Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

High-Performance Computing Lab for CSE **2024**

Student: Benedict Armstrong Discussed with:

Solution for Project 5

Due date: Monday 13 May 2024, 23:59 (midnight).

1. Introduction

All benchmarks and programs were run on the **Euler VII — phase 2 cluster** with **AMD EPYC 7763** cpus.

2. Parallel Space Solution of a nonlinear PDE using MPI [in total 60 points]

2.1. Initialize/finalize MPI and welcome message [5 Points]

2.2. Domain decomposition [10 Points]

2.3. Linear algebra kernels [5 Points]

2.4. The diffusion stencil: Ghost cells exchange [10 Points]

2.5. Implement parallel I/O [10 Points]

2.6. Strong scaling [10 Points]

2.7. Weak scaling [10 Points]

2.8. Bonus [20 Points]: Overlapping computation/computation details

3. Python for High-Performance Computing [in total 40 points]

3.1. Sum of ranks: MPI collectives [5 Points]

For this task I translated the cpp code from project04/ring to python. As outlined in the task description I implemented one version which communicates using python objects and another version which uses `NumPy` arrays. To test the code I ran the following commands:

```
mpirun -np 8 python3 slow_comm.py | sort > slow_comm.txt
mpirun -np 8 python3 fast_comm.py | sort > fast_comm.txt
```

The respective code and text files with the output can be found in `code/hpc_python/rank_sum`.

3.2. Ghost cell exchange between neighboring processes [5 Points]

Again i started this task by translating cpp code this time from project04/ghost to python. The implementation of the ghost cell exchange using `NumPy` arrays was pretty straight forward except for the sending of the first and last columns. As far as I could tell `mpi4py` doesn't really support sending non memory contiguous data. To work around this I created a copy of the data I wanted to send as a contiguous array.

```
```

```
left_s = data[1, 1:-1].copy()
right_s = data[-2, 1:-1].copy()
```

We also need to create a contiguous receiving buffer for the ghost cells.

```
left_r = np.zeros(SUBDOMAIN, dtype=np.int64)
right_r = np.zeros(SUBDOMAIN, dtype=np.int64)
```

After sending the ghost cells we can receive them and copy them into the correct position.

```
data[1:-1, 0] = left_r
data[1:-1, -1] = right_r
```

To test the code I ran the following commands:

```
mpirun -np 16 python3 ghost.py
```

Which as expected prints the correct output:

```
[[ 9  5  5  5  5  5  5  9]
 [ 8  9  9  9  9  9  9 10]
 [ 8  9  9  9  9  9  9 10]
 [ 8  9  9  9  9  9  9 10]
 [ 8  9  9  9  9  9  9 10]
 [ 8  9  9  9  9  9  9 10]
 [ 8  9  9  9  9  9  9 10]
 [ 9 13 13 13 13 13 13  9]]
```

3.3. A self-scheduling example: Parallel Mandelbrot [30 Points]