ScienceCluster DeLTA setup

1. Connect to ScienceCluster

The material on the ScienceCluster Training [1] website gives you a good overview of the usage ScienceCluster (SC). Here I just briefly cover the steps to set up and run DeLTA on SC.

Once you are signed up for SC, you can access a login node. Connect using the following command in a terminal. Replace 'shortname' with your own uzh-shortname.

```
ssh shortname@cluster.s3it.uzh.ch
```

You will be prompted for your Active Directory password

2. Creating a Python environment for DeLTA with a GPU node

Once you are on the login node, run the following steps to start an interactive session on a GPU compute node and create a virtual environment with DeLTA installed [1],[2].

```
# load the gpu module

module load gpu

# request an interactive session,
# which allows the package installer
# to see the GPU hardware

srun --pty -n 1 -c 2 --time=01:00:00 --gpus=1 --mem=8G bash -l

# (optional) confirm the gpu is available.
# The output should show basic information
# about at least one GPU.

nvidia-smi

# use mamba (drop-in replacement for conda)

module load mamba

# create a virtual environment named 'delta_env' and install packages

mamba create -n delta_env -c conda-forge delta2

# activate the virtual environment
```

```
source activate delta_env

# confirm that the GPU is correctly detected

python -c 'import tensorflow as tf; print("Built with CUDA:", tf.test.is_built_with_cuda());

# Download the pretrained models for DeLTA.
# I noticed that in my case, this doesn't work (see 'Download Models manually' below)
# Note the path to the models in the output ('Models will be downloaded to ...)
# in my case '/home/twechs/data/conda/envs/delta_env/lib/python3.11/site-packages/delta/asse

python -c 'import delta; delta.assets.download_assets(load_models=True, load_sets=False,loac

# when finished with your test, exit the interactive cluster session

conda deactivate
exit
```

Download Models manually

The Models might not get downloaded because they are too big to be scanned by google drives virus scan. You can check this by having a look at the file sizes of the files in the model folder.

```
ls -lh data/conda/envs/delta_env/lib/python3.11/site-packages/delta/assets/models/
```

Output:

```
total 711M
-rw-rw-r-- 1 twechs twechs 2.4K Apr 24 14:50 unet_moma_seg.hdf5
-rw-rw-r-- 1 twechs twechs 2.4K Apr 24 14:50 unet_moma_track.hdf5
-rw-rw-r-- 1 twechs twechs 2.4K Apr 24 14:50 unet_momachambers_seg.hdf5
-rw-rw-r-- 1 twechs twechs 356M Apr 24 16:20 unet_pads_seg.hdf5
-rw-rw-r-- 1 twechs twechs 356M Apr 24 16:21 unet_pads_track.hdf5
```

Here I already replaced the Models 'unet_pads_seg.hdf5' and 'unet_pads_track.hdf5'. The file size should be around 356 MB. If that is not the case for you, you can download them manually from the link below [4], and replace them in your model folder.

https://drive.google.com/drive/folders/1nTRVo0rPP9CR9F6WUunVXSXrLNMT_zCP?usp=sharing

3. Submit a compute job to run DeLTA

Here is a quick example of submitting a job on SC. For the example I have saved the example images in the folder 'test_imgs', the SLURM submission script 'sub_script.sh' and the DeLTA script 'test_dlt.py' in the 'data' folder on SC.

Command to submit the job:

```
sbatch sub_script.sh
```

SLURM job submission script:

```
#!/bin/bash

### Comment lines start with ## or #+space
### Slurm option lines start with #SBATCH

### Here are the SBATCH parameters that you should always consider:
#SBATCH --time=0-01:00:00   ## days-hours:minutes:seconds
#SBATCH --mem 7GB         ## 3GB ram (hardware ratio is < 4GB/core)
#SBATCH --ntasks=1          ## Not strictly necessary because default is 1
#SBATCH --cpus-per-task=1   ## Use greater than 1 for parallelized jobs
#SBATCH --gpus=1

### Here are other SBATCH parameters that you may benefit from using, currently commented ou
###SBATCH --job-name=hello1 ## job name
###SBATCH --output=job.out  ## standard out file


echo 'started at:' $(date)
module load gpu
module load mamba
source activate delta_env
python ./test_dlt.py
echo 'finished at:' $(date)
```

DeLTA script:

```
import delta

delta.config.load_config(presets='2D')

source_folder = "test_imgs"

reader = delta.utilities.xpreader(
    source_folder,
    prototype='PA_%01d_C%02d_T%03d.tif',
    fileorder='pct',
    filenamesindexing=0
)

print("""Initialized experiment reader:
    - %d positions
    - %d imaging channels
    - %d timepoints"""%(reader.positions, reader.channels, reader.timepoints)
)
```

```
ppln = delta.pipeline.Pipeline(reader)

ppln.process()
```

The terminal output from the job is written to a file in your current directory (slurm-[job id].out) and the resulting files in 'test_imgs/delta_results'.

4. Submit array jobs for parallel processing

On ScienceCluster you can automatically start several jobs in parallel which can accelerate the process quite a bit. With the following scripts, you can submit a job for different positions you imaged. The script expects all images of each position in a separate folder.

SLURM array job submission script:

```
#!/bin/bash

### Comment lines start with ## or #+space
### Slurm option lines start with #SBATCH

### Here are the SBATCH parameters that you should always consider:
#SBATCH --time=0-01:00:00   ## days-hours:minutes:seconds
#SBATCH --mem 7GB         ## 3GB ram (hardware ratio is < 4GB/core)
#SBATCH --ntasks=1          ## Not strictly necessary because default is 1
#SBATCH --cpus-per-task=1   ## Use greater than 1 for parallelized jobs
#SBATCH --gpus=1
#SBATCH --array=0-4

### Here are other SBATCH parameters that you may benefit from using, currently commented ou
###SBATCH --job-name=hello1 ## job name
###SBATCH --output=job.out  ## standard out file

## The following creates a list of all directories in 'inputdir'
posArr=($(ls -d test_positions/*/))

## The array of the input directory can use the TASK_ID as an index
## to set the input direcory for each job
echo 'started at:' $(date)
module load gpu
module load mamba
source activate delta_env
python ./run_dlt.py ${posArr[$SLURM_ARRAY_TASK_ID]}
echo 'finished at:' $(date)
```

DeLTA script for array job:

```
import delta
import os
import argparse

parser = argparse.ArgumentParser(description='Run DeLTA segmantation and tracking on the ima
parser.add_argument('directory', type=str, help="The path of the directory to parse")

args = parser.parse_args()

source_folder = args.directory

if not os.path.isdir(source_folder):
    print(f"Error: The directory '{source_folder}' does not exist.")
    sys.exit(1)

delta.config.load_config(presets='2D')

reader = delta.utilities.xpreader(
    source_folder,
    prototype='PA_0_C%02d_T%03d.tif',
    fileorder='ct',
    filenamesindexing=0
)

print("""Initialized experiment reader:
    - %d positions
    - %d imaging channels
    - %d timepoints"""%(reader.positions, reader.channels, reader.timepoints)
)


ppln = delta.pipeline.Pipeline(reader)

ppln.process()
```

Resources

1. ScienceCluster Training
2. DeLTA Installation
3. Example Python-Environments with GPU
4. DeLTA Models
5. ScienceCluster job array example