(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
8 June 2017 (08.06.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/095372 A1

(51) International Patent Classification:
G06F 12/02 (2006.01)

(21) International Application Number:
PCT/US2015/062970

(22) International Filing Date:
30 November 2015 (30.11.2015)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT PACKARD ENTERPRISE DEVELOPMENT LP** [US/US]; 11445 Compaq Center Drive West, Houston, Texas 77070 (US).

(72) Inventors: **KIRSHENBAUM, Evan R.**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US). **GIDRA, Lokesh**; 1501 Page Mill Rd., Palo Alto, California 94304-1100 (US).

(74) Agents: **D'ADDARIO, Daniel M.** et al.; Hewlett Packard Enterprise, 3404 E. Harmony Road, Mail Stop 79, Fort Collins, Colorado 80528 (US).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU, CZ, DE, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JP, KE, KG, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, ME, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, SM, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SD, SL, ST, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, RU, TJ, TM), European (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to the identity of the inventor (Rule 4.17(i))*

Published:

— *with international search report (Art. 21(3))*

(54) Title: POINTERS IN A MEMORY MANAGED SYSTEM

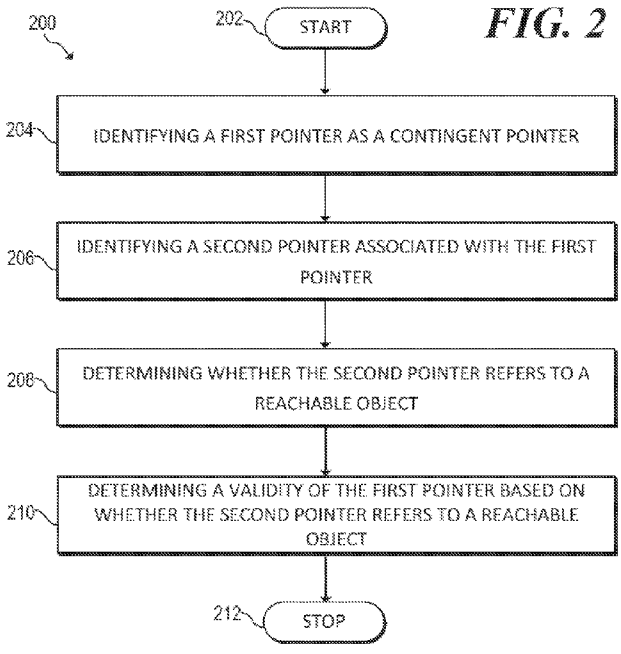

FIG. 2

(57) Abstract: In one example in accordance with the present disclosure, a method for using pointers in a memory managed system is presented. The method may include identifying a first pointer as a contingent pointer and identifying a second pointer associated with the first pointer. The method may also include determining whether the second pointer refers to a reachable object and determining a validity of the first pointer based on whether the second pointer refers to a reachable object.

WO 2017/095372 A1

POINTERS IN A MEMORY MANAGED SYSTEM

Background

[0001] Garbage collection is a type of memory management where a garbage collector reclaims memory occupied by objects that are no longer in use. Garbage collection may be used and/or required by certain programming languages. Although garbage collection may provide significant benefits, garbage collection may require some system overhead and thus may impact performance.

Brief Description of the Drawings

[0002] The following detailed description references the drawings, wherein:

[0003] FIG. 1 is a block diagram of an example computing environment in which using pointers in a memory managed system may be useful;

[0004] FIG. 2 is a flowchart of an example method for using pointers in a memory managed system;

[0005] FIG. 3 is a block diagram of an example system for using pointers in a memory managed system; and

[0006] FIG. 4 is a block diagram of an example system for using pointers in a memory managed system.

Detailed Description

[0007] The data accessed within a computer system may comprise a set of objects (also known as records, structures, instances, or simply data values) stored in the memory or storage of the computer system. Access to objects may be by use of pointers (or references or links) that contain sufficient information to identify the object they refer to. This information may include an address of the object, an offset of a common base address, a key that can be looked up in a table, or other information. These pointers may be stored within processor registers, on a program stack, in global variables, within objects, and elsewhere. In a system that employs garbage collection, some objects may be allocated on a heap managed by a

garbage collector. When an object is allocated, unused space is identified on the heap, and the object is created in that space. The job of the garbage collector is to distinguish between allocated objects that are reachable and those that are unreachable, where an object is considered reachable when it is possible for any program code to make reference to the object. An object is considered unreachable when it is not possible for any program code to make reference to the object.. When objects are determined to be unreachable, the garbage collector declares the space they occupy to be unallocated and returns the memory to the allocator for use in allocating new objects.

[0008] Garbage collection may involve determining a set of root pointers. A root pointer is a pointer to an object stored in a garbage-collected heap where the root pointer itself is observed by a garbage collector in a location other than on the garbage-collected heap. For example, the pointer may be observed in a processor register, on a program or thread stack, in a global or thread-local variable or structure, or on a heap other than the garbage-collected heap).

[0009] The root pointers may be found in processor registers, in global variables, on a program stack, etc. Garbage collection may include determining objects that are reachable starting from the root pointers. An object is considered reachable if it is pointed to by a root pointer or by a pointer contained in some reachable object. Pointers may be considered to be strong pointers or weak pointers. An object referred to by a strong pointer that is a root or is contained in a reachable object is considered reachable. A weak pointer, by contrast, does not affect the reachability of the thing it points to.

