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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
27 June 2019 (27.06.2019)

WIPO | PCT

(10) International Publication Number
WO 2019/120464 A1

(51) International Patent Classification:
G06F 9/46 (2006.01)

(21) International Application Number:
PCT/EP2017/083219

(22) International Filing Date:
18 December 2017 (18.12.2017)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HUAWEI TECHNOLOGIES CO., LTD.** [CN/CN]; Huawei Administration Building Bantian, Longgang District, Shenzhen, Guangdong 518129 (CN).

(72) Inventor; and

(71) Applicant *(for US only)*: **AVNI, Hillel** [IL/DE]; c/o Huawei Technologies Duesseldorf GmbH, Riesstr.25, 80992 Munich (DE).

(72) Inventor: **AVITZUR, Aharon**; c/o Huawei Technologies Duesseldorf GmbH, Riesstr. 25, 80992 Munich (DE).

(74) Agent: **KREUZ, Georg**; Huawei Technologies Duesseldorf GmbH, Riesstr. 8, 80992 Munich (DE).

(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, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IR, IS, JO, JP, KE, KG, KH, KN, KP, KR, KW, 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).

Published:

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

(54) **Title:** SCALABLE HARDWARE TRANSACTIONAL MEMORY

(57) **Abstract:** An apparatus for accessing data synchronized by hardware transactional memory includes a hardware processor which, prior to performing an HTM transaction with hardware transactional memory (HTM), allocates nodes of a data structure for use during the transaction and performs the HTM transaction using the allocated nodes.

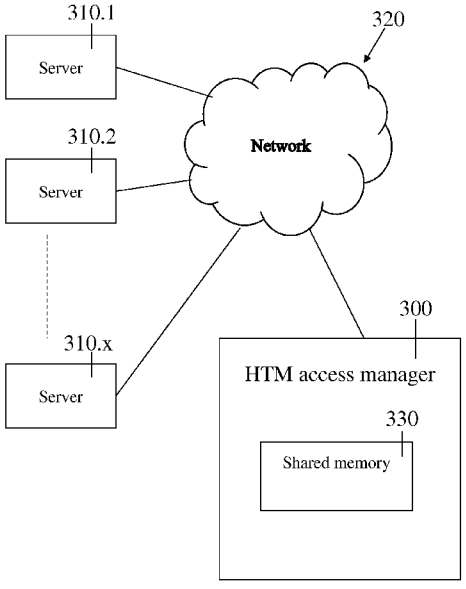

Fig. 3

WO 2019/120464 A1

SCALABLE HARDWARE TRANSACTIONAL MEMORY

BACKGROUND

The present invention, in some embodiments thereof, relates to hardware transactional memory and, more specifically, but not exclusively, to avoidance of conflicts in memories synchronized by hardware transactional memory.

Hardware transactional memory (HTM) is a hardware synchronization mechanism which enables parallel transactions that rarely conflict with each other to be efficiently executed without the need for using locks. HTM transactions may execute and commit changes to the data stored in the HTM, as long as there are no conflicts. When there is a conflict between concurrent transactions, the transactions abort.

The use of HTM with various types of data structures has been investigated. For example:

a) On a balanced tree such as an AVL tree, an insert or a delete may trigger log(n) rotations which make it not scalable with HTM.

b) Use of an adaptive radix tree (ART) with HTM is fast but not scalable because of contention in the memory allocation. An insert always triggers a memory allocation for a node. A delete always triggers a memory release of a node. The allocations and frees cause cache misses that abort the transactions. Such aborts occur relatively frequently in workloads with a high percentage of inserts and deletes, regardless of the contention on the data structure itself.

There has been research on the use of HTM friendly memory allocation (malloc). This research has only investigated the placement of allocated buffers relative to L1 cache associatively sets.

SUMMARY

HTM conflicts may arise due to memory allocations and frees during an HTM transaction. Embodiments of the invention avoid HTM conflicts that are due to memory allocations and frees by performing these memory management operations outside the HTM transaction itself. Data structure nodes that are required for performance of the HTM transaction (e.g. for insert and/or delete) are pre-allocated before the HTM transaction which wraps the actual insert and/or delete. Nodes released by the HTM transaction are freed after the HTM transaction is successfully completed. Optionally, nodes that are released by the insert and/or delete operations within the HTM transaction are buffered until they are freed after completion of the HTM transaction.

As used herein the terms "HTM transaction" and "transaction" mean a collection of operations that may execute and commit changes in the data structure in accordance with an HTM synchronization mechanism.

As used herein the term "memory management operation" means allocating and/or freeing a node in the data structure.

As used herein the term "outside the HTM transaction" means that the operation is performed either prior to beginning the HTM transaction or after successfully completing the HTM transaction.

As used herein the term "the node is released" and similar terms means that the node is marked as no longer being a valid node of the data structure. Nodes are released during performance of the HTM transaction.

As used herein the term "allocating a node" and similar terms means that the node is reserved for addition to the data structure if required by the HTM transaction.

As used herein the term "freeing a node" and similar terms means that the node is made available for future HTM transaction(s).

As used herein the term "data structure" means a specified way of organizing and storing data (such as a RADIX tree).

It is an object of the present invention to provide an apparatus, a system, a computer program product, and a method for avoiding conflicts and aborts within HTM transactions on a shared memory.

The foregoing and other objects are achieved by the features of the independent claims. Further implementation forms are apparent from the dependent claims, the description and the figures.

According to a first aspect, an apparatus for accessing data synchronized by hardware transactional memory includes a hardware processor for:

prior to performing a transaction with hardware transactional memory (HTM), allocating nodes of a data structure for use during the transaction; and

performing the transaction using the allocated nodes.

According to a second aspect, a method for accessing data synchronized by hardware transactional memory includes:

prior to performing a transaction with hardware transactional memory (HTM), allocating nodes of a data structure for use during the transaction; and

performing the transaction using the allocated nodes.

Because the nodes are pre-allocated before the HTM transaction begins, memory allocation operations (malloc) are not performed during the transaction and cannot conflict with any free functions which may occur during the HTM transaction.

In a further implementation form of the first and second aspects, data structure nodes released by the transaction are freed after transaction commit. This further reduces the memory management operations performed within the HTM transaction, reducing the complexity of the HTM transaction and efficiently releasing the shared memory for subsequent HTM transactions. Because the released nodes have not yet been freed, in case of an abort there is no need to re-allocate data structure nodes in their place.

In a further implementation form of the first and second aspects, nodes released by the transaction are buffered until successful commit of the transaction. Buffering the nodes assists in quick recovery of the state of the data structure before the HTM transaction was begun in cases of a transaction abort.

In a further implementation form of the first and second aspects, the HTM operates on a shared cache memory. Preventing aborts in transactions on a cache memory improves the accessibility and speed of the cache memory, which is key to efficient operation of processor(s) using the cache memory.

In a further implementation form of the first and second aspects, the data structure is a RADIX tree. The RADIX tree is an important data structure often used for accessing databases. Embodiments of the invention make the RADIX tree data structure scalable in HTM.

In a further implementation form of the first and second aspects, the transaction is an atomic transaction. Atomic transactions are very useful in concurrent programming where multiple processors try to modify the same shared data structure. The HTM makes their changes consistent.

In a further implementation form of the first and second aspects, the transaction includes an abort mechanism utilized in the case of a missed access to the data structure. Although removing the memory management operations outside of the HTM transaction greatly reduces the likelihood of an aborted transaction, such aborted transaction may nonetheless occur. The abort mechanism may assist in such cases by returning the data structure to its state prior to beginning the transaction, possibly using the buffered nodes.

In a further implementation form of the first aspect, the apparatus further includes an HTM module adapted to perform the data synchronization. A dedicated HTM module is particularly suitable for processor architectures which use dedicated modules or units (e.g. arithmetic-logic unit, load-store unit, etc.) to reduce the idle time of CPU components.

In a further implementation form of the first aspect, the hardware processor executes multiple threads performing concurrent transactions with HTM on a shared memory storing the data structure.

In a further implementation form of the first aspect, the apparatus includes multiple hardware processors performing concurrent transactions with HTM on a shared memory storing the data structure.

