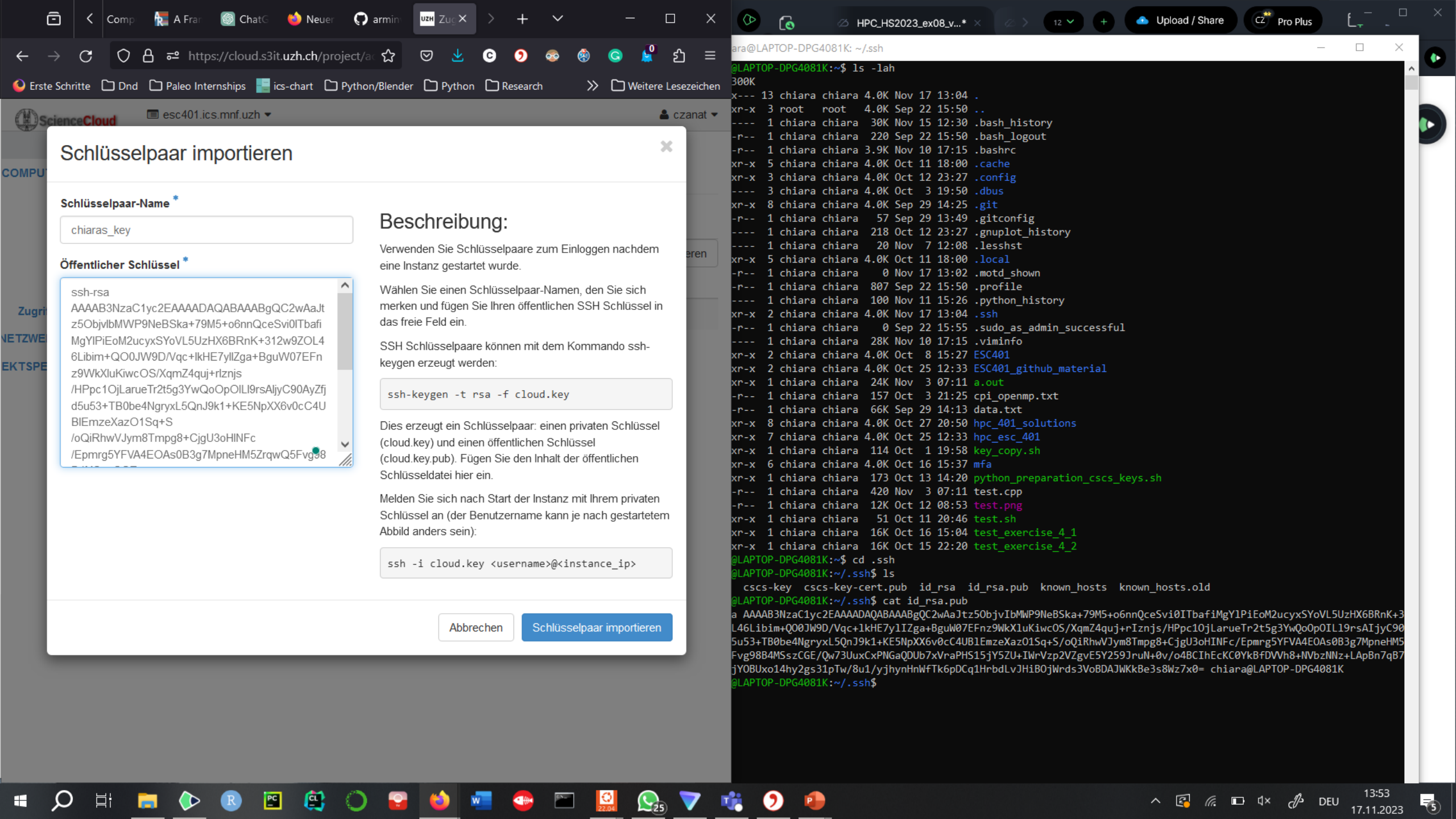
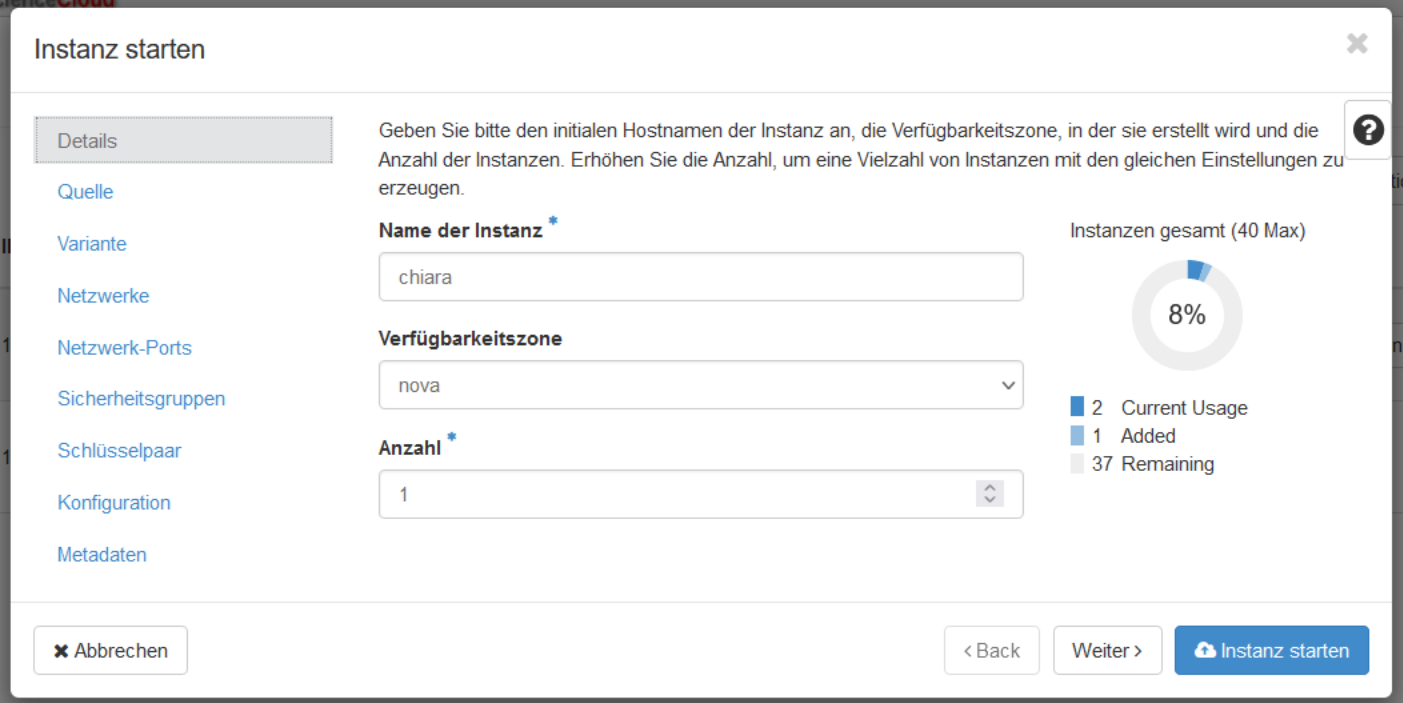

Schlüsselpaar importieren
Schlüsselpaar-Name *
chiaras_key
Öffentlicher Schlüssel *
ssh-rsa
AAAAB3NzaC1yc2EAAAADAQABAAABgQC2wAaJtz5ObjvIbMWP9NeBSka+79M5+o6nnQceSvi0ITbafiMgYlPiEoM2ucyxSYoVL5UzHX6BRnK+312w9ZOL46Libim+QO0JW9D/Vqc+lkHE7ylIZga+BguW07EFnz9WkXluKiwcOS/XqmZ4quj+rIznjs
/HPpc1OjLarueTr2t5g3YwQoOpOIL19rsAljyC90AyZfjd5u53+TB0be4NgryxL5QnJ9k1+KE5NpXX6v0cC4UBlEmzeXazO1Sq+S
/oQiRhwVJym8Tmpg8+CjgU3oHlNFc
/Epmrg5YFVA4EOAs0B3g7MpneHM5ZrqwQ5Fvg98
Beschreibung:
Verwenden Sie Schlüsselpaare zum Einloggen nachdem eine Instanz gestartet wurde.
Wählen Sie einen Schlüsselpaar-Namen, den Sie sich merken und fügen Sie Ihren öffentlichen SSH Schlüssel in das freie Feld ein.
SSH Schlüsselpaare können mit dem Kommando ssh-keygen erzeugt werden:
ssh-keygen -t rsa -f cloud.key
Dies erzeugt ein Schlüsselpaar: einen privaten Schlüssel (cloud.key) und einen öffentlichen Schlüssel (cloud.key.pub). Fügen Sie den Inhalt der öffentlichen Schlüsseldatei hier ein.
Melden Sie sich nach Start der Instanz mit Ihrem privaten Schlüssel an (der Benutzername kann je nach gestartetem Abbild anders sein):
ssh -i cloud.key <username>@<instance_ip>
Abbrechen
Schlüsselpaar importieren



his exercise, you will create and set up a virtual machine (VM) instance on the Science
ud.

1

- Connect and login to https://cloud.s3it.uzh.ch with your UZH Shortname and your Webpass password.
  Create an Ubuntu 18.04 instance with 8 cores and 32 GB of RAM. You can follow, step by step, the instructions given in the lecture slides.
  Once the instance has been created, you can login through `ssh` with the IP address relative to your instance. Remember that you need to connect to the UZH VPN first. If you haven't set up your VPN yet, follow these instructions: https://www.zi.uzh.ch/de/support/netzwerk/vpn.html
  (***Commit:***) Finally, update the list of packages and install the `cowsay` utility and test that your Ubuntu installation works fine.