[0010] Garbage collection may include collecting memory previously allocated to the objects that are no longer reachable and making the memory available for future allocation. As described above, Garbage collection may consider an object "reachable" if there is any way of reaching it by following pointers from a reachable object, and may collect objects once they become unreachable. But it is frequently the case that objects remain strictly reachable even though the logic of the program dictates that no future attempt will ever be made to reach them, and so the correctness of the program would not be compromised if the objects were collected. But these values, however, may continue to remain reachable, and in turn make other objects reachable, even after there is no longer any way to reach the objects

other than by these value chains. Programs may expend significant effort to identify and remove pointers that are no longer needed simply to facilitate garbage collection or they may tolerate giving up the ability to use the memory taken up unnecessarily by such objects.

[0011] Example systems for using pointers in a memory managed system use contingent pointers to allow the reachability of one object to be dependent on the reachability of another. A contingent pointer is a pointer whose reachability is dependent on the reachability of another pointer. In other words, by declaring one pointer as being contingent on a second pointer the garbage collector itself, with no knowledge of the program semantics, may determine the reachability of the pointer. This reduced overheard may allow more system resources to be allocated to do useful work rather than cleanup and that actually unreachable memory may be reclaimed sooner.

[0012] FIG. 1 is a block diagram of an example system 100 for using pointers in a memory managed system. System 100 may include a processor 102 and a memory 104 that may be coupled to each other through communication link (e.g., a bus).

[0013] Processor 102 may include a Central Processing Unit (CPU), a Graphics Processing Unit (GPU), or another suitable processor. In some examples, memory 104 stores machine readable instructions executed by processor 102 for operating system 100 as well as data used by programs executed by processor 102. Memory 104 may include any suitable combination of volatile and/or non-volatile memory, such as combinations of Random Access Memory (RAM), Read-Only Memory (ROM), memristor memory, spin-transfer torque memory, flash memory, and/or other suitable memory. Memory 104 may also include storage on non-volatile storage devices such as disks or solid-state devices (SSD). Processor 102 may include multiple cores. Processor 102 may also comprise multiple processors.

[0014] Memory 104 stores instructions to be executed by processor 102 including instructions for a contingent pointer identifier 110, an associated pointer identifier 112, a reliability determiner 114, a validity determiner 116, a null pointer identifier 118, an invalidity determiner 120, a location modifier 122, a request handler 124, and/or other components. According to various implementations, system 100 for using pointers in a memory managed system may be implemented in hardware and/or a combination of hardware and programming that configures hardware. Furthermore, in

FIG. 1 and other Figures described herein, different numbers of components or entities than depicted may be used.

[0015] Processor 102 may execute instructions of contingent pointer identifier 110 to identify a first pointer as a contingent pointer. In the system 100 for using pointers in a memory managed system, each pointer into the garbage collector (GC) managed space may be marked as being strong, weak, or contingent. In one example, each GC pointer may point to 8-byte aligned data, and at least one of the low-order three bits may be used to specify the type of pointer. The different types of pointers may also be differentiated at the language level. For example, in the C++ programming language, a strong pointer may be referred to as *gc_ptr<T>*, a weak pointer may be referred to as *weak_gc_ptr<T>*, and a contingent pointer may be referred to as *contingent_gc_ptr<T,X>*, where *T* is the type of the object being pointed to and *X* is the type of the object whose lifetime controls the contingent pointer. Contingent pointer identifier 110 may identify a first pointer as a contingent pointer during a tracing phase of garbage collection. Contingent pointer identifier 110 may identify an object referred to by the first pointer as reachable based on a validity of the first pointer (e.g., as discussed herein with respect to validity determiner 116). A contingent pointer may be considered valid if the associated pointer points to a reachable object. A contingent pointer determined to be valid may be treated as referring to a reachable object. When multiple pointers are associated with the contingent pointer, the first pointer may be considered valid based on the number of associated pointers that point to reachable objects when compared to a threshold (e.g., all of the associated pointers point to reachable objects, any of the associated pointers point to reachable objects), where the threshold may be specified by encoding (e.g., as discussed below with respect to associated pointer identifier 112).

[0016] Processor 102 may execute instructions of associated pointer identifier 112 to identify a pointer associated with the first pointer. The pointer may be, for example, a weak pointer, a strong pointer, a contingent pointer, etc. Associated pointer identifier 112 may identify multiple pointers associated with the first pointer, such as a second pointer, a third pointer, etc. One or more of the multiple pointers may be a strong pointer, a weak pointer, etc. In one example, the weak pointer may be the word immediately preceding the contingent pointer, although in other examples the weak pointer may be located elsewhere. As used herein, the term "word" refers

to a fixed-sized piece of data, for example 64 bits. In one example, a first bit of the low-order three bits may be used to specify a pointer as a contingent pointer and, when that bit is set, the remaining two bits may be used to specify the number and location of associated weak pointers relative to the address containing the contingent pointer or relative to some other address. For example, if read as a binary number, the remaining two bits allow encoding four possible values, and each value can encode a different number and/or relative location of associated weak pointers. As one example, a value of zero can indicate a single weak pointer in the word immediately preceding the contingent pointer, a value of one can indicate a single weak pointer two words preceding the contingent pointer, a value of two can indicate two weak pointers in the two words preceding the contingent pointer, and a value of three can indicate three weak pointers in the three words preceding the contingent pointer. In other examples, other encodings can be used. In this way, multiple contingent pointers may be associated with a single weak pointer.