Unless otherwise defined, all technical and/or scientific terms used herein have the same meaning as commonly understood by one of ordinary skill in the art to which the invention pertains. Although methods and materials similar or equivalent to those described herein can be used in the practice or testing of embodiments of the invention, exemplary methods and/or materials are described below. In case of conflict, the patent specification, including definitions, will control. In addition, the materials, methods, and examples are illustrative only and are not intended to be necessarily limiting.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWINGS

Some embodiments of the invention are herein described, by way of example only, with reference to the accompanying drawings. With specific reference now to the drawings in detail, it is stressed that the particulars shown are by way of example and for purposes of illustrative discussion of embodiments of the invention. In this regard, the description taken with the drawings makes apparent to those skilled in the art how embodiments of the invention may be practiced.

In the drawings:

Figs. 1A-1C are simplified block diagrams of an HTM access manager, according to respective embodiments of the invention;

Fig. 2 is a simplified block diagram of a server with HTM access manager, according to embodiments of the invention;

Fig. 3 is a simplified diagram of an HTM access manager accessed by servers over a network, according to embodiments of the invention;

Figs. 4, 5 and 6 are simplified flowcharts of methods for accessing data synchronized by hardware transactional memory, according to respective embodiments of the invention;

Fig. 7 is a simplified flowchart of a method for accessing data from an adaptive RADIX tree, according to exemplary embodiments of the invention; and

Figs 8 and 9 are graphs demonstrating the improved speed and scalability of an embodiment of the invention relative to prior art HTM and Skiplist.

DETAILED DESCRIPTION

The present invention, in some embodiments thereof, relates to hardware transactional memory and, more specifically, but not exclusively, to avoidance of conflicts in memories synchronized by hardware transactional memory.

HTM transactions often require allocation and/or freeing of one or more nodes of the data structure upon which the HTM transaction is performed. As described above, the memory management operations required to allocate or to free nodes within the HTM transaction may cause misses that abort the HTM transaction.

Embodiments of the invention move some or all of the memory management operations outside of the HTM transaction. Memory management operations may be moved out of the HTM transaction by one or both of:

i) Pre-allocating memory for use during the HTM transaction before starting the HTM transaction. The HTM transaction uses the pre-allocated memory when a node is created. Thus the memory allocation operations are not performed during the HTM transaction.

ii) Freeing data structure nodes that are released by the HTM transaction after the HTM transaction commits. Thus the memory free operations are not performed during the HTM transaction.

Moving the memory allocations and frees out of the HTM transaction significantly reduces the likelihood of HTM transaction aborts, as shown in Figs. 8-9 and discussed in more detail below.

HTM transactions are atomic. The atomicity of the HTM transaction protects the data structure from concurrent access by another HTM transaction, further limiting the probability of an abort of the current HTM transaction. Embodiments of the invention increase the probability that an HTM transaction will commit successfully to support the atomicity.

Before explaining at least one embodiment of the invention in detail, it is to be understood that the invention is not necessarily limited in its application to the details of construction and the arrangement of the components and/or methods set forth in the following description and/or illustrated in the drawings and/or the Examples. The invention is capable of other embodiments or of being practiced or carried out in various ways.

The present invention may be a system, a method, and/or a computer program product. The computer program product may include a computer readable storage medium (or media) having computer readable program instructions thereon for causing a processor to carry out aspects of the present invention.

The computer readable storage medium can be a tangible device that can retain and store instructions for use by an instruction execution device. The computer readable storage medium may be, for example, but is not limited to, an electronic storage device, a magnetic storage device, an optical storage device, an electromagnetic storage device, a semiconductor storage device, or any suitable combination of the foregoing.

Computer readable program instructions described herein can be downloaded to respective computing/processing devices from a computer readable storage medium or to an external computer or external storage device via a network, for example, the Internet, a local area network, a wide area network and/or a wireless network.

The computer readable program instructions may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through any type of network, including a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the Internet using an Internet Service Provider). In some embodiments, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGA), or programmable logic arrays (PLA) may execute the computer readable program instructions by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present invention.

Aspects of the present invention are described herein with reference to flowchart illustrations and/or block diagrams of methods, apparatus (systems), and computer program products according to embodiments of the invention. It will be understood that each block of the flowchart illustrations and/or block diagrams, and combinations of blocks in the flowchart illustrations and/or block diagrams, can be implemented by computer readable program instructions.

The flowchart and block diagrams in the Figures illustrate the architecture, functionality, and operation of possible implementations of systems, methods, and computer program products according to various embodiments of the present invention. In this regard, each block in the flowchart or block diagrams may represent a module, segment, or portion of instructions, which comprises one or more executable instructions for implementing the specified logical function(s). In some alternative implementations, the functions noted in the block may occur out of the order noted in the figures. For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse

order, depending upon the functionality involved. It will also be noted that each block of the block diagrams and/or flowchart illustration, and combinations of blocks in the block diagrams and/or flowchart illustration, can be implemented by special purpose hardware-based systems that perform the specified functions or acts or carry out combinations of special purpose hardware and computer instructions.

HTM access manager

Reference is now made to Figs. 1A-1C, which are simplified block diagrams of an HTM access manager, according to respective embodiments of the invention. HTM access manager 100 manages access to a shared memory which is synchronized by HTM. For clarity, Figs. 1A-1C illustrate respective non-limiting embodiments in which there is a single shared memory. In alternate embodiments of the invention, the HTM access manager may perform HTM transactions on multiple shared memories, internal and/or external to HTM access manager.

Optionally the shared memory is a shared cache memory of the hardware processor (such as the L1 cache).

HTM access manager 100 includes at least one processor 110, and optionally a memory storing code instructions 120 to be executed by processor 110.

Optionally HTM access manager 100 includes an HTM module (not shown) which performs the HTM transaction on the shared memory. Further optionally, the HTM module performs the pre-allocating and/or releasing of nodes outside the HTM transaction.

Prior to performing an HTM transaction on a data structure stored on a shared memory, HTM access manager 100 allocates nodes of the data structure for use during the transaction (denoted herein pre-allocated nodes). The HTM transaction is performed using the pre-allocated nodes.

Optionally the data structure has more than one type of node, and multiple types of node are pre-allocated for use during the transaction.

Optionally, information about how many nodes to pre-allocate for each type of node is derived from the HTM transaction algorithm. Typically, an HTM transaction will use at most one node from one of four different sizes per insert/delete operation.

Optionally, after the HTM transaction is performed (e.g. after commit) HTM access manager 100 frees data structure nodes which were released by the transaction. Further optionally, the released nodes are freed prior to the next HTM transaction on the shared memory.

Optionally, the released nodes are buffered until the HTM transaction is successful committed.

Optionally, during the HTM transaction the released nodes are marked (e.g. added to a to-be freed list). Thus the released nodes are known and are easily freed when the HTM transaction commits.

Optionally the data structure is a RADIX tree (also denoted radix tree or compact prefix tree). A RADIX tree represents a space-optimized tree in which each node that is the only child of a parent node is merged with its parent. RADIX trees are in the heart of many applications. Specifically the adaptive RADIX tree (ART) is used as an index in databases.

In a RADIX tree, each insert or delete operation allocates and/or deletes at least one node to build the new node and new path. The insert or delete operation writes one cache line as the last step of the operation so contention on the cache which may abort HTM is unlikely. This lack of contention makes RADIX trees very suitable for use with HTM.

HTM access manager 100 performs memory management (i.e. node pre-allocation and/or free) outside the HTM transaction thus preventing aborts due to misses to the shared memory during the transaction. This yields the benefit of making the very useful RADIX tree data structure scalable in HTM. The RADIX tree data structure is particularly beneficial for keys which are long strings.

Alternate embodiments of the invention may use a different type of data structure, for example an AVL tree.

Optionally, the HTM transaction includes an abort mechanism which is utilized in the case of a missed access to the data structure. Further optionally, the abort mechanism frees the pre-allocated nodes for use by a different transaction.

Optionally, HTM access manager 100 includes at least one processor that executes multiple threads which perform concurrent HTM transactions on shared memory 130. HTM access manager 100 performs memory management operations outside the HTM transactions issued by respective threads, thereby preventing aborts due to concurrent accesses to the data structure by multiple threads.

Optionally, HTM access manager 100 includes multiple processors, which perform concurrent HTM transactions on the shared memory. HTM access manager 100 performs memory management operations outside the HTM transactions issued by respective processors, thereby preventing aborts due to concurrent accesses to the data structure by multiple processors.

Figs. 1A-1B illustrate respective embodiments in which shared memory 130 is an internal memory of HTM access manager 100. In Fig. 1A, HTM access manager 100 includes a single processor 110, so that concurrent accesses to shared memory 130 may arise from parallel threads executing in processor 110.