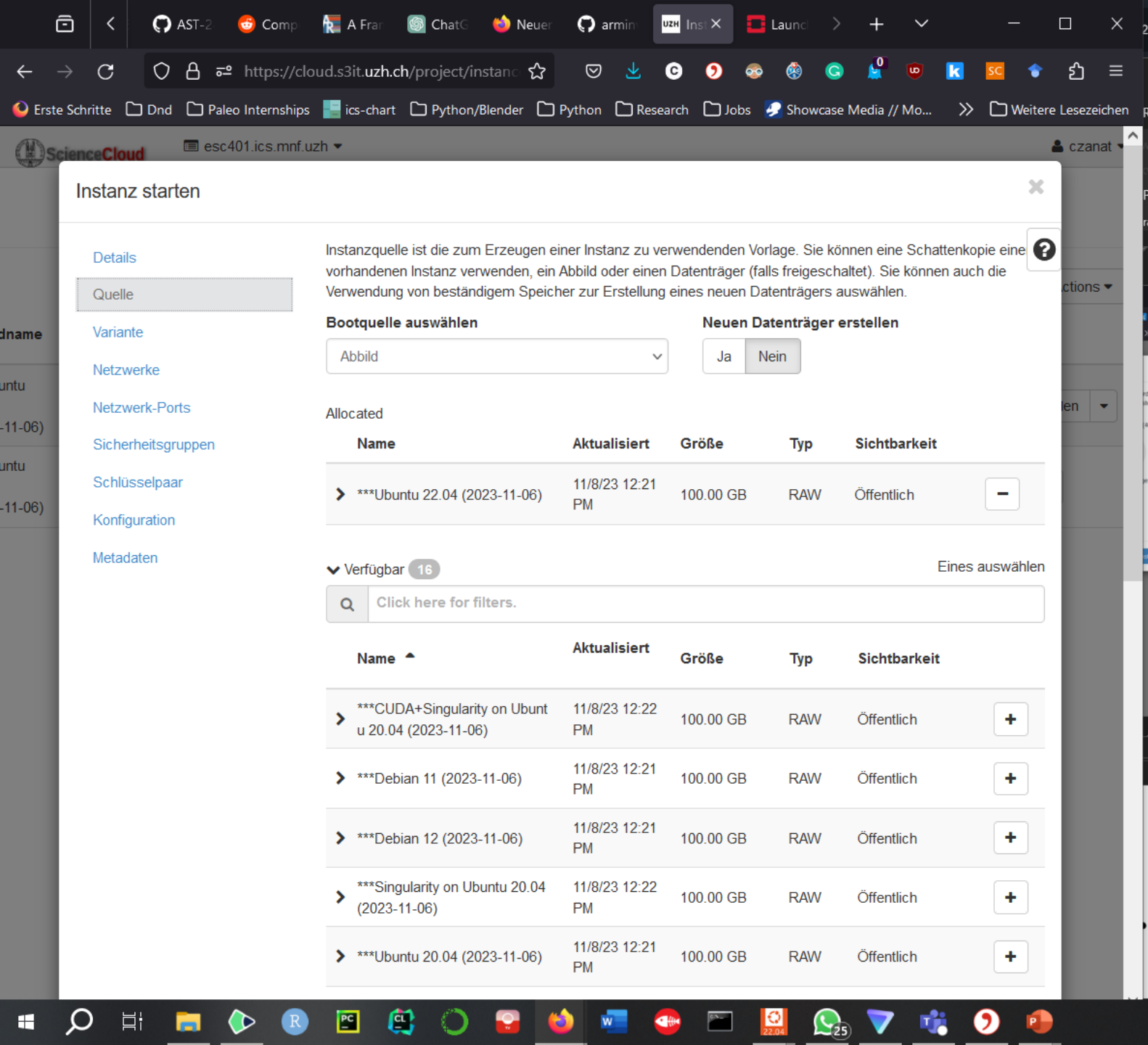
- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:

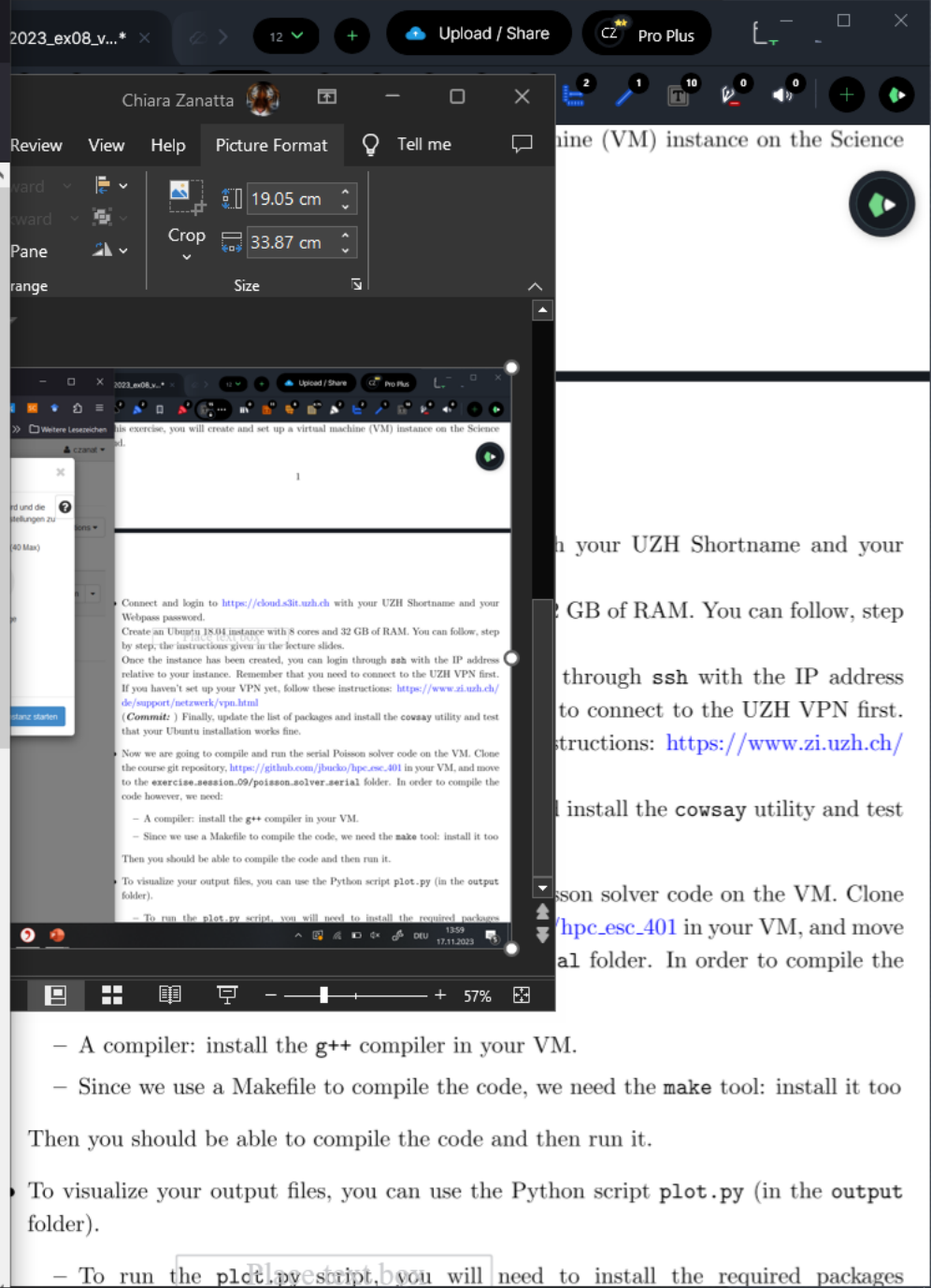
  – A compiler: install the `g++` compiler in your VM.
  – Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).

  – To run the `plot.py` script, you will need to install the required packages





hine (VM) instance on the Science

n your UZH Shortname and your

GB of RAM. You can follow, step

through ssh with the IP address
to connect to the UZH VPN first.
structions: https://www.zi.uzh.ch/

