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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
05 March 2020 (05.03.2020)

WIPO | PCT

(10) International Publication Number
WO 2020/046377 A1

(51) International Patent Classification:
G06F 11/30 (2006.01) *G06F 16/00* (2019.01)

(21) International Application Number:
PCT/US2018/049134

(22) International Filing Date:
31 August 2018 (31.08.2018)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant: **HEWLETT-PACKARD DEVELOPMENT COMPANY, L.P.** [US/US]; 10300 Energy Drive, Spring, Texas 77389 (US).

(72) Inventor: **ALCORN, Byron A.**; 3390 E. Harmony Rd., Fort Collins, Colorado 80528 (US).

(74) Agent: **MATHEW, Wilson** et al.; HP Inc., 3390 E. Harmony Road, Mail Stop 35, 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, 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).

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))*

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

WO 2020/046377 A1

(54) Title: MULTI-TRANSACTION WORKLOAD REPLICATION

(57) Abstract: In one example in accordance with the present disclosure, a method is described. During a workload playback mode, sequence values associated with sequential boundary events of multiple recorded transactions are identified. A counter is incremented through the sequence values and boundary events are replicated as the counter reaches an associated sequence value.

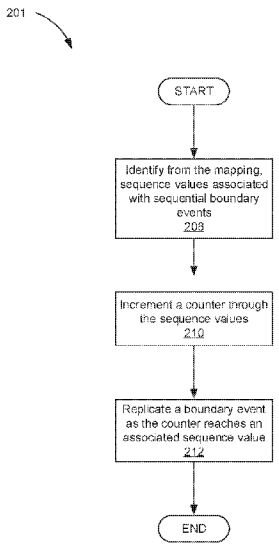

Fig. 2B

MULTI-TRANSACTION WORKLOAD REPLICATION

BACKGROUND

[0001] Computing devices play a role in many people's day-to-day lives and computers are relied on by many users. Computing devices can be used in the home for personal use. Computing devices can also be used commercially or professionally. Such commercial computing devices may be referred to as workstations or commercial PCs. Criteria may be used to evaluate and quantify the performance of different computing devices.

BRIEF DESCRIPTION OF THE DRAWINGS

[0002] The accompanying drawings illustrate various examples of the principles described herein and are part of the specification. The illustrated examples are given merely for illustration, and do not limit the scope of the claims.

[0003] Fig. 1 is a block diagram of a system for multi-transaction workload replication, according to an example of the principles described herein.

[0004] Figs. 2A and 2B are flowcharts of methods for multi-transaction workload replication, according to an example of the principles described herein.

[0005] Fig. 3 is a flowchart of a method for multi-transaction workload playback, according to another example of the principles described herein.

[0006] Fig. 4 is a diagram of assigning sequence values during multi-transaction workload replication, according to an example of the principles described herein.

[0007] Fig. 5 is a diagram of a machine-readable storage medium for multi-transaction workload replication, according to an example of the principles described herein.

[0008] Throughout the drawings, identical reference numbers designate similar, but not necessarily identical, elements. The figures are not necessarily to scale, and the size of some parts may be exaggerated to more clearly illustrate the example shown. Moreover, the drawings provide examples and/or implementations consistent with the description; however, the description is not limited to the examples and/or implementations provided in the drawings.

DETAILED DESCRIPTION

[0009] A workload refers to the amount of processing that a computing device has to execute to produce a specified amount of work. Workloads are made up of transactions to subsystems within a computing device. When gathering data related to workloads, the data can be focused on a single subsystem like storage or on many subsystems like storage, memory, and networking. Workloads for which the computing device is used varies almost as much as the computing devices themselves. The workload of a computing device or a subsystem within the computing device may dictate which of a variety of computing devices or subsystems is best suited for that workload. For example, different computing devices have different performance characteristics, with no one computing device being best suited to handle the transactions that make up the workload. A workload is made up of threads, a thread being an instruction set or command stream that is communicated between an application running on a computing device and a subsystem of the computing device. In the command stream of each thread, multiple transactions may exist to perform a specific function. The quantity and characteristics of the thread transactions may determine which computing device architecture is best suited for a particular workload.

[0010] With the innumerable options of computing devices available, it may be difficult to assess which of the different computing devices is best able

to meet the workload demands of a particular computing environment. Accordingly, the present specification describes a system and method for evaluating computing device performance given a particular workload. Specifically, during a capture mode, boundary events for the different transactions of an actual workload in a particular environment are recorded. Then during the capture mode or subsequent to the capture mode, sequence values are assigned to each transaction.

[0011] Then during a playback mode, a counter is incremented through the sequence values and the boundary events are replicated when the counter reaches an associated sequence value. So doing replicates the workload, and accounts for the interaction between different transactions and different threads. Without accounting for the interaction of the transactions and threads, any replication of the workload of the computing device is inaccurate as such interactions affect the performance of the computing device.

[0012] Specifically, the present specification describes a method. According to the method, during a workload playback mode, 1) sequence values associated with sequential boundary events of all recorded transactions are identified, 2) a counter is incremented through the sequence values, and 3) boundary events are replicated as the counter reaches an associated sequence value.

[0013] The present specification also describes a system. The system includes a transaction recorder to record data for each of multiple transactions between threads of an application and a computing device subsystem. The data includes at least boundary events for a transaction and other data to replicate the transactions during workload playback. A counter of the system increments sequence values with each sequential boundary event and an assigner assigns incremental sequence values to sequential boundary events for the multiple transactions. A replicator of the system replicates, during a playback mode, boundary events as the counter increments and reaches an associated sequence value. A database of transactions maps recorded data, including associated boundary events, to the associated sequence value.

[0014] A tangible machine-readable storage medium is encoded with instructions executable by a processor. The machine-readable storage medium includes instructions to during a workload capture mode 1) records boundary events for multiple transactions between threads of an application and a computing device subsystem, 2) incrementally assigns sequence values to sequential boundary events, and 3) stores a mapping between assigned sequence values and recorded boundary events. The tangible machine-readable storage medium is also encoded with instructions executable by the processor to, during a workload playback mode 1) increment a counter through the sequence values, 2) retrieve, based on the mapping, a recorded boundary event associated with a current sequence value, 3) replicate the boundary event, and 4) increment the counter to a subsequent sequence value.

