

(12) **UK Patent**

(19) **GB**

(11) **2574990**

(13) **B**

(45) Date of B Publication

20.05.2020

(54) Title of the Invention: **Method for partial updating data content in a distributed storage network**

(51) INT CL: **G06F 11/10** (2006.01) **G06F 3/06** (2006.01)

(21) Application No: **1914709.9**

(22) Date of Filing: **27.02.2018**

Date Lodged: **11.10.2019**

(30) Priority Data:
(31) **15459259** (32) **15.03.2017** (33) **US**

(86) International Application Data:
PCT/IB2018/051222 En 27.02.2018

(87) International Publication Data:
WO2018/167590 En 20.09.2018

(43) Date of Reproduction by UK Office **25.12.2019**

(72) Inventor(s):
Adam Michael Gray
Wesley Leggette

(73) Proprietor(s):
International Business Machines Corporation
(Incorporated in USA - New York)
New Orchard Road, Armonk, New York 10504,
United States of America

(74) Agent and/or Address for Service:
IBM United Kingdom Limited
Intellectual Property Law, Hursley Park,
WINCHESTER, Hampshire, SO21 2JN,
United Kingdom

(56) Documents Cited:
CN 105930103 A **US 20160328297 A1**
US 20160070617 A1 **US 20140351624 A1**

(58) Field of Search:
As for published application 2574990 A viz:
INT CL **G06F, H04L**
Other: **WPI, EPODOC**
updated as appropriate