[0017] Processor 102 may execute instructions of associated pointer identifier 112 to identify a second pointer associated with the first pointer. Identifying the second pointer associated with the first pointer may further include identifying a first memory location containing the first pointer and identifying a second memory location at an offset from the first memory location.

[0018] Processor 102 may execute instructions of reachability determiner 114 to determine whether a pointer associated with the first pointer (e.g., as discussed herein with respect to associated pointer identifier 112) refers to a reachable object. For example, reachability determiner 114 may determine whether a second pointer refers to a reachable object, whether a third pointer refers to a reachability object, etc. As used herein, a reachable object is an object in a GC managed memory heap that is potentially accessible by a program. During a tracing phase of garbage collection, all objects marked as live, or currently being used by at least one application thread sharing the GC managed memory heap, are walked and marked, or "colored" white, gray, or black. As used herein, "walking" or "walked" is a process by which objects are analyzed to determine if the object is reachable. A white object is an object with an unknown reachability status, a gray object is an object that is known to be reachable, but whose internal references have not yet been walked, and a black object is a reachable object whose internal references have been walked. Initially, all objects

may be colored white. Objects pointed to by root pointers may be colored gray and put in a queue. Gray objects may then be removed from the queue one at a time to be walked. When all of the references that indicate reachability of their referents have been processed, the object may be colored black. When the queue is empty, the tracing phase is complete, and all objects are determined to be either black (reachable) or white (unreachable). The unreachable objects may then be collected in a subsequent sweeping phase of the garbage collection.

[0019] Processor 102 may execute instructions of validity determiner 116 to determine a validity of the first pointer based on whether the pointer associated with the first pointer refers to a reachable object. The validity determiner 116 may further determine the validity of the first pointer based on whether the third pointer refers to the reachable object. A contingent pointer may be considered valid if the associated pointer points to a reachable object. A contingent pointer determined to be valid may be treated as referring to a reachable object.

[0020] If the validity determiner 116 determines that the pointer associated with the first pointer (e.g., as discussed herein with respect to associated pointer identifier 112) refers to a reachable object, the first pointer may be considered valid. If the validity determiner 116 determines that the pointer associated with the first pointer does not refer to a reachable object the first pointer may not be consider valid and/or may be considered invalid. In this manner, the validity of the first pointer, i.e. the contingent pointer, is based on the reachability of the pointer associated with the first pointer. Processor 102 may execute instructions of validity determiner 116 to identify an object referred to by the first pointer as reachable based on the validity of the first pointer.

[0021] Processor 102 may execute instructions of null pointer identifier 118 to identify whether the pointer associated with the first pointer is a null pointer. A null pointer is a pointer with a special value, or "null" value, which indicates that the pointer does not refer to an object. For example, a null pointer may be indicated by all the bits of the pointer value being zero. Null pointer identifier 118 may identify a pointer as a null pointer by analyzing the pointer and determining if the pointer has the null value. If the pointer associated with the first pointer is not identified as a null pointer, validity determiner 116 may determine that the pointer associated with the first pointer refers

to a reachable object. If the pointer associated with the first pointer is identified as a null pointer, validity determiner 116 may determine that the pointer associated with the first pointer does not refer to a reachable object.

[0022] During the tracing phase of the garbage collector, when a contingent pointer is seen while walking a gray object, the associated weak pointer may be consulted. If the associated weak pointer has the null value, the location in the gray object containing the contingent pointer's value may be replaced by the null value. If the associated weak pointer does not have the null value, the contingent pointer may be treated as a strong pointer and the object it refers to may be considered to be a reachable object. For example, if the contingent pointer refers to a white object, the object associated with the contingent pointer is colored gray and added to the gray object queue.

[0023] Once the objects corresponding to the associated weak pointers become unreachable, the governing weak pointers may be cleared in a first GC cycle, and the contingent pointers may be cleared in a second GC cycle. Once the references to an object are cleared, the corresponding object will be seen as unreachable and collected in the second cycle.

[0024] To use a contingent pointer, a strong pointer may first be created based on the contents of the contingent pointer. When the strong pointer is created, the governing weak pointer may be consulted to determine the value of the weak pointer. If the weak pointer has a null value, the contingent pointer's value may be replaced by the null value, and the null value may be returned as the value for the strong pointer. If the weak pointer does not have the null value, the strong pointer may have the same value as the contingent pointer, but marked as strong.

[0025] Processor 102 may execute instructions of invalidity determiner 120 to determine that the pointer is invalid. A contingent pointer may be considered invalid if the associated pointer does not point to a reachable object. A pointer determined to be invalid may be treated as not referring to a reachable object regardless of the actual reachability of any object the pointer may otherwise indicate. Processor 102 may execute instructions of location modifier 122 to modify a location containing the first pointer upon determining that the first pointer is invalid. Location modifier 122 may modify the location by, for example, moving the location of the first pointer in memory from a first location to a second location.

