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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
5 October 2017 (05.10.2017)

WIPO | PCT

(10) International Publication Number
WO 2017/171809 A1

(51) International Patent Classification:
G06F 12/0868 (2016.01) *G06F 12/0815* (2016.01)
G06F 12/0804 (2016.01)

(21) International Application Number:
PCT/US2016/025368

(22) International Filing Date:
31 March 2016 (31.03.2016)

(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: **NAWAB, Faisal**; 1501 Page Mill Rd., Palo Alto, California 94304 (US). **IZRAELEVITZ, Joseph**; 1501 Page Mill Rd., Palo Alto, California 94304 (US). **KELLY, Terence P**; 1501 Page Mill Rd., Palo Alto, California 94304 (US). **MORREY III, Charles B.**; 1501 Page Mill Rd., Palo Alto, California 94304 (US). **CHAKRABARTI, Dhruva**; 1501 Page Mill Rd., Palo Alto, California 94304 (US).

(74) Agents: **PAGAR, Preetam** 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))*

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) Title: MEMORY SYSTEM TO ACCESS UNCORRUPTED DATA

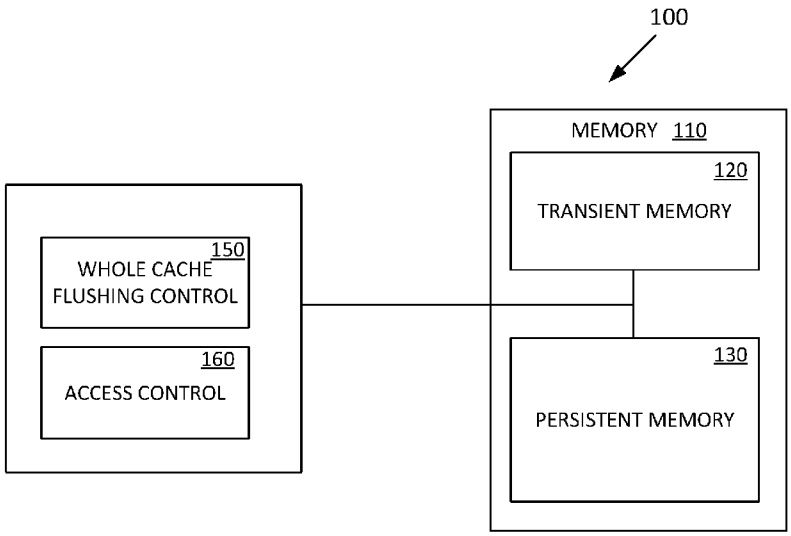

Figure 1

(57) Abstract: An example system includes a transient memory device, a persistent memory device, and a memory controller. The memory controller includes a whole cache flushing controller to cause all data in the transient memory device to be written to the persistent memory device and an access controller to identify location of most recent uncorrupted data in the persistent memory device.

WO 2017/171809 A1

Published:

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

MEMORY SYSTEM TO ACCESS UNCORRUPTED DATA

BACKGROUND

[0001] Various memory systems include a cache which may function as a working space for storing information in a transient manner. Such cache memory may be used by various routines which may require quick or frequent access to certain data. For more permanent storage, the memory systems may also include a non-volatile memory. Data may be written to the non-volatile memory from the cache memory. In various systems, data stored in cache memory may be subject to erasure, destruction or corruption. Such data is referred to as "transient" data. Data that has been written to or stored in the non-volatile memory is referred to as "persistent" data and may not be subject to erasure, destruction or corruption in the case of a failure event.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] For a more complete understanding of various examples, reference is now made to the following description taken in connection with the accompanying drawings in which:

[0003] Figure 1 illustrates an example system with a transient memory device and a persistent memory device;

[0004] Figure 2 illustrates an example whole cache flushing controller;

[0005] Figure 3 is a flow chart illustrating an example process for a whole cache memory flush;

[0006] Figure 4 is a flow chart illustrating an example process for a restart upon a failure;

[0007] Figure 5 illustrates an example access controller including a hash map;

[0008] Figure 6 illustrates an example structure of an example bucket in a hash map;

[0009] Figure 7 is a flow chart illustrating an example process for fulfilling an access request;

[0010] Figure 8 is a flow chart illustrating an example process for fulfilling an update request; and

[0011] Figure 9 illustrates a block diagram of an example system with a computer-readable storage medium including instructions executable by a processor for fulfilling an access request.

DETAILED DESCRIPTION

[0012] Various examples described herein provide for fault-tolerance in memory systems. In this regard, memory systems are provided with whole-cache flushing to persistent memory and embedded logging which allows access of persistent data. The whole-cache flushing, or writing to persistent (e.g., non-volatile memory) may be performed at regular intervals, as well as in conjunction with a detected failure. The embedded logging is included in a hash map with pointers to data at three epochs, or intervals, thus allowing access to the most recent persistent data which was not subject to a failure. As used herein, the terms "epoch" and "interval" may be used interchangeably.

[0013] Referring now to the figures, Figure 1 illustrates an example system 100 with a transient memory device and a persistent memory device. The example system 100 includes a memory system 110 which may include multiple memory devices. For example, in the example system 100 of Figure 1, the memory system 110 includes a transient memory device 120 (such as a cache memory) and a persistent memory device 130. In various examples, the transient memory device 120 may be a random access memory (RAM), such as a dynamic RAM (DRAM). In other examples, the transient memory device 120 may be cache memory provided in a processor, such as a central processing unit (CPU) cache. The transient memory device 120 may be accessed by, for example, routines executing on a processor. The persistent memory device 130 may be a non-volatile memory and may include any of a variety of storage devices.

[0014] The example system 100 further includes a whole cache flushing controller 150 and an access controller 160. In some examples, the whole cache flushing controller 150 and the access controller 160 may be provided in a memory controller (not shown). Such memory controllers may control operation of and access to the memory system 110. For example, the memory controller may interface with a processor (not shown in Figure 1) executing one or more routines which may read from or write to one or more memory devices in the memory system 110. In other examples, the whole cache flushing controller 150 and the access controller 160 may be implemented within processor as instructions to be executed by the processor, for example.

[0015] The whole cache flushing controller 150 may manage flushing of data from the transient memory device 120 to the persistent memory device 130. As used herein, flushing refers to writing of data which may be stored in the transient memory device 120 in a transient

manner to the persistent memory device 130 in a persistent manner. An example whole cache flushing controller 150 is described below with reference to Figure 2. The example access controller 160 of the example system 100 controls and manages access of the various memory devices in the memory system 110. For example, the access controller 160 may manage read or write requests received from a processor (not shown). In one regard, the access controller 160 manages access to identify and point to data which is persistent even following a failure event, for example, as described below with reference to Figures 6-8. An example access controller 160 is described below with reference to Figure 5.

[0016] Referring now to Figure 2, an example whole cache flushing controller 150 is illustrated, along with a timeline illustrating the flushing of data in the transient memory device 120 to persistent data in the persistent memory device 130. As used herein, whole cache flushing is distinguished from cache-line flushing. In cache-line flushing, data is written from a transient memory to a persistent memory with only a single cache line being written at a time to persistent memory, resulting in significant overhead. In various examples, a single cache line may refer to a single unit which is the smallest granularity of a write from a transient memory to a persistent memory. By contrast, in whole cache flushing, all the data in the transient memory device 120 is written to the persistent memory device 130 on a periodic or arbitrary basis. In various examples, whole cache flushing may include individual cache-line writing to persistent data for certain meta-data. Further, in some examples, whole cache flushing may be implemented as a series of smaller flushes (e.g., cache-line flushes) which are collectively associated with a particular interval or epoch.

