

## Claims

1. A method for executing an application expeditiously on at least one computer processor, the method comprising:

simultaneously executing a plurality of instances of the application on the at least one computer processor, each instance being compiled according to a respective compiler option that is different from respective compiler options used to compile all other instances from the plurality of instances; and

for each one of a first plurality of results to be produced by the application:

monitoring, corresponding to the result, a respective output generated by each instance; and

labelling from the monitored outputs the output occurring earliest as output of the application corresponding to the result, and labelling all other outputs as duplicates, thereby speeding up a computing performance of the at least one computer processor.

2. The method of claim 1, wherein:

the plurality of instances comprises a first instance and a second instance;

the compiler option for the first instance comprises ahead-of-time (AIT) compilation; and the compiler option for the second instance comprises just-in-time (JIT) compilation, wherein JIT compilation is based on, at least in part, runtime information obtained from at least one prior execution of the second instance.

3. The method of claim 1, wherein:

the plurality of instances comprises a first instance; and

the compiler option for the first instance is selected from a group consisting of unconstrained memory usage, minimization of memory usage, maximization of concurrent operations, and constrained concurrency of operations.

4. The method of claim 1, further comprising:

labelling one instance from the plurality of instances as a primary instance;

labelling all other instances as secondary instances; and

for each result from a second plurality of results to be produced by the application, suppressing, corresponding to the result, respective outputs from the secondary instances.

5. The method of claim 4, further comprising:

identifying a failure of the primary instance after producing  $k$  results from the second plurality of results,  $k > 1$ ;

labelling the primary instance as a failed instance;

selecting a secondary instance that executed logic associated with computation of each of the  $k$  results; and

re-labelling the selected secondary instances as the primary instance, whereby the relabelled primary instance produces  $(k+1)$ -th result.

6. The method of claim 1, wherein at least a portion of source code of the software application is specified using a programming language that is at least partially interpreted.

7. A system for executing an application expeditiously on at least one computer processor, comprising:

a first processor; and

a first memory in electrical communication with the first processor, the first memory comprising instructions which, when executed by a processing unit comprising at least one of the first processor and a second processor, program the processing unit to:

simultaneously execute a plurality of instances of the application on the at least one computer processor, each instance being compiled according to a respective compiler option that is different from respective compiler options used to compile all other instances from the plurality of instances; and

for each one of a first plurality of results to be produced by the application:

monitor, corresponding to the result, a respective output generated by each instance; and

label from the monitored outputs the output occurring earliest as output of the application corresponding to the result, and label all other outputs as duplicates, thereby speeding up a computing performance of the at least one computer processor.

8. The system of claim 7, wherein:

the plurality of instances comprises a first instance and a second instance;

the compiler option for the first instance comprises ahead-of-time (AIT) compilation; and the compiler option for the second instance comprises just-in-time (JIT) compilation, wherein JIT compilation is based on, at least in part, runtime information obtained from at least one prior execution of the second instance.

9. The system of claim 7, wherein:

the plurality of instances comprises a first instance; and

the compiler option for the first instance is selected from a group consisting of unconstrained memory usage, minimization of memory usage, maximization of concurrent operations, and constrained concurrency of operations.

10. The system of claim 7, wherein the instructions further program the processing unit to: label one instance from the plurality of instances as a primary instance;

label all other instances as secondary instances; and

for each result from a second plurality of results to be produced by the application, suppress, corresponding to the result, respective outputs from the secondary instances.

11. The system of claim 10, wherein the instructions further program the processing unit to: identify a failure of the primary instance after producing  $k$  results from the second plurality of results,  $k > 1$ ;

label the primary instance as a failed instance;

select a secondary instance that executed logic associated with computation of each of the  $k$  results; and

re-label the selected secondary instances as the primary instance, whereby the re-labelled primary instance produces  $(k+1)$ -th result.

12. The system of claim 7, wherein at least a portion of source code of the software application is specified using a programming language that is at least partially interpreted.