[0015] In summary, using such a replication system and method 1) tests individual computing device subsystems; 2) accounts for complex multi-threading and the relationships between the threads; 2) maintains the sequencing of transactions regardless of threading; 4) accurately captures and plays back computing workloads; and 5) allows for the accurate testing of workloads on a range of computing device subsystems. However, the devices disclosed herein may address other matters and deficiencies in a number of technical areas.

[0016] Fig. 1 is a block diagram of a system (100) for multi-transaction workload replication, according to an example of the principles described herein. Note that the system (100) may be a distributed system (100) with different components on different devices. For example, the transaction recorder (102) may be on a computing device separate from a computing device on which the replicator (108) is disposed.

[0017] The system (100) includes a transaction recorder (102) to record data for each of multiple transactions between threads of an application and a computing device subsystem. During use, an application is run on a computing device. To perform certain operations, the application executes threads, which threads perform a number of transactions with the respective computing subsystem. For example, when the computing subsystem is a memory storage

device, the transactions may be reading and/or writing data to the memory storage device. The transaction recorder (102) records data relating to each transaction. Specifically, the transaction recorder (102) records boundary events for each transaction and data associated with that transaction. A boundary event for a transaction refers to either a starting point or a terminating point for a particular transaction. The transaction recorder (102) also records other data which is used to replicate transactions during workload playback. For example, the transaction recorder (102) may record a transaction size, an address for the transaction as well as a command associated with the transaction. In some examples other information may also be captured. For example, flags associated with the transaction may be recorded, which flags help to identify the characteristics of the transaction. As a specific example of a flag, Force Unit Access could be set for a write transaction which tells the transaction to write data all the way to persistence. As yet another example, the transaction recorder (102) records a thread identifier for each transaction so that each transaction for a given thread can be correlated to that thread.

[0018] A counter (104) of the system (100) increments a sequence value with each sequential boundary event and an assigner (106) assigns an incremented sequence value to a sequential boundary event of the multiple transactions. For example, initially the counter (104) may be set to the value of 1. A first thread may initiate a first transaction. The initiation event for the first transaction may be assigned a value of 1 by the assigner (106), and the counter (104) increments the sequence value to 2. Subsequently, a second thread may initiate a second transaction. The initiation event of the second transaction may be assigned the incremented value of 2 by the assigner (106), and the counter (104) increments again, this time to the sequence value to 3. In this example, a termination event is recorded by the transaction recorder (102) for the first transaction. This termination event is assigned the value of 3 by the assigner (106), and the counter (104) increments the sequence value to 4. A termination event is then recorded by the transaction recorder (102) for the second transaction. This termination event is assigned the value of 4 by the assigner

(106), and the counter (104) increments the sequence value to "5." In this example, the sequential events are indicated in the table below.

Sequential Value	Boundary Event
1	1st transaction initiation event
2	2nd transaction initiation event
3	1st transaction termination event
4	2nd transaction termination event

[0019] Thus, a sequence of events is generated. In some examples, the functions described to this point, i.e., recording transactions and assigning sequence values to boundary events, etc. may occur during a workload capture mode. That is, a user attempting to provide data by which a particular subsystem is evaluated, can initiate a capture mode where in sequential occurrence of different boundary events for multiple threads are recorded. In another example, the transaction recorder (102) operates during a capture mode to capture a timestamp and duration. The sequencing, or assigning of sequence values to the boundary events, happens in a post-processing operation after the capture is complete.

[0020] Then during a playback mode, those sequential occurrences are replicated such that a subsystem may be evaluated. As described above, the replication may be performed on a different computing device from the recording computing device.

[0021] A replicator (108) of the system (100), which may be on a different physical device, replicates the boundary events as the counter (104) increments. That is, the counter (104), during the capture mode or in a post-capture mode, increments to trigger the assignation of a new sequence value to a particular boundary event. During a playback mode, the counter (104) increments to log the replication of the particular boundary event. For example, at the start of the playback mode, the counter (104) may be set to the value of 1. At this point in time, the replicator (108) replicates the first transaction initiation event. The counter (104) is then incremented to 2 and the replicator (108)

replicates the second transaction initiation event. The computing device on which the replicator (108) is disposed, completes the transaction previously initiated and the counter (104) has to be 1 minus the transaction number before the transaction can be closed. New transactions cannot start until the all the appropriate previous termination events have occurred. Put another way, a transaction initiation is trigged when the counter (104) is one less than the transaction start sequence number and a transaction termination event is triggered when the playback system completes the transaction and the counter (104) is one less than the termination sequence number.

[0022] The system (100) also includes a database (110) that maps the recorded data to the associated sequence values. That is, the database (110) includes a mapping wherein the replicator (108) knows which boundary event to trigger or wait for responsive to the counter (104) reaching a particular sequence value. The database (110) is also used during assignation of sequence values.

[0023] As described herein, the system (100) provides a subsystem evaluation operation which accounts for the complex interactional nature of multiple computing threads. That is, in modern computing devices, threads execute multiple transactions with the various subsystems of a computing device, with these transactions often times overlapping. Some benchmarking programs may have a user to guess as to what the workload looks like and use that estimate to evaluate the performance of different platforms. Another approach is to use a benchmarking application which relies on template data to estimate computer performance.

[0024] In yet another approach, an attempt may be made to replicate the workload of a computing device. However, such replications do not accurately emulate a user's workload as they do not portray the interaction of various computing threads. For example, such a trace-based system captures the individual transactions per thread, and then during replication presumes all transactions are executing in parallel. But that is not how a computing device executes transactions. That is, without understanding transaction interaction, a workload cannot be accurately represented. In fact, the workload captured

without understanding the transaction interaction stimulates the storage subsystem being evaluated in ways that are not valid, making the playback an invalid representation of the subsystem dynamics.