l install the cowsay utility and test

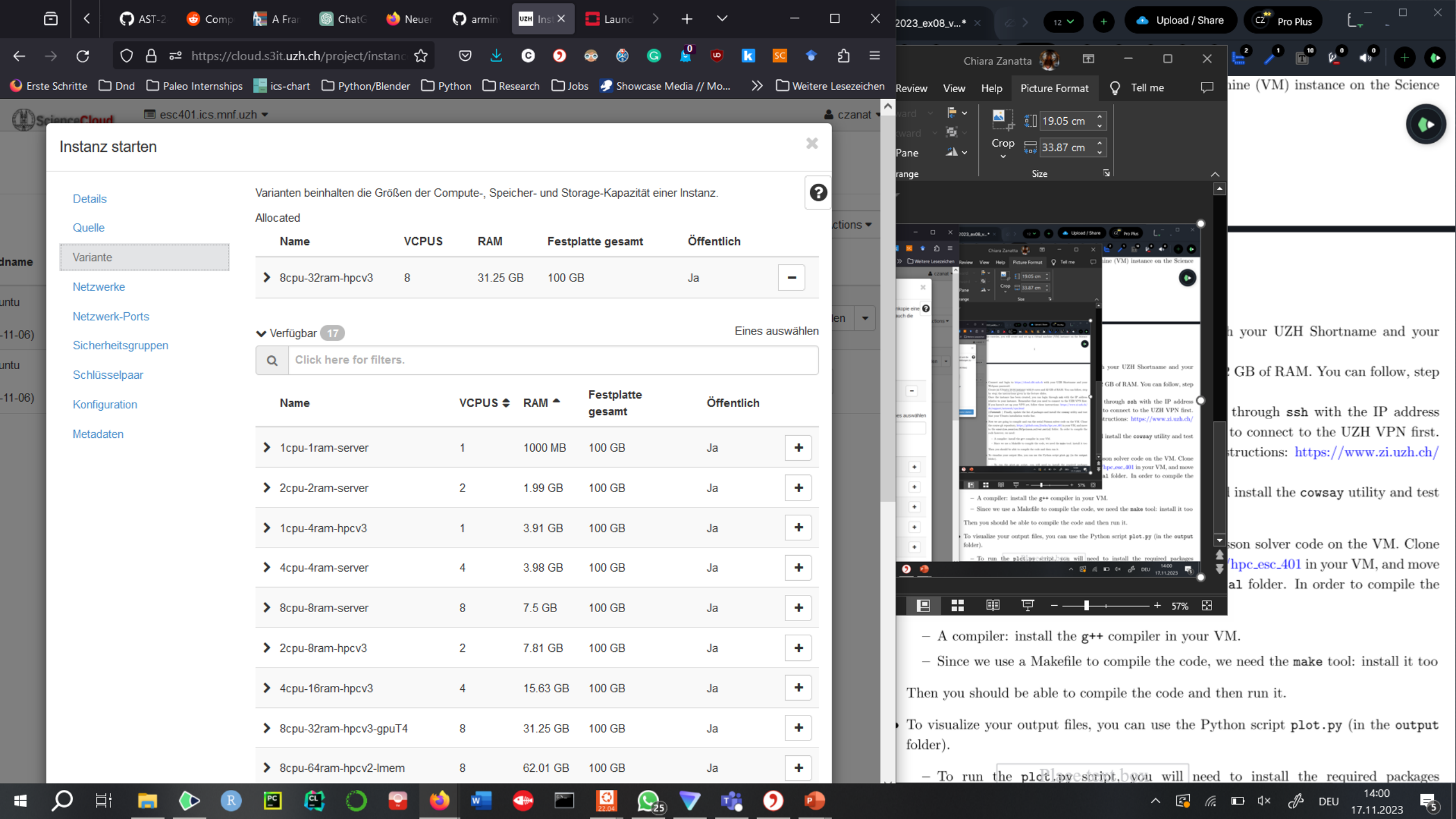
sson solver code on the VM. Clone
/hpc_esc_401 in your VM, and move
al folder. In order to compile the

– A compiler: install the g++ compiler in your VM.
– Since we use a Makefile to compile the code, we need the make tool: install it too

Then you should be able to compile the code and then run it.

To visualize your output files, you can use the Python script plot.py (in the output folder).

– To run the plot.py script, you will need to install the required packages


Instanz starten
Details
Quelle
Variante
Netzwerke
Netzwerk-Ports
Sicherheitsgruppen
Schlüsselpaar
Konfiguration
Metadaten
Varianten beinhalten die Größen der Compute-, Speicher- und Storage-Kapazität einer Instanz.
Allocated
Name VCPUS RAM Festplatte gesamt Öffentlich
8cpu-32ram-hpcv3 8 31.25 GB 100 GB Ja
Verfügbar 17
Eines auswählen
Click here for filters.
Name VCPUS RAM Festplatte gesamt Öffentlich
1cpu-1ram-server 1 1000 MB 100 GB Ja
2cpu-2ram-server 2 1.99 GB 100 GB Ja
1cpu-4ram-hpcv3 1 3.91 GB 100 GB Ja
4cpu-4ram-server 4 3.98 GB 100 GB Ja
8cpu-8ram-server 8 7.5 GB 100 GB Ja
2cpu-8ram-hpcv3 2 7.81 GB 100 GB Ja
4cpu-16ram-hpcv3 4 15.63 GB 100 GB Ja
8cpu-32ram-hpcv3-gpuT4 8 31.25 GB 100 GB Ja
8cpu-64ram-hpcv2-lmem 8 62.01 GB 100 GB Ja

Projekt

COMPUTE
- Übersicht
- Instanzen
- Datenträger
- Abbilder
- Zugriff & Sicherheit

NETZWERK

OBJEKTSPEICHER

Identität

Instanzen

Instanzname = | Filter | Instanz starten | Instanzen löschen | More Actions

	Instanzname	Abbildname	IP-Adresse	Größe	Schlüsselpaar	Status	Verfügbarkeitszone	Aufgabe	Zustand	Zeit seit Erzeugung	Actions
☐	chiara	***Ubuntu 22.04 (2023-11-06)	172.23.53.192	8cpu-32ram-hpcv3	chiaras_key	Baue	nova	wird erzeugt	Kein Zustand	0 Minuten	Floating IP zuweisen
☐	Roman_HPC	***Ubuntu 22.04 (2023-11-06)	172.23.53.191	8cpu-32ram-hpcv3	roman_key	Aktiv	nova	Keine	Läuft	1 Minute	Schattenkopie erstellen
☐	docker-test-nk	***Ubuntu 22.04 (2023-11-06)	172.23.53.189	8cpu-32ram-hpcv3	nkl3	Aktiv	nova	Keine	Läuft	2 Stunden, 1 Minute	Schattenkopie erstellen
☐	nknk	***Ubuntu 20.04 (2023-11-06)	172.23.53.1	8cpu-32ram-hpcv3	nkl3	Abschaltung	nova	Keine	Heruntergefahren	20 Stunden, 48 Minuten	Instanz starten

Displaying 4 items

ubuntu@chiara: ~

```
chiara@LAPTOP-DPG4081K:~$ cd .ssh
chiara@LAPTOP-DPG4081K:~/.ssh$ ssh -i id_rsa ubuntu@172.23.53.192
^C
chiara@LAPTOP-DPG4081K:~/.ssh$ cd
chiara@LAPTOP-DPG4081K:~$ ssh -i .ssh/id_rsa ubuntu@172.23.53.198
^C
chiara@LAPTOP-DPG4081K:~$ ssh -i .ssh/id_rsa ubuntu@172.23.53.198
The authenticity of host '172.23.53.198 (172.23.53.198)' can't be established.
ED25519 key fingerprint is SHA256:5dYVZdczL/AExtQS4i5atGzRRcHkaU3vfNeZh9t+mbQ.
This key is not known by any other names
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '172.23.53.198' (ED25519) to the list of known hosts.
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Nov 17 13:16:04 UTC 2023

  System load:  0.16064453125     Processes:             147
  Usage of /:   1.9% of 96.73GB   Users logged in:       0
  Memory usage: 0%                IPv4 address for ens3: 172.23.53.198
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

0 updates can be applied immediately.

Enable ESM Apps to receive additional future security updates.
See https://ubuntu.com/esm or run: sudo pro status


The list of available updates is more than a week old.
To check for new updates run: sudo apt update


The programs included with the Ubuntu system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Ubuntu comes with ABSOLUTELY NO WARRANTY, to the extent permitted by
applicable law.

Last login: Mon Nov  6 13:20:59 2023 from 172.23.255.4
ubuntu@chiara:~$
```

HPC_HS2023_ex08_v...*

Upload / Share

Pro Plus

In this exercise, you will create and set up a virtual machine (VM) instance on the Science Cloud.

- Connect and login to https://cloud.s3it.uzh.ch with your UZH Shortname and your Webpass password.
  Create an Ubuntu 18.04 instance with 8 cores and 32 GB of RAM. You can follow, step by step, the instructions given in the lecture slides.
  Once the instance has been created, you can login through `ssh` with the IP address relative to your instance. Remember that you need to connect to the UZH VPN first. If you haven't set up your VPN yet, follow these instructions: https://www.zi.uzh.ch/de/support/netzwerk/vpn.html
  (***Commit:***) Finally, update the list of packages and install the `cowsay` utility and test that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:

  – A compiler: install the `g++` compiler in your VM.
  – Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).

  – To run the `plot.py` script, you will need to install the required packages

DEU 14:16 17.11.2023

```
ubuntu@chiara: ~
Last login: Mon Nov  6 13:20:59 2023 from 172.23.255.4
ubuntu@chiara:~$ sudo apt update
Hit:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy InRelease
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates InRelease [119 kB]
Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-backports InRelease [109 kB]
Get:4 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 Packages [1163 kB]
Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main Translation-en [248 kB]
Get:6 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/restricted amd64 Packages [1114 kB]
Get:7 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/restricted Translation-en [181 kB]
Get:8 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 Packages [998 kB]
Get:9 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/universe Translation-en [219 kB]
Get:10 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [953 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [188 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [1093 kB]
Get:14 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [178 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [795 kB]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [147 kB]
Fetched 7613 kB in 8s (973 kB/s)
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
9 packages can be upgraded. Run 'apt list --upgradable' to see them.
ubuntu@chiara:~$ apt list --upgradable
Listing... Done
apt-utils/jammy-updates 2.4.11 amd64 [upgradable from: 2.4.10]
apt/jammy-updates 2.4.11 amd64 [upgradable from: 2.4.10]
kpartx/jammy-updates 0.8.8-1ubuntu1.22.04.3 amd64 [upgradable from: 0.8.8-1ubuntu1.22.04.1]
libapt-pkg6.0/jammy-updates 2.4.11 amd64 [upgradable from: 2.4.10]
libprocps8/jammy-updates,jammy-security 2:3.3.17-6ubuntu2.1 amd64 [upgradable from: 2:3.3.17-6ubuntu2]
multipath-tools/jammy-updates 0.8.8-1ubuntu1.22.04.3 amd64 [upgradable from: 0.8.8-1ubuntu1.22.04.1]
procps/jammy-updates,jammy-security 2:3.3.17-6ubuntu2.1 amd64 [upgradable from: 2:3.3.17-6ubuntu2]
python3-urllib3/jammy-updates,jammy-security 1.26.5-1~exp1ubuntu0.1 all [upgradable from: 1.26.5-1~exp1]
ubuntu-advantage-tools/jammy-updates 30~22.04 amd64 [upgradable from: 29.4~22.04]
ubuntu@chiara:~$ sudo apt-get install cowsay
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  filters cowsay-off
The following NEW packages will be installed:
  cowsay
0 upgraded, 1 newly installed, 0 to remove and 9 not upgraded.
Need to get 18.6 kB of archives.
After this operation, 93.2 kB of additional disk space will be used.
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 cowsay all 3.03+dfsg2-8 [18.6 kB]
Fetched 18.6 kB in 0s (64.1 kB/s)
Selecting previously unselected package cowsay.
(Reading database ... 95279 files and directories currently installed.)
Preparing to unpack .../cowsay_3.03+dfsg2-8_all.deb ...
Unpacking cowsay (3.03+dfsg2-8) ...
Setting up cowsay (3.03+dfsg2-8) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...
```