In Fig. 1B, HTM access manager 140 includes multiple processors 110.1-110.m, in which case concurrent accesses to shared memory 130 may arise from concurrent HTM transactions by multiple processors and/or parallel threads executing in a given processor or processors.

Fig. 1C illustrates an embodiment in which the shared memory is external to HTM access manager 150. HTM access manager 150 includes interface 160 with which it communicates with the external shared memory.

Reference is now made to Fig. 2, which is a simplified block diagram of a server with HTM access manager, according to embodiments of the invention. Server 210 includes HTM access manager 220 and shared memory 230. Concurrent accesses to shared memory 230 may originate at multiple endpoints 200.1-200.n. HTM access manager 220 performs memory management operations outside the HTM transactions which result from memory access operations by endpoints 200.1-200.n, thereby preventing aborts due to concurrent accesses to shared memory 230 by endpoints 200.1-200.n.

Reference is now made to Fig. 3, which is a simplified diagram of an HTM access manager accessed by servers over a network, according to embodiments of the invention. In the embodiment of Fig. 3, HTM access manager 300 includes shared memory 330. Servers 310.1-310.x access shared memory 330 by communicating with HTM access manager 300 over network 320. HTM access manager 300 performs memory management operations outside the HTM transactions which result from memory access operations by servers 310.1-310.x, thereby preventing aborts due to concurrent accesses to shared memory 330 by the multiple servers.

Method for accessing data synchronized by HTM

Reference is now made to Figs. 4-6, which are simplified flowcharts of a method for accessing data synchronized by hardware transactional memory, according to respective embodiments of the invention. As described above, nodes required for the HTM transaction are pre-allocated before the HTM transaction is started. Optionally, nodes released during the HTM transaction are freed after the HTM transaction ends.

Referring to Fig. 4, prior to performing an HTM transaction, in 410 data structure nodes are pre-allocated for use during the HTM transaction. In 420 the HTM transaction is performed using the pre-allocated nodes.

Optionally, in 430 the nodes of the data structure released by the transaction are freed after the HTM transaction ends.

Optionally, in 440 the nodes released by the HTM transaction are buffered until after the transaction is performed (e.g. after HTM transaction commit).

Optionally, the data structure is a RADIX tree.

Optionally, the HTM transaction includes an abort mechanism utilized in the case of a missed access.

Other implementations of the method may include some or all of the abovementioned optional features of the HTM access manager, such as a different type of data structure and/or different types of concurrent access to the shared memory.

Referring to Fig. 5, in 510 memory is allocated for all data structure nodes that might be used by insert(s) and/or delete(s) in the upcoming HTM transaction. In steady state this implies replacing consumed nodes. In 520 nodes released by earlier transaction(s) are freed. 510 and 520 take place before beginning the HTM transaction in 530. In 540-550 insert and/or delete operation(s) are performed using the pre-allocated memory for new nodes and buffering released nodes without freeing their memory. In 560 the HTM transaction ends.

In Fig. 6 all the memory management which has inherent contention (and which would prevent HTM scaling) is executed outside the HTM context by using two techniques:

a) Pre-allocating all possible required data structure nodes (use standard memory allocation to prepare the potentially allocated nodes BEFORE starting HTM).

b) Post-transaction, freeing the nodes which were released inside the HTM transaction after the HTM transaction commits.

Note that the HTM transaction begins at 630, is executed in 640 and is committed in 650. 610, 620 and 660 all occur outside the HTM transaction.

In 610 an insert or delete operation begins.

If there are not enough pre-allocated nodes of each node size, in 620 all possibly needed nodes are pre-allocated using standard memory allocation before starting the HTM transaction in 630. If there are enough allocated buffers to perform the insert or delete without memory allocation during the HTM transaction, the HTM transaction starts directly in 630.

The HTM transaction is executed in 640. New data structure nodes (e.g. RADIX tree) are taken from the pre-allocated nodes. Nodes to be freed (i.e. released nodes) are marked and buffered into post released pools. The HTM transaction is committed in 650.

In 660, the buffered nodes are freed.

Reference is now made to Fig. 7, which is a simplified flowchart of a method for accessing data from an adaptive RADIX tree, according to exemplary embodiments of the invention. Exemplary code for implementing the method is presented below.

In the embodiments of Fig. 7, the data structure is an adaptive RADIX tree with path compression (ART). The ART data structure requires an insert and/or a delete operation to allocate and/or free nodes from a known, small set of different sized nodes.

Optionally, the ART has four type of nodes, with fanouts of: 4, 16, 48 and 256. An insert or delete may need to allocate at most one of these nodes and may free a (typically small) number of nodes.

In 710 the HTM transaction starts.

The ART operation starts in 720. If a node needs to be freed (i.e. is released), in 730 it is added to the local to-be freed list. If a node of a given type (T) is needed, in 740 the type T node is pre-allocated using the alloc-node(T) function presented below.

In 750 the HTM transaction is comitted.

In 760 the refresh-new function presented below is used to fill the allocated node type, if needed. If nodes were released by the HTM transaction, the local to-be freed list is freed in 770.

Fig. 7 shows a non-limiting embodiment in which 730 and 740 are both performed in parallel. In alternate implementations 730 and 740 may be performed in series. Similarly, Fig. 7 shows a non-limiting embodiment in which 760 and 770 are both performed in parallel. In alternate implementations 760 and 770 may be performed in series.

Optionally, the HTM is a hardware block which is exposed by the architecture, e.g. x86, by 2 ISA: HTM_BEGIN and HTM_END. If a block of code B is executed between these ISA instructions, it is atomic, i.e. if in B, a processor P1 writes an address that was read by a concurrent execution of B on processor P2, and the concurrent execution will be aborted, i.e. canceled. An HTM conflict occurs when two (or more) concurrent transactions access the same memory address and one of them is a write. As a result, either the writer or the readers are aborted.

Following is exemplary code for performing the functions presented in Fig. 7. Each ART operation (e.g. insert/delete/lookup) is wrapped with an HTM transaction.

An exemplary embodiment of code for an insert operation according is:

```
void *art_insert_os(art_tree *t, const unsigned char
*key, int key_len, void *value)
{
    void *rc;
    htmbegin(&global_fallback_lock);
    rc = _art_insert_os(t, key, key_len, value);
    htmend(&global_fallback_lock);
    return rc;
}
```

The above insert operation makes the ART concurrent, but not scalable. The reason for unscalability is that malloc and free functions, when executed inside HTM, generate conflict aborts. To prevent the conflict aborts, the refresh-new function presented below allocates all the nodes that may be required by an insert or delete locally in the thread, and out of HTM context. In addition, the refresh-new function frees all the buffers that were released in the HTM transaction out of the HTM context. When releasing in the HTM transaction, the to-be freed buffer is only registered for freeing after the HTM transaction commit:

```
static void refresh-new(){
    if(new_n4 == NULL)
   _    n4   = calloc(1, sizeof(art_node4));
    if(new_n16 == NULL)
       _n16  = calloc(1, sizeof(art_node16));
    if(new_n48 == NULL)
        n48  = calloc(1, sizeof(art_node48));
    if(new_n256 == NULL)
        n256 = calloc(1, sizeof(art_node256));
    if(new_l == NULL)
        l    = malloc(sizeof(art_leaf)+32);
    while(there is a buffer n to free) free_n;
}
```

All potentially required nodes 4, 16, 48 and 256 fanout nodes are pre-allocated.

Inside the HTM transaction, when a node is required it is allocated from the local pre-allocated nodes, according to its type, by the alloc-node function:

```
static art_node* alloc-node(uint8_t type) {
    art_node* n;
    switch (type) {
        case NODE4:
            n = new_n4;
            new_n4 = NULL;
            break;
        case NODE16:
            n = new_n16;
            new_n16 = NULL;
            break;
        case NODE48:
            n = new_n48;
            new_n48 = NULL;
            break;
        case NODE256:
            n = new_n256;;
            new_n256 = NULL;
            break;
        default:
            abort();
    }
    n->type = type;
    return n;
}
```

Upon releasing in HTM context, the local to-be-freed buffer is added to the to-be-freed buffers pool, which are freed out of HTM context:

```
Add(localtobefreed, n);
```

Reference is now made to Figs. 8 and 9 which are graphs comparing the performance of a typical database workload with concurrent indexing. The indexes compared are:

i) ART with HTM (i.e. prior art original ART code and labeled ART-HTM-ORIG in Figs. 8-9);

ii) ART with HTM implemented in accordance with an embodiment of the invention (i.e. HTM with no memory management in the transaction and labeled ART-HTM-NO-MM in Figs. 8-9); and

iii) Skiplist (a concurrent ordered map known in the art).

