

What is claimed is:

1. A computing system including:

a storage system storing one or more graph-based program specifications, at least a first graph-based program specification including a plurality of components corresponding to tasks and directed links between ports of the components, the first graph-based program specification including:

- (1) a first type of link configuration defined by respective output and input ports of linked components, and
- (2) a second type of link configuration defined by respective output and input ports of linked components, where the second type of link configuration is different from the first type of link configuration;

a compiler configured to generate a target program specification from the first graph-based program specification, where the compiler recognizes different types of link configurations and provides in the target program specification one or more occurrences of a target primitive for executing a function for each occurrence of a data element flowing over a link having the second type of link configuration; and

one or more computing nodes, each including at least one processor, with at least a first of the computing nodes being configured to:

initiate execution of the target program specification, and

determine at runtime, for one or more components associated with the one or more occurrences of the target primitive, at least one of: (1) an order in which instances of tasks corresponding to the components are to be invoked, or (2) a computing node on which instances of tasks corresponding to the components are to be executed.

2. The computing system of claim 1, wherein the first computing node is further configured to:

for a first link, having the first type of link configuration, from a port of a first component in the first graph-based program specification to a port of a second component in the first graph-based program specification, invoke tasks corresponding to the first and second components serially, and

for a second link, having the second type of link configuration, from a port of a third component in the first graph-based program specification to a port of a fourth component in the first graph-based program specification, invoke multiple instances of a task corresponding to the fourth component for processing different respective subsets of data elements provided by the third component.

3. The computing system of claim 2, wherein the first computing node is further configured to assign at least some of the multiple instances to be executed on different ones of multiple computing nodes.

4. The computing system of claim 3, wherein the target primitive identifies a function corresponding to a set of one or more tasks and identifies a collection of data elements corresponding to a link having the second type of link configuration.

5. The computing system of claim 4, wherein at least some of the multiple instances are applied to different respective data elements in the collection identified by the target primitive.

6. The computing system of claim 5, wherein the number of instances invoked is equal to the number of data elements in the collection identified by the target primitive, and each instance is applied to a different respective data element.

7. The computing system of claim 5, wherein the number of instances invoked is determined independently from any information in the first graph-based program specification.

8. The computing system of claim 7, wherein the number of instances invoked is at least as large as a number of computing nodes at which data referenced by the first graph-based program specification is stored.

9. The computing system of claim 5, wherein applying the function identified by the target primitive to a data element includes performing the set of one or more tasks corresponding to the function using the data element as a data source for the one or more tasks.

10. The computing system of claim 3, wherein assigning at least some of the multiple instances to be executed on different computing nodes includes determining, at runtime, a number of computing nodes on which the multiple instances are to be executed.

11. The computing system of claim 1, wherein a link having the first type of link configuration indicates to the compiler that a single data element is to be passed between the linked components, and a link having the second type of link configuration indicates to the compiler that multiple data elements are to be passed between the linked components.

12. The computing system of claim 1, wherein the first graph-based program specification includes a third type of link configuration of a link defined by respective input and output ports of linked components that indicates to the compiler that there is to be serial execution of the linked components, and indicates to the compiler that a single data element is to be passed between tasks of the linked components.

13. The computing system of claim 1, wherein the compiler is configured to identify one or more sets of one or more tasks corresponding to respective components.

14. The computing system of claim 13, wherein each set of one or more tasks has a single collection of data elements as a data source that provides different data elements to different instances of the set of one or more tasks.

15. The computing system of claim 13, wherein each set of one or more tasks corresponds to at least one directed acyclic graph of one or more nodes, where each node of the graph corresponds to a task or an identified set and each directed edge of the graph corresponds to a link having the first type of link configuration.

16. The computing system of claim 15, wherein the directed acyclic graph of one or more nodes has a single root node.

17. The computing system of claim 1, wherein each computing node is configured to execute at least one scheduler for spawning processes to execute tasks in a corresponding queue.

18. The computing system of claim 17, wherein the first computing node is configured to move one or more tasks into a queue of a first scheduler executed by the first computing node from a queue of a second scheduler executed by the first computing node.

19. The computing system of claim 17, wherein the first computing node is configured to move one or more tasks from a queue of a scheduler executed by the first computing node into a queue of a scheduler executed by a second computing node.