HPC_HS2023_ex08_v...* — Upload / Share — Pro Plus

- Connect and login to https://cloud.s3it.uzh.ch with your UZH Shortname and your Webpass password.
  Create an Ubuntu 18.04 instance with 8 cores and 32 GB of RAM. You can follow, step by step, the instructions given in the lecture slides.
  Once the instance has been created, you can login through `ssh` with the IP address relative to your instance. Remember that you need to connect to the UZH VPN first. If you haven't set up your VPN yet, follow these instructions: https://www.zi.uzh.ch/de/support/netzwerk/vpn.html
  (***Commit:***) Finally, update the list of packages and install the `cowsay` utility and test that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:

  – A compiler: install the `g++` compiler in your VM.

  – Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).

  – To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover,

Place text box

DEU 14:19 17.11.2023

ubuntu@chiara: ~

```
ubuntu-advantage-tools/jammy-updates 30~22.04 amd64 [upgradable from: 29.4~22.04]
ubuntu@chiara:~$ sudo apt-get install cowsay
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
Suggested packages:
  filters cowsay-off
The following NEW packages will be installed:
  cowsay
0 upgraded, 1 newly installed, 0 to remove and 9 not upgraded.
Need to get 18.6 kB of archives.
After this operation, 93.2 kB of additional disk space will be used.
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 cowsay all 3.03+dfsg2-8 [18.6 kB]
Fetched 18.6 kB in 0s (64.1 kB/s)
Selecting previously unselected package cowsay.
(Reading database ... 95279 files and directories currently installed.)
Preparing to unpack .../cowsay_3.03+dfsg2-8_all.deb ...
Unpacking cowsay (3.03+dfsg2-8) ...
Setting up cowsay (3.03+dfsg2-8) ...
Processing triggers for man-db (2.10.2-1) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@chiara:~$ cowplot hi
Command 'cowplot' not found, did you mean:
  command 'colplot' from deb colplot (5.2.0-1.1)
Try: sudo apt install <deb name>
ubuntu@chiara:~$ cowsay how
 _____
< how >
 -----
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
ubuntu@chiara:~$ cowsay hello there
 _____________
< hello there >
 -------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
ubuntu@chiara:~$
```

HPC_HS2023_ex08_v...* — Upload / Share — Pro Plus

- Connect and login to https://cloud.s3it.uzh.ch with your UZH Shortname and your Webpass password.
  Create an Ubuntu 18.04 instance with 8 cores and 32 GB of RAM. You can follow, step by step, the instructions given in the lecture slides.
  Once the instance has been created, you can login through `ssh` with the IP address relative to your instance. Remember that you need to connect to the UZH VPN first. If you haven't set up your VPN yet, follow these instructions: https://www.zi.uzh.ch/de/support/netzwerk/vpn.html
  (***Commit:***) Finally, update the list of packages and install the `cowsay` utility and test that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:

  – A compiler: install the `g++` compiler in your VM.
  – Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).

  – To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover,

DEU 14:20 17.11.2023

ubuntu@chiara: ~/hpc_esc_401/exercise_session_09/poisson_solver_serial

```
 -----
       \   ^__^
        \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
ubuntu@chiara:~$ cowsay hello there
 _____________
< hello there >
 -------------
       \   ^__^
        \  (oo)\_______
           (__)\       )\/\
               ||----w |
               ||     ||
ubuntu@chiara:~$ git clone https://github.com/jbucko/hpc_esc_401
Cloning into 'hpc_esc_401'...
remote: Enumerating objects: 297, done.
remote: Counting objects: 100% (297/297), done.
remote: Compressing objects: 100% (213/213), done.
remote: Total 297 (delta 118), reused 233 (delta 69), pack-reused 0
Receiving objects: 100% (297/297), 2.13 MiB | 11.70 MiB/s, done.
Resolving deltas: 100% (118/118), done.
ubuntu@chiara:~$ ls
hpc_esc_401
ubuntu@chiara:~$ cd hpc_esc_401/exercise_session_09/poisson_solver_serial/
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial$ ls
Makefile  init.cpp  init.h  io.cpp  io.h  jacobi.cpp  jacobi.h  main.cpp  output  params.txt
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial$
```

HPC_HS2023_ex08_v...*

Upload / Share

Pro Plus

Once the instance has been created, you can login through `ssh` with the IP address relative to your instance. Remember that you need to connect to the UZH VPN first. If you haven't set up your VPN yet, follow these instructions: https://www.zi.uzh.ch/de/support/netzwerk/vpn.html
(***Commit:***) Finally, update the list of packages and install the `cowsay` utility and test that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:
  - A compiler: install the `g++` compiler in your VM.
  - Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).
  - To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover, you will have to install `Tkinter`, which is the Python interface to the Tk GUI toolkit shipped with Python: `pip3 install tk`, then `sudo apt install -q python-tk`. (if installation of any python package fails with error `Failed building wheel...`, consider updating wheel using `python3 -m pip install --upgrade pip setuptools wheel`. Also, `plot.py` is written with `python3`, so if running command `python -V` prints "Python 2.x.xx" then you should use command `python3` instead.

- Create and save a snapshot of your instance on the Science Cloud.

```
import matplotlib
matplotlib.use("TkAgg")
```

Commit: Provide the step-by-step solution on how you managed to compile and run Poisson solver and create the snapshot on the Science cloud.

DEU 14:22 17.11.2023

ubuntu@chiara: ~/hpc_esc_401/exercise_session_09/poisson_solver_serial

```
< hello there >
 -------------
        \   ^__^
         \  (oo)\_______
            (__)\       )\/\
                ||----w |
                ||     ||
ubuntu@chiara:~$ git clone https://github.com/jbucko/hpc_esc_401
Cloning into 'hpc_esc_401'...
remote: Enumerating objects: 297, done.
remote: Counting objects: 100% (297/297), done.
remote: Compressing objects: 100% (213/213), done.
remote: Total 297 (delta 118), reused 233 (delta 69), pack-reused 0
Receiving objects: 100% (297/297), 2.13 MiB | 11.70 MiB/s, done.
Resolving deltas: 100% (118/118), done.
ubuntu@chiara:~$ ls
hpc_esc_401
ubuntu@chiara:~$ cd hpc_esc_401/exercise_session_09/poisson_solver_serial/
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial$ ls
Makefile  init.cpp  init.h  io.cpp  io.h  jacobi.cpp  jacobi.h  main.cpp  output  params.txt
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial$ sudo apt install g++ make cmake
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  cmake-data cpp cpp-11 dh-elpa-helper emacsen-common fontconfig-config fonts-dejavu-core g++-11 gcc gcc-11
  gcc-11-base libasan6 libatomic1 libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libdeflate0
  libfontconfig1 libgcc-11-dev libgd3 libgomp1 libisl23 libitm1 libjbig0 libjpeg-turbo8 libjpeg8 libjsoncpp25
  liblsan0 libmpc3 libnsl-dev libquadmath0 librhash0 libstdc++-11-dev libtiff5 libtirpc-dev libtsan0 libubsan1
  libwebp7 libxpm4 linux-libc-dev manpages-dev rpcsvc-proto
Suggested packages:
  cmake-doc ninja-build cmake-format cpp-doc gcc-11-locales g++-multilib g++-11-multilib gcc-11-doc gcc-multilib
  autoconf automake libtool flex bison gdb gcc-doc gcc-11-multilib glibc-doc libgd-tools libstdc++-11-doc make-doc
The following NEW packages will be installed:
  cmake cmake-data cpp cpp-11 dh-elpa-helper emacsen-common fontconfig-config fonts-dejavu-core g++ g++-11 gcc
  gcc-11 gcc-11-base libasan6 libatomic1 libc-dev-bin libc-devtools libc6-dev libcc1-0 libcrypt-dev libdeflate0
  libfontconfig1 libgcc-11-dev libgd3 libgomp1 libisl23 libitm1 libjbig0 libjpeg-turbo8 libjpeg8 libjsoncpp25
  liblsan0 libmpc3 libnsl-dev libquadmath0 librhash0 libstdc++-11-dev libtiff5 libtirpc-dev libtsan0 libubsan1
  libwebp7 libxpm4 linux-libc-dev make manpages-dev rpcsvc-proto
0 upgraded, 47 newly installed, 0 to remove and 9 not upgraded.
Need to get 69.5 MB of archives.
After this operation, 234 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libjsoncpp25 amd64 1.9.5-3 [80.0 kB]
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 librhash0 amd64 1.4.2-1ubuntu1 [125 kB]
Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 dh-elpa-helper all 2.0.9ubuntu1 [7610 B]
Get:4 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 emacsen-common all 3.0.4 [14.9 kB]
Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cmake-data all 3.22.1-1ubuntu1.22.04.1 [1
913 kB]
Get:6 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 cmake amd64 3.22.1-1ubuntu1.22.04.1 [5013
 kB]
Get:7 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gcc-11-base amd64 11.4.0-1ubuntu1~22.04 [
20.2 kB]
Get:8 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libisl23 amd64 0.24-2build1 [727 kB]
Get:9 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libmpc3 amd64 1.2.1-2build1 [46.9 kB]
```

HPC_HS2023_ex08_v...* — Upload / Share — Pro Plus

Once the instance has been created, you can login through `ssh` with the IP address relative to your instance. Remember that you need to connect to the UZH VPN first. If you haven't set up your VPN yet, follow these instructions: https://www.zi.uzh.ch/de/support/netzwerk/vpn.html
(***Commit:***) Finally, update the list of packages and install the `cowsay` utility and test that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:
  - A compiler: install the `g++` compiler in your VM. Add text here
  - Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).
  - To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover, you will have to install `Tkinter`, which is the Python interface to the Tk GUI toolkit shipped with Python: `pip3 install tk`, then `sudo apt install -q python-tk`. (if installation of any python package fails with error `Failed building wheel...`, consider updating wheel using `python3 -m pip install --upgrade pip setuptools wheel`. Also, `plot.py` is written with `python3`, so if running command `python -V` prints "Python 2.x.xx" then you should use command `python3` instead.