Fig. 8 demonstrates that embodiments of ART-HTM-NO-MM are scalable, whereas the original ART code (ART-HTM-ORIG) is not scalable.

Fig. 9 demonstrates that the Lookups only workload is the same for both ART-HTM-NO-MM and ART-HTM-ORIG.

In summary, Figs. 8 and 9 demonstrate that the ART-HTM-NO-MM embodiment of the invention is faster than Skiplist and faster and more scalable than standard HTM.

Other systems, methods, features, and advantages of the present disclosure will be or become apparent to one with skill in the art upon examination of the following drawings and detailed description. It is intended that all such additional systems, methods, features, and advantages be included within this description, be within the scope of the present disclosure, and be protected by the accompanying claims.

The descriptions of the various embodiments of the present invention have been presented for purposes of illustration, but are not intended to be exhaustive or limited to the embodiments disclosed. Many modifications and variations will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the described embodiments. The terminology used herein was chosen to best explain the principles of the embodiments, the practical application or technical improvement over technologies found in the marketplace, or to enable others of ordinary skill in the art to understand the embodiments disclosed herein.

It is expected that during the life of a patent maturing from this application many relevant HTMs, HTM transactions, processors, shared memories and data structures will be developed and the scope of the terms HTM, HTM transaction, processor, shared memory and data structure is intended to include all such new technologies a priori.

As used herein the term “about” refers to ± 10 %.

The terms “comprises”, “comprising”, “includes”, “including”, “having” and their conjugates mean “including but not limited to”. This term encompasses the terms “consisting of” and “consisting essentially of”.

The phrase “consisting essentially of” means that the composition or method may include additional ingredients and/or steps, but only if the additional ingredients and/or steps do not materially alter the basic and novel characteristics of the claimed composition or method.

As used herein, the singular form “a”, “an” and “the” include plural references unless the context clearly dictates otherwise. For example, the term “a compound” or “at least one compound” may include a plurality of compounds, including mixtures thereof.

The word “exemplary” is used herein to mean “serving as an example, instance or illustration”. Any embodiment described as “exemplary” is not necessarily to be construed as

preferred or advantageous over other embodiments and/or to exclude the incorporation of features from other embodiments.

The word "optionally" is used herein to mean "is provided in some embodiments and not provided in other embodiments". Any particular embodiment of the invention may include a plurality of "optional" features unless such features conflict.

Throughout this application, various embodiments of this invention may be presented in a range format. It should be understood that the description in range format is merely for convenience and brevity and should not be construed as an inflexible limitation on the scope of the invention. Accordingly, the description of a range should be considered to have specifically disclosed all the possible subranges as well as individual numerical values within that range. For example, description of a range such as from 1 to 6 should be considered to have specifically disclosed subranges such as from 1 to 3, from 1 to 4, from 1 to 5, from 2 to 4, from 2 to 6, from 3 to 6 etc., as well as individual numbers within that range, for example, 1, 2, 3, 4, 5, and 6. This applies regardless of the breadth of the range.

Whenever a numerical range is indicated herein, it is meant to include any cited numeral (fractional or integral) within the indicated range. The phrases "ranging/ranges between" a first indicate number and a second indicate number and "ranging/ranges from" a first indicate number "to" a second indicate number are used herein interchangeably and are meant to include the first and second indicated numbers and all the fractional and integral numerals therebetween.

It is appreciated that certain features of the invention, which are, for clarity, described in the context of separate embodiments, may also be provided in combination in a single embodiment. Conversely, various features of the invention, which are, for brevity, described in the context of a single embodiment, may also be provided separately or in any suitable subcombination or as suitable in any other described embodiment of the invention. Certain features described in the context of various embodiments are not to be considered essential features of those embodiments, unless the embodiment is inoperative without those elements.

All publications, patents and patent applications mentioned in this specification are herein incorporated in their entirety by reference into the specification, to the same extent as if each individual publication, patent or patent application was specifically and individually indicated to be incorporated herein by reference. In addition, citation or identification of any reference in this application shall not be construed as an admission that such reference is available as prior art to the present invention. To the extent that section headings are used, they should not be construed as necessarily limiting.

CLAIMS

1. An apparatus for accessing data synchronized by hardware transactional memory, comprising a hardware processor (110) for:

prior to performing a transaction with hardware transactional memory (HTM), allocating nodes of a data structure for use during the transaction; and

performing the transaction using the allocated nodes.

2. An apparatus according to claim 1, wherein the hardware processor (110) is further for:

freeing nodes of the data structure released by the transaction after transaction commit.

3. An apparatus according to claim 2, wherein the hardware processor (110) is further for buffering the nodes released by the transaction until successful commit of the transaction.

4. An apparatus according to any one of claims 1-3, wherein the HTM operates on a shared cache memory of the hardware processor.

5. An apparatus according to any one of claims 1-4, further comprising an HTM module adapted to perform the data synchronization.

6. An apparatus according to any one of claims 1-5, wherein the data structure is a RADIX tree.

7. An apparatus according to any one of claims 1-6, wherein the transaction comprises an abort mechanism utilized in the case of a missed access to the data structure.

8. An apparatus according to any one of claims 1-7, wherein the hardware processor (110) executes a plurality of threads performing concurrent transactions with HTM on a shared memory (130) storing the data structure.

9. An apparatus according to any one of claims 1-7, wherein the apparatus comprises a plurality of hardware processors (110.1-110.n) performing concurrent transactions with HTM on a shared memory (130) storing the data structure.

10. A method for accessing data synchronized by hardware transactional memory, comprising:

prior to performing a transaction with hardware transactional memory (HTM), allocating nodes of a data structure for use during the transaction; and

performing the transaction using the allocated nodes.

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

after performing the transaction, freeing nodes of the data structure released by the transaction.

12. A method according to claim 11, further comprising buffering the nodes released by the transaction until successful commit of the transaction.

13. A method according to any one of claims 10-12, wherein the data structure is a RADIX tree.

14. A method according to any one of claims 10-13, wherein the transaction comprises an abort mechanism utilized in the case of a missed access to the data structure.

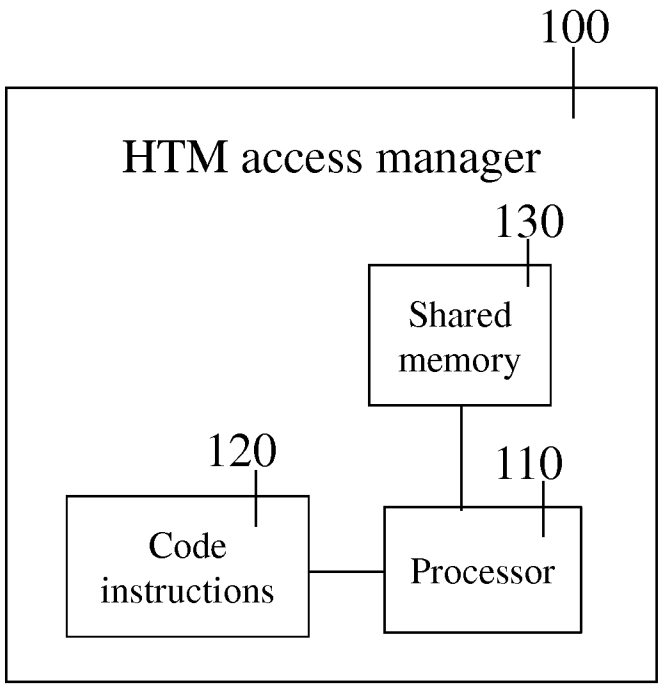
100
HTM access manager
130
Shared memory
120
110
Code instructions
Processor