[0017] As illustrated in the example of Figure 2, the example whole cache flushing controller 150 includes such meta-data in the form of an interval counter 210. Additionally, the example cache flushing controller 150 includes a failure interval list 220. The interval counter 210 tracks the intervals for which a whole cache flush is performed. As illustrated in the timeline, a whole cache flush is performed for each time interval, or epoch. At the end of each interval, the interval counter 210 is incremented to track the current interval. While a new interval generally starts at regular time intervals (as described below with reference to Figure 3), in some examples, a new interval may be defined upon the detection or determination of certain failure events (described in greater detail below with reference to Figure 4). Thus, upon the detection or determination of a failure, a new interval is started, the interval counter 210 is incremented, and a

whole cache flush is performed. The failure interval list 220 may maintain a list of epoch identifiers corresponding to an interval associated with a failure.

[0018] Referring now to Figure 3, a flow chart illustrates an example process for a whole cache memory flush occurring at a regular time interval. The example process 300 may be implemented in the memory controller 140 of Figure 1, for example. The process 300 begins at the end of a time interval with the incrementing of the interval counter 210 (block 310). The length of the time interval at which a new interval is started may be selected based on, for example, the size or speed of the transient memory device.

[0019] As noted above, whole cache flushing may include individual cache-line writing to persistent data for certain meta-data. In this regard, a cache line containing the interval counter 210 may be written to the persistent memory device (block 320).

[0020] In various examples, at any time, the transient memory device may be in use by zero, one or more processes, or thread, being executed by a processor. Prior to flushing all data from the transient memory, the memory controller 140 may notify each thread of the impending flush (block 330). In one example, the memory controller 140 may await an acknowledgement from each thread before proceeding with the flush. The acknowledgement may allow process 300 to ensure that each thread is informed that a new interval has begun or been entered, for example. Thus, at block 340 of the process 300, the memory controller determines whether each thread has been informed. In some examples, the process may continue after a predetermined length of time. At block 350, the entire contents of the transient memory device are written to the persistent memory device. In various examples, a new whole cache flush may begin only after a previous whole cache flush has been completed.

[0021] Referring now to Figure 4, a flow chart illustrates an example process for a global restart to recover from a detected or determined failure. In the example process 400, identifiers of the most recent interval, or epoch, during which the failure occurred, as well as the interval immediately preceding the failure interval, are added to the interval failure list 220 illustrated in Figure 4 (block 410). The failure interval list 220 may be updated in the persistent memory by writing the new entries to the failure interval list 220 to the persistent memory (block 420).

[0022] As noted above, while a new interval is generally defined at regular time intervals, a failure event may also lead to the start of a new interval. In this regard, at block 430, the interval counter 210 is incremented, and the interval counter 210 is written to the persistent memory

(block 440). Again, as noted above, in the context of whole cache flushing, only certain meta-data may be written to the persistent memory as a cache-line flush. The flushing at blocks 420 and 440 are examples of such meta-data flushing. At block 450, the process 400 resumes operation of the memory system. In this regard, all processes or threads interrupted by the failure may be restarted. In this regard, various processes or threads may re-submit access requests, for example.

[0023] Referring now to Figure 5, an example access controller 160 of Figure 1 is illustrated in greater detail. In various examples, identification of data that was not corrupted by failures in the context of whole cache flushing may be facilitated by a hash map 510. In the examples illustrated in Figures 1 and 5, the hash map 510 is provided in the access controller 160. In other examples, the hash map 510 may be provided in other positions, such as within the persistent memory device 130 of Figure 1. As apparent from the timeline in Figure 2, a gap may exist between a whole cache flush and a failure event. Thus, any data written to the transient memory device within that gap may be considered corrupt or unreliable. In this regard, the example hash map 510 may provide pointers to only data which was written to persistent memory before the failure event, for example.

[0024] The example hash map 510 maintains buckets 512a-n shared by all threads. In various examples, the hash map 510 implements buckets 512a-n as append-only or prepend-only lists.

[0025] Referring now to Figure 6, a structure of an example bucket in the hash map 510 of Figure 5 is illustrated. In various examples, the example bucket 512 contains three pointers 612, 614, 616 identifying a memory location in either the transient memory device or the persistent memory device. In one example, the example bucket 512 is 256 bits in length in which each pointer is a 64-bit field. The example bucket 512 further includes a 64-bit state word 618 indicative of the state of the corresponding bucket. The state word 618 may include a 58-bit field for an interval identifier 620 indicative of the interval during which the bucket 512 was last updated. In addition, the 64-bit state word 618 includes three 2-bit role indicators 622 indicative of the role of each of the pointers 612, 614, 616.

[0026] Each of the role indicators 622 indicates whether the corresponding pointer 612, 614, 616 is indicated as a “current pointer”, a “previous pointer” or a “persistent pointer.” In one example, the role of each pointer 612, 614, 616 may be derived by comparing the interval

identified in the interval identifier 620 to the failure interval list 220 and the interval counter 210. In various examples, the "current pointer" (P1 612 in the example of Figure 6) points to the most recently added record (record "e") at or before a first interval, which is the interval identified in the interval identifier 620. Further, the "previous pointer" (P2 614 in the example of Figure 6) may point to the most recently added record (record "d") at or before a second interval which is the interval immediately preceding the interval identified in the interval identifier 620 (the first interval), and the "persistent pointer" (P3 616 in the example of Figure 6) may point to the most recently added record (record "b") at or before a third interval immediately preceding the second period. The size of each record a-e may be arbitrary and may depend on the data or information held in the record.

[0027] The roles of the pointers 612, 614, 616 may change over time. For example, at one point, P1 may be the current pointer, P2 the previous pointer and P3 the persistent pointer. At the next interval, P2 may become the persistent pointer, P1 the previous pointer and P3 the current pointer.

[0028] The pointers 612, 614, 616 may be updated each time data corresponding to the bucket is written to memory. Upon addition of a new data entry, the interval identifier 620 may be updated if necessary. For example, if the new data entry is the first data entry for the present interval, the interval identifier 620 may be updated to the value indicated in the interval counter 210 of Figure 2. Next, the pointers 612, 614, 616 may be updated to reflect the addition of the new data entry. In one example, the pointers may be rotated such that the oldest pointer is rotated to the front. For example, in the case of the pointers 612, 614, 616 illustrated in Figure 6, upon the addition of a new data entry, the formerly persistent pointer P3 may now point to the new data entry. The other two pointers P1 and P2 may now be rotated backward. Thus, the formerly previous pointer P2 may now become a persistent pointer, and the formerly current pointer P1 may now become a previous pointer. In other examples, the roles may change in an arbitrary manner.

[0029] As indicated by the timeline of Figure 2, there may be a time differential between the end of an interval and the flushing of the transient memory device. Thus, if a failure occurs within that time differential, any data written in that time differential, as well as any data written in the previous interval is potentially not persistent. Thus, in some examples, a failure may result in two intervals for which data may be corrupt. Thus, with reference to Figure 6, the persistent

pointer always points to persistent data, while both the current pointer and the previous pointer may point to transient data. Thus, in various examples described herein, following a failure event, the state word 618 may identify a persistent pointer from which only persistent data is accessible.

