OpenMP: An Advanced Example

Computational Science II (CAAM 520)

Christopher Thiele

Rice University, Spring 2020

Motivation

Our examples so far were simple in the sense that adding an OpenMP directive to a loop was usually sufficient.

In general, parallelization can be more complicated due to dependencies between loop iterations.

$\longrightarrow$ Let us consider such an example.

The heat equation

We want to solve the heat equation

$$
\begin{aligned}
\partial_t u - \Delta u &= f && \text{in } \Omega \times (0, T), \\
u &= 0 && \text{on } \partial\Omega \times (0, T), \\
u &= u_0 && \text{on } \Omega \times \{0\},
\end{aligned}
$$

where $\Omega = [0, 1]^3$, u is the temperature, and f describes heat sources and heat sinks inside Ω.

Poisson's equation

Instead of solving the full time-dependent problem, we are interested in the ***steady state*** solution which satisfies $\partial_t u = 0$.

This leads to the ***Poisson problem***

$$\begin{aligned} -\Delta u &= f && \text{in } \Omega, \\ u &= 0 && \text{on } \partial\Omega. \end{aligned}$$

Finite difference discretization

To discretize the equation, we introduce n^3 grid points

$$x_{ijk} = \left(ih, jh, kh\right)^T,$$

where $i, j, k = 0, \ldots, n-1$ and $h = \frac{1}{n-1}$.

For convenience, we define

$$u_{ijk} = u(x_{ijk})$$

etc.

Finite difference discretization

Discretizing the equation using finite differences (at an interior point x_{ijk}) yields

$$\frac{-u_{i-1jk} - u_{ij-1k} - u_{ijk-1} + 6u_{ijk} - u_{i+1jk} - u_{ij+1k} - u_{ijk+1}}{h^2} = f_{ijk},$$

or equivalently

$$u_{ijk} = \frac{h^2 f_{ijk} + u_{i-1jk} + u_{ij-1k} + u_{ijk-1} + u_{i+1jk} + u_{ij+1k} + u_{ijk+1}}{6},$$

i.e., a linear system.

The Jacobi iteration

The linear system can be written in matrix form and solved, e.g., with Gaussian elimination.

Since it is ***sparse***, it can also be solved iteratively. The simplest iterative method is the ***Jacobi*** iteration

$$u_{ijk}^{\text{new}} \leftarrow \frac{h^2 f_{ijk} + u_{i-1jk}^{\text{old}} + u_{ij-1k}^{\text{old}} + u_{ijk-1}^{\text{old}} + u_{i+1jk}^{\text{old}} + u_{ij+1k}^{\text{old}} + u_{ijk+1}^{\text{old}}}{6},$$

$\rightarrow$ This method is straightforward to parallelize, but requires many iterations.

The Gauss–Seidel iteration

Another method, which converges faster in the sense that it requires fewer iterations, is the ***Gauss–Seidel*** iteration

$$u_{ijk}^{\text{new}} \leftarrow \frac{h^2 f_{ijk} + u_{i-1jk}^{\text{new}} + u_{ij-1k}^{\text{new}} + u_{ijk-1}^{\text{new}} + u_{i+1jk}^{\text{old}} + u_{ij+1k}^{\text{old}} + u_{ijk+1}^{\text{old}}}{6}.$$

$\longrightarrow$ Since u_{ijk}^{new} depends on updated values at other grid points, how can we parallelize the iteration?

Remark

In practice, the tradeoff between the number of iterations and the per-iteration cost is nontrivial.

It depends on the problem at hand whether the Jacobi method or the Gauss–Seidel method yields in lower time-to-solution.