerek yoktur.

- **Çıkış**:
  `deactivate`

- **pip dokümantasyon**

- **Paket Yüklemek:**

```
pip install paket_ismi
pip install ‘paket_ismi==paket_versiyon’
pip install ‘paket_ismi>=1.20,<1.25’
pip install -r requirements.txt
```

- `pip install 'https://github.com/numpy/numpy/releases/download/v1.26.2/numpy-1.26.2.tar.gz'`

- `wget https://github.com/psf/requests/archive/refs/tags/v2.27.1.tar.gz`
  `pip install ./requests-2.29.0.tar.gz`

- **GitHub’dan kurma**:

```
git clone https://github.com/urllib3/urllib3.git
cd urllib3
pip install .
```

- **Paketler nelerdir?**

```
pip freeze
pip list
```

- **Dışarı aktarım**

```
pip freeze > requirements.txt
```

- **İçeri aktarım**

```
virtualenv ./py3.11/proje3
source ./py3.11/proje3/bin/activate
pip install -r requirements.txt
```

- **Paket kaldırmak:**

```
pip uninstall numpy
```

- **Paketler nerede?**

  - ✓ `python -m site`
  - ✓ `python -m site --user-site`
  - ✓ `python -c 'import site; print(site.getsitepackages())'`
  - ✓ `python -m paket_ismi --version`
  - ✓ `pip show paket_ismi`

Conda Python Paket Yönetimi: conda and pip

- **Sanal ortam bilgisi:**
```
conda info
conda env list
```

- **Konfigürasyon**
```
conda config --show
conda config --show default_channels
conda config --show channels
conda config --add channels <channel_ismi>
conda config --remove channels <channel_ismi>
```

- **Oluşturma**
```
conda create --name <env_ismi>
conda create --name <env_ismi_2> --clone <env_ismi_1>
```

- **Aktarım:**
  `conda env export --no-builds > requeriments.yml`
  `conda env create --prefix /dizin/miniconda3/envs/env_ismi --file requirements.yml`

- **Aktif etme:**
  `conda activate env_ismi`
  `source activate env_ismi`

- **Silme:** `conda remove --name <env_ismi> --all`

- **Ortam içerisinde paketler:**

  - Paket arama: `conda search numpy`
  - Paket yükleme : `conda install paket_ismi=<versiyon>`
  - Paketleri listeleme : `conda list --name env_ismi`
  - Paket silme : `conda remove --name proje1 paket_ismi`

güncelleme ve geri alma

- Güncel Python paketleri
- Uyumluluğu korumak veya sorunları gidermek

✓ `conda update -n base -c defaults conda --repodata-fn=repodata.json`

✓ `conda update --all`

✓ `conda update <paket_ismi>`

✓ `conda list --revision`

✓ `conda install --revision <revizyon_numarası>`

.bashrc .bash_profile .profile

- Bash kabuğunun yapılandırılması ve kullanıcı oturumları sırasında çalıştırılacak komutları içeren dosyalardır.

- ```
  touch .bash_profile
  rm .profile .bashrc
  ln -s .bash_profile .profile
  ln -s .bash_profile .bashrc
  ```

- Her girişte kullanıc miniconda aktif edilmek istenilirse
  - `vi .bashrc`
  - `eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash hook)"`
  - `conda activate <env_ismi>`
  ```

SLURM iş yöneticisi

Örnek slurm dosyası

```
1. #!/bin/bash

2. #SBATCH --account=your_account
   #SBATCH --mail-user=your_email_address
   #SBATCH --mail-type=ALL
   #SBATCH --workdir=/isler/dizin
   #SBATCH --output=%A.out #%A=JOB_ID
   #SBATCH --error=%A.err
   #SBATCH --job-name=test
   #SBATCH --partition=debug
   #SBATCH --time=00:15:00

