Organizing Collaboration in Open Source Teams

Gregor von Laszewski, Fidel Leal, Shannon Kerr, Erin Seliger, Cooper Young, Agness Lungu

Abstract

For more information, please contact: laszewski@gmail.com

Contents

Note: Do not edit this document, only edit the documents in the chapters directory

1 Organizing Collaborative Research Teams

1.1 Communication (Gregor)

To organize a research team, it is of utmost importance to establish simple communication pathways. This includes ways to conduct text chat video conferencing, code sharing, editing, and task management.

In the following sections, we will list some useful tools that can be used by the research team while keeping the learning curve to a minimum. In general, it is good to ask the participants if they already use particular tools in a category and if all in the team use them to adopt them. However, this may limit the general availability in case the team grows into the open-source community. Hence, it is important to consider licensing issues and if possible adopt free tools for the research team.

1.1.1 Video Conferencing (Gregor)

Videoconferencing has undoubtfully become a major component of research teams. It allows face meetings without the need for traveling. Thus you can spend the time saved on travel. Also, it allows researchers to continue if unexpected events take place that does not allow in-person meetings such as the recent COVID epidemic.

There are many confenencing tools available that can be used. You may even use multiple dependent on the particular meeting or preferences by the subgroup. In order to keep things simple it is however recommended to jsut use one tool.

1.1.1.1 Google Meet (Shannon, Gregor) Google Meet is an online platform that allows groups of people to host video and audio conference calls. It has evolved from google hangout.

- What are good features? (Shannon)

  Some good features of Google meet include that it has a button for closed captioning or subtitles in the case that you're having trouble hearing the speaker. It is also compatible across devices so it is very easy and accessible for all group members. The sound quality is very good.

- What are not so good features? (Gregor)

  1. Goggle meet does not provide an easy way to have other take control of a remote desktop. However, it is possible to use Google Remote Desktop for it.
     - https://remotedesktop.google.com/?pli=1
  2. Google meet does not have the ability to share each others desktops at tehe same time. This feature was available in hangout but is no longer available as far as we can tell.

- Why do we use google meet and not zoom? (Gregor)

  Google offers many services that are useful for collaboration. This includes Google drive, docs, presentations, mail, calendar, and groups. features such as google drive. As they can be accessed through a single account, it is obvious that Google meet provides a valuable set of services to any research team.

1.1.1.2 Zoom (Fidel, Gregor) Zoom is a cloud-based communications platform that provides support for one-on-one, group meetings, and webinars and has recently become one of the most popular video-conferencing tools.

- What are good features? (Fidel, Gregor)

  Some of Zoom's most popular features include live chat, screen sharing, a whiteboard, and virtual reactions for meeting participants. Additionally, the ability to record meetings to the cloud or personal devices, create breakout rooms and seamlessly move between them, remotely control a participant's screen and the functionality to share files in the meeting make Zoom into an attractive collaborative tool. Zoom allows users to join any session through an established meeting URL. Participants do not need to be signed in or even have a Zoom account. Additionally, people joining from places with limited Internet access can call into the meeting's audio channel using dedicated telephone numbers.

- What are not so good features? (Gregor, Fidel)

  Free account holders can host unlimited one-on-one meetings (meeting durations up to 24 hrs). In contrast to Google meet, there is a duration limit of 40 mins for meetings with three or more participants, making it difficult even for small teams to take advantage of the platform's features. Additionally, we observed that video quality can be unstable, and the overall platform performance can quickly deteriorate over limited bandwidth connections. In such cases, we recommend switching the camera from the participant that has issues. Furthermore, if you do at the same time a lot of calculations on your machine it may affect the quality of the call. This applied to older machines and should allow you to give a beautiful argument to get a new computer. In some cases, you may have a second computer and can use one for sharing your session, while the other one is used for sound, or you use a cell phone for the later

- How can someone take control of a remote desktop? (Fidel)

  To take control of a remote desktop, the remote user must activate screen sharing. Once the screen sharing is activated, we need to click the `View Options` dropdown menu (usually at the top of the screen) and click on `Request remote control`. The remote user will then get a prompt to approve the remote control request.

  Institutional accounts may have the remote control functionality disabled by their account administrator. For further details, refer to the Zoom support pages.

1.1.2 Realtime nad Offline Text Messaging (Gregor)

In many research projects participants may be in different timezones or have schedules that do not provide ovelapping times for video-conferencing. For this reason it is important to support a chat like feature, that allows the researchers to catch up with activities that took place they were not avaialable for. Tools such as e-mail have filled this demand for quite some time. Recently addionlal tools such as Slack have appearde that enhance the e-mail activity while also allwing real time text messaging,

1.1.2.1 Slack (Gregor, Cooper) Slack is a communication software that is used for groups to send and receive text messages and access services that can be added to slack.

