'104 Patent, Claim 11

11. An apparatus comprising:

a memory containing intermediate form object code constituted by a set of instructions, certain of said instructions containing one or more symbolic references; and

a processor configured to execute said instructions containing one or more symbolic references by determining a numerical reference corresponding to said symbolic reference, storing said numerical references, and obtaining data in accordance to said numerical references.

'104 Patent, Claim 27

27. A computer-implemented method comprising:

receiving a program with a set of original instructions written in an intermediate form code;

generating a set of new instructions for the program that contain numeric references resulting from invocation of a routine to resolve any symbolic data references in the set of original instructions; and

executing the program using the set of new instructions.

'104 Patent, Claim 29

29. A computer program product containing instructions for causing a computer to perform a method, the method comprising:

receiving a program with a set of original instructions written in an intermediate form code;

generating a set of new instructions for the program that contain numeric references resulting from invocation of a routine to resolve any symbolic data references in the set of original instructions; and

executing the program using the set of new instructions.

'104 Patent, Claim 39

39. A computer-implemented method comprising:

receiving a program formed of instructions written in an intermediate form code compiled from source code;

analyzing each instruction to determine whether it contains a symbolic field reference; and

executing the program by performing an operation identified by each instruction, wherein data from a storage location identified by a numeric reference is thereafter used for the operation when the instruction contains a symbolic field reference, the numeric reference having been resolved from the symbolic field reference.

104 Patent, Claim 40

40. A data processing system, comprising:

a processor; and

a memory comprising a control program for causing the processor to (i) receive a program formed of instructions written in an intermediate form code compiled from source code, (ii) analyze each instruction to determine whether it contains a symbolic field reference, and (iii) execute the program by performing an operation identified by each instruction, wherein data from a storage location identified by a numeric reference is thereafter used for the operation when the instruction contains a symbolic field reference, the numeric reference having been resolved from the symbolic field reference.

'104 Patent, Claim 41

41. A computer program product containing control instructions for causing a computer to perform a method, the method comprising:

receiving a program formed of instructions written in an intermediate form code compiled from source code;

analyzing each instruction to determine whether it contains a symbolic field reference; and

executing the program by performing an operation identified by each instruction, wherein data from a storage location identified by a numeric reference is used thereafter for the operation when the instruction contains a symbolic field reference, the numeric reference having been resolved from the symbolic field reference.

205 Patent, Claim 1

1. In a computer system, a method for increasing the execution speed of virtual machine instructions at runtime, the method comprising:

receiving a first virtual machine instruction;

generating, at runtime, a new virtual machine instruction that represents or references one or more native instructions that can be executed instead of said first virtual machine instruction; and

executing said new virtual machine instruction instead of said first virtual machine instruction.

205 Patent, Claim 2

2. The method of claim 1, further comprising overwriting a selected virtual machine instruction with a new virtual machine instruction, the new virtual machine instruction specifying execution of the at least one native machine instruction.

'720 Patent, Claim 1

1. A system for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:

A processor;

A memory;

a class preloader to obtain a representation of at least one class from a source definition provided as object-oriented program code;

a master runtime system process to interpret and to instantiate the representation as a class definition in a memory space of the master runtime system process;

a runtime environment to clone the memory space as a child runtime system process responsive to a process request and to execute the child runtime system process; and

a copy-on-write process cloning mechanism to instantiate the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process, and to defer copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

720 Patent, Claim 6

6. A system according to claim 1, further comprising:

a process cloning mechanism to instantiate the child runtime system process by copying the memory space of the master runtime system process into a separate memory space for the child runtime system process.

720 Patent, Claim 10

10. A method for dynamic preloading of classes through memory space cloning of a master runtime system process, comprising:

executing a master runtime system process;

obtaining a representation of at least one class from a source definition provided as object-oriented program code;

interpreting and instantiating the representation as a class definition in a memory space of the master runtime system process; and

cloning the memory space as a child runtime system process responsive to a process request and executing the child runtime system process;

wherein cloning the memory space as a child runtime system process involves instantiating the child runtime system process by copying references to the memory space of the master runtime system process into a separate memory space for the child runtime system process; and

wherein copying references to the memory space of the master runtime system process defers copying of the memory space of the master runtime system process until the child runtime system process needs to modify the referenced memory space of the master runtime system process.

720 Patent, Claim 19

19. A computer-readable storage medium holding code for performing the method according to claim 10.

'720 Patent, Claim 21

21. A system according to claim 1, further comprising:

a resource controller to set operating system level resource management parameters on the child runtime system process.

720 Patent, Claim 22

22. A method according to claim 10, further comprising:

setting operating system level resource management parameters on the child runtime system process.