3. ### Sunucu ve Cekirdek ayarlama

4. ### Modül ve ortam ayarlama

5. ### İşler

6. exit
```

- **SLURM için kullanışlı değişkenler**
- $SLURM_JOB_ID (job id)
- $SLURM_NNODES (sunucu sayısı)
- $SLURM_NTASKS (MPI iş sayısı)
- $SLURM_CPUS_PER_TASK (Her MPI için CPU çekirdek sayısı)
- $SLURM_SUBMIT_DIR (işin yüklendiği klasör)

Sunucu ve Çekirdek Ayarlama

- **Seri Hesaplama**
  Tek Sunucu/Is/Cekirdek
- `#SBATCH --ntasks=1`
- `#SBATCH --nodes=1`
- `#SBATCH --ntasks-per-node=1`
- `#SBATCH --cpus-per-task=1`

- **Paylaşımlı Bellek**
  Tek Sunucu/Is Çok Cekirdek
  OpenMP kullanılır.
- Tek bir is icin 20 tane cekirdek ayrılır.
  Aynı hafiza kullanilir.
- `#SBATCH --ntasks=1`
- `#SBATCH --nodes=1`
- `#SBATCH --ntasks-per-node=1`
- `#SBATCH --cpus-per-task=20`

Sunucu ve Çekirdek Ayarlama

- **Dağıtık Hesaplama: MPI**
  20 tane cekirdek ayırır ama her biri ayrı bir is gibi calisir.
- `#SBATCH --ntasks=20`
- `#SBATCH --nodes=1`
- `#SBATCH --cpus-per-task=1`

- **Dağıtık Hesaplama: MPI**
  5 sunucu ayarlanır.
  Her sunucu icinde 4 is calisir.
  Her isi tek cekirdek uzerinde gerceklestirir.
- `#SBATCH --ntasks=20`
- `#SBATCH --nodes=5`
- `#SBATCH --ntasks-per-node=4`
- `#SBATCH --cpus-per-task=1`

Sunucu ve Çekirdek Ayarlama

- **Dağıtık ve Paylaşımlı Bellek**
  Hibrit : MPI ve OpenMP
- 5 sunucu ayarlanır. Her sunucu icinde 4 is calisir. Her is 10 cekirdek uzerinden gerceklestirir.
- `#SBATCH --ntasks=20`
- `#SBATCH --nodes=5`
- `#SBATCH --ntasks-per-node=4`
- `#SBATCH --cpus-per-task=10`

- **GPU Hızlandırcı**
- `#SBATCH --gres=gpu:1`

- **OpenMP**

```
export OMP_NUM_THREADS=1

export MKL_NUM_THREADS=1
```

Modül veya ortam ayarlama

- `module load centos7.3/comp/gcc/12.1`
- `module load centos7.3/lib/openmpi/4.1.5-gcc-12.1`
- `module load centos7.3/comp/python/3.11.2-openmpi-4.1.5-gcc-12.1-GOLD`
- `source /truba/home/$USER/python/proje1/bin/activate`
- `python python_dosyası.py`

Modül veya ortam ayarlama

- **Kurulum**

```
eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash
hook)"
conda create --name proje1
conda activate proje1
conda install pyhton3.10 openmpi mpi4py numpy
```

- **SLURM dosyası**

```
eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash
hook)”
conda activate proje1
python python_dosyasi.py
```

multi_node_mpi.slurm

```
#!/bin/bash

#SBATCH --account=your_account
#SBATCH --mal-user=your_email_address
#SBATCH --mail-type=ALL
#SBATCH --workdir=/isler/dizin
#SBATCH --output=%A.out #%A=JOB_ID
#SBATCH --error=%A.err
#SBATCH --job-name=test
#SBATCH --partition=debug
#SBATCH --time=00:15:00

#SBATCH --ntasks=4
#SBATCH --nodes=2
#SBATCH --cpus-per-task=1

eval "$(/truba/home/$USER/miniconda3/bin/conda shell.bash hook)”
conda activate proje1

mpirun -np $SLURM_NTASKS python mpi.py > out_mpi.txt

exit
```

MPI için basit bir örnek

```
from mpi4py import MPI
import os

if __name__ == "__main__":
        size = MPI.COMM_WORLD.Get_size()
        rank = MPI.COMM_WORLD.Get_rank()
        name = MPI.Get_processor_name()
        pid = os.getppid()
        print("Merhaba, Islem numara: {} Toplam MPI sayisi {}
 Sunucu ismi {} Python islem {}".format(rank,size,name,pid))
```

sbatch multi_node_mpi.slurm

```
Merhaba, Islem numara: 3 Toplam MPI sayisi 4 Sunucu ismi barbun137.yonetim
Python islem 32909
Merhaba, Islem numara: 0 Toplam MPI sayisi 4 Sunucu ismi barbun136.yonetim
Python islem 305899
Merhaba, Islem numara: 1 Toplam MPI sayisi 4 Sunucu ismi barbun136.yonetim
Python islem 305899
Merhaba, Islem numara: 2 Toplam MPI sayisi 4 Sunucu ismi barbun136.yonetim
Python islem 305899
```

TEŞEKKÜRLER

grid-teknik@ulakbim.gov.tr