- What are good features? (Gregor)

Slack offers a GUI that is liked by many while focussing on a stream of messages. It can be used on any device. It is easy to send photos, which may be useful in case the device you need to discuss is not on the internet but you need to share the content for example of its screen.

- What are not so good features? (Gregor)

Slack is stream-based and does not provide a good mechanism for organizing messages. The thread feature is far inferior to that of even a simple e-mail client. If one is involved in many slack workspaces, it becomes difficult to manage them. Most importantly Slack comes in its free version with only a limited number of free messages. This means you will have to pay once you exceed the limit. Thus even the integration of useful services such as GitHub notifications is not recommended as you will too quickly exceed the limits. A posting policy needs to be established. Those that are not using slack frequently may be out of touch quickly.

Although there is an unread threads feature, it may be filled with messages if you do not use slack daily just to keep up.

To support separation of topics it is advisable to create a number of channels such as “general” or a chanel for a particular topic. However it is also important to limit the number of channels so it does not become too confusing

☐ TODO: Open, can slack configured to send e-mail?

1.2 Creating Text (Gregor)

☐ TODO: Gregor, Introduction

1.2.1 Dictation (Windows 10 try out, Agness)

Sometimes it is conveneinet to directly dictate the text for a manual or tutorial into an editor. On MacOS and Windows you will find useful tools for this. A Voice to text recorder may also help you in case you have a recoded video of yourself to generate transcripts. Disadvantag for a lot of non Native english speakers is that the accuraccy may be limited and that not using them leads to unacceptable results. Some of them can be trained. Try it out and let us know if they work for you and if you are satisfied with their accuracy. Examples include:

- Apple Dictation (free for Apple devices) You can directly dictate into various applicatipns that help you improving your text such as Grammarly and MS Word if you have installed them

*cWindows 10 Voice Recognition (free for Windows users). You can directly record into MS Word so you get a free grammar checcker

> Agnes: I tried windows voice recognition and it did not work on my computer * Google Docs offers buidl in voice typing for dictating Google voice has a very good recognition success rate it is fast but has at times difficultis recognizing names. In case it makes mistakes, you will have to clean it up after dictation.

- ☐ TODO: Agnes, can we dictate into MS word? ... leran how to write in third person

1.2.2 Grammar Checkers (Gregor, review Erin)

When developing content for tutorials and documentation it is important to check their correctness with a grammar checcker. We have made the best experience with Grammarly followed by MS word. The best way to use them is to copy and paste small sections into them from your document and then check them. After you are satisfied copy the contents back to the original one, while overwriting the old text.

- Grammarly
  - Pro: works well, is available for free, the free version is good enough
  - Cons: not everything is done correct. In some text it shows false errors, but it’s still very good
- Word
  - Pros: those familiar with word will know it
  - Cons: sometimes Grammarly performs better, Copying text back and forth can introduce errors when it comes to using quotes and other symbols. Thus you must check all symbols after you copy it into a markdown document.

Recommendation: use Grammarly

1.2.3 Editors

You will need likely multiple two editors as part of your research activities. This is motivated by the fact that we do lots of development on yourl local machine, but also do remote development via terminal access

to a remote computer that does not have a GUI. In case you only want t learn one editor to do all of this, just use emacs. We have listed bellow some editors and youmay want to chose

- emacs vs vi/vim for terminal editing
- pycharm vs MS code for fancyful python code development

There is a third option that we will use and that is jupyter which allows us to interactively develp python code. Jupyter is important as it is used by many data scientist due to its ad hoc interactive mode while programming. However, also pycharm and vscode prode options to view and run jupyter notebooks. However, not everything that works in jupyter is working on these platforms.

Recommendation: Use emacs and pyCharm

Terminal Editors

- emacs
  - Pro: terminal, established, very good markdown support, block format with ESC-q, keyboard shortcuts also used in bash and ther shells, has a python and markdown mode
  - Cons: some users have a hard time remembering keyboard shortcuts, the editor may get stuck in some unkown mode that you activated by accident. ALL this is easy to fix by remembering CRTL-X-s (save) and CTRL-G (get out of a strange mode)
- vi
  - use vim instead, vi is availabel on all Linux machines, but has rather archaic editing controls.
- vim
  - Pros: like vi, but with cursor control
  - Cons: often awkward to use

IDEs: * Pycharm * Pro: best Python editor * Cons: needs lots of resources, steep learning curve * Code (MS) * Pro: ofetn used on Raspberry * Cons: Pycharm seems to have more features from the start

