High Performance Computing for Science and Engineering II.

Exercise Set 3

Florian Mahlknecht

2020-03-31

Please refer to the source code for the complete solution, this document shows only some minor highlights.

1 OMP Barrier [Q3]

Regarding question 3, the barrier is needed, in order to make sure that the swap operation is not executed on scattered results. Afterwards a single threads needs to to the increments in count and time, and a barrier (implicit or explicit) needs to make sure the exit criteria is checked for all threads under the same condition, see listing 1.

```
do {

  // [...]

  // to make sure all calculation is done before it's swapped
  #pragma omp barrier

  #pragma omp single
  {
    std::swap(u_new, u);
    std::swap(u_new, u_old);
    t += dt;
    count++;
  }

  // implicit barrier is important here to get the exit criteria right (!)
  // [e.g. omp master does not have an implicit barrier,
  //  in that case another explicit barrier would be needed]

} while (t < t_end);
```

Listing 1: Barrier implementation

fmahlknecht@student.ethz.ch
19-945-351

2 Communication tolerant programming [Q5]

For communication hiding two approaches have been implemented. The most straight forward one is to exclude the boundaries in the hidden computation, as shown in listing 2.

```
// skip boundaries
for (int i0 = 2 + i0_min; i0 < i0_max; i0++)
  for (int i1 = 2 + i1_min; i1 < i1_max; i1++)
    for (int i2 = 2 + i2_min; i2 < i2_max; i2++)
      UpdateGridPoint(i0, i1, i2);


MPI_Waitall(local_request.size(), local_request.data(),MPI_STATUSES_IGNORE);
unpack_all();

for (int i1 = 1 + i1_min; i1 < i1_max + 1; i1++)
  for (int i2 = 1 + i2_min; i2 < i2_max + 1; i2++) {
    UpdateGridPoint(1+i0_min, i1, i2);
    UpdateGridPoint(i0_max, i1, i2);
  }

for (int i0 = 2 + i0_min; i0 < i0_max; i0++)
  for (int i2 = 1 + i2_min; i2 < i2_max + 1; i2++) {
    UpdateGridPoint(i0, 1+i1_min, i2);
    UpdateGridPoint(i0, i1_max, i2);
  }

for (int i0 = 2 + i0_min; i0 < i0_max; i0++)
  for (int i1 = 2 + i1_min; i1 < i1_max; i1++) {
    UpdateGridPoint(i0, i1, 1+i2_min);
    UpdateGridPoint(i0, i1, i2_max);
  }
```

Listing 2: Simple version of communication hiding

Threads which are inside the mpi-subcube, do not wait and continue the calculation of the boundaries immediately, thanks to the variable size of the request vector. However, for cache coherency this may not be ideal, since for those threads which actually could perform all the calculations at once, it may happen that adjacent data has to be reloaded from RAM. Listing 3 shows a way to mitigate this minor issue. Both of them have been tested to produce identical checksums.

Please note that regarding the if statements inside the for loops, the compiler optimization should optimize this out, since the checks do not depend on the iteration variables. By writing it this way it has been avoided to duplicate the lines of code to 6x doulbe for loops with the if statement in front. It should be a low hanging fruit to pick for the optimizer, especially in `-O3`.

Florian Mahlknecht
fmahlknecht@student.ethz.ch
19-945-351

```
// skip boundaries if they are mpi-wide
for (int i0 = i0_min + (t0==0 ? 2:1); i0 < i0_max + (t0==p-1 ? 0:1); i0++)
  for (int i1 = i1_min + (t1==0 ? 2:1); i1 < i1_max + (t1==p-1 ? 0:1); i1++)
    for (int i2 = i2_min + (t2==0 ? 2:1); i2 < i2_max + (t2==p-1 ? 0:1); i2++)
      UpdateGridPoint(i0, i1, i2);


MPI_Waitall(local_request.size(), local_request.data(),MPI_STATUSES_IGNORE);
unpack_all();

for (int i1 = 1 + i1_min; i1 < i1_max + 1; i1++)
  for (int i2 = 1 + i2_min; i2 < i2_max + 1; i2++) {
    if (t0 == 0) UpdateGridPoint(1+i0_min, i1, i2);
    if (t0 == p-1) UpdateGridPoint(i0_max, i1, i2);
  }

for (int i0 = i0_min + (t0==0 ? 2:1); i0 < i0_max + (t0==p-1 ? 0:1); i0++)
  for (int i2 = 1 + i2_min; i2 < i2_max + 1; i2++) {
    if (t1 == 0) UpdateGridPoint(i0, 1+i1_min, i2);
    if (t1 == p-1) UpdateGridPoint(i0, i1_max, i2);
  }

for (int i0 = i0_min + (t0==0 ? 2:1); i0 < i0_max + (t0==p-1 ? 0:1); i0++)
  for (int i1 = i1_min + (t1==0 ? 2:1); i1 < i1_max +  (t1==p-1 ? 0:1); i1++) {
    if (t2 == 0) UpdateGridPoint(i0, i1, 1+i2_min);
    if (t2 == p-1) UpdateGridPoint(i0, i1, i2_max);
  }
```

Listing 3: More sophisticated version of communication hiding

2.1 Results on Euler

The code was tested on one full node with Xeon Gold 6150, with 8 and 1 threads respectively. 8 mpi processes have been used.

```
export OMP_NUM_THREADS=8 ; mpirun -n 8 ./main 512 2 0.25
0 t=0
10 t=0.0112764
[...]
220 t=0.24808
Total time = 9.57147
Checksum = 4.09911e+06
```

Compared to 1 thread a significant speed up could be noticed.

```
export OMP_NUM_THREADS=1 ; mpirun -n 8 ./main 512 2 0.25
0 t=0
10 t=0.0112764
[...]
220 t=0.24808
Total time = 25.7277
Checksum = 4.09911e+06
```


fmahlknecht@student.ethz.ch
19-945-351