[0025] For example, a system that assumes that all threads are pulling data from a subsystem in parallel, does not accurately represent a computing workload and therefore does not provide an accurate metric of subsystem performance. Thus the present specification describes a system (100) wherein multi-threaded transactions are sequenced such that the original workload is accurately captured and replayed. Doing so allows a user or administrator to accurate capture any workload and play it back on complex subsystems that would be difficult to profile with other benchmarking applications. For example, those workloads that are tiered with solid state drives (SSDs) for hot data.

[0026] Using the present system (100), rather than industry standard storage benchmarks, provides the capability to upload data that is specific to a user, thus providing a tailored metric by which computer subsystem performance can be measured. Accordingly, in some examples, the system (100) may be disposed on the computing device of the user. As described above, the system (100) may be used to track and evaluate the performance of a variety of subsystems. For example, the subsystem being evaluated may be a storage subsystem. In this example, the transactions that are recorded are storage transactions to and from the storage component of a computing device.

[0027] Figs. 2A and 2B are flowcharts of methods (200 and 201) for multi-transaction workload replication, according to an example of the principles described herein. Specifically, Fig. 2A depicts a method (200) of multi-transaction workload capture and Fig. 2B depicts a method (201) of multi-transaction workload playback. As described above, the methods (200, 201) may be executed by the system (Fig. 1, 100) albeit on different devices.

[0028] According to the method (200), data to replicate a transaction is recorded (block 202) for multiple transactions. Such an operation may be performed by the transaction recorder (Fig. 1, 102). That is, during operation, threads of an application execute transactions between a particular computing device subsystem, such as a storage subsystem. Each transaction has an

initiation event and a termination event. The boundary events for a particular transaction may be non-sequential. That is, different transactions may overlap one another. As a specific example, a first transaction may start, and then a second transaction may start before the first transaction ends. Accordingly, at some point in time the workload includes two transactions executing in parallel. It is this interaction of overlapping, or non-sequential, transactions that may be lost when using other subsystem evaluation systems.

[0029] Accordingly, the method (200) records (block 202) such boundary events, i.e., initiation and termination events for each transaction. Other data may also be recorded. Examples of other data that is recorded includes a size of the transaction, a command of the transaction, an address of the transaction, flags associated with the transaction, a thread identifier for the transaction. This other data helps the playback system know what to execute. While particular reference is made to particular types of additional information, other additional information may also be collected.

[0030] For example, a size of each transaction may be recorded. Recording the size of each transaction aids in an accurate replication of the workload. For example, assume a particular transaction has an initiation event assigned a sequence value of 1, a size of 1 megabyte, and a termination event assigned a sequence value of 2. In this example, during playback the initiation event for the particular transaction is triggered as the counter (Fig. 1, 104) reaches the value 1. Rather than immediately incrementing to the value of 2 wherein the termination event is triggered, to accurately replicate the particular transaction, the counter (Fig. 1, 104) may remain the same until the transaction completes on the playback device, at which time the counter (Fig. 1, 104) is then incremented to the value of 2.

[0031] As the data is recorded (block 202), sequence values are incrementally assigned (block 204) to sequential boundary events. That is, a counter (Fig. 1, 104) starts at a particular sequence value. As indication is received of a boundary event, the current value indicated by the counter (Fig. 1, 104) is assigned to the boundary event. That is, each boundary event is assigned (block 204) a sequence value as it happens over time. The sequential

value assigned is independent of a type of boundary event (i.e., initiation or termination), a thread that is executing the transaction, and the state of other transactions. For example, a first boundary event is assigned (block 204) a sequence value of 1 and a second boundary event, be it a termination event of a first transaction or an initiation event of a second transaction, is assigned (block 204) a sequence value of 2. Such a process is repeated for all transactions executed during a period of time, which may be predetermined by a user. Following such a schema, the interaction and/or overlap of particular transactions is maintained in a simple, easy to replicate format. These sequence values may be stored (block 206) and associated with a particular transaction in a stimulus file, or other database (Fig. 1, 110). That is, the recorded information for each transaction is stored, specifically the boundary events, the associated sequence values, and the other information such as the transaction size and thread identifier. With this information in hand, a replication, or emulation of the workload over a window of interest can then be replicated.

[0032] Fig. 2B then depicts a method (201) wherein the transactions are replicated. That is, it is identified (block 208) from the mapping, sequence values that are associated with boundary events. From this mapping, a replication of an entire workload can be replicated. Specifically, the counter (Fig. 1, 104) is reset and incremented (block 210). As the counter (Fig. 1, 104) increments (block 210) through the sequence values, corresponding boundary events are replicated (block 212) when associated sequence values are reached. Returning to the example above, the counter (Fig. 1, 104) is set to an initial value of 1 and the first boundary event is triggered. Specifically, the boundary event that is mapped to the sequence value of 1 in the database (Fig. 1, 110) is triggered. Following the triggering of the first boundary event, the counter (Fig. 1, 104) is incremented (block 203) to a value of 2 and a second boundary event is executed (block 212). This second boundary event may be an initiation event or a termination event. In the case of a termination event, the playback may stall until the playback system has completed the transaction.

[0033] Thus, the present system (Fig. 1, 100) by recording sequential boundary events provides an elegant and simple way of recording sequential transaction events and maintaining the relationship between the transactions. Such a sequencing allows for a reproduction of the transactions, and their relationship, i.e., timing with regards to other transactions, during a playback mode. With the data regarding transaction interaction and sequencing available, a more accurate replication of the computing device workload is replicated, which ensures that any performance metrics output by the system (Fig. 1, 100) are more reflective of an actual workload, rather than an estimation or simulation of an actual workload. As such, a user may have more confidence in the accuracy of the output performance metrics and the ultimately selected subsystem.

[0034] Fig. 3 is a flowchart of a method (300) for multi-transaction workload playback, according to another example of the principles described herein.

[0035] During replication, sequence values associated with sequential boundary events ae identified (block 301) and the counter (Fig. 1, 104) is reset and incremented (block 302) through the sequence values. As the counter (Fig. 1, 104) indicates a particular sequence value, the associated boundary event is replicated (block 303). In some examples, this may be performed as described above in connection with Fig. 2B.