1.3 Documentation Development with Markdown ()

1.3.1 Markdown ()

☐ TODO: Gregor, Open, point to book

- What is markdown?
- Why do we use it?
- What a re good editors for markdon?
- Pointer to Gregor book
- Collaborative editing with HackMD.io

1.4 ToDo Lists

☐ TODO: Gregor, improve TODO section

It is important to communicate quickly some tasks in the document that we write as a team. In order to do this we wuse the keyword TODO , followed usually by an explanation if needed. As a TODO can be hopefully resolved quickly it shoudl be able yp complete them in 1-2 hours. Any TODO that may take longer we also add to our GitHub project in order for it to be recorded and if we identify or delays in its execution we can assign additional team members to help on this tasks.

Once a team member has identified a TODO item, the team member can simple put his name behwind it, as well as the data and time so others know you work on it. You can aslo communicate on slack about the task you do if you run into issues or have questions.

All: if you see a TODO , and want to do it (e.g. have 1-2 hours time, put your name to it so others know you will work on it. Do not assign a TODO to you if you do not have time and will do it in a month from now,

Research tasks need to be done immediately. HOwever we will also assign some longer term tasks to you so you can work ahead and in parallel, if your task is not done it will be assigned to some one else to mitigate that the time delay does not block the project.)

All: add tasks in github so we can assign todos and monitor progress

Gregor: Describe In detail how this is done

1.5 Git and GitHub

Git is a distributed version control system to support working on project in teams while allowing different team members to contribute and to currate the contribution through reviews.

GitHub is a service offered for free with the limitation that the repositories shoudl not be larger than 1GB and the individual files must be smaller then 200MB. Github is very popular for OpenSource projects and through its free offering allows community building around OpenSource Projects.

1.5.1 Git from the commandline

Git can easily be installed on all platforms including

- macOS: You will need to install Xcode which includes not only git but user Linux programs such as Makefile
- Ubuntu: You can install it via `apt install git`
- Windows: You can install it via *Git Bash* which is distributed from https://git-scm.com/downloads. When installing read carefully the available options. We recommend you install a desktop shortcut.

1.6 Git form IDEs

Pycharm is one of the best editors for Python. It does provide build in support to interact with GitHub. This document here and I already went to some of your contributions and made improvements or two then OK

1.7 GitHub from a GUI

Some may fancy using a Graphical user interface to interact with GitHub. However, in many cases, the terminal access is simpler. However, if you like to browse the repositories and see the commit tree, these GUI interfaces are useful. Several such interfaces are available at:

- https://git-scm.com/downloads/guis

1.7.1 GitHub Commands

- What are the most important commands?
  - pull
    - ☐ TODO: Shannon, git pull
  - git add FILENAME
    - ☐ TODO: Agnes, git add
  - commit -a
    - ☐ TODO: Agnes, git commit
  - push
    - ☐ TODO: Erin, git push
  - checkout branchanme
    - ☐ TODO: Shannon, git branch
- What is a branch and how do we use it form the commandline?
  - ☐ TODO: Shannon, git branch

- What if you committed something you did not want to and pushed it?

That is realy bad and you need to contact Gregor THis will take hours to fix, so be careful. So make sure you code does not contain passwords in plaintext.

Also do never ever use the git command `git add .` as that adds all files and you could have files that you do not want to commit. instead **always** use `git add FILENAME`, where FILENAME is the file you like to add

1.7.2 Task Management

- Our tasks in Github
  ☐ TODO: Gregor, generalize
- Our issues in GitHub
  ☐ TODO: Gregor generalize

1.7.3 Code Management

Our cod is managed as opens ource code in github.

- Our code in GitHub

  To check out use

  ```
  git clone git@github.com:cloudmesh/cloudmesh-mpi.git
  ```

  or

  ```
  git clone https://github.com/cloudmesh/cloudmesh-mpi.git
  ```

1.7.4 Github Actions

We have not yet used them

☐ TODO: Gregor, provide description what they are

2 Appendix

2.1 Hardware of current students

- Fidel Leal,
  - Equipment
    - MacBook Pro 2015, 16GB RAM i7, SSD 512GB
    - PC, 64-bit, 8GB RAM, i5, SSD <240GB, speed>
      - Windows 10 Education
    - Editor: Pycharm, vim
- Shannon Kerr,
  - Equipment
    - Dell Inspiron, i5, 8GB, 1.6GHz 5482,
    - HDD 1TB 5.4K 6GB/s
    - Windows 64bit Home
    - Editor: Vim
- Erin Seliger
  - Equipment
    - Windows hp 2020, 16GB RAM, i7, 64-bit operating system
    - Windows Pro 64bit
    - Editor: Vim
- Cooper Young
  - Equipment
    - Dell Inspiron 7000, i7 2 Ghz, 16GB RAM, Intel Optane 512GB SSD