- Create and save a snapshot of your instance on the Science Cloud.

```
import matplotlib
matplotlib.use("TkAgg")
```

Commit: Provide the step-by-step solution on how you managed to compile and run Poisson solver and create the snapshot on the Science cloud.

DEU 14:25 17.11.2023

ubuntu@chiara: ~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output

```
source = genfromtxt(path_to_source, delimiter=',')
data = genfromtxt(path_to_output, delimiter=',')

nstep = int(path_to_output[7:14])

# plot the map
fig, ax = plt.subplots(1,2, figsize=(7,3), sharex=True)
fig.subplots_adjust(left=0.02, bottom=0.06, right=0.95, top=0.94, wspace=0.05)


im1 = ax[0].imshow(source.T, origin='lower')
im2 = ax[1].imshow(data.T, origin='lower')

ax[0].set_xlabel("ny")
ax[1].set_xlabel("ny")

ax[0].set_ylabel("nx")

ax[0].set_title("Source term")
ax[1].set_title("Approx. Solution, step={}".format(nstep))

fig.colorbar(im1, ax=ax[0])
fig.colorbar(im2, ax=ax[1])

plt.tight_layout()
plt.savefig("test.png")
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ sudo apt install -q python3-pip
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  build-essential bzip2 dpkg-dev fakeroot javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libjs-jquery
  libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev lto-disabled-list python3-dev python3-wheel
  python3.10-dev zlib1g-dev
Suggested packages:
  bzip2-doc debian-keyring apache2 | lighttpd | httpd bzr
The following NEW packages will be installed:
  build-essential bzip2 dpkg-dev fakeroot javascript-common libalgorithm-diff-perl libalgorithm-diff-xs-perl
  libalgorithm-merge-perl libdpkg-perl libexpat1-dev libfakeroot libfile-fcntllock-perl libjs-jquery
  libjs-sphinxdoc libjs-underscore libpython3-dev libpython3.10-dev lto-disabled-list python3-dev python3-pip
  python3-wheel python3.10-dev zlib1g-dev
0 upgraded, 23 newly installed, 0 to remove and 9 not upgraded.
Need to get 8938 kB of archives.
After this operation, 37.5 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libdpkg-perl all 1.21.1ubuntu2.2 [237 kB]
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 bzip2 amd64 1.0.8-5build1 [34.8 kB]
Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 lto-disabled-list all 24 [12.5 kB]
Get:4 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 dpkg-dev all 1.21.1ubuntu2.2 [922 kB]
Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 build-essential amd64 12.9ubuntu3 [4744 B]
Get:6 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libfakeroot amd64 1.28-1ubuntu1 [31.5 kB]
Get:7 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 fakeroot amd64 1.28-1ubuntu1 [60.4 kB]
Get:8 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 javascript-common all 11+nmu1 [5936 B]
Get:9 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libalgorithm-diff-perl all 1.201-1 [41.8 kB]
```

HPC_HS2023_ex08_v...* | Upload / Share | Pro Plus

that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:
  - A compiler: install the `g++` compiler in your VM. Add text here
  - Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).
  - To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover, you will have to install `Tkinter`, which is the Python interface to the Tk GUI toolkit shipped with Python: `pip3 install tk`, then `sudo apt install -q python-tk`. (if installation of any python package fails with error `Failed building wheel...`, consider updating wheel using `python3 -m pip install --upgrade pip setuptools wheel`. Also, `plot.py` is written with `python3`, so if running command `python -V` prints "Python 2.x.xx" then you should use command `python3` instead.

- Create and save a snapshot of your instance on the Science Cloud.

```
import matplotlib
matplotlib.use("TkAgg")
```

Commit: Provide the step-by-step solution on how you managed to compile and run Poisson solver and create the snapshot on the Science cloud.

DEU 14:30 17.11.2023

ubuntu@chiara: ~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output

```
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ cat plot.py
#!/usr/bin/env python3

import numpy as np
import matplotlib.pyplot as plt
from matplotlib.colors import  LogNorm
from numpy import genfromtxt

# path the the file
path_to_output = "output_0014000.csv"
path_to_source = "output_source.csv"


# read image data
source = genfromtxt(path_to_source, delimiter=',')
data = genfromtxt(path_to_output, delimiter=',')

nstep = int(path_to_output[7:14])

# plot the map
fig, ax = plt.subplots(1,2, figsize=(7,3), sharex=True)
fig.subplots_adjust(left=0.02, bottom=0.06, right=0.95, top=0.94, wspace=0.05)


im1 = ax[0].imshow(source.T, origin='lower')
im2 = ax[1].imshow(data.T, origin='lower')

ax[0].set_xlabel("ny")
ax[1].set_xlabel("ny")

ax[0].set_ylabel("nx")

ax[0].set_title("Source term")
ax[1].set_title("Approx. Solution, step={}".format(nstep))

fig.colorbar(im1, ax=ax[0])
fig.colorbar(im2, ax=ax[1])

plt.tight_layout()
plt.savefig("test.png")
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ pip3 install numpy
```

HPC_HS2023_ex08_v...*

Upload / Share

Pro Plus

that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:
  - A compiler: install the `g++` compiler in your VM. Add text here
  - Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).
  - To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover, you will have to install `Tkinter`, which is the Python interface to the Tk GUI toolkit shipped with Python: `pip3 install tk`, then `sudo apt install -q python-tk`. (if installation of any python package fails with error `Failed building wheel...`, consider updating wheel using `python3 -m pip install --upgrade pip setuptools wheel`. Also, `plot.py` is written with `python3`, so if running command `python -V` prints "Python 2.x.xx" then you should use command `python3` instead.

- Create and save a snapshot of your instance on the Science Cloud.

```
import matplotlib
matplotlib.use("TkAgg")
```

Commit: Provide the step-by-step solution on how you managed to compile and run Poisson solver and create the snapshot on the Science cloud.

DEU 14:31 17.11.2023

ubuntu@chiara: ~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output

```
ax[0].set_title("Source term")
ax[1].set_title("Approx. Solution, step={}".format(nstep))

fig.colorbar(im1, ax=ax[0])
fig.colorbar(im2, ax=ax[1])

plt.tight_layout()
plt.savefig("test.png")
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ pip3 install numpy
Defaulting to user installation because normal site-packages is not writeable
Collecting numpy
  Downloading numpy-1.26.2-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (18.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 18.2/18.2 MB 31.8 MB/s eta 0:00:00
Installing collected packages: numpy
  WARNING: The script f2py is installed in '/home/ubuntu/.local/bin' which is not on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed numpy-1.26.2
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ pip3 install matplotlib
Defaulting to user installation because normal site-packages is not writeable
Collecting matplotlib
  Downloading matplotlib-3.8.1-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.6/11.6 MB 8.3 MB/s eta 0:00:00
Collecting pillow>=8
  Downloading Pillow-10.1.0-cp310-cp310-manylinux_2_28_x86_64.whl (3.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 3.6/3.6 MB 4.7 MB/s eta 0:00:00
Requirement already satisfied: pyparsing>=2.3.1 in /usr/lib/python3/dist-packages (from matplotlib) (2.4.7)
Collecting packaging>=20.0
  Downloading packaging-23.2-py3-none-any.whl (53 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.0/53.0 KB 4.0 MB/s eta 0:00:00
Collecting contourpy>=1.0.1
  Downloading contourpy-1.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (310 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 310.7/310.7 KB 4.8 MB/s eta 0:00:00
Collecting kiwisolver>=1.3.1
  Downloading kiwisolver-1.4.5-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 15.0 MB/s eta 0:00:00
Requirement already satisfied: numpy<2,>=1.21 in /home/ubuntu/.local/lib/python3.10/site-packages (from matplotlib) (
1.26.2)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.44.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 6.5 MB/s eta 0:00:00
Collecting cycler>=0.10
  Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.7/247.7 KB 13.0 MB/s eta 0:00:00
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python-dateutil>=2.7->matplotlib) (1.
16.0)
Installing collected packages: python-dateutil, pillow, packaging, kiwisolver, fonttools, cycler, contourpy, matplotl
ib
  WARNING: The scripts fonttools, pyftmerge, pyftsubset and ttx are installed in '/home/ubuntu/.local/bin' which is n
ot on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed contourpy-1.2.0 cycler-0.12.1 fonttools-4.44.3 kiwisolver-1.4.5 matplotlib-3.8.1 packaging-23.
2 pillow-10.1.0 python-dateutil-2.8.2
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$
```

HPC_HS2023_ex08_v...*

Upload / Share

Pro Plus

that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:
  - A compiler: install the `g++` compiler in your VM. Add text here
  - Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).
  - To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover, you will have to install `Tkinter`, which is the Python interface to the Tk GUI toolkit shipped with Python: `pip3 install tk`, then `sudo apt install -q python-tk`. (if installation of any python package fails with error `Failed building wheel...`, consider updating wheel using `python3 -m pip install --upgrade pip setuptools wheel`. Also, `plot.py` is written with `python3`, so if running command `python -V` prints "Python 2.x.xx" then you should use command `python3` instead.