[0030] Referring now to Figure 7, a flow chart illustrates an example process for fulfilling a data access request. In the example process 700 of Figure 7, a data access request may be received from, for example, a processor executing a routine (block 710). Upon receiving the request, the process 700 determines whether a failure event occurred in the current interval (block 720). In this regard, the current interval may be determined based on the interval identifier 620 of the state word 618 of Figure 6. Further, the process 700 may access the failure interval list 220 illustrated in the example of Figure 2 to determine if the current interval included a failure event. If the current interval is not included in the failure interval list 220, the process 700 proceeds to block 730 and fulfills the data access request using the current pointer. In this regard, the process 700 may use the role identifiers 618 to determine which identifier is the current identifier and use the appropriate pointer 612, 614, 616.

[0031] If, at block 720, it is determined that a failure event occurred during the current interval, the process determines whether a failure event occurred in the previous interval (block 740). As noted above, the previous interval is the interval immediately preceding the current interval which may have been determined based on the interval identifier 620 of the state word 618. Again, the process 700 may access the failure interval list 220 illustrated in the example of Figure 2 to determine if the previous interval included a failure event. If the previous interval is not included in the failure interval list 220, the process 700 proceeds to block 750 and fulfills the data request using the previous pointer, which may be determined using the role identifiers 618. If, at block 740, it is determined that a failure event occurred during the previous interval, the process 700 fulfills the data access request using the persistent pointer, which may be determined using the role identifiers 618 (block 760).

[0032] The example process 700 of Figure 7 or similar processes may be used to perform various categories of tasks, including read-only accesses and updates (e.g., inserts, overwrites, or deletes). Read-only processes may consult the failure interval list 220 of Figure 2 and the state word field 618 of the bucket 512 corresponding to the appropriate thread to determine which data (e.g., transient data) to overlook, as described above with reference to Figure 7.

[0033] Update processes may use a process similar to that described above with reference to Figure 7. In the case of updates, the process may additionally begin by locking the affected bucket 512 before proceeding with the example steps of the example process 700. In various examples, an update process may append a record, change the corresponding state word 618 and potentially change one or more pointers 612, 614, 616. In this regard, the interval identifier 620 of the state word 618 may be updated to reflect the current interval, and the role identifiers 622 may be updated to reflect updated roles of the pointers 612, 614, 616. An example update process is described below with reference to Figure 8.

[0034] Referring now to Figure 8, a flow chart illustrates an example process for fulfilling an update request. In the example process 800 of Figure 8, an update request may be received from, for example, a processor executing a routine (block 810). Upon receiving the request, the process 800 determines whether if the current interval (e.g., the interval identifier 620 of Figure 6) is the same as the value of the interval corresponding to the update (block 812). In this regard, if the update request is not the first update request in the update interval, the interval identifier 620 will have already been updated. In this case, the process 800 proceeds to block 814 and makes no changes to the pointer roles 622 of Figure 6. In the flow chart of Figure 8, the pointer roles 622 are indicated as "(P1, P2, P3)" with the P1 representing the current pointer, P2 representing the previous pointer, and P3 representing the persistent pointer.

[0035] If, at block 812, the current interval (e.g., the interval identifier 620 of Figure 6) is determined to be not the same as the value of the interval corresponding to the update, the process 800 may determine that the update request is for the first update of the update interval. Accordingly, the process 800 proceeds to block 816 and determines whether the current interval is the interval prior to the update interval. If so, the process 800 determines whether a failure event occurred in the current interval (block 818). In this regard, the process 800 may access the failure interval list 220 illustrated in the example of Figure 2 to determine if the current interval included a failure event. If no failure event is determined to have occurred during the current interval, the process proceeds to block 820 and changes pointer roles 622 to (Persistent, Current, Previous). Thus, P1 representing the Current pointer now points to the previously Persistent pointer, P2 representing the Previous pointer now points to the previously Current pointer, and P3 representing the Persistent pointer now points to the previously Previous pointer. On the

other hand, if the current interval is determined to have included a failure event, the process proceeds to block 822 and makes no changes to the pointer roles.

[0036] Referring again to block 816, if the process determines that the current interval is not the interval prior to the update interval and is, thus, an earlier interval, the process 800 proceeds to block 824 and determines whether a failure event occurred in the current interval. If no failure event is determined to have occurred during the current interval, the process proceeds to block 826 and changes pointer roles 622 to (Persistent, Previous, Current). Thus, P1 representing the Current pointer now points to the previously Persistent pointer, P2 representing the Previous pointer still points to the previously Previous pointer, and P3 representing the Persistent pointer now points to the previously Current pointer. On the other hand, if the current interval is determined to have included a failure event, the process proceeds to block 828 and determines whether a failure event occurred in the previous interval. If no failure event is determined to have occurred during the previous interval, the process proceeds to block 830 and changes pointer roles 622 to (Current, Persistent, Previous). Thus, P1 representing the Current pointer still points to the previously Current pointer, P2 representing the Previous pointer now points to the previously Persistent pointer, and P3 representing the Persistent pointer now points to the previously Previous pointer. On the other hand, if the previous interval is determined to have included a failure event, the process proceeds to block 832 and makes no changes to the pointer roles.

[0037] Thus, the example processes described above with reference to Figures 7 and 8 may be used with the example whole cache flushing described above. Further, the example processes may be used within various transactions related to example memory devices or systems. For example, the examples processes 700, 800 may be used in read-modify transactions or read-only transactions.

[0038] Referring now to Figure 9, a block diagram of an example system is illustrated with a computer-readable storage medium including instructions executable by a processor for fulfilling an access request. The system 900 includes a processor 910 and a computer-readable storage medium 920. The computer-readable storage medium 920 includes example instructions 921-923 executable by the processor 810 to perform various functionalities described herein.

[0039] The example instructions include accessing using current pointer instructions 921. As described above, it may be determined whether a failure event occurred during the current

interval. The determination may be made based on the interval identifier 620 and the failure interval list 220. If no failure event occurred during the current interval, an access request may be fulfilled using the current pointer.

[0040] The example instructions further include accessing using previous pointer instructions 922. If a failure event is determined to have occurred during the current interval, it may be determined whether a failure event occurred during the previous interval. If no failure event occurred during the previous interval, the access request may be fulfilled using the previous pointer.

[0041] The example instructions further include accessing using persistent pointer instructions 923. If a failure event is determined to have occurred during the current interval and the previous interval, the access request may be fulfilled using the persistent pointer.

[0042] Thus, in accordance with various examples described herein, a memory system is provided with writing of data from a transient memory to a persistent memory with reduced overhead by using whole cache flushing. Further, accessibility is facilitated using embedded logging which ensures that only persistent data is accessed following a failure event.

[0043] Software implementations of various examples can be accomplished with standard programming techniques with rule-based logic and other logic to accomplish various database searching steps or processes, correlation steps or processes, comparison steps or processes and decision steps or processes.

[0044] The foregoing description of various examples has been presented for purposes of illustration and description. The foregoing description is not intended to be exhaustive or limiting to the examples disclosed, and modifications and variations are possible in light of the above teachings or may be acquired from practice of various examples. The examples discussed herein were chosen and described in order to explain the principles and the nature of various examples of the present disclosure and its practical application to enable one skilled in the art to utilize the present disclosure in various examples and with various modifications as are suited to the particular use contemplated. The features of the examples described herein may be combined in all possible combinations of methods, apparatus, modules, systems, and computer program products.

[0045] It is also noted herein that while the above describes examples, these descriptions should not be viewed in a limiting sense. Rather, there are several variations and modifications which may be made without departing from the scope as defined in the appended claims.

