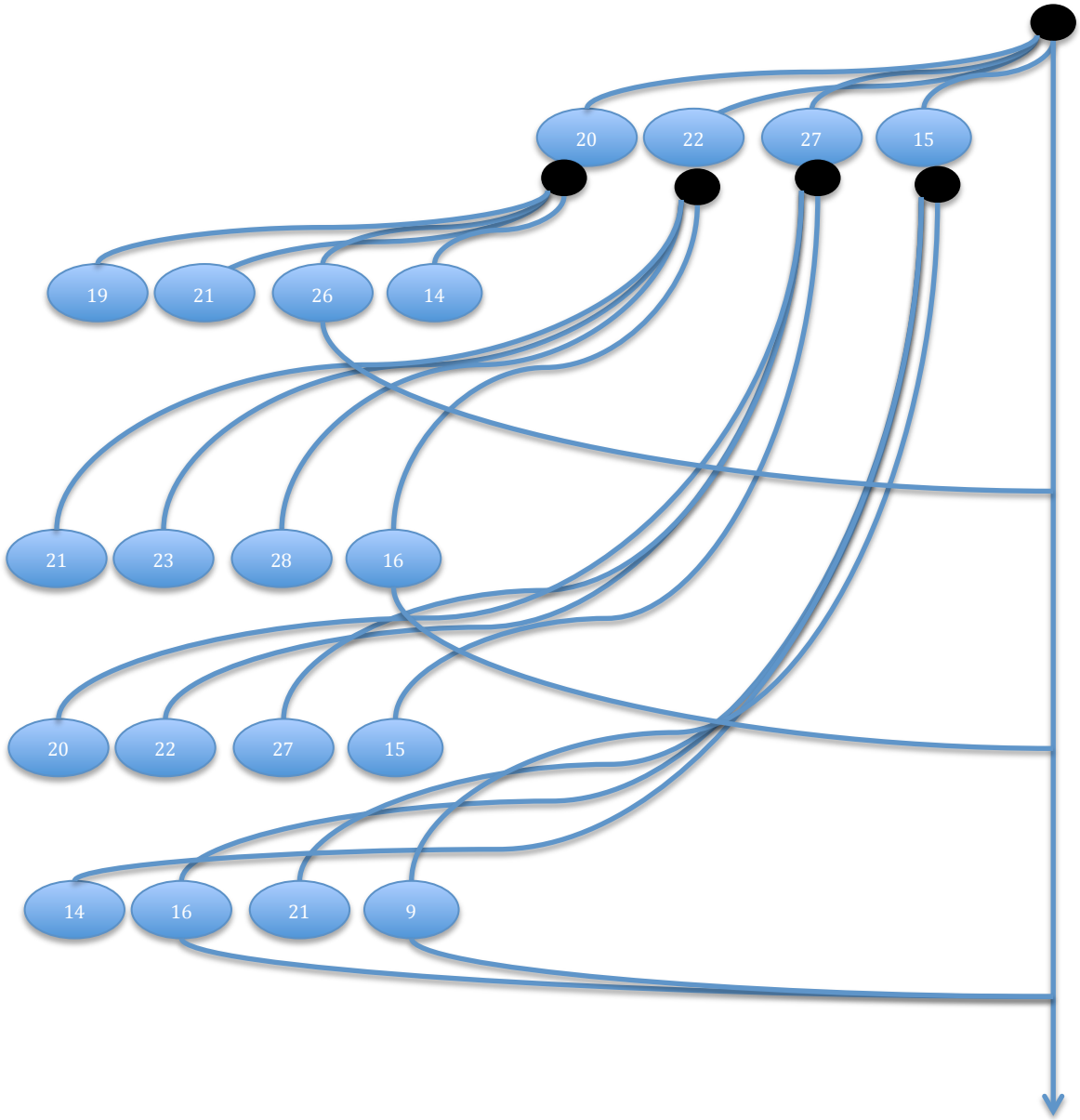
Marc Leef
Allen Malony
CIS 410
5/6/14

Homework 4

Fork-Join Pattern:

1) The fork-join pattern is useful for parallel computing as it allows child threads to be spawned that perform some separate piece of computation before rejoining with its parent. As a pattern, fork-join acts as a concurrency control mechanism that can greatly aid parallel computation. Fork allows us to create child threads while join allows the children to rejoin with their parents upon completion.
2) a) Partial DAG of FloodFill(21, white, blue):



b) The amount of concurrency is directly proportional to the number of times the operation can be forked. For this operation, because each method call can potentially produce 3 forks plus the parent process, the amount of concurrency is equal to the size of the grid divided by 4.
c) Yes, because multiple workers could be focusing on the same cells simultaneously, situations may arise where the color characteristics of cells are being modified without updating workers' knowledge of these changes.

3) Parallel slack is the amount of extra parallelism available above the minimum necessary to use the parallel resources of the hardware. Decomposing a problem more and more creates increased amounts of parallel slack, and as the fork join pattern involves mainly decomposition, it is important to consider when solving problems using fork join.

Pipeline Pattern:

1) When the number of data items surpasses the number of serial tasks, the number of tasks becomes the bottleneck and limits performance of the pipeline by decreasing the degree of available parallelism.
2) Stage bound workers: Each worker takes a waiting item, performs computation on said item, and then passes it along. Straightforward strategy but there is no data locality. Item bound workers: Worker is responsible for an item throughout the entire pipeline, more complicated to implement but provides for much better data locality. Hybrid workers: Workers begin as item bound but can change their behavior depending on whether or not the item is ready to be worked on. Hybrid seems like the best strategy for most situations, if no data locality is required to solve the problem then stage bound workers are the best choice because of their relative ease of implementation.