[0036] According to the method (300) while replicating (block 303) the boundary events, a termination boundary event for the transactions are stalled (block 304) until completion of the transaction. For example, as described above a particular transaction has an initiation event assigned a sequence value of 1, a size of 1 megabyte, and a termination event assigned a sequence value of 2. In this example, rather than immediately incrementing to the value of 2, to accurately replicate the particular transaction, the counter (Fig. 1, 104) may be delayed in incrementing until the playback device completes the transaction. When the sequence value is one less than the termination sequence value and the playback device has completed the transaction, the counter (Fig. 1, 104) is then incremented to the value of 2 and the replication of the termination event

for that transaction is logged. Thus an accurate replication of each transaction can be replicated. In some examples, the database (Fig. 1, 110) with the associated mappings may be sent to a server to allow for easier playback.

[0037] Following replication of all of the multiple transactions of a workload, a metric may be output (block 305) regarding subsystem performance. That is, a user may be running an evaluated subsystem with the replicated workload emulated thereon. Once replication, or playback of each workload is complete, a metric is output (block 305) which indicates how the particular subsystem performed running that particular workload. In some cases, the metric, or other information regarding subsystem performance may be transmitted (block 306) either in full or redacted form, to an aggregating server. In this case, the transmitted (block 306) information could be relied on to increase or adjust computing device development or implementation. Accordingly, not only does the system (Fig. 1, 100) provide a method for customizing a particular computing device subsystem based on specific operating parameters of that computing device, but the system (Fig. 1, 100) also enhances the development and progress of future iterations of computing devices.

[0038] Fig. 4 is a diagram of assigning sequence values during multi-transaction (414) workload replication, according to an example of the principles described herein. That is, Fig. 4 depicts the various transactions (414) that may be executed and that are therefore replicated during a workload playback mode. As described above, a workload may include multiple threads (412) that are executing different transactions (414) during their operation. While Fig. 4 depicts three threads (412-1, 412-2, 412-3), a workload may include any number of threads (412). Fig. 4 also depicts the transactions (414) that are carried out by the different threads (412). As described above, a transaction (414) is an interaction with a subsystem. For example, a thread (412) may request data from or write data to a storage subsystem. Each instance of reading/writing data may constitute a particular transaction (414).

[0039] As described above, each transaction (414) has a starting point and an ending point that is given a particular sequence value based on the

relative starting time and duration of the transaction. As can be seen in Fig. 4, the sequence value is a global value across a workload and is incremented on each transaction (414) initialization and each transaction (414) termination, which global value is stored in a database (Fig. 1, 110).

[0040] As can be seen in Fig. 4, in some cases an initiation boundary event for a particular transaction (414) may occur sequentially between an initiation boundary event and a termination boundary event for another transaction (414). For example, the initiation event for a second transaction (414-2), indicated by the sequence value 2 occurs sequentially between the initiation event for a first transaction (414-1), indicated by the value 1 and a termination event, indicated by the value 3 for the first transaction (414-1).

[0041] Similarly, in another example, a termination boundary event for a particular transaction (414) may occur sequentially between an initiation boundary event and a termination boundary event for another transaction (414). For example, the termination event, indicated by the value 7 or a third transaction (414-3) occurs sequentially between the initiation event, indicated by a 6 of a fourth transaction (414-4) and an initiation event, indicated by an 8 of the fifth transaction (414-5).

[0042] Fig. 4 also indicates how a termination event for a particular transaction (414), does not necessarily occur immediately after the triggering of a preceding event, but after a time period consistent with the amount of time for the transaction to playback on the system. For example, termination of the first transaction (414-1) isn't executed immediately following triggering of the initialization of the second transaction (414-2) as indicated by sequential values, but is rather delayed a bit based on the completion of the first transaction (414-1). However, once a termination event is completed, a subsequently triggered initiation event can immediately be triggered. For example, the initialization of the third transaction (414-3) can be triggered immediately following the termination of the first transaction (414-1). Likewise once a termination event is completed, a subsequent termination event can be completed if the playback system has completed that transaction. For example, if the playback system completes the seventh transaction (414-7) before the sixth transaction (414-6),

the termination event 13 triggers the completing of the sixth transaction (414-6) before termination event 14 is allowed to trigger the completing of the seventh transaction (414-7).

[0043] Note that as depicted in Fig. 4, the system (Fig. 1, 100) can record synchronous and asynchronous transactions (414). That is, both the first thread (414-1) and the second thread (414-2) do not initialize a new transaction (414) until an immediately preceding transaction (414) in that thread (412) has terminated. Such operation is referred to as synchronous transactions. By comparison, the third thread (412-3) performs asynchronous transactions wherein a subsequent transaction, (i.e., the seventh transaction (414-7)) can begin before the immediately preceding transaction (i.e., the sixth transaction (414-6)) has terminated.

[0044] During playback, each thread (412) waits to issue its next transaction (414) until the sequence value is one less than the sequence value associated with the start of a transaction (414). At this point, the counter (Fig. 1, 104) increases to the next value. As described above, at the completion of a transaction (414), the thread (412) waits until the sequence value is one less than the termination transaction value and the playback system has completed the transaction, then the counter (Fig. 1, 104) is incremented. If the thread (412) has multiple outstanding transactions, it tracks the completion sequence value and completion state for each transaction.

[0045] Fig. 5 is a diagram of a machine-readable storage medium (516) for multi-transaction (Fig. 4, 414) workload replication, according to an example of the principles described herein. To achieve its desired functionality, a computing system includes various hardware components. Specifically, the computing system includes a processor. Machine-readable storage medium (516) is communicatively coupled to the processor. The machine-readable storage medium (516) includes a number of instruction sets (518, 520, 522, 524, 526, 528, 530) for performing a designated function. The machine-readable storage medium (516) causes the processor to execute the designated function of the instruction sets (518, 520, 522, 524, 526, 528, 530).

[0046] Although the following descriptions refer to a single machine-readable storage medium (516), the descriptions may also apply to multiple machine-readable storage mediums. In such examples, the instruction sets (518, 520, 522, 524, 526, 528, 530) may be distributed (e.g., stored) across multiple machine-readable storage mediums.