        * Windows 10 Education 64bit
        * Vim, Pycharm, Pico
- Agness Lungu
  - Equipment
    * Lenovo V570, 16GB RAM, intel(R) core(TM) i5-2430M, 64-bit operating system,
    * Windows 10 Education
    * editor: Vim, Pycharm

3 Introduction to MPI

- What is MPI and why do you want to use it
- What are some example MPI functionalities and usage patterns (send receive, embarrassing parallel)

3.1 Resources

- https://research.computing.yale.edu/sites/default/files/files/mpi4py.pdf
- https://www.nesi.org.nz/sites/default/files/mpi-in-python.pdf
- https://www.kth.se/blogs/pdc/2019/08/parallel-programming-in-python-mpi4py-part-1/
- http://education.molssi.org/parallel-programming/03-distributed-examples-mpi4py/index.html
- http://www.ceci-hpc.be/assets/training/mpi4py.pdf
- https://www.csc.fi/documents/200270/224366/mpi4py.pdf/825c582a-9d6d-4d18-a4ad-6cb6c43fefd8

3.2 MPI

☐ TODO: Open, what is mpi
☐ TODO: Open, Ring
☐ TODO: Open kmeans
☐ TODO: Who?, calculation of pi
☐ TODO: Who?, find number count of 8 in randome numbers between 1-10

3.3 Installation

☐ TODO: Cooper, how to find te number of cores in linux and gitbash so we can use this t define the -n core parameter

Linux: nproc

osx: sysctl hw.physicalcpu hw.logicalcpu

- which one can we use?

windows: ??? we want command in gitbash that gives it

3.3.1 Installation of mpi4py on Windows

1. First you need to download msmpi from

   - https://docs.microsoft.com/en-us/message-passing-interface/microsoft-mpi#ms-mpi-downloads

   Go to the download link and download and install it. Select the two packages and click Next. When downloaded clock on them to complete the setup

   ☐ TODO: Cooper, this seems incomplete is this correct. I changed it as previous install instructions were also incomplete.

```
msmpisetup.exe
msmpisdk.msi
```

2. Open the system control panel and click on `Advanced system settings` and then `Environment Variables`
3. Under the user variables box click on `Path`
4. Click New in order to add `C:\Program Files (x86)\Microsoft SDKs\MPI` and `C:\Program Files\Microsoft MPI\Bin` to the Path
5. Close any open bash windows and then open a new one
6. Type the command

```
$ which mpiexec
```

to verify if it works.

7. After you verified it is available, install mpi4py with

```
$ bash $ pip install mpi4py $
```

8. The installation can be tested with `mpiexec -n 4 python -m mpi4py.bench helloworld` (depending on the number of cores/nodes available to you, it may be necessary to reduce the number of copies that follow the -n option):

```
(ENV3) pi@red:~ $ mpiexec -n 4 python -m mpi4py.bench helloworld
Hello, World! I am process 0 of 4 on red.
Hello, World! I am process 1 of 4 on red.
Hello, World! I am process 2 of 4 on red.
Hello, World! I am process 3 of 4 on red.
```

3.3.2 Installing mpi4py in a Raspberry Pi

1. Activate our virtual environment:

```
$ python -m venv ~/ENV3
$ source ~/ENV3/bin/activate
```

2. Install Open MPI in your pi by entering the following command:

```
sudo apt-get install openmpi-bin
```

After installation is complete you can check if it was successful by using

```
mpicc --showme:version
```

3. Enter

```
pip install mpi4py
```

to download and install mpi4py.

4. The installation can be tested with `mpiexec -n 4 python -m mpi4py.bench helloworld` (depending on the number of cores/nodes available to you, it may be necessary to reduce the number of copies that follow the -n option) In a PI4, the previous test returned:

```
(ENV3) pi@red:~ $ mpiexec -n 4 python -m mpi4py.bench helloworld
Hello, World! I am process 0 of 4 on red.
Hello, World! I am process 1 of 4 on red.
Hello, World! I am process 2 of 4 on red.
Hello, World! I am process 3 of 4 on red.
```

3.3.3 Installing mpi4py in MacOS

☐ TODO: Agnes, incomplete

A similar process can be followed to install mpi4py in MacOS. In this case, we can use Homebrew to get Open MPI by entering:

```
$ brew install open-mpi
```

Once Open MPI is working, steps 3 and 4 from the PI4 installation can be followed in order to download and install mpi4py.