- Create and save a snapshot of your instance on the Science Cloud.

```
import matplotlib
matplotlib.use("TkAgg")
```

Commit: Provide the step-by-step solution on how you managed to compile and run Poisson solver and create the snapshot on the Science cloud.

DEU 14:33 17.11.2023

ubuntu@chiara: ~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output

```
Requirement already satisfied: pyparsing>=2.3.1 in /usr/lib/python3/dist-packages (from matplotlib) (2.4.7)
Collecting packaging>=20.0
  Downloading packaging-23.2-py3-none-any.whl (53 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 53.0/53.0 KB 4.0 MB/s eta 0:00:00
Collecting contourpy>=1.0.1
  Downloading contourpy-1.2.0-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (310 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 310.7/310.7 KB 4.8 MB/s eta 0:00:00
Collecting kiwisolver>=1.3.1
  Downloading kiwisolver-1.4.5-cp310-cp310-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 15.0 MB/s eta 0:00:00
Requirement already satisfied: numpy<2,>=1.21 in /home/ubuntu/.local/lib/python3.10/site-packages (from matplotlib) (
1.26.2)
Collecting fonttools>=4.22.0
  Downloading fonttools-4.44.3-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (4.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 6.5 MB/s eta 0:00:00
Collecting cycler>=0.10
  Downloading cycler-0.12.1-py3-none-any.whl (8.3 kB)
Collecting python-dateutil>=2.7
  Downloading python_dateutil-2.8.2-py2.py3-none-any.whl (247 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 247.7/247.7 KB 13.0 MB/s eta 0:00:00
Requirement already satisfied: six>=1.5 in /usr/lib/python3/dist-packages (from python-dateutil>=2.7->matplotlib) (1.
16.0)
Installing collected packages: python-dateutil, pillow, packaging, kiwisolver, fonttools, cycler, contourpy, matplotl
ib
  WARNING: The scripts fonttools, pyftmerge, pyftsubset and ttx are installed in '/home/ubuntu/.local/bin' which is n
ot on PATH.
  Consider adding this directory to PATH or, if you prefer to suppress this warning, use --no-warn-script-location.
Successfully installed contourpy-1.2.0 cycler-0.12.1 fonttools-4.44.3 kiwisolver-1.4.5 matplotlib-3.8.1 packaging-23.
2 pillow-10.1.0 python-dateutil-2.8.2
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ pip3 install tk
Defaulting to user installation because normal site-packages is not writeable
Collecting tk
  Downloading tk-0.1.0-py3-none-any.whl (3.9 kB)
Installing collected packages: tk
Successfully installed tk-0.1.0
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ sudo apt install -q python-tk
Reading package lists...
Building dependency tree...
Reading state information...
The following additional packages will be installed:
  blt libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib libtk8.6 libxft2 libxrender1 libxss1 python2
  python2-minimal python2.7 python2.7-minimal tk8.6-blt2.5 x11-common
Suggested packages:
  blt-demo tk8.6 tix python-tk-dbg python2-doc python2.7-doc binfmt-support
The following NEW packages will be installed:
  blt libpython2-stdlib libpython2.7-minimal libpython2.7-stdlib libtk8.6 libxft2 libxrender1 libxss1 python-tk
  python2 python2-minimal python2.7 python2.7-minimal tk8.6-blt2.5 x11-common
0 upgraded, 15 newly installed, 0 to remove and 9 not upgraded.
Need to get 5555 kB of archives.
After this operation, 21.4 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libpython2.7-minimal amd64 2.7.18-13u
buntu1.1 [347 kB]
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 python2.7-minimal amd64 2.7.18-13ubun
tu1.1 [1394 kB]
```

HPC_HS2023_ex08_v...* — Upload / Share — Pro Plus

that your Ubuntu installation works fine.

- Now we are going to compile and run the serial Poisson solver code on the VM. Clone the course git repository, https://github.com/jbucko/hpc_esc_401 in your VM, and move to the `exercise_session_09/poisson_solver_serial` folder. In order to compile the code however, we need:
  - A compiler: install the `g++` compiler in your VM. Add text here
  - Since we use a Makefile to compile the code, we need the `make` tool: install it too

  Then you should be able to compile the code and then run it.

- To visualize your output files, you can use the Python script `plot.py` (in the `output` folder).
  - To run the `plot.py` script, you will need to install the required packages (`numpy` etc.). First install pip with `sudo apt install -q python3-pip`, then install the needed packages with `pip3 install <package_name>`. Moreover, you will have to install `Tkinter`, which is the Python interface to the Tk GUI toolkit shipped with Python: `pip3 install tk`, then `sudo apt install -q python-tk`. (if installation of any python package fails with error `Failed building wheel...`, consider updating wheel using `python3 -m pip install --upgrade pip setuptools wheel`. Also, `plot.py` is written with `python3`, so if running command `python -V` prints "Python 2.x.xx" then you should use command `python3` instead.

- Create and save a snapshot of your instance on the Science Cloud.

```
import matplotlib
matplotlib.use("TkAgg")
```

Commit: Provide the step-by-step solution on how you managed to compile and run Poisson solver and create the snapshot on the Science cloud.

DEU 14:34 17.11.2023

Wortwolke

Mitmachen
- Wikiartikel anlegen
- Howto anlegen
- Wiki-Referenz
- Wiki-Syntax
- Baustellen
- Artikelideen
- Ungetestete Artikel
- Ausbaufähige Artikel
- Fehlerhafte Artikel
- Rund ums Wiki

Konfiguration
- Backlinks anzeigen
- Exportieren
  - Metadaten
  - Rohformat
  - HTML

diesen Artikel für eine Ubuntu-Version, welche aktuell unterstützt wird. Dazu sind die Hinweise zum Testen von Artikeln zu beachten.

Zum Verständnis dieses Artikels sind folgende Seiten hilfreich:

1. Installation von Programmen
2. Ein Terminal öffnen

Inhaltsverzeichnis

FEH

Feh ist ein vielseitiger, sehr schneller Bildbetrachter und wird über die Kommandozeile gesteuert. Das Programm ist insbesondere bei Nutzern eines alternativen Fenstermanagers wie z.B. Openbox beliebt, um ein Hintergrundbild zu setzen. Das Programm bietet unter anderem folgende Funktionen:

- Bildanzeige (lokal und entfernt)
- Diashow
- Sortierung
- Thumbnail-Browsen
- dynamisches Zoomen
- Maus-Wheel/Tastaturunterstützung

Bild mit Transparenz

Installation

Folgendes Paket muss installiert [1] werden:

- **feh** (*universe*)

Befehl zum Installieren der Pakete:

```
sudo apt-get install feh
```

Oder mit apturl installieren, Link: apt://feh

Benutzung

ubuntu@chiara: ~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output

```
chiara@LAPTOP-DPG4081K:~$ ssh -i .ssh/id_rsa ubuntu@172.23.53.198
Welcome to Ubuntu 22.04.3 LTS (GNU/Linux 5.15.0-88-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri Nov 17 18:44:43 UTC 2023

  System load:  0.0               Processes:             147
  Usage of /:   2.6% of 96.73GB   Users logged in:       0
  Memory usage: 1%                IPv4 address for ens3: 172.23.53.198
  Swap usage:   0%


Expanded Security Maintenance for Applications is not enabled.

8 updates can be applied immediately.
3 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable

4 additional security updates can be applied with ESM Apps.
Learn more about enabling ESM Apps service at https://ubuntu.com/esm


Last login: Fri Nov 17 13:16:05 2023 from 89.206.112.11
ubuntu@chiara:~$ ls
hpc_esc_401
ubuntu@chiara:~$ cd hpc_esc_401/
ubuntu@chiara:~/hpc_esc_401$ ls
README.md           exercise_session_03  exercise_session_05  exercise_session_07  exercise_session_09
exercise_session_02  exercise_session_04  exercise_session_06  exercise_session_08
ubuntu@chiara:~/hpc_esc_401$ cd exercise_session_09
ubuntu@chiara:~/hpc_esc_401/exercise_session_09$ ls
poisson_solver_serial
ubuntu@chiara:~/hpc_esc_401/exercise_session_09$ cd poisson_solver_serial/
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial$ ls
Makefile  init.h  io.cpp  io.o        jacobi.h  main      output
init.cpp  init.o  io.h    jacobi.cpp  jacobi.o  main.cpp  params.txt
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial$ cd output/
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ python -V
Python 3.10.12
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ python plot.py
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ ls
output_0001000.csv  output_0010000.csv  output_0019000.csv  output_0028000.csv  output_0037000.csv
output_0002000.csv  output_0011000.csv  output_0020000.csv  output_0029000.csv  output_0038000.csv
output_0003000.csv  output_0012000.csv  output_0021000.csv  output_0030000.csv  output_0039000.csv
output_0004000.csv  output_0013000.csv  output_0022000.csv  output_0031000.csv  output_0040000.csv
output_0005000.csv  output_0014000.csv  output_0023000.csv  output_0032000.csv  output_0041000.csv
output_0006000.csv  output_0015000.csv  output_0024000.csv  output_0033000.csv  output_source.csv
output_0007000.csv  output_0016000.csv  output_0025000.csv  output_0034000.csv  plot.py
output_0008000.csv  output_0017000.csv  output_0026000.csv  output_0035000.csv  test.png
output_0009000.csv  output_0018000.csv  output_0027000.csv  output_0036000.csv
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ test.png
```

But it's also possible to do the hands-on labs on other systems or even on your own laptops.

