Odd-Even Communication Group

Problem

Parallelization is required on a series of ordered data mapped to a sequential topology. The ordered data are dependent on only its neighbors, and each needs to wait before the update from the dependent neighbor arrives. Having previous data elements processed before the current one, it forms a sequential chain of dependence, that of counter-intuitive to parallelization. But special property is present in the dataset that makes the very start of the update anywhere in the dataset (i.e. does not have to be the beginning of the dataset). This allows the dataset to begin updating not necessary from the beginning, but anywhere, and possibly in different points simultaneously. Note this does not mean the dependence is removed. It is still present and governs how elements interact.

Driving Forces

1. Dependence relationship needs to be followed.
2. Result needs to stay valid, any any algorithmic that were present in the sequential code must be also preserved.
3. Simultaneous communication must occur (parallelization must be more than processors sequentially passing data to another).
4. Need to push in as many *concurrent communications* as possible. Equivalently we need to have as many local start as possible.

Solution

One could easily approach the problem with pipeline primitive pattern (because of sequential chain of dependence), beginning with one processor at start and slowly filling more processors with subsequent updates. The overhead of waiting for the pipeline to fill up might not be affordable in many situations, especially for large number of processors.

Exploiting the local starting point will be beneficial to attain full parallelism from start. Namely, we will have as many local starting points as possible in the sequential topology, and feed its results as an update for its neighbors. Since one data element is dependent on its neighbor, the two adjacent elements cannot locally start side by side. Natural solution is to have local start at *every other* data elements—the odd elements. The result that comes out from the odd elements will be fed to the elements that are between the odd elements—the even elements. As one might have noticed, the results from the even elements now will feed the odd elements and the cycle will go on until a termination condition has met. This will allow full concurrent communication and computation possible while satisfying the dependency constraints of the elements.

As in pipeline, the update process will result by phases. We can think of the odd-even alternation as two split stages within a phase. Synchronization has to follow between stages, but message-passing calls will naturally become a barrier between the stages.

Benefit

1. Sequential chain of dependence is parallelized without consuming time to pipeline.
2. Deadlock can be prevented in message passing environment. [DEMO]
3. Neighbor-to-neighbor update pattern is preserved.

Difficulties

1. Must identify in the dataset if local start is possible and not interfere with the outcome of the dataset (convergence, numerical stability, algorithmic violation). The local start availability is only present in certain classes of problems.
2. If more than one dimension, odd-even pairs needs to be carefully alternated.

Related Pattern

Pipeline, Geometric Data

Example

Gauss-Seidel for Laplace Equation (Red-Black)
Odd-Even Transposition Sort