476 Patent, Claim 14

14. A computer-readable medium bearing instructions for providing security, the instructions including instructions for performing the steps of:

detecting when a request for an action is made by a principal;

determining whether said action is authorized based on an association between permissions and a plurality of routines in a calling hierarchy associated with said principal;

wherein each routine of said plurality of routines is associated with a class; and

wherein said association between permissions and said plurality of routines is based on a second association between classes and protection domains.

702 Patent, Claim 1

1. A method of pre-processing class files comprising:

determining plurality of duplicated elements in a plurality of class files;

forming a shared table comprising said plurality of duplicated elements;

removing said duplicated elements from said plurality of class files to obtain a plurality of reduced class files; and

forming a multi-class file comprising said plurality of reduced class files and said shared table.

702 Patent, Claim 6

5. The method of claim 1, wherein said step of determining a plurality of duplicated elements comprises:

determining one or more constants shared between two or more class files.

6. The method of claim 5, wherein said step of forming a shared table comprises:

forming a shared constant table comprising said one or more constants shared between said two or more class files.

Oracle contends missing from Palay

702 Patent, Claim 7

7. A computer program product comprising:

a computer usable medium having computer readable program code embodied therein for pre-processing class files, said computer program product comprising:

computer readable program code configured to cause a computer to determine a plurality of duplicated elements in a plurality of class files;

computer readable program code configured to cause a computer to form a shared table comprising said plurality of duplicated elements;

computer readable program code configured to cause a computer to remove said duplicated elements from said plurality of class files to obtain a plurality of reduced class files; and

computer readable program code configured to cause a computer to form a multi-class file comprising said plurality of reduced class files and said shared table.

702 Patent, Claim 12

11. The computer program product of claim 7, wherein said computer readable program code configured to cause a computer to determine said plurality of duplicated elements comprises:

computer readable program code configured to cause a computer to determine one or more constants shared between two or more class files.

12. The computer program product of claim 11, wherein said computer readable program code configured to cause a computer to form said shared table comprises:

computer readable program code configured to cause a computer to form a shared constant table comprising said one or more constants shared between said two or more class files.

Oracle contends missing from Palay

702 Patent, Claim 13

13. An apparatus comprising:

a processor;

a memory coupled to said processor;

a plurality of class files stored in said memory;

a process executing on said processor, said process configured to form a multi-class file comprising:

a plurality of reduced class files obtained from said plurality of class files by removing one or more elements that are duplicated between two or more of said plurality of class files; and

a shared table comprising said duplicated elements.

702 Patent, Claim 15

15. The apparatus of claim 13, wherein said duplicated elements comprise elements of constant pools of respective class files, said shared table comprising a shared constant pool.

Oracle contends missing from Palay

702 Patent, Claim 16

16. The apparatus of claim 13, further comprising:

a virtual machine having a class loader and a runtime data area, said class loader configured to obtain and load said multi-class file into said runtime data area.

520 Patent, Claim 1

1. A method in a data processing system for statically initializing an array, comprising the steps of:

compiling source code containing the array with static values to generate a class file with a clinit method containing byte codes to statically initialize the array to the static values;

receiving the class file into a preloader;

simulating execution of the byte codes of the clinit method against a memory without executing the byte codes to identify the static initialization of the array by the pre loader;

storing into an output file an instruction requesting the static initialization of the array; and

interpreting the instruction by a virtual machine to perform the static initialization of the array.

520 Patent, Claim 8

6. A method in a data processing system, comprising the steps of:

receiving code to be run on a processing component to perform an operation;

play executing the code without running the code on the processing component to identify the operation if the code were run by the processing component; and

creating an instruction for the processing component to perform the operation.

8. The method of claim 6 wherein the operation statically initializes an array and wherein the play executing step includes the step of:

play executing the code to identify the static initialization of the array.

'520 Patent, Claim 12

12. A data processing system comprising:

a storage device containing:

a program with source code that statically initializes a data structure; and

class files, wherein one of the class files contains a clinit method that statically initializes the data structure;

a memory containing:

a compiler for compiling the program and generating the class files; and

a preloader for consolidating the class files, for play executing the clinit method to determine the static initialization the clinit method performs, and for creating an instruction to perform the static initialization; and

a processor for running the compiler and the preloader.

'520 Patent, Claim 20

18. A computer-readable medium containing instructions for controlling a data processing system to perform a method, comprising the steps of:

receiving code to be run on a processing component to perform an operation;

simulating execution of the code without running the code on the processing component to identify the operation if the code were run by the processing component; and

creating an instruction for the processing component to perform the operation.

20. The computer-readable medium of claim 18 wherein the operation statically initializes an array and wherein the simulating step includes the step of:

simulating execution of the code to identify the static initialization of the array.