Additional Fields
Other: **None**

GB 2574990 B

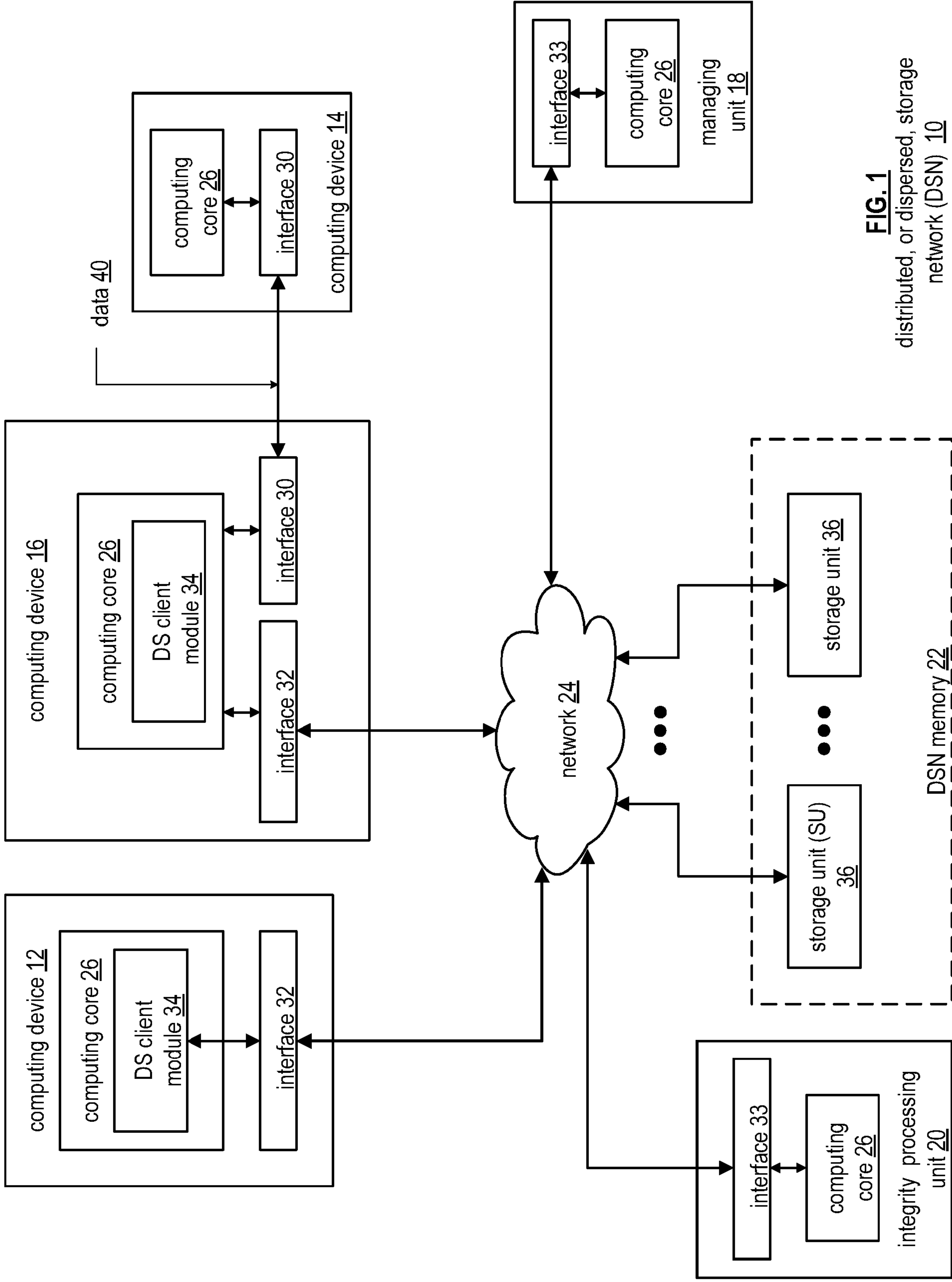


FIG. 1
distributed, or dispersed, storage
network (DSN) 10

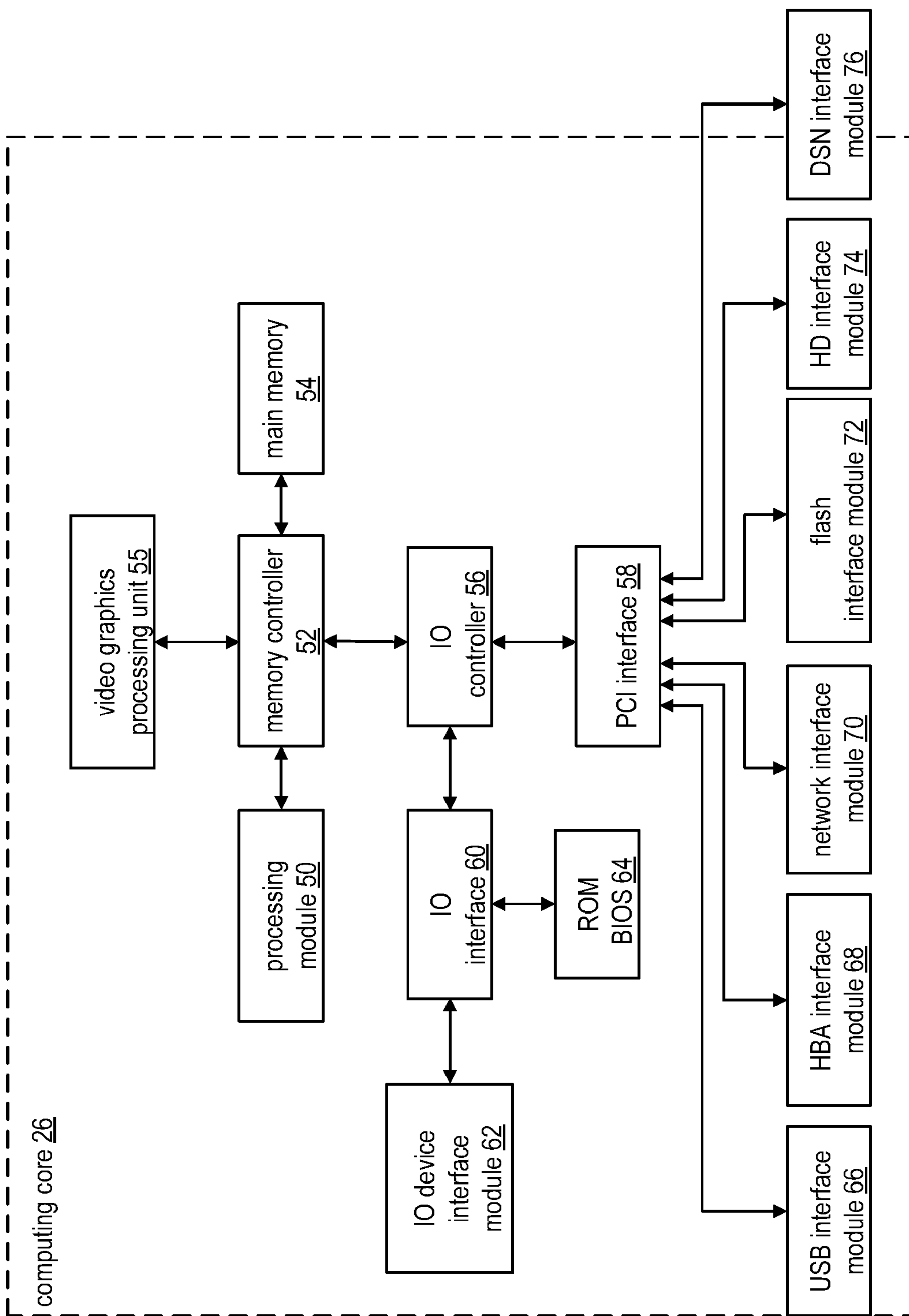


FIG. 2

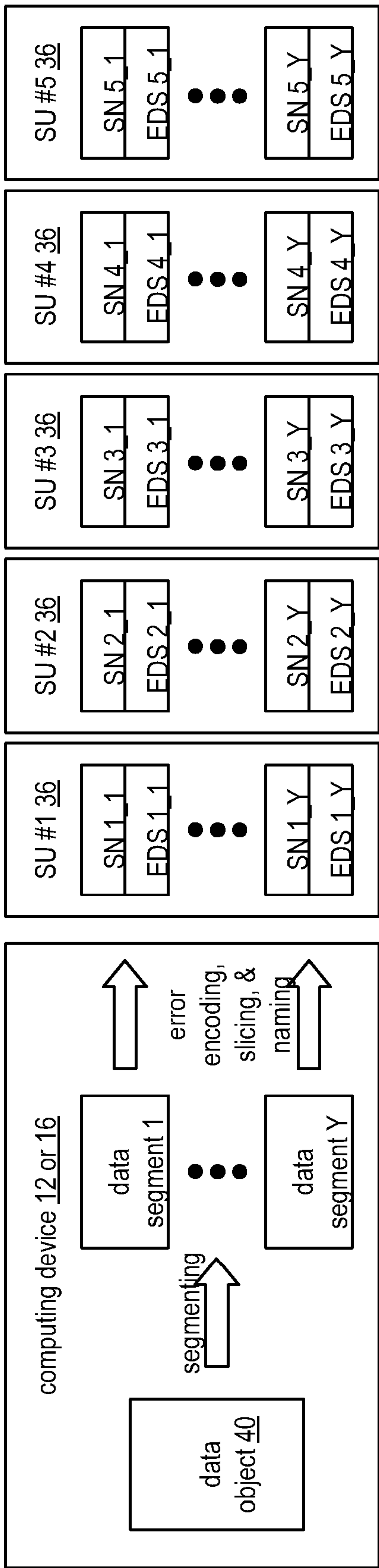


FIG. 3

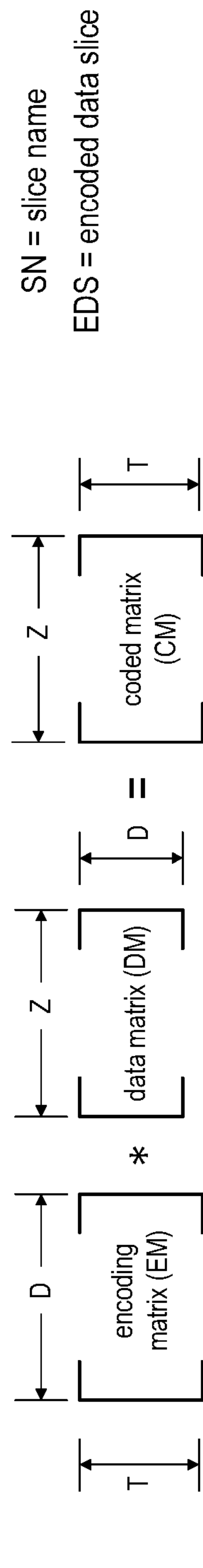


FIG. 4

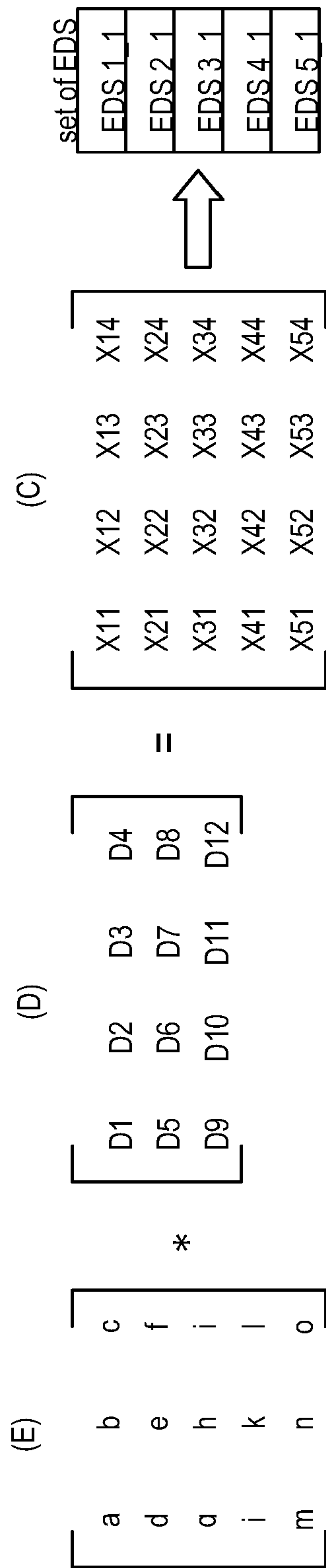


FIG. 5

slice name 80			
pillar #	data segment #	vault ID	data object ID rev. info

FIG. 6

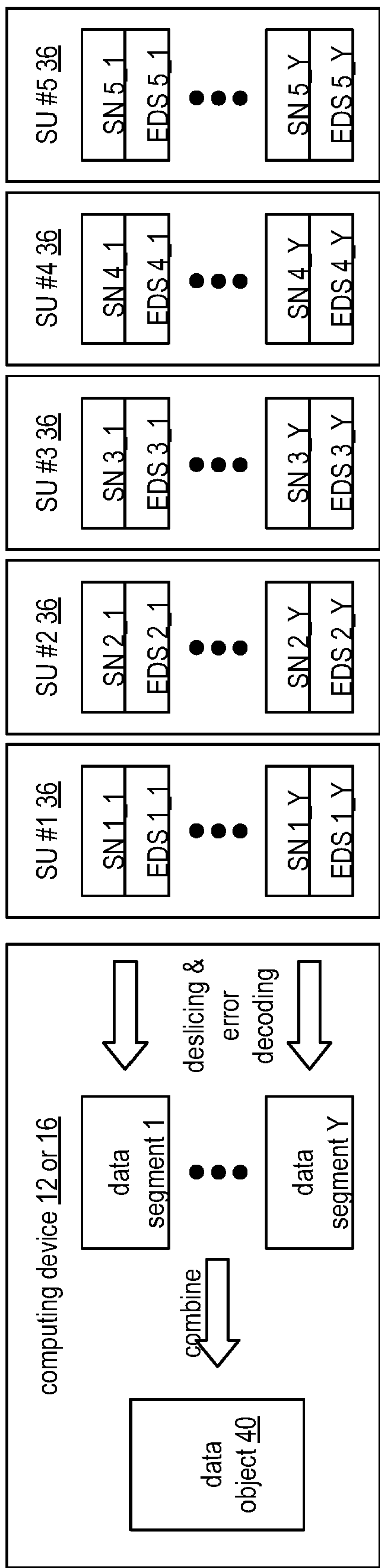


FIG. 7

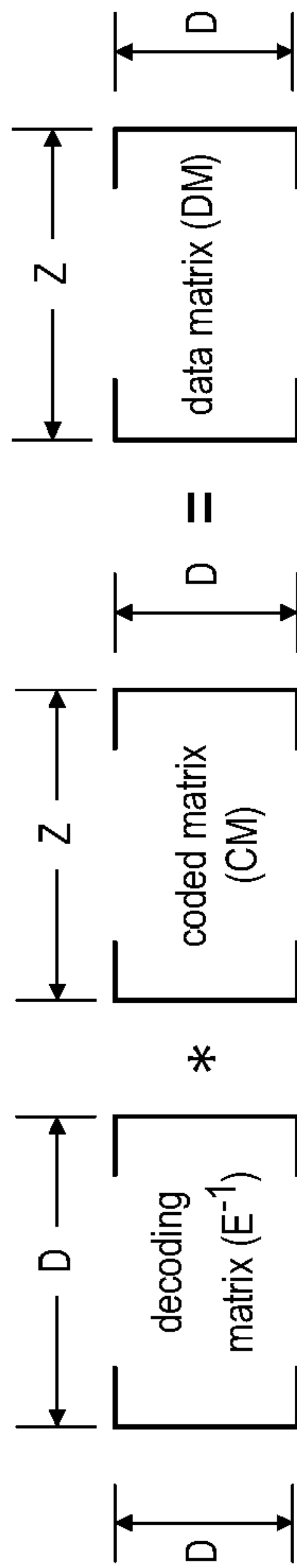


FIG. 8

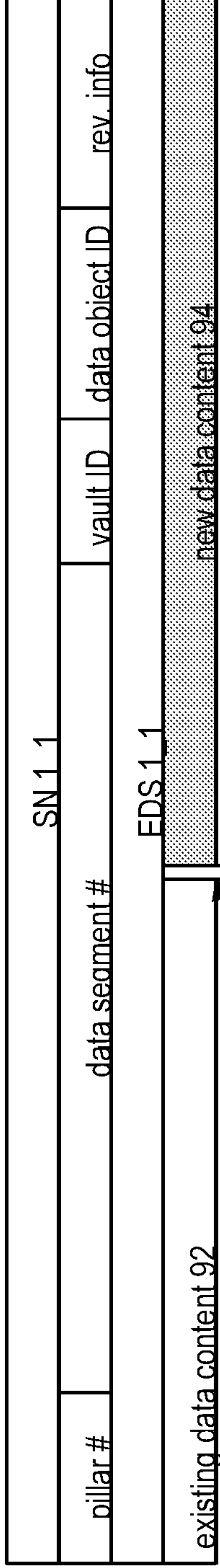


FIG. 9

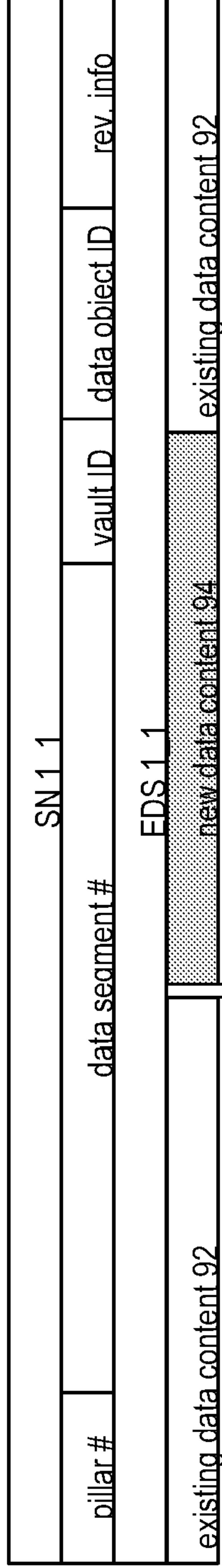


FIG. 10

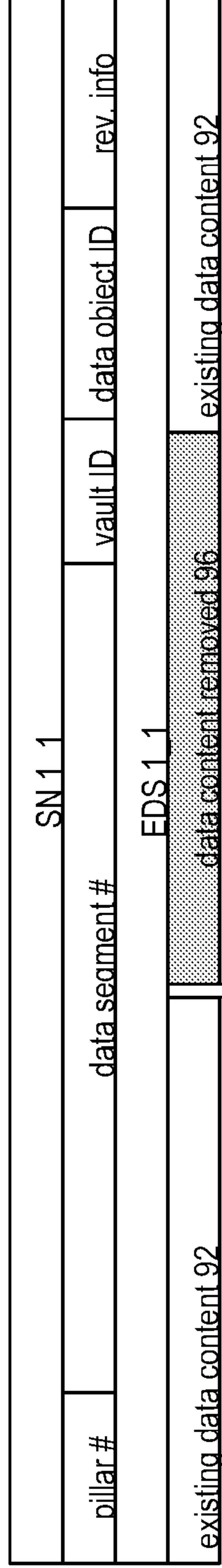


FIG. 11

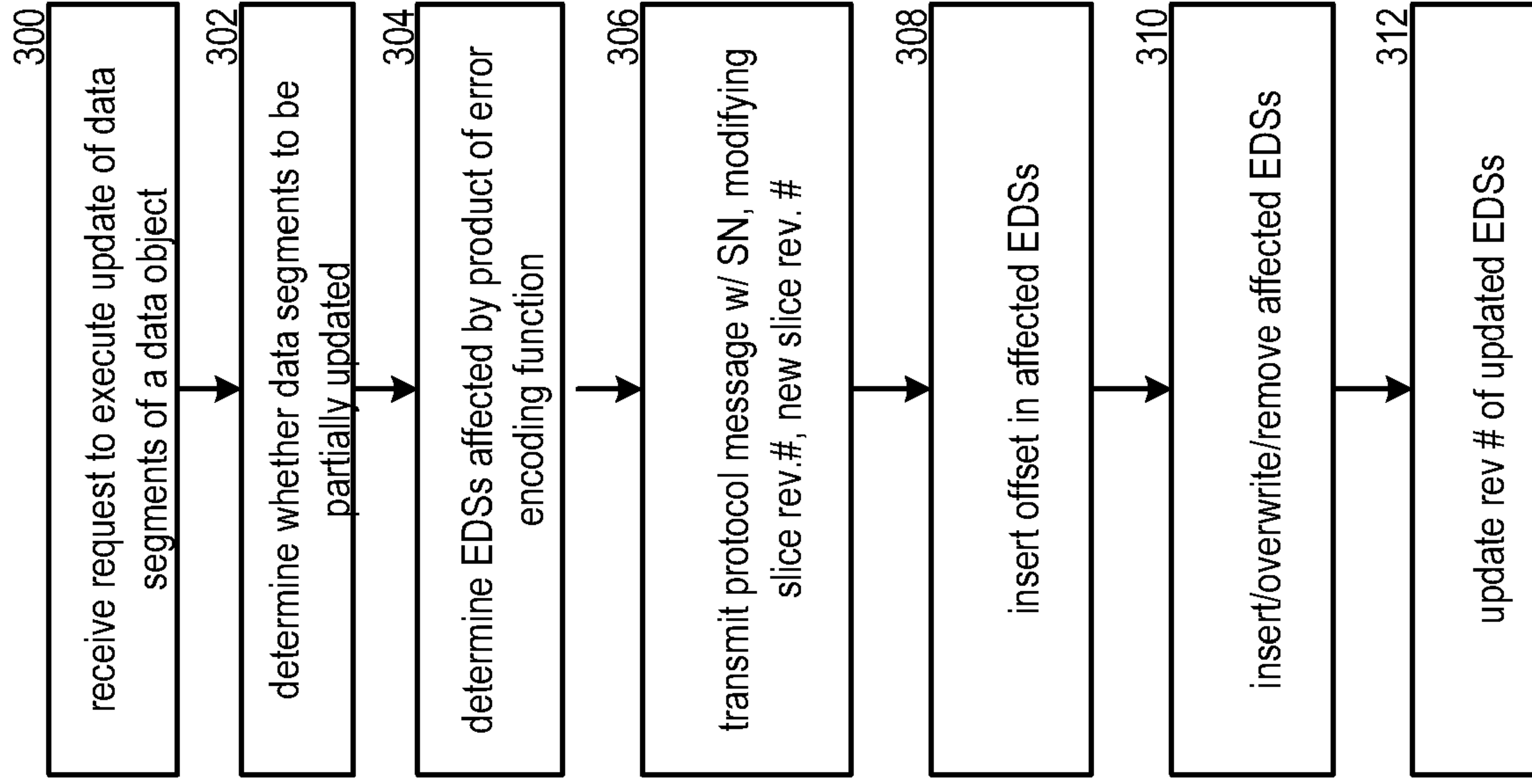


FIG. 12

METHOD FOR PARTIAL UPDATING DATA CONTENT IN A DISTRIBUTED STORAGE NETWORK

BACKGROUND OF THE INVENTION

TECHNICAL FIELD OF THE INVENTION

[0001] This invention relates generally to computer networks and more particularly to dispersing error encoded data.

DESCRIPTION OF RELATED ART

[0002] Computing devices are known to communicate data, process data, and/or store data. Such computing devices range from wireless smart phones, laptops, tablets, personal computers (PC), work stations, and video game devices, to data centers that support millions of web searches, stock trades, or on-line purchases every day. In general, a computing device includes a central processing unit (CPU), a memory system, user input/output interfaces, peripheral device interfaces, and an interconnecting bus structure.

[0003] As is further known, a computer may effectively extend its CPU by using "cloud computing" to perform one or more computing functions (e.g., a service, an application, an algorithm, an arithmetic logic function, etc.) on behalf of the computer. Further, for large services, applications, and/or functions, cloud computing may be performed by multiple cloud computing resources in a distributed manner to improve the response time for completion of the service, application, and/or function. For example, Hadoop is an open source software framework that supports distributed applications enabling application execution by thousands of computers.

[0004] In addition to cloud computing, a computer may use "cloud storage" as part of its memory system. As is known, cloud storage enables a user, via its computer, to store files, applications, etc. on an Internet storage system. The Internet storage system may include a RAID (redundant array of independent disks) system and/or a dispersed storage system that uses an error correction scheme to encode data for storage.

SUMMARY OF THE INVENTION

[0005] According to one aspect, there is provided a method for execution by one or more processing modules of one or more computing devices of a dispersed storage network (DSN), the method comprises: receiving a request to update one or more data segments of a data object; determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating; when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, selecting at least

one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.

[0006] According to another aspect, there is provided a computing device comprises: an interface for interfacing with a network; memory; and a first processing module operably coupled to the interface and to the memory, wherein the first processing module is operable to: receive request to update one or more data segments of a data object; determine whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating; when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, select at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and execute the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.