WHAT IS CLAIMED IS:

1. A system, comprising:

a transient memory device;

a persistent memory device;

a whole cache flushing controller to cause all data in the transient memory device to be written to the persistent memory device; and

an access controller to identify location of most recent uncorrupted data in the persistent memory device or the transient memory device.

2. The system of claim 1, wherein the whole cache flushing controller is to cause all data in the transient memory device to be written to the persistent memory at regular intervals.

3. The system of claim 2, wherein the whole cache flushing controller is to cause all data in the transient memory device to be written to the persistent memory at a detected failure.

4. The system of claim 2, wherein the access controller includes at least one bucket, the bucket including a pointer portion including pointers corresponding to data entries.

5. The system of claim 4, wherein the pointer portion includes a first pointer corresponding to a first interval, a second pointer corresponding to a second interval and a third pointer corresponding to a third interval, wherein the first interval is the current interval, the second interval is an interval immediately preceding the first interval, and the third interval is an interval preceding the second interval.

6. The system of claim 1, wherein each pointer points to a data entry most recently added to a corresponding interval.

7. A method, comprising:

receiving a request for access to a memory system, the memory system having a transient memory device and a persistent memory device, data from the transient memory device being

written to the persistent memory device at regular intervals;

accessing the memory system using a pointer to a first interval when no failure is determined in the first interval;

accessing the memory system using a pointer to a second interval when a failure is determined in the first interval and no failure is determined in the second interval, the second interval immediately preceding the first interval; and

accessing the memory system using a pointer to a third interval when failures are determined in the first interval and in the second interval, the third interval preceding the second interval.

8. The method of claim 7, wherein a failure is determined in the first interval if the first interval is indicated as the current interval in a bucket corresponding to the request for access and the current interval is included in a list of intervals with failures.

9. The method of claim 7, wherein the pointer is accessed from a bucket corresponding to the request, the bucket including a pointer portion with pointers corresponding to data entries.

10. The method of claim 9, wherein the pointer portion includes a first pointer corresponding to the first interval, a second pointer corresponding to the second interval and a third pointer corresponding to the third interval.

11. The method of claim 7, wherein each pointer points to a data entry most recently added to a corresponding interval.

12. A non-transitory computer-readable storage medium encoded with instructions executable by a processor of a computing system, the computer-readable storage medium comprising instructions to:

access a memory system, the memory system having a transient memory device and a persistent memory device, data from the transient memory device being written to the persistent memory device at regular intervals, wherein the memory system is accessed using a pointer to a first interval when no failure is detected in the first interval;

access the memory system using a pointer to a second interval when a failure is detected in the first interval and no failure is detected in the second interval, the second interval immediately preceding the first interval; and

access the memory system using a pointer to a third interval when failures are detected in the first interval and in the second interval, the third interval immediately preceding the second interval.

13. The non-transitory computer-readable storage medium of claim 12, wherein a failure is determined in the first interval if the first interval is indicated as the current interval in a bucket corresponding to the request for access and the current interval is included in a list of intervals with failures.

14. The non-transitory computer-readable storage medium of claim 12, wherein the pointer is accessed from a bucket corresponding to the request, the bucket including a pointer portion with pointers corresponding to data entries.

15. The non-transitory computer-readable storage medium of claim 14, wherein the pointer portion includes a first pointer corresponding to the first interval, a second pointer corresponding to the second interval and a third pointer corresponding to the third interval.

