Exercise 6

To compile: unzip our uploaded code, and run `make` inside `code/`. The slurm scripts are stored inside `code/slurm/`.

To debug: run the debug outputs (`*.dbg`) and attach gdb to respective pids

6.1 Heat Relaxation II — Parallel Implementation

6.2 Heat Relaxation II — Experiments

The implementation in subsection 6.1 resulted in the values below (Table 1, Table 2, and Table 3). We choose the by-slot-Mapping (default of mpirun) as to reduce the number of hops between nodes. This Mapping starts filling a node's possible slots with ranks until full and then continues with another node. This results in rank 3 and 4 and ranks 7 and 8 communication between nodes (for 9 or more ranks).

Table 1: Time [μs] / iteration

Grid size	NP = 01	NP = 02	NP = 04	NP = 06	NP = 08	NP = 10	NP = 12
128x128	68	39	44	526	2373	693	743
512x512	3363	851	757	3007	932	1417	3294
1024x1024	19474	9524	5684	6003	2722	3068	2573
2048x2048	238361	134284	30991	22886	16389	13306	13621
4096x4096	855031	551723	365791	130806	78609	68622	61378

Table 2: Speedup

Grid size	NP = 01	NP = 02	NP = 04	NP = 06	NP = 08	NP = 10	NP = 12
128x128	1.0000	1.7289	1.5388	0.1299	0.0288	0.0987	0.0920
512x512	1.0000	3.9487	4.4397	1.1185	3.6086	2.3725	1.0209
1024x1024	1.0000	2.0447	3.4260	3.2438	7.1544	6.3462	7.5672
2048x2048	1.0000	1.7750	7.6911	10.4150	14.5433	17.9135	17.4986
4096x4096	1.0000	1.5497	2.3375	6.5366	10.8769	12.4600	13.9305

Table 3: Efficiency

Grid size	NP = 01	NP = 02	NP = 04	NP = 06	NP = 08	NP = 10	NP = 12
128x128	1.0000	0.8645	0.3847	0.0217	0.0036	0.0099	0.0077
512x512	1.0000	1.9743	1.1099	0.1864	0.4511	0.2373	0.0851
1024x1024	1.0000	1.0224	0.8565	0.5406	0.8943	0.6346	0.6306
2048x2048	1.0000	0.8875	1.9228	1.7358	1.8179	1.7913	1.4582
4096x4096	1.0000	0.7749	0.5844	1.0894	1.3596	1.2460	1.1609

- A speedup is observed, that correlates to the number of jobs (i.e. for 10 jobs we reach a speedup o approx 10, for a sufficiently large problem size)
- For problem sizes too small, performance drops, due to the communication overhead dominating
- Additionally super linear speedups were observed, probably due to better cache utilization.

6.3 Heat Relaxation II — Tracing

6.3.1 Expectations From Tracing

Since the stencil is a memory and computation heavy tasks we would expect most time being spend inside the stencil computation i.e. `relaxation_step()` function in our case. Since only halo values are sent among tasks we don't expect communication to be as intensive.

6.3.2 Expectations From Experiments

The performed experiments meet our expectations. Smaller workloads get worse with rising number of tasks since communication overhead increases. For larger workloads we reach speedups approximately proportional to the number of tasks.

6.3.3 Tracing

Using *scorep* to trace our application confirms our initial expectation:

type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
ALL	146,889,537	22,594,391	4.83	100.0	0.21	ALL
USR	146,863,080	22,593,123	2.39	49.5	0.11	USR
MPI	26,260	1,240	2.44	50.4	1963.92	MPI
COM	156	24	0.01	0.2	310.43	COM
SCOREP	41	4	0.00	0.0	24.52	SCOREP

Figure 1: Tracing result for problem size of 128x128. Most time (around 50%) is spent in MPI as expected.

type	max_buf[B]	visits	time[s]	time[%]	time/visit[us]	region
ALL	3,813,300,597	586,642,871	65.92	100.0	0.11	ALL
USR	3,813,297,540	586,642,683	62.92	95.5	0.11	USR
MPI	2,860	160	2.99	4.5	18691.30	MPI
COM	156	24	0.00	0.0	117.39	COM
SCOREP	41	4	0.00	0.0	25.83	SCOREP

Figure 2: Using a larger problem size of 2048x2048 MPI time becomes almost negligible (around 4%)

For large problems the number of iterations does not seem to change the result too much. Looking at user time we can also see that most of the time is spent accessing vector items:

```
time[s] time[%] time/visit[us]  region
 629.17   100.0          0.11  ALL
 625.42    99.4          0.11  USR
   3.73     0.6       3010.76  MPI
   0.02     0.0        643.23  COM
   0.00     0.0         25.46  SCOREP

 134.27    21.3          0.05  _ZNKSt6vectorIdSaIdEEixEm
 263.76    41.9          0.10  _ZNK4GridclEmm
  45.06     7.2          0.11  _ZN4GridclEmm
  22.60     3.6          0.05  _ZNSt6vectorIdSaIdEEixEm
   0.00     0.0          0.13  _ZNSt6vectorI4GridSaIS0_EEixEm
   0.18     0.0        221.66  MPI_Sendrecv
   2.26     0.4       5644.62  MPI_Barrier
 159.59    25.4     398963.85  _Z15relaxation_stepRK4GridRS_
   0.00     0.0          0.08  _ZNKSt8__detail18_Mod_range_hashingclEmm
   0.00     0.0          0.13  _ZNSt8__detail21_Hashtable_ebo_helperILi2ENS_18_Mod_range_hashingELb
```

Figure 3: We can see that 25% of time is spent `relaxation_step()` doing actual computation. 42% of the time is spent fetching elements from the grid

6.3.4 Insights from tracing

Using this information we could optimize memory access further (speed up index calculation).