[0007] According to another aspect, there is provided a computer program product comprising a computer readable storage medium having program instructions embodied therewith, the program instructions executable by one or more computing devices of a dispersed storage network (DSN) to perform the steps of: receiving a request to update one or more data segments of a data object; determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating; when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, selecting at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.

BRIEF DESCRIPTION OF THE SEVERAL VIEWS OF THE DRAWING(S)

[0008] Embodiments of the invention will now be described, by way of example only, with reference to the accompanying drawings in which:

Figure 1 is a schematic block diagram of a dispersed or distributed storage network (DSN) in accordance with an embodiment of the present invention;

Figure 2 is a schematic block diagram of a computing core in accordance with an embodiment of the present invention;

Figure 3 is a schematic block diagram of an example of dispersed storage error encoding of data in accordance with an embodiment of the present invention;

Figure 4 is a schematic block diagram of a generic example of an error encoding function in accordance with an embodiment of the present invention;

Figure 5 is a schematic block diagram of a specific example of an error encoding function in accordance with an embodiment of the present invention;

Figure 6 is a schematic block diagram of an example of a slice name of an encoded data slice (EDS) in accordance with an embodiment of the present invention;

Figure 7 is a schematic block diagram of an example of dispersed storage error decoding of data in accordance with an embodiment of the present invention;

Figure 8 is a schematic block diagram of a generic example of an error decoding function in accordance with an embodiment of the present invention;

Figure 9 is schematic of an EDS resulting from a partial updating process in accordance with an embodiment of the present invention;

Figure 10 is schematic of another example of an EDS resulting from a partial updating process based on the use of an offset in accordance with an embodiment of the present invention;

Figure 11 is schematic of another example of an EDS resulting from a partial updating process based on the use of an offset in accordance with an embodiment of the present invention; and

Figure 12 is a logic diagram of an example of a method of partially updating DSN content in accordance with an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

[0009] Figure 1 is a schematic block diagram of an embodiment of a dispersed, or distributed, storage network (DSN) 10 that includes a plurality of computing devices 12 -16, a managing unit 18, an integrity processing unit 20, and a DSN memory 22. The components of the DSN 10 are coupled to a network 24, which may include one or more wireless and/or wire lined communication systems; one or more non-public intranet systems and/or public internet systems; and/or one or more local area networks (LAN) and/or wide area networks (WAN).