3.4 Hello World

To test if it works a build in test program is available.

To run it on on a single host with n cores (lest assume you have 2 cores), you can use:

```
mpiexec -n 4 python -m mpi4py.bench helloworld
Hello, World! I am process 0 of 5 on localhost.
Hello, World! I am process 1 of 5 on localhost.
Hello, World! I am process 2 of 5 on localhost.
Hello, World! I am process 3 of 5 on localhost.
```

Note that the messages can be in different order.

To run it on mulitple hosts with each having n cores please create a hostfile as follows:

☐ TODO: Open, how to run it on multiple hosts on the PI

3.5 Machine file, hostfile, rankfile

Run sudo apt-get install -y python-mpi4py on all nodes.

Test the installation: mpiexec -n 5 python -m mpi4py helloworld

THIS CAN BE DONE BEST WITH CLOUDMESH

FIRTS TEST BY HAND

☐ TODO: Open, VERIFY

```
mpirun.openmpi \
  -np 2 \
  -machinefile /home/pi/mpi_testing/machinefile \
  python helloworld.py
```

The machinefile contains the ipaddresses

```
pi@192. ....
yout add the ip addresses
```

☐ TODO: Open, learn about and evaluate and test if we can do

```
mpirun -r my_rankfile --report-bindings ...

Where the rankfile contains:
rank 0=compute17 slot=1:0
rank 1=compute17 slot=1:1
rank 2=compute18 slot=1:0
rank 3=compute18 slot=1:1
```

3.6 MPI Functionality examples

3.7 MPI Collective Communication functionality examples

3.7.1 Broadcast `comm.bcast()`

In this example, we broadcast a two-entry Python dictionary from a root process to the rest of the processes in our communicator group.

```python
from mpi4py import MPI

# Set up the MPI Communicator
comm = MPI.COMM_WORLD

# Get the rank of the current process in the communicator group
rank = comm.Get_rank()

if rank == 0:  # Process with rank 0 gets the data to be broadcast
    data = {'size' : [1,3,8],
            'name' : ['disk1', 'disk2', 'disk3']}
else:  # Other processes' data is empty
    data = None

# Print data in each process
print("before broadcast, data on rank %d is "%comm.rank, data)

# Data from process with rank 0 is broadcast to other processes in our
# communicator group
data = comm.bcast(data, root=0)

# Print data in each process after broadcast
print("after broadcast, data on rank %d is "%comm.rank, data)
```

After running `mpiexec -n 4 python bcast.py` we get the following:

```
before broadcast, data on rank 0 is
  {'size': [1, 3, 8], 'name': ['disk1', 'disk2', 'disk3']}
before broadcast, data on rank 1 is  None
before broadcast, data on rank 2 is  None
before broadcast, data on rank 3 is  None
after broadcast, data on rank 0 is
  {'size': [1, 3, 8], 'name': ['disk1', 'disk2', 'disk3']}
after broadcast, data on rank 1 is
  {'size': [1, 3, 8], 'name': ['disk1', 'disk2', 'disk3']}
after broadcast, data on rank 2 is
  {'size': [1, 3, 8], 'name': ['disk1', 'disk2', 'disk3']}
after broadcast, data on rank 3 is
  {'size': [1, 3, 8], 'name': ['disk1', 'disk2', 'disk3']}
```

As we can see, the process with rank 1, received the data broadcast from rank 0.

3.7.1.1 Scatter `comm.scatter()` In this example, with scatter the members of a list among the processes in the communicator group.

```python
from mpi4py import MPI

# Communicator
comm = MPI.COMM_WORLD

# Number of processes in the communicator group
size = comm.Get_size()

# Get the rank of the current process in the communicator group
rank = comm.Get_rank()

# Process with rank 0 gets a list with the data to be scattered
if rank == 0:
    data = [(i+1)**2 for i in range(size)]
else:
    data = None

# Print data in each process
print("before scattering, data on rank %d is "%comm.rank, data)

# Scattering occurs
data = comm.scatter(data, root=0)

# Print data in each process after scattering
print("data for rank %d is "%comm.rank, data)
```

Executing `mpiexec -n 4 python scatter.py` yields:

```
before scattering, data on rank 2 is  None
before scattering, data on rank 3 is  None
before scattering, data on rank 0 is  [1, 4, 9, 16]
before scattering, data on rank 1 is  None
data for rank 2 is  9
data for rank 1 is  4
data for rank 3 is  16
data for rank 0 is  1
```

The members of the list from process 0 have been successfully scattered among the rest of the processes in the communicator group.