[0026] Processor 102 may execute instructions of request handler 124 to receive a request to dereference the first pointer. Dereferencing refers to a process whereby a pointer is used to identify a memory location and a value stored in the identified memory location is retrieved, modified, or used as a location containing instructions for a processor to execute. The identification of a memory location may involve identifying a memory location pointed to by the pointer and may further involve computing an offset from that location. The offset may be fixed (as when the pointer identifies an object and the offset identifies a field within the object) or variable (as when the pointer identifies an array and the offset identifies an element of the array). The request may arise as part of the execution of a program executing on processor 102, and the execution of instructions of request handler 124 may take place within the same thread as the instructions that initiated the request. Processor 102 may also execute instructions of request handler 124 to grant access to an object referred to by the pointer upon a determination that the pointer is valid (e.g., as discussed herein with respect to validity determiner 114). Request handler 124 may grant access to the object by, for example, generating a pointer referring to the object. The pointer may be, for example, a strong pointer. Processor 102 may also execute instructions of request handler 124 denying the request to dereference the first pointer upon a determination that the first pointer is invalid. Request handler 124 may deny the request to dereference by generating a null pointer. In another example, request handler 124 may grant access by performing a requested operation (e.g., reading or modifying a memory location) and may deny the request by refusing to perform the requested operation and returning a value indicative of such refusal or throwing an exception.

[0027] FIG. 2 is a flowchart of an example method 200 for using pointers in a memory managed system. Method 200 may be described below as being executed or performed by a system, for example, system 100 of FIG. 1, system 300 of FIG. 3 or system 400 of FIG. 4. Other suitable systems and/or computing devices may be used as well. Method 200 may be implemented in the form of executable instructions stored on at least one machine-readable storage medium of the system and executed by at least one processor of the system. Alternatively or in addition, method 200 may be implemented in the form of electronic circuitry (e.g., hardware). In alternate examples of the present disclosure, at least one step of method 200 may be executed substantially concurrently or in a different order than shown in FIG. 2. In alternate examples of the

present disclosure, method 200 may include more or fewer steps than are shown in FIG. 2. In some examples, at least one of the steps of method 200 may, at certain times, be ongoing and/or may repeat.

[0028] Method 200 may start at step 202 and continue to step 204, where the method may include identifying a first pointer as a contingent pointer. Identifying the first pointer as the contingent pointer may take place during a tracing phase of garbage collection. At step 206, the method may include identifying a second pointer associated with the first pointer. The second pointer may be, for example, a strong pointer, a weak pointer, a contingent pointer, etc. Identifying the second pointer associated with the first pointer may further include identifying a first memory location containing the first pointer and identifying a second memory location at an offset from the first memory location. At step 208, the method may include determining whether the second pointer refers to a reachable object. At step 210, the method may include determining a validity of the first pointer based on whether the second pointer refers to a reachable object. Determining the validity of the first pointer Method 200 may eventually continue to step 212, where method 200 may stop.

[0029] Figure 3 is a block diagram illustrating one example of a processing system 300 for implementing the system 300 for using pointers in a memory managed system. System 300 may include a processor 302 and a memory 304 that may be coupled to each other through a communication link (e.g., a bus). Processor 302 may include a Central Processing Unit (CPU), a Graphics Processing Unit (GPU), or another suitable processor. In some examples, memory 304 stores machine readable instructions executed by processor 302 for operating system 300. Memory 304 may include any suitable combination of volatile and/or non-volatile memory, such as combinations of Random Access Memory (RAM), Read-Only Memory (ROM), flash memory, memristor memory, spin-transfer torque memory, and/or other suitable memory.

[0030] Memory 304 stores instructions to be executed by processor 302 including instructions for a pointer generator 310, pointer identifier 312 and validity determiner 314. The components of system 300 may be implemented in the form of executable instructions stored on at least one machine-readable storage medium of system 300 and executed by at least one processor of system 300. Alternatively or in addition, each of the components of system 300 may be implemented in the form of at

least one hardware device including electronic circuitry for implementing the functionality of the component.

[0031] Processor 302 may execute instructions of pointer generator 310 to generate a contingent pointer. The contingent pointer may be paired with a second pointer. The second pointer may be, for example, a strong pointer, a weak pointer, a contingent pointer, etc. Generating the contingent pointer may take place during a tracing phase of garbage collection. Once the contingent pointer has been generated, an object referred to by the contingent pointer may be identified as reachable based on a validity of the contingent pointer. Processor 302 may execute instructions of pointer identifier 312 to identify an unknown pointer as the contingent pointer. An unknown pointer is a pointer that has not yet been classified by the system 300. Pointer identifier 312 may identify the unknown pointer as the contingent pointer during one or more phases of a garbage collection cycle, such as a tracing phase. Processor 302 may execute instructions of validity determiner 314 to determine a validity of the contingent pointer based on whether the second pointer refers to a reachable object.

[0032] FIG. 4 is a block diagram of an example system 400 for using pointers in a memory managed system. System 400 may be similar to system 100 of FIG. 1, for example. In the example illustrated in FIG. 4, system 400 includes a processor 402 and a machine-readable storage medium 404. Although the following descriptions refer to a single processor and a single machine-readable storage medium, the descriptions may also apply to a system with multiple processors and multiple machine-readable storage media. In such examples, the instructions may be distributed (e.g., stored) across multiple machine-readable storage mediums and the instructions may be distributed across (e.g., executed by) across multiple processors.

[0033] Processor 402 may be one or more central processing units (CPUs), graphics processing units (GPUs), microprocessors, and/or other hardware devices suitable for retrieval and execution of instructions stored in machine-readable storage medium 404. In the example illustrated in FIG. 4, processor 402 may fetch, decode, and execute instructions 406, 408 and 410 for using pointers in a memory managed system. As an alternative or in addition to retrieving and executing instructions, processor 402 may include one or more electronic circuits comprising a number of electronic components for performing the functionality of at least one of the instructions in machine-readable storage medium 404. With respect to the executable instruction

representations (e.g., boxes) described and shown herein, it should be understood that part or all of the executable instructions and/or electronic circuits included within one box may, in alternate examples, be included in a different box shown in the figures or in a different box not shown.