[0010] The DSN memory 22 includes a plurality of storage units 36 that may be located at geographically different sites (e.g., one in Chicago, one in Milwaukee, etc.), at a common site, or a combination thereof. For example, if the DSN memory 22 includes eight storage units 36, each storage unit is located at a different site. As another example, if the DSN memory 22 includes eight storage units 36, all eight storage units are located at the same site. As yet another example, if the DSN memory 22 includes eight storage units 36, a first pair of storage units are at a first common site, a second pair of storage units are at a second common site, a third pair of storage units are at a third common site, and a fourth pair of storage units are at a fourth common site. Note that a DSN memory 22 may include more or less than eight storage units 36. Further note that each storage unit 36 includes a computing core (as shown in Figure 2, or components thereof) and a plurality of memory devices for storing dispersed error encoded data.

[0011] Each of the computing devices 12 – 16, the managing unit 18, and the integrity processing unit 20 include a computing core 26, which includes network interfaces 30-33. Computing devices 12-16 may each be a portable computing device and/or a fixed computing device. A portable computing device may be a social networking device, a gaming device, a cell phone, a smart phone, a digital assistant, a digital music player, a digital video player, a laptop computer, a handheld computer, a tablet, a video game controller, and/or any other portable device that includes a computing core. A fixed computing device may be a computer (PC), a computer server, a cable set-top box, a satellite receiver, a television set, a printer, a fax machine, home entertainment equipment, a video game console, and/or any type of home or office computing equipment. Note that each of the managing unit 18 and the integrity processing unit 20 may be separate computing devices, may be a common computing device, and/or may be integrated into one or more of the computing devices 12-16 and/or into one or more of the storage units 36.

[0012] Each interface 30, 32, and 33 includes software and hardware to support one or more communication links via the network 24 indirectly and/or directly. For example, interface 30 supports a communication link (e.g., wired, wireless, direct, via a LAN, via the network 24, etc.) between computing devices 14 and 16. As another example, interface 32 supports communication links (e.g., a wired connection, a wireless connection, a LAN connection, and/or any other type of connection to/from the network 24) between computing devices 12 and 16 and the DSN memory 22. As yet another example, interface 33 supports a communication link for each of the managing unit 18 and the integrity processing unit 20 to the network 24.

[0013] Computing devices 12 and 16 include a dispersed storage (DS) client module 34, which enables the computing device to dispersed storage error encode and decode data (e.g., data 40) as subsequently described with reference to one or more of Figures 3-8. In this example embodiment, computing device 16 functions as a dispersed storage processing agent for computing device 14. In this role, computing device 16 dispersed storage error encodes and decodes data on behalf of computing device 14. With the use of dispersed storage error encoding and decoding, the DSN 10 is tolerant of a significant number of storage unit failures (the number of

failures is based on parameters of the dispersed storage error encoding function) without loss of data and without the need for a redundant or backup copies of the data. Further, the DSN 10 stores data for an indefinite period of time without data loss and in a secure manner (e.g., the system is very resistant to unauthorized attempts at accessing the data).

