US012386752B2

(12) **United States Patent**
Subramanian et al.

(10) **Patent No.: US 12,386,752 B2**
(45) **Date of Patent: Aug. 12, 2025**

(54) **SYSTEMS AND METHODS FOR COOPERATIVE CACHING BETWEEN FILE SYSTEMS AND BLOCK STORAGE**

(71) Applicant: **Dell Products L.P.**, Round Rock, TX (US)

(72) Inventors: **Vasudevan Subramanian**, Chapel Hill, NC (US); **Philippe Armangau**, Kalispell, MT (US); **Walter Forrester**, Berkeley Heights, NJ (US); **Frederic Corniquet**, Le Pecq (FR)

(73) Assignee: **Dell Products L.P.**, Round Rock, TX (US)

(*) Notice: Subject to any disclaimer, the term of this patent is extended or adjusted under 35 U.S.C. 154(b) by 226 days.

(21) Appl. No.: **18/160,077**

(22) Filed: **Jan. 26, 2023**

(65) **Prior Publication Data**

US 2024/0256458 A1 Aug. 1, 2024

(51) **Int. Cl.**
G06F 12/0871 (2016.01)
G06F 12/0895 (2016.01)
G06F 13/16 (2006.01)

(52) **U.S. Cl.**
CPC ***G06F 12/0871*** (2013.01); ***G06F 12/0895*** (2013.01); ***G06F 13/1668*** (2013.01)

(58) **Field of Classification Search**
CPC G06F 12/0871; G06F 12/0895; G06F 12/0804; G06F 12/0868; G06F 13/1668
See application file for complete search history.

(56) **References Cited**

U.S. PATENT DOCUMENTS

6,851,024 B1 * 2/2005 Wilkes G06F 12/0866 711/E12.019
2021/0263862 A1 * 8/2021 Gupta G06F 11/3452
2022/0100518 A1 * 3/2022 Tomei G06F 12/0895

OTHER PUBLICATIONS

Park et al., BCD Deduplication: Effective Memory Compression using Partial Cache-Line Deduplication, 2021, ACM, pp. 52-64 (Year: 2021).*

* cited by examiner

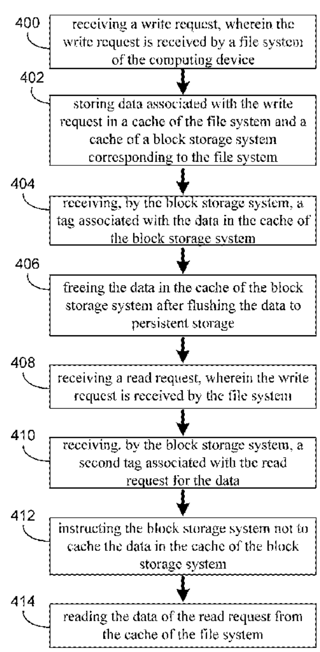
Primary Examiner — Gary W. Cygiel
(74) *Attorney, Agent, or Firm* — Brian J. Colandreo; Heath M. Sargeant; Holland & Knight LLP

(57) **ABSTRACT**

A method, computer program product, and computer system for receiving, by a computing device, a write request, wherein the write request may be received by a file system of the computing device. Data associated with the write request may be stored in a cache of the file system and a cache of a block storage system corresponding to the file system. A tag associated with the data in the cache of the block storage system may be received by the block storage system. The data in the cache of the block storage system may be freed after flushing the data to persistent storage based upon, at least in part, the tag associated with the data in the cache of the block storage system. As a result, there is optimization for the read path due to no double caching in the block cache.

17 Claims, 5 Drawing Sheets

10



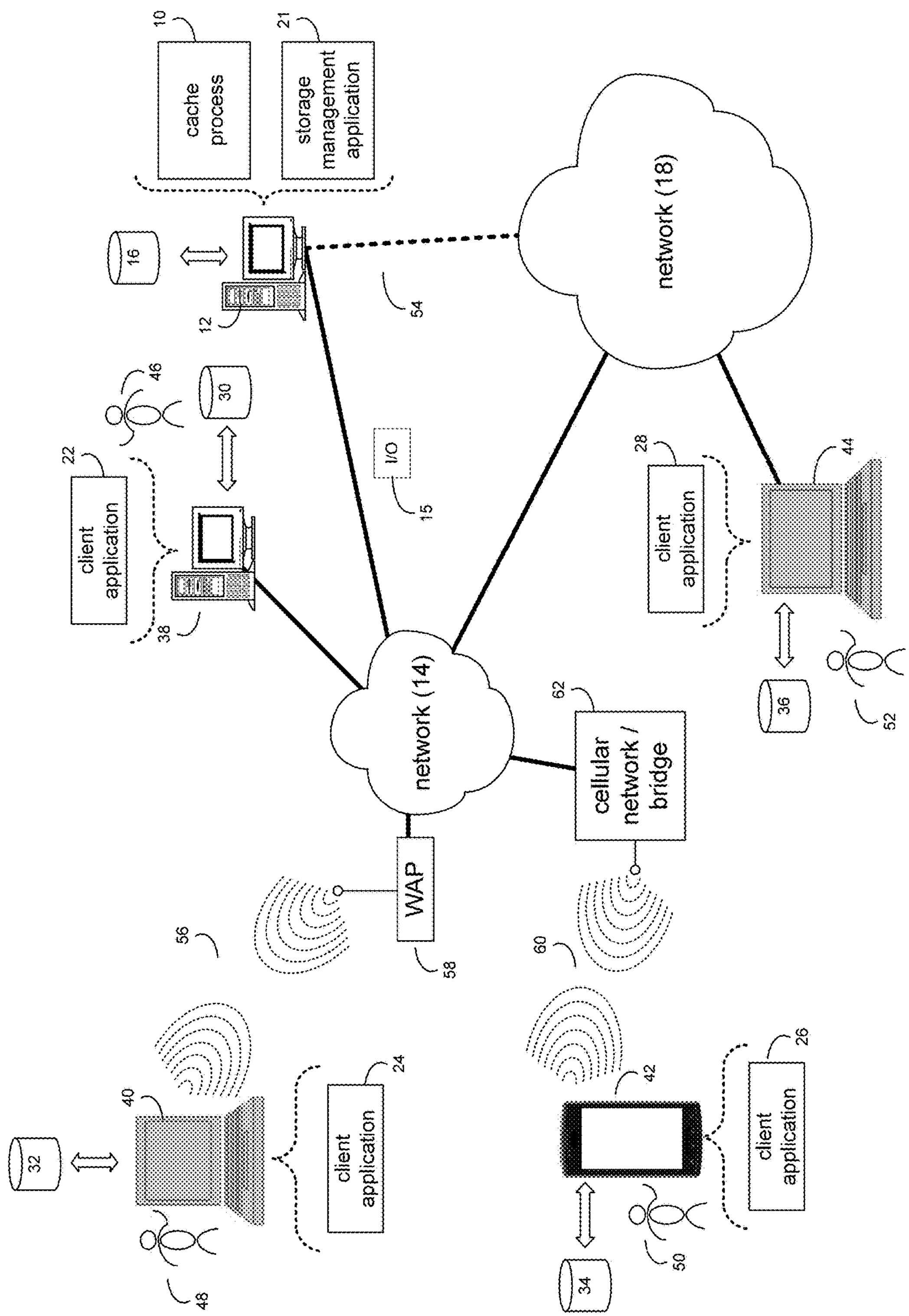

10
cache process
21
storage management application
16
12
54
network (18)
46
30
22
client application
38
I/O
15
28
client application
44
36
52
network (14)
62
cellular network / bridge
WAP
56
58
60
40
32
24
client application
48
42
26
client application
34
50

