Mainframe & Parallel Programming Assignment Report

- Description of what was done (and why)
- Description / Printout of results
- Source code & if applicable executable

Setup

I chose python as I wanted to further learn python and I was curious if it is suitable for parallel programming.

Assignment 1 was implemented using the message passing scheme with open-mpi[1] and the unofficial python library mpi4py[2] which provides bindings for the Message Passing Interface standard for python. The histogram graphs are created using the pyplot library[3].

Assignment 2 uses the shared address paradigm and is implemented using python's multiprocessing[4] package and numpy[5] for better and faster array handling.

To install all required dependencies run the following commands within the projects directory:

```
$ virtualenv venv
$ source venv/bin/activate
$ pip install -r requirements.txt
```

Assignment 1

Datasource

As datasource I downloaded the address list of members of the canton of Zurich [6].

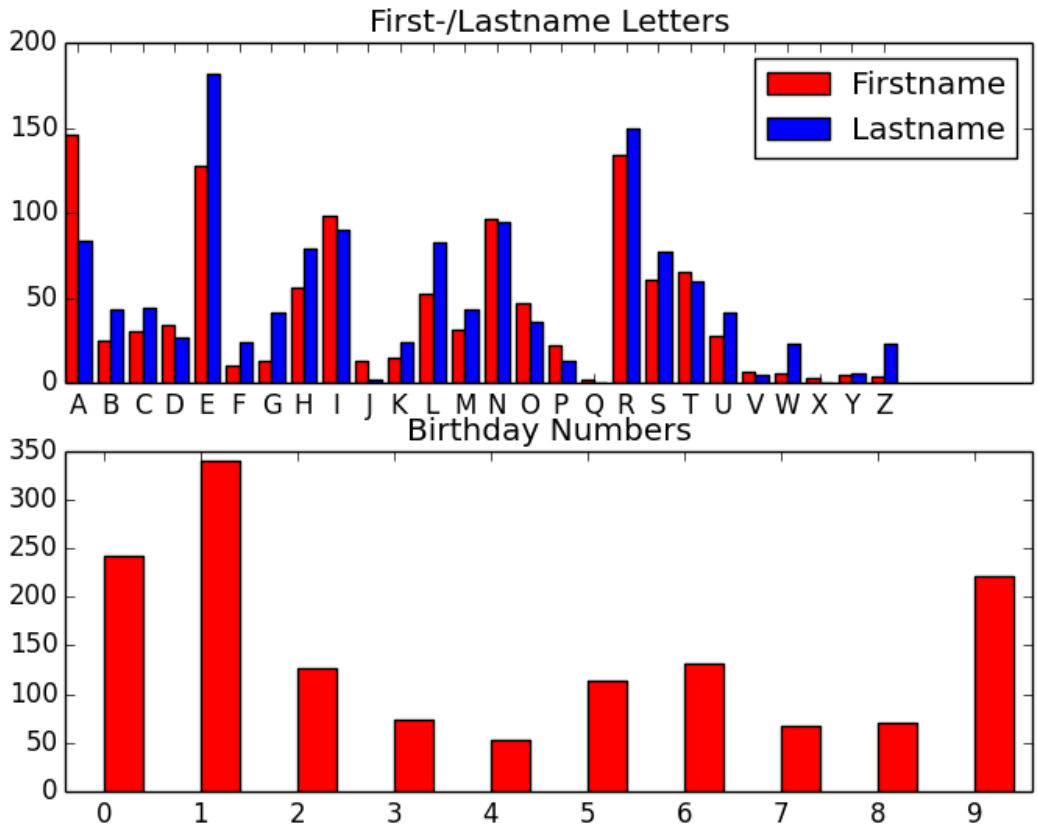
This source doesn't include any phone numbers, therefore I performed the exercises using the individual birth date of each member.

[1] http://www.open-mpi.org/
[2] http://mpi4py.scipy.org/
[3] http://matplotlib.org/api/pyplot_api.html
[4] https://docs.python.org/2/library/multiprocessing.html
[5] http://www.numpy.org/
[6] http://www.kantonsrat.zh.ch/mitglieder/mitglieder.aspx

Tasks

For all members the histogram of letters/numbers of first name, last name, phone number

```
$ mpiexec -n 4 python histogram_member.py
```



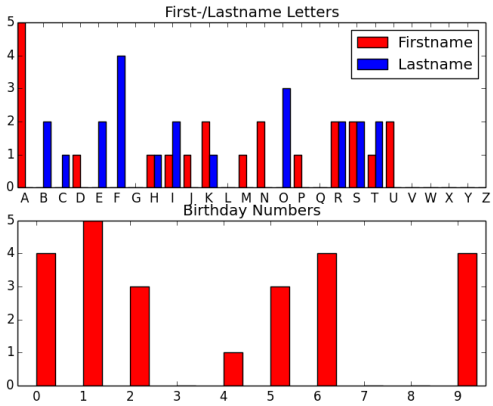
This command loads for all processes the list of all members into memory and then each process processes only every rank'th entry from the datafile.
The results are then summarized in the root process using a comm.Reduce.