3.7.1.2 Gather `comm.gather()` In this example, data from each process in the communicator group is gathered in the process with rank 0.

```
from mpi4py import MPI

# Communicator
comm = MPI.COMM_WORLD

# Number of processes in the communicator group
size = comm.Get_size()

# Get the rank of the current process in the communicator group
rank = comm.Get_rank()

# Each process gets different data, depending on its rank number
data = (rank+1)**2

# Print data in each process
print("before gathering, data on rank %d is "%comm.rank, data)

# Gathering occurs
data = comm.gather(data, root=0)

# Process 0 prints out the gathered data, rest of the processes
# print their data as well
if rank == 0:
    print("after gathering, process 0's data is ", data)
else:
    print("after gathering, data in rank %d is "%comm.rank, data)
```

Executing `mpiexec -n 4 python gather.py` yields:

```
before gathering, data on rank 2 is  9
before gathering, data on rank 3 is  16
before gathering, data on rank 0 is  1
before gathering, data on rank 1 is  4
after gathering, data in rank 2 is  None
after gathering, data in rank 1 is  None
after gathering, data in rank 3 is  None
after gathering, process 0's data is  [1, 4, 9, 16]
```

The data from processes with rank `1` to `size - 1` have been successfully gathered in process 0.

3.7.1.3 Broadcasting buffer-like objects `comm.Bcast()` In this example, we broadcast a NumPy array from process 0 to the rest of the processes in the communicator group.

```
from mpi4py import MPI
import numpy as np

# Communicator
comm = MPI.COMM_WORLD

# Get the rank of the current process in the communicator group
rank = comm.Get_rank()

# Rank 0 gets a NumPy array containing values from 0 to 9
if rank == 0:
    data = np.arange(0,10,1, dtype='i')

# Rest of the processes get an empty buffer
else:
    data = np.zeros(10, dtype='i')

# Print data in each process
print("before broadcasting, data for rank %d is: "%comm.rank, data)

# Broadcast occurs
comm.Bcast(data, root=0)

# Print data in each process after broadcast
print("after broadcasting, data for rank %d is: "%comm.rank, data)
```

Executing `mpiexec -n 4 python npbcast.py` yields:

```
before broadcasting, data for rank 1 is:  [0 0 0 0 0 0 0 0 0 0]
before broadcasting, data for rank 2 is:  [0 0 0 0 0 0 0 0 0 0]
before broadcasting, data for rank 3 is:  [0 0 0 0 0 0 0 0 0 0]
before broadcasting, data for rank 0 is:  [0 1 2 3 4 5 6 7 8 9]
after broadcasting, data for rank 0 is:  [0 1 2 3 4 5 6 7 8 9]
after broadcasting, data for rank 2 is:  [0 1 2 3 4 5 6 7 8 9]
after broadcasting, data for rank 3 is:  [0 1 2 3 4 5 6 7 8 9]
after broadcasting, data for rank 1 is:  [0 1 2 3 4 5 6 7 8 9]
```

As we can see, the values in the array at process with rank 0 have been broadcast to the rest of the processes in the communicator group.

3.7.1.4 Scattering buffer-like objects `comm.Scatter()` In this example, we scatter a NumPy array among the processes in the communicator group.

```
from mpi4py import MPI
import numpy as np

# Communicator
comm = MPI.COMM_WORLD

# Number of processes in the communicator group
size = comm.Get_size()

# Get the rank of the current process in the communicator group
rank = comm.Get_rank()

# Data to be sent
sendbuf = None

# Process with rank 0 populates sendbuf with a 2-D array,
# based on the number of processes in our communicator group
if rank == 0:
    sendbuf = np.zeros([size, 10], dtype='i')
    sendbuf.T[:,:] = range(size)

    # Print the content of sendbuf before scattering
    print('sendbuf in 0: ', sendbuf)

# Each process getd a buffer (initially containing just zeros)
# to store scattered data.
recvbuf = np.zeros(10, dtype='i')

# Print the content of recvbuf in each process before scattering
print('recvbuf in  %d: '%rank, recvbuf)

# Scattering occurs
comm.Scatter(sendbuf, recvbuf, root=0)

# Print the content of sendbuf in each process after scattering
print('Buffer in process %d contains: '%rank, recvbuf)
```

Executing `mpiexec -n 4 python npscatter.py` yields:

```
recvbuf in  1:  [0 0 0 0 0 0 0 0 0 0]
recvbuf in  2:  [0 0 0 0 0 0 0 0 0 0]
recvbuf in  3:  [0 0 0 0 0 0 0 0 0 0]
sendbuf in 0:  [[0 0 0 0 0 0 0 0 0 0]
                [1 1 1 1 1 1 1 1 1 1]
                [2 2 2 2 2 2 2 2 2 2]
                [3 3 3 3 3 3 3 3 3 3]]
recvbuf in  0:  [0 0 0 0 0 0 0 0 0 0]
Buffer in process 2 contains:  [2 2 2 2 2 2 2 2 2 2]
Buffer in process 0 contains:  [0 0 0 0 0 0 0 0 0 0]
Buffer in process 3 contains:  [3 3 3 3 3 3 3 3 3 3]
Buffer in process 1 contains:  [1 1 1 1 1 1 1 1 1 1]
```

As we can see, the values in the 2-D array at process with rank 0, have been scattered among all our processes

in the communicator group, based on their rank value.