[0014] In operation, the managing unit 18 performs DS management services. For example, the managing unit 18 establishes distributed data storage parameters (e.g., vault creation, distributed storage parameters, security parameters, billing information, user profile information, etc.) for computing devices 12-14 individually or as part of a group of user devices. As a specific example, the managing unit 18 coordinates creation of a vault (e.g., a virtual memory block associated with a portion of an overall namespace of the DSN) within the DSN memory 22 for a user device, a group of devices, or for public access and establishes per vault dispersed storage (DS) error encoding parameters for a vault. The managing unit 18 facilitates storage of DS error encoding parameters for each vault by updating registry information of the DSN 10, where the registry information may be stored in the DSN memory 22, a computing device 12-16, the managing unit 18, and/or the integrity processing unit 20.

[0015] The managing unit 18 creates and stores user profile information (e.g., an access control list (ACL)) in local memory and/or within memory of the DSN memory 22. The user profile information includes authentication information, permissions, and/or the security parameters. The security parameters may include encryption/decryption scheme, one or more encryption keys, key generation scheme, and/or data encoding/decoding scheme.

[0016] The managing unit 18 creates billing information for a particular user, a user group, a vault access, public vault access, etc. For instance, the managing unit 18 tracks the number of times a user accesses a non-public vault and/or public vaults, which can be used to generate a per-access billing information. In another instance, the managing unit 18 tracks the amount of data stored and/or retrieved by a user device and/or a user group, which can be used to generate a per-data-amount billing information.

[0017] As another example, the managing unit 18 performs network operations, network administration, and/or network maintenance. Network operations includes authenticating user data allocation requests (e.g., read and/or write requests), managing creation of vaults, establishing authentication credentials for user devices, adding/deleting components (e.g., user devices, storage units, and/or computing devices with a DS client module 34) to/from the DSN 10, and/or establishing authentication credentials for the storage units 36. Network administration includes monitoring devices and/or units for failures, maintaining vault information, determining device and/or unit activation status, determining device and/or unit loading, and/or determining any other system level operation that affects the performance level of the DSN 10. Network maintenance includes facilitating replacing, upgrading, repairing, and/or expanding a device and/or unit of the DSN 10.

[0018] The integrity processing unit 20 performs rebuilding of 'bad' or missing encoded data slices. At a high level, the integrity processing unit 20 performs rebuilding by periodically attempting to retrieve/list encoded data slices, and/or slice names of the encoded data slices, from the DSN memory 22. For retrieved encoded slices, they are checked for errors due to data corruption, outdated version, etc. If a slice includes an error, it is flagged as a 'bad' slice. For encoded data slices that were not received and/or not listed, they are flagged as missing slices. Bad and/or missing slices are subsequently rebuilt using other retrieved encoded data slices that are deemed to be good slices to produce rebuilt slices. The rebuilt slices are stored in the DSN memory 22.

[0019] Figure 2 is a schematic block diagram of an embodiment of a computing core 26 that includes a processing module 50, a memory controller 52, main memory 54, a video graphics processing unit 55, an input/output (IO) controller 56, a peripheral component interconnect (PCI) interface 58, an IO interface module 60, at least one IO device interface module 62, a read only memory (ROM) basic input output system (BIOS) 64, and one or more memory interface modules. The one or more memory interface module(s) includes one or more of a universal serial bus (USB) interface module 66, a host bus adapter (HBA) interface module 68, a network interface module 70, a flash interface module 72, a hard drive interface module 74, and a DSN interface module 76.

[0020] The DSN interface module 76 functions to mimic a conventional operating system (OS) file system interface (e.g., network file system (NFS), flash file system (FFS), disk file system (DFS), file transfer protocol (FTP), web-based distributed authoring and versioning (WebDAV), etc.) and/or a block memory interface (e.g., small computer system interface (SCSI), internet small computer system interface (iSCSI), etc.). The DSN interface module 76 and/or the network interface module 70 may function as one or more of the interface 30-33 of Figure 1. Note that the IO device interface module 62 and/or the memory interface modules 66-76 may be collectively or individually referred to as IO ports.

[0021] Figure 3 is a schematic block diagram of an example of dispersed storage error encoding of data. When a computing device 12 or 16 has data to store it disperse storage error encodes the data in accordance with a dispersed storage error encoding process based on dispersed storage error encoding parameters. The dispersed storage error encoding parameters include an encoding function (e.g., information dispersal algorithm, Reed-Solomon, Cauchy Reed-Solomon, systematic encoding, non-systematic encoding, on-line codes, etc.), a data segmenting protocol (e.g., data segment size, fixed, variable, etc.), and per data segment encoding values. The per data segment encoding values include a total, or pillar width, number (T) of encoded data slices per encoding of a data segment (i.e., in a set of encoded data slices); a decode threshold number (D) of encoded data slices of a set of encoded data slices that are needed to recover the data segment; a read threshold number (R) of encoded data slices to indicate a number of encoded data slices per set to be read from storage for decoding of the data segment; and/or a write threshold number (W) to indicate a number of encoded data slices per set that must be accurately stored before the encoded data segment is deemed to have been properly stored. The dispersed storage error encoding parameters may further include slicing information (e.g., the number of encoded data slices

that will be created for each data segment) and/or slice security information (e.g., per encoded data slice encryption, compression, integrity checksum, etc.).

[0022] In the present example, Cauchy Reed-Solomon has been selected as the encoding function (a generic example is shown in Figure 4 and a specific example is shown in Figure 5); the data segmenting protocol is to divide the data object into fixed sized data segments; and the per data segment encoding values include: a pillar width of 5, a decode threshold of 3, a read threshold of 4, and a write threshold of 4. In accordance with the data segmenting protocol, the computing device 12 or 16 divides the data (e.g., a file (e.g., text, video, audio, etc.), a data object, or other data arrangement) into a plurality of fixed sized data segments (e.g., 1 through Y of a fixed size in range of Kilo-bytes to Tera-bytes or more). The number of data segments created is dependent of the size of the data and the data segmenting protocol.

[0023] The computing device 12 or 16 then disperse storage error encodes a data segment using the selected encoding function (e.g., Cauchy Reed-Solomon) to produce a set of encoded data slices. Figure 4 illustrates a generic Cauchy Reed-Solomon encoding function, which includes an encoding matrix (EM), a data matrix (DM), and a coded matrix (CM). The size of the encoding matrix (EM) is dependent on the pillar width number (T) and the decode threshold number (D) of selected per data segment encoding values. To produce the data matrix (DM), the data segment is divided into a plurality of data blocks and the data blocks are arranged into D number of rows with Z data blocks per row. Note that Z is a function of the number of data blocks created from the data segment and the decode threshold number (D). The coded matrix is produced by matrix multiplying the data matrix by the encoding matrix.

[0024] Figure 5 illustrates a specific example of Cauchy Reed-Solomon encoding with a pillar number (T) of five and decode threshold number of three. In this example, a first data segment is divided into twelve data blocks (D1 – D12). The coded matrix includes five rows of coded data blocks, where the first row of X11 – X14 corresponds to a first encoded data slice (EDS 1_1), the second row of X21 – X24 corresponds to a second encoded data slice (EDS 2_1), the third row of X31 – X34 corresponds to a third encoded data slice (EDS 3_1), the fourth row of X41 – X44 corresponds to a fourth encoded data slice (EDS 4_1), and the fifth row of X51 – X54 corresponds to a fifth encoded data slice (EDS 5_1). Note that the second number of the EDS designation corresponds to the data segment number.

[0025] Returning to the discussion of Figure 3, the computing device also creates a slice name (SN) for each encoded data slice (EDS) in the set of encoded data slices. A typical format for a slice name 80 is shown in Figure 6. As shown, the slice name (SN) 80 includes a pillar number of the encoded data slice (e.g., one of 1-T), a data segment number (e.g., one of 1 – Y), a vault identifier (ID), a data object identifier (ID), and may further include revision level information of the encoded data slices. The slice name functions as, at least part of, a DSN address for the encoded data slice for storage and retrieval from the DSN memory 22.