20. A computing system including:

means for storing one or more graph-based program specifications, at least a first graph-based program specification including a plurality of components corresponding to tasks and directed links between ports of the components, the first graph-based program specification including:

- (1) a first type of link configuration defined by respective output and input ports of linked components, and
- (2) a second type of link configuration defined by respective output and input ports of linked components, where the second type of link configuration is different from the first type of link configuration;

means for generating a target program specification from the first graph-based program specification, and recognizing different types of link configurations and providing in the target program specification one or more occurrences of a target primitive for executing a function for each occurrence of a data element flowing over a link having the second type of link configuration; and

one or more computing nodes, each including at least one processor, with at least a first of the computing nodes being configured to:

initiate execution of the target program specification, and

determine at runtime, for one or more components associated with the one or more occurrences of the target primitive, at least one of: (1) an order in which instances of tasks corresponding to the components are to be invoked, or (2) a computing node on which instances of tasks corresponding to the components are to be executed.

21. A method including:

storing one or more graph-based program specifications, at least a first graph-based program specification including a plurality of components corresponding to tasks and directed links between ports of the components, the first graph-based program specification including:

(1) a first type of link configuration defined by respective output and input ports of linked components, and

(2) a second type of link configuration defined by respective output and input ports of linked components, where the second type of link configuration is different from the first type of link configuration;

generating a target program specification from the first graph-based program specification, and recognizing different types of link configurations and providing in the target program specification one or more occurrences of a target primitive for executing a function for each occurrence of a data element flowing over a link having the second type of link configuration; and

using at least one computing node,

initiating execution of the target program specification, and

determining at runtime, for one or more components associated with the one or more occurrences of the target primitive, at least one of: (1) an order in which instances of tasks corresponding to the components are to be invoked, or (2) a computing node on which instances of tasks corresponding to the components are to be executed.

22. Software stored on a non-transitory medium, the software including instructions for causing a computing system to:

store one or more graph-based program specifications, at least a first graph-based program specification including a plurality of components corresponding to tasks and directed links between ports of the components, the first graph-based program specification including:

- (1) a first type of link configuration defined by respective output and input ports of linked components, and
- (2) a second type of link configuration defined by respective output and input ports of linked components, where the second type of link configuration is different from the first type of link configuration;

generate a target program specification from the first graph-based program specification, and recognize different types of link configurations and provide in the target program specification one or more occurrences of a target primitive for executing a function for each occurrence of a data element flowing over a link having the second type of link configuration; and

using at least one computing node,

initiate execution of the target program specification, and

determine at runtime, for one or more components associated with the one or more occurrences of the target primitive, at least one of: (1) an order in which instances of tasks corresponding to the components are to be invoked, or (2) a computing node on which instances of tasks corresponding to the components are to be executed.

23. The computing system of claim 1, wherein the compiler provides an occurrence of the target primitive for each link having the second type of link configuration, but not for any link having the first type of link configuration.

24. The method of claim 21, wherein the first computing node is further configured to:

for a first link, having the first type of link configuration, from a port of a first component in the first graph-based program specification to a port of a second component in the first graph-based program specification, invoke tasks corresponding to the first and second components serially, and

for a second link, having the second type of link configuration, from a port of a third component in the first graph-based program specification to a port of a fourth component in the first graph-based program specification, invoke multiple instances of a task corresponding to the fourth component for processing different respective subsets of data elements provided by the third component.

25. The method of claim 24, wherein the first computing node is further configured to assign at least some of the multiple instances to be executed on different ones of multiple computing nodes.

26. The method of claim 25, wherein the target primitive identifies a function corresponding to a set of one or more tasks and identifies a collection of data elements corresponding to a link having the second type of link configuration.

27. The method of claim 26, wherein at least some of the multiple instances are applied to different respective data elements in the collection identified by the target primitive.

28. The method of claim 27, wherein the number of instances invoked is equal to the number of data elements in the collection identified by the target primitive, and each instance is applied to a different respective data element.

29. The method of claim 27, wherein the number of instances invoked is determined independently from any information in the first graph-based program specification.

30. The method of claim 29, wherein the number of instances invoked is at least as large as a number of computing nodes at which data referenced by the first graph-based program specification is stored.

31. The method of claim 27, wherein applying the function identified by the target primitive to a data element includes performing the set of one or more tasks corresponding to the function using the data element as a data source for the one or more tasks.

32. The method of claim 25, wherein assigning at least some of the multiple instances to be executed on different computing nodes includes determining, at runtime, a number of computing nodes on which the multiple instances are to be executed.

33. The method of claim 21, wherein a link having the first type of link configuration indicates to the compiler that a single data element is to be passed between the linked components, and a link having the second type of link configuration indicates to the compiler that multiple data elements are to be passed between the linked components.

34. The method of claim 21, wherein the first graph-based program specification includes a third type of link configuration of a link defined by respective input and output ports of linked components that indicates to the compiler that there is to be serial execution of the linked components, and indicates to the compiler that a single data element is to be passed between tasks of the linked components.

