coursera

3.4 Lecture Summary

3 Loop Parallelism

3.4 One-Dimensional Iterative Averaging

Lecture Summary: In this lecture, we discussed a simple *stencil* computation to solve the recurrence, $X_i = (X_{i-1} + X_{i+1})/2$ with boundary conditions, $X_0 = 0$ and $X_n = 1$. Though we can easily derive an analytical solution for this example, ($X_i = i/n$), most stencil codes in practice do not have known analytical solutions and rely on computation to obtain a solution.

The Jacobi method for solving such equations typically utilizes two arrays, oldX[] and newX[]. A naive approach to parallelizing this method would result in the following pseudocode:

```
for (iter: [0:nsteps-1]) {
  forall (i: [1:n-1]) {
    newX[i] = (oldX[i-1] + oldX[i+1]) / 2;
  }
  swap pointers newX and oldX;
}
```

Though easy to understand, this approach creates $nsteps \times (n-1)$ tasks, which is too many. Barriers can help reduce the number of tasks created as follows:

```
forall ( i: [1:n-1]) {
  localNewX = newX; localOldX = oldX;
  for (iter: [0:nsteps-1]) {
    localNewX[i] = (localOldX[i-1] + localOldX[i+1]) / 2;
    NEXT; // Barrier
    swap pointers localNewX and localOldX;
  }
}
```

In this case, only $(n-1)$ tasks are created, and there are *nsteps* barrier (*next*) operations, each of which involves all $(n-1)$ tasks. This is a significant improvement since creating tasks is usually more expensive than performing barrier operations.

Optional Reading:

1. Wikipedia article on Stencil codes