Prof. Dr. Peter Bastian, Santiago Ospina De Los Ríos
IWR, Universität Heidelberg

Submission date: 2023-12-19

Notes:

- You may volunteer for individual subtasks.

Exercise 1 *Parallel Jacobi Iteration*

In this exercise we will introduce a convergence criterion to the Jacobi iteration from the lecture and think about parallelization.

1. Familiarize yourself with the stencil code and the theory from the lecture.
2. Implement a convergence check. Let u^k be the grid values after k iterations. Define the defect norm

$$\|u^k\| = \sqrt{\sum_{i_0=1}^{n-2}\sum_{i_1=1}^{n-2}(4u^k_{i_0,i_1} - u^k_{i_0-1,i_1} - u^k_{i_0+1,i_1} - u^k_{i_0,i_1-1} - u^k_{i_0,i_1+1})^2}$$

   Then the iteration is stopped if

$$\|u^k\| \leq TOL \cdot \|u^0\|$$

   with TOL a user given tolerance. The computation of the norm may be amorthized over several iterations, i.e., the check is not done in every iteration.
3. Test omp-parallel version including the convergence check. Plot the number of iterations needed for $TOL = 10^{-3}$ and varying the n. Use initial values

$$u^0_{i_0,i_1} = 0 \qquad 1 \leq i_0, i_1 \leq n-2$$

4. Choose **one** of the two following exercises:
   (a) Implement a parallel Jacobi version with improved reuse (”wave”-algorithm). Use the following approach:

   For any integer parameter $K \geq 2$, even, perform $M \cdot K$ iterations with diamond-shaped regions executed in a wave-front fashion as illustrated in figure 1.

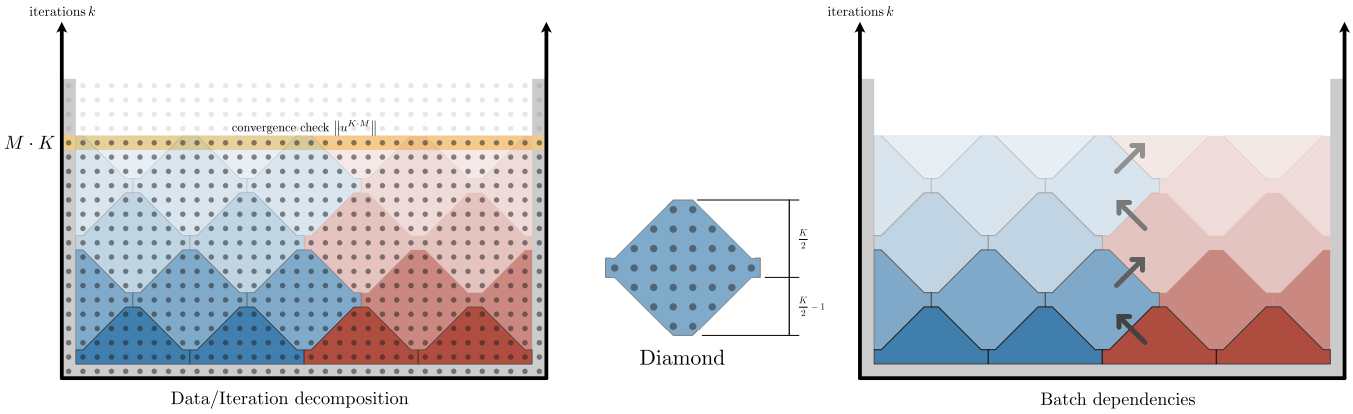

Figure 1: 1D example of a parallel Jacobi diamond wavefront iteration with $K = 8$, $M = 2$, and $P = 2$. Each point represents the value $u_{i_0}^k$ where the horizontal axis is for the i_0 index of the grid and the vertical axis the k-th iteration of the Jacobi method. The values within blue and red figures can be done by different threads in parallel where the darker shades are processed before lighter shades as long as different threads fulfill the batch dependencies represented on the right figure.

Check convergence after $M \cdot K$ iterations. Use C++ threads or extend the TBB version from the lecture. You may assume any further divisibilities you may need. Plot the performance of your implementation for different n and number of threads.

(b) Implement a parallel Jacobi version with data grid dependencies. Use the following approach:

Decompose the grid values into P chunks of equal size, i.e., $P^{\frac{1}{d}}$ chunks per direction as shown in figure 2. Compute the iteration k of each chunk in parallel and check convergence after M iterations. When computing the iteration k for a given chunk, make sure to resolve its data dependencies. In other words, the border values of the chunks need to wait until the neighboring chunk is done with iteration $k-1$ for its border values as shown with the red arrows in the figure 2. Try minimize the waiting time by switching between processing the boundary or interior values.

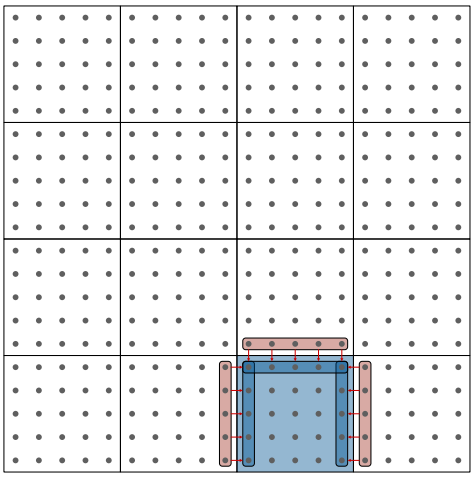

Figure 2: 2D example of a parallel Jacobi with grid decomposition of $P = 16$ chunks. The chunk in blue is computing iteration k and needs that the neighboring chunks are ready with their border for iteration $k-1$ before the blue chunk can compute its border values.

(0+3+3+10 Points)