You will need a compiler for C, C++ or Fortran that supports OpenMP, and an MPI library. Only for MPI it is in principle possible to do the exercises also in python and using mpi4py.

If you have access to a system that has the required software installed, feel free to use that.

Local installation

Note that we will not provide support for local installations.

Linux

Linux users need to install the GNU compilers and a couple of MPI packages, e.g. for Ubuntu:

```
user@ubuntu$ sudo apt-get install gcc g++ gfortran
user@ubuntu$ sudo apt-get install openmpi-bin
user@ubuntu$ sudo apt-get install libopenmpi-dev
```

Mac

Mac users need to install compilers from the Xcode developer package. It is easiest to install MPI using the Homebrew package manager - here are Instructions on how to install Xcode and Homebrew.
Now install OpenMPI:

```
user@mac$ brew install open-mpi
```

Windows

One solution is to install the Linux Subsystem for Windows (https://docs.microsoft.com/en-us/windows/wsl/install-win10), a recent feature of Windows 10. This will allow you to run Linux terminal applications on Windows. Alternatively, you can install a Linux virtual machine (e.g. Ubuntu) and follow the Linux installation instructions above.

Another option would be to install MPI natively on Windows using the Intel Parallel Studio compilers and the Intel MPI library (free access for students https://software.intel.com/en-us/qualify-for-free-software/student).

```
ubuntu@chiara:~/hpc_esc_401/exercise_session_09/poisson_solver_serial/output$ cd ../../..
ubuntu@chiara:~/hpc_esc_401$ ls
README.md           exercise_session_03  exercise_session_05  exercise_session_07  exercise_session_09
exercise_session_02  exercise_session_04  exercise_session_06  exercise_session_08
ubuntu@chiara:~/hpc_esc_401$ cd exercise_session_07
ubuntu@chiara:~/hpc_esc_401/exercise_session_07$ ls
poisson_MPI  ring_and_deadlocks  solutions
ubuntu@chiara:~/hpc_esc_401/exercise_session_07$ cd poisson_MPI/
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/poisson_MPI$ ls
Makefile  init.h  io.h        jacobi.h  mpi_module.cpp  output      simple_mpi.py
init.cpp  io.cpp  jacobi.cpp  main.cpp  mpi_module.h    params.txt
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/poisson_MPI$ sudo apt-get install openmpi-bin
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  ibverbs-providers libevent-pthreads-2.1-7 libfabric1 libhwloc-plugins libhwloc15 libibverbs1 libnl-route-3-200
  libopenmpi3 libpciaccess0 libpmix2 libpsm-infinipath1 libpsm2-2 librdmacm1 libucx0 libxnvctrl0 ocl-icd-libopencl1
  openmpi-common
Suggested packages:
  libhwloc-contrib-plugins opencl-icd gfortran | fortran-compiler
The following NEW packages will be installed:
  ibverbs-providers libevent-pthreads-2.1-7 libfabric1 libhwloc-plugins libhwloc15 libibverbs1 libnl-route-3-200
  libopenmpi3 libpciaccess0 libpmix2 libpsm-infinipath1 libpsm2-2 librdmacm1 libucx0 libxnvctrl0 ocl-icd-libopencl1
  openmpi-bin openmpi-common
0 upgraded, 18 newly installed, 0 to remove and 9 not upgraded.
Need to get 6190 kB of archives.
After this operation, 20.8 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libnl-route-3-200 amd64 3.5.0-0.1 [180 kB]
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libibverbs1 amd64 39.0-1 [69.3 kB]
Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 ibverbs-providers amd64 39.0-1 [341 kB]
Get:4 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libevent-pthreads-2.1-7 amd64 2.1.12-stable-1buil
d3 [7642 B]
Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 libpsm-infinipath1 amd64 3.3+20.604758e7-6.1
[170 kB]
Get:6 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 libpsm2-2 amd64 11.2.185-1 [182 kB]
Get:7 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 librdmacm1 amd64 39.0-1 [71.2 kB]
Get:8 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 libfabric1 amd64 1.11.0-3 [558 kB]
Get:9 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libhwloc15 amd64 2.7.0-2ubuntu1 [159
kB]
Get:10 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libpciaccess0 amd64 0.16-3 [19.1 kB]
Get:11 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 libxnvctrl0 amd64 510.47.03-0ubuntu1 [11.5 kB]
Get:12 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 ocl-icd-libopencl1 amd64 2.2.14-3 [39.1 kB]
Get:13 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/universe amd64 libhwloc-plugins amd64 2.7.0-2ubuntu
1 [15.6 kB]
Get:14 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 libpmix2 amd64 4.1.2-2ubuntu1 [604 kB]
Get:15 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 libucx0 amd64 1.12.1~rc2-1 [891 kB]
Get:16 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 libopenmpi3 amd64 4.1.2-2ubuntu1 [2594 kB]
Get:17 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 openmpi-common all 4.1.2-2ubuntu1 [162 kB]
Get:18 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/universe amd64 openmpi-bin amd64 4.1.2-2ubuntu1 [116 kB]
Fetched 6190 kB in 2s (2980 kB/s)
Selecting previously unselected package libnl-route-3-200:amd64.
(Reading database ... 106694 files and directories currently installed.)
```

But it's also possible to do the hands-on labs on other systems or even on your own laptops.

You will need a compiler for C, C++ or Fortran that supports OpenMP, and an MPI library. Only for MPI it is in principle possible to do the exercises also in python and using mpi4py.

If you have access to a system that has the required software installed, feel free to use that.

Local installation

Note that we will not provide support for local installations.

Linux

Linux users need to install the GNU compilers and a couple of MPI packages, e.g. for Ubuntu:

```
user@ubuntu$ sudo apt-get install gcc g++ gfortran
user@ubuntu$ sudo apt-get install openmpi-bin
user@ubuntu$ sudo apt-get install libopenmpi-dev
```

Mac

Mac users need to install compilers from the Xcode developer package. It is easiest to install MPI using the Homebrew package manager - here are Instructions on how to install Xcode and Homebrew.
Now install OpenMPI:

```
user@mac$ brew install open-mpi
```

Windows

One solution is to install the Linux Subsystem for Windows (https://docs.microsoft.com/en-us/windows/wsl/install-win10), a recent feature of Windows 10. This will allow you to run Linux terminal applications on Windows. Alternatively, you can install a Linux virtual machine (e.g. Ubuntu) and follow the Linux installation instructions above.

Another option would be to install MPI natively on Windows using the Intel Parallel Studio compilers and the Intel MPI library (free access for students https://software.intel.com/en-us/qualify-for-free-software/student).

```
ubuntu@chiara: ~/hpc_esc_401/exercise_session_07/poisson_MPI
Setting up librdmacm1:amd64 (39.0-1) ...
Setting up libucx0:amd64 (1.12.1~rc2-1) ...
Setting up libpmix2:amd64 (4.1.2-2ubuntu1) ...
Setting up libfabric1:amd64 (1.11.0-3) ...
Setting up libopenmpi3:amd64 (4.1.2-2ubuntu1) ...
Setting up openmpi-bin (4.1.2-2ubuntu1) ...
update-alternatives: using /usr/bin/mpirun.openmpi to provide /usr/bin/mpirun (mpirun) in auto mode
update-alternatives: using /usr/bin/mpicc.openmpi to provide /usr/bin/mpicc (mpi) in auto mode
Processing triggers for man-db (2.10.2-1) ...
Processing triggers for libc-bin (2.35-0ubuntu3.4) ...
Scanning processes...
Scanning linux images...

Running kernel seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/poisson_MPI$ sudo apt-get install libopenmpi-dev
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following additional packages will be installed:
  autoconf automake autotools-dev gfortran gfortran-11 libcaf-openmpi-3 libcoarrays-dev libcoarrays-openmpi-dev
  libevent-2.1-7 libevent-dev libevent-extra-2.1-7 libevent-openssl-2.1-7 libgfortran-11-dev libgfortran5
  libhwloc-dev libibverbs-dev libjs-jquery-ui libltdl-dev libltdl7 libnl-3-dev libnl-route-3-dev libnuma-dev
  libpmix-dev libtool m4
Suggested packages:
  autoconf-archive gnu-standards autoconf-doc gettext gfortran-multilib gfortran-doc gfortran-11-multilib
  gfortran-11-doc libjs-jquery-ui-docs libtool-doc openmpi-doc gcj-jdk m4-doc
The following NEW packages will be installed:
  autoconf automake autotools-dev gfortran gfortran-11 libcaf-openmpi-3 libcoarrays-dev libcoarrays-openmpi-dev
  libevent-2.1-7 libevent-dev libevent-extra-2.1-7 libevent-openssl-2.1-7 libgfortran-11-dev libgfortran5
  libhwloc-dev libibverbs-dev libjs-jquery-ui libltdl-dev libltdl7 libnl-3-dev libnl-route-3-dev libnuma-dev
  libopenmpi-dev libpmix-dev libtool m4
0 upgraded, 26 newly installed, 0 to remove and 9 not upgraded.
Need to get 18.6 MB of archives.
After this operation, 77.7 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 m4 amd64 1.4.18-5ubuntu2 [199 kB]
Get:2 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 autoconf all 2.71-2 [338 kB]
Get:3 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 autotools-dev all 20220109.1 [44.9 kB]
Get:4 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 automake all 1:1.16.5-1.3 [558 kB]
Get:5 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgfortran5 amd64 12.3.0-1ubuntu1~22.04
[879 kB]
Get:6 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 libgfortran-11-dev amd64 11.4.0-1ubuntu1~
22.04 [842 kB]
Get:7 http://nova.clouds.archive.ubuntu.com/ubuntu jammy-updates/main amd64 gfortran-11 amd64 11.4.0-1ubuntu1~22.04 [
11.2 MB]
Get:8 http://nova.clouds.archive.ubuntu.com/ubuntu jammy/main amd64 gfortran amd64 4:11.2.0-1ubuntu1 [1182 B]
```

– WORKDIR: Specifies the directory from where you want to run any following command.