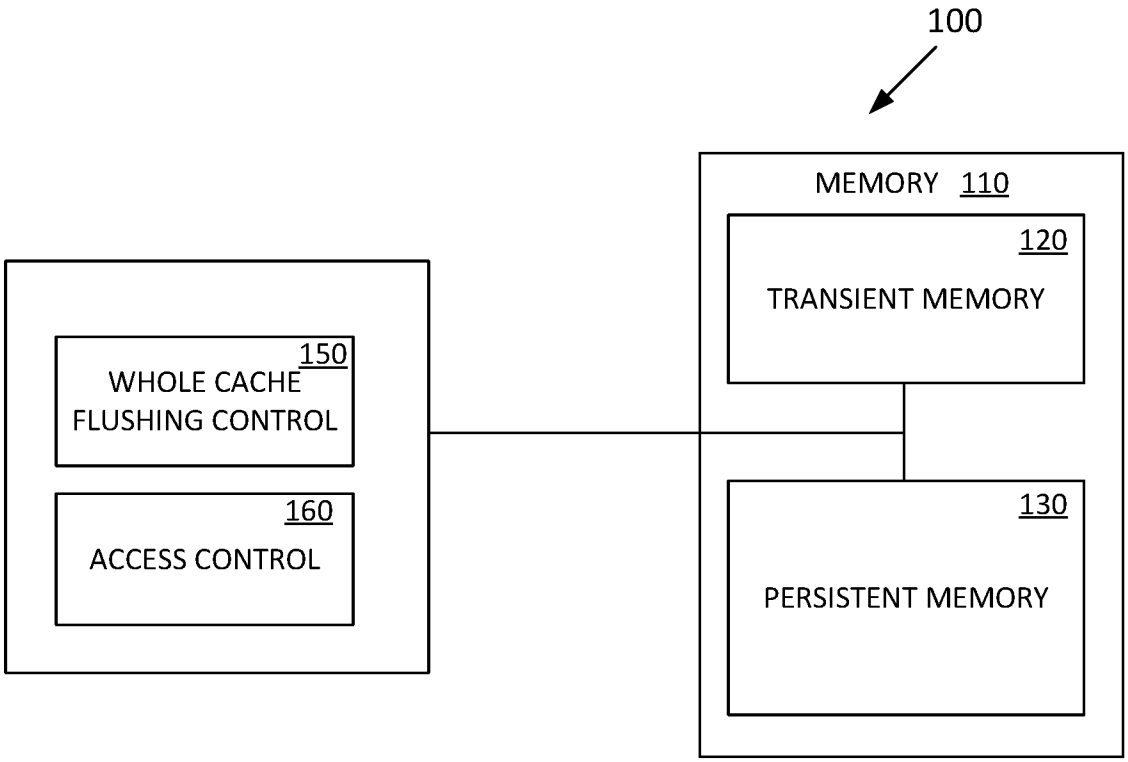

Figure 1

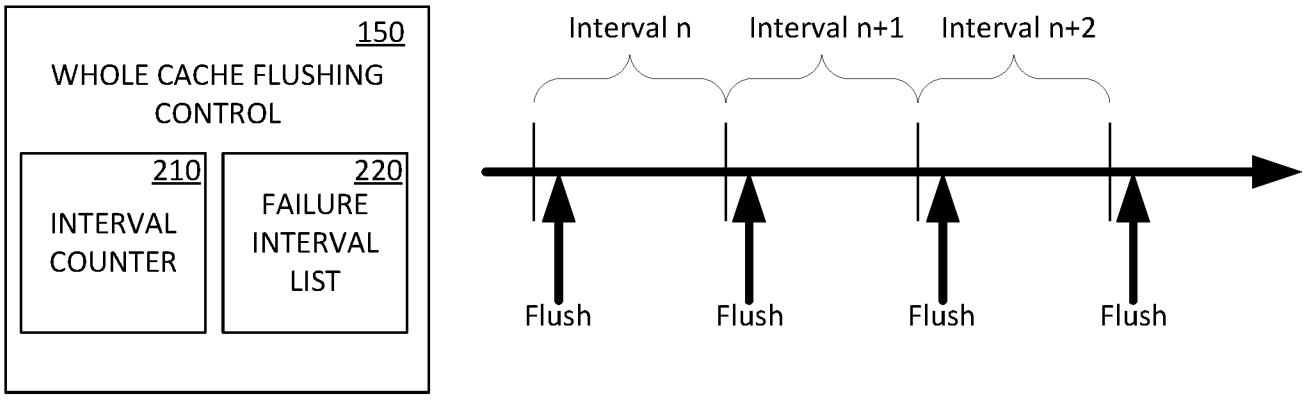

Figure 2

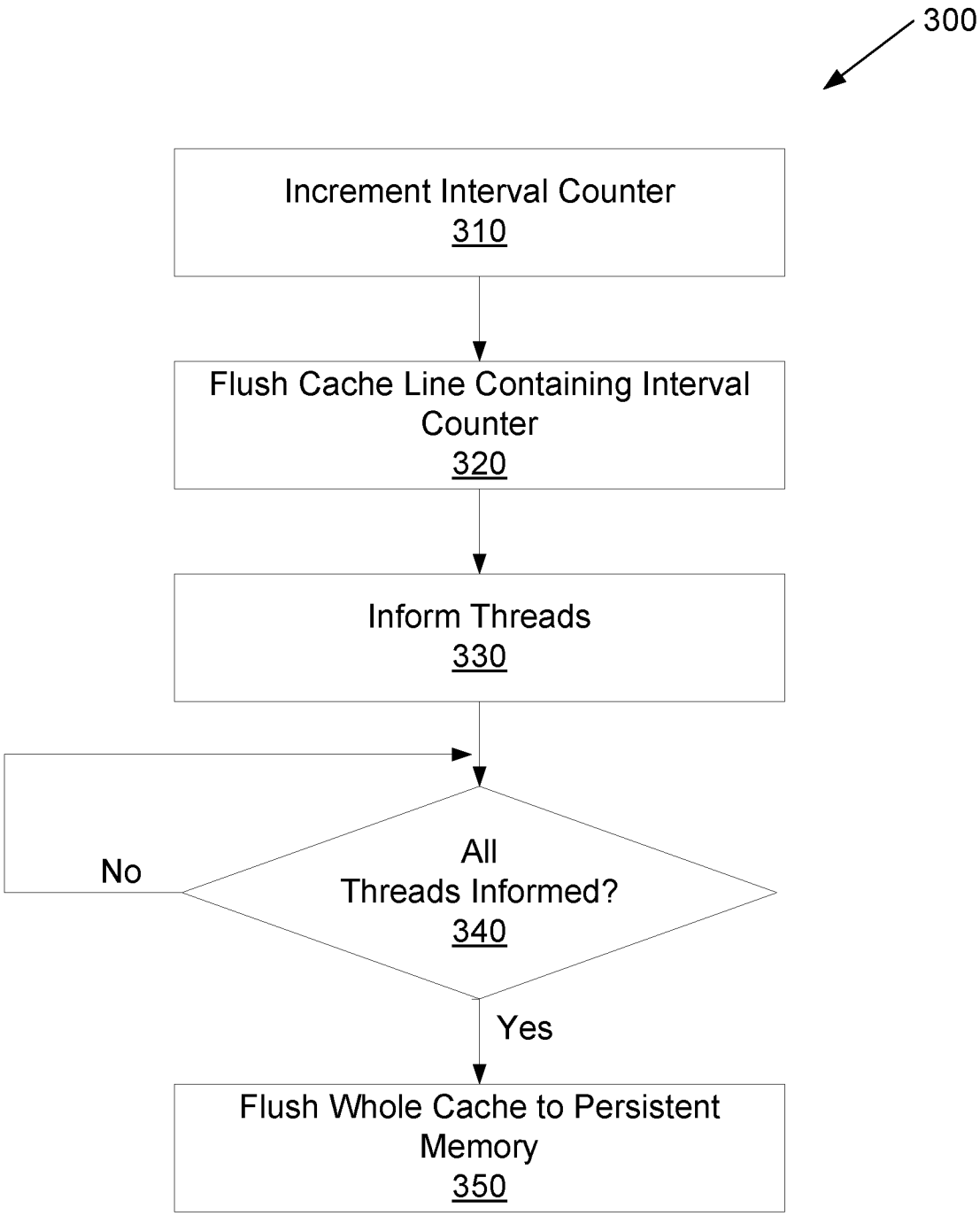
300
Increment Interval Counter
310
Flush Cache Line Containing Interval Counter
320
Inform Threads
330
All Threads Informed?
340
No
Yes
Flush Whole Cache to Persistent Memory
350

Figure 3

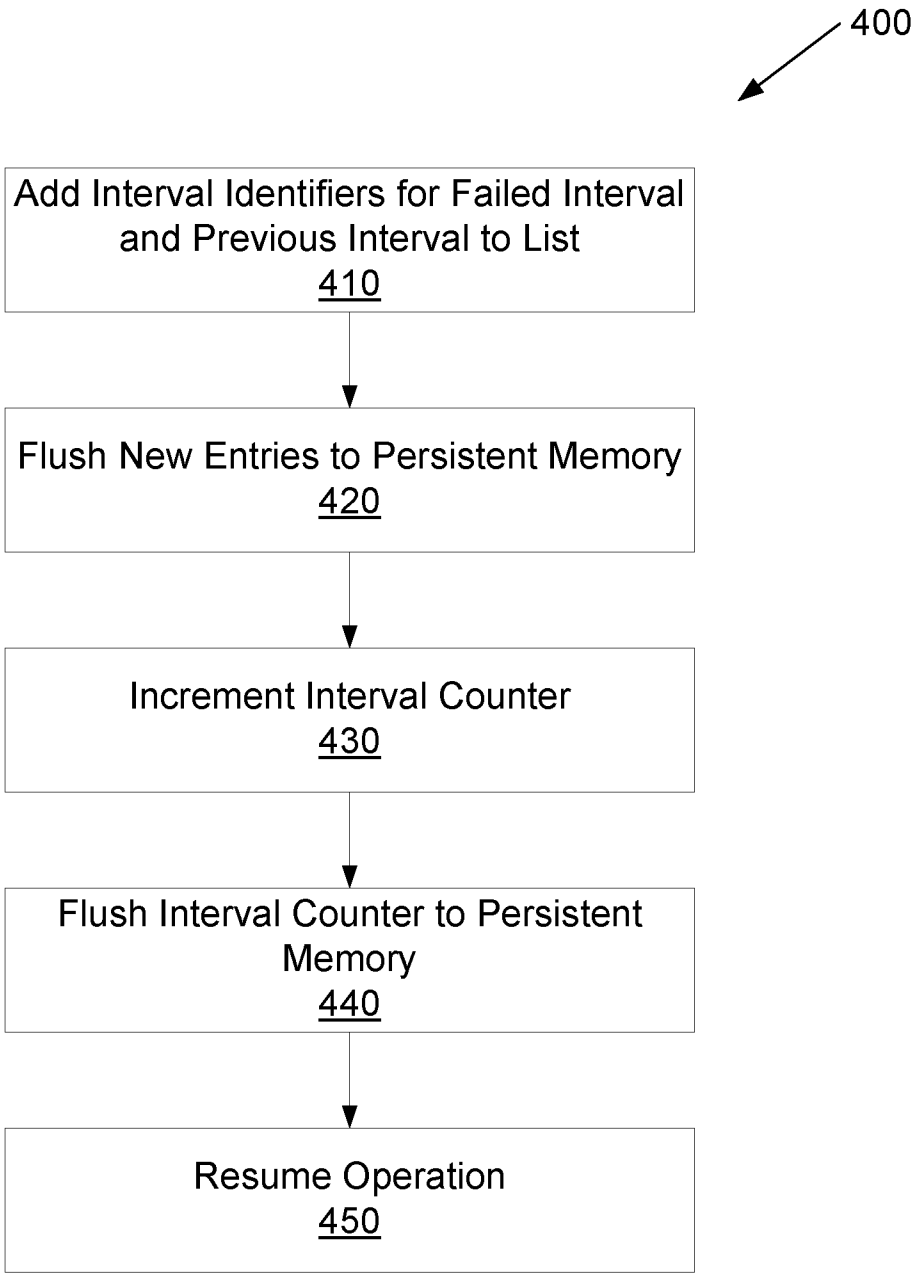
400
Add Interval Identifiers for Failed Interval and Previous Interval to List
410
Flush New Entries to Persistent Memory
420
Increment Interval Counter
430
Flush Interval Counter to Persistent Memory
440
Resume Operation
450

