What is claimed is:

1. A method for compiling a graph-based program specification to execute on a computing system that supports concurrent execution of operations, the method including:

receiving the graph-based program specification, the graph-based program specification including a graph that includes:

a plurality of components, each corresponding to at least one operation and including one or more ports, the ports including scalar data ports for sending or receiving a single data element at a time and collection data ports for sending or receiving a collection of multiple data elements at a time; and

a plurality of directed links, each directed link connecting an upstream component of the plurality of components to a downstream component of the plurality of components; and

processing the graph-based program specification to generate processed code representing one or more groups of operations, the processing including:

identifying a first boundary of a first group of the operations including identifying a first entry point to the first group of operations and a first exit point from the first group of the operations based on a first identification of one or more links connecting a collection data port of a component to a scalar data port of a component of the plurality of components;

identifying a possible level of concurrency in the first group of the operations based at least in part on a topology of the graph, such that multiple operations in the first group are not prevented by the topology of the graph from executing concurrently;

analyzing at least some of the operations in the first group to determine runtime characteristics associated with the analyzed operations, the analyzing including identifying a second boundary of an iterating set of operations in the first group of operations including identifying a second entry point to the iterating set of operations and a second exit point from the iterating set of operations based on a second identification of one or more links connecting a collection data port of a component to a scalar data port of a component of the plurality of components; and

generating processed code for executing the operations in the first group, where the processed code enforces a reduced level of concurrency in the first group, lower than the identified possible level of concurrency, based at least in part on the determined runtime characteristics.

2. The method of claim 1 wherein analyzing at least some of the operations in the first group to determine runtime characteristics associated with the analyzed operations includes, for each of the analyzed operations, determining if that analyzed operation is a latent operation that potentially performs a computation over a period of time that is larger than a first threshold or that potentially waits for a response for a period of time that is larger than the first threshold.

3. The method of claim 2 wherein analyzing at least some of the operations in the first group to determine runtime characteristics associated with the analyzed operations includes, for each of the analyzed operation, determining if that analyzed operation is a member of the iterating set, wherein the iterating set includes a set of one or more operations that executes multiple times in response to a single initiation of execution of the first group.

4. The method of claim 3 wherein the first threshold is defined as a first period of time greater than a second period of time required to spawn a task for executing one or more operations.

5. The method of claim 3 wherein the processed code enforces a reduced level of concurrency in the first group, lower than the identified possible level of concurrency, if neither of these conditions is met:

(1) at least two operations in the first group that are not prevented by the topology of the graph from executing concurrently were both determined to be latent operations, or

(2) at least one operation in the first group was determined to be a member of the iterating set of one or more operations.

6. The method of claim 5 wherein the processed code enforces a reduced level of concurrency in the first group, lower than the identified possible level of concurrency, if the following additional condition is also not met: (3) a configuration of two or more operations in the first group imply an existence of the iterating set.

7. The method of claim 1 wherein enforcing a reduced level of concurrency in the first group, lower than the identified possible level of concurrency, includes enforcing serial execution of all of the operations in the first group.

8. The method of claim 1 wherein one or more of the directed links indicate data flow between an upstream component and a downstream component.

9. The method of claim 8 wherein one or more of the directed links indicate control flow between an upstream component and a downstream component.

10. The method of claim 1 wherein one or more of the directed links indicate control flow between an upstream component and a downstream component.

11. The method of claim 1 wherein processing the graph-based program specification to generate processed code representing one or more groups of operations further includes:

generating ordering information that specifies at least a partial ordering among the first group of the operations, where the ordering information is based at least in part on the topology of the graph; and

wherein identifying the possible level of concurrency in the first group includes identifying a level of concurrency allowed by the partial ordering such that multiple operations in the first group are not prevented by the partial ordering from executing concurrently.

12. The method of claim 11 wherein generating the ordering information includes topologically sorting the first group of operations.

13. The method of claim 1 wherein identifying the first entry point includes identifying a transition from a collection output port of a first component associated with a first operation to a scalar input port of a second component associated with a second operation included in the first group of operations and identifying the first exit point includes identifying a transition from a scalar output port of a third component associated with a third operation included in the first group of operations to a collection input port of a fourth component associated with a fourth operation.

14. The method of claim 1 wherein identifying the second entry point includes identifying a transition from a collection output port of a first component associated with a first operation to a scalar input port of a second component associated with a second operation included in the iterating set of operations and identifying the first exit point includes identifying a transition from a scalar output port of a third component associated with a third operation included in the iterating set of operations to a collection input port of a fourth component associated with a fourth operation.

15. Software stored in a non-transitory form on a computer-readable medium, for compiling a graph-based program specification to execute on a computing system that supports concurrent execution of operations, the software including instructions for causing the computing system to perform the method of any one of claims 1 to 14.

16. A computing system for compiling a graph-based program specification, the computing system including:
an input device or port configured to receive the graph-based program specification; and
at least one hardware processor configured to perform the method of any one of claims 1 to 14.

17. A computing system for compiling a graph-based program specification, the computing system including:
means for receiving the graph-based program specification, the graph-based program specification including a graph that includes:
a plurality of components, each corresponding to an operation and including one or more ports, the ports including scalar data ports for sending or receiving a single data element at a time and collection data ports for sending or receiving a collection of multiple data elements at a time; and
a plurality of directed links, each directed link connecting an upstream component of the plurality of components to a downstream component of the plurality of components; and
means for processing the graph-based program specification to generate processed code representing one or more groups of operations, the processing including:

identifying a first boundary of a first group of the operations including identifying a first entry point to the first group of operations and a first exit point from the first group of the operations based on a first identification of one or more links connecting a collection data port of a component to a scalar data port of a component of the plurality of components;

identifying a possible level of concurrency in the first group of the operations based at least in part on a topology of the graph, such that multiple operations in the first group are not prevented by the topology of the graph from executing concurrently;

analyzing at least some of the operations in the first group to determine runtime characteristics associated with the analyzed operations, the analyzing including identifying a second boundary of an iterating set of operations in the first group of operations including identifying a second entry point to the iterating set of operations and a second exit point from the iterating set of operations based on a second identification of one or more links connecting a collection data port of a component to a scalar data port of a component of the plurality of components; and

generating processed code for executing the operations in the first group, where the processed code enforces a reduced level of concurrency in the first group, lower than the identified possible level of concurrency, based at least in part on the determined runtime characteristics.