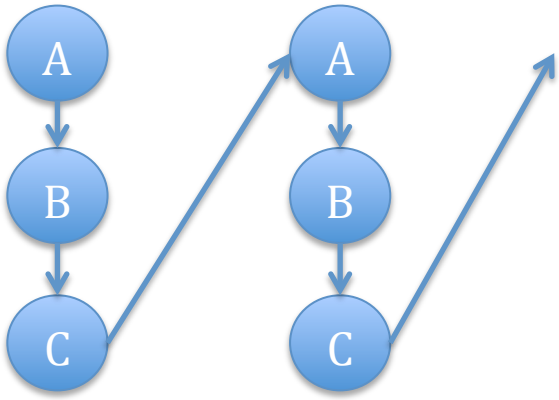
Marc Leef
Allen Malony
CIS 410
4/21/14

Homework 2

Dependencies:



1) Data is dependent on previous tasks.
2) The loop can be parallelized in chunks of four. More specifically, i-4 must be calculated by the time the loop reaches i, as a[i][j] becomes equal to a[i-4][j]. Therefore, 4 operations can be performed in parallel before moving on to the next 4 and so on.
3) #pragma omp parallel for schedule(static, 4)

```
1 for i = 5 to 100 do
2       for j = i-2 to i do
3               a[i][j] = a[i - 4][j];
4       end
5 end
```

Patterns:

1) We would choose stencil. The stencil pattern allows us to use the elemental function of our map pattern to access single elements and their neighbors. Many stencils operating on various parts of the input data at once could produce outputs in parallel because of the fixed offset nature of the output's neighbors.
2) We would use gather. Gather reads a collection of data from a different data collection, based on a number of indices, which is exactly what we need in this context. The output array is shaped by the inputted indices and the operations performed combine elements of map and serial random read operations.

Map Pattern:

1) Because loops are so prevalent in algorithm design, it makes sense to try and optimize their performance – especially if tasks performed in the loop body are independent from one another. The map pattern solves this perfectly, by replicating a function over every element of a set in parallel. This pattern achieves greater performance over the slower iterative approach.

2) Yes it is possible, pseudo code:

```
#pragma omp parallel for (or any other loop parallelizing approach)
for(int i = 0; i < A.rows; i++) {
        for(int j = 0; j < A.cols; j++) {
                C[i][0] +=  A[i][j] * B[j][0]
        }
}
```

3) To avoid recalculating the same data multiple times and to improve efficiency. The fusing of the maps reduces patterns to increase algorithm effectiveness.

Collectives Pattern:

1) Every word will get its own vector. The merge operation compares the words to produce a sorted sub list, with the empty vector just being a blank word. This approach allows a reduction to be performed, similar to a merge sort, to alphabetize the words in parallel.
2) The work needs to be broken down into smaller chunks that can be executed serially. Without this tiling process, there won't be any sub problems to solve and eventually join together, an important characteristic to patterns such as reduce and scan.
3) The gradual accumulation of miniscule floating-point errors can result in quite significant errors in the final product. As a result of the structure of patterns such as scan and reduce, small errors can be compounded and produce significantly deviated results.