debugging on the workstation rather on the parallel machine. The hierarchical nature of flowgraphs allows us to build applications incrementally and from the bottom–up in the following series of steps (See Figure 3):

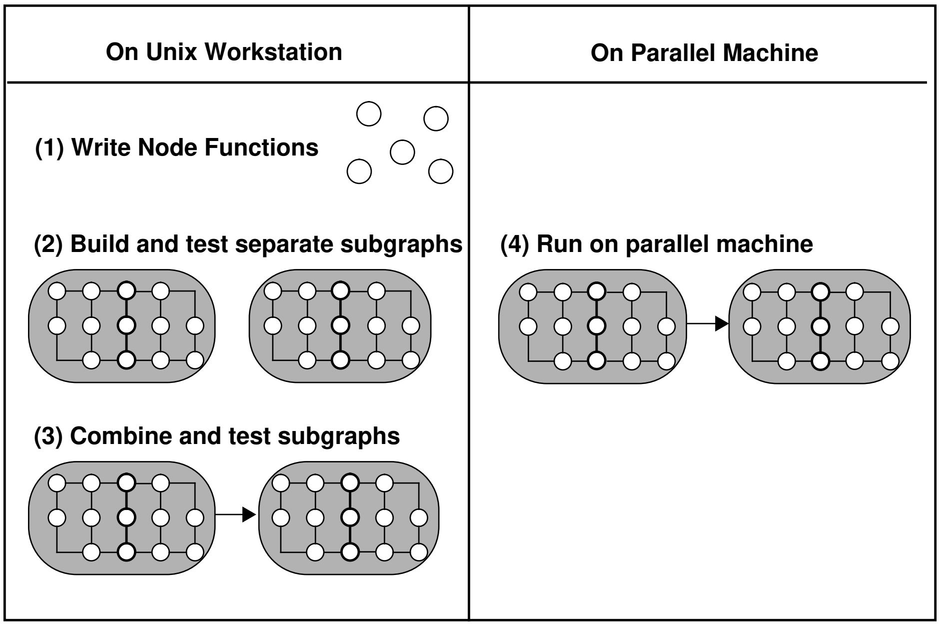

Figure 3: Typical process of programming Assign applications. Notice that the system is constructed incrementally and that most of the work is done on the Unix workstation.

Step 1: Write node functions. These are sequential C functions, instances of which will correspond to nodes in the flowgraph. Each of these functions can be tested independently on the Unix workstation.

Step 2: Generate and test subgraphs. ASSIGN flowgraphs are hierarchical, meaning that graphs can be formed by composing smaller subgraphs. Typically, we independently generate and test a collection of smaller subgraphs on the Unix workstation, using the node functions developed in the previous step. As we shall see, flowgraphs are generated by writing and running a C program called a *graph generator*.

Step 3: Combine and test subgraphs. Once we are satisfied that each of the subgraphs is producing correct results, we combine them, *without change*, to form the final graph. The testing of the final graph can be performed entirely on the Unix workstation.

Step 4: Run final graph on parallel machine. When we are satisfied that the final graph is producing correct results, we can recompile the graph, *without change*, and then run it on the parallel machine. It is important to note that most of the program development is done on the Unix workstation and not on the parallel machine.