[0047] The machine-readable storage medium (516) represents any tangible and non-transitory memory capable of storing data such as programmed instructions or data structures used by the computing system.

[0048] Referring to Fig. 5, record instructions (518), when executed by a processor, may cause the computing system to, during a workload capture mode, record boundary events for multiple transactions (Fig. 4, 414) between threads (Fig. 4, 412) of an application and a computing device subsystem. Assign instructions (520), when executed by a processor, may cause the computing system to, during the workload capture mode, incrementally assign sequence values to sequential boundary events. Store instructions (522), when executed by a processor, may cause the computing system to, during the workload capture mode, store a mapping between assigned sequence values and recorded boundary events. Counter instructions (524), when executed by a processor, may cause the computing system to, during a workload playback mode, increment the counter (Fig. 1, 104) through sequence values. Retrieve instructions (526), when executed by a processor, may cause the computing system to, during a workload playback mode, retrieve from the mapping, a recorded boundary event associated with a current sequence value. Replicate instructions (528), when executed by a processor, may cause the computing system to, during a workload playback mode, replicate the boundary event. Increment instructions (530), when executed by a processor, may cause the computing system to, during a workload playback mode, increment the counter to a subsequent sequence value.

[0049] In summary, using such a replication system and method 1) tests individual computing device subsystems; 2) accounts for complex multi-threading and the relationships between the threads; 2) maintains the sequencing of transactions regardless of threading; 4) accurately captures and plays back computing workloads; and 5) allows for the accurate testing of

workloads on a range of computing device subsystems. However, the devices disclosed herein may address other matters and deficiencies in a number of technical areas.

CLAIMS

What is claimed is:

1. A system, comprising:

a transaction recorder to record data for each of multiple transactions between threads of an application and a computing device subsystem, wherein the data comprises at least boundary events for each transaction and other data to replicate the transactions during workload playback;

a counter to increment sequence values with each sequential boundary event;

an assigner to assign incremental sequence values to sequential boundary events of the multiple transactions;

a replicator to, during the playback mode and as the counter increments, replicate a boundary event as the counter reaches an associated sequence value; and

a database to map recorded data to the associated sequence value.

2. The system of claim 1, wherein the threads perform at least one of synchronous and asynchronous transactions.

3. The system of claim 1, wherein the computing device subsystem is a storage subsystem.

4. The system of claim 1, wherein the counter increments sequence values and the assigner assigns incremental sequence values to sequential boundary events during a capture mode or following the capture mode.

5. A method, comprising:

during a workload playback mode:

identifying from a mapping, sequence values associated with sequential boundary events of multiple recorded transactions;

incrementing a counter through the sequence values; and

replicating a boundary event as the counter reaches an associated sequence value.

6. The method of claim 5, further comprising:

during a workload capture mode:

recording, for each of multiple transactions between threads of an application and a computing device subsystem, data to replicate the transactions; and

incrementally assigning sequence values to sequential boundary events.

7. The method of claim 6, wherein boundary events for at least one transaction are non-sequential.

8. The method of claim 6, wherein an initiation boundary event for a second transaction occurs sequentially between an initiation boundary event and a termination boundary event for a first transaction.

9. The method of claim 6, wherein a termination boundary event for a second transaction occurs sequentially between an initiation boundary event and a termination boundary event for a first transaction.

10. The method of claim 6, further comprising, during the workload playback mode, triggering a termination boundary event for a transaction based on a completed playback of the transaction and the counter being one value less than the sequence value associated with the termination boundary event for the transaction.

11. The method of claim 6, further comprising storing in a database a mapping between the multiple transactions and associated boundary events.

12. The method of claim 5, further comprising outputting a metric regarding subsystem performance.

13. The method of claim 5, further comprising transmitting information regarding subsystem performance to an aggregating server.

14. A computer program product comprising:

a tangible computer readable storage medium comprising computer readable program code embodied therewith, the computer readable program code comprising program instructions that, when executed, causes a processor to:

during a workload capture mode:

record boundary events for multiple transactions between threads of an application and a computing device subsystem;

incrementally assign sequence values to sequential boundary events; and

store a mapping between assigned sequence values and recorded boundary events; and

during a workload playback mode:

increment a counter through the sequence values;

retrieve, based on the mapping, a recorded boundary event associated with a current sequence value;

replicate the boundary event; and

increment the counter to a subsequent sequence value.

15. The computer program product of claim 14, wherein at least one of the transactions overlaps with another transaction of the multiple transactions.

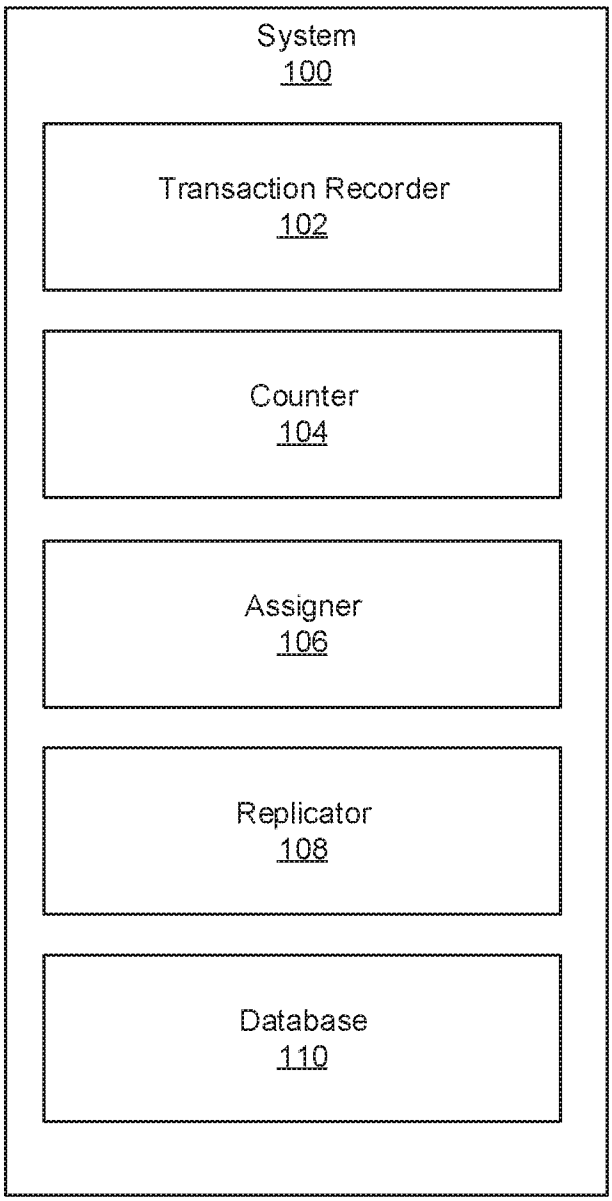
System
100
Transaction Recorder
102
Counter
104
Assigner
106
Replicator
108
Database
110