Fig. 1A

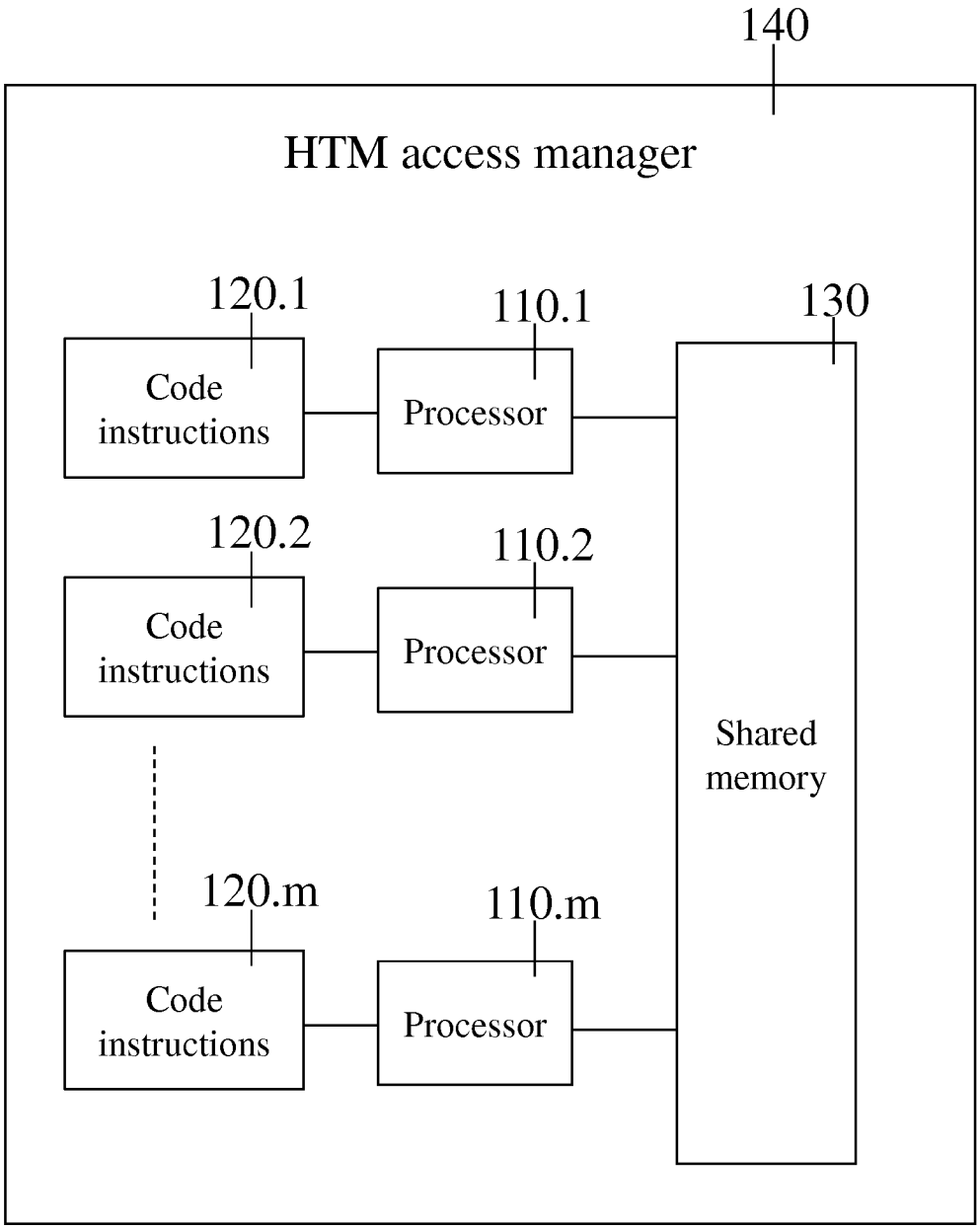
140
HTM access manager
120.1
110.1
130
Code instructions
Processor
120.2
110.2
Code instructions
Processor
Shared memory
120.m
110.m
Code instructions
Processor

Fig. 1B

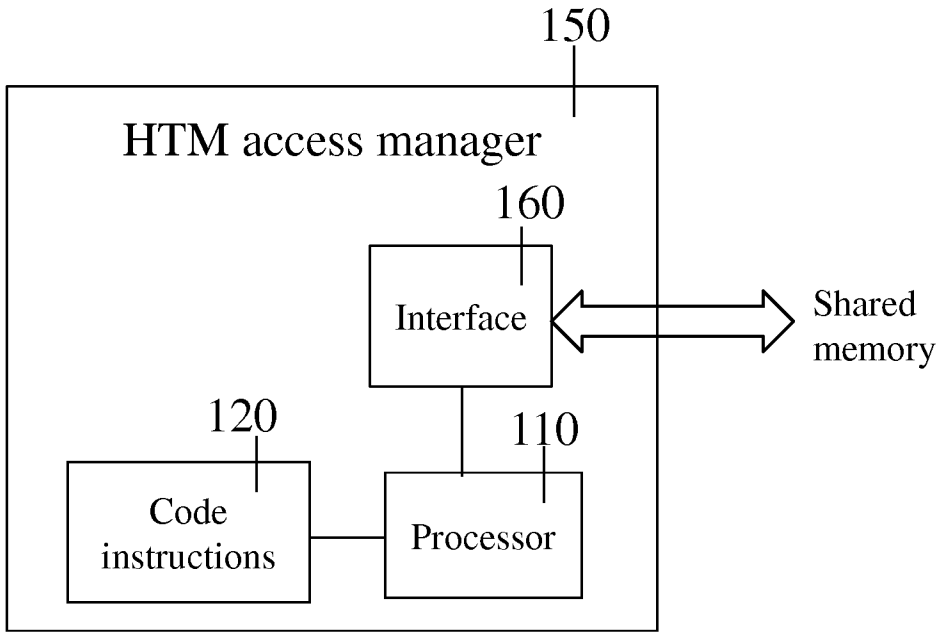
150
HTM access manager
160
Interface
Shared memory
120
110
Code instructions
Processor

Fig. 1C

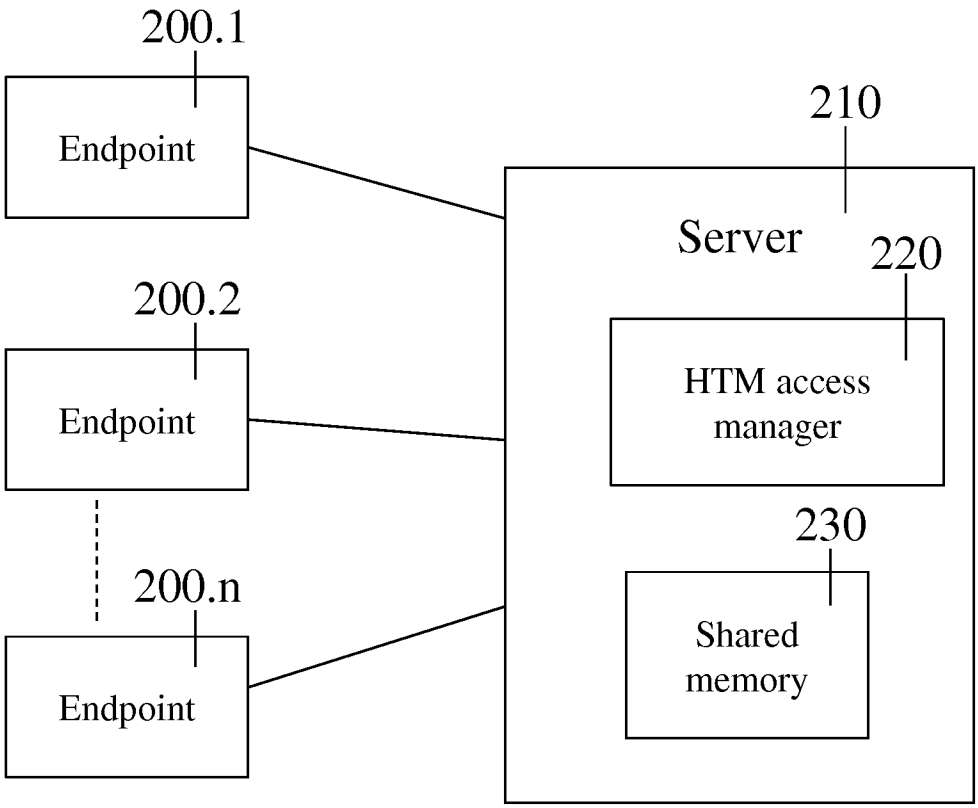
200.1
Endpoint
210
Server
220
200.2
Endpoint
HTM access manager
230
200.n
Shared memory
Endpoint