Figure 4

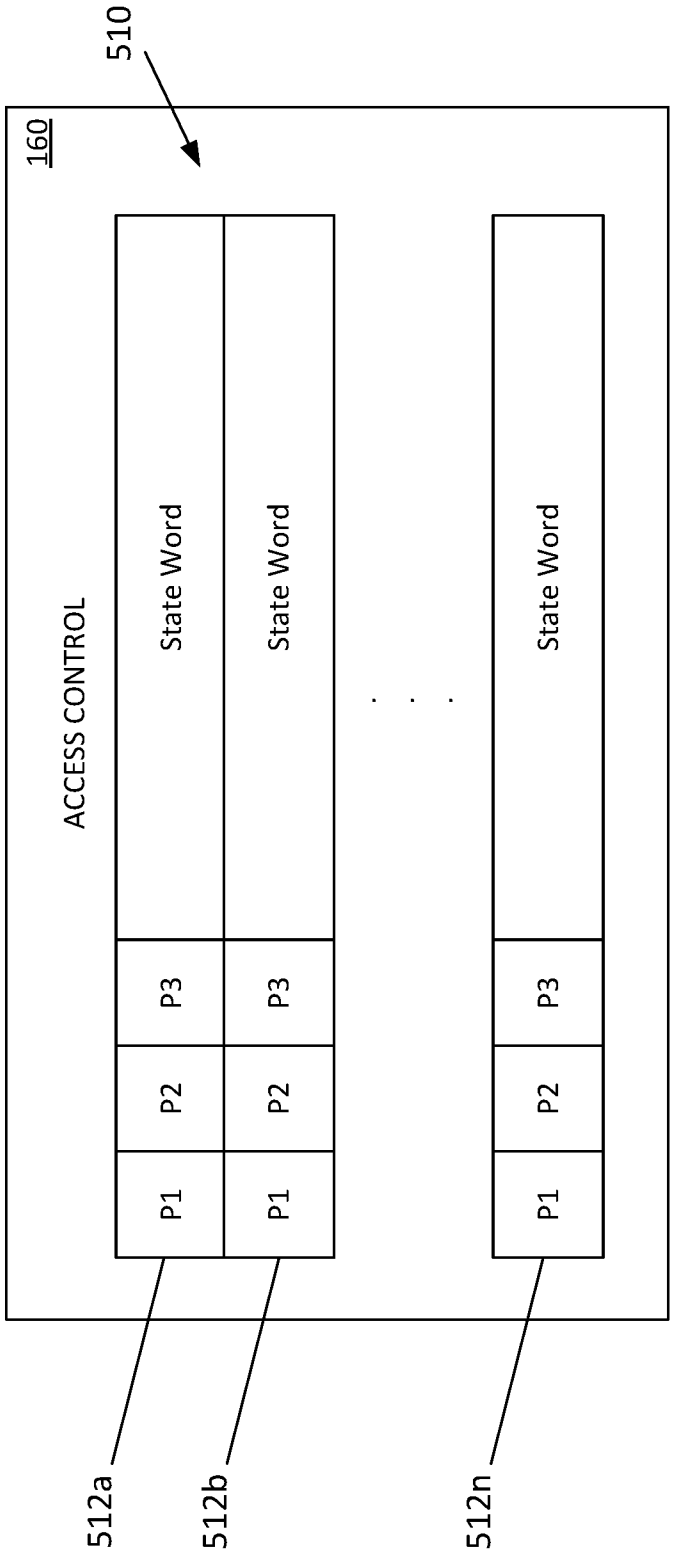
510
160
ACCESS CONTROL
State Word
State Word
State Word
P3
P2
P1
P3
P2
P1
P3
P2
P1
512a
512b
512n

Figure 5

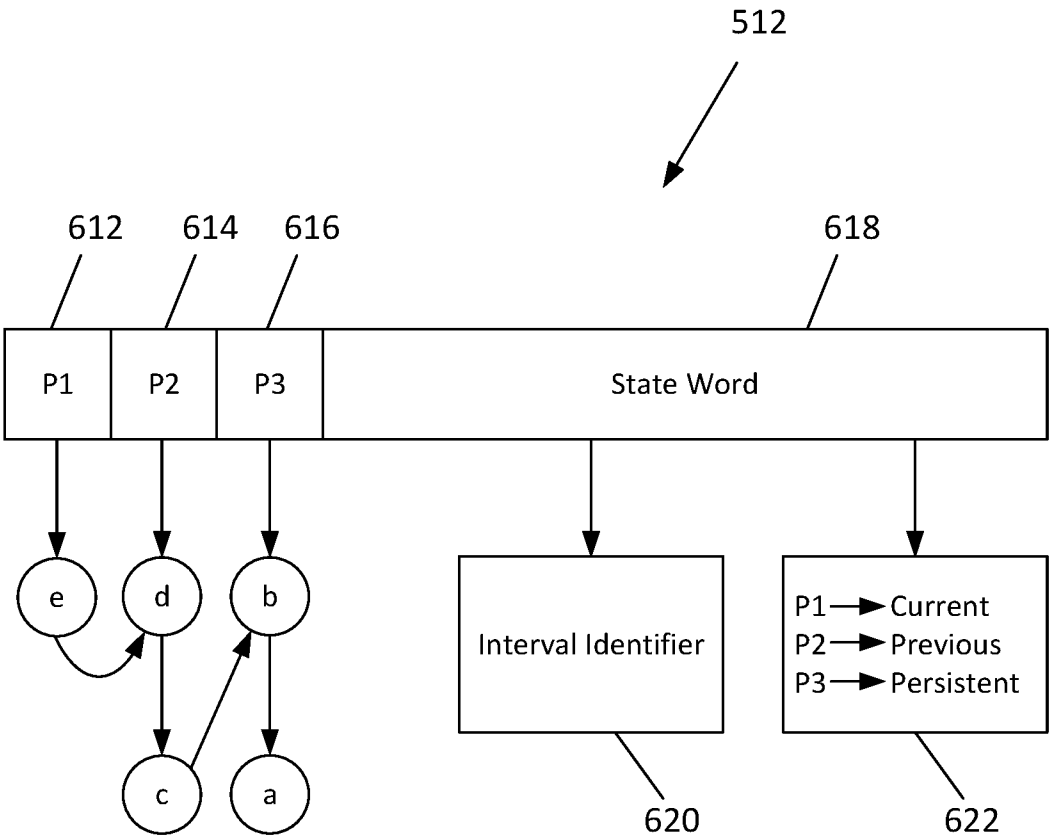
512
612
614
616
618
P1
P2
P3
State Word
e
d
b
c
a
Interval Identifier
P1 → Current
P2 → Previous
P3 → Persistent
620
622