[0034] Machine-readable storage medium 404 may be any electronic, magnetic, optical, or other physical storage device that stores executable instructions. Thus, machine-readable storage medium 404 may be, for example, Random Access Memory (RAM), an Electrically-Erasable Programmable Read-Only Memory (EEPROM), a memristor memory device, a spin-transfer torque device, a flash memory device, a storage drive, a solid state device (SSD), an optical disc, and the like. Machine-readable storage medium 404 may be disposed within system 400, as shown in FIG. 4. In this situation, the executable instructions may be "installed" on the system 400. Alternatively, machine-readable storage medium 404 may be a portable, external or remote storage medium, for example, that allows system 400 to download the instructions from the portable/external/remote storage medium. In this situation, the executable instructions may be part of an "installation package". As described herein, machine-readable storage medium 404 may be encoded with executable instructions for using pointers in a memory managed system.

[0035] Referring to FIG. 4, pointer identify instructions 406, when executed by a processor (e.g., 402), may cause system 400 to identify a plurality of contingent pointers. Each contingent pointer in the plurality may be associated with a second pointer. The second pointer may be, for example, a strong pointer, a weak pointer, a contingent pointer, etc. Identifying the plurality of contingent pointers may take place during a tracing phase of garbage collection. Identifying the plurality of contingent pointers may further include identifying an object referred to one of the contingent pointers as reachable based on a validity of that contingent pointer. Reachability determine instructions 408, when executed by a processor (e.g. 402) may cause system 400 to determine that the second pointer refers to a reachable object. Validity determine instructions 410, when executed by a processor (e.g. 402) may cause system 400 to determine that a contingent pointer belonging to the plurality is valid because the second pointer refers to the reachable object.

[0036] The foregoing disclosure describes a number of examples for using pointers in a memory managed system. The disclosed examples may include systems, devices,

computer-readable storage media, and methods for using pointers in a memory managed system. For purposes of explanation, certain examples are described with reference to the components illustrated in FIGS. 1-4. The functionality of the illustrated components may overlap, however, and may be present in a fewer or greater number of elements and components. Further, all or part of the functionality of illustrated elements may co-exist or be distributed among several geographically dispersed locations. Further, the disclosed examples may be implemented in various environments and are not limited to the illustrated examples.

[0037] Further, the sequence of operations described in connection with FIGS. 1-4 are examples and are not intended to be limiting. Additional or fewer operations or combinations of operations may be used or may vary without departing from the scope of the disclosed examples. Furthermore, implementations consistent with the disclosed examples need not perform the sequence of operations in any particular order. Thus, the present disclosure merely sets forth possible examples of implementations, and many variations and modifications may be made to the described examples.

CLAIMS

1. A method for using pointers in a memory managed system, the method comprising:

identifying a first pointer referring to a first object as a contingent pointer;

identifying a second pointer associated with the first pointer;

determining whether the second pointer refers to a reachable object; and

determining a validity of the first pointer based on whether the second pointer refers to a reachable object.

2. The method of claim 1 further comprising:

identifying whether the second pointer is a null pointer; and

determining that the second pointer refers to a reachable object when the second pointer is not identified as a null pointer; and

determining that the second pointer does not refer to a reachable object when the second pointer is identified as a null pointer.

3. The method of claim 1, further comprising:

determining that the first pointer is invalid; and

modifying a location containing the first pointer upon determining that the first pointer is invalid.

4. The method of claim 1, further comprising:

identifying a third pointer associated with the first pointer;

determining whether the third pointer refers to a reachable object; and

wherein determining the validity of the first pointer is further based on whether the third pointer refers to the reachable object.

5. The method of claim 1 wherein identifying the second pointer comprises:

identifying a first memory location containing the first pointer; and

identifying a second memory location at an offset from the first memory location.

6. The method of claim 1, further comprising:

receiving a request to dereference the first pointer;

granting access to the first object upon a determination that the first pointer is valid; and

denying the request to dereference the first pointer upon a determination that the first pointer is invalid.

7. The method of claim 6, wherein:

granting access to the first object further comprises generating a strong pointer referring to the first object; and

denying the request to dereference comprises generating a null pointer.

8. The method of claim 1 wherein identifying the first pointer takes place during a tracing phase of a garbage collector, the method further comprising:

identifying the first object as reachable based on the validity of the first pointer.

9. A system for garbage collection comprising:

a pointer generator to generate a contingent pointer, wherein the contingent pointer is paired with a second pointer;

a pointer identifier to identify an unknown pointer as the contingent pointer during a tracing phase of garbage collection; and

a validity determiner to determine a validity of the contingent pointer based on whether the second pointer refers to a reachable object.

10. The system of claim 9, further comprising:

an invalidity determiner to determine that the contingent pointer is invalid; and

a location modifier to modify a location containing the contingent pointer upon determining that the contingent pointer is invalid.

11. The system of claim 9, further comprising:

a reachability determiner to determine whether a third pointer refers to the reachable object; and

the validity determiner further to determine the validity of the contingent pointer based on whether the third pointer refers to the reachable object.