3.7.1.5 Gathering buffer-like objects `comm.Gather()` In this example, we gather a NumPy array from the processes in the communicator group into a 2-D array in process with rank 0.

```
from mpi4py import MPI
import numpy as np

# Communicator group
comm = MPI.COMM_WORLD

# Number of processes in the communicator group
size = comm.Get_size()

# Get the rank of the current process in the communicator group
rank = comm.Get_rank()

# Each process gets an array with data based on its rank.
sendbuf = np.zeros(10, dtype='i') + rank

# Print the data in sendbuf before gathering
print('Buffer in process %d before gathering: '%rank, sendbuf)

# Variable to store gathered data
recvbuf = None

# Process with rank 0 initializes recvbuf to a 2-D array conatining
# only zeros. The size of the array is determined by the number of
# processes in the communicator group
if rank == 0:
    recvbuf = np.zeros([size,10], dtype='i')

    # Print recvbuf
    print('recvbuf in process 0 before gathering: ', recvbuf)

# Gathering occurs
comm.Gather(sendbuf, recvbuf, root=0)

# Print recvbuf in process with rank 0 after gathering
if rank == 0:
        print('recvbuf in process 0 after gathering: \n', recvbuf)
```

Executing `mpiexec -n 4 python npgather.py` yields:

```
Buffer in process 2 before gathering:  [2 2 2 2 2 2 2 2 2 2]
Buffer in process 3 before gathering:  [3 3 3 3 3 3 3 3 3 3]
Buffer in process 0 before gathering:  [0 0 0 0 0 0 0 0 0 0]
Buffer in process 1 before gathering:  [1 1 1 1 1 1 1 1 1 1]
recvbuf in process 0 before gathering:
 [[0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]
  [0 0 0 0 0 0 0 0 0 0]]
recvbuf in process 0 after gathering:
 [[0 0 0 0 0 0 0 0 0 0]
  [1 1 1 1 1 1 1 1 1 1]
  [2 2 2 2 2 2 2 2 2 2]
  [3 3 3 3 3 3 3 3 3 3]]
```

The values contained in the buffers from the different processes in the group have been gathered in the 2-D array in process with rank 0.

3.7.1.6 send receive

☐ TODO, Fidel send recieve

3.7.1.7 Dynamic Process Management with `spawn` In this example, we have two python programs, the first one being the manager and the second being the worker.

```python
#!/usr/bin/env python
from mpi4py import MPI
import numpy
import sys
import time
print("Hello")
comm = MPI.COMM_SELF.Spawn(sys.executable,
                           args=['worker.py'],
                           maxprocs=5)
rank = comm.Get_rank()
print(f"b and rank: {rank}")

N = numpy.array(100, 'i')
comm.Bcast([N, MPI.INT], root=MPI.ROOT)
#print(f"ROOT: {MPI.ROOT}")
print('c')
PI = numpy.array(0.0, 'd')

print('d')
comm.Reduce(None, [PI, MPI.DOUBLE],
            op=MPI.SUM, root=MPI.ROOT)
print(PI)

comm.Disconnect()
```

```
#!/usr/bin/env python
from mpi4py import MPI
import numpy
import time
import sys
comm = MPI.Comm.Get_parent()
size = comm.Get_size()
rank = comm.Get_rank()

N = numpy.array(0, dtype='i')
comm.Bcast([N, MPI.INT], root=0)
print(f"N: {N} rank: {rank}")

h = 1.0 / N
s = 0.0
for i in range(rank, N, size):
    x = h * (i + 0.5)
    s += 4.0 / (1.0 + x**2)
PI = numpy.array(s * h, dtype='d')
comm.Reduce([PI, MPI.DOUBLE], None,
            op=MPI.SUM, root=0)

#time.sleep(60)
comm.Disconnect()
#MPI.Finalize()
#sys.exit()
#MPI.Unpublish_name()
#MPI.Close_port()
```

To execute the example please go to the examples directoy and run the manager program

```
$ cd examples/spawn
$ mpiexec -n 4 python manager.py
```

This will result in:

```
N: 100 rank: 4
N: 100 rank: 1
N: 100 rank: 3
N: 100 rank: 2
Hello
b and rank: 0
c
d
3.1416009869231245
N: 100 rank: 0
N: 100 rank: 1
N: 100 rank: 4
N: 100 rank: 3
N: 100 rank: 2
Hello
b and rank: 0
c
d
3.1416009869231245
N: 100 rank: 0
```

This output depends on which child process is received first. The output can vary.