Fig. 2

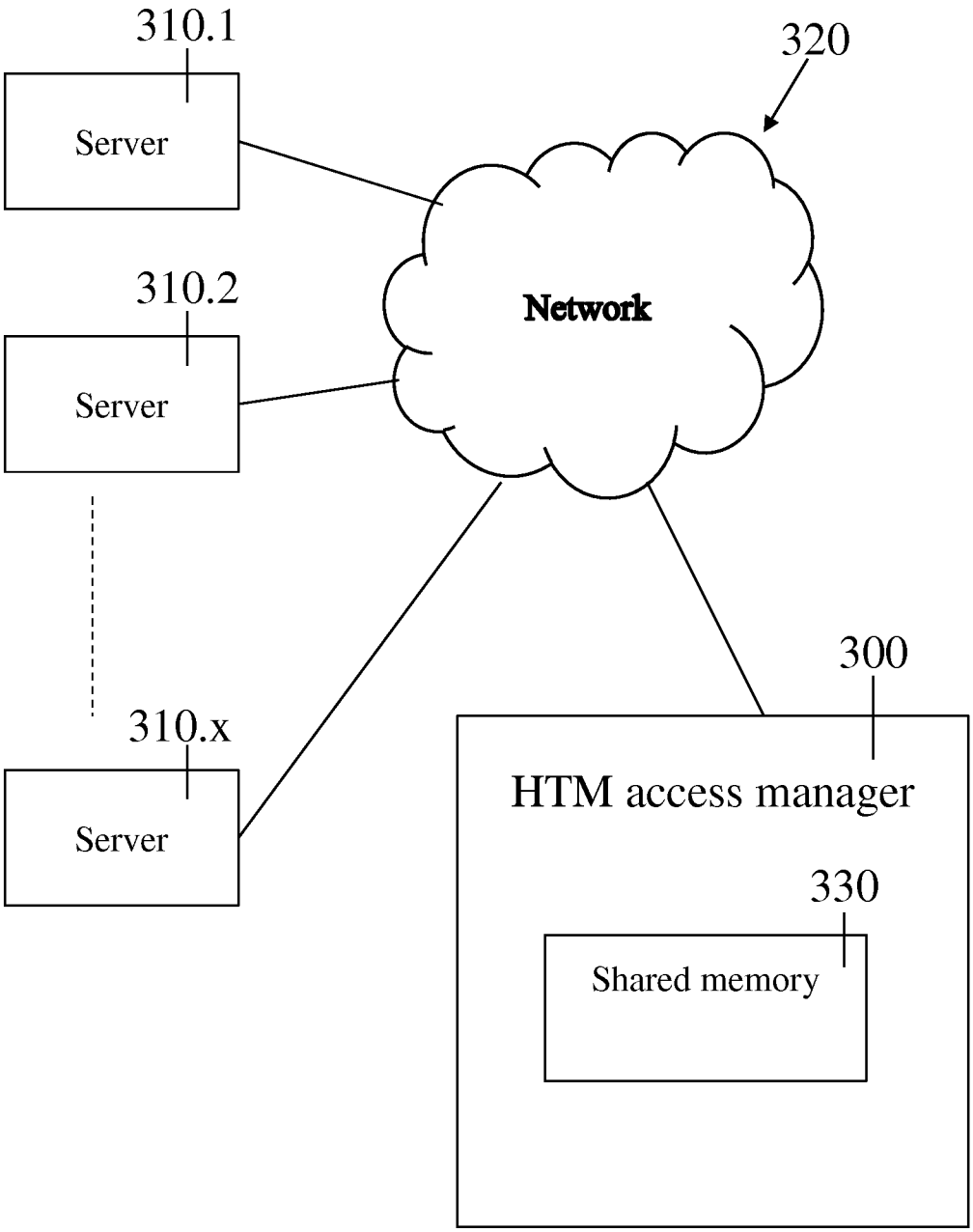
310.1
Server
320
310.2
Server
Network
310.x
Server
300
HTM access manager
330
Shared memory

Fig. 3

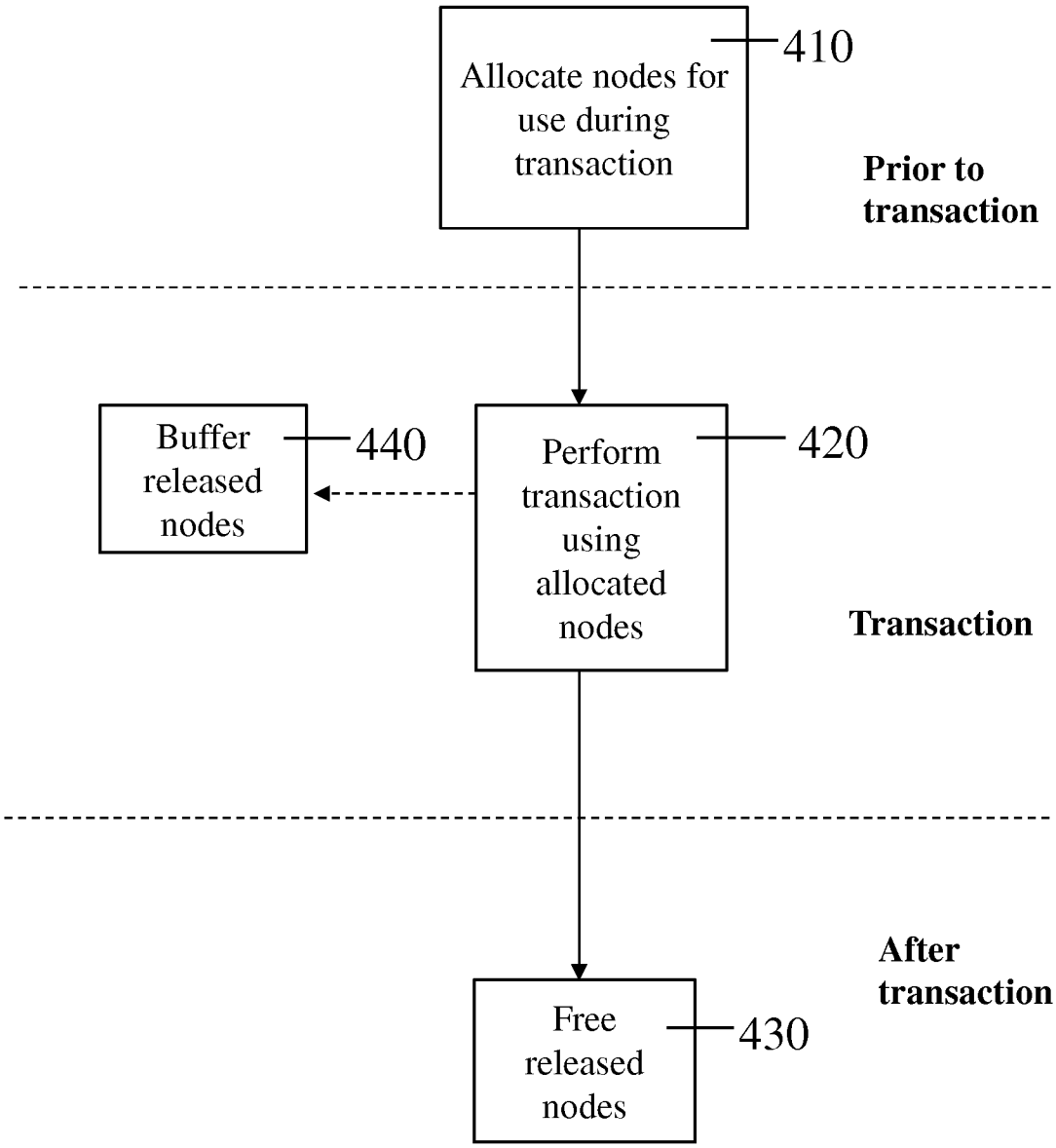
Allocate nodes for use during transaction
410
Prior to transaction
Buffer released nodes
440
Perform transaction using allocated nodes
420
Transaction
After transaction
Free released nodes
430