35. The method of claim 21, wherein the compiler is configured to identify one or more sets of one or more tasks corresponding to respective components.

36. The method of claim 35, wherein each set of one or more tasks has a single collection of data elements as a data source that provides different data elements to different instances of the set of one or more tasks.

37. The method of claim 35, wherein each set of one or more tasks corresponds to at least one directed acyclic graph of one or more nodes, where each node of the graph corresponds to a task or an identified set and each directed edge of the graph corresponds to a link having the first type of link configuration.

38. The method of claim 37, wherein the directed acyclic graph of one or more nodes has a single root node.

39. The method of claim 21, wherein each computing node is configured to execute at least one scheduler for spawning processes to execute tasks in a corresponding queue.

40. The method of claim 39, wherein the first computing node is configured to move one or more tasks into a queue of a first scheduler executed by the first computing node from a queue of a second scheduler executed by the first computing node.

41. The method of claim 39, wherein the first computing node is configured to move one or more tasks from a queue of a scheduler executed by the first computing node into a queue of a scheduler executed by a second computing node.

42. The method of claim 21, wherein the compiler provides an occurrence of the target primitive for each link having the second type of link configuration, but not for any link having the first type of link configuration.

43. The software of claim 22, wherein the first computing node is further configured to:
for a first link, having the first type of link configuration, from a port of a first component in the first graph-based program specification to a port of a second component in the first graph-based program specification, invoke tasks corresponding to the first and second components serially, and
for a second link, having the second type of link configuration, from a port of a third component in the first graph-based program specification to a port of a fourth component in the first graph-based program specification, invoke multiple instances of a task corresponding to the fourth component for processing different respective subsets of data elements provided by the third component.

44. The software of claim 43, wherein the first computing node is further configured to assign at least some of the multiple instances to be executed on different ones of multiple computing nodes.

45. The software of claim 44, wherein the target primitive identifies a function corresponding to a set of one or more tasks and identifies a collection of data elements corresponding to a link having the second type of link configuration.

46. The software of claim 45, wherein at least some of the multiple instances are applied to different respective data elements in the collection identified by the target primitive.

47. The software of claim 46, wherein the number of instances invoked is equal to the number of data elements in the collection identified by the target primitive, and each instance is applied to a different respective data element.

48. The software of claim 46, wherein the number of instances invoked is determined independently from any information in the first graph-based program specification.

49. The software of claim 48, wherein the number of instances invoked is at least as large as a number of computing nodes at which data referenced by the first graph-based program specification is stored.

50. The software of claim 46, wherein applying the function identified by the target primitive to a data element includes performing the set of one or more tasks corresponding to the function using the data element as a data source for the one or more tasks.

51. The software of claim 44, wherein assigning at least some of the multiple instances to be executed on different computing nodes includes determining, at runtime, a number of computing nodes on which the multiple instances are to be executed.

52. The software of claim 22, wherein a link having the first type of link configuration indicates to the compiler that a single data element is to be passed between the linked components, and a link having the second type of link configuration indicates to the compiler that multiple data elements are to be passed between the linked components.

53. The software of claim 22, wherein the first graph-based program specification includes a third type of link configuration of a link defined by respective input and output ports of linked components that indicates to the compiler that there is to be serial execution of the linked components, and indicates to the compiler that a single data element is to be passed between tasks of the linked components.

54. The software of claim 22, wherein the compiler is configured to identify one or more sets of one or more tasks corresponding to respective components.

55. The software of claim 54, wherein each set of one or more tasks has a single collection of data elements as a data source that provides different data elements to different instances of the set of one or more tasks.

56. The software of claim 54, wherein each set of one or more tasks corresponds to at least one directed acyclic graph of one or more nodes, where each node of the graph corresponds to a task or an identified set and each directed edge of the graph corresponds to a link having the first type of link configuration.

57. The software of claim 56, wherein the directed acyclic graph of one or more nodes has a single root node.

58. The software of claim 22, wherein each computing node is configured to execute at least one scheduler for spawning processes to execute tasks in a corresponding queue.

59. The software of claim 58, wherein the first computing node is configured to move one or more tasks into a queue of a first scheduler executed by the first computing node from a queue of a second scheduler executed by the first computing node.

60. The software of claim 58, wherein the first computing node is configured to move one or more tasks from a queue of a scheduler executed by the first computing node into a queue of a scheduler executed by a second computing node.

61. The software of claim 22, wherein the compiler provides an occurrence of the target primitive for each link having the second type of link configuration, but not for any link having the first type of link configuration.