– CMD: Contains the run-time commands of your container, basically launching the code in your case.

To build your image, add your Dockerfile (named `Dockerfile`) to the root of the Poisson solver directory and execute the command "`docker build --tag poisson .`" and make sure that your image is correctly build with "`docker images`".

Run the container on your local machine with `docker run` to test your setup. You may notice that the Output directory in your computer does not contain any new file. This is because files were written in the container. In order to access the output files, you need to mount a host directory into the container. To do that, use the `-v` or `--mount` flags and specify the host directory you want to link to the container one.

- Now that your Docker image is setup, you can push it to your Docker Hub repository. First setup a Docker Hub account, then create a new repository named `poisson` on the web page. To push the docker image, you first have to rename your image like `<Docker ID>/<Repository Name>:<tag>`, to do so you can run the command `docker tag poisson:latest <Docker ID>/poisson:latest`. Finally push the image with `docker push <Docker ID>/poisson:latest`

- Pull the Docker image from the Docker Hub registry and run it on your VM. Install instructions https://docs.docker.com/engine/install/ubuntu/, post install instructions (Manage Docker as a non-root user) https://docs.docker.com/engine/install/linux-postinstall/

ubuntu@chiara: ~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal

```
CFLAGS=-O3 -ffast-math -mavx2

CC=mpiCC

all: mpi_module.o init.o io.o jacobi.o main.o main
init.o: init.cpp init.h
	${CC} ${CFLAGS} -c -o init.o init.cpp
io.o: io.cpp io.h
	${CC} ${CFLAGS} -c -o io.o io.cpp
jacobi.o: jacobi.cpp jacobi.h
	${CC} ${CFLAGS} -c -o jacobi.o jacobi.cpp
mpi_module.o: mpi_module.cpp mpi_module.h
	${CC} ${CFLAGS} -c -o mpi_module.o mpi_module.cpp
main.o: main.cpp
	${CC} ${CFLAGS} -c -o main.o main.cpp
main: mpi_module.o jacobi.o io.o init.o main.o
	${CC} -o main main.o init.o io.o jacobi.o mpi_module.o

clean:
	rm -f main main.o init.o io.o jacobi.o mpi_module.o
```

3,8 All

DEU 20:02 17.11.2023

– WORKDIR: Specifies the directory from where you want to run any following command.

– CMD: Contains the run-time commands of your container, basically launching the code in your case.

To build your image, add your Dockerfile (named `Dockerfile`) to the root of the Poisson solver directory and execute the command "`docker build --tag poisson .`" and make sure that your image is correctly build with "`docker images`".

Run the container on your local machine with `docker run` to test your setup. You may notice that the Output directory in your computer does not contain any new file. This is because files were written in the container. In order to access the output files, you need to mount a host directory into the container. To do that, use the `-v` or `--mount` flags and specify the host directory you want to link to the container one.

- Now that your Docker image is setup, you can push it to your Docker Hub repository. First setup a Docker Hub account, then create a new repository named `poisson` on the web page. To push the docker image, you first have to rename your image like `<Docker ID>/<Repository Name>:<tag>`, to do so you can run the command `docker tag poisson:latest <Docker ID>/poisson:latest`. Finally push the image with `docker push <Docker ID>/poisson:latest`

- Pull the Docker image from the Docker Hub registry and run it on your VM. Install instructions https://docs.docker.com/engine/install/ubuntu/, post install instructions (Manage Docker as a non-root user) https://docs.docker.com/engine/install/linux-postinstall/

```
ubuntu@chiara: ~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal
--------------------------------------------------------------------------
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/poisson_MPI$ cd ../..
ubuntu@chiara:~/hpc_esc_401$ ls
README.md           exercise_session_03  exercise_session_05  exercise_session_07  exercise_session_09
exercise_session_02  exercise_session_04  exercise_session_06  exercise_session_08
ubuntu@chiara:~/hpc_esc_401$ git pull
Already up to date.
ubuntu@chiara:~/hpc_esc_401$ cd
.git/               exercise_session_03/ exercise_session_05/ exercise_session_07/ exercise_session_09/
exercise_session_02/ exercise_session_04/ exercise_session_06/ exercise_session_08/
ubuntu@chiara:~/hpc_esc_401$ cd exercise_session_07/
poisson_MPI/       ring_and_deadlocks/ solutions/
ubuntu@chiara:~/hpc_esc_401$ cd exercise_session_07/solutions/
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions$ ls
01_ring_and_deadlocks  02_MPI_poisson_solver  README.md
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions$ cd 02_MPI_poisson_solver/
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver$ ls
README.md  poisson_MPI_full  poisson_MPI_solution_minimal
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver$ cd poisson_MPI_solution_minimal/
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal$ ls
Makefile  init.h  io.h        jacobi.h  mpi_module.cpp  output       plot_MPI.py          simple_mpi.py
init.cpp  io.cpp  jacobi.cpp  main.cpp  mpi_module.h    params.txt  run_MPI_minimal.job
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal$ vim Mak
efile
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal$ make al
l
mpiCC -O3 -ffast-math -mavx2 -c -o mpi_module.o mpi_module.cpp
mpiCC -O3 -ffast-math -mavx2 -c -o init.o init.cpp
mpiCC -O3 -ffast-math -mavx2 -c -o io.o io.cpp
mpiCC -O3 -ffast-math -mavx2 -c -o jacobi.o jacobi.cpp
mpiCC -O3 -ffast-math -mavx2 -c -o main.o main.cpp
mpiCC -o main main.o init.o io.o jacobi.o mpi_module.o
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal$ ls
Makefile  init.o  io.o        jacobi.o  main.o          mpi_module.o  plot_MPI.py
init.cpp  io.cpp  jacobi.cpp  main      mpi_module.cpp  output        run_MPI_minimal.job
init.h    io.h    jacobi.h    main.cpp  mpi_module.h    params.txt    simple_mpi.py
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal$
```

HPC_HS2023_ex08_v...*

Upload / Share

– WORKDIR: Specifies the directory from where you want
command.

– CMD: Contains the run-time commands of your container,
code in your case.

To build your image, add your Dockerfile (named `Dockerfile`) t
son solver directory and execute the command "`docker build -`
make sure that your image is correctly build with "`docker imag`

Run the container on your local machine with `docker run` to tes
notice that the Output directory in your computer does not cont
is because files were written in the container. In order to acces
need to mount a host directory into the container. To do that,
flags and specify the host directory you want to link to the cont

- Now that your Docker image is setup, you can push it to your D
First setup a Docker Hub account, then create a new repositor
the web page. To push the docker image, you first have to re
`<Docker ID>/<Repository Name>:<tag>`, to do so you can run
`tag poisson:latest <Docker ID>/poisson:latest`. Finally
`docker push <Docker ID>/poisson:latest`

- Pull the Docker image from the Docker Hub registry and ru
stall instructions https://docs.docker.com/engine/install/ubuntu
tions (Manage Docker as a non-root user) https://docs.docke
linux-postinstall/

ubuntu@chiara: ~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal

```
Step 19315, Diff=1.03955e-08
Step 19316, Diff=1.03875e-08
Step 19317, Diff=1.03795e-08
Step 19318, Diff=1.03714e-08
Step 19319, Diff=1.03635e-08
Step 19320, Diff=1.03555e-08
Step 19321, Diff=1.03475e-08
Step 19322, Diff=1.03395e-08
Step 19323, Diff=1.03315e-08
Step 19324, Diff=1.03236e-08
Step 19325, Diff=1.03156e-08
Step 19326, Diff=1.03077e-08
Step 19327, Diff=1.02997e-08
Step 19328, Diff=1.02918e-08
Step 19329, Diff=1.02838e-08
Step 19330, Diff=1.02759e-08
Step 19331, Diff=1.0268e-08
Step 19332, Diff=1.02601e-08
Step 19333, Diff=1.02522e-08
Step 19334, Diff=1.02442e-08
Step 19335, Diff=1.02364e-08
Step 19336, Diff=1.02285e-08
Step 19337, Diff=1.02206e-08
Step 19338, Diff=1.02127e-08
Step 19339, Diff=1.02048e-08
Step 19340, Diff=1.0197e-08
Step 19341, Diff=1.01891e-08
Step 19342, Diff=1.01812e-08
Step 19343, Diff=1.01734e-08
Step 19344, Diff=1.01655e-08
Step 19345, Diff=1.01577e-08
Step 19346, Diff=1.01499e-08
Step 19347, Diff=1.0142e-08
Step 19348, Diff=1.01342e-08
Step 19349, Diff=1.01264e-08
Step 19350, Diff=1.01186e-08
Step 19351, Diff=1.01108e-08
Step 19352, Diff=1.0103e-08
Step 19353, Diff=1.00952e-08
Step 19354, Diff=1.00874e-08
Step 19355, Diff=1.00797e-08
Step 19356, Diff=1.00719e-08
Step 19357, Diff=1.00641e-08
Step 19358, Diff=1.00564e-08
Step 19359, Diff=1.00486e-08
Step 19360, Diff=1.00409e-08
Step 19361, Diff=1.00331e-08
Step 19362, Diff=1.00254e-08
Step 19363, Diff=1.00177e-08
Step 19364, Diff=1.00099e-08
ubuntu@chiara:~/hpc_esc_401/exercise_session_07/solutions/02_MPI_poisson_solver/poisson_MPI_solution_minimal$ mpirun -np 4 main
```

DEU 20:04 17.11.2023