Fig. 4

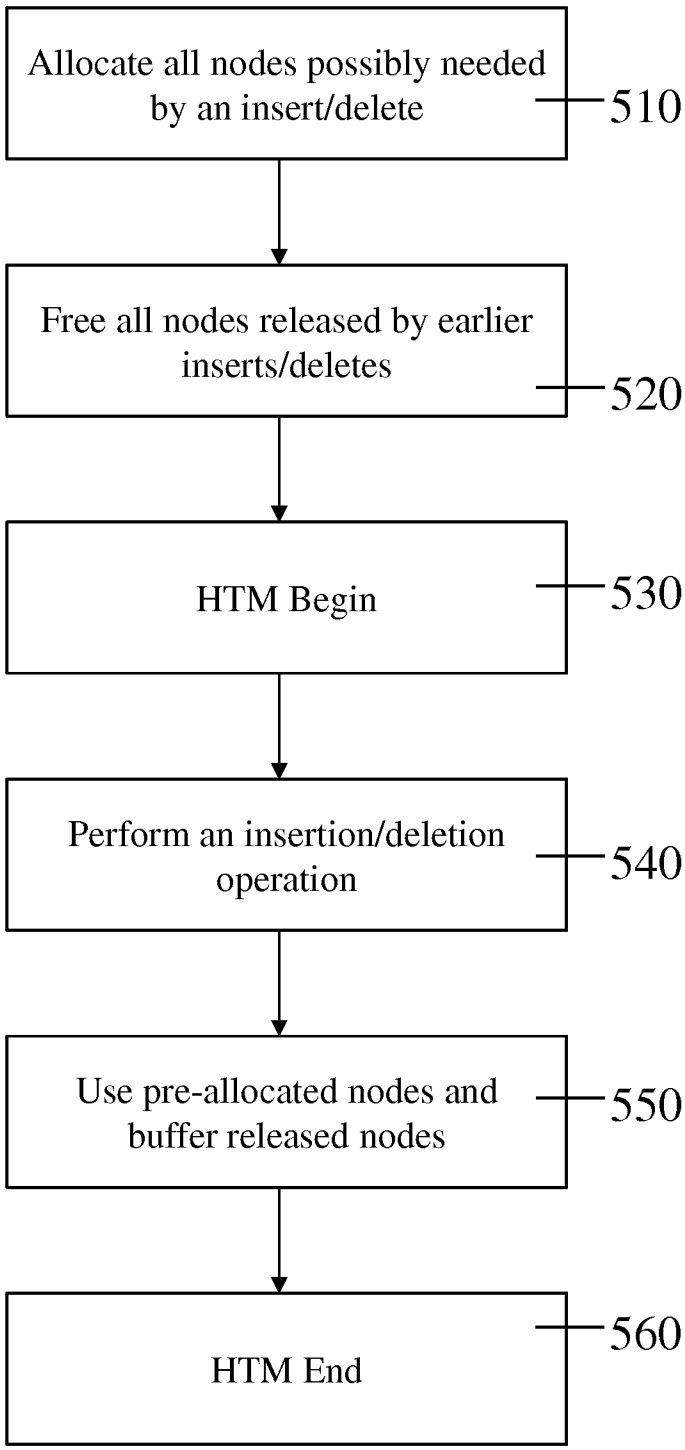
Allocate all nodes possibly needed by an insert/delete
510
Free all nodes released by earlier inserts/deletes
520
HTM Begin
530
Perform an insertion/deletion operation
540
Use pre-allocated nodes and buffer released nodes
550
HTM End
560

Fig. 5

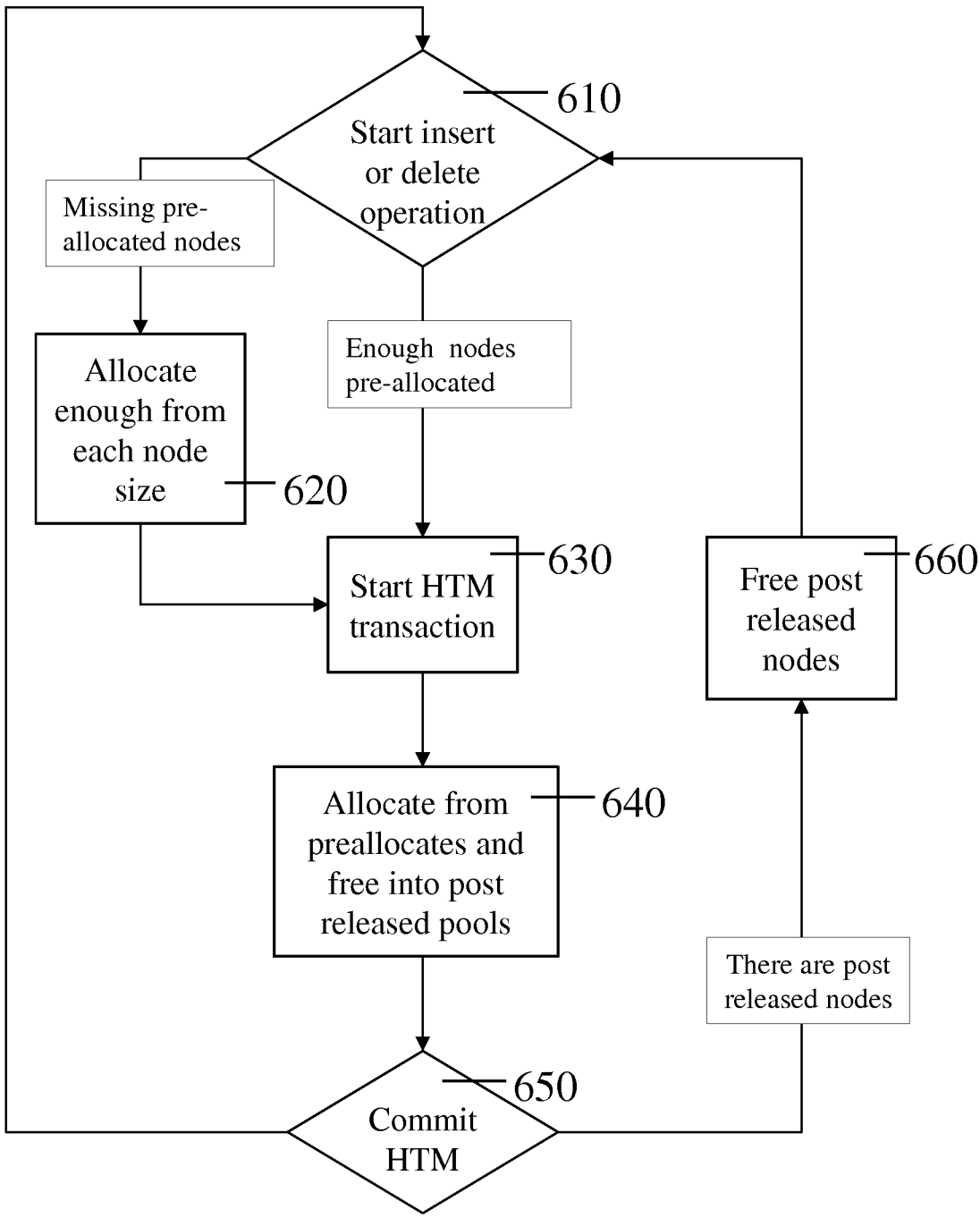
610
Start insert or delete operation
Missing pre-allocated nodes
Enough nodes pre-allocated
Allocate enough from each node size
620
630
Start HTM transaction
660
Free post released nodes
Allocate from preallocates and free into post released pools
640
There are post released nodes
650
Commit HTM