FIG. 1

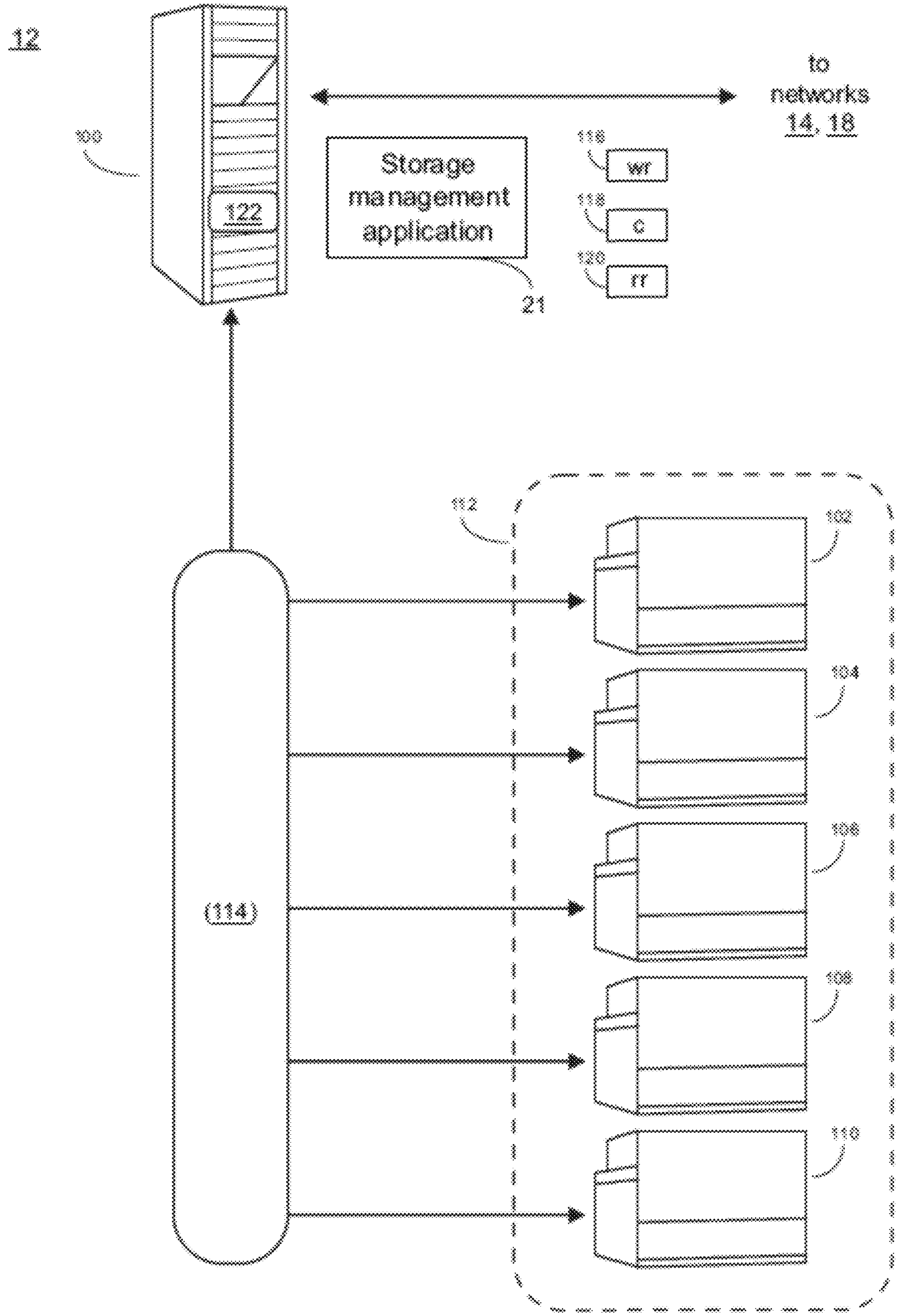

12
100
122
Storage management application
21
to networks 14, 18
116
wr
118
c
120
rr
112
102
104
106
108
110
(114)

FIG. 2

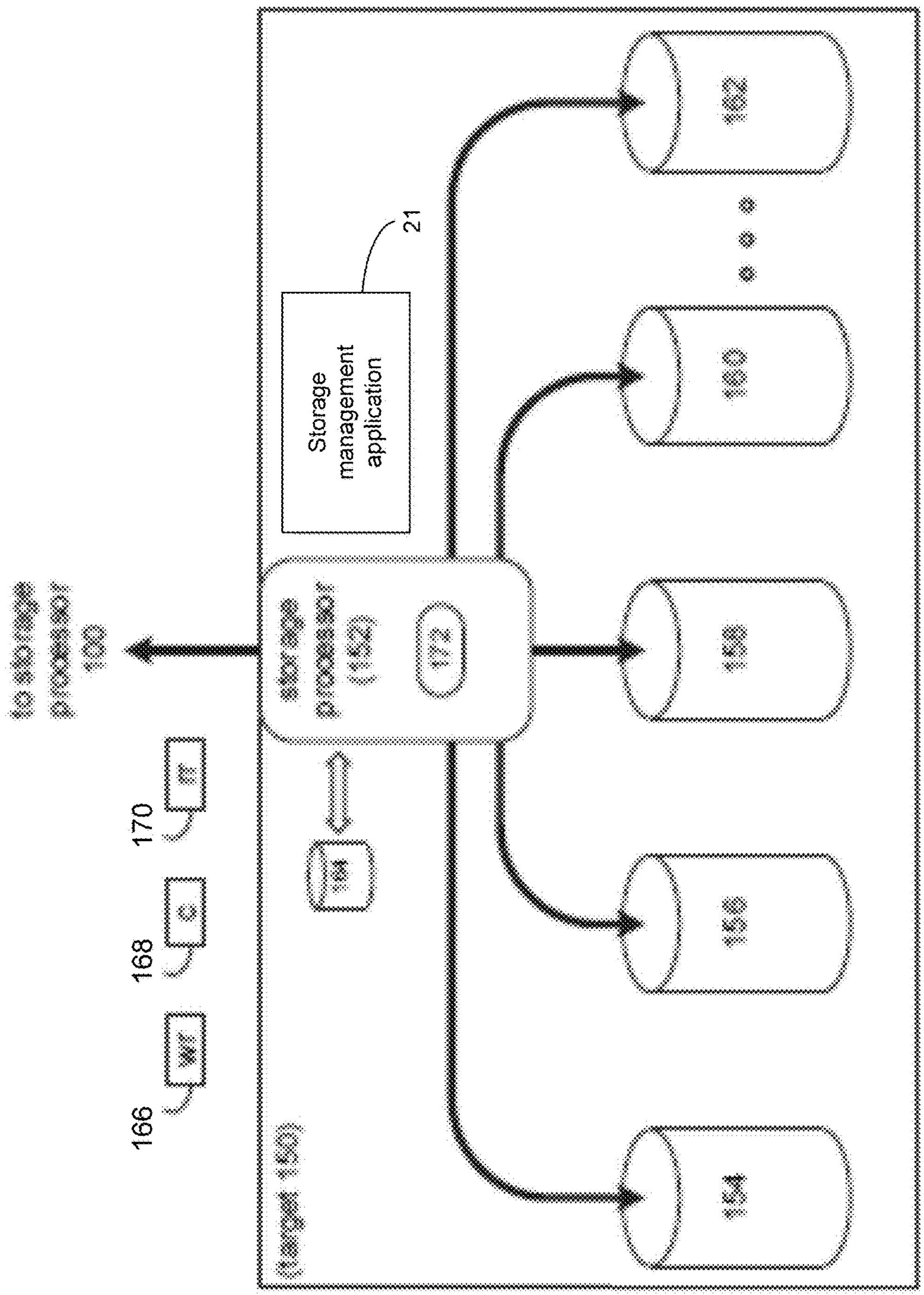

Storage management application
21
to storage processor 100
storage processor (152)
172
164
166
168
170
WT
C
FT
154
156
158
160
162
(target 150)