> `WARNING:` When running this program it may not terminate. To terminate use for now `CTRL-C`.

3.7.1.8 task processing (spawn, pull, . . .)

☐ TODO: Cooper, spawn, pull

3.7.1.8.1 Futures

☐ TODO: Open, futures

https://mpi4py.readthedocs.io/en/stable/mpi4py.futures.html

3.7.1.9 Examples for other collective communication methods

☐ TODO: Agnes, introduction

3.8 MPI-IO

☐ TODO: Agnes, MPI-IO

3.8.1 Collective I/O with NumPy arrays

☐ TODO: Agnes - IO and Numpy

3.8.2 Non-contiguous Collective I/O with NumPy arrays and datatypes

☐ TODO: Agnes, noncontigious IO

3.9 Monte Carlo calculation of Pi

☐ TODO: Shannon, improve

☐ TODO: Shannon WHAT IS THE PROBLEM GOAL

We start with the Mathematical formulation of the Monte Carlo calulation of pi. For each quadrant of the unit square, the area is pi. Therefore, the ratio of the area outside of the circle is pi over four. With this in mind, we can use the Monte Carlo Method for the calculation of pi.

☐ TODO: SHannon, Drawing

☐ TODO: Open, HOW AND WHY DO WE NEED MULTIPLE COMPUTERS

3.9.1 Program

☐ TODO: Shannon, PI montecarlo

☐ TODO: Shannon, Example program to run Montecarlo on multiple hosts

☐ TODO: Shannon, Benchmarking of the code

Use for benchmarking * cloudmesh.common (not thread safe, but still can be used, research how to use it in multiple threads) * other strategies to benchmark, you research (only if really needed * Use numba to speed up the code * describe how to install * showcase basic usage on our monte carlo function * display results with matplotlib

3.10 GPU Programming with MPI

Only possibly for someone with GPU (contact me if you do) Once we are finished with MPI we will use and look at python dask and other frameworks as well as rest services to interface with the mpi programs. This way we will be able to expose the cluster to anyone and they do not even know they use a cluster while exposing this as a single function ... (edited)

The github repo is used by all of you to have write access and contribute to the research effort easily and in parallel. You will get out of this as much as you put in. Thus it is important to set several dedicated hours aside (ideally each week) and contribute your work to others.

It is difficult to asses how long the previous task takes as we just get started and we need to learn first how we work together as a team. If I were to do this alone it may take a week, but as you are less experienced it would likely take longer. However to decrease the time needed we can split up work and each of you will work on a dedicated topic (but you can still work in smaller teams if you desire). We will start assigning tasks in github once this is all set up.

Idea:

WE WILL NO LONGER USE HACKMD AS GITHUB IS BETTER FOR INTEGRATION WITH CODE

- tutorial about hackmd.io https://hackmd.io/t9SkKiSLR5qW9RUUA_CT-A
- Github vs hackmd

We will use initially hackmd so we avoid issues with github and we can learn github once we do more coding.

3.11 Installing WSL on Windows 10

TODO: Cooper, what is wsl

To install WSL2 your computer must have Hyper-V support enabled. THis doe snot work on Windows HOmew and you need to upgrade to Windows Pro, Edu or some other Windows 10 version that supports it. Windows Edu is typically free for educational institutions. The HYper-V must be enabled from your Bios and you need to change your settings if it is not enabled.

More information about WSL is provided at

- https://docs.microsoft.com/en-us/windows/wsl/install-win10 for further detail

To install WSL2 you can follow these directions while using Powershell as an administrative user and run

```
dism.exe /online /enable-feature /featurename:Microsoft-Windows-Subsystem-Linux /all /norestart
dism.exe /online /enable-feature /featurename:VirtualMachinePlatform /all /norestart
wsl --set-default-version 2
```

Next, Download Ubuntu 20.04 LTS from the Microsoft store

- https://www.microsoft.com/en-us/p/ubuntu/9nblggh4msv6?activetab=pivot:overviewtab

Run Ubuntu and create a username and passphrase.

Make sur enot to just give an empty passphrase but chose a secure one.

☐ TODO: Cooper, how to start it the next time