All resulting graphs will be create in the `histograms` sub-directory.

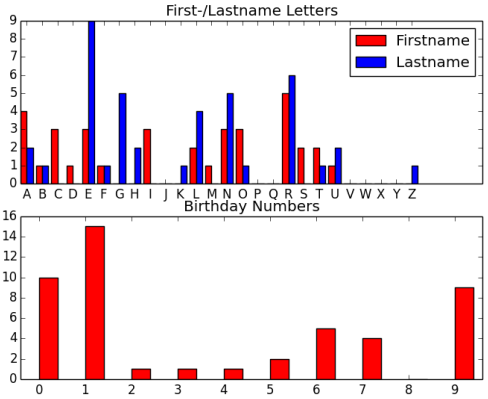
For all parties the histogram of letters/numbers of first name, last name, phone number

```
$ mpiexec -n 3 python histogram_party.py
```

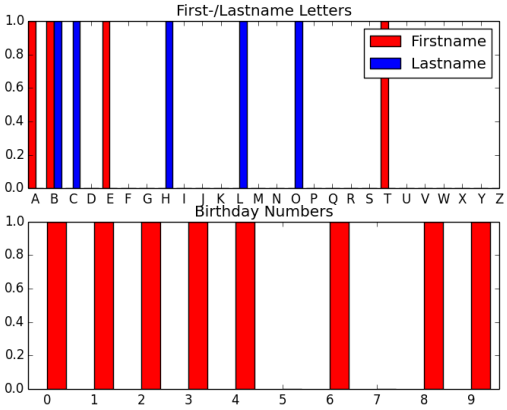
AL



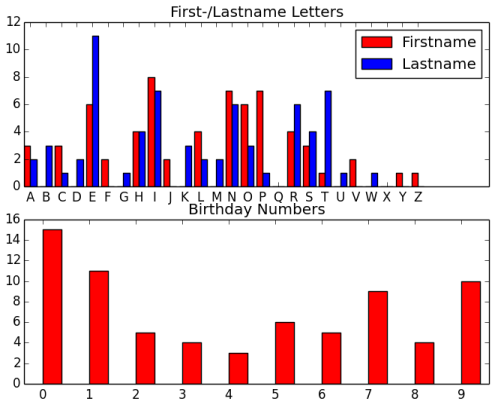
BDP



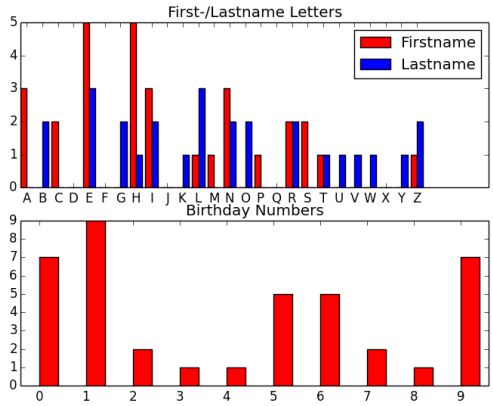
CSP



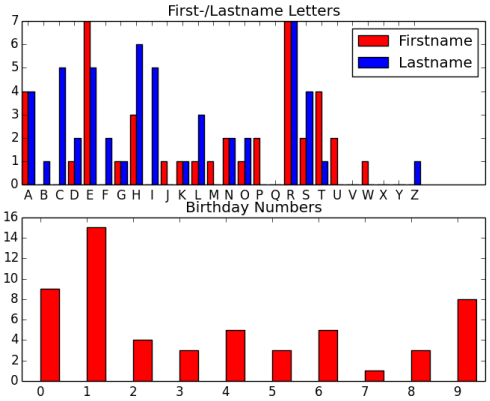
CVP



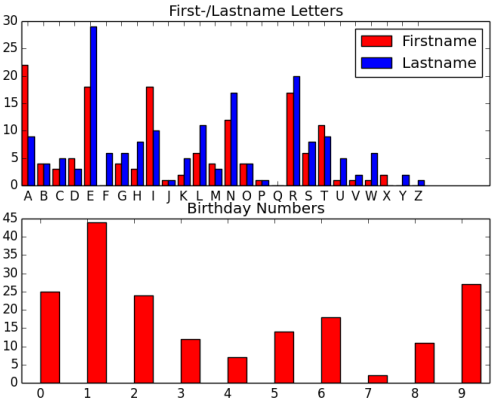
EDU



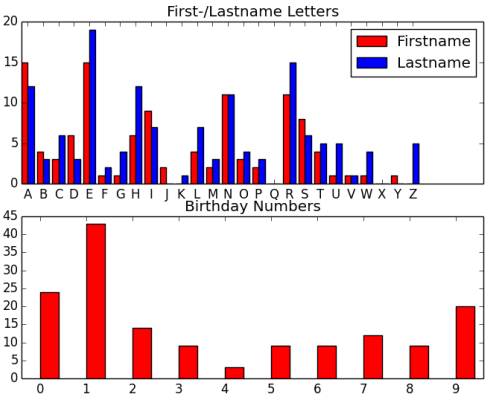
EVP



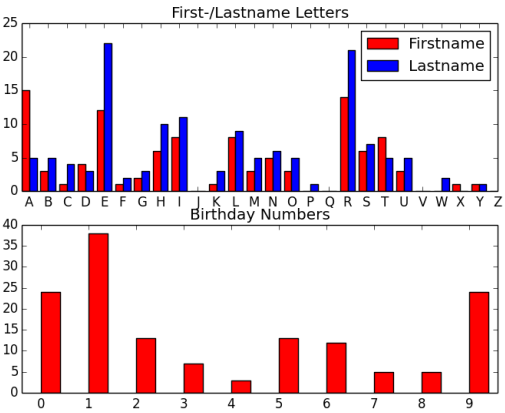
FDP



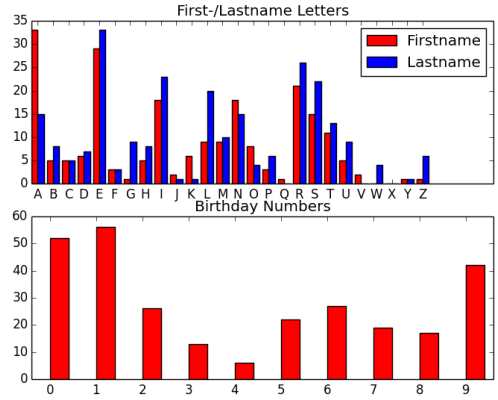
GLP



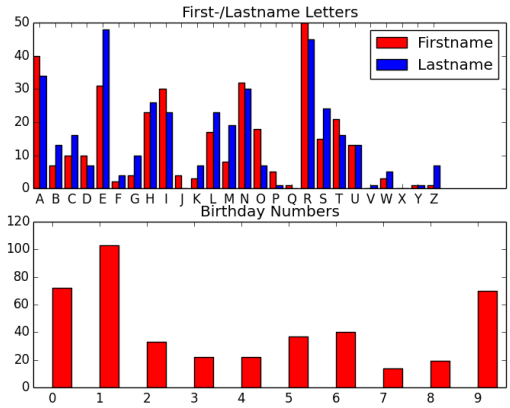
Grüne



SP



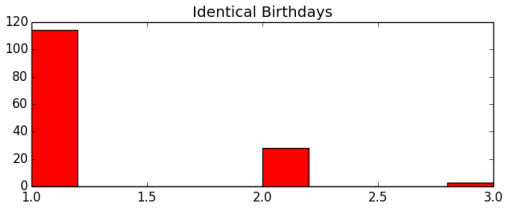
SVP

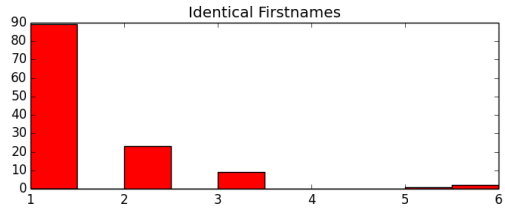


Same logic as histograms per member but grouped by party.

Histogram of identical first names, birth date

```
$ mpiexec -n 4 python histogram_identical_names.py
$ mpiexec -n 4 python histogram_identical_birthdays.py
```





Each process reads only every rank'th row from the datafile into memory and processes the rows while reading from the datafile.

These commands use comm.send where every process sends its currently processing firstname / birthday to the root process which collects all incoming messages with comm.Irecv from any source and sums their occurrences.

Assignment 2

Tasks

1. compute the position of the centre of the mass
2. 2^10 times randomly select 2^10 particles and compute the position of the center of the selected masses
3. shift a randomly selected sub set of particles a random distance towards the center of the mass system
4. find all particles at a given distance range from a given location

```
$ python 3dset.py
```

Full command output is available in the `3dset.output.txt` file.

Each task consists of a mp_*TASK* function which takes care of distributing the task to every process and merging the individual results together and a *TASK* function which computes the actual result.

The particles are created in a 3D room of size 2^16 in each dimension. All task variables are adjustable at the beginning of the script file.

Assignment Part 2 (d): Finds all particles within a distance of 1500 from location (32669, 32824, 32726).