FIG. 3

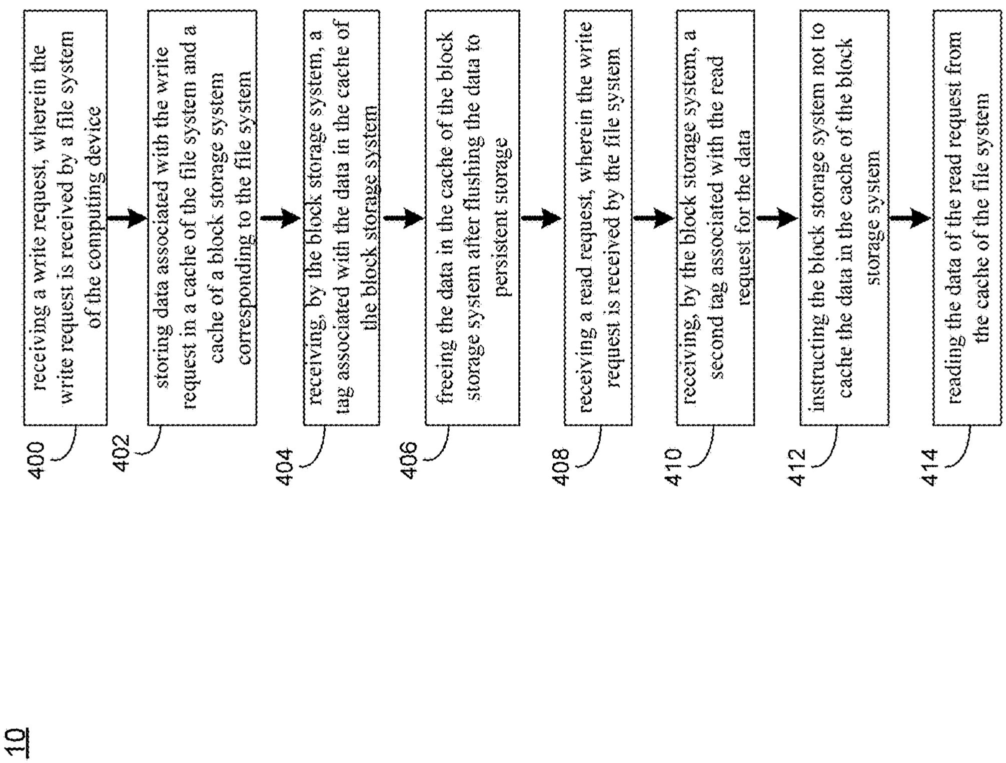

10
400
receiving a write request, wherein the write request is received by a file system of the computing device
402
storing data associated with the write request in a cache of the file system and a cache of a block storage system corresponding to the file system
404
receiving, by the block storage system, a tag associated with the data in the cache of the block storage system
406
freeing the data in the cache of the block storage system after flushing the data to persistent storage
408
receiving a read request, wherein the write request is received by the file system
410
receiving, by the block storage system, a second tag associated with the read request for the data
412
instructing the block storage system not to cache the data in the cache of the block storage system
414
reading the data of the read request from the cache of the file system

FIG. 4

500

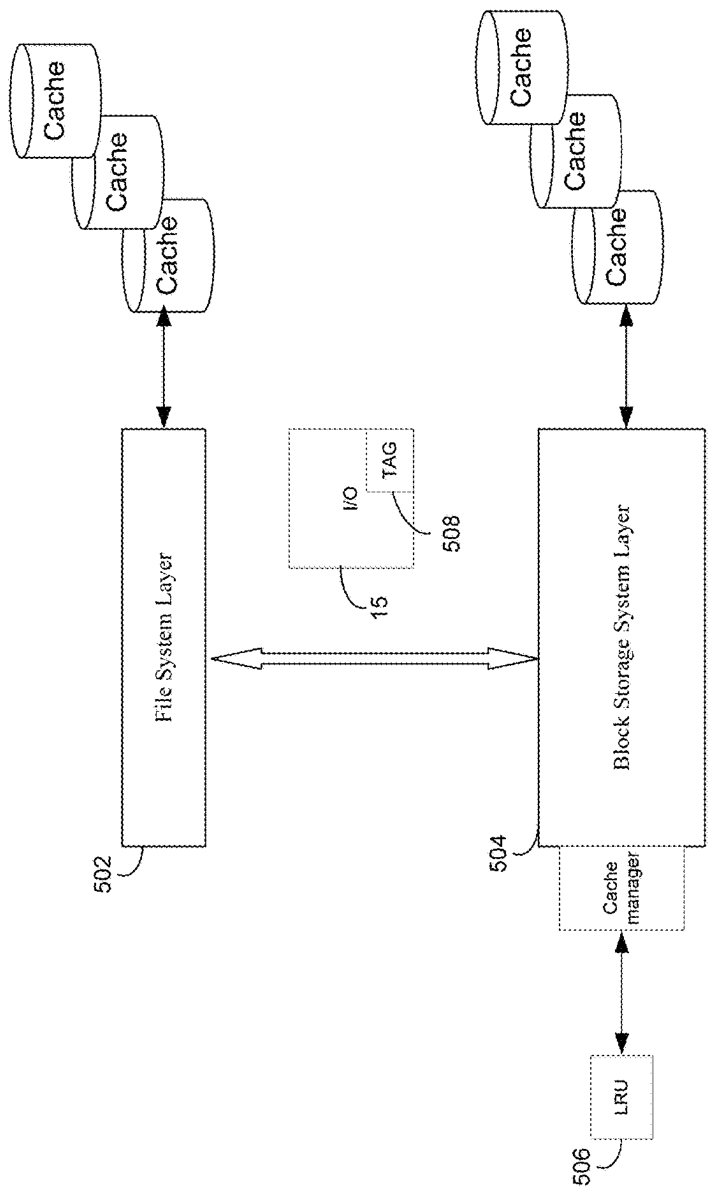

Cache
Cache
Cache
Cache
Cache
Cache
File System Layer
502
I/O
TAG
508
15
Block Storage System Layer
504
Cache
manager
LRU
506

FIG. 5

SYSTEMS AND METHODS FOR COOPERATIVE CACHING BETWEEN FILE SYSTEMS AND BLOCK STORAGE

BACKGROUND

File systems and underlying block storage have both read and write caches. Generally, if both the file system and the block storage cache the same page, it may lead to a lower overcall cache hit ratio.

BRIEF SUMMARY OF DISCLOSURE

In one example implementation, a method, performed by one or more computing devices, may include but is not limited to receiving, by a computing device, a write request, wherein the write request may be received by a file system of the computing device. Data associated with the write request may be stored in a cache of the file system and a cache of a block storage system corresponding to the file system. A tag associated with the data in the cache of the block storage system may be received by the block storage system. The data in the cache of the block storage system may be freed after flushing the data to persistent storage based upon, at least in part, the tag associated with the data in the cache of the block storage system.

One or more of the following example features may be included. A read request may be received, wherein the read request may be received by the file system. The block storage system may receive a second tag associated with the read request for the data. The block storage system may be instructed not to cache the data in the cache of the block storage system based upon, at least in part, the second tag associated with the read request for the data. The data of the read request may be read from the cache of the file system. Duplication of the data in the cache of the block storage system cache may be avoided based upon, at least in part, the tag associated with the data in the cache of the block storage system. The cache of the block storage system may be a clean cache.

In another example implementation, a computing system may include one or more processors and one or more memories configured to perform operations that may include but are not limited to receiving a write request, wherein the write request may be received by a file system of the computing device. Data associated with the write request may be stored in a cache of the file system and a cache of a block storage system corresponding to the file system. A tag associated with the data in the cache of the block storage system may be received by the block storage system. The data in the cache of the block storage system may be freed after flushing the data to persistent storage based upon, at least in part, the tag associated with the data in the cache of the block storage system.

