Instruction for Hands-on Exercises

1. Install and start docker program on your laptop
    a. Linux
        i. Install
            - sudo apt install docker (Ubuntu), see https://docs.docker.com/engine/install/ubuntu/
            - sudo dnf install docker (Fedora), see https://docs.docker.com/engine/install/fedora/

        ii. Start Docker daemon
            - sudo systemctl start docker
            - sudo systemctl enable docker (Docker always starts at reboot)
    b. Windows
        - Sign up Docker Hub and download installer from https://docs.docker.com/docker-for-windows/install/
        - Launch Docker Desktop
        - Open powershell or cmd
    c. Mac
        - Sign up Docker Hub and download installer from https://docs.docker.com/docker-for-mac/install/
        - Launch Docker Desktop

2. Get the Docker image
    a. docker pull liuyangzhuan/gptune:2.6
    b. docker images
       (shows you what docker images you have available)

3. Run the Docker image (run as root)
    a. docker run -it liuyangzhuan/gptune:2.6
       (create a container from the image and run it in interactive mode)

4. Testing
   cd /app/GPTune/   ## root directory of the docker image
   edit run_examples.sh ## vim,emacs,nano are available in the image

At line 38, change “nodes=1” to the number of nodes on your machine (nodes=1 for most laptops/PCs). At line 247, change “cores=4” to the number of cores per node on your machine. Then keep a copy by “cp run_examples.sh run_examples.sh_backup”

a. **GPTune-Demo**: cp run_examples.sh_backup run_examples.sh. Uncomment lines 293-298 and run the script as follows:

   ```
   bash run_examples.sh
   ```

   This example minimizes an analytic function with one task input parameter *t* and tuning parameters *x*. The default setting generates 20 samples and 1 task. You can add command line options -nrun xxx and -ntask xxx after “python ./demo.py” to vary these numbers. The optimal tuning parameter and function value are printed after “Popt” and “Oopt”. The tuner runtime profile is printed after “stats:”. All function evaluation data are stored in ./examples/GPTune-Demo/gptune.db/GPTune-Demo.json.

b. **Scalapack-PDGEQRF**: cp run_examples.sh_backup run_examples.sh. Uncomment lines 300-305 and run the script. This example minimizes runtime of QR factorization of ntask=2 randomly generated matrices with sizes at most mmax=1000 x nmax=1000, and tuning parameters: blocking sizes, thread count, and MPI process grid. GPTune will generate run=40 samples per task. The tuner runtime profile is printed after “stats:”. All function evaluation data are stored in ./examples/Scalapack-PDGEQRF/gptune.db/PDGEQRF.json. The optimal tuning parameter and function value are printed after “Popt” and “Oopt” for each task “m:x n:y”. mmax, nmax, ntask, nrun can be changed at line 305.

c. **Scalapack-PDGEQRF_RCI**: cp run_examples.sh_backup run_examples.sh. Uncomment lines 374-380 and run the script. The application is the same as above. However, this example uses the reverse communication interface (RCI) of GPTune. nrun, mmax, nmax, ntask can be changed at line 380. All function evaluation data are stored in ./examples/Scalapack-PDGEQRF_RCI/gptune.db/PDGEQRF.json.

d. **More examples**: Uncomment corresponding lines [SuperLU_DIST (line 309-314), STRUMPACK (line 317-322, 323-330), MFEM (line 334-339), ButterflyPACK (line 342-347)]. You can also work through all examples in run_ppopp.sh (see detailed comments there) to reproduce experiments and figures of the paper *GPTune: Multitask Learning for Autotuning Exascale Applications*, PPoPP21.

e. **Using shared repository**: Users can access our shared repository at https://gptune.lbl.gov and download/upload obtained function evaluation data. We encourage attendees to upload the generated GPTune-Demo's JSON file (the demo example does not require specific software/machine configuration, so it is easy to try). We provide a tester account (ID: gptune-tester, Password: gptuneTester).
   - The GPTune-Demo's JSON file can be copied out of the docker image by:
   docker cp [container_ID]:/app/GPTune/examples/GPTune-Demo/gptune.db/GPTune-Demo.json [local_dir]
   ## the [container_ID] can be found by docker ps -a (run in your local system, not inside docker image), [local_dir] is your desired local path.
   - After signing-in (https://gptune.lbl.gov/account/login/), access to the upload form (https://gptune.lbl.gov/repo/upload/).
   - Choose GPTune-Demo (defined by user "ycho") from the tuning problem list and "AnyMachine" from the machine list. Then, you can upload your GPTune-Demo JSON file.
   - You will be able to view the submitted data by using our dashboard (https://gptune.lbl.gov/repo/dashboard/).

f. **cGP:** The user can try the cGP package for optimizing a 2D function 'BUKIN_N6':
```
export PATH=/app/GPTune/env/bin/:$PATH
cd /app/cGP/
python cGP.py -h   # this prints out all options of cGP
```

python cGP.py -p 10 -s 30 -c 2   # this optimizes BUKIN_N6 function with 10 initial pilot samples, 30 sequential samples, and using 2 clusters/cores.

5. Other useful commands:
    a. Attach to a running container
       docker image ls (list all available images)
       docker container ls (list all running containers)
       docker attach container_id (attach to a running container with local changes
6. Installation without Docker. If you prefer not to use Docker, you can also install GPTune directly. The installation will take up to 2 hours depending on your system.
    a. Download GPTune:
        i. git clone https://github.com/gptune/GPTune.git
        ii. cd GPTune
    b. Install GPTune
        i. Ubuntu-like OS: bash config_cleanlinux.sh
        ii. Mac OS: Edit lines 6-9 of config_macbook.zsh to make sure they match the version numbers provided by homebrew. zsh config_macbook.zsh
        iii. NERSC Cori: bash config_cori.sh
    c. Run GPTune
        i. Edit top parts of run_examples.sh (and/or run_ppopp.sh) so that they match your system: Ubuntu (lines 34-38), Mac OS (lines 11-15), Cori (lines 18-22).
        ii. bash run_examples.sh (see section “4. Testing” above.)