[0026] As a result of encoding, the computing device 12 or 16 produces a plurality of sets of encoded data slices, which are provided with their respective slice names to the storage units for storage. As shown, the first set of encoded data slices includes EDS 1_1 through EDS 5_1 and the first set of slice names includes SN 1_1 through SN 5_1 and the last set of encoded data slices includes EDS 1_Y through EDS 5_Y and the last set of slice names includes SN 1_Y through SN 5_Y.

[0027] Figure 7 is a schematic block diagram of an example of dispersed storage error decoding of a data object that was dispersed storage error encoded and stored in the example of Figure 4. In this example, the computing device 12 or 16 retrieves from the storage units at least the decode threshold number of encoded data slices per data segment. As a specific example, the computing device retrieves a read threshold number of encoded data slices.

[0028] To recover a data segment from a decode threshold number of encoded data slices, the computing device uses a decoding function as shown in Figure 8. As shown, the decoding function is essentially an inverse of the encoding function of Figure 4. The coded matrix includes a decode threshold number of rows (e.g., three in this example) and the decoding matrix is an inversion of the encoding matrix that includes the corresponding rows of the coded matrix. For example, if the coded matrix includes rows 1, 2, and 4, the encoding matrix is reduced to rows 1, 2, and 4, and then inverted to produce the decoding matrix.

[0029] Referring again to Figure 3, a data segmenting protocol is used to segment a data object into smaller segments with per data segment encoding values. Data segmenting may provide a number of advantages, including, but not limited to increased performance, improved access, and increased system efficiency. Some data types, such as meta data, index nodes, or other structured data are updated incrementally and are, thus, amenable to a partial updating process. For example, a DSN may store a first segment of data along with metadata for the data object. In this example access to the first byte of data is accessible faster while allowing the system provide for efficient streaming of the remaining bytes by allowing the system to load metadata for the object.

[0030] When updating attributes of an object with embedded content, system performance may be negatively affected, as the DS units may need to reassemble and transmit ever greater amounts of data and further to parse associated metadata out of the data object, even when executing trivial operations such simple attribute modification. This type of structured data can be common, for example, when emulating POSIX workflow on object storage, where data object access control and/or time stamps may be manipulated as frequently as during every access. Figures 9-11 are schematics of partially updated EDSs resulting from a partial updating process based on the use of an offset. The partial updating process can be considered, for example, to be extensions of the dsnet protocol with the operations detailed in Figures 9-11. As a specific example, in Figure 9 the partial updating process inserts an offset at the provided offset position and new content 94 into EDS 1_1 at the offset without

overwriting the existing data content 92 that existed prior to the partial updating process (i.e. the existing data content 92 is pushed behind the new data content 94).

[0031] Figure 10 is schematic of another example of an EDS resulting from a partial updating process based on the use of an offset. In this example, the partial updating process inserts an offset at a provided offset position and new data content 94 into the existing data content 92, replacing the existing data content 92 up to the length of the new data content 94.

[0032] Figure 11 is schematic of another example of an EDS resulting from a partial updating process based on the use of an offset. As a specific example, an offset and either a data range or an end position is provided in the partial updating process, wherein the partial updating process removes data content removed 96 from the existing data content 92. In this example the partial updating process includes inserting the offset at the offset position and removes data until either an endpoint is received or the specified range of data is removed.

[0033] Figure 12 is a logic diagram of an example of a method of partially updating DSN content in accordance with an embodiment of the present invention. As a specific example, at step 300 the DSN receives a request to partially update one or more data segments of a data object. The DSN determines in step 302 whether the data segments being updated are amenable to partial updating, and when the data segments being updated are amenable to partial updating, the DSN further determines in step 304 which EDSs associated with the data segments being updated will be affected by the partial update. Referring to Figure 5, and using the specific example of Cauchy Reed-Solomon encoding, the DSM may determine which EDSs will be affected by the partial update based on the product of a matrix operation. In the example of Figure 5, the first row of X11 – X14 corresponds to a first encoded data slice (EDS 1_1), and $X11 = aD1 + bD5 + D9$. Thus, if only D1 is to be partially updated the first encoded data slice (EDS 1_1) will need to be partially updated, along with any encoded data slices that include D1.

[0034] Once the EDSs affected by the partial updating are determined, the DSN transmits in step 306 a protocol message including the EDS slice name, the revision # for the EDS (slice) to be modified and the revision # for the new (partially updated) EDS to the SU, where the offset can be inserted into the affected EDSs at step 308 and new content may be prepended, appended, inserted and obsolete data can be deleted as determined by the DSN in step 310. The revision #s for the affected EDS are updated in step 312. In an alternative example, new revision #s may be provided for all EDS associated with the data segment being partially updated, even when one or more EDSs are unaffected by partial update. Using this example, the aggregate data required to execute a modification is correlated to the actual modification itself, instead of being dependent on the data source being updated.

[0035] It is noted that terminologies as may be used herein such as bit stream, stream, signal sequence, etc. (or their equivalents) have been used interchangeably to describe digital information whose content corresponds to any of a number of desired types (e.g., data, video, speech, audio, etc. any of which may generally be referred to as 'data').

[0036] As may be used herein, the terms "substantially" and "approximately" provides an industry-accepted tolerance for its corresponding term and/or relativity between items. Such an industry-accepted tolerance ranges from less than one percent to fifty percent and corresponds to, but is not limited to, component values, integrated circuit process variations, temperature variations, rise and fall times, and/or thermal noise. Such relativity between items ranges from a difference of a few percent to magnitude differences. As may also be used herein, the term(s) "configured to", "operably coupled to", "coupled to", and/or "coupling" includes direct coupling between items and/or indirect coupling between items via an intervening item (e.g., an item includes, but is not limited to, a component, an element, a circuit, and/or a module) where, for an example of indirect coupling, the intervening item does not modify the information of a signal but may adjust its current level, voltage level, and/or power level. As may further be used herein, inferred coupling (i.e., where one element is coupled to another element by inference) includes direct and indirect coupling between two items in the same manner as "coupled to". As may even further be used herein, the term "configured to", "operable to", "coupled to", or "operably coupled to" indicates that an item includes one or more of power connections, input(s), output(s), etc., to perform, when activated, one or more its corresponding functions and may further include inferred coupling to one or more other items. As may still further be used herein, the term "associated with", includes direct and/or indirect coupling of separate items and/or one item being embedded within another item.

[0037] As may be used herein, the term "compares favorably", indicates that a comparison between two or more items, signals, etc., provides a desired relationship. For example, when the desired relationship is that signal 1 has a greater magnitude than signal 2, a favorable comparison may be achieved when the magnitude of signal 1 is greater than that of signal 2 or when the magnitude of signal 2 is less than that of signal 1. As may be used herein, the term "compares unfavorably", indicates that a comparison between two or more items, signals, etc., fails to provide the desired relationship.

[0038] As may also be used herein, the terms "processing module", "processing circuit", "processor", and/or "processing unit" may be a single processing device or a plurality of processing devices. Such a processing device may be a microprocessor, micro-controller, digital signal processor, microcomputer, central processing unit, field programmable gate array, programmable logic device, state machine, logic circuitry, analog circuitry, digital circuitry, and/or any device that manipulates signals (analog and/or digital) based on hard coding of the circuitry and/or operational instructions. The processing module, module, processing circuit, and/or processing unit may be, or further include, memory and/or an integrated memory element, which may be a single memory device, a plurality of memory devices, and/or embedded circuitry of another processing module, module, processing circuit, and/or

processing unit. Such a memory device may be a read-only memory, random access memory, volatile memory, non-volatile memory, static memory, dynamic memory, flash memory, cache memory, and/or any device that stores digital information. Note that if the processing module, module, processing circuit, and/or processing unit includes more than one processing device, the processing devices may be centrally located (e.g., directly coupled together via a wired and/or wireless bus structure) or may be distributedly located (e.g., cloud computing via indirect coupling via a local area network and/or a wide area network). Further note that if the processing module, module, processing circuit, and/or processing unit implements one or more of its functions via a state machine, analog circuitry, digital circuitry, and/or logic circuitry, the memory and/or memory element storing the corresponding operational instructions may be embedded within, or external to, the circuitry comprising the state machine, analog circuitry, digital circuitry, and/or logic circuitry. Still further note that, the memory element may store, and the processing module, module, processing circuit, and/or processing unit executes, hard coded and/or operational instructions corresponding to at least some of the steps and/or functions illustrated in one or more of the Figures. Such a memory device or memory element can be included in an article of manufacture.