12. A non-transitory machine-readable storage medium comprising instructions executable by a processor of a computing device for using pointers in a memory managed system, the machine-readable storage medium comprising:

instructions to identify a plurality of contingent pointers, wherein each contingent pointer in the plurality is associated with a second pointer;

instructions to determine that the second pointer refers to a reachable object; and

instructions to determine that a contingent pointer belonging to the plurality is valid because the second pointer refers to the reachable object.

13. The non-transitory machine-readable storage medium of claim 12, further comprising:

instructions to receive a request to dereference the contingent pointer;

instructions to grant access to an object referred to by the contingent pointer upon a determination that the first pointer is valid; and

instructions to deny the request to dereference the first pointer upon a determination that the contingent pointer is invalid.

14. The non-transitory machine-readable storage medium of claim 12, further comprising:

instructions to identify a first memory location containing the contingent pointer; and

instructions to identify a second memory location at an offset from the first memory location.

15. The non-transitory machine-readable storage medium of claim 12, further comprising:

instructions to identifying the object referred to by contingent first pointer as reachable based on the validity of the first pointer.

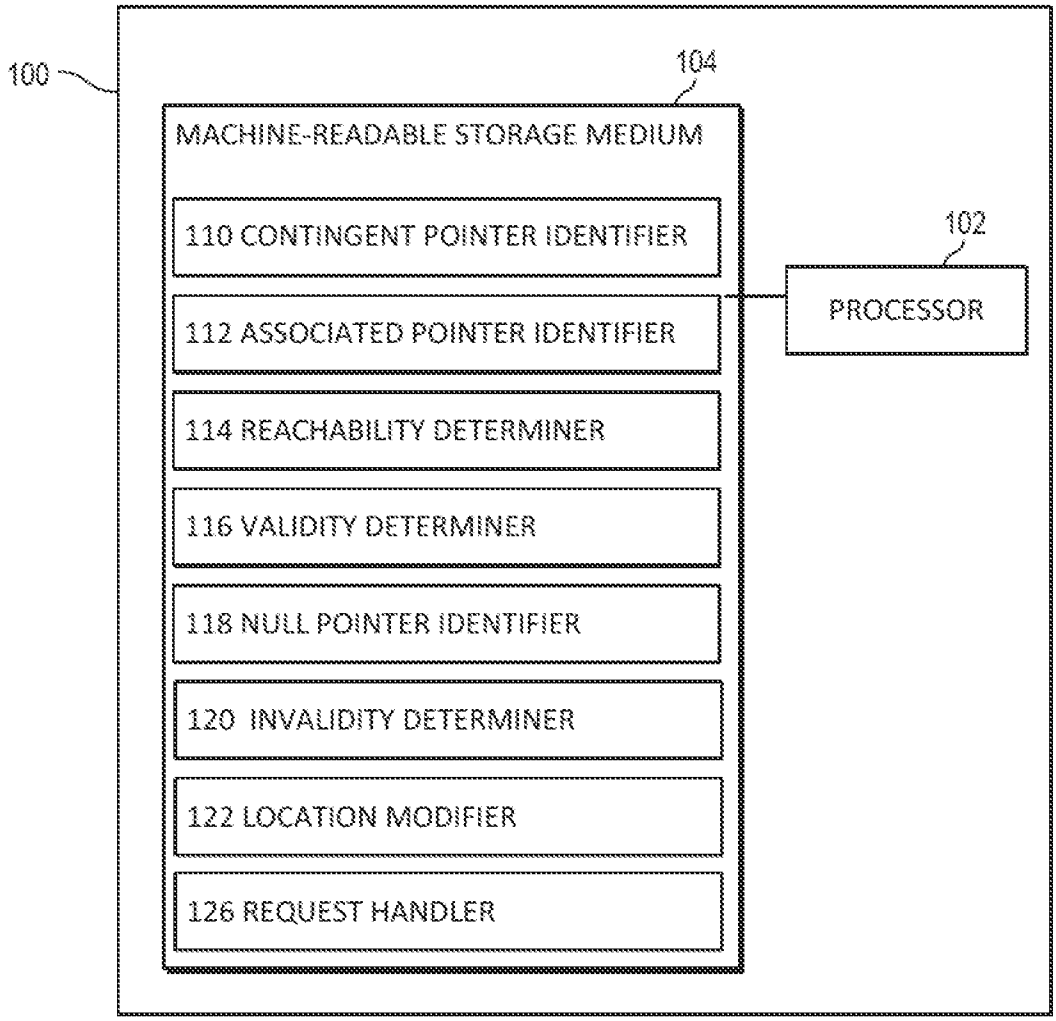
100
104
MACHINE-READABLE STORAGE MEDIUM
110 CONTINGENT POINTER IDENTIFIER
112 ASSOCIATED POINTER IDENTIFIER
114 REACHABILITY DETERMINER
116 VALIDITY DETERMINER
118 NULL POINTER IDENTIFIER
120 INVALIDITY DETERMINER
122 LOCATION MODIFIER
126 REQUEST HANDLER
102
PROCESSOR