Figure 6

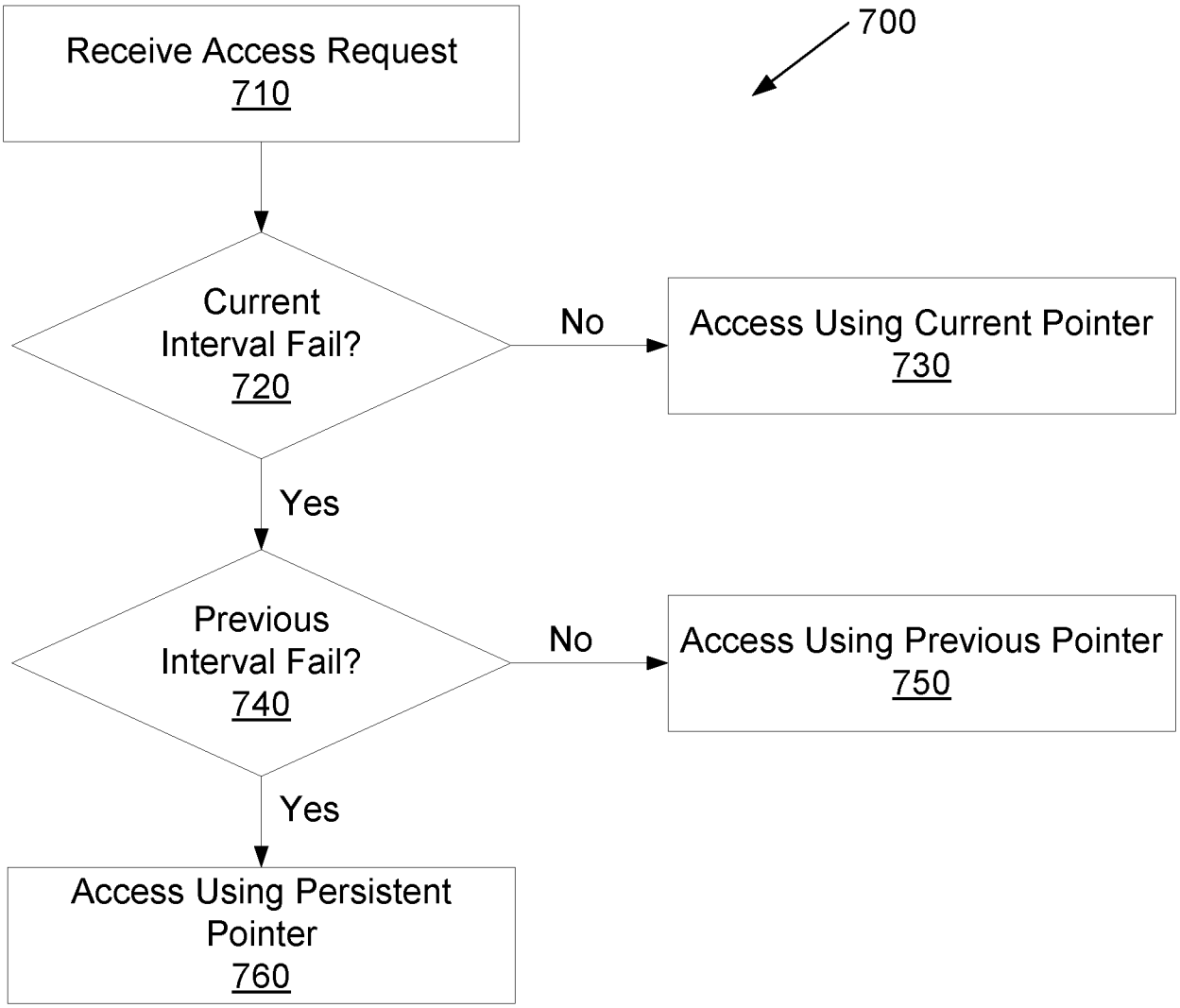
700
Receive Access Request
710
Current Interval Fail?
720
No
Access Using Current Pointer
730
Yes
Previous Interval Fail?
740
No
Access Using Previous Pointer
750
Yes
Access Using Persistent Pointer
760

Figure 7

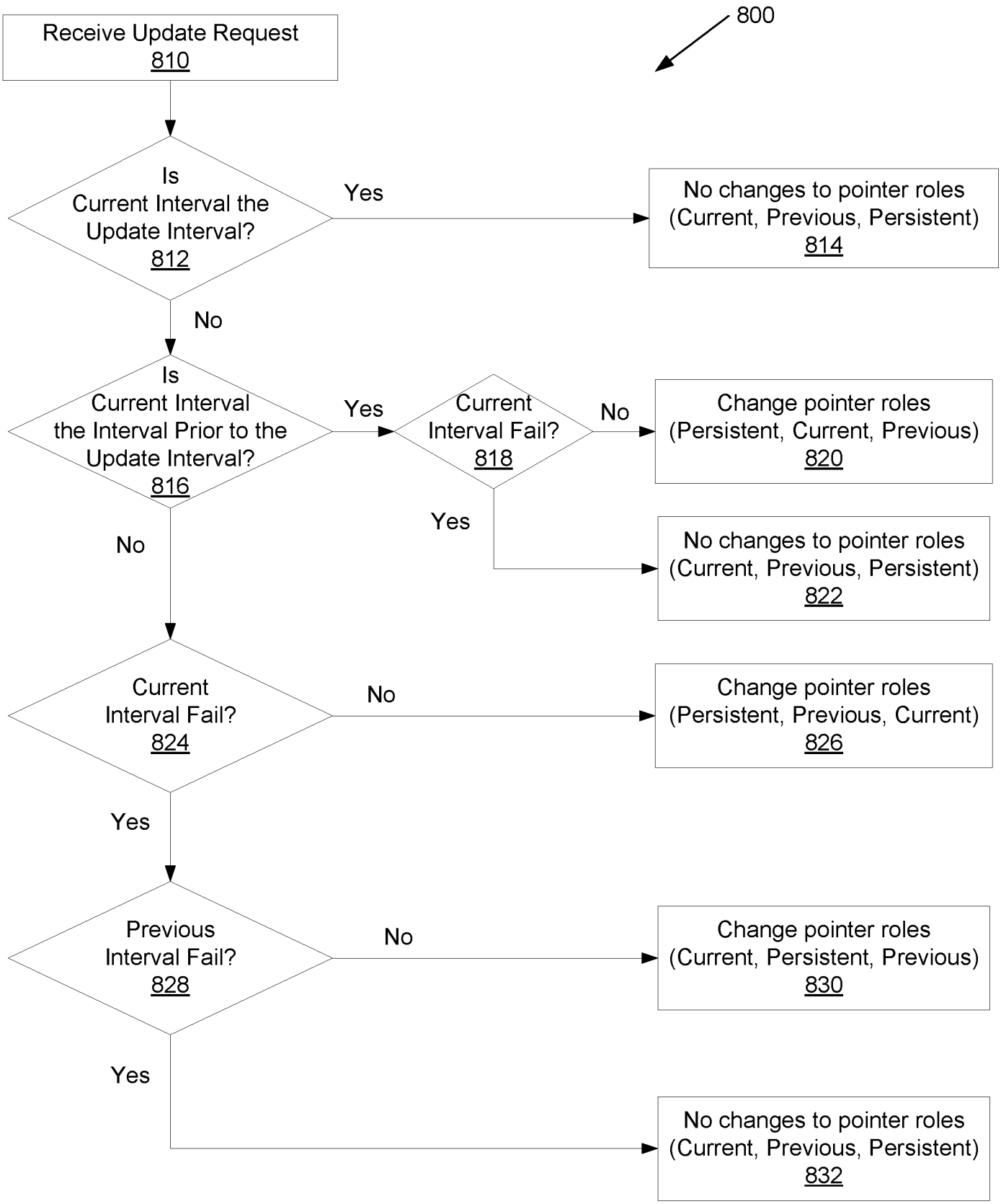
800
Receive Update Request
810
Is
Current Interval the
Update Interval?
812
Yes
No changes to pointer roles
(Current, Previous, Persistent)
814
No
Is
Current Interval
the Interval Prior to the
Update Interval?
816
Yes
Current
Interval Fail?
818
No
Change pointer roles
(Persistent, Current, Previous)
820
Yes
No changes to pointer roles
(Current, Previous, Persistent)
822
No
Current
Interval Fail?
824
No
Change pointer roles
(Persistent, Previous, Current)
826
Yes
Previous
Interval Fail?
828
No
Change pointer roles
(Current, Persistent, Previous)
830
Yes
No changes to pointer roles
(Current, Previous, Persistent)
832