One or more of the following example features may be included. A read request may be received, wherein the read request may be received by the file system. The block storage system may receive a second tag associated with the read request for the data. The block storage system may be instructed not to cache the data in the cache of the block storage system based upon, at least in part, the second tag associated with the read request for the data. The data of the read request may be read from the cache of the file system. Duplication of the data in the cache of the block storage system cache may be avoided based upon, at least in part, the tag associated with the data in the cache of the block storage system. The cache of the block storage system may be a clean cache.

In another example implementation, a computer program product may reside on a computer readable storage medium having a plurality of instructions stored thereon which, when executed across one or more processors, may cause at least a portion of the one or more processors to perform operations that may include but are not limited to receiving a write request, wherein the write request may be received by a file system of the computing device. Data associated with the write request may be stored in a cache of the file system and a cache of a block storage system corresponding to the file system. A tag associated with the data in the cache of the block storage system may be received by the block storage system. The data in the cache of the block storage system may be freed after flushing the data to persistent storage based upon, at least in part, the tag associated with the data in the cache of the block storage system.

One or more of the following example features may be included. A read request may be received, wherein the read request may be received by the file system. The block storage system may receive a second tag associated with the read request for the data. The block storage system may be instructed not to cache the data in the cache of the block storage system based upon, at least in part, the second tag associated with the read request for the data. The data of the read request may be read from the cache of the file system. Duplication of the data in the cache of the block storage system cache may be avoided based upon, at least in part, the tag associated with the data in the cache of the block storage system. The cache of the block storage system may be a clean cache.

The details of one or more example implementations are set forth in the accompanying drawings and the description below. Other possible example features and/or possible example advantages will become apparent from the description, the drawings, and the claims. Some implementations may not have those possible example features and/or possible example advantages, and such possible example features and/or possible example advantages may not necessarily be required of some implementations.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. **1** is an example diagrammatic view of a cache process coupled to an example distributed computing network according to one or more example implementations of the disclosure;

FIG. **2** is an example diagrammatic view of a storage system of FIG. **1** according to one or more example implementations of the disclosure;

FIG. **3** is an example diagrammatic view of a storage target of FIG. **1** according to one or more example implementations of the disclosure;

FIG. **4** is an example flowchart of a cache process according to one or more example implementations of the disclosure; and

FIG. **5** is an example diagrammatic view of a file system layer and a block storage system layer of a cache process according to one or more example implementations of the disclosure.

Like reference symbols in the various drawings indicate like elements.

DETAILED DESCRIPTION

System Overview:

In some implementations, the present disclosure may be embodied as a method, system, or computer program product. Accordingly, in some implementations, the present disclosure may take the form of an entirely hardware implementation, an entirely software implementation (including firmware, resident software, micro-code, etc.) or an implementation combining software and hardware aspects that may all generally be referred to herein as a "circuit," "module" or "system." Furthermore, in some implementations, the present disclosure may take the form of a computer program product on a computer-usable storage medium having computer-usable program code embodied in the medium.

In some implementations, any suitable computer usable or computer readable medium (or media) may be utilized. The computer readable medium may be a computer readable signal medium or a computer readable storage medium. The computer-usable, or computer-readable, storage medium (including a storage device associated with a computing device or client electronic device) may be, for example, but is not limited to, an electronic, magnetic, optical, electromagnetic, infrared, or semiconductor system, apparatus, device, or any suitable combination of the foregoing. More specific examples (a non-exhaustive list) of the computer-readable medium may include the following: an electrical connection having one or more wires, a portable computer diskette, a hard disk, a random access memory (RAM), a read-only memory (ROM), an erasable programmable read-only memory (EPROM or Flash memory), an optical fiber, a portable compact disc read-only memory (CD-ROM), an optical storage device, a digital versatile disk (DVD), a static random access memory (SRAM), a memory stick, a floppy disk, a mechanically encoded device such as punch-cards or raised structures in a groove having instructions recorded thereon, a media such as those supporting the internet or an intranet, or a magnetic storage device. Note that the computer-usable or computer-readable medium could even be a suitable medium upon which the program is stored, scanned, compiled, interpreted, or otherwise processed in a suitable manner, if necessary, and then stored in a computer memory. In the context of the present disclosure, a computer-usable or computer-readable, storage medium may be any tangible medium that can contain or store a program for use by or in connection with the instruction execution system, apparatus, or device.

In some implementations, a computer readable signal medium may include a propagated data signal with computer readable program code embodied therein, for example, in baseband or as part of a carrier wave. In some implementations, such a propagated signal may take any of a variety of forms, including, but not limited to, electromagnetic, optical, or any suitable combination thereof. In some implementations, the computer readable program code may be transmitted using any appropriate medium, including but not limited to the internet, wireline, optical fiber cable, RF, etc. In some implementations, a computer readable signal medium may be any computer readable medium that is not a computer readable storage medium and that can communicate, propagate, or transport a program for use by or in connection with an instruction execution system, apparatus, or device.

In some implementations, computer program code for carrying out operations of the present disclosure may be assembler instructions, instruction-set-architecture (ISA) instructions, machine instructions, machine dependent instructions, microcode, firmware instructions, state-setting data, or either source code or object code written in any combination of one or more programming languages, including an object oriented programming language such as Java®, Smalltalk, C++ or the like. Java® and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates. However, the computer program code for carrying out operations of the present disclosure may also be written in conventional procedural programming languages, such as the "C" programming language, PASCAL, or similar programming languages, as well as in scripting languages such as Javascript, PERL, or Python. The program code may execute entirely on the user's computer, partly on the user's computer, as a stand-alone software package, partly on the user's computer and partly on a remote computer or entirely on the remote computer or server. In the latter scenario, the remote computer may be connected to the user's computer through a local area network (LAN) or a wide area network (WAN), or the connection may be made to an external computer (for example, through the internet using an Internet Service Provider). In some implementations, electronic circuitry including, for example, programmable logic circuitry, field-programmable gate arrays (FPGAs) or other hardware accelerators, micro-controller units (MCUs), or programmable logic arrays (PLAs) may execute the computer readable program instructions/code by utilizing state information of the computer readable program instructions to personalize the electronic circuitry, in order to perform aspects of the present disclosure.

In some implementations, the flowchart and block diagrams in the figures illustrate the architecture, functionality, and operation of possible implementations of apparatus (systems), methods and computer program products according to various implementations of the present disclosure. Each block in the flowchart and/or block diagrams, and combinations of blocks in the flowchart and/or block diagrams, may represent a module, segment, or portion of code, which comprises one or more executable computer program instructions for implementing the specified logical function(s)/act(s). These computer program instructions may be provided to a processor of a general purpose computer, special purpose computer, or other programmable data processing apparatus to produce a machine, such that the computer program instructions, which may execute via the processor of the computer or other programmable data processing apparatus, create the ability to implement one or more of the functions/acts specified in the flowchart and/or block diagram block or blocks or combinations thereof. It should be noted that, in some implementations, the functions noted in the block(s) may occur out of the order noted in the figures (or combined or omitted). For example, two blocks shown in succession may, in fact, be executed substantially concurrently, or the blocks may sometimes be executed in the reverse order, depending upon the functionality involved.

In some implementations, these computer program instructions may also be stored in a computer-readable memory that can direct a computer or other programmable data processing apparatus to function in a particular manner, such that the instructions stored in the computer-readable memory produce an article of manufacture including instruction means which implement the function/act specified in the flowchart and/or block diagram block or blocks or combinations thereof.

In some implementations, the computer program instructions may also be loaded onto a computer or other programmable data processing apparatus to cause a series of operational steps to be performed (not necessarily in a particular order) on the computer or other programmable apparatus to produce a computer implemented process such that the instructions which execute on the computer or other programmable apparatus provide steps for implementing the functions/acts (not necessarily in a particular order) specified in the flowchart and/or block diagram block or blocks or combinations thereof.