[0039] One or more embodiments have been described above with the aid of method steps illustrating the performance of specified functions and relationships thereof. The boundaries and sequence of these functional building blocks and method steps have been arbitrarily defined herein for convenience of description. Alternate boundaries and sequences can be defined so long as the specified functions and relationships are appropriately performed. Any such alternate boundaries or sequences are thus within the scope and spirit of the claims. Further, the boundaries of these functional building blocks have been arbitrarily defined for convenience of description. Alternate boundaries could be defined as long as the certain significant functions are appropriately performed. Similarly, flow diagram blocks may also have been arbitrarily defined herein to illustrate certain significant functionality.

[0040] To the extent used, the flow diagram block boundaries and sequence could have been defined otherwise and still perform the certain significant functionality. Such alternate definitions of both functional building blocks and flow diagram blocks and sequences are thus within the scope and spirit of the claims. One of average skill in the art will also recognize that the functional building blocks, and other illustrative blocks, modules and components herein, can be implemented as illustrated or by discrete components, application specific integrated circuits, processors executing appropriate software and the like or any combination thereof.

[0041] In addition, a flow diagram may include a “start” and/or “continue” indication. The “start” and “continue” indications reflect that the steps presented can optionally be incorporated in or otherwise used in conjunction with other routines. In this context, “start” indicates the beginning of the first step presented and may be preceded by other activities not specifically shown. Further, the “continue” indication reflects that the steps presented may be performed multiple times and/or may be succeeded by other activities not specifically shown.

Further, while a flow diagram indicates a particular ordering of steps, other orderings are likewise possible provided that the principles of causality are maintained.

[0042] The one or more embodiments are used herein to illustrate one or more aspects, one or more features, one or more concepts, and/or one or more examples. A physical embodiment of an apparatus, an article of manufacture, a machine, and/or of a process may include one or more of the aspects, features, concepts, examples, etc. described with reference to one or more of the embodiments discussed herein. Further, from figure to figure, the embodiments may incorporate the same or similarly named functions, steps, modules, etc. that may use the same or different reference numbers and, as such, the functions, steps, modules, etc. may be the same or similar functions, steps, modules, etc. or different ones.

[0043] Unless specifically stated to the contra, signals to, from, and/or between elements in a figure of any of the figures presented herein may be analog or digital, continuous time or discrete time, and single-ended or differential. For instance, if a signal path is shown as a single-ended path, it also represents a differential signal path. Similarly, if a signal path is shown as a differential path, it also represents a single-ended signal path. While one or more particular architectures are described herein, other architectures can likewise be implemented that use one or more data buses not expressly shown, direct connectivity between elements, and/or indirect coupling between other elements as recognized by one of average skill in the art.

[0044] The term "module" is used in the description of one or more of the embodiments. A module implements one or more functions via a device such as a processor or other processing device or other hardware that may include or operate in association with a memory that stores operational instructions. A module may operate independently and/or in conjunction with software and/or firmware. As also used herein, a module may contain one or more sub-modules, each of which may be one or more modules.

[0045] As may further be used herein, a computer readable memory includes one or more memory elements. A memory element may be a separate memory device, multiple memory devices, or a set of memory locations within a memory device. Such a memory device may be a read-only memory, random access memory, volatile memory, non-volatile memory, static memory, dynamic memory, flash memory, cache memory, and/or any device that stores digital information. The memory device may be in a form a solid state memory, a hard drive memory, cloud memory, thumb drive, server memory, computing device memory, and/or other physical medium for storing digital information.

[0046] While particular combinations of various functions and features of the one or more embodiments have been expressly described herein, other combinations of these features and functions are likewise possible. The present disclosure is not limited by the particular examples disclosed herein and expressly incorporates these other combinations.

According to preferred embodiments, there are provided the following clauses:

1. A method for execution by one or more processing modules of one or more computing devices of a dispersed storage network (DSN), the method comprises:
 - receiving a request to update one or more data segments of a data object;
 - determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating;
 - when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, selecting at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and
 - executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.
2. The method of clause 1 further comprises:
 - updating a revision number for the at least one EDS of the one of more EDSs eligible for partial updating for which partial updating process is executed.
3. The method of clause 1 further comprises:
 - updating a revision number for each of the (EDSs) of the plurality of EDSs associated with the one or more data segments of the data object when the partial updating process is applied to at least one EDS of the one of more EDSs eligible for partial updating.
4. The method of clause 1, wherein the determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating is based on the data object including embedded content.
5. The method of clause 1, wherein the executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating includes an operation that includes at least one offset in each of the EDSs on which partial updating is executed.
6. The method of clause 5, wherein the executing the partial updating process includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating; and wherein updated content is inserted into the at least one EDS of the one of more EDSs eligible for partial updating without overwriting

content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

7. The method of clause 5, wherein the executing the partial updating process includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating; and wherein updated content inserted into the at least one EDS of the one of more EDSs eligible for the partial updating overwrites at least some content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

8. The method of clause 5 wherein there is at least one end position for at least one of the EDSs on which partial updating is executed, and wherein the executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process between the at least one offset and the end position.

9. The method of clause 5 wherein there is at least one range for at least one of the EDSs on which partial updating is executed, wherein the executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process for the range starting at the at least one offset.

10. The method of clause 1, wherein the determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating is based on the one or more encoded data slices (EDSs) associated with particular data segments of the data object to be updated being a product of a dispersed storage error encoding function of the particular data segments of the data object to be updated.

11. A computing device comprises:
an interface for interfacing with a network;
memory; and
a first processing module operably coupled to the interface and to the memory, wherein the first processing module is operable to:
receive request to update one or more data segments of a data object;
determine whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating;
when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, select at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more

EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and

execute the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.

12. The computing device of clause 11, wherein the first processing module is further operable to:
update a revision number for the at least one EDS of the one of more EDSs eligible for partial updating for which partial updating process is executed.

13. The computing device of clause 11, wherein the first processing module is further operable to:
update a revision number for each of the (EDSs) of the plurality of EDSs associated with the one or more data segments of the data object when the partial updating process is applied to at least one EDS of the one of more EDSs eligible for partial updating.

14. The computing device of clause 11, wherein the first processing module is further operable to:
determine whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object is eligible for partial updating based on the data object including embedded content.

15. The computing device of clause 11, wherein the first processing module is further operable to:
include an operation that includes at least one offset in each of the EDSs on which partial updating is executed.

16. The computing device of clause 15, wherein the operation that includes at least one offset in each of the EDSs on which partial updating is executed includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating and further includes inserting updated content into the at least one EDS of the one of more EDSs eligible for partial updating without overwriting content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

17. The computing device of clause 15, wherein the first processing module is further operable to:
insert the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating; and wherein updated content inserted into the at least one EDS of the one of more EDSs eligible for partial updating overwrites at least some content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

18. The computing device of clause 15, wherein the operation that includes at least one offset in each of the EDSs on which partial updating is executed includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating, and further includes inserting an end position for at least one of the EDSs on which partial updating is executed, and

wherein executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process between the at least one offset and the end position.

19. The computing device of clause 15, wherein the operation that includes at least one offset in each of the EDSs on which partial updating is executed includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating, and further includes at least one range for at least one of the EDSs on which partial updating is executed, wherein the executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for the partial updating prior to the executing the partial updating process for the range starting at the at least one offset.

