UNIVERSITÄT SIEGEN

Fakultät IV
Betriebssysteme und verteilte Systeme
Prof. Dr. rer. nat. Roland Wismüller

Excercise Sheet 8
Solution

Lecture Parallel Processing
Winter Term 2024/25

Exercise 1: Parallelization of the Jacobi method using MPI (Compulsory Exercise, Weight 3! Submit until Tuesday, January 28th, 10:00 via moodle)

Exercise 2: Parallelization of the Gauss/Seidel method using MPI (For Motivated Students)

The solution below uses non-blocking receive and buffered send operations in order to avoid any blocking of the processes. In addition, the `i`-loop has been split into three parts (`i = 1`, `i = 2 .. local_n-3`, and `i = local_n-2`) in order to avoid the overhead of an `if`-statement in the loop.

Only the `solcer()` function is shown here. The `main()` Program is the same as for the Jacobi solver.

```
int solver(double **a, int local_n, int n)
{
  int i,j,k;
  MPI_Status status;
  MPI_Request request1, request2;
  int bufsize = 100 * n * sizeof(double);
  char *buffer = new char[bufsize];
  int flag;

  //
  // Simple estimation for the number of iterations, which is needed to
  // achieve the required accuracy.
  //
  int kmax = (int)(0.35 / eps);

  assert(local_n >= 3);

  MPI_Buffer_attach(buffer, bufsize);

  if (myrank != 0) {
    MPI_Irecv(&a[0][1], n-2, MPI_DOUBLE, myrank-1, 1, MPI_COMM_WORLD,
         &request1);
  }
  //
  // Perform 'kmax' iterations
  //
  for (k=0; k<kmax; k++) {

    if ((k != 0) && (myrank != nprocs-1)) {
      MPI_Irecv(&a[local_n-1][1], n-2, MPI_DOUBLE, myrank+1, 2,
           MPI_COMM_WORLD, &request2);
    }
    if (myrank != 0) {
      MPI_Wait(&request1, &status);
    }
```

```
    // Iteration i = 1
    for (j=1; j<n-1; j++) {
      a[1][j] = 0.25 * (a[1][j-1] + a[0][j] +
                a[2][j] + a[1][j+1]);
    }

    if ((k != kmax-1) && (myrank != 0)) {
      MPI_Irecv(&a[0][1], n-2, MPI_DOUBLE, myrank-1, 1, MPI_COMM_WORLD,
            &request1);
      MPI_Bsend(&a[1][1], n-2, MPI_DOUBLE, myrank-1, 2, MPI_COMM_WORLD);
    }

    // Iterations i = 2 .. local_n-3
    for (i=2; i<local_n-2; i++) {
      if ((i & 31) == 0) {
        if ((k != kmax-1) && (myrank != 0))
          MPI_Test(&request1, &flag, &status);
        if ((k != 0) && (myrank != nprocs-1))
          MPI_Test(&request2, &flag, &status);
      }
      for (j=1; j<n-1; j++) {
        a[i][j] = 0.25 * (a[i][j-1] + a[i-1][j] +
                  a[i+1][j] + a[i][j+1]);
      }
    }

    if ((k != 0) && (myrank != nprocs-1)) {
      MPI_Wait(&request2, &status);
    }

    // Iteration i = local_n-2
    for (j=1; j<n-1; j++) {
      a[local_n-2][j] = 0.25 * (a[local_n-2][j-1] + a[local_n-3][j] +
                a[local_n-1][j] + a[local_n-2][j+1]);
    }

    if (myrank != nprocs-1) {
      MPI_Bsend(&a[local_n-2][1], n-2, MPI_DOUBLE, myrank+1, 1,
           MPI_COMM_WORLD);
    }
  }

  MPI_Buffer_detach(&buffer, &bufsize);

  return kmax;
}
```

Exercise 3: Acquaint yourself with the electronic exam system