Referring now to the example implementation of FIG. **1**, there is shown cache process **10** that may reside on and may be executed by a computer (e.g., computer **12**), which may be connected to a network (e.g., network **14**) (e.g., the internet or a local area network). Examples of computer **12** (and/or one or more of the client electronic devices noted below) may include, but are not limited to, a storage system (e.g., a Network Attached Storage (NAS) system, a Storage Area Network (SAN)), a personal computer(s), a laptop computer(s), mobile computing device(s), a server computer, a series of server computers, a mainframe computer(s), or a computing cloud(s). As is known in the art, a SAN may include one or more of the client electronic devices, including a Redundant Array of Inexpensive Disks/Redundant Array of Independent Disks (RAID) device and a NAS system. In some implementations, each of the aforementioned may be generally described as a computing device. In certain implementations, a computing device may be a physical or virtual device. In many implementations, a computing device may be any device capable of performing operations, such as a dedicated processor, a portion of a processor, a virtual processor, a portion of a virtual processor, portion of a virtual device, or a virtual device. In some implementations, a processor may be a physical processor or a virtual processor. In some implementations, a virtual processor may correspond to one or more parts of one or more physical processors. In some implementations, the instructions/logic may be distributed and executed across one or more processors, virtual or physical, to execute the instructions/logic. Computer **12** may execute an operating system, for example, but not limited to, Microsoft® Windows®; Mac® OS X®; Red Hat® Linux®, Windows® Mobile, Chrome OS, Blackberry OS, Fire OS, or a custom operating system. (Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States, other countries or both; Mac and OS X are registered trademarks of Apple Inc. in the United States, other countries or both; Red Hat is a registered trademark of Red Hat Corporation in the United States, other countries or both; and Linux is a registered trademark of Linus Torvalds in the United States, other countries or both).

In some implementations, as will be discussed below in greater detail, a cache process, such as cache process **10** of FIG. **1**, may receive, by a computing device, a write request, wherein the write request may be received by a file system of the computing device. Data associated with the write request may be stored in a cache of the file system and a cache of a block storage system corresponding to the file system. A tag associated with the data in the cache of the block storage system may be received by the block storage system. The data in the cache of the block storage system may be freed after flushing the data to persistent storage based upon, at least in part, the tag associated with the data in the cache of the block storage system.

In some implementations, the instruction sets and subroutines of cache process **10**, which may be stored on storage device, such as storage device **16**, coupled to computer **12**, may be executed by one or more processors and one or more memory architectures included within computer **12**. In some implementations, storage device **16** may include but is not limited to: a hard disk drive; all forms of flash memory storage devices; a tape drive; an optical drive; a RAID array (or other array); a random access memory (RAM); a read-only memory (ROM); or combination thereof. In some implementations, storage device **16** may be organized as an extent, an extent pool, a RAID extent (e.g., an example 4D+1P R5, where the RAID extent may include, e.g., five storage device extents that may be allocated from, e.g., five different storage devices), a mapped RAID (e.g., a collection of RAID extents), or combination thereof.

In some implementations, network **14** may be connected to one or more secondary networks (e.g., network **18**), examples of which may include but are not limited to: a local area network; a wide area network or other telecommunications network facility; or an intranet, for example. The phrase "telecommunications network facility," as used herein, may refer to a facility configured to transmit, and/or receive transmissions to/from one or more mobile client electronic devices (e.g., cellphones, etc.) as well as many others.

In some implementations, computer **12** may include a data store, such as a database (e.g., relational database, object-oriented database, triplestore database, etc.) and may be located within any suitable memory location, such as storage device **16** coupled to computer **12**. In some implementations, data, metadata, information, etc. described throughout the present disclosure may be stored in the data store. In some implementations, computer **12** may utilize any known database management system such as, but not limited to, DB2, in order to provide multi-user access to one or more databases, such as the above noted relational database. In some implementations, the data store may also be a custom database, such as, for example, a flat file database or an XML database. In some implementations, any other form(s) of a data storage structure and/or organization may also be used. In some implementations, cache process **10** may be a component of the data store, a standalone application that interfaces with the above noted data store and/or an applet/application that is accessed via client applications **22**, **24**, **26**, **28**. In some implementations, the above noted data store may be, in whole or in part, distributed in a cloud computing topology. In this way, computer **12** and storage device **16** may refer to multiple devices, which may also be distributed throughout the network.

In some implementations, computer **12** may execute a storage management application (e.g., storage management application **21**), examples of which may include, but are not limited to, e.g., a storage system application, a cloud computing application, a data synchronization application, a data migration application, a garbage collection application, or other application that allows for the implementation and/or management of data in a clustered (or non-clustered) environment (or the like). In some implementations, cache process **10** and/or storage management application **21** may be accessed via one or more of client applications **22**, **24**, **26**, **28**. In some implementations, cache process **10** may be a standalone application, or may be an applet/application/script/extension that may interact with and/or be executed within storage management application **21**, a component of storage management application **21**, and/or one or more of client applications **22**, **24**, **26**, **28**. In some implementations, storage management application **21** may be a standalone application, or may be an applet/application/script/extension that may interact with and/or be executed within cache process **10**, a component of cache process **10**, and/or one or more of client applications **22**, **24**, **26**, **28**. In some implementations, one or more of client applications **22**, **24**, **26**, **28** may be a standalone application, or may be an applet/application/script/extension that may interact with and/or be executed within and/or be a component of cache process **10** and/or storage management application **21**. Examples of client applications **22**, **24**, **26**, **28** may include, but are not limited to, e.g., a storage system application, a cloud computing application, a data synchronization application, a data migration application, a garbage collection application, or other application that allows for the implementation and/or management of data in a clustered (or non-clustered) environment (or the like), a standard and/or mobile web browser, an email application (e.g., an email client application), a textual and/or a graphical user interface, a customized web browser, a plugin, an Application Programming Interface (API), or a custom application. The instruction sets and subroutines of client applications **22**, **24**, **26**, **28**, which may be stored on storage devices **30**, **32**, **34**, **36**, coupled to client electronic devices **38**, **40**, **42**, **44**, may be executed by one or more processors and one or more memory architectures incorporated into client electronic devices **38**, **40**, **42**, **44**.

In some implementations, one or more of storage devices **30**, **32**, **34**, **36**, may include but are not limited to: hard disk drives; flash drives, tape drives; optical drives; RAID arrays; random access memories (RAM); and read-only memories (ROM). Examples of client electronic devices **38**, **40**, **42**, **44** (and/or computer **12**) may include, but are not limited to, a personal computer (e.g., client electronic device **38**), a laptop computer (e.g., client electronic device **40**), a smart/data-enabled, cellular phone (e.g., client electronic device **42**), a notebook computer (e.g., client electronic device **44**), a tablet, a server, a television, a smart television, a smart speaker, an Internet of Things (IoT) device, a media (e.g., video, photo, etc.) capturing device, and a dedicated network device. Client electronic devices **38**, **40**, **42**, **44** may each execute an operating system, examples of which may include but are not limited to, Android™, Apple® iOS®, Mac® OS X®; Red Hat® Linux®, Windows® Mobile, Chrome OS, Blackberry OS, Fire OS, or a custom operating system.

In some implementations, one or more of client applications **22**, **24**, **26**, **28** may be configured to effectuate some or all of the functionality of cache process **10** (and vice versa). Accordingly, in some implementations, cache process **10** may be a purely server-side application, a purely client-side application, or a hybrid server-side/client-side application that is cooperatively executed by one or more of client applications **22**, **24**, **26**, **28** and/or cache process **10**.

In some implementations, one or more of client applications **22**, **24**, **26**, **28** may be configured to effectuate some or all of the functionality of storage management application **21** (and vice versa). Accordingly, in some implementations, storage management application **21** may be a purely server-side application, a purely client-side application, or a hybrid server-side/client-side application that is cooperatively executed by one or more of client applications **22**, **24**, **26**, **28** and/or storage management application **21**. As one or more of client applications **22**, **24**, **26**, **28**, cache process **10**, and storage management application **21**, taken singly or in any combination, may effectuate some or all of the same functionality, any description of effectuating such functionality via one or more of client applications **22**, **24**, **26**, **28**, cache process **10**, storage management application **21**, or combination thereof, and any described interaction(s) between one or more of client applications **22**, **24**, **26**, **28**, cache process **10**, storage management application **21**, or combination thereof to effectuate such functionality, should be taken as an example only and not to limit the scope of the disclosure.