FIG. 1

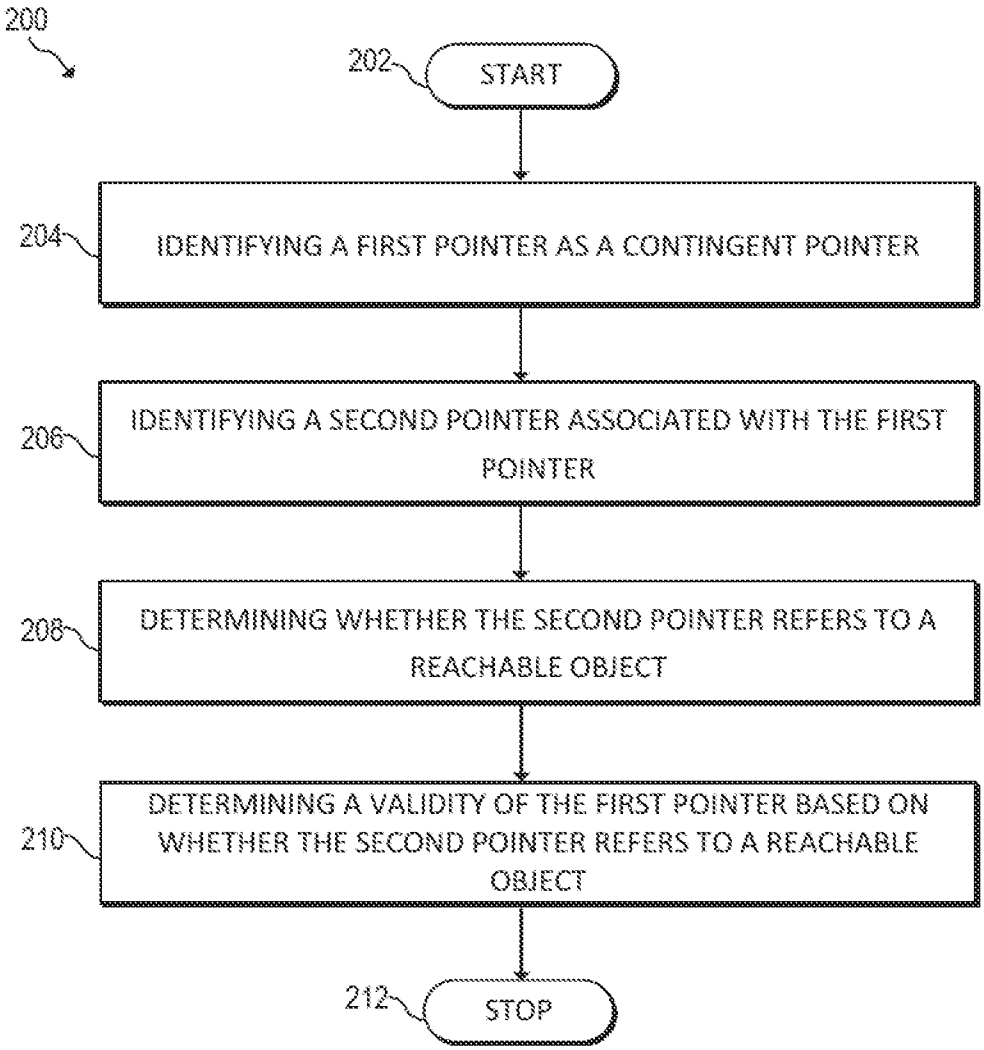
200
202
START
204
IDENTIFYING A FIRST POINTER AS A CONTINGENT POINTER
206
IDENTIFYING A SECOND POINTER ASSOCIATED WITH THE FIRST POINTER
208
DETERMINING WHETHER THE SECOND POINTER REFERS TO A REACHABLE OBJECT
210
DETERMINING A VALIDITY OF THE FIRST POINTER BASED ON WHETHER THE SECOND POINTER REFERS TO A REACHABLE OBJECT
212
STOP

FIG. 2

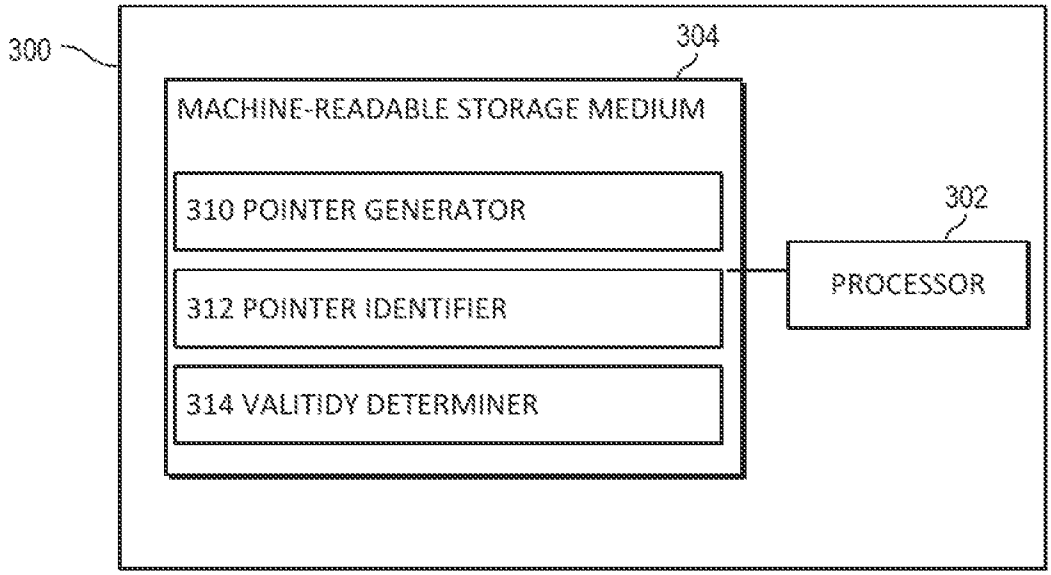
300
304
MACHINE-READABLE STORAGE MEDIUM
310 POINTER GENERATOR
312 POINTER IDENTIFIER
314 VALITIDY DETERMINER
302
PROCESSOR