20. The computing device of clause 11, wherein the first processing module is further operable to:
determine whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating based on the one or more encoded data slices (EDSs) associated with particular data segments of the data object to be updated being a product of a dispersed storage error encoding function of the particular data segments of the data object to be updated.

21. A computer program product comprising a computer readable storage medium having program instructions embodied therewith, the program instructions executable by one or more computing devices of a dispersed storage network (DSN) to perform the steps of:

receiving a request to update one or more data segments of a data object;

determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating;

when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, selecting at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and

executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.

22. The computer program product of clause 21, wherein the program instructions are executable by one or more computing devices of a dispersed storage network (DSN) to further perform the step of:
updating a revision number for the at least one EDS of the one of more EDSs eligible for partial updating for which partial updating process is executed.
23. The computer program product of clause 21, wherein the program instructions are executable by one or more computing devices of a dispersed storage network (DSN) to further perform the step of:
updating a revision number for each of the (EDSs) of the plurality of EDSs associated with the one or more data segments of the data object when the partial updating process is applied to at least one EDS of the one of more EDSs eligible for partial updating.
24. The computer program product of clause 21, wherein the determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating is based on the data object including embedded content.
25. The computer program product of clause 21, wherein the executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating includes an operation that includes at least one offset in each of the EDSs on which partial updating is executed.
26. The computer program product of clause 25, wherein the executing the partial updating process includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating; and wherein updated content is inserted into the at least one EDS of the one of more EDSs eligible for partial updating without overwriting content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.
27. The computer program product of clause 25, wherein the executing the partial updating process includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating; and wherein updated content inserted into the at least one EDS of the one of more EDSs eligible for the partial updating overwrites at least some content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.
28. The computer program product of clause 25, wherein there is at least one end position for at least one of the EDSs on which partial updating is executed, and wherein the executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process between the at least one offset and the end position.
29. The computer program product of clause 25, wherein there is at least one range for at least one of the EDSs on which partial updating is executed, wherein the executing the partial updating process includes removing

content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process for the range starting at the at least one offset.

30. The computer program product of clause 21, wherein the determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating is based on the one or more encoded data slices (EDSs) associated with particular data segments of the data object to be updated being a product of a dispersed storage error encoding function of the particular data segments of the data object to be updated.

CLAIMS

1. A method for execution by one or more processing modules of one or more computing devices of a dispersed storage network (DSN), the method comprises:
 - receiving a request to update one or more data segments of a data object;
 - determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating;
 - when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, selecting at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and
 - executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.
2. The method of claim 1 further comprises:
 - updating a revision number for the at least one EDS of the one of more EDSs eligible for partial updating for which partial updating process is executed.
3. The method of claim 1 further comprises:
 - updating a revision number for each of the (EDSs) of the plurality of EDSs associated with the one or more data segments of the data object when the partial updating process is applied to at least one EDS of the one of more EDSs eligible for partial updating.
4. The method of claim 1, wherein the determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating is based on the data object including embedded content.
5. The method of claim 1, wherein the executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating includes an operation that includes at least one offset in each of the EDSs on which partial updating is executed.
6. The method of claim 5, wherein the executing the partial updating process includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating; and wherein updated content is inserted into the at least one EDS of the one of more EDSs eligible for partial updating without overwriting

content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

7. The method of claim 5, wherein the executing the partial updating process includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating; and wherein updated content inserted into the at least one EDS of the one of more EDSs eligible for the partial updating overwrites at least some content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

8. The method of claim 5 wherein there is at least one end position for at least one of the EDSs on which partial updating is executed, and wherein the executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process between the at least one offset and the end position.

9. The method of claim 5 wherein there is at least one range for at least one of the EDSs on which partial updating is executed, wherein the executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process for the range starting at the at least one offset.

10. The method of claim 1, wherein the determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating is based on the one or more encoded data slices (EDSs) associated with particular data segments of the data object to be updated being a product of a dispersed storage error encoding function of the particular data segments of the data object to be updated.

11. A computing device comprises:

an interface for interfacing with a network;

memory; and

a first processing module operably coupled to the interface and to the memory, wherein the first processing module is operable to:

receive request to update one or more data segments of a data object;

determine whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating;

when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, select at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs

eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and

execute the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.

12. The computing device of claim 11, wherein the first processing module is further operable to:
update a revision number for the at least one EDS of the one of more EDSs eligible for partial updating for which partial updating process is executed.

13. The computing device of claim 11, wherein the first processing module is further operable to:
update a revision number for each of the (EDSs) of the plurality of EDSs associated with the one or more data segments of the data object when the partial updating process is applied to at least one EDS of the one of more EDSs eligible for partial updating.

14. The computing device of claim 11, wherein the first processing module is further operable to:
determine whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object is eligible for partial updating based on the data object including embedded content.

15. The computing device of claim 11, wherein the first processing module is further operable to:
include an operation that includes at least one offset in each of the EDSs on which partial updating is executed.

16. The computing device of claim 15, wherein the operation that includes at least one offset in each of the EDSs on which partial updating is executed includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating and further includes inserting updated content into the at least one EDS of the one of more EDSs eligible for partial updating without overwriting content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

17. The computing device of claim 15, wherein the first processing module is further operable to:
insert the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating;
and wherein updated content inserted into the at least one EDS of the one of more EDSs eligible for partial updating overwrites at least some content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process.

18. The computing device of claim 15, wherein the operation that includes at least one offset in each of the EDSs on which partial updating is executed includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating, and further includes inserting an end position for at least one of the EDSs on which partial updating is executed, and

wherein executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for partial updating prior to the executing the partial updating process between the at least one offset and the end position.

19. The computing device of claim 15, wherein the operation that includes at least one offset in each of the EDSs on which partial updating is executed includes inserting the at least one offset into the at least one EDS of the one of more EDSs eligible for partial updating, and further includes at least one range for at least one of the EDSs on which partial updating is executed, wherein the executing the partial updating process includes removing content existing in the at least one EDS of the one of more EDSs eligible for the partial updating prior to the executing the partial updating process for the range starting at the at least one offset.

20. The computing device of claim 11, wherein the first processing module is further operable to:
determine whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating based on the one or more encoded data slices (EDSs) associated with particular data segments of the data object to be updated being a product of a dispersed storage error encoding function of the particular data segments of the data object to be updated.

21. A computer program product comprising a computer readable storage medium having program instructions embodied therewith, the program instructions executable by one or more computing devices of a dispersed storage network (DSN). to perform the steps of:

receiving a request to update one or more data segments of a data object;

determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the one or more data segments of the data object are eligible for partial updating;

when one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object that are to be updated are eligible for partial updating, selecting at least one EDS of the one of more EDSs eligible for partial updating that would be affected and at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during a partial updating process applied to the one of more EDSs; and

executing the partial updating process for the at least one EDS of the one of more EDSs eligible for partial updating that would be affected during the partial updating process being applied to the one of more EDSs while excluding the at least one other EDS of the one of more EDSs eligible for partial updating that would be unaffected during the partial updating process.

22. The computer program product of claim 21, wherein the program instructions are executable by one or more computing devices of a dispersed storage network (DSN) to further perform the step of:

updating a revision number for the at least one EDS of the one of more EDSs eligible for partial updating for which partial updating process is executed.

23. The computer program product of claim 21, wherein the program instructions are executable by one or more computing devices of a dispersed storage network (DSN) to further perform the step of:

updating a revision number for each of the (EDSs) of the plurality of EDSs associated with the one or more data segments of the data object when the partial updating process is applied to at least one EDS of the one of more EDSs eligible for partial updating.

24. The computer program product of claim 21, wherein the determining whether one or more encoded data slices (EDSs) of a plurality of EDSs associated with the data segments of the data object are eligible for partial updating is based on the data object including embedded content.

25. A computer program comprising program code means for performing the method of any of claims 1 to 10 when the program is executed by a computer.