In some implementations, one or more of users **46**, **48**, **50**, **52** may access computer **12** and cache process **10** (e.g., using one or more of client electronic devices **38**, **40**, **42**, **44**) directly through network **14** or through secondary network **18**. Further, computer **12** may be connected to network **14** through secondary network **18**, as illustrated with phantom link line **54**. Cache process **10** may include one or more user interfaces, such as browsers and textual or graphical user interfaces, through which users **46**, **48**, **50**, **52** may access cache process **10**.

In some implementations, the various client electronic devices may be directly or indirectly coupled to network **14** (or network **18**). For example, client electronic device **38** is shown directly coupled to network **14** via a hardwired network connection. Further, client electronic device **44** is shown directly coupled to network **18** via a hardwired network connection. Client electronic device **40** is shown wirelessly coupled to network **14** via wireless communication channel **56** established between client electronic device **40** and wireless access point (i.e., WAP) **58**, which is shown directly coupled to network **14**. WAP 58 may be, for example, an IEEE 802.11a, 802.11b, 802.11g, 802.11n, 802.11ac, Wi-Fi®, RFID, and/or Bluetooth™ (including Bluetooth™ Low Energy) device that is capable of establishing wireless communication channel **56** between client electronic device **40** and WAP 58. Client electronic device **42** is shown wirelessly coupled to network **14** via wireless communication channel **60** established between client electronic device **42** and cellular network/bridge **62**, which is shown by example directly coupled to network **14**.

In some implementations, some or all of the IEEE 802.11x specifications may use Ethernet protocol and carrier sense multiple access with collision avoidance (i.e., CSMA/CA) for path sharing. The various 802.11x specifications may use phase-shift keying (i.e., PSK) modulation or complementary code keying (i.e., CCK) modulation, for example. Bluetooth™ (including Bluetooth™ Low Energy) is a telecommunications industry specification that allows, e.g., mobile phones, computers, smart phones, and other electronic devices to be interconnected using a short-range wireless connection. Other forms of interconnection (e.g., Near Field Communication (NFC)) may also be used.

In some implementations, various I/O requests (e.g., I/O request **15**) may be sent from, e.g., client applications **22**, **24**, **26**, **28** to, e.g., computer **12**. Examples of I/O request **15** may include but are not limited to, data write requests (e.g., a request that content be written to computer **12**) and data read requests (e.g., a request that content be read from computer **12**).

Data Storage System:

Referring also to the example implementation of FIGS. **2-3** (e.g., where computer **12** may be configured as a data storage system), computer **12** may include storage processor **100** and a plurality of storage targets (e.g., storage targets **102**, **104**, **106**, **108**, **110**). In some implementations, storage targets **102**, **104**, **106**, **108**, **110** may include any of the above-noted storage devices. In some implementations, storage targets **102**, **104**, **106**, **108**, **110** may be configured to provide various levels of performance and/or high availability. For example, storage targets **102**, **104**, **106**, **108**, **110** may be configured to form a non-fully-duplicative fault-tolerant data storage system (such as a non-fully-duplicative RAID data storage system), examples of which may include but are not limited to: RAID 3 arrays, RAID 4 arrays, RAID 5 arrays, and/or RAID 6 arrays. It will be appreciated that various other types of RAID arrays may be used without departing from the scope of the present disclosure.

While in this particular example, computer **12** is shown to include five storage targets (e.g., storage targets **102**, **104**, **106**, **108**, **110**), this is for example purposes only and is not intended limit the present disclosure. For instance, the actual number of storage targets may be increased or decreased depending upon, e.g., the level of redundancy/performance/capacity required.

Further, the storage targets (e.g., storage targets **102**, **104**, **106**, **108**, **110**) included with computer **12** may be configured to form a plurality of discrete storage arrays. For instance, and assuming for example purposes only that computer **12** includes, e.g., ten discrete storage targets, a first five targets (of the ten storage targets) may be configured to form a first RAID array and a second five targets (of the ten storage targets) may be configured to form a second RAID array.

In some implementations, one or more of storage targets **102**, **104**, **106**, **108**, **110** may be configured to store coded data (e.g., via storage management process **21**), wherein such coded data may allow for the regeneration of data lost/corrupted on one or more of storage targets **102**, **104**, **106**, **108**, **110**. Examples of such coded data may include but is not limited to parity data and Reed-Solomon data. Such coded data may be distributed across all of storage targets **102**, **104**, **106**, **108**, **110** or may be stored within a specific storage target.

Examples of storage targets **102**, **104**, **106**, **108**, **110** may include one or more data arrays, wherein a combination of storage targets **102**, **104**, **106**, **108**, **110** (and any processing/control systems associated with storage management application **21**) may form data array **112**.

The manner in which computer **12** is implemented may vary depending upon e.g., the level of redundancy/performance/capacity required. For example, computer **12** may be configured as a SAN (i.e., a Storage Area Network), in which storage processor **100** may be, e.g., a dedicated computing system and each of storage targets **102**, **104**, **106**, **108**, **110** may be a RAID device. An example of storage processor **100** may include but is not limited to a VPLEX™, VNX™, PowerStore™, or Unity™ system offered by Dell EMC™ of Hopkinton, MA.

In the example where computer **12** is configured as a SAN, the various components of computer **12** (e.g., storage processor **100**, and storage targets **102**, **104**, **106**, **108**, **110**) may be coupled using network infrastructure **114**, examples of which may include but are not limited to an Ethernet (e.g., Layer **2** or Layer **3**) network, a fiber channel network, an InfiniBand network, or any other circuit switched/packet switched network.

As discussed above, various I/O requests (e.g., I/O request **15**) may be generated. For example, these I/O requests may be sent from, e.g., client applications **22**, **24**, **26**, **28** to, e.g., computer **12**. Additionally/alternatively (e.g., when storage processor **100** is configured as an application server or otherwise), these I/O requests may be internally generated within storage processor **100** (e.g., via storage management process **21**). Examples of I/O request **15** may include but are not limited to data write request **116** (e.g., a request that content **118** be written to computer **12**) and data read request **120** (e.g., a request that content **118** be read from computer **12**).

In some implementations, during operation of storage processor **100**, content **118** to be written to computer **12** may be received and/or processed by storage processor **100** (e.g., via storage management process **21**). Additionally/alternatively (e.g., when storage processor **100** is configured as an application server or otherwise), content **118** to be written to computer **12** may be internally generated by storage processor **100** (e.g., via storage management process **21**).

As discussed above, the instruction sets and subroutines of storage management application **21**, which may be stored on storage device **16** included within computer **12**, may be executed by one or more processors and one or more memory architectures included with computer **12**. Accordingly, in addition to being executed on storage processor **100**, some or all of the instruction sets and subroutines of storage management application **21** (and/or cache process **10**) may be executed by one or more processors and one or more memory architectures included with data array **112**.

In some implementations, storage processor **100** may include front end cache memory system **122**. Examples of front end cache memory system **122** may include but are not limited to a volatile, solid-state, cache memory system (e.g., a dynamic RAM cache memory system), a non-volatile, solid-state, cache memory system (e.g., a flash-based, cache memory system), and/or any of the above-noted storage devices.

In some implementations, storage processor **100** may initially store content **118** within front end cache memory system **122**. Depending upon the manner in which front end cache memory system **122** is configured, storage processor **100** (e.g., via storage management process **21**) may immediately write content **118** to data array **112** (e.g., if front end cache memory system **122** is configured as a write-through cache) or may subsequently write content **118** to data array **112** (e.g., if front end cache memory system **122** is configured as a write-back cache).