FIG. 3

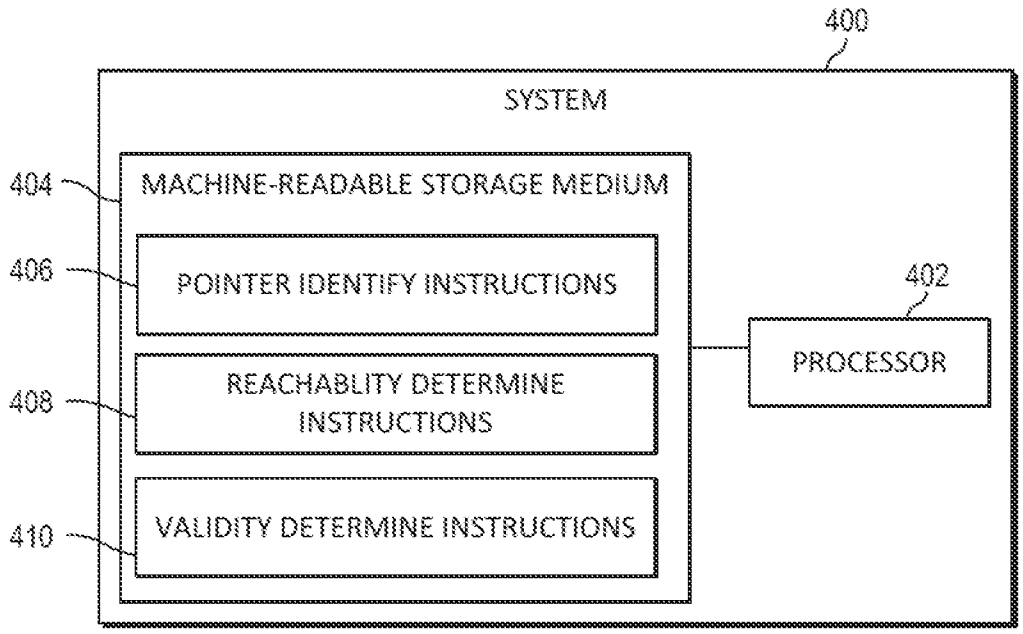
400
SYSTEM
404
MACHINE-READABLE STORAGE MEDIUM
406
POINTER IDENTIFY INSTRUCTIONS
402
PROCESSOR
408
REACHABLITY DETERMINE INSTRUCTIONS
410
VALIDITY DETERMINE INSTRUCTIONS

FIG. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2015/062970

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/02(2006.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)
G06F 12/02; G06F 17/30; G06F 12/00; G06F 9/00; G06F 9/45; G06F 13/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean utility models and applications for utility models
Japanese utility models and applications for utility models

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
eKOMPASS(KIPO internal) & Keywords: contingent, pointer, associated, reachable, object, validity, identifying, determining, garbage, collection, memory

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 05848423 A (ZAHIR EBRAHIM et al.) 08 December 1998 See column 2, lines 54-60; column 3, lines 1-7; column 5, lines 53-56; column 7, lines 25-30; and figures 1, 4A.	1-5,8-12,14-15
A		6-7,13
Y	US 05465351 A (STEVEN E. LEMMO) 07 November 1995 See column 9, lines 43-45, 65-67; column 10, lines 1-3, 25-29; and figures 5-7.	1-5,8-12,14-15
A	US 2010-0070955 A1 (VINEET KAHLON) 18 March 2010 See paragraphs [0057]-[0098]; and figure 2.	1-15
A	US 2007-0288708 A1 (BRATIN SAHA et al.) 13 December 2007 See paragraphs [0065]-[0067]; and figure 8.	1-15
A	US 2005-0027761 A1 (GANSHA WU et al.) 03 February 2005 See paragraphs [0017]-[0020]; and figure 1.	1-15

☐ Further documents are listed in the continuation of Box C. ☒ See patent family annex.

* Special categories of cited documents:
"A" document defining the general state of the art which is not considered to be of particular relevance
"E" earlier application or patent but published on or after the international filing date
"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)
"O" document referring to an oral disclosure, use, exhibition or other means
"P" document published prior to the international filing date but later than the priority date claimed
"T" later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention
"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone
"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents,such combination being obvious to a person skilled in the art
"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
26 August 2016 (26.08.2016)	**26 August 2016 (26.08.2016)**

Name and mailing address of the ISA/KR
International Application Division
Korean Intellectual Property Office
189 Cheongsa-ro, Seo-gu, Daejeon, 35208, Republic of Korea
Facsimile No. +82-42-481-8578

Authorized officer
CHIN, Sang Bum
Telephone No. +82-42-481-8398

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2015/062970

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 05848423 A	08/12/1998	None	
US 05465351 A	07/11/1995	WO 94-04976 A1	03/03/1994
US 2010-0070955 A1	18/03/2010	None	
US 2007-0288708 A1	13/12/2007	US 7478210 B2	13/01/2009
US 2005-0027761 A1	03/02/2005	CN 1829977 A	06/09/2006
		CN 1829977 B	11/08/2010
		EP 1652094 A1	03/05/2006
		EP 1652094 B1	21/11/2012
		ES 2398420 T3	19/03/2013
		US 7089273 B2	08/08/2006
		WO 2005-013134 A1	10/02/2005