Figure 8

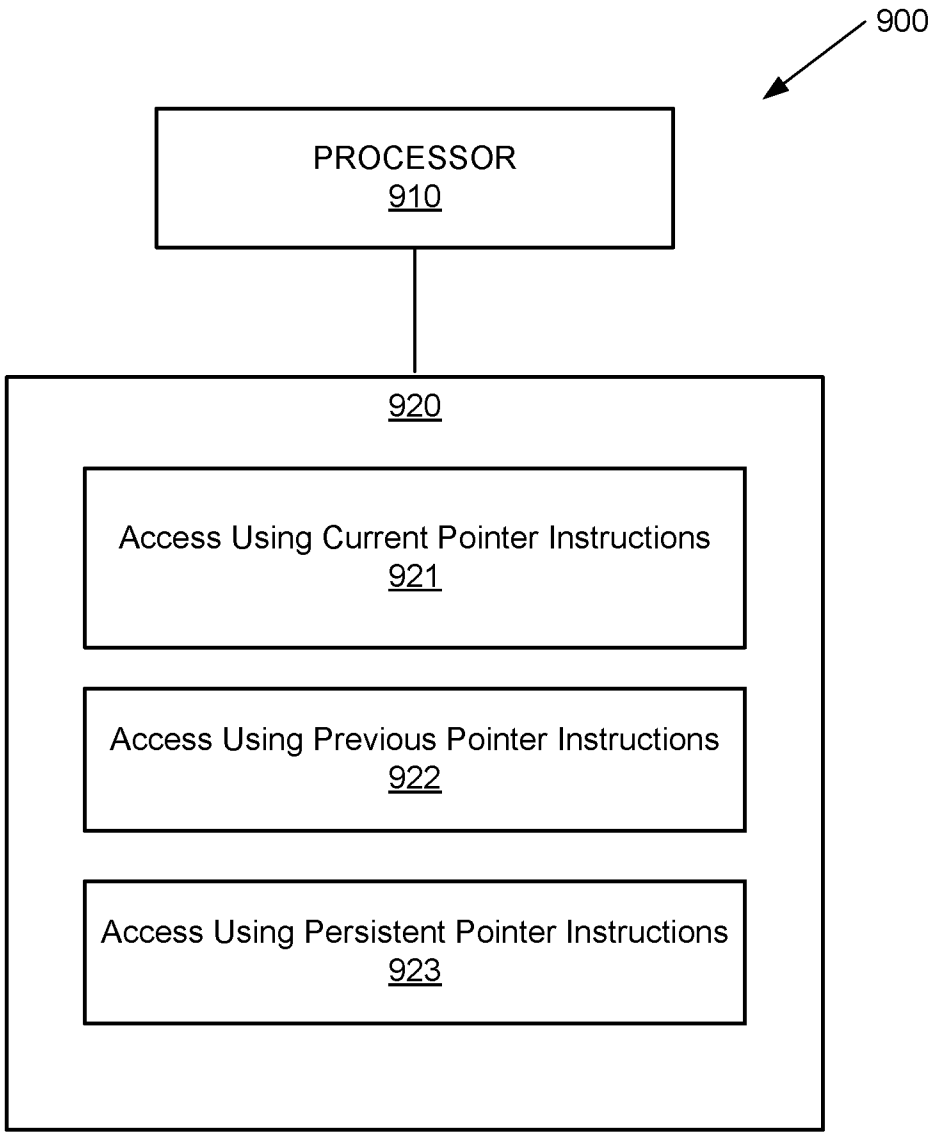
900
PROCESSOR
910
920
Access Using Current Pointer Instructions
921
Access Using Previous Pointer Instructions
922
Access Using Persistent Pointer Instructions
923

Figure 9

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2016/025368

A. CLASSIFICATION OF SUBJECT MATTER

G06F 12/0868(2016.01)i, G06F 12/0804(2016.01)i, G06F 12/0815(2016.01)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 12/0868; G06F 12/16; G06F 17/30; G06F 11/14; G06F 11/00; G06F 12/08; G06F 12/0804; G06F 12/0815

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: transient memory, persistent memory, whole cache flushing, uncorrupted data, interval, failure, bucket, pointer

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
Y	US 2009-0182784 A1 (JAIN ROHIT et al.) 16 July 2009 See paragraphs [0023]-[0074]; and figures 1-6.	1-15
Y	US 7055055 B1 (ERIC D. SCHNEIDER et al.) 30 May 2006 See column 2, lines 18-24; and column 10, lines 52-54.	1-15
A	US 2009-0100230 A1 (WAI LAM) 16 April 2009 See paragraphs [0008], [0050]-[0054]; and figures 6A-6D.	1-15
A	US 2007-0234101 A1 (TOBY JAMES KOKTAN et al.) 04 October 2007 See paragraphs [0037]-[0066]; and figure 1.	1-15
A	US 2015-0178171 A1 (INTERNATIONAL BUSINESS MACHINES CORPORATION) 25 June 2015 See paragraphs [0048]-[0059]; and figures 5A-7.	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
14 December 2016 (14.12.2016)	**15 December 2016 (15.12.2016)**

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/US2016/025368

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2009-0182784 A1	16/07/2009	US 7877360 B2	25/01/2011
US 7055055 B1	30/05/2006	AU 2000-44862 A1	10/11/2000
		EP 1090348 A1	11/04/2001
		EP 1090348 B1	22/12/2004
		JP 03984789 B2	03/10/2007
		JP 2002-543493 A	17/12/2002
		WO 00-65447 A1	02/11/2000
US 2009-0100230 A1	16/04/2009	US 2005-005070 A1	06/01/2005
		US 2007-113016 A1	17/05/2007
		US 2007-113017 A1	17/05/2007
		US 7165145 B2	16/01/2007
		US 7418547 B2	26/08/2008
		US 7467259 B2	16/12/2008
		US 8041892 B2	18/10/2011
US 2007-0234101 A1	04/10/2007	US 7734949 B2	08/06/2010
US 2015-0178171 A1	25/06/2015	US 2015-370654 A1	24/12/2015
		US 9158633 B2	13/10/2015