In some implementations, one or more of storage targets **102**, **104**, **106**, **108**, **110** may include a backend cache memory system. Examples of the backend cache memory system may include but are not limited to a volatile, solid-state, cache memory system (e.g., a dynamic RAM cache memory system), a non-volatile, solid-state, cache memory system (e.g., a flash-based, cache memory system), and/or any of the above-noted storage devices.

Storage Targets:

As discussed above, one or more of storage targets **102**, **104**, **106**, **108**, **110** may be a RAID device. For instance, and referring also to FIG. **3**, there is shown example target **150**, wherein target **150** may be one example implementation of a RAID implementation of, e.g., storage target **102**, storage target **104**, storage target **106**, storage target **108**, and/or storage target **110**. An example of target **150** may include but is not limited to a VPLEX™, VNX™, PowerStore™, or Unity™ system offered by Dell EMC™ of Hopkinton, MA. Examples of storage devices **154**, **156**, **158**, **160**, **162** may include one or more electro-mechanical hard disk drives, one or more solid-state/flash devices, and/or any of the above-noted storage devices. It will be appreciated that while the term "disk" or "drive" may be used throughout, these may refer to and be used interchangeably with any types of appropriate storage devices as the context and functionality of the storage device permits.

In some implementations, target **150** may include storage processor **152** and a plurality of storage devices (e.g., storage devices **154**, **156**, **158**, **160**, **162**). Storage devices **154**, **156**, **158**, **160**, **162** may be configured to provide various levels of performance and/or high availability (e.g., via storage management process **21**). For example, one or more of storage devices **154**, **156**, **158**, **160**, **162** (or any of the above-noted storage devices) may be configured as a RAID 0 array, in which data is striped across storage devices. By striping data across a plurality of storage devices, improved performance may be realized. However, RAID 0 arrays may not provide a level of high availability. Accordingly, one or more of storage devices **154**, **156**, **158**, **160**, **162** (or any of the above-noted storage devices) may be configured as a RAID 1 array, in which data is mirrored between storage devices. By mirroring data between storage devices, a level of high availability may be achieved as multiple copies of the data may be stored within storage devices **154**, **156**, **158**, **160**, **162**.

While storage devices **154**, **156**, **158**, **160**, **162** are discussed above as being configured in a RAID 0 or RAID 1 array, this is for example purposes only and not intended to limit the present disclosure, as other configurations are possible. For example, storage devices **154**, **156**, **158**, **160**, **162** may be configured as a RAID 3, RAID 4, RAID 5 or RAID 6 array.

While in this particular example, target **150** is shown to include five storage devices (e.g., storage devices **154**, **156**, **158**, **160**, **162**), this is for example purposes only and not intended to limit the present disclosure. For instance, the actual number of storage devices may be increased or decreased depending upon, e.g., the level of redundancy/performance/capacity required.

In some implementations, one or more of storage devices **154**, **156**, **158**, **160**, **162** may be configured to store (e.g., via storage management process **21**) coded data, wherein such coded data may allow for the regeneration of data lost/corrupted on one or more of storage devices **154**, **156**, **158**, **160**, **162**. Examples of such coded data may include but are not limited to parity data and Reed-Solomon data. Such coded data may be distributed across all of storage devices **154**, **156**, **158**, **160**, **162** or may be stored within a specific storage device.

The manner in which target **150** is implemented may vary depending upon e.g., the level of redundancy/performance/capacity required. For example, target **150** may be a RAID device in which storage processor **152** is a RAID controller card and storage devices **154**, **156**, **158**, **160**, **162** are individual "hot-swappable" hard disk drives. Another example of target **150** may be a RAID system, examples of which may include but are not limited to an NAS (i.e., Network Attached Storage) device or a SAN (i.e., Storage Area Network).

In some implementations, storage target **150** may execute all or a portion of storage management application **21**. The instruction sets and subroutines of storage management application **21**, which may be stored on a storage device (e.g., storage device **164**) coupled to storage processor **152**, may be executed by one or more processors and one or more memory architectures included with storage processor **152**. Storage device **164** may include but is not limited to any of the above-noted storage devices.

As discussed above, computer **12** may be configured as a SAN, wherein storage processor **100** may be a dedicated computing system and each of storage targets **102**, **104**, **106**, **108**, **110** may be a RAID device. Accordingly, when storage processor **100** processes data requests **116**, **120**, storage processor **100** (e.g., via storage management process **21**) may provide the appropriate requests/content (e.g., write request **166**, content **168** and read request **170**) to, e.g., storage target **150** (which is representative of storage targets **102**, **104**, **106**, **108** and/or **110**).

In some implementations, during operation of storage processor **152**, content **168** to be written to target **150** may be processed by storage processor **152** (e.g., via storage management process **21**). Storage processor **152** may include cache memory system **172**. Examples of cache memory system **172** may include but are not limited to a volatile, solid-state, cache memory system (e.g., a dynamic RAM cache memory system) and/or a non-volatile, solid-state, cache memory system (e.g., a flash-based, cache memory system). During operation of storage processor **152**, content **168** to be written to target **150** may be received by storage processor **152** (e.g., via storage management process **21**) and initially stored (e.g., via storage management process **21**) within front end cache memory system **172**.

File systems and underlying block storage have both read and write caches. Generally, if both the file system and the block storage cache the same page, it may lead to a lower overcall cache hit ratio. Therefore, as will be discussed in greater detail below, the present disclosure may enable a cooperative caching model to allow caching of a specific page either in the file system cache or in the block storage cache. Computing devices (e.g., servers) running a file system on top of block storage may have a large memory, and hence, a larger cache. The cooperative caching model of the present disclosure may help optimize the caching by avoiding the duplication.

The Cache Process:

As discussed above and referring also at least to the example implementations of FIGS. **4**-**5**, cache process **10** may receive **400**, by a computing device, a write request, wherein the write request may be received by a file system of the computing device. Cache process **10** may store **402** data associated with the write request in a cache of the file system and a cache of a block storage system corresponding to the file system. Cache process **10** may receive **404** a tag associated with the data in the cache of the block storage system. Cache process **10** may free **406** the data in the cache of the block storage system after flushing the data to persistent storage based upon, at least in part, the tag associated with the data in the cache of the block storage system.

In some implementations, cache process **10** may receive **400**, by a computing device, a write request, wherein the write request may be received by a file system of the computing device, and in some implementations, cache process **10** may store **402** data associated with the write request in a cache of the file system and a cache of a block storage system corresponding to the file system. For instance, in some implementations, assume for example purposes only that a file system (e.g., SDNAS or other file system) receives **400** a write request. Referring at least to the example implementation of FIG. **5**, a file system layer (e.g., file system layer **502**) and a block storage system layer (e.g., block system layer **504**) that may be used with cache process **10** is shown. Typically, and in the example, file systems and the underlying block storage have both read and write caches. Generally, the file system and the block storage system (e.g., PowerStore or other object oriented storage system) will both store **402** the data (e.g., dirty pages) in their respective caches. As a non-limiting example, in PowerStore, the flushed dirty SDNAS pages are retained in the PowerStore clean cache and least recently used (LRU) list (e.g., LRU list **506**), although other types of caches may also be used without departing from the scope of the present disclosure.

In some implementations, cache process **10** may receive **404** (e.g., via the block storage system) a tag associated with the data in the cache of the block storage system, and in some implementations, cache process **10** may free **406** the data in the cache of the block storage system after flushing the data to persistent storage based upon, at least in part, the tag associated with the data in the cache of the block storage system. For instance, and continuing with the above example, cache process **10** may pass a new metadata (MD) tag (e.g., tag **508**) with the data (e.g., IO **15**) from the file system to be received **404** by the block storage system data path to indicate that after flushing the (dirty) pages associated with the write request in the block storage system cache to persistent storage, the (now clean) pages can be discarded (freed) **406** from the block storage system cache. Typically, in normal operation, the pages are freed based upon, for example, the page moving from the tail of a LRU list when initially cached until it reaches the head (oldest) of the LRU list, where it would then be marked as being freed for the next page to be stored in the cache. However, by using the new MD tag being sent with the data, the block storage system knows (e.g., via a cache manager of cache process **10**) that it can free the pages from the cache as soon as the pages have been flushed for use right away, without having to wait per current LRU cache operations (e.g., in the LRU list) for the page to move from the tail of the list to the head of the list.