Fig. 1

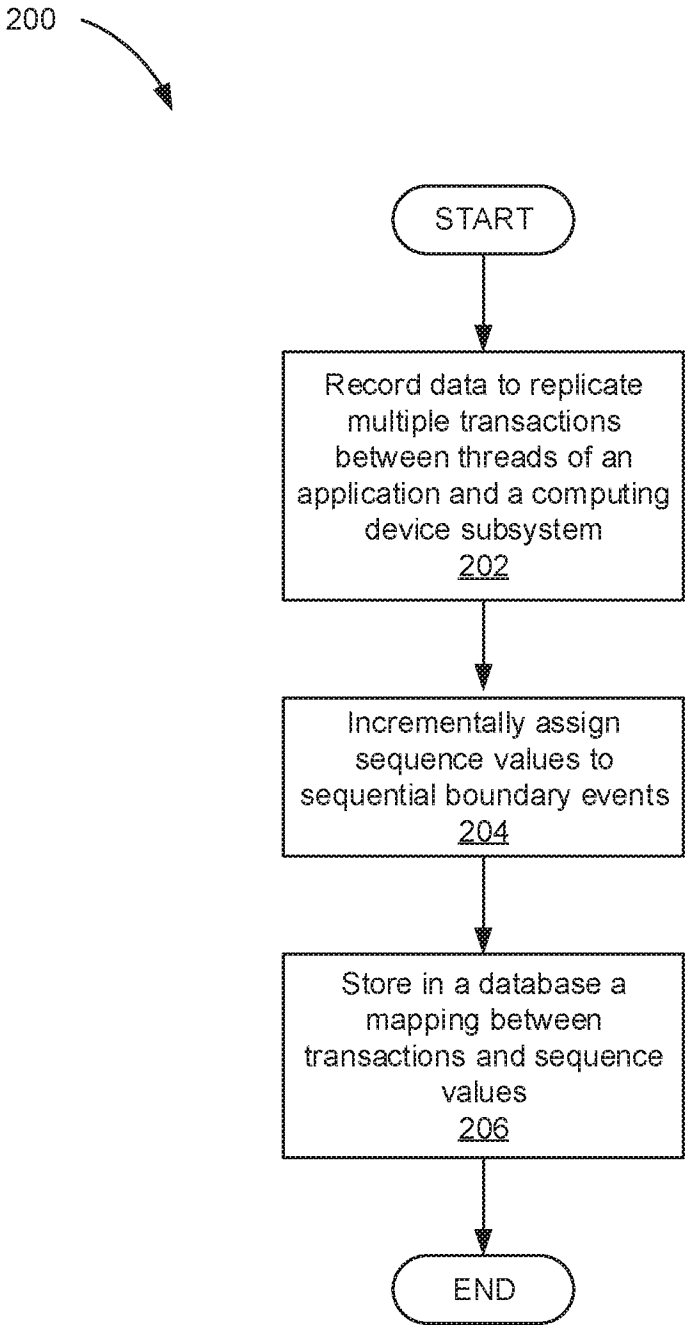
200
START
Record data to replicate multiple transactions between threads of an application and a computing device subsystem
202
Incrementally assign sequence values to sequential boundary events
204
Store in a database a mapping between transactions and sequence values
206
END

Fig. 2A

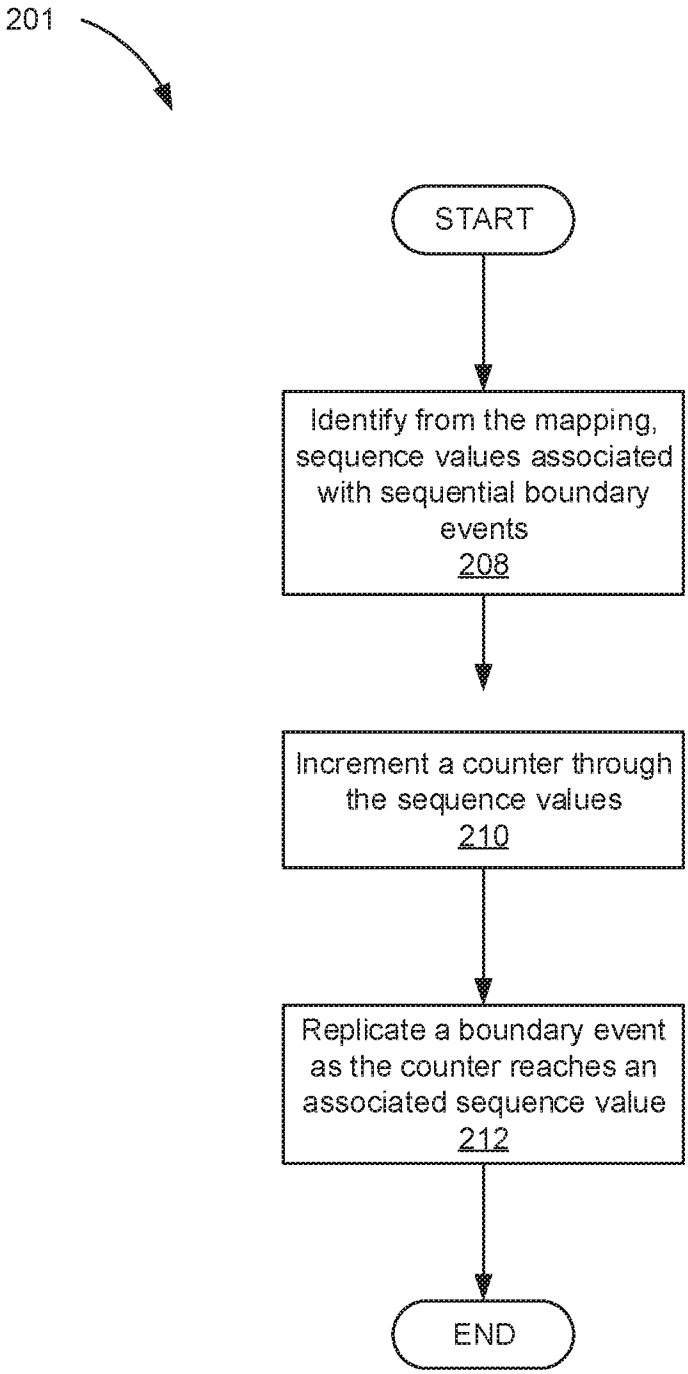
201
START
Identify from the mapping, sequence values associated with sequential boundary events
208
Increment a counter through the sequence values
210
Replicate a boundary event as the counter reaches an associated sequence value
212
END