Fig. 6

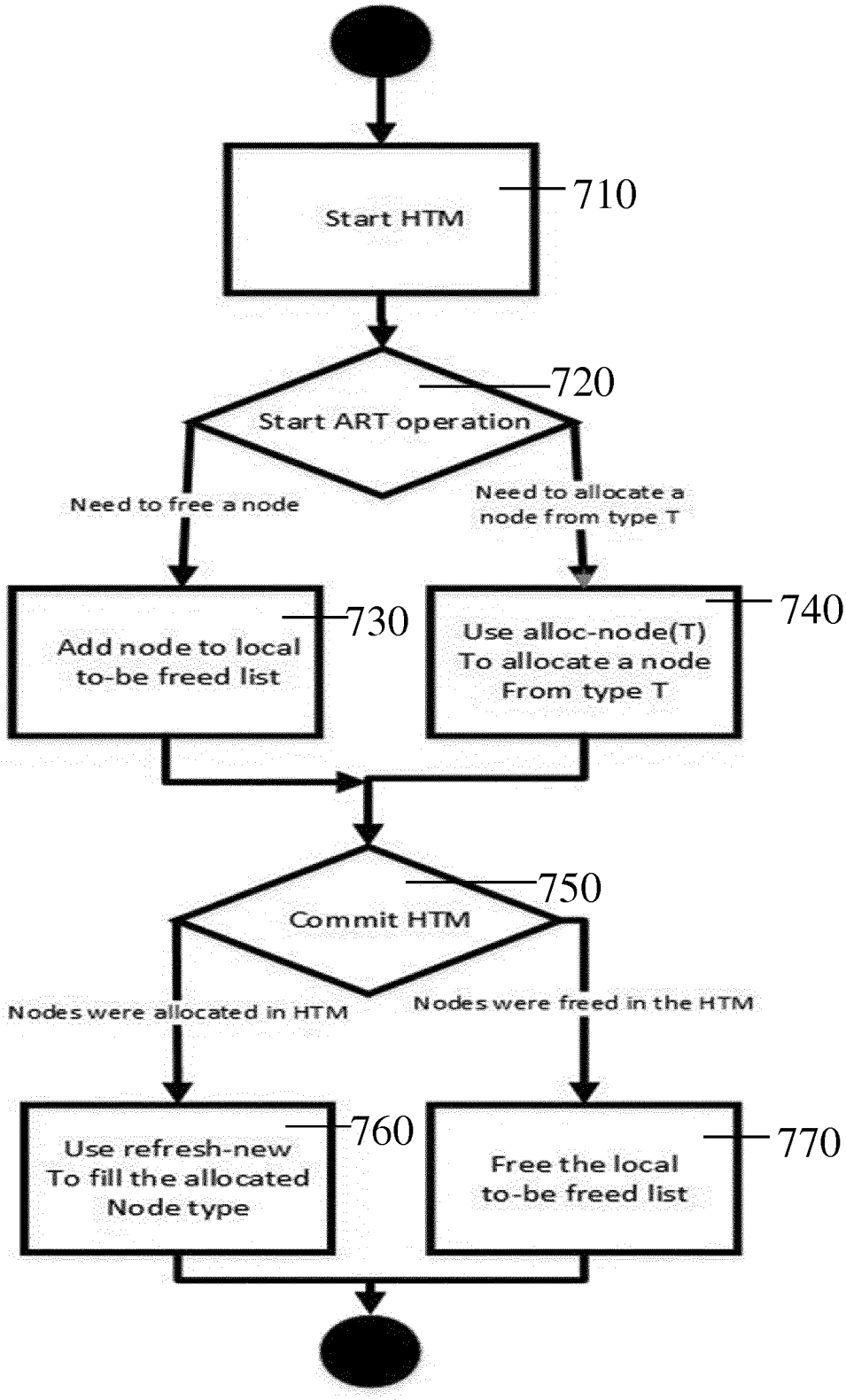
Start HTM
710
Start ART operation
720
Need to free a node
Need to allocate a node from type T
Add node to local to-be freed list
730
Use alloc-node(T) To allocate a node From type T
740
Commit HTM
750
Nodes were allocated in HTM
Nodes were freed in the HTM
Use refresh-new To fill the allocated Node type
760
Free the local to-be freed list
770

Fig. 7

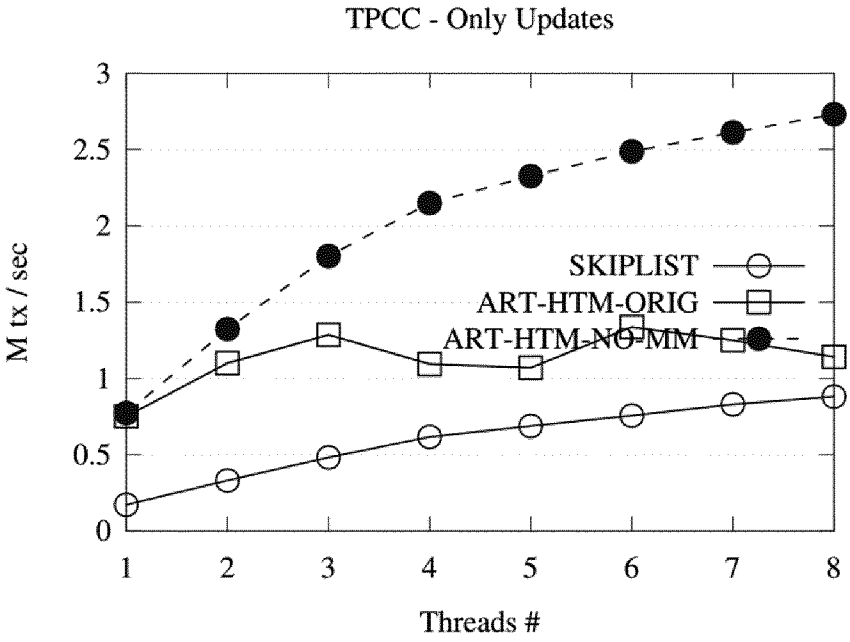
TPCC - Only Updates
M tx / sec
Threads #
SKIPLIST
ART-HTM-ORIG
ART-HTM-NO-MM
0
0.5
1
1.5
2
2.5
3
1
2
3
4
5
6
7
8

Fig. 8

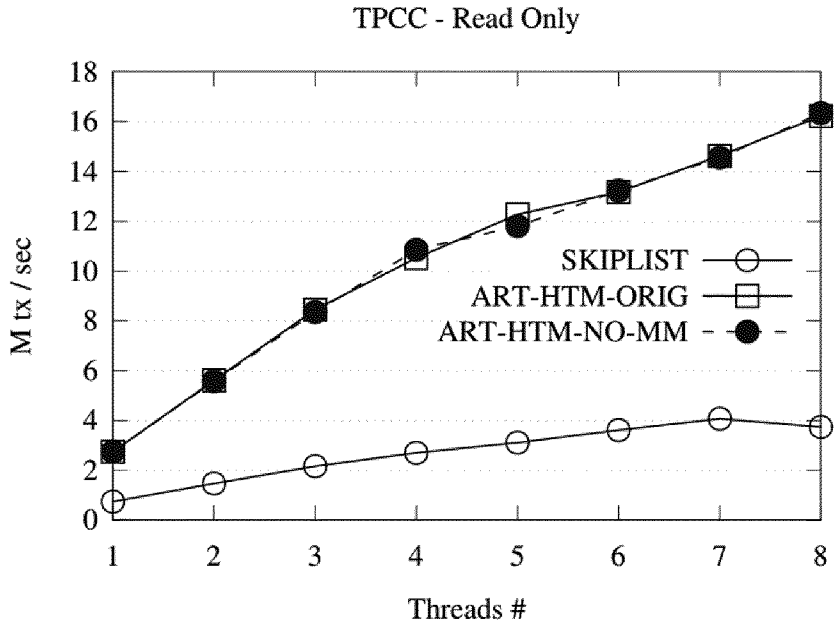
TPCC - Read Only
M tx / sec
Threads #
SKIPLIST
ART-HTM-ORIG
ART-HTM-NO-MM
0
2
4
6
8
10
12
14
16
18
1
2
3
4
5
6
7
8

Fig. 9

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2017/083219

A. CLASSIFICATION OF SUBJECT MATTER
INV. G06F9/46
ADD.

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

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
EPO-Internal, WPI Data

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2015/074219 A1 (CHIN BILL YING [US] ET AL) 12 March 2015 (2015-03-12) paragraphs [0030], [0041], [0052], [0059], [0073] - [0077], [0086], [0103], [0108], [0110], [0111], [0124], [0126]	1-14
X	US 2012/310987 A1 (DRAGOJEVIC ALEKSANDAR [CH] ET AL) 6 December 2012 (2012-12-06) paragraphs [0007] - [0010], [0026] - [0030]	1-14
X	US 2016/004557 A1 (CASTANOS JOSE G [US] ET AL) 7 January 2016 (2016-01-07) paragraphs [0019], [0029], [0063], [0074], [0080]	1-14

-/--

[X] Further documents are listed in the continuation of Box C.

[X] 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
28 August 2018	05/09/2018

Name and mailing address of the ISA/	Authorized officer
European Patent Office, P.B. 5818 Patentlaan 2 NL - 2280 HV Rijswijk Tel. (+31-70) 340-2040, Fax: (+31-70) 340-3016	Buzgan, C

2

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

International application No
PCT/EP2017/083219

C(Continuation). DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2013/013899 A1 (BARTON CHRISTOPHER M [CA] ET AL) 10 January 2013 (2013-01-10) paragraphs [0019], [0020], [0024] - [0026], [0043], [0054], [0059], [0063] - [0065]	1-14

2

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No
PCT/EP2017/083219

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US 2015074219 A1	12-03-2015	US 2015074219 A1	12-03-2015
		US 2015081986 A1	19-03-2015
		US 2015082085 A1	19-03-2015
		US 2017199760 A1	13-07-2017
US 2012310987 A1	06-12-2012	NONE	
US 2016004557 A1	07-01-2016	JP 5901835 B2	13-04-2016
		JP WO2014129247 A1	02-02-2017
		US 2016004557 A1	07-01-2016
		WO 2014129247 A1	28-08-2014
US 2013013899 A1	10-01-2013	US 2013013899 A1	10-01-2013
		WO 2013006525 A1	10-01-2013