In some implementations, cache process **10** may instruct the block storage system not to compress the data in the cache of the block storage system cache based upon, at least in part, the tag associated with the data in the cache of the block storage system, and duplication of the data in the cache of the block storage system cache may be avoided based upon, at least in part, the tag associated with the data in the cache of the block storage system. For instance, not only does the present disclosure permit the block storage system cache to be used for other purposes, but without the need to store the same page twice (e.g., once in the file system cache and again in the block storage system cache) the overall system may be optimized by avoiding the duplication. Another added benefit is that since the data in the block storage system cache is going to be freed shortly after being flushed, there is no need to compress the data in the block storage system cache, which is what typically happens. This will save additional resources. Moreover, data may already be compressed on disk. The other way to store compressed is to just copy the compressed data read from disk instead of re-compressing.

Notably, future reads of the above-noted page will get the cache hits in the file system. For instance, in some implementations, cache process **10** may receive **408** a read request, wherein the read request may be received by the file system. Continuing with the above example, assume that a read request is received **408** for the previously cached data. In the example, similar to the write request discussed above, the file system and the block storage system will typically both store **402** the requested data in their respective caches. However, in some implementations, cache process **10** may (e.g., via the block storage system) receive **410** a second tag associated with the read request for the data, and in some implementations, cache process **10** may instruct **412** the block storage system not to cache the data in the cache of the block storage system based upon, at least in part, the second tag associated with the read request for the data. For example, when the file system executes a read, the read data (whether it is NAS metadata or NAS user data—or similar data from a different file system) does not have to be cached in both the file system cache and the block storage system cache. Instead, cache process **10** (e.g., via the file system) may pass a tag along with the block read requests to let block cache layer of the block storage system know not to cache the pages in the block layer. As a result, in some implementations, cache process **10** may read **414** the data of the read request from the cache of the file system, as all cache hits may be from the file system buffer cache.

The terminology used herein is for the purpose of describing particular implementations only and is not intended to be limiting of the disclosure. As used herein, the singular forms "a", "an" and "the" are intended to include the plural forms as well, unless the context clearly indicates otherwise. As used herein, the language "at least one of A, B, and C" (and the like) should be interpreted as covering only A, only B, only C, or any combination of the three, unless the context clearly indicates otherwise. It will be further understood that the terms "comprises" and/or "comprising," when used in this specification, specify the presence of stated features, integers, steps (not necessarily in a particular order), operations, elements, and/or components, but do not preclude the presence or addition of one or more other features, integers, steps (not necessarily in a particular order), operations, elements, components, and/or groups thereof.

The corresponding structures, materials, acts, and equivalents (e.g., of all means or step plus function elements) that may be in the claims below are intended to include any structure, material, or act for performing the function in combination with other claimed elements as specifically claimed. The description of the present disclosure has been presented for purposes of illustration and description, but is not intended to be exhaustive or limited to the disclosure in the form disclosed. Many modifications, variations, substitutions, and any combinations thereof will be apparent to those of ordinary skill in the art without departing from the scope and spirit of the disclosure. The implementation(s) were chosen and described in order to explain the principles of the disclosure and the practical application, and to enable others of ordinary skill in the art to understand the disclosure for various implementation(s) with various modifications and/or any combinations of implementation(s) as are suited to the particular use contemplated.

Having thus described the disclosure of the present application in detail and by reference to implementation(s) thereof, it will be apparent that modifications, variations, and any combinations of implementation(s) (including any modifications, variations, substitutions, and combinations thereof) are possible without departing from the scope of the disclosure defined in the appended claims.

What is claimed is:

1. A computer-implemented method comprising:

receiving, by a computing device, a write request, wherein the write request is received by a file system of the computing device;

storing data associated with the write request in a cache of the file system and a cache of a block storage system corresponding to the file system;

receiving, by the block storage system, a first tag associated with the data in the cache of the block storage system,

wherein compression of the data in the cache of the block storage system cache is avoided based upon, at least in part, the first tag associated with the data in the cache of the block storage system, and

wherein duplication of the data in the cache of the block storage system cache is avoided based upon, at least in part, the first tag associated with the data in the cache of the block storage system; and

freeing the data in the cache of the block storage system after flushing the data to persistent storage based upon, at least in part, the first tag associated with the data in the cache of the block storage system.

2. The computer-implemented method of claim **1** further comprising receiving a read request, wherein the read request is received by the file system.

3. The computer-implemented method of claim **2** further comprising receiving, by the block storage system, a second tag associated with the read request.

4. The computer-implemented method of claim **3** further comprising instructing the block storage system not to cache the data associated with the read request in the cache of the block storage system based upon, at least in part, the second tag associated with the read request.

5. The computer-implemented method of claim **4** further comprising reading the data associated with the read request from the cache of the file system.

6. The computer-implemented method of claim **1** wherein the cache of the block storage system is a clean cache.

7. A computer program product residing on a non-transitory computer readable storage medium having a plurality of instructions stored thereon which, when executed across one or more processors, causes at least a portion of the one or more processors to perform operations comprising:

receiving a write request, wherein the write request is received by a file system of the computing device;

storing data associated with the write request in a cache of the file system and a cache of a block storage system corresponding to the file system;

receiving, by the block storage system, a first tag associated with the data in the cache of the block storage system,

wherein compression of the data in the cache of the block storage system cache is avoided based upon, at least in part, the first tag associated with the data in the cache of the block storage system, and

wherein duplication of the data in the cache of the block storage system cache is avoided based upon, at least in part, the first tag associated with the data in the cache of the block storage system; and

freeing the data in the cache of the block storage system after flushing the data to persistent storage based upon, at least in part, the first tag associated with the data in the cache of the block storage system.

8. The computer program product of claim **7** wherein the operations further comprise receiving a read request, wherein the read request is received by the file system.

9. The computer program product of claim **8** wherein the operations further comprise receiving, by the block storage system, a second tag associated with the read request.

10. The computer program product of claim **9** wherein the operations further comprise instructing the block storage system not to cache the data associated with the read request in the cache of the block storage system based upon, at least in part, the second tag associated with the read request.

11. The computer program product of claim **10** wherein the operations further comprise reading the data associated with the read request from the cache of the file system.

12. The computer program product of claim **7** wherein the cache of the block storage system is a clean cache.

13. A computing system including one or more processors and one or more memories configured to perform operations comprising:

receiving a write request, wherein the write request is received by a file system of the computing device;

storing data associated with the write request in a cache of the file system and a cache of a block storage system corresponding to the file system;

receiving, by the block storage system, a first tag associated with the data in the cache of the block storage system,

wherein compression of the data in the cache of the block storage system cache is avoided based upon, at least in part, the first tag associated with the data in the cache of the block storage system, and

wherein duplication of the data in the cache of the block storage system cache is avoided based upon, at least in part, the first tag associated with the data in the cache of the block storage system; and

freeing the data in the cache of the block storage system after flushing the data to persistent storage based upon, at least in part, the first tag associated with the data in the cache of the block storage system.

14. The computing system of claim **13** wherein the operations further comprise receiving a read request, wherein the read request is received by the file system.

15. The computing system of claim **14** wherein the operations further comprise receiving, by the block storage system, a second tag associated with the read request.

16. The computing system of claim **15** wherein the operations further comprise instructing the block storage system not to cache the data associated with the read request in the cache of the block storage system based upon, at least in part, the second tag associated with the read request.

17. The computing system of claim **16** wherein the operations further comprise reading the data associated with the read request from the cache of the file system.

* * * * *