Fig. 2B

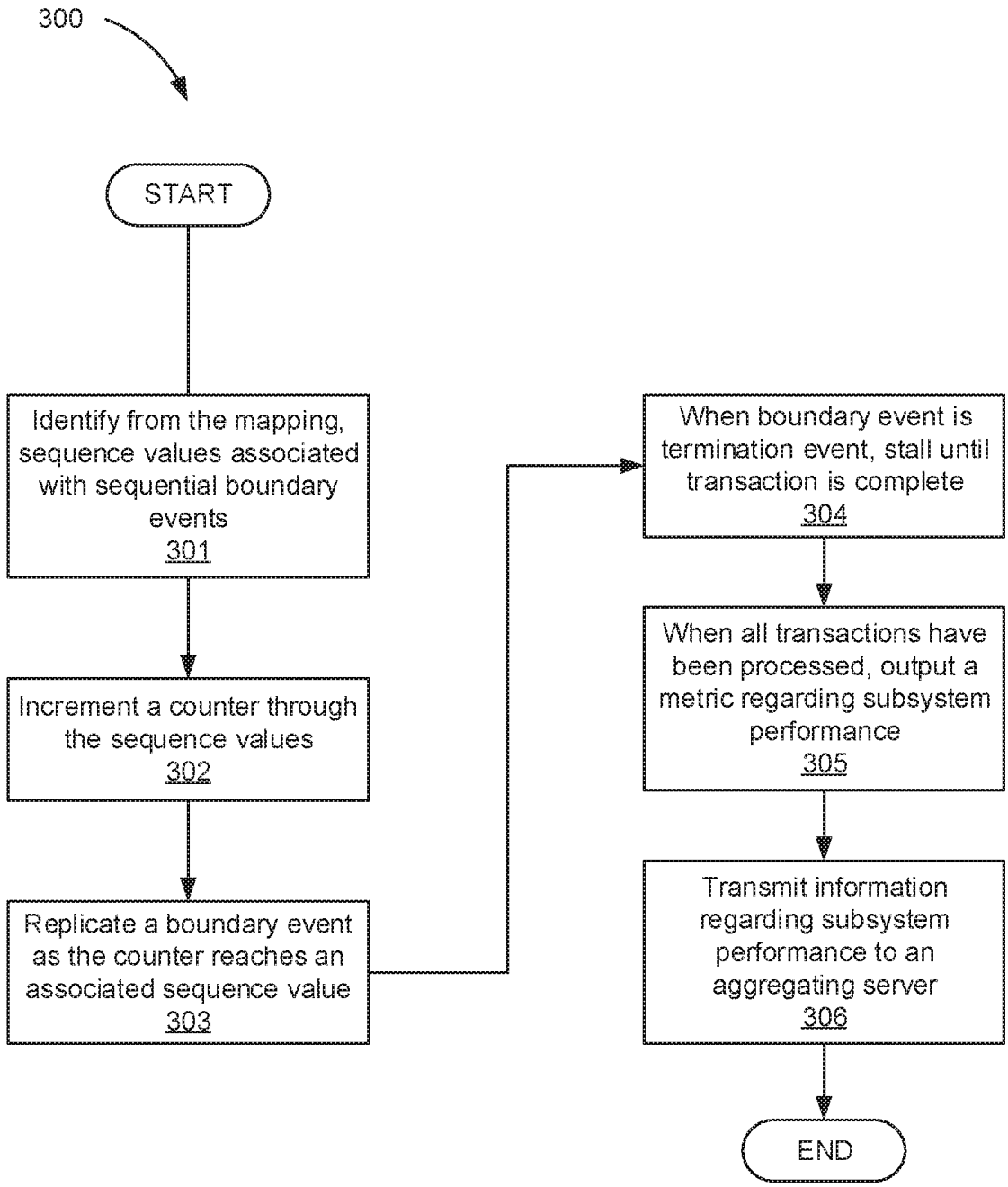
300
START
Identify from the mapping, sequence values associated with sequential boundary events
301
Increment a counter through the sequence values
302
Replicate a boundary event as the counter reaches an associated sequence value
303
When boundary event is termination event, stall until transaction is complete
304
When all transactions have been processed, output a metric regarding subsystem performance
305
Transmit information regarding subsystem performance to an aggregating server
306
END

Fig. 3

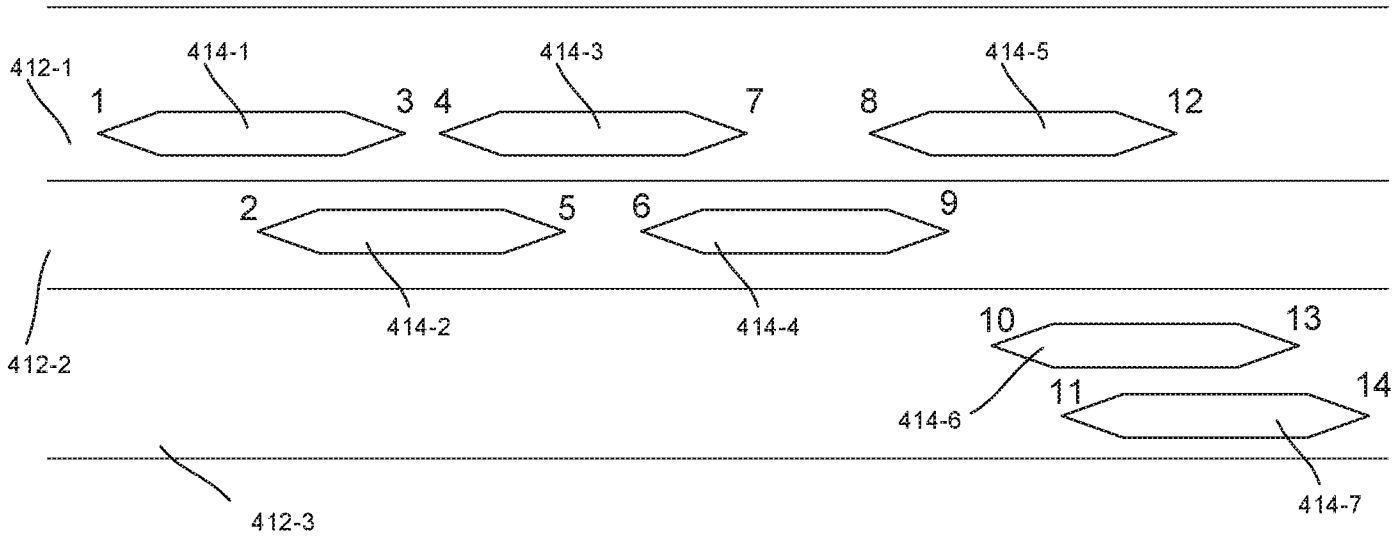
412-1
414-1
414-3
414-5
1
3
4
7
8
12
2
5
6
9
414-2
414-4
412-2
10
13
11
14
414-6
414-7
412-3

Fig. 4

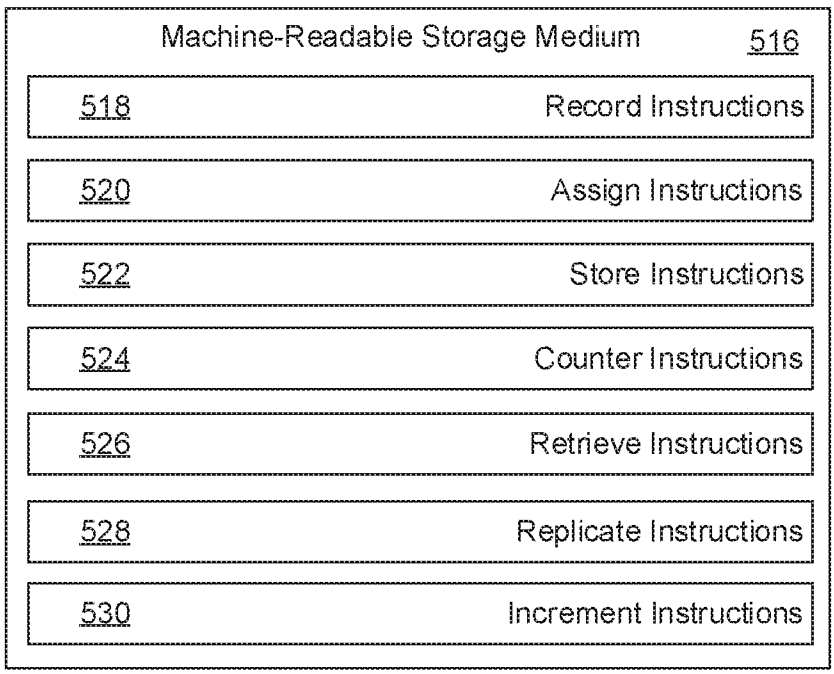
Machine-Readable Storage Medium
516
518
Record Instructions
520
Assign Instructions
522
Store Instructions
524
Counter Instructions
526
Retrieve Instructions
528
Replicate Instructions
530
Increment Instructions

Fig. 5

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US 2018/049134

A. CLASSIFICATION OF SUBJECT MATTER

G06F 11/30 *(2006.01)*
G06F 16/00 *(2019.01)*

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 9/00, 9/06, 9/44, 9/455, 9/46, 9/54, 11/00, 11/30, 11/34, 16/00, 16/20, 16/27, 17/00, H04L 12/00-12/26, 29/00-29/08

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)

PatSearch (RUPTO Internal), USPTO, PAJ, Espacenet, Information Retrieval System of FIPS

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	US 2012/0221519 A1 (ORACLE INTERNATIONAL CORPORATION) 30.08.2012, abstract, paragraphs [0056], [0061], [0105], [0106], [0109], [0118], [0120], [0129], [0136], [0140], claim 1	1-11, 14, 15
Y		12, 13
Y	US 2017/0322972 A1 (SAP SE) 09.11.2017, paragraphs [0050], [0156], [0159]	12, 13
A	US 7213113 B2 (EMC CORPORATION) 01.05.2007	1-15
A	US 2018/0034721 A1 (FACEBOOK, INC.) 01.02.2018	1-15
A	US 2016/0342447 A1 (KRYSTALLIZE TECHNOLOGIES, INC.) 24.11.2016	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 document 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
15 May 2019 (15.05.2019)	30 May 2019 (30.05.2019)

Name and mailing address of the ISA/RU:
Federal Institute of Industrial Property,
Berezhkovskaya nab., 30-1, Moscow, G-59,
GSP-3, Russia, 125993
Facsimile No: (8-495) 531-63-18, (8-499) 243-33-37

Authorized officer
D. Starshinov

Telephone No. 8(495) 531-64-81